BIG DATA LANDSCAPE:

NoSQL Systems

by Asst. Prof. Kulsawasd Jitkajornwanich, Ph.D.
kulsawasd.ji@kmitl.ac.th

[Adapted from slides and/or other materials from: Bengfort & Kim “Data Analytics with Hadoop”, White “Hadoop: The Definitive Guide”, Schutt & O’Neil “Doing Data Science”, Elmasri & Navathe “Fundamentals of Database Systems”, Cielen et al. “Introducing Data Science”, and Provost & Fawcett “Data Science for Business”]

Kulsawasd Jitkajornwanich, Ph.D.

Assistant Professor of Computer Science

- Received Master's and PhD degrees in Computer Science from The University of Texas at Arlington, TX, USA

  Dissertation: "Analysis and Modeling Techniques for Geo-Spatial and Spatio-Temporal Datasets"

- Received Bachelor's degree (Hons.) in Computer Science from Chulalongkorn University, Thailand
- Conducting research/projects and academic services in the areas of database, NOSQL, big data analytics, social sciences and custom location-based applications
- Work Experiences:

Learning Objectives:

- 1) Understand concepts of nosql systems as well as other related terminology
- 2) To be able to communicate nosql concepts with the team effectively

Course Outline (part 2)

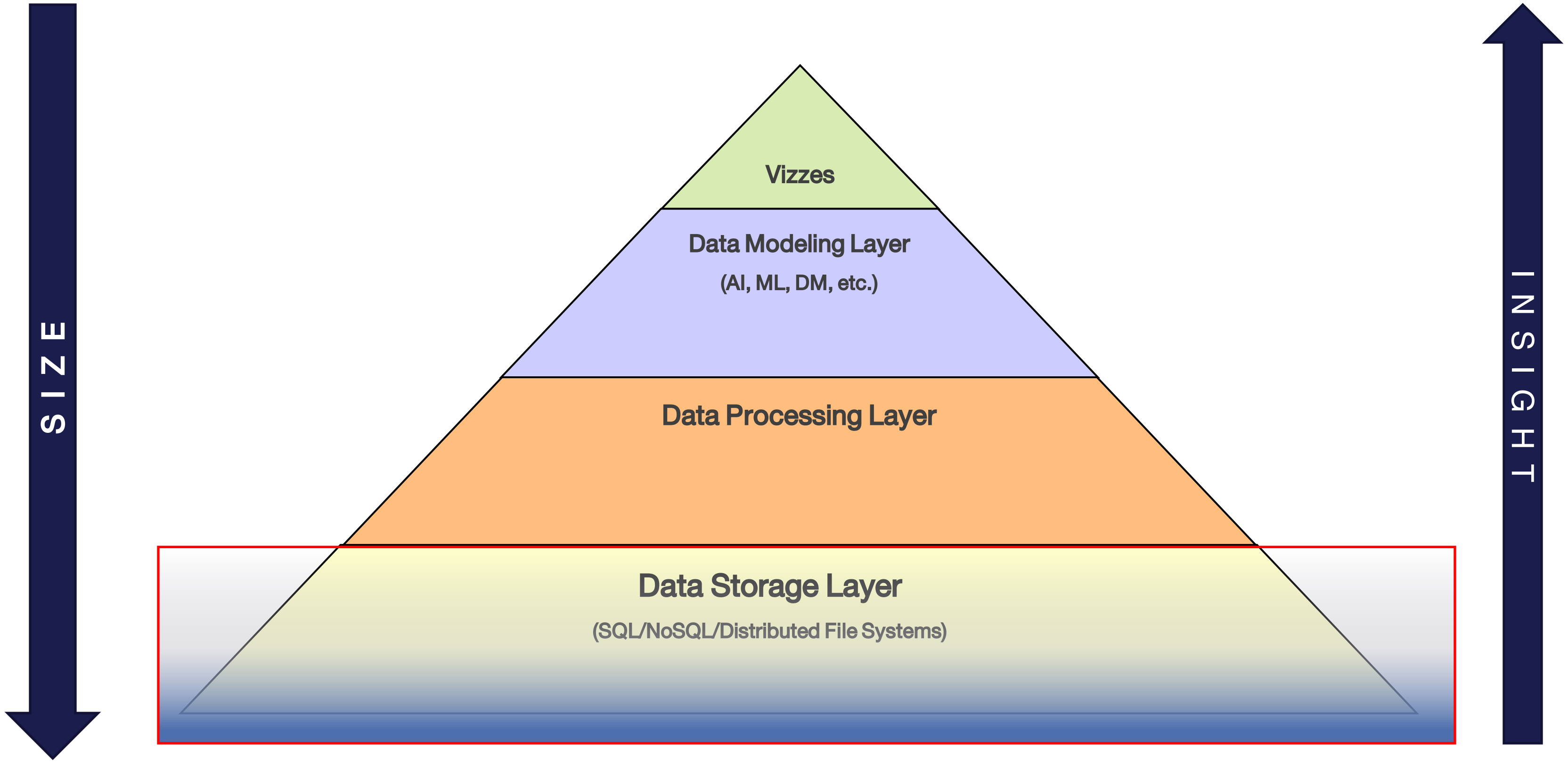
NoSQL System Fundamentals

- What is NoSQL? And NoSQL Background
- CAP Theorem
- Types of NoSQL systems
    - Some examples for each type of NoSQL

MongoDB: Document-Based NoSQL

- MongoDB Concepts & Data model
- Comparison to RDBMS
- CRUD operations
- Replication & Sharding
- Hands-on/Lab Exercises

Pyramid of Big Data Architecture Layers



Background

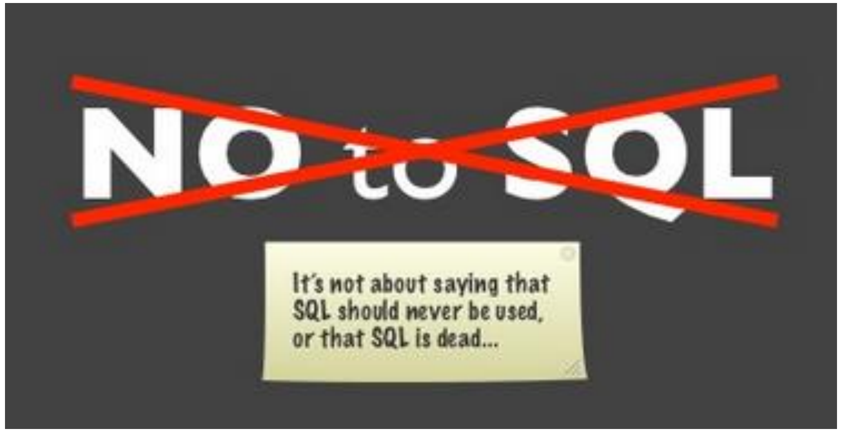
- Relational databases → **mainstay** of business
- Web-based applications caused spikes
    - explosion of social media sites (Facebook, Twitter) with large data needs
    - rise of cloud-based solutions such as Amazon S3 (simple storage solution, supporting 'Data Lakes')
- Hooking RDBMS to web-based application becomes trouble

Issues with *scaling up*

- Best way to provide ACID and rich query model is to have the dataset on a single machine
- Limits to ***scaling up*** (or ***vertical scaling***: make a "single" machine more powerful) → dataset is just too big!
- ***Scaling out*** (or ***horizontal scaling***: adding more smaller/cheaper servers) is a better choice
- Different approaches for horizontal scaling (multi-node database):
  - Distributed/parallel RDBMS
  - **NOSQL** (Non-RDBMS)

What is NOSQL?

- The Name:
  - Generally interpreted as "**N**ot **O**nly **SQL**"
  - The term NOSQL was introduced by *Carl Strozzi* in 1998 to name his file-based database
  - It was again re-introduced by *Eric Evans* when an event was organized to discuss open source distributed databases
  - Eric states that "*… but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for. …*"



What is NOSQL?

- Key features (advantages):
  - non-relational
  - don't require schema
  - data are replicated to multiple nodes (so, identical & fault-tolerant) and can be partitioned:
    - down nodes easily replaced
    - no single point of failure
  - horizontal scalable
  - cheap, easy to implement (open-source)
  - massive write performance
  - fast key-value access

What is NOSQL?

- Disadvantages:
  - Don't fully support relational features
    - no join, group by, order by operations (EXCEPT within 'partitions'--table)
    - no referential integrity constraints across partitions
  - No declarative query language (e.g., SQL) → more programming
  - Relaxed ACID (see CAP theorem) → fewer guarantees
  - No easy integration with other applications that support SQL

What is NOSQL?

RDBMS	NoSQL
– User-friendly/mature features (join, group-by)	– New tech./technical background required
– Strong consistency *(ACID)*	– Relaxed consistency *(BASE)*
– Can be slow (if > PB)	– High availability
– Structured (relational) data	– Dynamic/complex data structure
– Vertical scaling (“scaling up”)	– Horizontal scaling (“scaling out”)

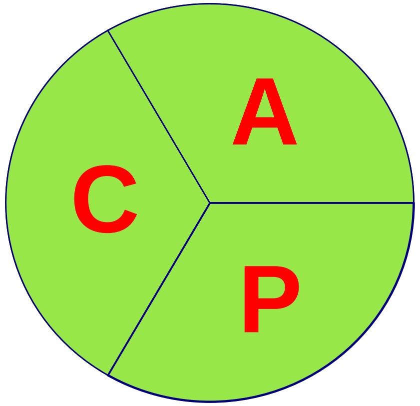
Who is using them?

3 major papers for NOSQL

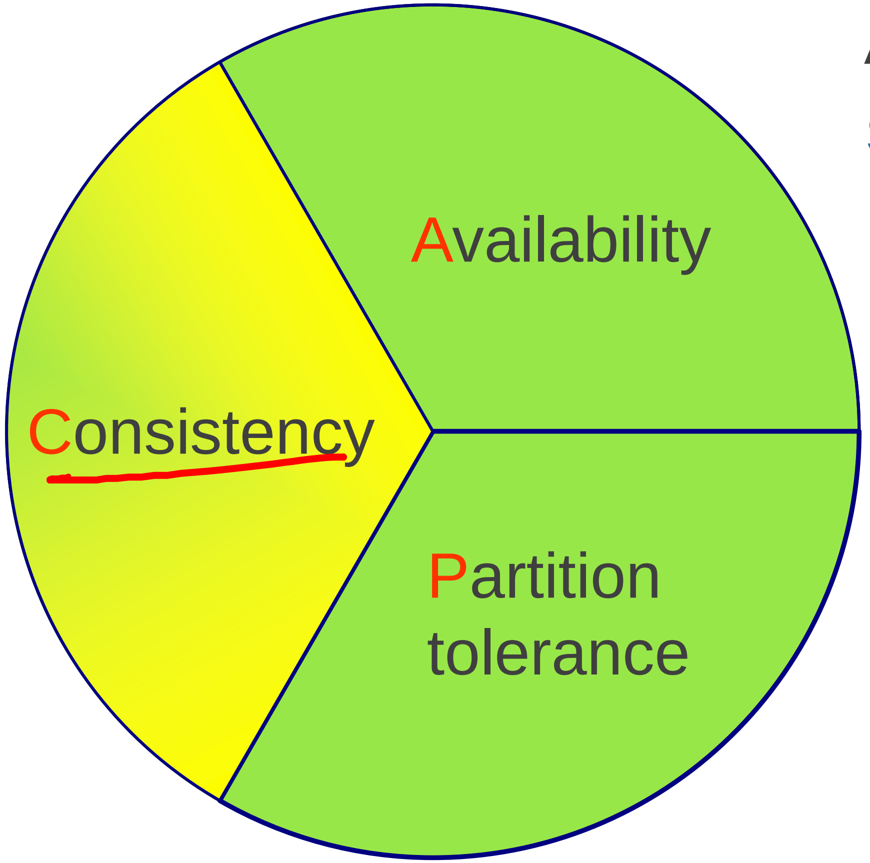
- Three major papers were the "seeds" of the NOSQL movement:
  - BigTable (Google)
  - DynamoDB (Amazon)
    - Ring partition and replication
    - Gossip protocol (discovery and error detection)
    - Distributed key-value data stores
    - Eventual consistency
  - **CAP Theorem**

CAP Theorem

- Suppose three properties of a distributed system (w/ data replication)
    - **Consistency:**
        - always return the most up-to-date values
    - **Availability:**
        - always respond (either failed or succeeded)
    - **Partition tolerance:**
        - can still operate even if there is a network fault resulting in two or more partitions (allowing some messages to get lost)

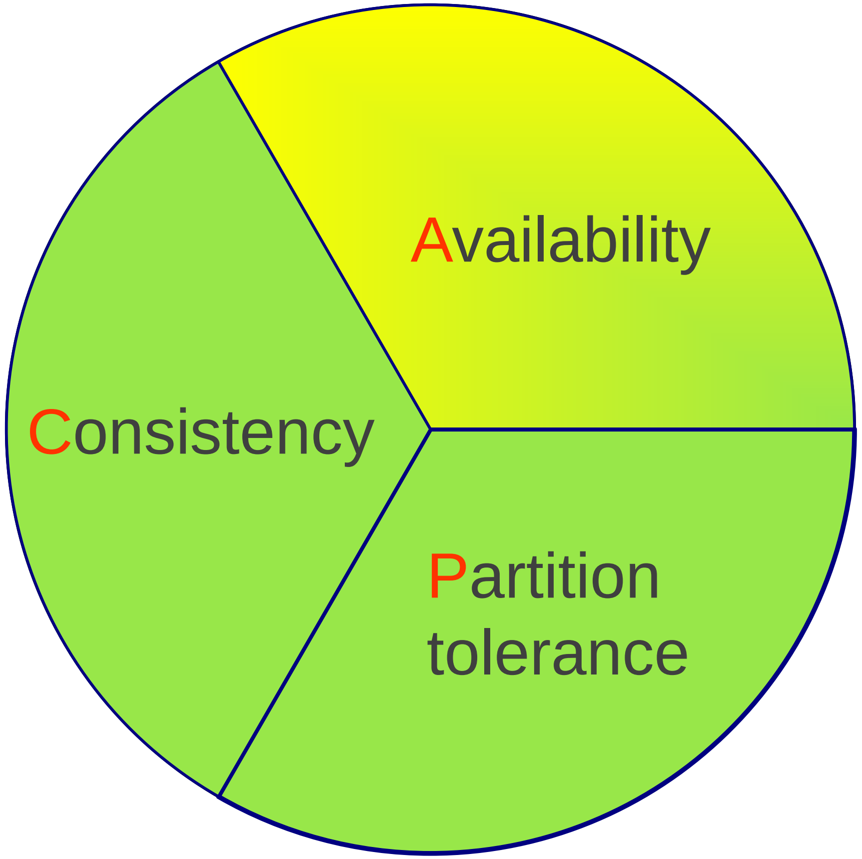


CAP Theorem



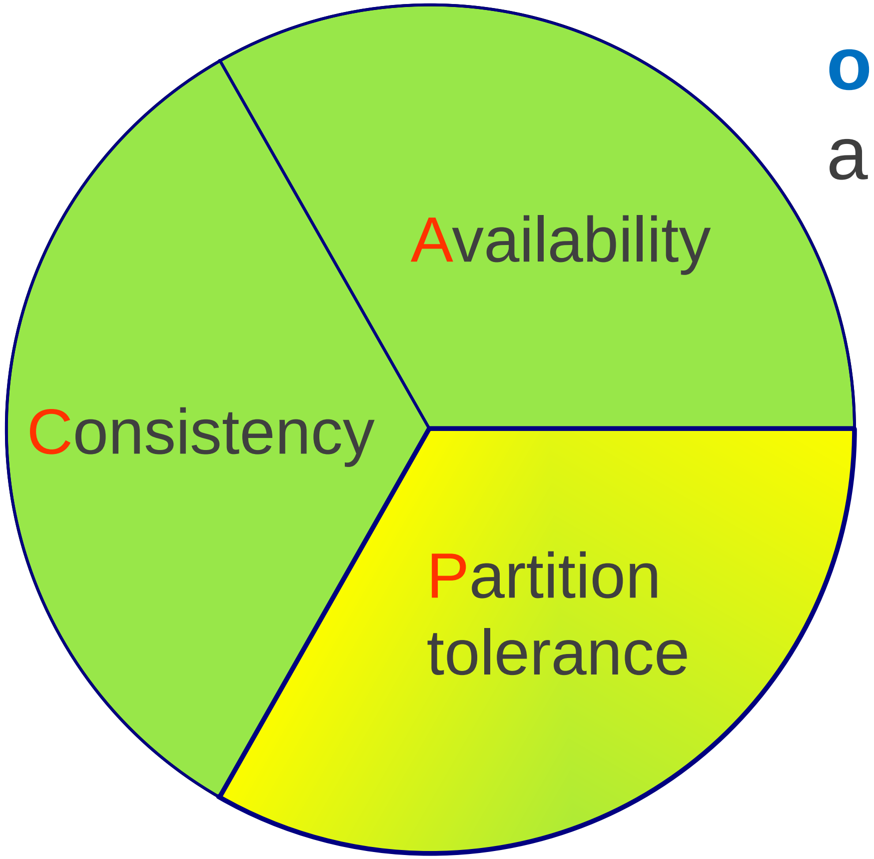
All client always have the **same view of the data**

CAP Theorem



Each client **always** can read and write.

CAP Theorem



A system can **continue to operate** in the presence of a network partitions

CAP Theorem

- **A consistency model determines rules for visibility and apparent order of updates**
- Example:
  - Row X is replicated on nodes M and N
  - Client A writes row X to node N
  - Some period of time t elapses
  - Client B reads row X from node M
  - **Does client B see the write from client A?**
  - Consistency is a continuum with tradeoffs
  - **For NOSQL, the answer would be: "maybe"**
  - **CAP theorem states: "*strong consistency can't be achieved at the same time as availability and partition-tolerance*"**

NOSQL categories

1. **Key-value**
   - Example: DynamoDB, Voldermort, Scalaris
2. **Document-based**
   - Example: MongoDB, CouchDB
3. **Column-based**
   - Example: BigTable, Cassandra, Hbased
4. **Graph-based**
   - Example: Neo4J, InfoGrid

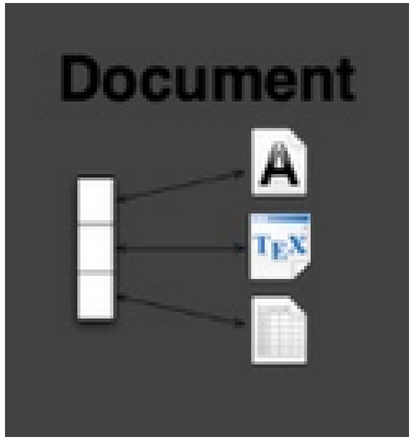
- "No-schema" is a common characteristic of most NOSQL storage systems
- Provide "flexible" data types (eg, JSON, XML, …)
- Rigid constraints on index creation
  - Mostly ONE index and SINGLE attribute (not composite) are allowed

NOSQL categories

1. **Key-value**
   - Example: DynamoDB, Voldermort, Scalaris
2. **Document-based**
   - Example: MongoDB, CouchDB
3. **Column-based**
   - Example: BigTable, Cassandra, Hbased
4. **Graph-based**
   - Example: Neo4J, InfoGrid

- "No-schema" is a common characteristic of most NOSQL storage systems
- Provide "flexible" data types (eg, JSON, XML, ...)
- Rigid constraints on index creation
  - Mostly ONE index and SINGLE attribute (not composite) are allowed

Document-based



- Can model more complex objects
- Inspired by Lotus Notes
- Data model: collection of documents
- **Indexing:** automatically indexed on ONE selected unique attribute (hash OR range (partition))
- Document: JSON (**J**ava**S**cript **O**bject **N**otation) is a data model supporting objects, records, structs, lists, array, maps, dates, Boolean (with **nesting**), XML, other more complex semi-structures.
- Comparison to key-value NOSQL:
  - Slower but more flexible since: 1) condition on an un-indexed attribute can be specified in the query, 2) normalization concept is allowed, 3) only ONE attribute can be indexed, and 4) more complex objects can be modelled.

Document-based

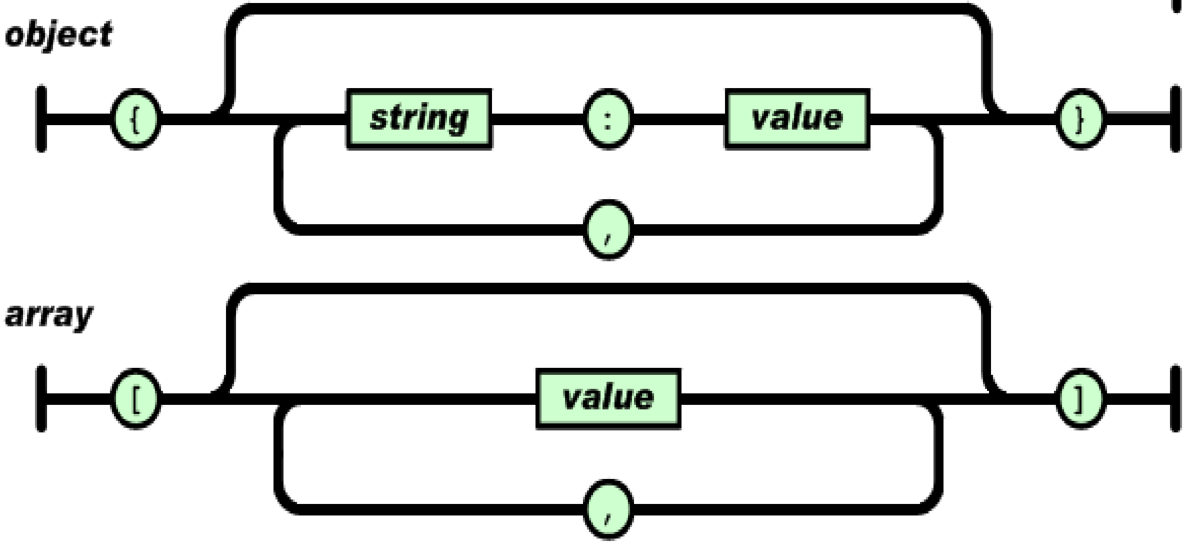
- Example: (MongoDB) document
  - {Name:"Jaroslav",

    Address:"Malostranske nám. 25, 118 00 Praha 1”,

    Grandchildren: {Claire: "7", Barbara: "6", "Magda: "3", "Kirsten: "1", "Otis: "3", Richard: "1“}

    Phones: [“123-456-7890”, “234-567-8963”]

    }



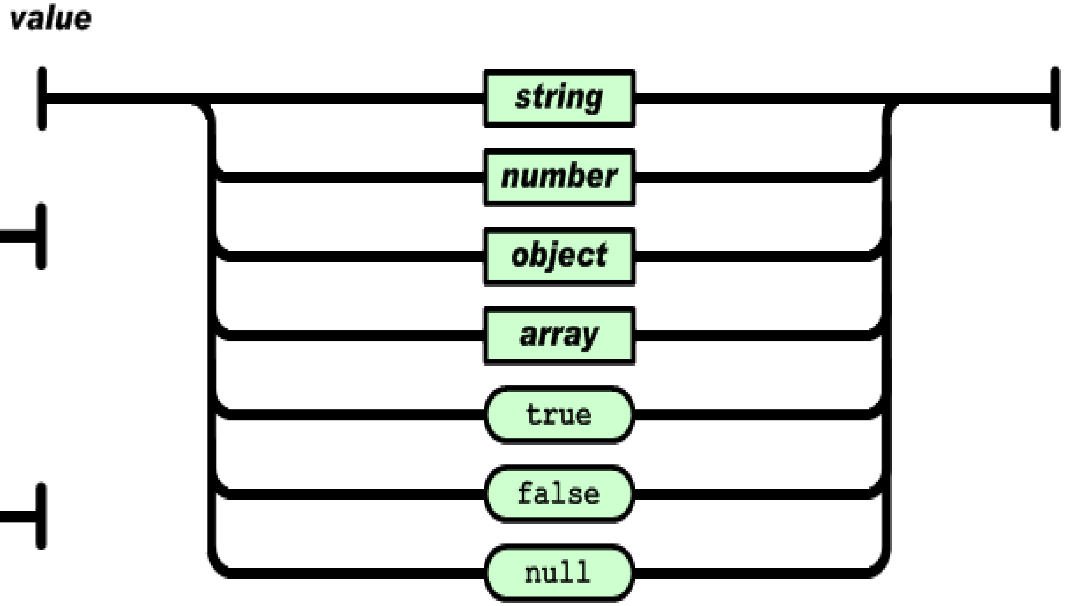


Figure 3.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Continued next page...

Document–based

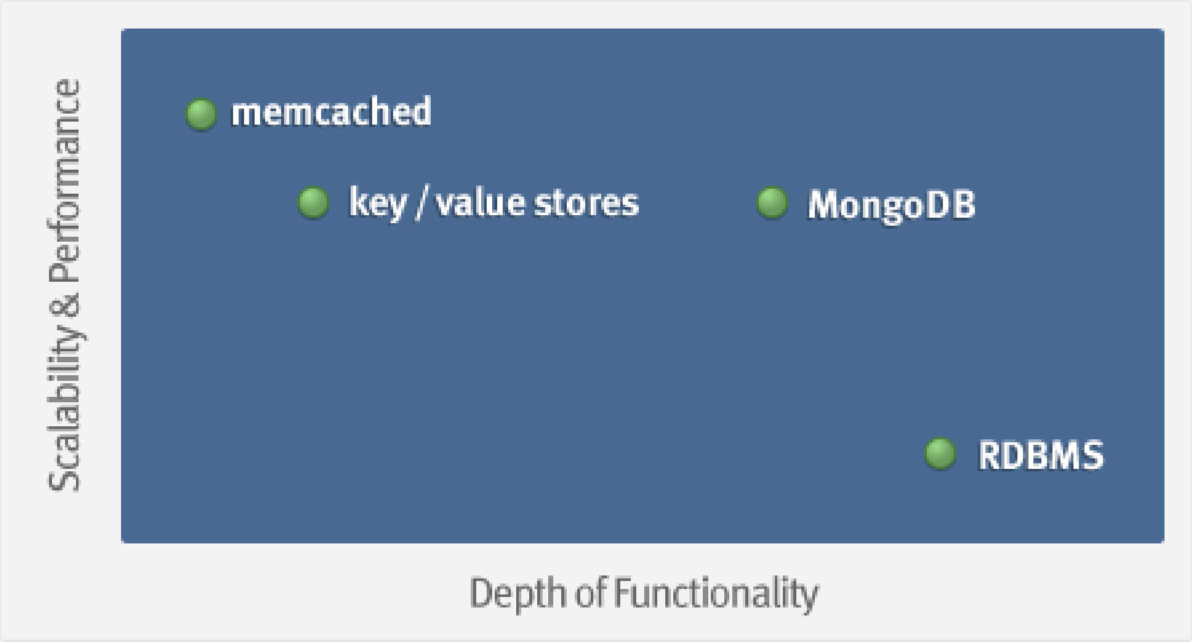
Name	Producer	Data model	Querying
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update,)
Couchbase	Couchbase[1]	document as a list of named (structured) items (JSON document)	by key and key range, views via Javascript and MapReduce

MongoDB: Document-based NoSQL

- stands for humongous
- created by10gen
- open source (Implemented in C++)
- document-oriented database designed with
  - 1) scalability
  - 2) developer agility in mind
- Instead of storing data in tables and rows as we would do with a relational database, in MongoDB we store BSON (binary representation of JSON) documents with dynamic schemas (schema-less).

MongoDB: Background

- Goal:
  - bridge the gap between key-value stores (which are fast and scalable) and relational databases (which have rich functionality).



MongoDB: Background

- Key features:
  - scale "horizontally" over commodity hardware
  - support RDBMS features:
    - Ad hoc queries
    - Fully featured indexes
      - Single-field
      - Compound
      - Multi-key
      - Geospatial
      - Text
    - Aggregation operations (through *map-reduce*)

Data model

- DB ← Collections ← Documents
- **Collection** = set of "related" documents sharing "common" indexes
- **Primary key** (`object_id`) is automatically created and indexed for a document
  - Help narrow search span; otherwise mongoDB needs to scan all documents in every shards

Data model

- Schema design:
  - Embedding
    - nesting of objects and arrays inside a BSON document (pre-joined) → fast
  - Link (referencing)
    - references between documents → requires follow-up query
- Principle:
  - "embedding when you can, link when you must"

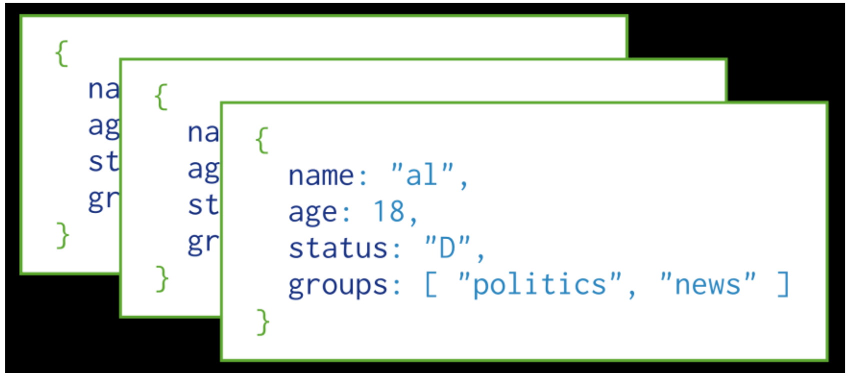
Comparison to RDBMS

RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (BSON → JSON)
Column	Field
Index	Index
Join	Embedding/Referencing
Foreign Key	Referencing
Primary Key	ObjectID

CRUD operations

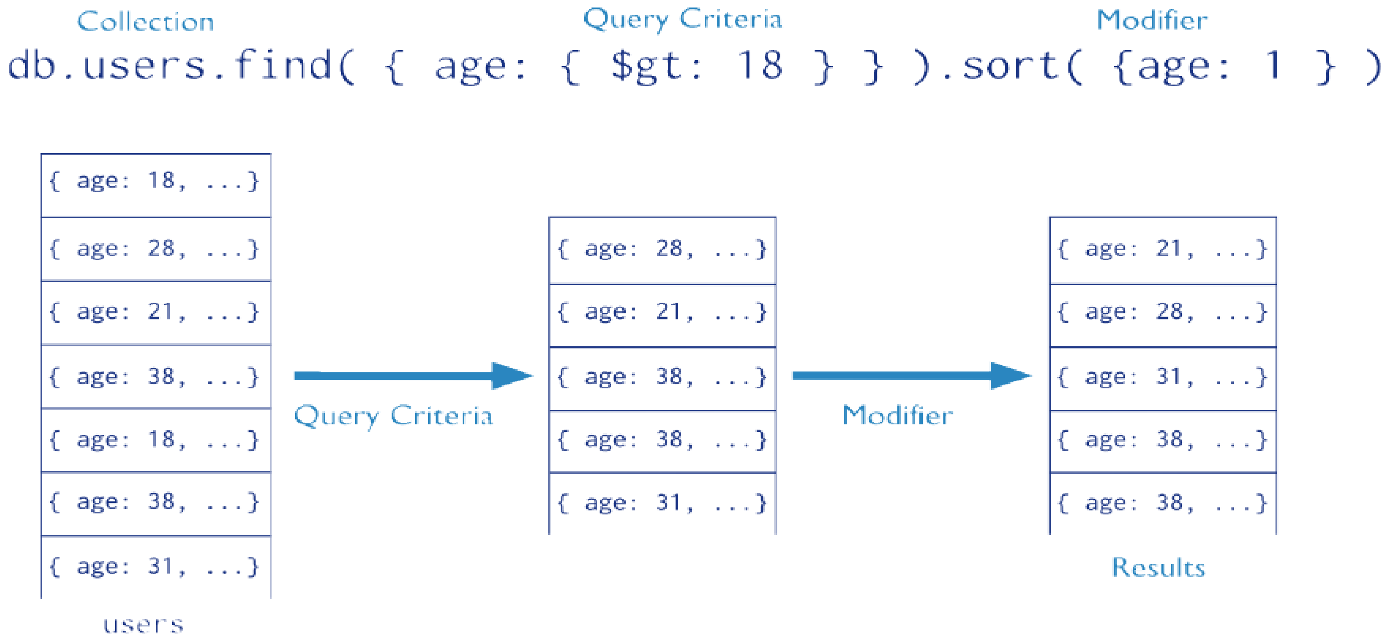
- CRUD
  - stands for **C**reate, **R**ead, **U**pdate, and **D**elete
  - used for reading and manipulating data
- Example: JSON documents and collection

```
{
   name: "sue",                      ← field: value
   age: 26,                          ← field: value
   status: "A",                      ← field: value
   groups: [ "news", "sports" ]      ← field: value
}
```



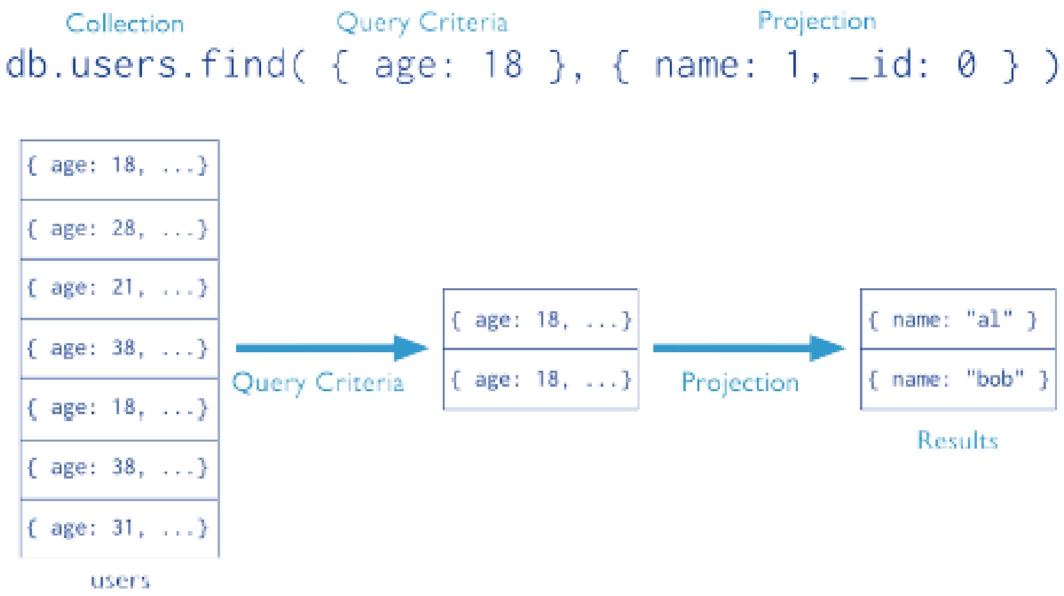
CRUD operations

- target *a specific collection at a time*
  - **NOT** support JOIN operations

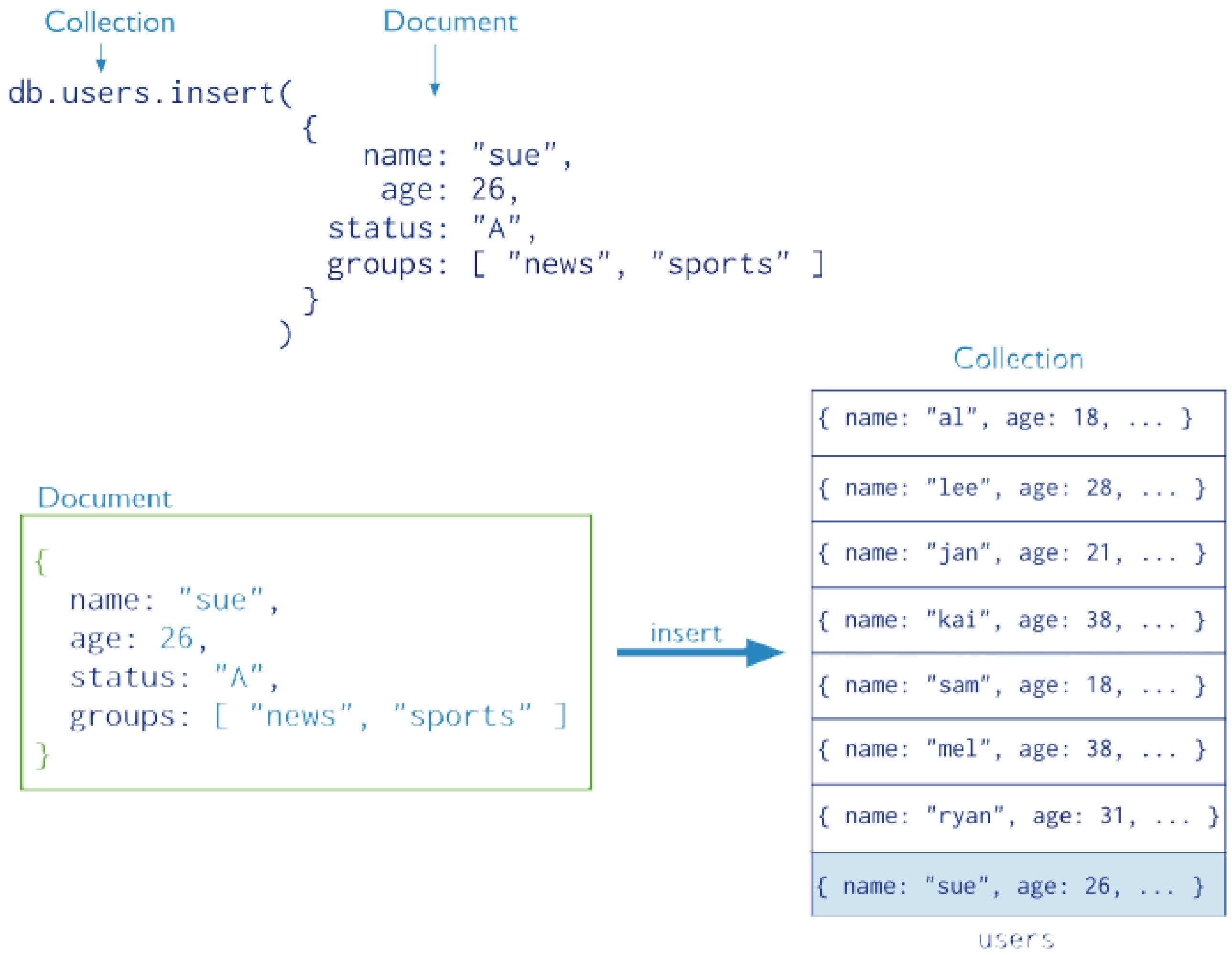


CRUD operations

- More example:



CRUD operations



CRUD operations

```
SELECT _id, name, address
FROM   users
WHERE  age > 18
LIMIT  5
```

projection
table
select criteria
cursor modifier

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

collection
query criteria
projection
cursor modifier

- Note: Results always include object id *(id)* unless explicitly specify (id:0)

- Results always include object id *(id)* unless explicitly specify (id:0)

CRUD operations

- Create
  - db.collection.insert(<document>)
  - db.collection.update(<query cond>, <update>, { upsert: true })
- Read
  - db.collection.find(<query cond>, <projection>)
- Update
  - db.collection.update(<query cond>, <update>, <options>)
- Delete
  - db.collection.remove(<query cond>, <justOne>)

Wait! Why ''Just One''? What about SQL system?

Conclusion

- NOSQL databases cover only a part of data-intensive cloud applications (mainly Web applications)
- Problems with cloud computing:
  - SaaS (**S**oftware **as a** **S**ervice or on-demand software) applications require enterprise-level functionality, including ACID transactions, security, and other features associated with commercial RDBMS technology, i.e. NOSQL should not be the only option in the cloud
  - Hybrid solutions:
    - Voldemort with MySQL as one of storage backend
    - deal with NOSQL data as semi-structured data

      → integrating RDBMS and NOSQL via SQL/XML

Conclusion

- next generation of highly scalable and elastic RDBMS: *NewSQL databases*
  - they are designed to scale out horizontally on shared nothing machines,
  - still provide ACID guarantees,
  - applications interact with the database primarily using SQL,
  - the system employs a lock-free concurrency control scheme to avoid user shut down,
  - the system provides higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc, etc.

Any Questions?