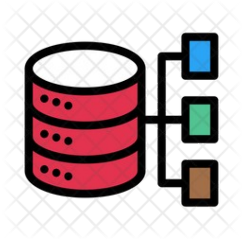
NoSQL Database

Tushar B. Kute,
http://tusharkute.com



NoSQL

- A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
- Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century, triggered by the needs of Web 2.0 companies.
- NoSQL databases are increasingly used in big data and real-time web applications.
- NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot-persistent architectures

NoSQL

- Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), finer control over availability and limiting the object-relational impedance mismatch.
- The data structures used by NoSQL databases (e.g. key–value pair, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL.
- The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

NoSQL – Types

- Wide column: Azure Cosmos DB, Accumulo, Cassandra, Scylla, HBase.
- Document: Azure Cosmos DB, Apache CouchDB, ArangoDB, BaseX, Clusterpoint, Couchbase, eXist-db, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- Key–value: Azure Cosmos DB, Aerospike, Apache Ignite, ArangoDB, Berkeley DB, Couchbase, Dynamo, FoundationDB, InfinityDB, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, SciDB, SDBM/Flat File dbm, ZooKeeper
- Graph: Azure Cosmos DB, AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso

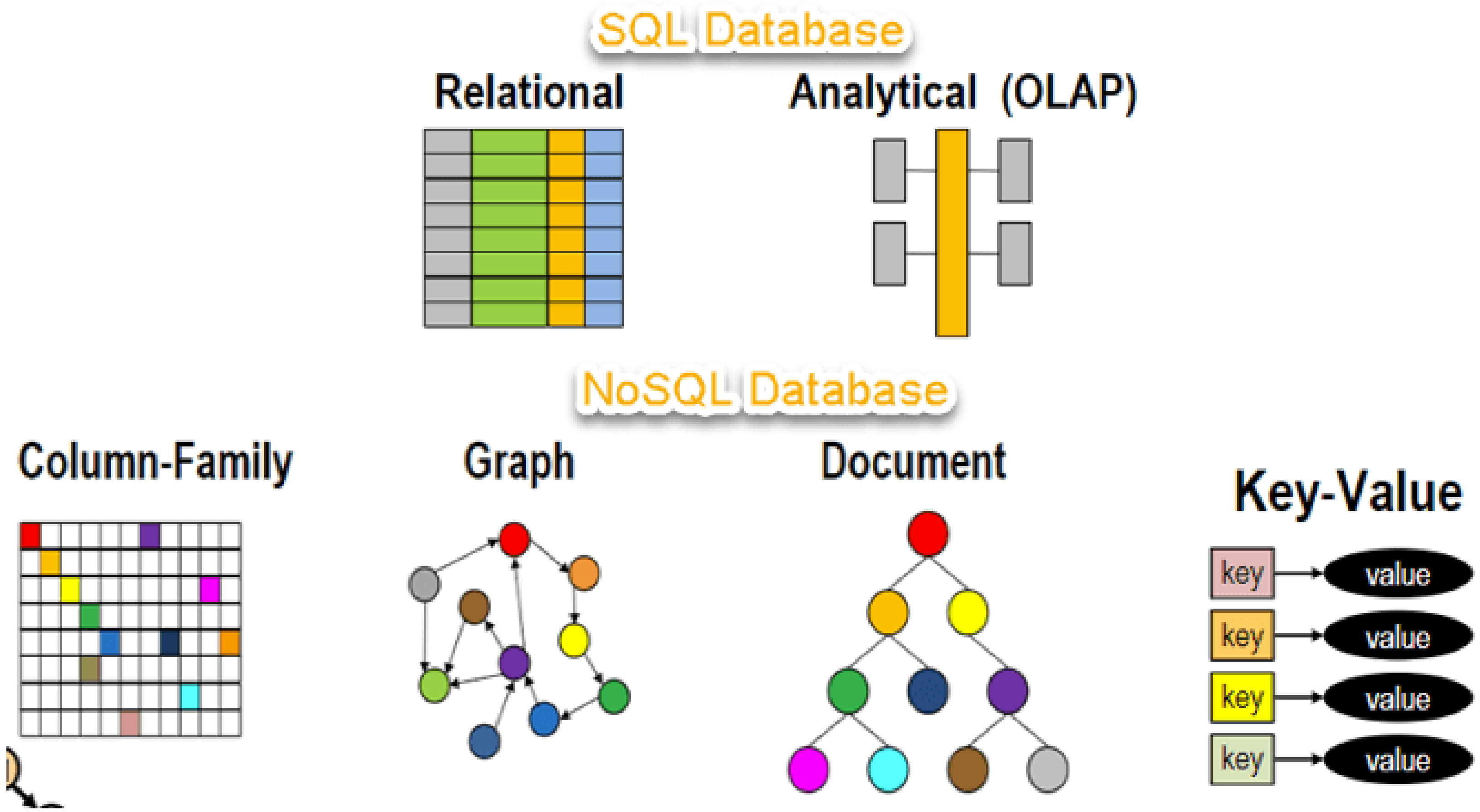
Database as a Service – SQL

- Amazon Aurora, MySQL based service
- Amazon Relational Database Service
- Clustrix Database as a Service
- Crunchy Bridge, PostgreSQL as a Service.
- EnterpriseDB Postgres Plus Cloud Database
- Google Cloud SQL
- Heroku PostgreSQL as a Service (shared and dedicated database options)
- MariaDB SkySQL
- Microsoft Azure SQL Database (MS SQL)
- Oracle Database Cloud Service
- Snowflake Cloud Data Warehouse
- Xeround Cloud Database* – MySQL front-end (*service no longer available)

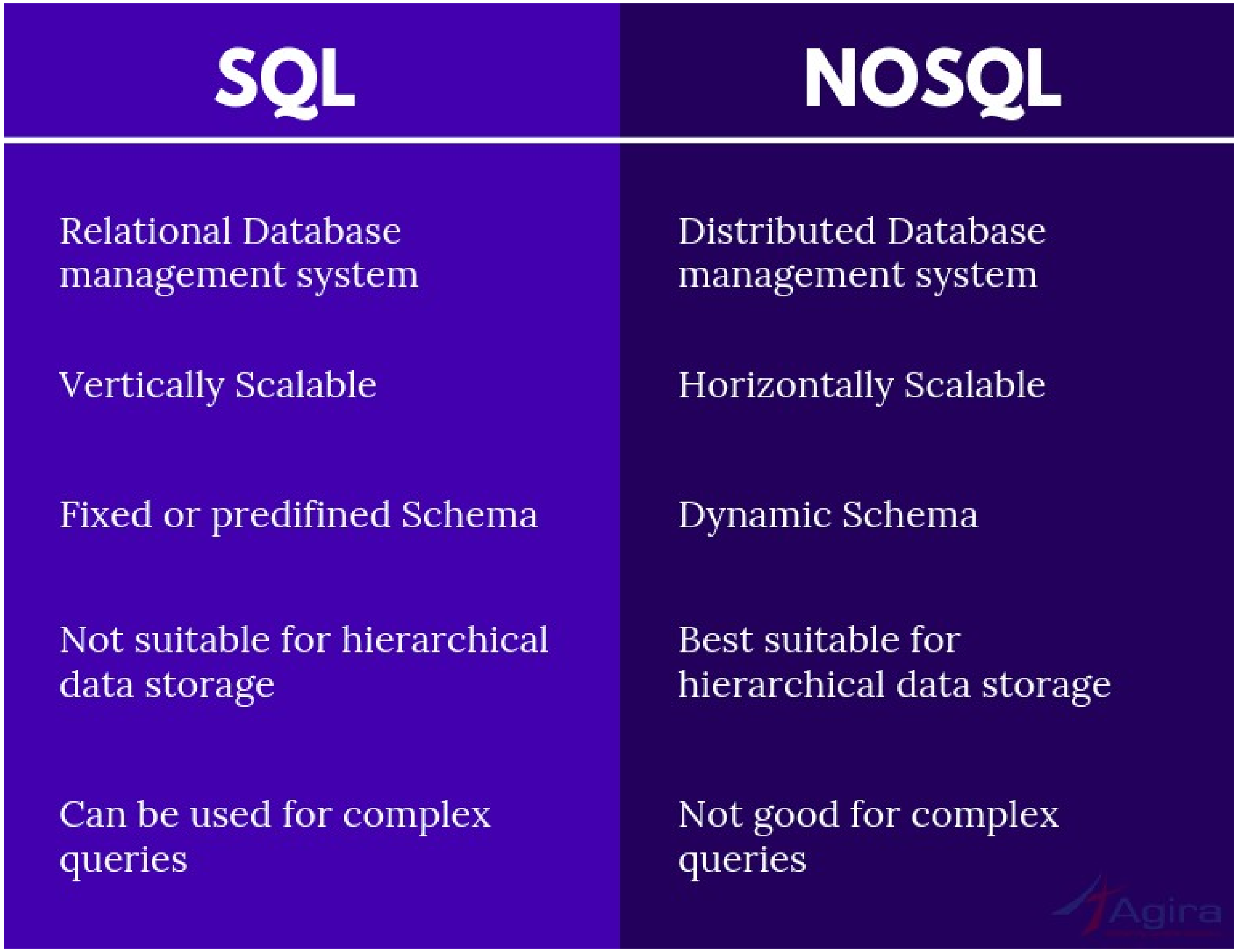
Database as a Service – NoSQL

- Amazon DynamoDB
- Amazon SimpleDB
- Azure Cosmos DB
- Cloudant Data Layer (CouchDB)
- EnterpriseDB Postgres Plus Cloud Database
- Google Cloud Bigtable
- Google Cloud Datastore
- MongoDB Database as a Service (several options)
- RavenDB Cloud Database as a Service
- Oracle NoSQL Database Cloud Service
- Amazon DocumentDB

Database Types



Comparison

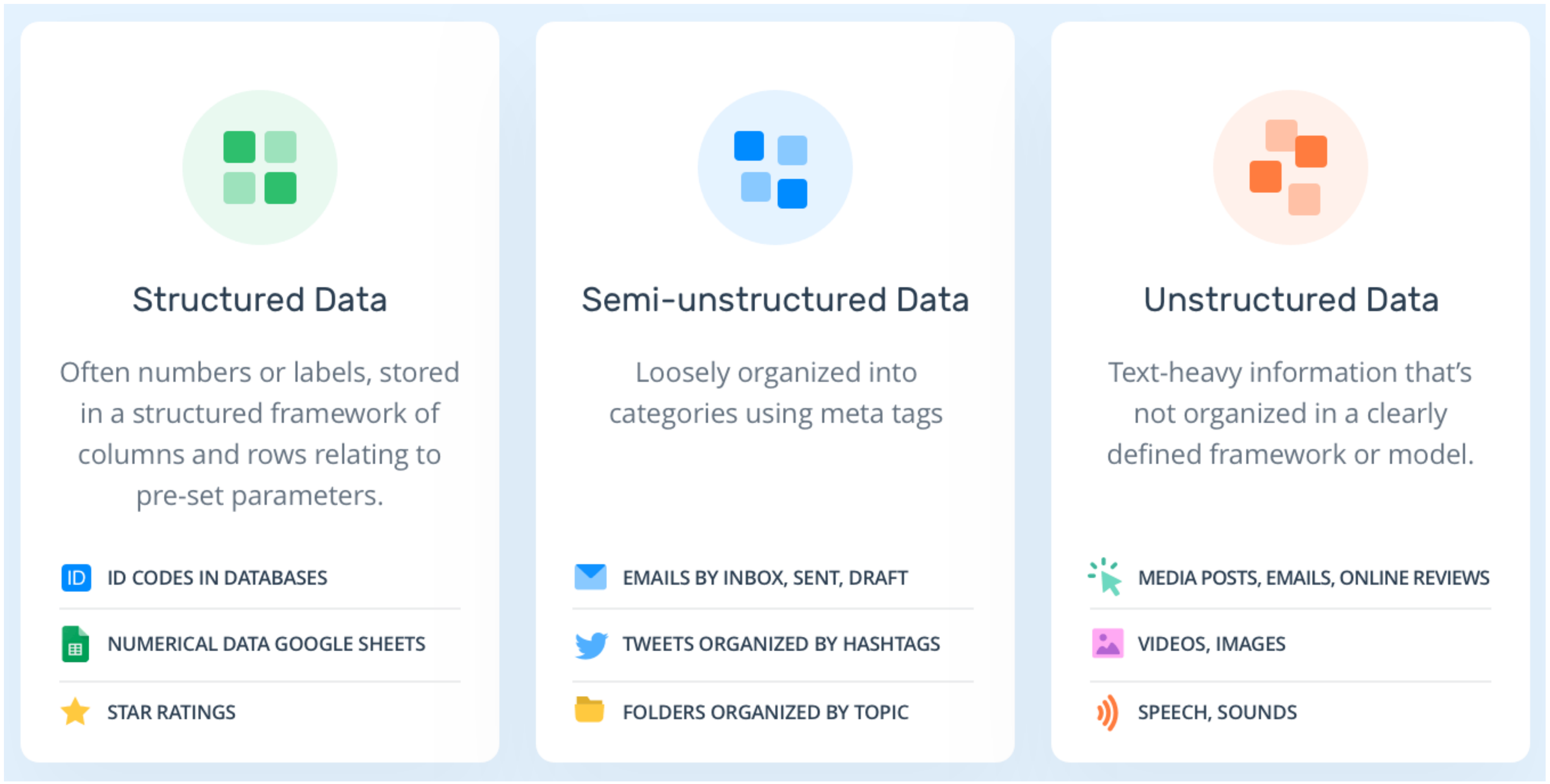


SQL	NOSQL
Relational Database management system	Distributed Database management system
Vertically Scalable	Horizontally Scalable
Fixed or predifined Schema	Dynamic Schema
Not suitable for hierarchical data storage	Best suitable for hierarchical data storage
Can be used for complex queries	Not good for complex queries

Why NoSQL?

- NoSQL databases are used in nearly every industry.
- Use cases range from the highly critical (e.g., storing financial data and healthcare records) to the more fun and frivolous (e.g., storing IoT readings from a smart kitty litter box).

Types of Data



Structured Data

Often numbers or labels, stored in a structured framework of columns and rows relating to pre-set parameters.

- ID CODES IN DATABASES
- NUMERICAL DATA GOOGLE SHEETS
- STAR RATINGS

Semi-unstructured Data

Loosely organized into categories using meta tags

- EMAILS BY INBOX, SENT, DRAFT
- TWEETS ORGANIZED BY HASHTAGS
- FOLDERS ORGANIZED BY TOPIC

Unstructured Data

Text-heavy information that's not organized in a clearly defined framework or model.

- MEDIA POSTS, EMAILS, ONLINE REVIEWS
- VIDEOS, IMAGES
- SPEECH, SOUNDS

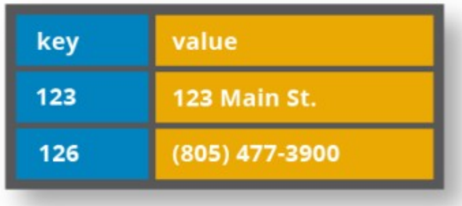
Storage Architectures

- Within the NoSQL system, there are multiple ways to store and retrieve records that surpass the limitations of relational systems. The primary ways data can be stored in NoSQL include:
  - Key-Value Store
  - Document Store
  - Column Store
  - Graph Store
- Some NoSQL databases work with more than one type of record store.

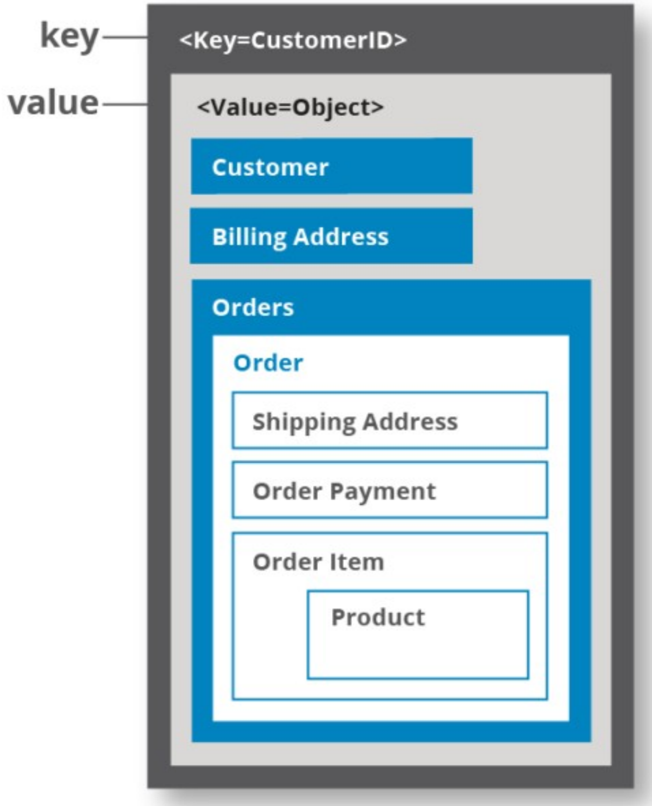
Key-Value Store

- The key-value store is a database system that stores record assets of unique identifiers with an associated (paired) value.
- This data pairing is referred to as a “key-value pair.” The “key” is the unique identifier. The “value” is the data being identified, or its location.
- A major benefit of key-value stores is that they are fast for data retrieval. Where relational systems store data across rows and columns and need to query across the database to return a record, the key-value store is more flexible and only has to search for the key, then return the associated value.

Key-Value Store



key	value
123	123 Main St.
126	(805) 477-3900



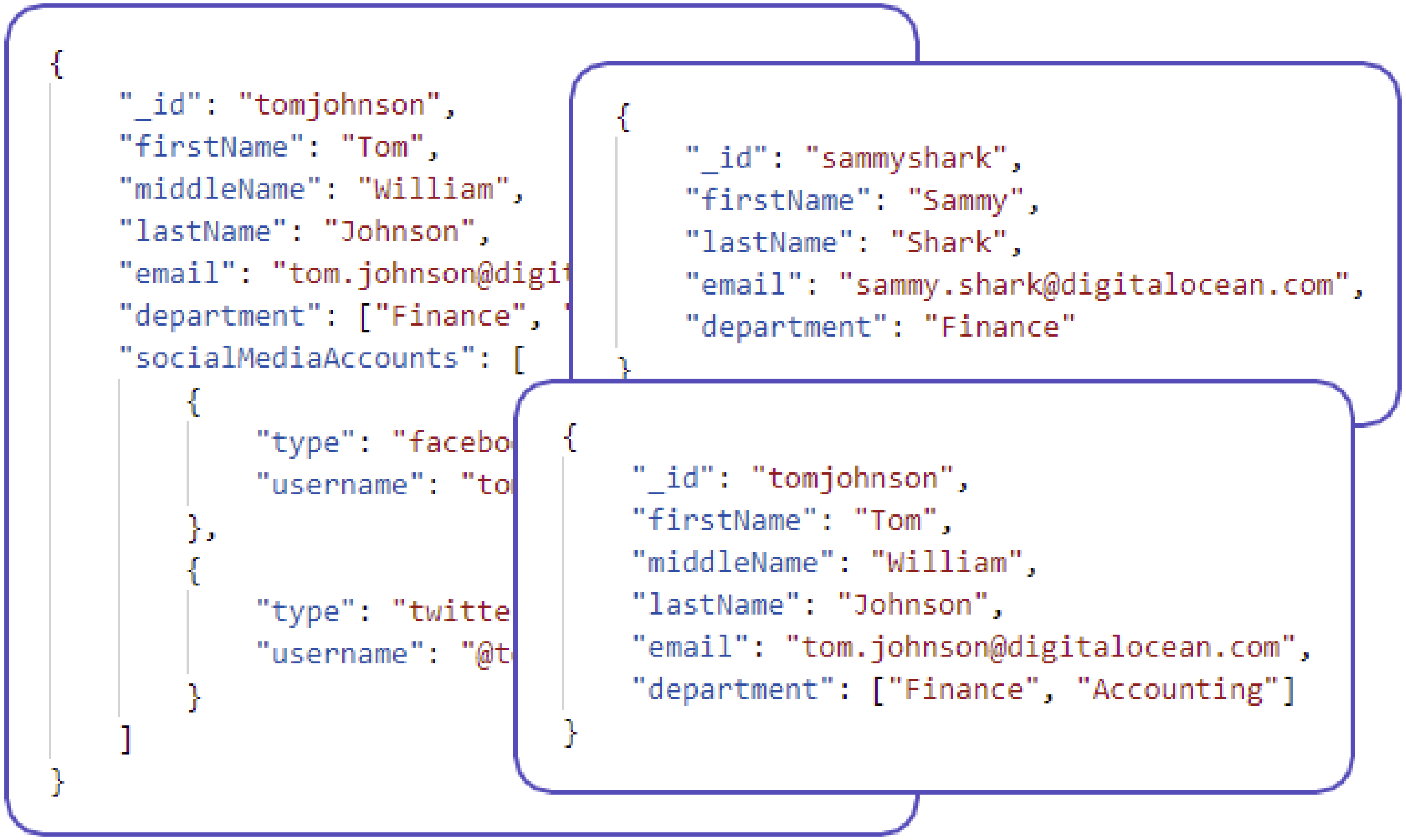
Key-Value Store

- Due to the speed of returns, and their flexibility, key-value stores are particularly useful in certain cases such as:
  - Storing, recalling, and updating product information, pricing, categories, and other eCommerce-related functions.
  - Storing user details, preferences, and session information for rapid recall and rewrites.
  - Generating real-time data to provide relevant advertising as users move through different areas of a platform or website.

Document Store

- Another storage option, the document-store, stores data in a semi-structured document. The data can then be ordered with markers.
- Information in this data type needs to be encoded in XML, JSON, BSON, or as a YAML file and is never stored in a table (which is why it is unsuitable for relational storage). Instead, complex datasets are contained in a single record.
- Retrieval occurs when a key is used to locate the document, and then that document is searched for the information required.

Document Store



```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digit
    "department": ["Finance", "
    "socialMediaAccounts": [
        {
            "type": "facebo
            "username": "to
        },
        {
            "type": "twitte
            "username": "@t
        }
    ]
}
```

```
{
    "_id": "sammyshark",
    "firstName": "Sammy",
    "lastName": "Shark",
    "email": "sammy.shark@digitalocean.com",
    "department": "Finance"
}
```

```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digitalocean.com",
    "department": ["Finance", "Accounting"]
}
```

Document Store

- A benefit of the document store is that different types of documents can be contained within a single store, and updates to those documents do not need to be related to the database.
- Also, because there are no fields within this store, and therefore, no empty cells for missing records, the document store is incredibly efficient at returning data fast.

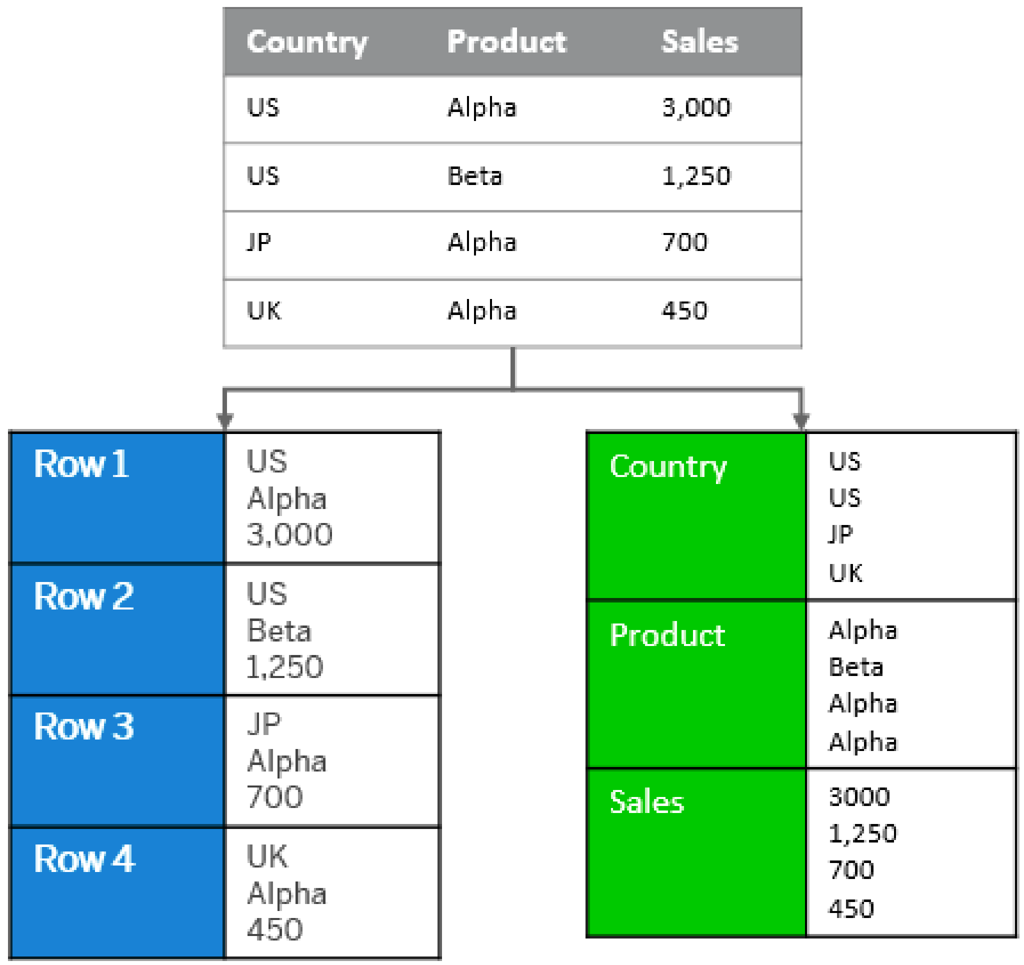
Document Store

- Document stores are highly useful when:
  - When working with JSON, BSON, XML, YAML files
  - You need to make changes to your data schema often
  - When you work with unstructured or semi-structured data
  - When you need something simple for development
- Document stores are flexible and easily scalable, and developers can work within them, even without prior knowledge of the system.

Column Store

- Column-based stores can also be a good NoSQL storage option. It records data in columns, rather than in rows.
- By storing data in columns, it is contained as a single, ongoing entry.
- This minimizes the number of disks accessed and avoids pulling in unnecessary memory, which speeds up record retrieval since a query does not need to pass over irrelevant rows to return information. Instead, only the information within the column is queried.

Column Store



Country	Product	Sales
US	Alpha	3,000
US	Beta	1,250
JP	Alpha	700
UK	Alpha	450

Row 1	US Alpha 3,000
Row 2	US Beta 1,250
Row 3	JP Alpha 700
Row 4	UK Alpha 450

Row-based storage

Country	US US JP UK
Product	Alpha Beta Alpha Alpha
Sales	3000 1,250 700 450

Column-based storage

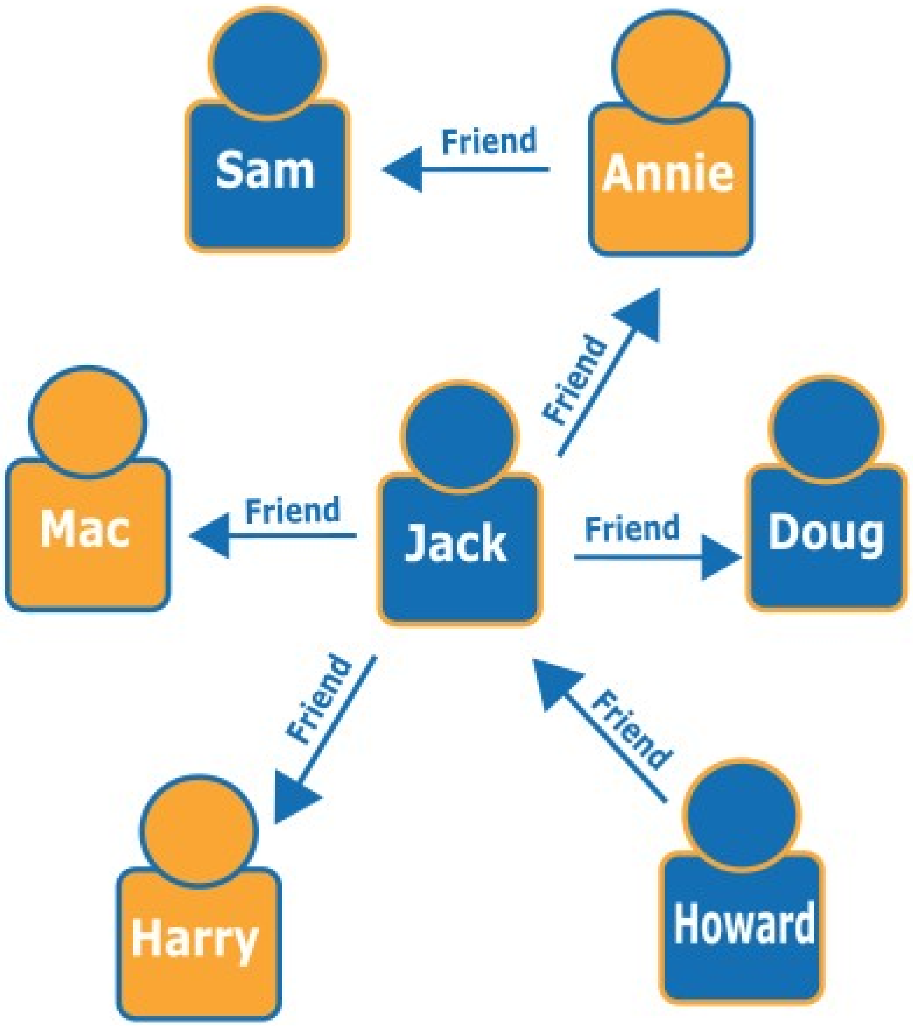
Column Store

- Column stores are most frequently used by companies that deal with large data warehousing setups.
- The data is structured as a table with columns and rows, and is then stored logically in a column-wise format so irrelevant data does not have to be bypassed before the target data is accessed and returned.

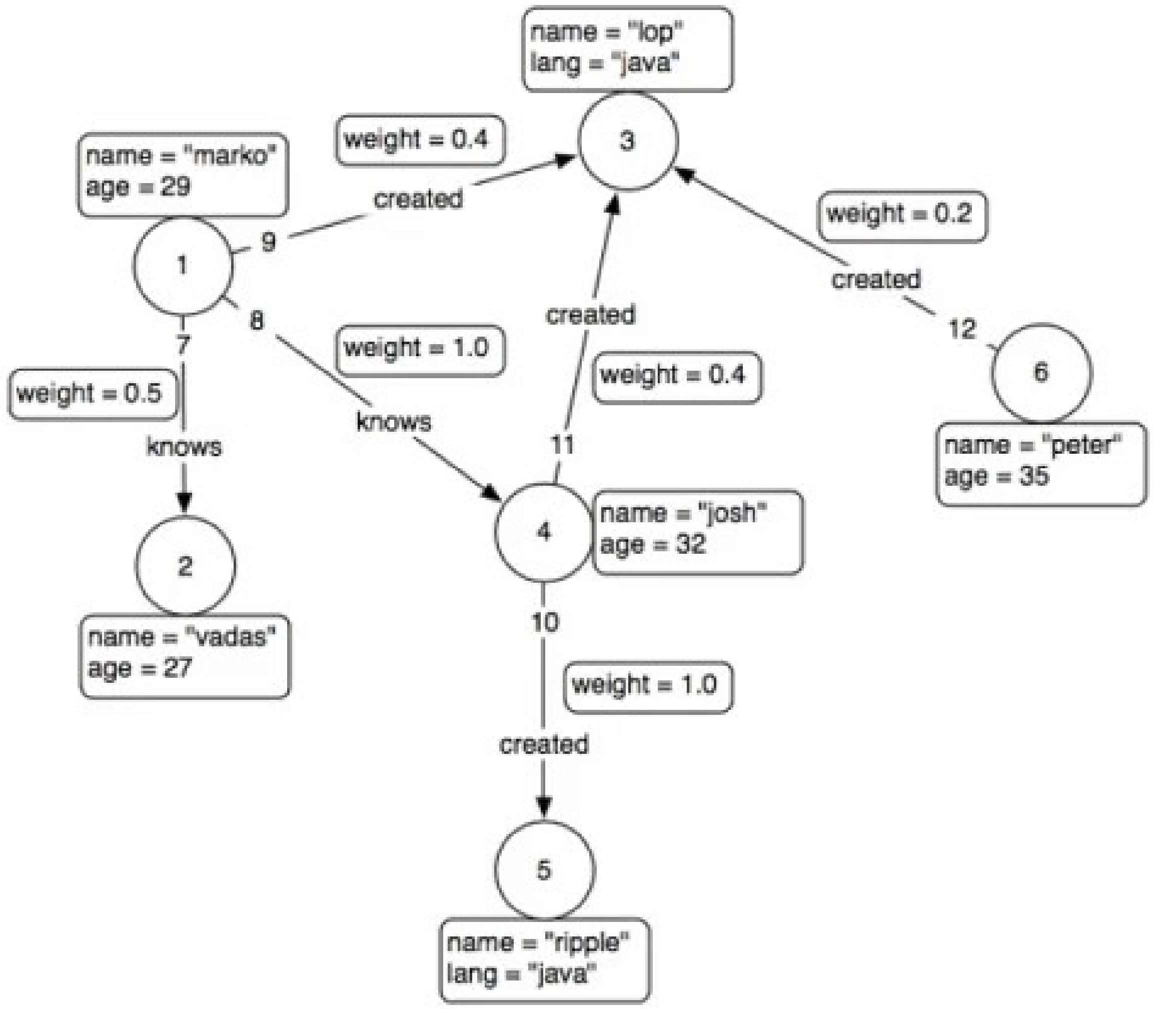
Graph Store

- For some businesses, relationships and connections between data take priority, so a graph NoSQL storage makes the most sense.
- Graph stores represent data in graphs instead of tables which makes them highly flexible and easily extendable.
- The graph NoSQL storage database returns search results fast and speeds up indexing by representing data as networks of nodes and edges.
- In a graph store, data is stored in nodes and then connected with relationships in edges which are then grouped by labels.

Graph Store


Sam
Friend
Annie
Friend
Mac
Friend
Jack
Friend
Doug
Friend
Friend
Harry
Howard

Graph Store


name = "lop"
lang = "java"
3
weight = 0.4
name = "marko"
age = 29
created
9
1
8
7
weight = 0.5
knows
weight = 1.0
knows
created
weight = 0.4
11
weight = 0.2
created
12
6
name = "peter"
age = 35
2
name = "vadas"
age = 27
4
name = "josh"
age = 32
10
weight = 1.0
created
5
name = "ripple"
lang = "java"

Graph Store

- Graph stores are most useful for things like:
  - Data visualization and graph-style analytics
  - Fraud prevention and enterprise operations
  - Geospatial routing
  - Payment systems
  - Social networking systems

Why NoSQL?

- Support large numbers of concurrent users (tens of thousands, perhaps millions)
- Deliver highly responsive experiences to a globally distributed base of users
- Be always available – no downtime
- Handle semi- and unstructured data
- Rapidly adapt to changing requirements with frequent updates and new features

When NoSQL?

- When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:
  - Fast-paced Agile development
  - Storage of structured and semi-structured data
  - Huge volumes of data
  - Requirements for scale-out architecture
  - Modern application paradigms like microservices and real-time streaming

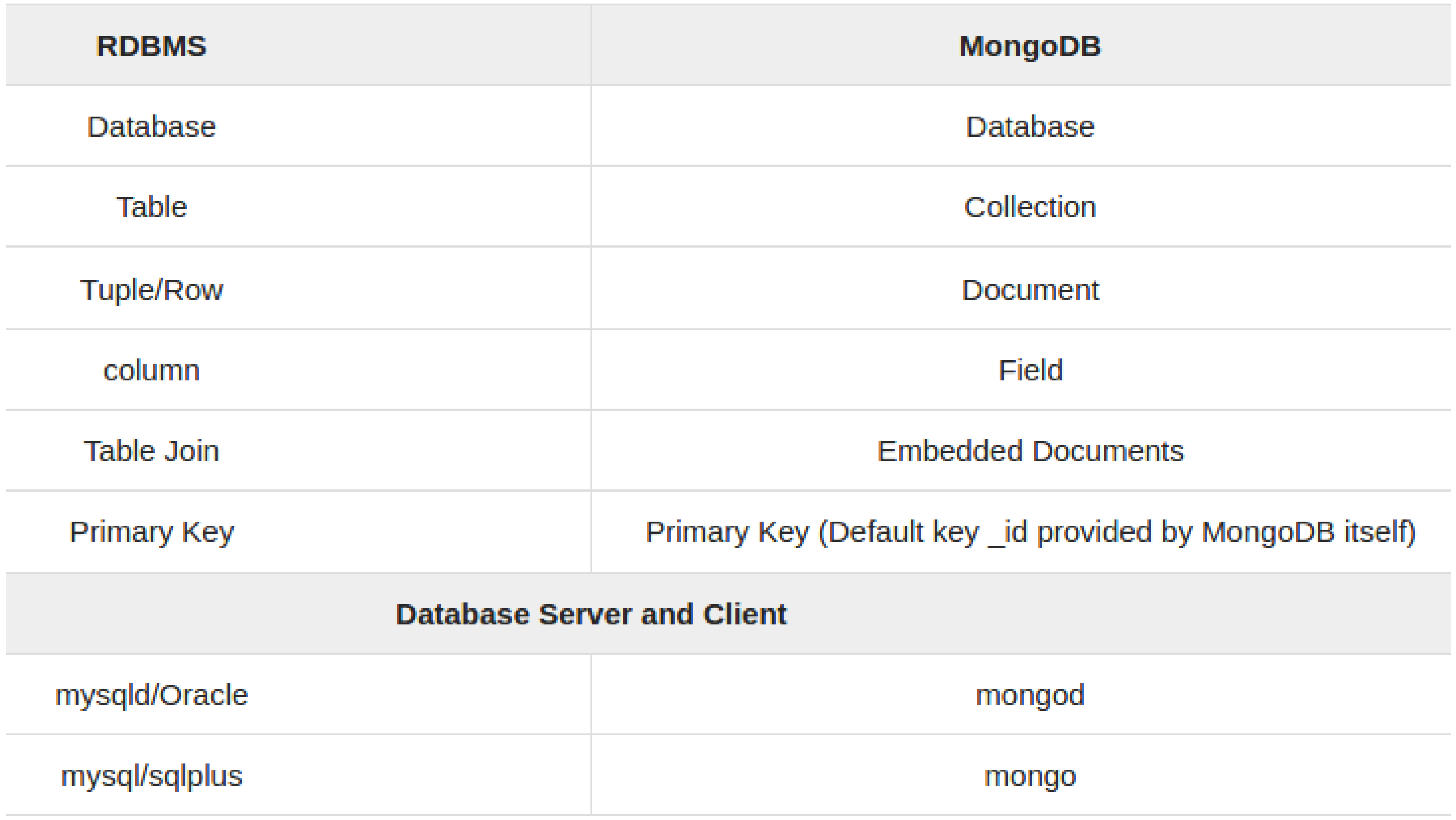
Misconcepts

- Over the years, many misconceptions about NoSQL databases have spread throughout the developer community. Two of the most common misconceptions:
  - Relationship data is best suited for relational databases.
  - NoSQL databases don't support ACID transactions.

Terminologies

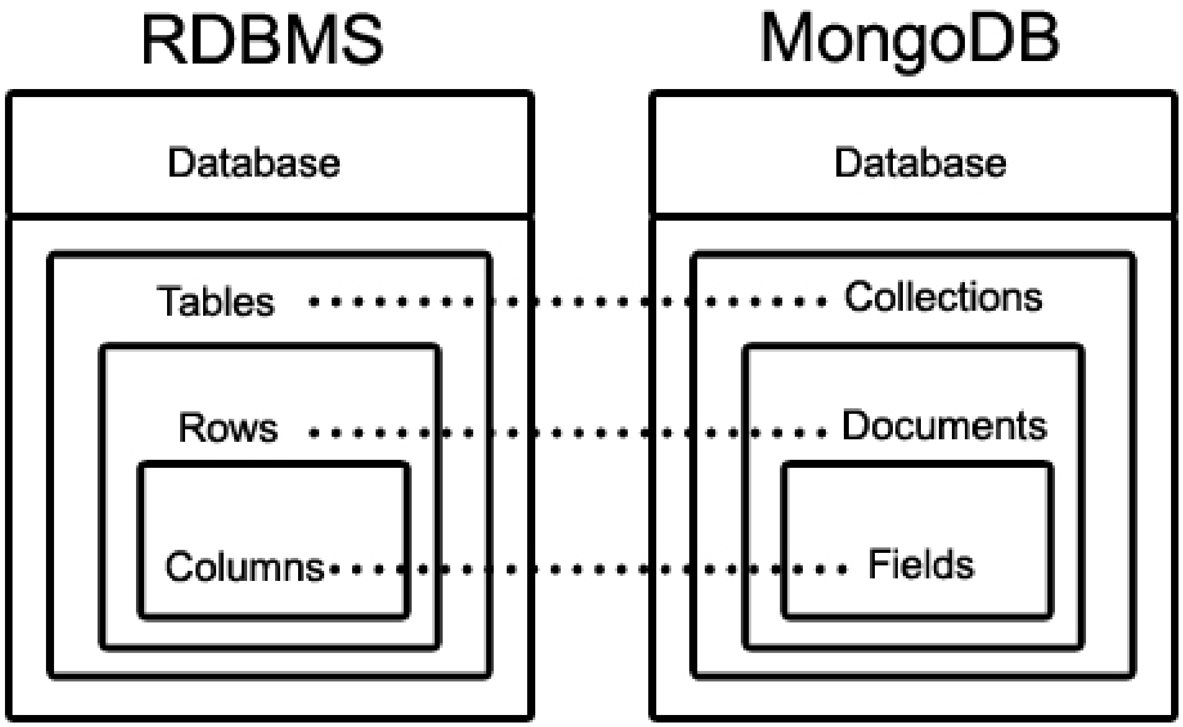
- Database
  - Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
- Collection
  - Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database.
  - Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
- Document
  - A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

RDBMS vs. MongoDB



RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	mongod
mysql/sqlplus	mongo

RDBMS vs. MongoDB



Example Document

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'MiTU Skillologies',
   url: 'https://www.mitu.co.in',
   tags: ['mongodb', 'database', 'NoSQL'],
   comments: [
      {
         user:'user1',
         message: 'My first comment',
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

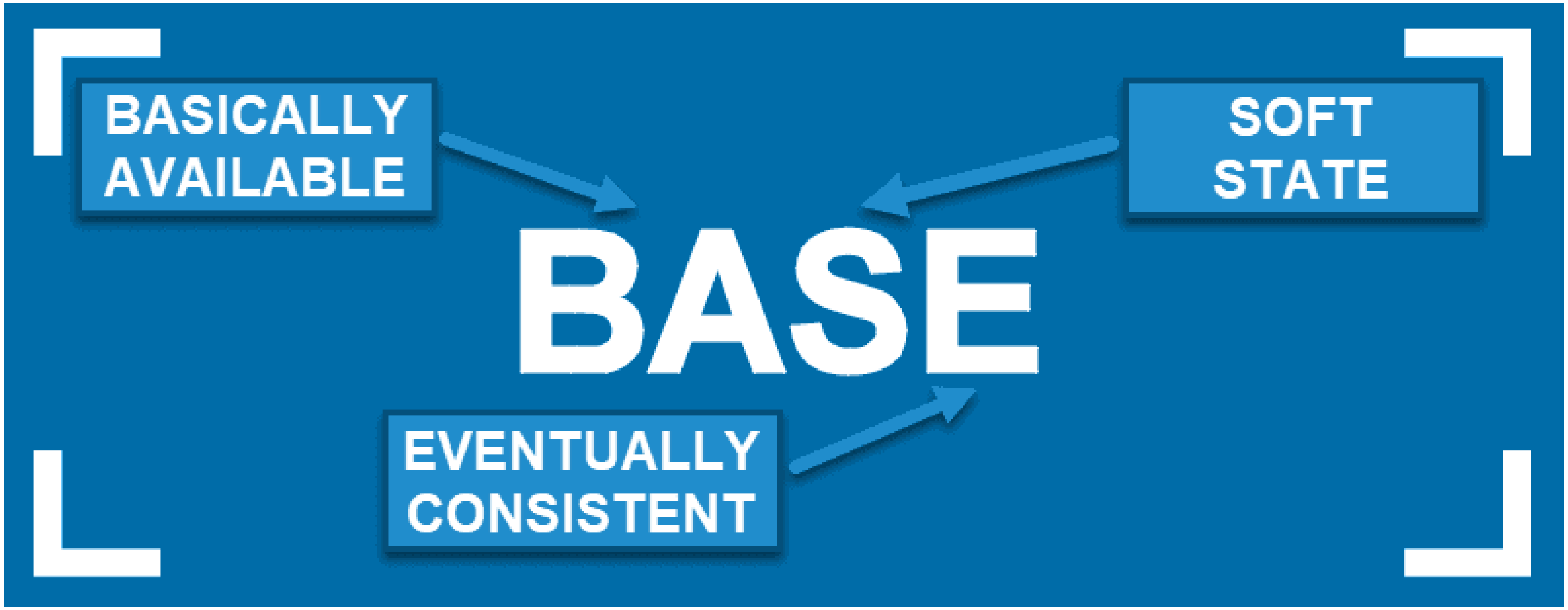
Advantages

- Schema less – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Ease of scale-out – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

BASE Model

- The rise of NoSQL databases provided a flexible and fluid way to manipulate data.
- As a result, a new database model was designed, reflecting these properties.
- The acronym BASE is slightly more confusing than ACID.
- However, the words behind it suggest ways in which the BASE model is different.

BASE Model


BASICALLY AVAILABLE
SOFT STATE
BASE
EVENTUALLY CONSISTENT

BASE Model

- Basically Available – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.
- Soft State – Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers.
- Eventually Consistent – The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

BASE Model – Which Databases?

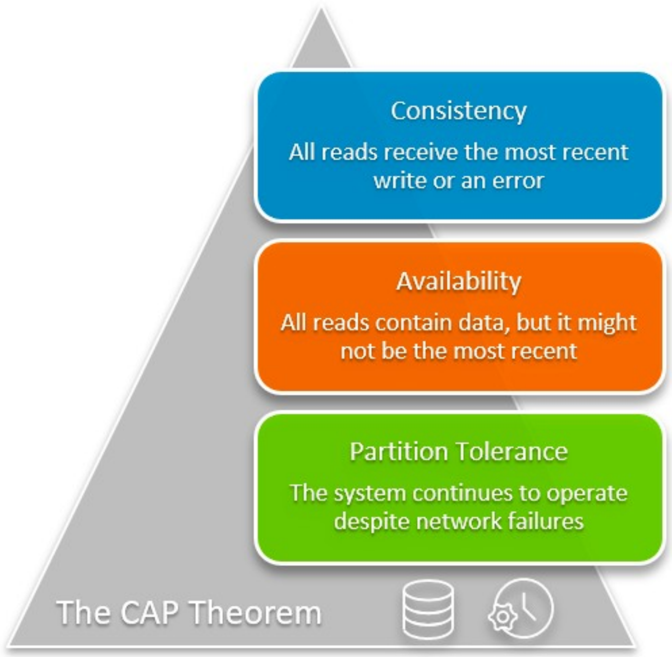
- Just as SQL databases are almost uniformly ACID compliant, NoSQL databases tend to conform to BASE principles.
- MongoDB, Cassandra and Redis are among the most popular NoSQL solutions, together with Amazon DynamoDB and Couchbase.

Which is best?

- It is not possible to give a straight answer to the question of which database model is better. Therefore, a decision must be reached by considering all the aspects of the project.
- Given their highly structured nature, ACID-compliant databases will be a better fit for those who require consistency, predictability, and reliability.
- Those who consider growth to be among their priorities will likely want to choose the BASE model, because it enables easier scaling up and provides more flexibility.
- However, BASE also requires developers who will know how to deal with the limitations of the model.

CAP Theorem

- The CAP theorem is a belief from theoretical computer science about distributed data stores that claims, in the event of a network failure on a distributed database, it is possible to provide either consistency or availability—but not both.


Consistency
All reads receive the most recent write or an error
Availability
All reads contain data, but it might not be the most recent
Partition Tolerance
The system continues to operate despite network failures
The CAP Theorem

CAP Theorem

- The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores:
  - Consistency. All reads receive the most recent write or an error.
  - Availability. All reads contain data, but it might not be the most recent.
  - Partition tolerance. The system continues to operate despite network failures (ie; dropped partitions, slow network connections, or unavailable network connections between nodes.)

CAP Theorem

- In normal operations, your data store provides all three functions.
- But the CAP theorem maintains that when a distributed database experiences a network failure, you can provide either consistency or availability.
- It’s a tradeoff. All other times, all three can be provided. But, in the event of a network failure, a choice must be made.

MongoDB

- MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.
- Database
  - Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
- Collection
  - Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Installation on Ubuntu Linux

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

- To start the mongodb type:

```
mongo
```

MongoDB Compass

- MongoDB compass is nothing but a graphical user interface that can be connected to the MongoDB database and used to find, analyze, modify, and visualize the data stored in the database without requiring any knowledge of queries.
- MongoDB acts as an alternative to Mongo Shell. Mongo Shell can also perform all the aforementioned tasks but requires a lot of technical expertise.
- MongoDB is GUI-based whereas Shell is more technicality based i.e it uses specific commands and queries. MongoDB Compass is present in the Github repo since it is an open-source tool.

MongoDB Compass

- All the data stored in the database can be visualized and explored.
- Data stored in the database can be modified, created, updated, or deleted.
- Shows the Server Statistics in Real-time.
- The Visual representations clearly depict the performance issues and recommend plans.
- All the indexes can be managed.
- JSON schema validation rules can be sued to validate data.
- A wide range of plugins is available.

MongoDB Compass: Installation

- Directly download the required files using the command mentioned below(using wget command). To open the terminal click on ctrl+alt+T.

```
wget https://downloads.mongodb.com/compass/mongodb-compass_1.30.1_amd64.deb
```

- If you have downloaded the file using the terminal use the following command to install the MongoDB Compass Ubuntu.

```
sudo dpkg -i mongodb-compass_1.30.1_amd64.deb
```

Thank you

@mitu_skillologies @mITuSkillologies @mitu_group @mitu-skillologies @MITUSkillologies

kaggle
@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com