NOSQL DATABASE

MONGODB DAY 1

Mohamed Elsalamony
Technical Program Manager | PL | Tech Lead | Scrum Master (ASM ®)| Instructor| Mentor

LECTURE AGENDA

- **Objective** :
  - Understand **Different between SQL and NoSQL**
  - Understand **Database Types** and **Their Differences**
  - Learn How to **Choose the Appropriate Database**
  - Gain Proficiency in **JSON**
  - Perform Simple **CRUD** Operations (Create, Read, Update, Delete)

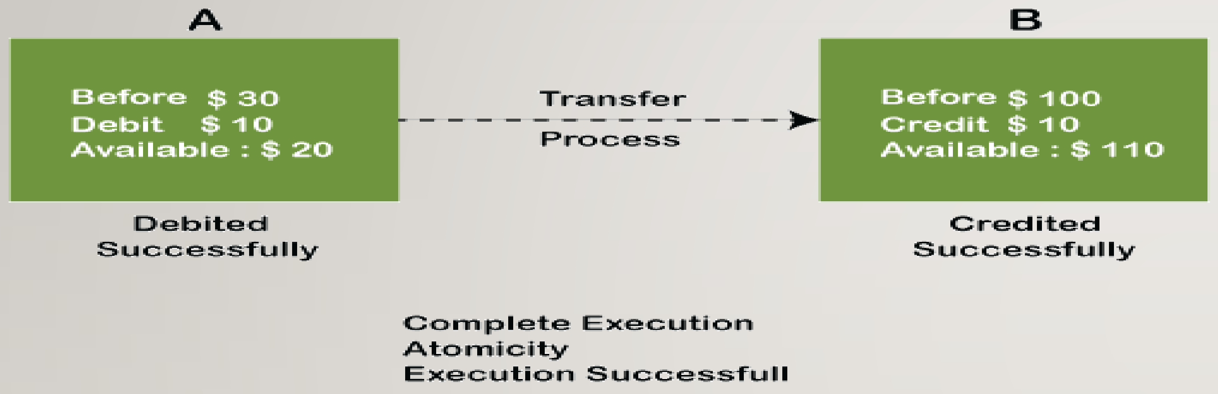
LECTURE AGENDA

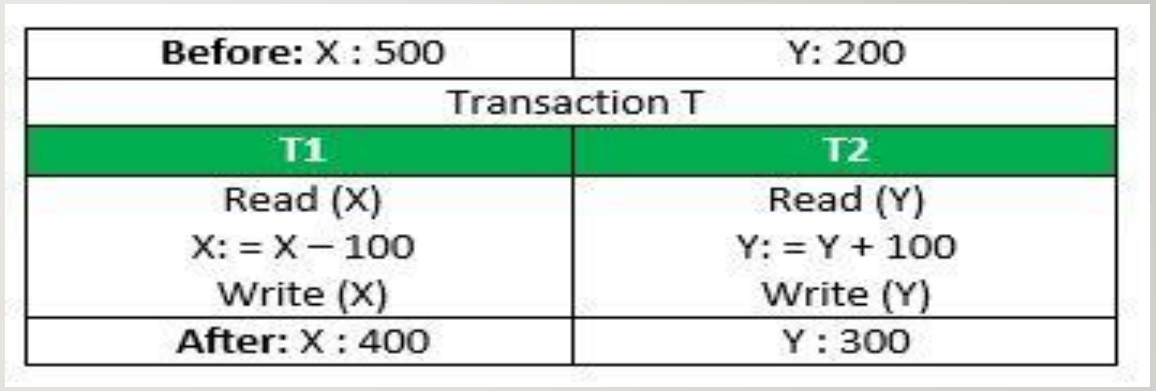
- SQL vs NoSQL
- NoSQL Types
- Why use NoSQL
- When to use NoSQL / Not use
- Structured vs non-structured vs semi-structured
- CAP Theory
- Mongodb Syntax
- JSON
- Mapping SQL to Mongodb
- Intro and Coding on **Robo 3T Studio**

STRUCTURE VS NO STRUCTURE

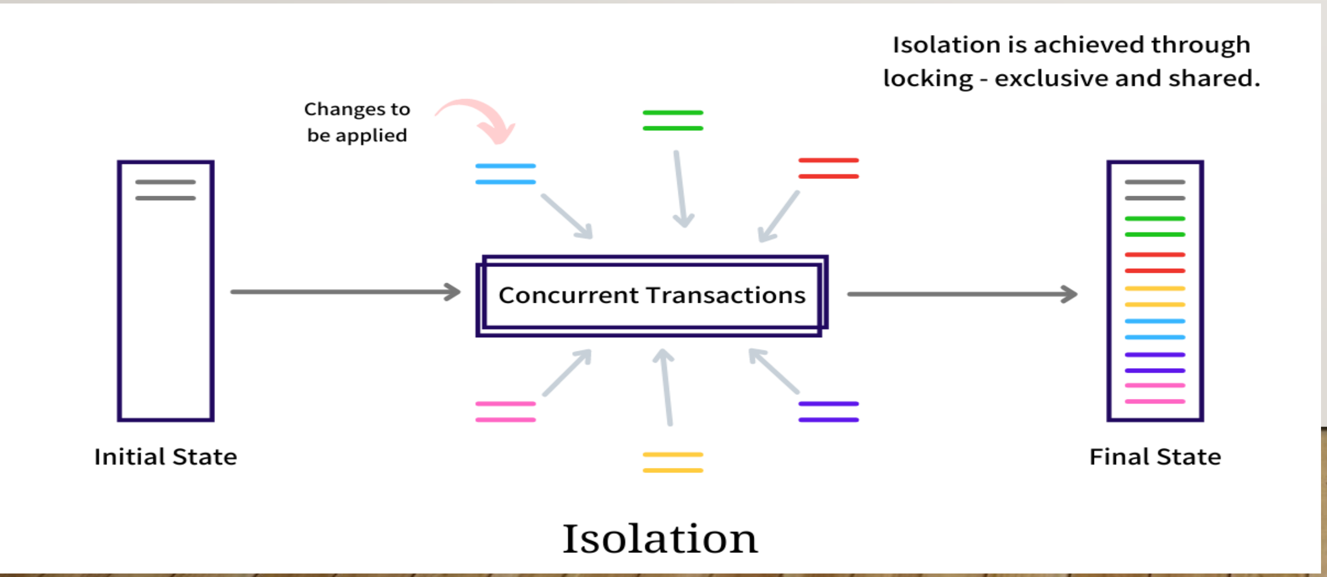
Key	**SQL**	**NO SQL**
Relational/No Relational	RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	Non-relational or distributed database system.
Schema	These databases have fixed or static or predefined schema	They have dynamic schema
Used for	These databases are **not** suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
Scale	Vertically Scalable eg. PC Increase CPU , Ram [Limit]	Horizontally scalable eg. More than On PCs
property	Follows **ACID** property(atomicity, consistency, isolation, and durability) ***(Put consistency over Availability)***	Follows **BASE** (Basically Available , Soft state, Eventually consistent) ***(Put Availability over Consistency)***
Examples	MySQL, PostgreSQL, Oracle, MS-SQL Server	MongoDB, GraphQL, HBase, Neo4j, Cassandra

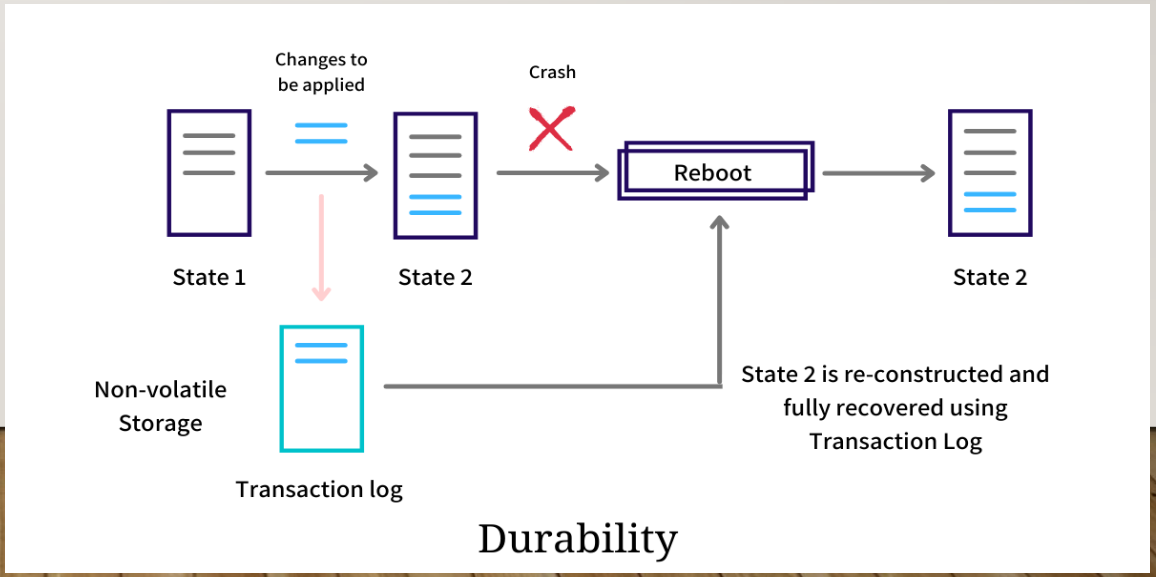
ACID - BASE



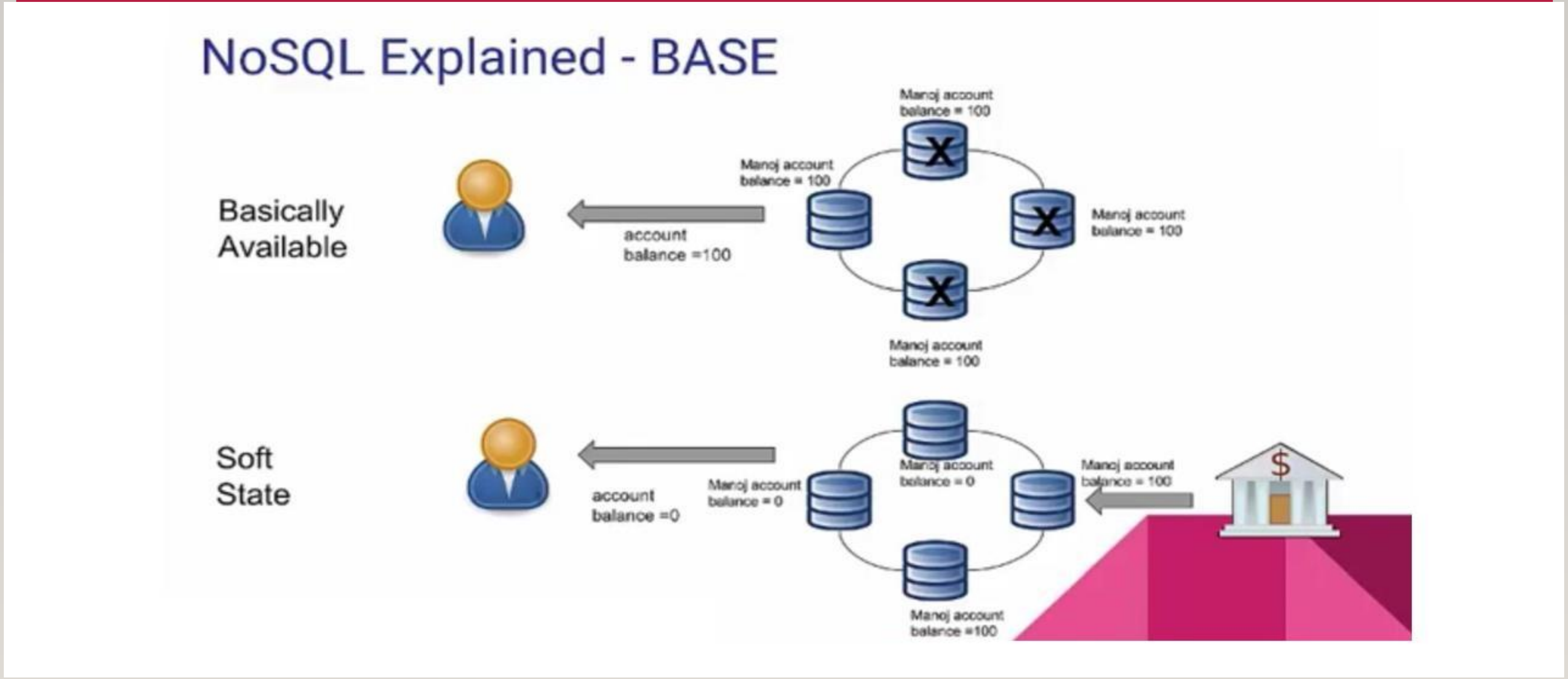


Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

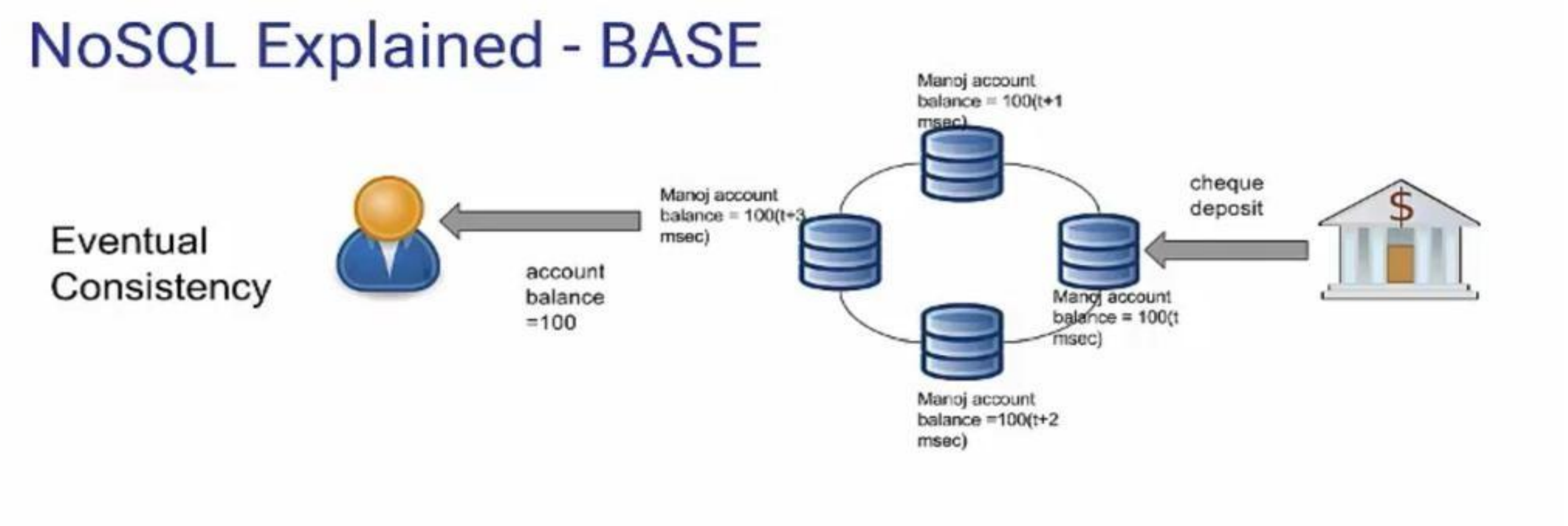




ACID - BASE

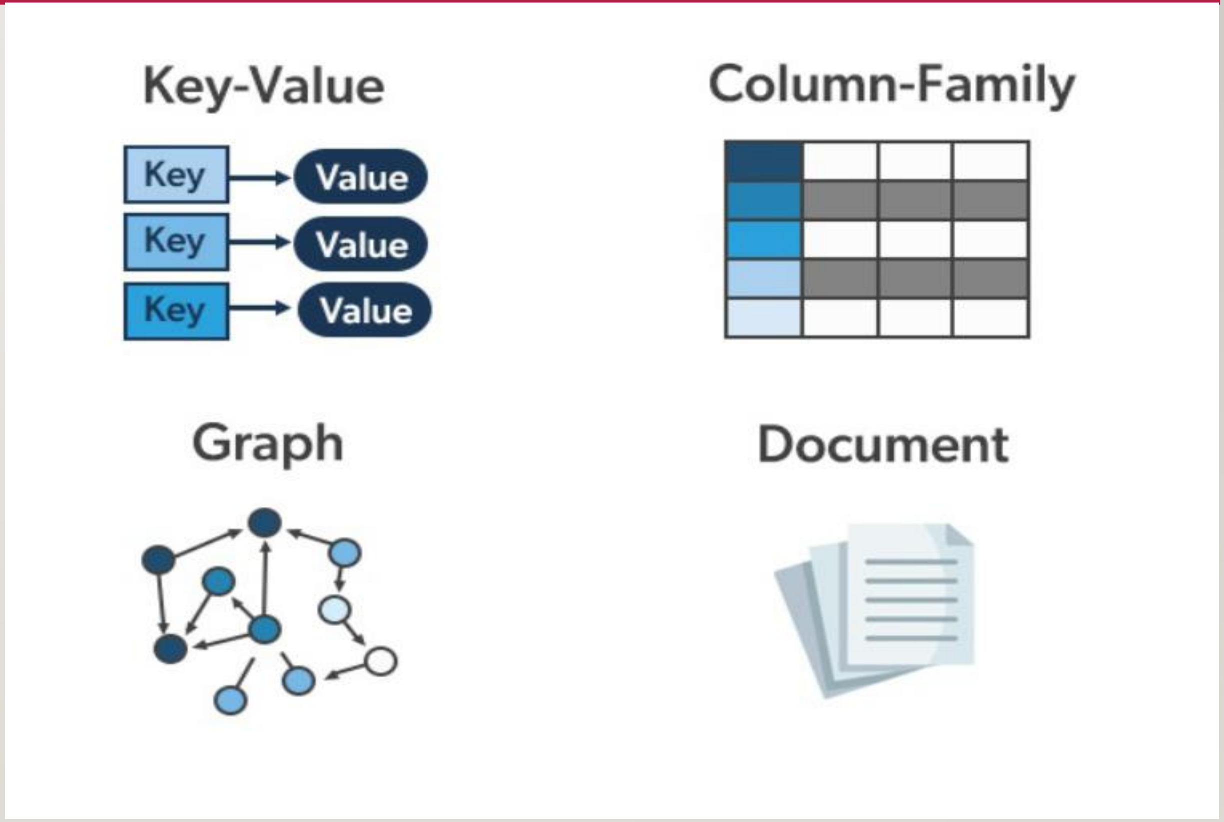

NoSQL Explained - BASE
Basically Available
Manoj account balance = 100
Manoj account balance = 100
Manoj account balance = 100
Manoj account balance = 100
account balance =100
Soft State
account balance =0
Manoj account balance = 0
Manoj account balance = 0
Manoj account balance = 100
Manoj account balance =100
$

ACID - BASE


NoSQL Explained - BASE
Eventual Consistency
account balance =100
Manoj account balance = 100(t+3 msec)
Manoj account balance = 100(t+1 msec)
Manoj account balance = 100(t msec)
Manoj account balance =100(t+2 msec)
cheque deposit
$

NOSQL TYPES

- **Key-value** =>FoundationDB
- **Document** =>MongoDB
- **Column Family**=>Cassandra
- **Graph** =>Neo4J



KEY-VALUE STORE

- Store Student Data
- Key : Student ID
- Value : Object

{

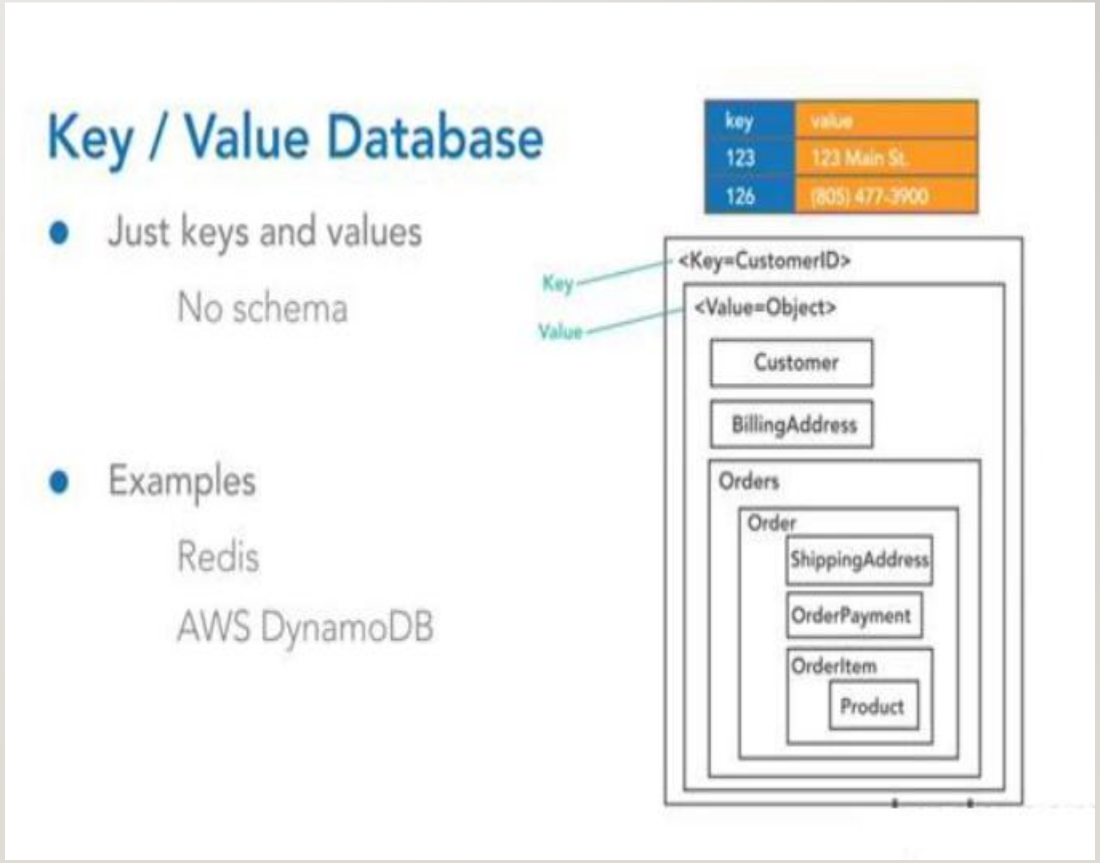
Name :”Ahmed” , Address:“Alex”

},

{

Name:”Eman”,Address:”Alex”

}



Key / Value Database

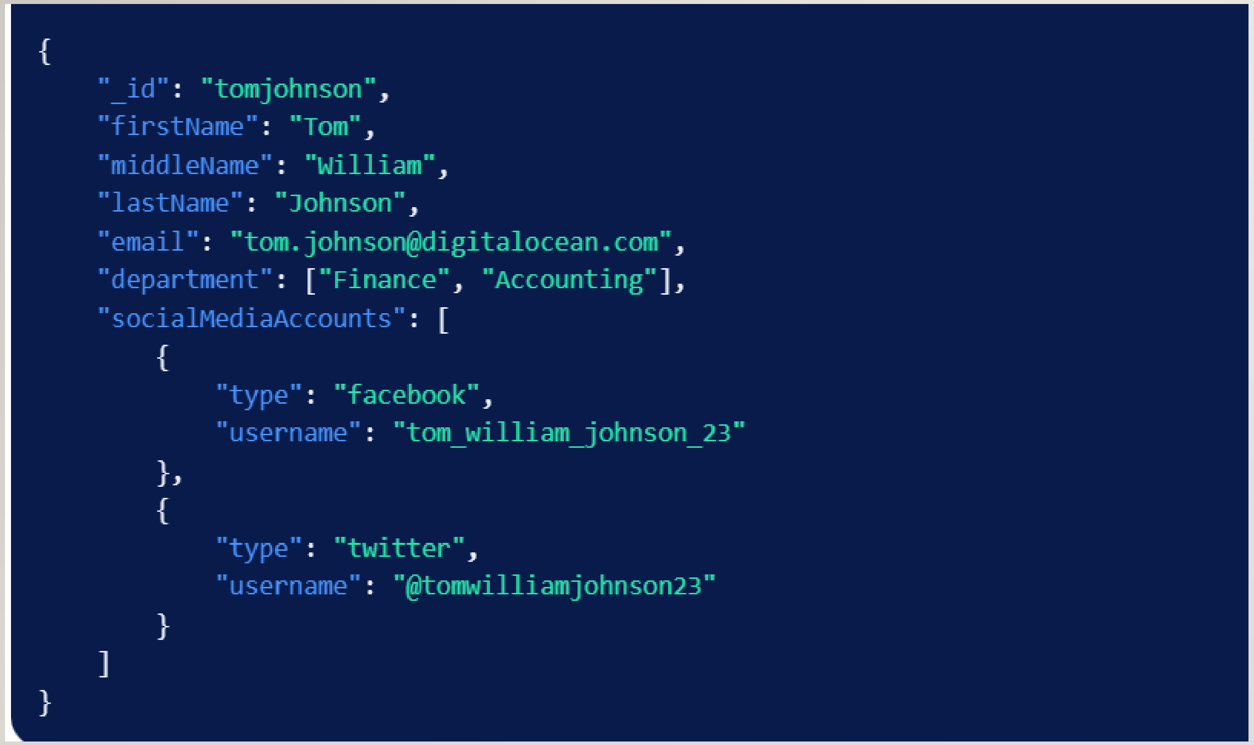
- Just keys and values
  - No schema
- Examples
  - Redis
  - AWS DynamoDB

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

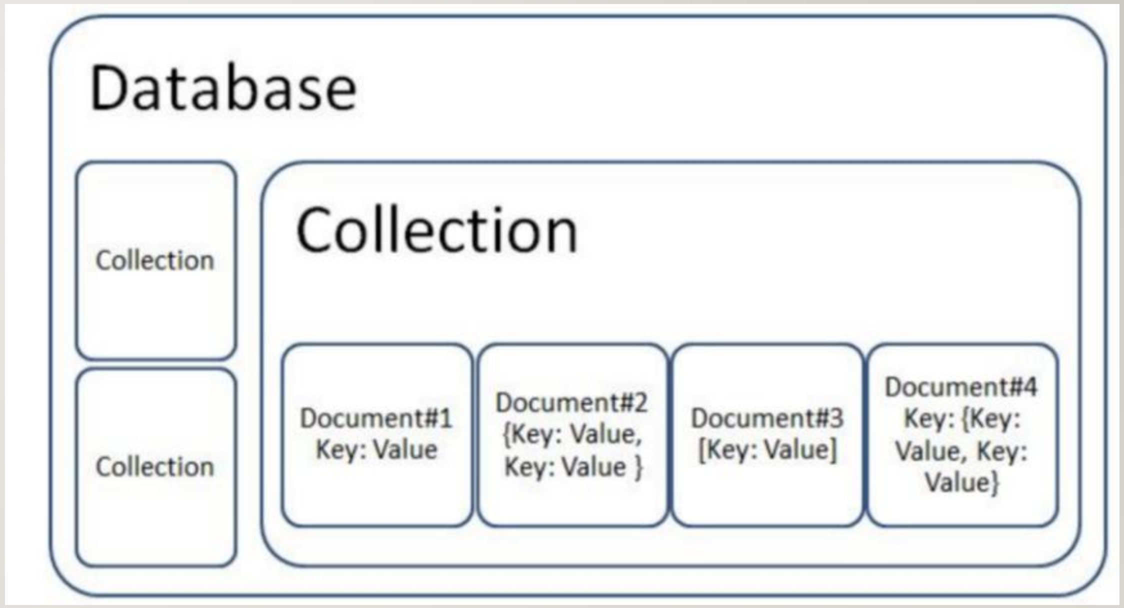
KEY-VALUE STORE

Advantages	Disadvantages
Every thing is Object which **no** Structure needed Ex: Student has {Name, Age} other has {Name, Email}.	Each student has Track key as text Student1:Track :OS” Student2:Track :OS” No Relational as SQL **Redundant data**.
DB is Object and Programming Language is Object Which facilitate work.	

DOCUMENT



```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digitalocean.com",
    "department": ["Finance", "Accounting"],
    "socialMediaAccounts": [
        {
            "type": "facebook",
            "username": "tom_william_johnson_23"
        },
        {
            "type": "twitter",
            "username": "@tomwilliamjohnson23"
        }
    ]
}
```

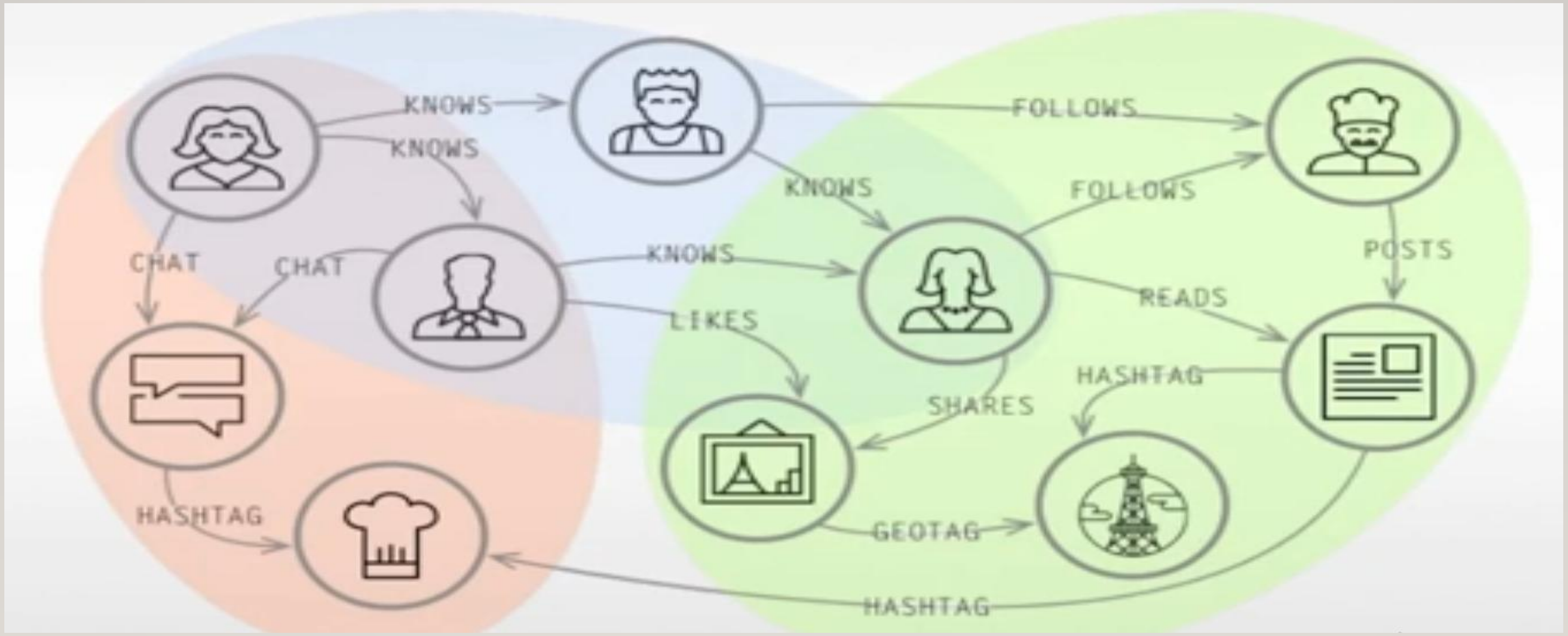


Document [NoSQL] = Record [SQL]
Group of document (Collection)[NoSQL] = Table [SQL]

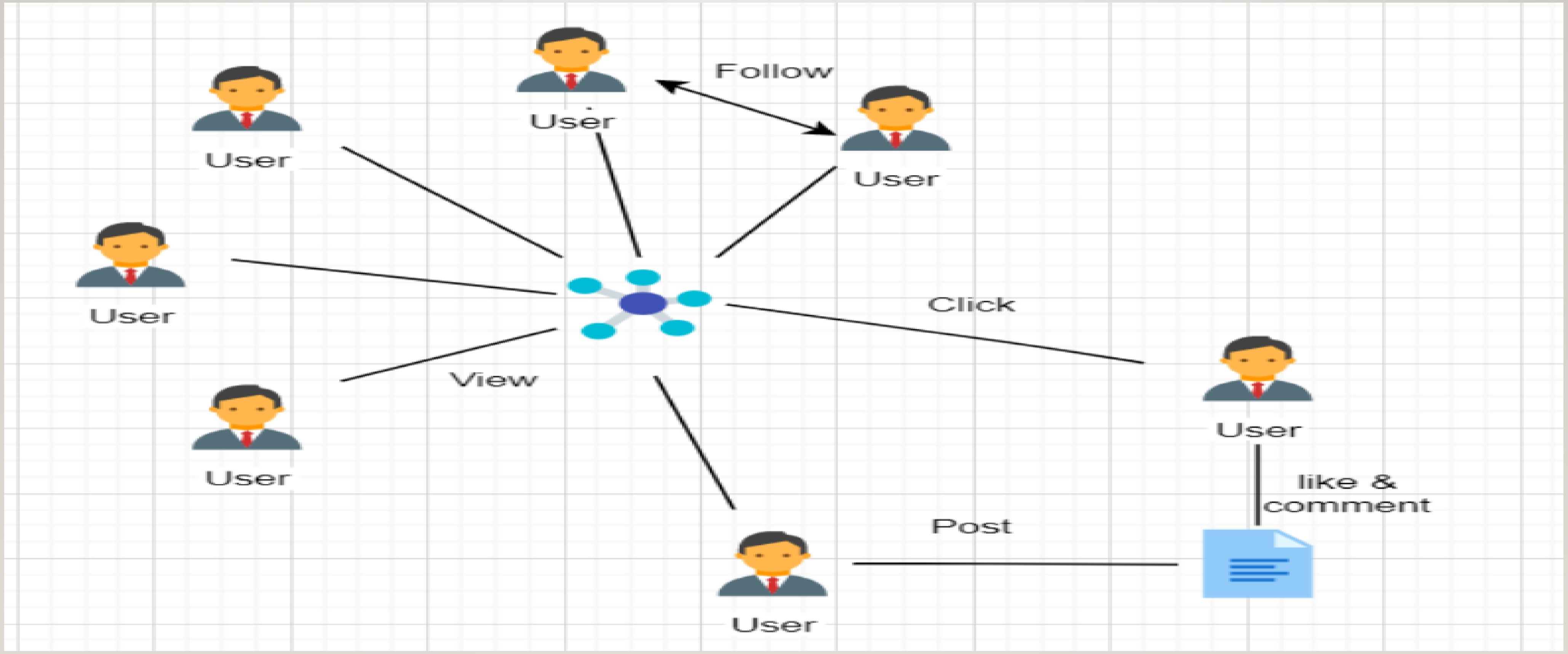
DOCUMENT STORE

Advantages	**Disadvantages**
Grouping **Document** into **Collection** Every thing is Object which **no** Structure needed Ex: Student has {Name, Age} other has {Name, Email}.	Each student has Track key as text Student1:Track :OS" Student2:Track :OS" No Relational as SQL **Redundant data.**
DB is Object and Programming Language is Object Which facilitate work.	

GRAPH


KNOWS
KNOWS
FOLLOWS
KNOWS
FOLLOWS
CHAT
CHAT
KNOWS
POSTS
READS
LIKES
HASHTAG
SHARES
GEOTAG
HASHTAG
HASHTAG

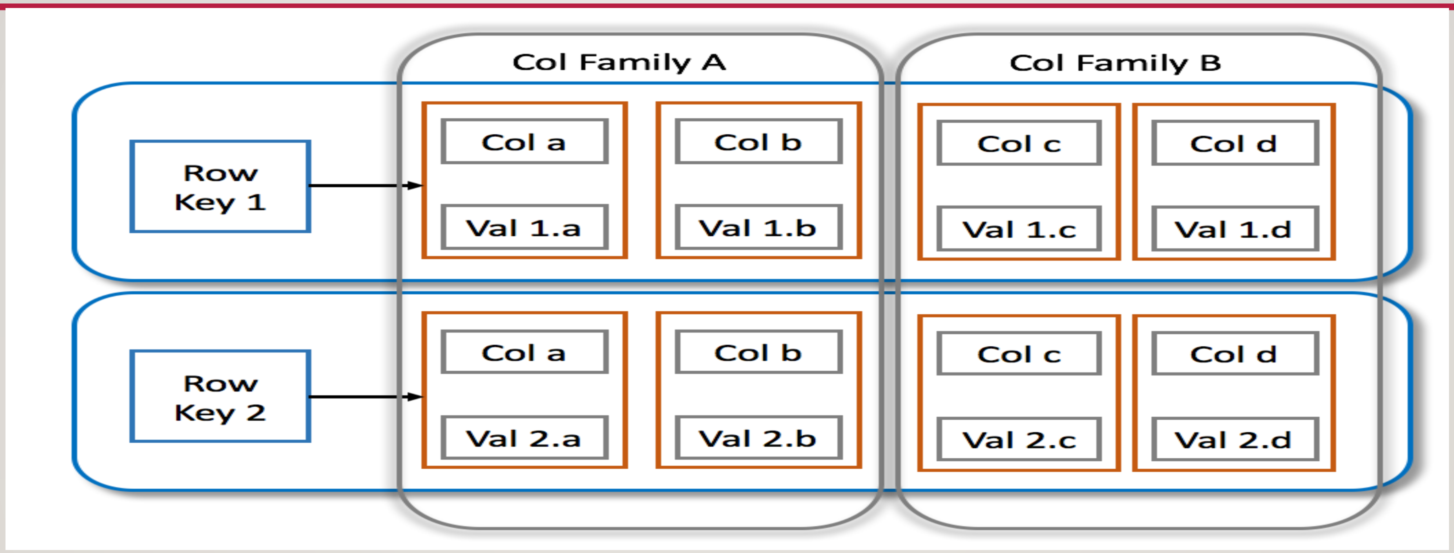
GRAPH


User
Follow
User
User
User
Click
User
View
User
like &
comment
Post
User

GRAPH

Advantages	**Disadvantages**
SQL **Relational not Actually** Registered into Database [FK] Relational **combined** into Runtime. [Cost] NOSQL Graph : Relation Register into Database [Good **Performance in retrieve huge data**]	Difficult to scale, as designed as one-tier architecture
Usage : Liking in Social Network, Friendship : Register data inside this feature.	No uniform query language
Flexible and agile structures	

COLUMN FAMILY



RowKey	Column Values	
1234	ph:cell=9867	email:1=x@abc
3678	social:twitter=#bigtable	ph:home=1234
5987	email:2=y@wqa	social:facebook=a@fb.com

WHY NOSQL ?

- Problems ?

  -Huge data

  -Backup

  - Application is Complex

  -High Availability Database

- Fix SQL
  - Expand Scale
  - Clustered [High Cost] [SQL Admin]

- Fix NOSQL [Comes to fix this problems]
  - Built-in feature
    - Divide tables into more than one cluster **[mongodb sharding]**
    - Replication

WHY NOSQL ? **[3V , 1C]**

Big Data is one of the key **forces** driving the growth and popularity of NoSQL for business.

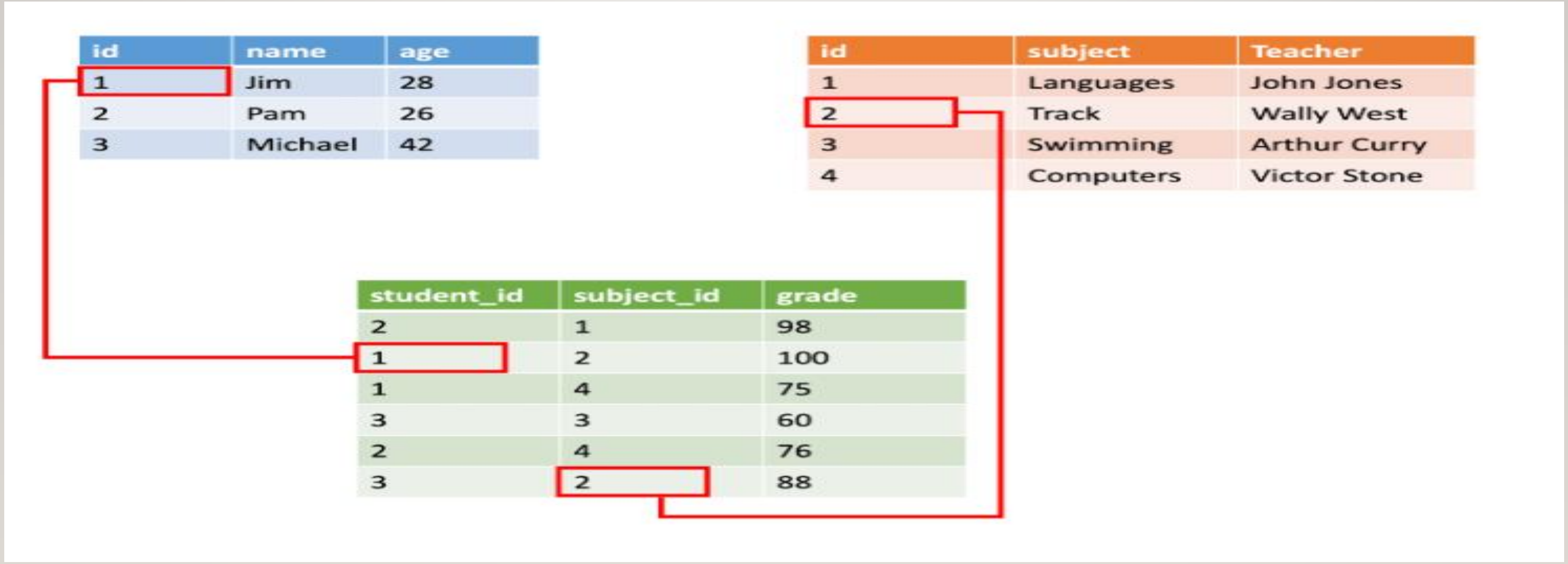
A Big Data project is normally typified by:

- **High data velocity**: lots of data coming in very quickly, possibly from different locations.
  - Writes and Read like Facebook , IOT
- **Data variety**: storage of data that is structured, semi-structured and unstructured.
- **Data volume**: data that involves many terabytes or petabytes in size.
  - SQL retrieval decrees when Data is highly increase.
- **Data complexity**: data that is stored and managed in different locations or data centers.
  - More than one node [Cluster] on different servers.

STRUCTURED, SEMI-STRUCTURED AND UNSTRUCTURED

- **Structured data** is generally tabular data that is represented by columns and rows in a database.
- Databases that hold tables in this form are called ***relational databases***.
- The mathematical term "*relation*" specify to a formed set of data held as a table.
- In structured data, all row in a table has the same set of columns.
- SQL (Structured Query Language) programming language used for structured data.

STRUCTURED DATA



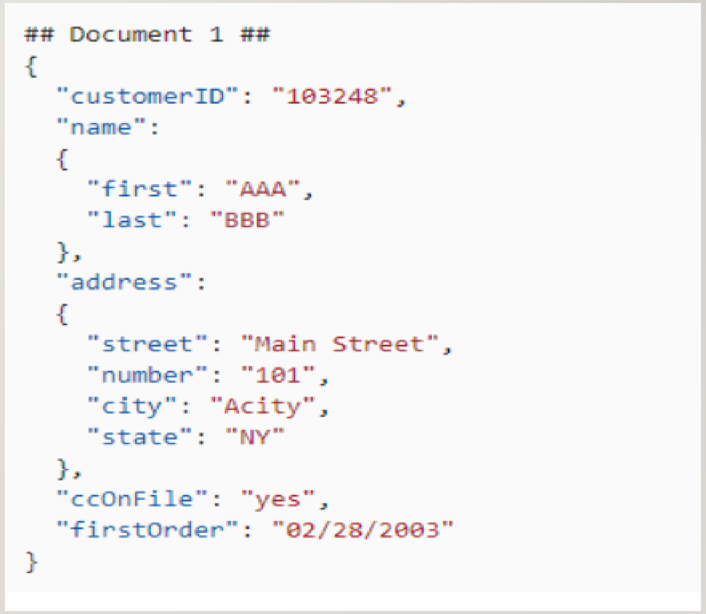
id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject	Teacher
1	Languages	John Jones
2	Track	Wally West
3	Swimming	Arthur Curry
4	Computers	Victor Stone

student_id	subject_id	grade
2	1	98
1	2	100
1	4	75
3	3	60
2	4	76
3	2	88

SEMI-STRUCTURED

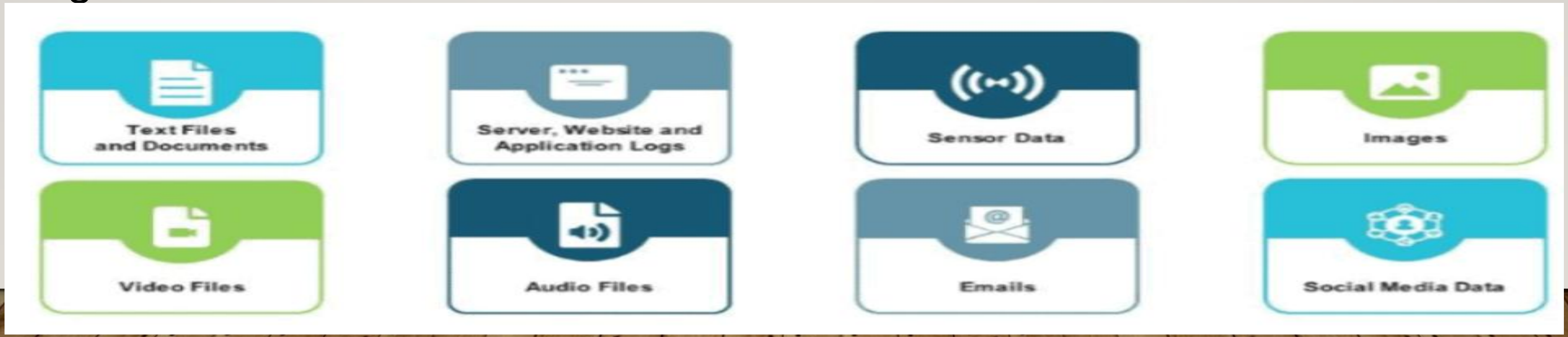
- **Semi-structured** data is information that doesn't consist of Structured data (relational database) but still has some structure to it.
- Semi-structured data consist of documents held in
- ***JavaScript Object Notation*** **(JSON) format**.
- It also includes ***key-value*** stores and ***graph*** databases.



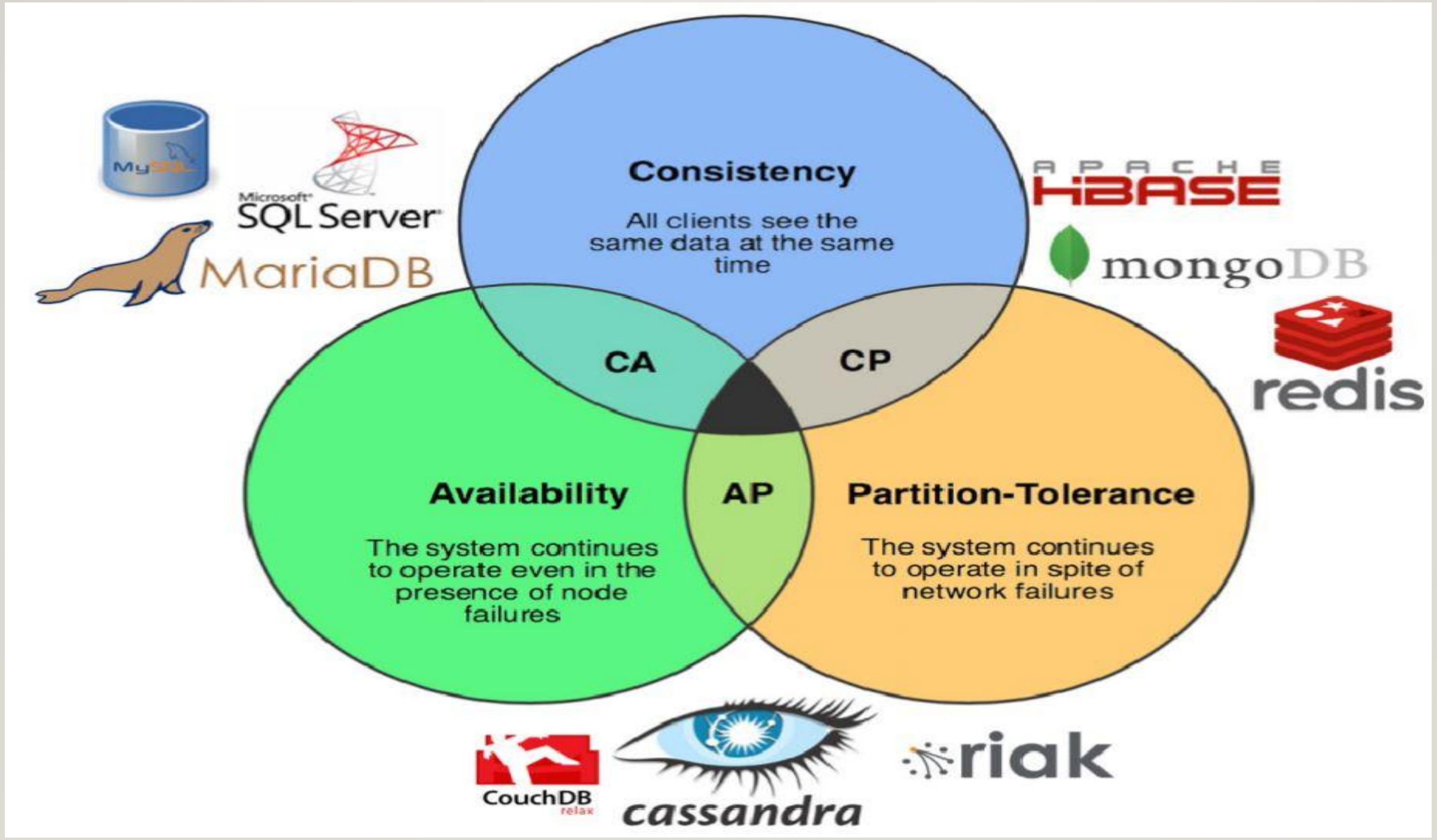
```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

UNSTRUCTURED DATA

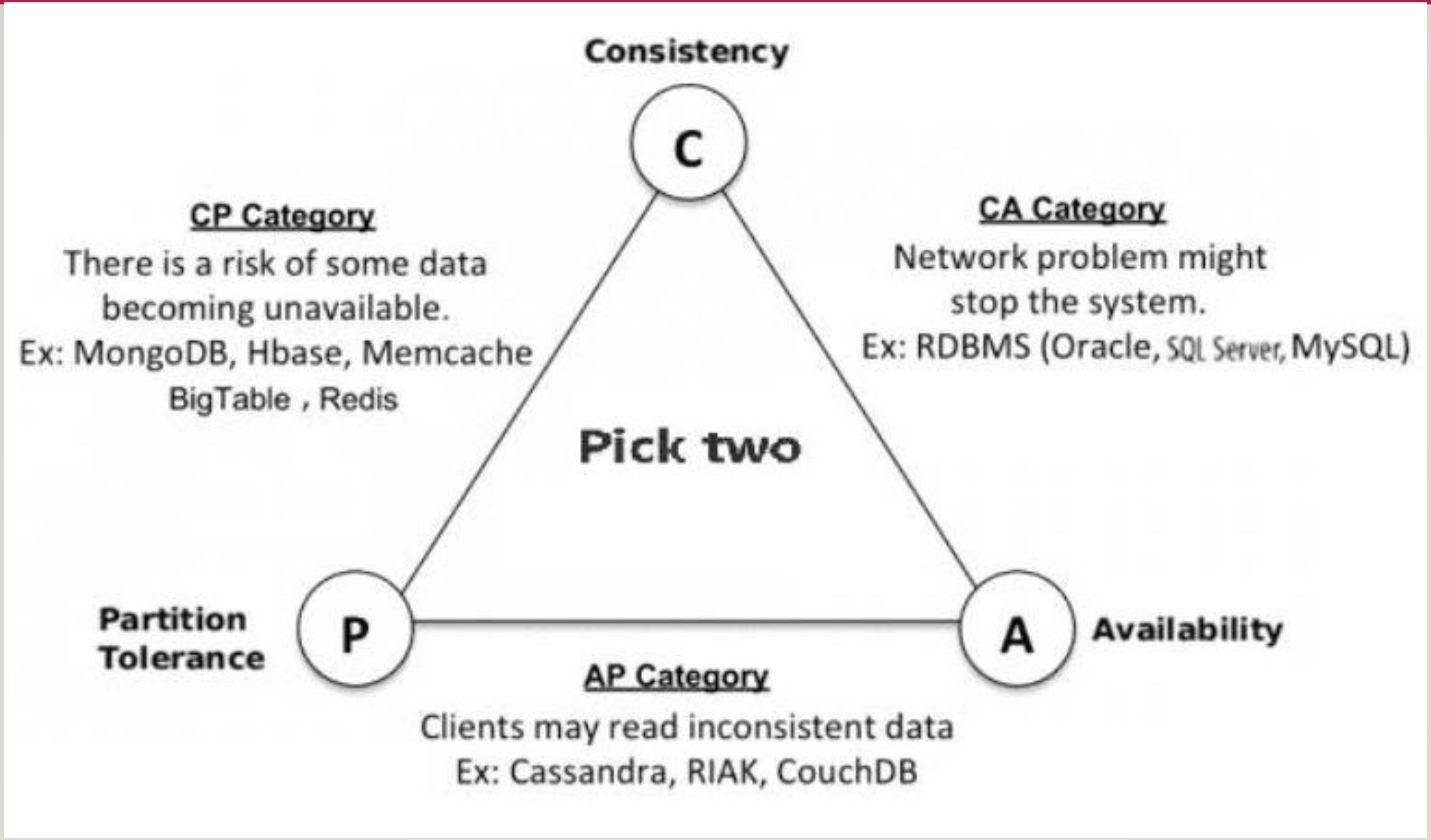
- **Unstructured data** is information that either does not organize in a pre-defined manner or not have a pre-defined data model.
- Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well.
- **Videos, audio, and binary** data files might not have a specific structure. They're assigned to as **unstructured** data.



CAP THEORY


Consistency
All clients see the same data at the same time
CA
CP
AP
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
MySQL
Microsoft SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB relax
cassandra
riak

CAP THEORY (CP)


Consistency
C
CP Category
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache BigTable , Redis
CA Category
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
AP Category
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

CAR THEORY

SCENARIOS WHERE NOSQL **SHOULD** BE USED

- Your **relational database will not scale** to your traffic at an acceptable cost.
- In a NoSQL database, there is **no fixed schema and no joins**. NoSQL can take advantage of "scaling out". Scaling out refers to spreading the load over many commodity systems.
- It's useful for creating **prototypes** or fast applications as it provides a tool to **develop new features easily.**
- You have local data transactions which do not have to be very durable. e.g. "**liking**“

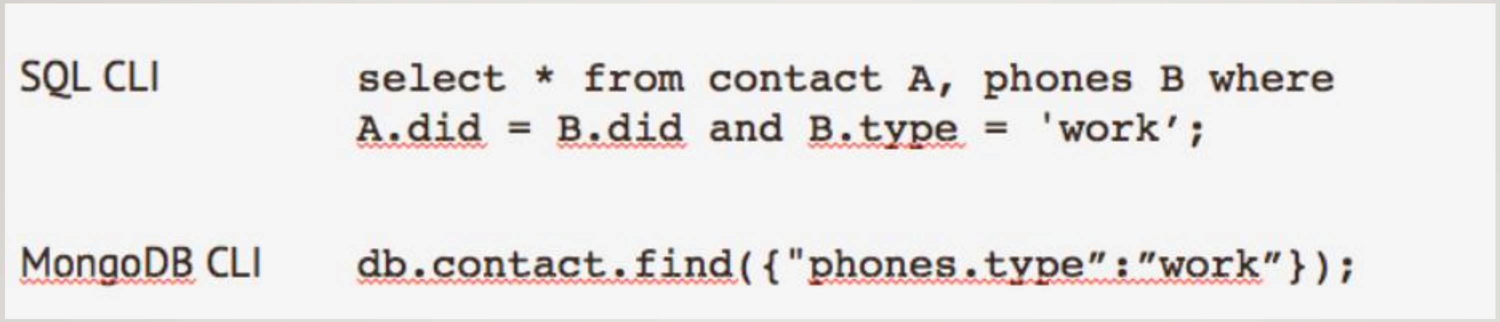
items on websites.

- **Agile sprints, quick iteration**, and frequent code pushes
- **Object-oriented programming** that is easy to use and flexible

SCENARIOS WHERE NOSQL SHOULD **NOT** BE USED:

- It cannot necessarily guarantee the **ACID**(Atomicity ,Consistency, Isolation, Durability)properties for your transactions.
- Normally an interface is provided for storing your data. Do not try to use a complicated query in that interface.
- The developer should always keep in mind that NoSQL database is not built on tables and usually doesn't use structured query language.
- If consistency is mandatory and there will be no drastic changes in terms of the data volume.

SQL VS MONGODB CODE

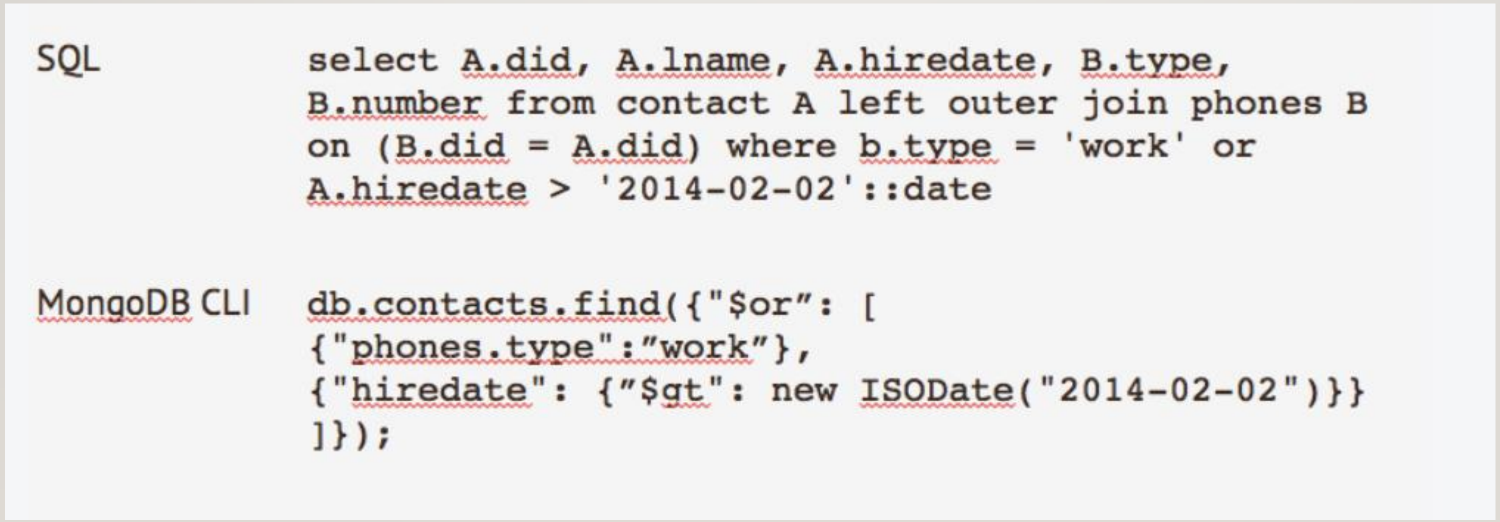


SQL CLI

```
select * from contact A, phones B where
A.did = B.did and B.type = 'work';
```

MongoDB CLI

```
db.contact.find({"phones.type”:”work”});
```



SQL

```
select A.did, A.lname, A.hiredate, B.type,
B.number from contact A left outer join phones B
on (B.did = A.did) where b.type = 'work' or
A.hiredate > '2014-02-02'::date
```

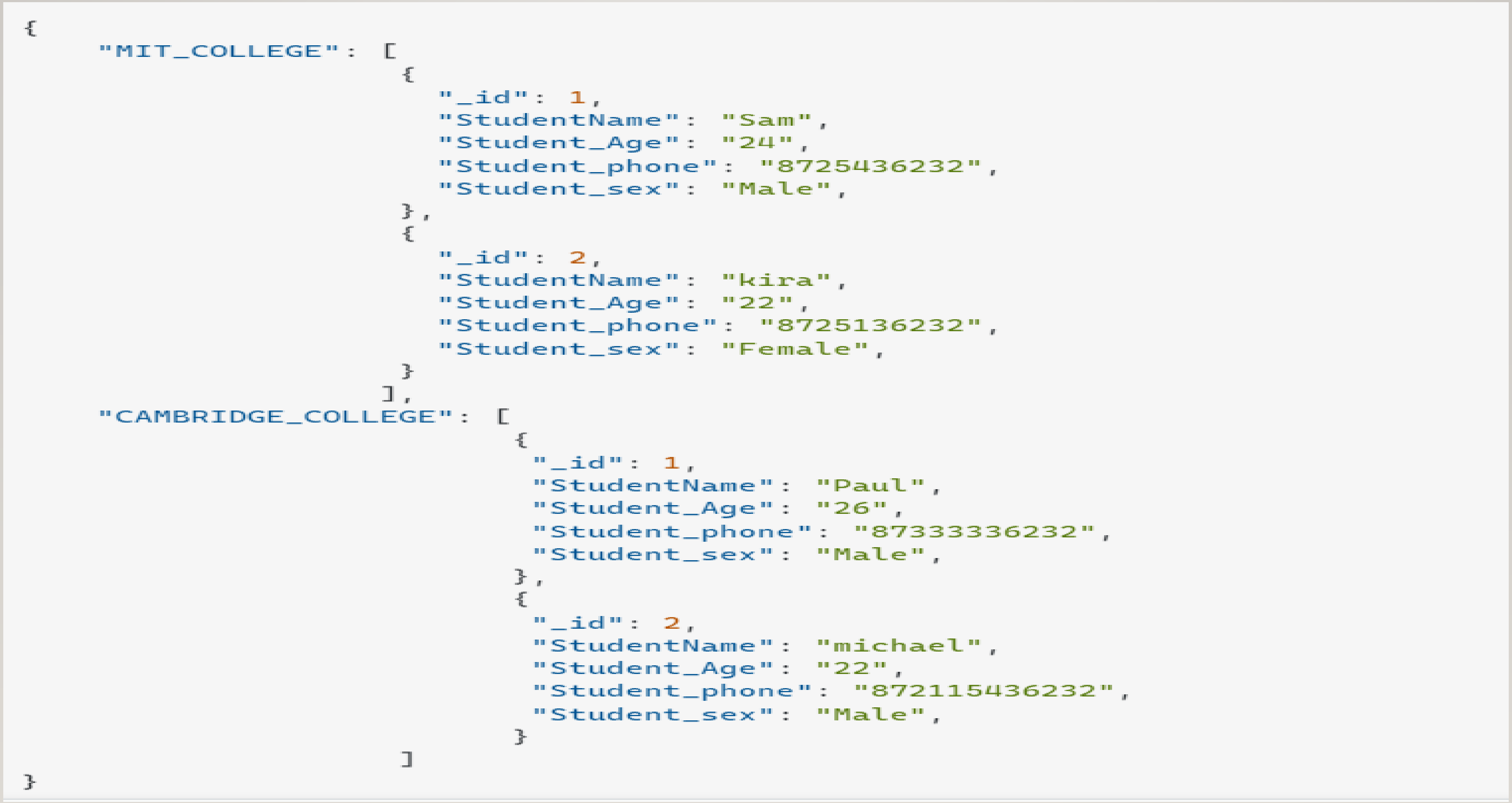
MongoDB CLI

```
db.contacts.find({"$or”: [
{"phones.type":”work”},
{"hiredate": {”$gt": new ISODate("2014-02-02")}}
]});
```

MONGO BASED ON JSON

- JSON : JavaScript Object Notation
- {"**name**":"John", "**age**":30, "**city**":"New York"}

- Mongo Based On **BSON** (Binary JSON) to save date and time , text.
- To Test your JSON is Valid you can use :
  - http://jsonviewer.stack.hu/



```
{
    "MIT_COLLEGE": [
                    {
                      "_id": 1,
                      "StudentName": "Sam",
                      "Student_Age": "24",
                      "Student_phone": "8725436232",
                      "Student_sex": "Male",
                    },
                    {
                      "_id": 2,
                      "StudentName": "kira",
                      "Student_Age": "22",
                      "Student_phone": "8725136232",
                      "Student_sex": "Female",
                    }
                   ],
    "CAMBRIDGE_COLLEGE": [
                          {
                           "_id": 1,
                           "StudentName": "Paul",
                           "Student_Age": "26",
                           "Student_phone": "87333336232",
                           "Student_sex": "Male",
                          },
                          {
                           "_id": 2,
                           "StudentName": "michael",
                           "Student_Age": "22",
                           "Student_phone": "872115436232",
                           "Student_sex": "Male",
                          }
                    ]
}
```

SQL to MongoDB Mapping Chart

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	$lookup, embedded documents
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.
SELECT INTO NEW_TABLE	$out See the SQL to Aggregation Mapping Chart.
MERGE INTO TABLE	$merge (Available starting in MongoDB 4.2) See the SQL to Aggregation Mapping Chart.
UNION ALL	$unionWith (Available starting in MongoDB 4.4)
transactions	transactions

MONGO OBJECTID

- Returns a new ObjectId. The 12-byte ObjectId consists of:
- A 4-byte timestamp, representing the ObjectId's ***creation, measured in seconds*** since the Unix epoch.
- A 5-byte random value generated once per process. This random value is unique to the ***machine*** and process.
- A 3-byte incrementing ***counter, initialized to a random value.***
- **ObjectId()**

Difference between Cassandra and MongoDB :

S.NO.	Cassandra	MongoDB
1.	Developed by Apache Software foundation and released on July 2008.	Developed by MongoDB Inc. and initially released on 11 February 2009.
2.	Cassandra is written only in Java language.	MongoDB is written in C++, Go, JavaScript, Python languages.
3.	Writing scalability in Cassandra is very high and efficient.	Writing scalability is limited in MongoDB
4.	Read performance is highly efficient in Cassandra as it takes O(1) time.	Read performance is not that fast in MongoDB when compared to Cassandra.
5.	Cassandra has only cursory support for secondary indexes i.e secondary indexing is restricted.	MongoDB does supports the concept of secondary indexes.

6.	Cassandra only supports **JSON** data format.	MongoDB supports both **JSON and BSON** data formats.
7.	The replication method that Cassandra supports is **Selectable Replication Factor**.	The replication method that MongoDB supports is **Master Slave Replication**
8.	Cassandra does not provides **ACID** transactions but can be tuned to support ACID properties.	MongoDB provides **Multi-document ACID transactions with snapshot isolation.**
9.	Server operating systems for Cassandra are **BSD, Linux, OS X, Windows**.	Server operating systems for MongoDB are **Solaris, Linux, OS X, Windows.**
10.	Famous companies like Hulu, **Instagram**, Intuit, **Netflix**, **IBM**, Reddit, etc uses Cassandra.	Famous companies like **Adobe**, Amadeus, Lyft, ViaVarejo, Craftbase, **Facebook**, etc uses MongoDB.

Difference between Cassandra and MongoDB : (**Similarities**)

- NOSQL
- Not ACID-Compliant
- Open-Source
- Cross-Platform

Difference between Cassandra and MongoDB : (**Differential**)

Cassandra	MongoDB
Structured	Unstructured
Similar to SQL	Based on JSON Formatting
Write-Heavy Loads	Read-Heavy Loads

DEMO

- **Check Database Version**:
  - db.version()
- **Display All Databases**:
  - show dbs
- **Insert New Documents (Single and Multiple):**
  - insertOne
  - insertMany
  - ObjectId()
- **Query to Find Data:**
  - Without Conditions
  - With Conditions
- **Update Document**
- **Delete Document**
- **Import Data**

THANK YOU

Any Question