Mahidol University
Wisdom of the Land

BIG DATA PROCESSING

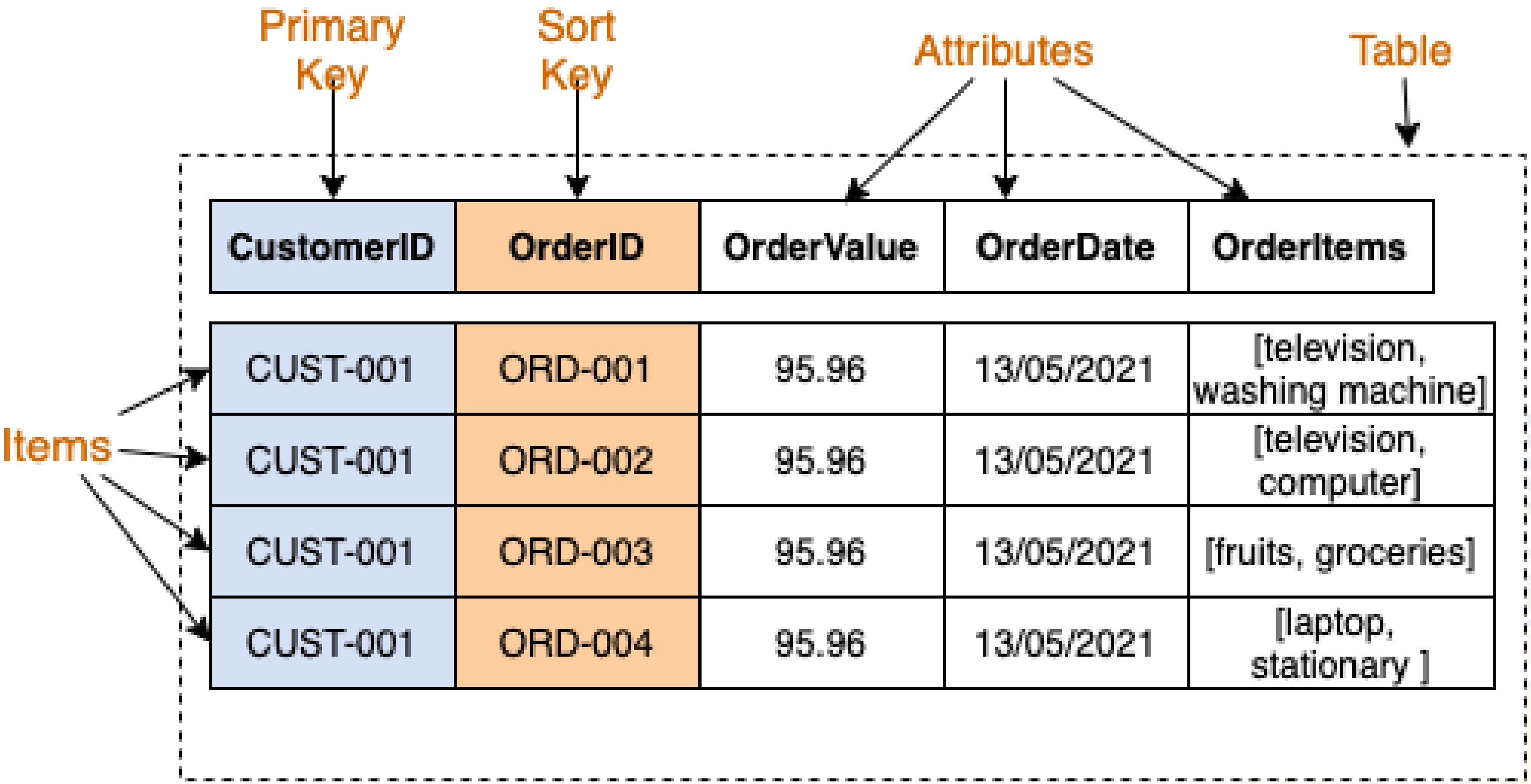
EGCI 466 – No SQL Databases

Key-value

Key-value

- Least Complex
- Each keys has it’s own value
- Ideal for simple Create Read Update and Delete(CRUD)
- Scale well (by key)

Key-value



CustomerID	OrderID	OrderValue	OrderDate	OrderItems
CUST-001	ORD-001	95.96	13/05/2021	[television, washing machine]
CUST-001	ORD-002	95.96	13/05/2021	[television, computer]
CUST-001	ORD-003	95.96	13/05/2021	[fruits, groceries]
CUST-001	ORD-004	95.96	13/05/2021	[laptop, stationary]

Key-value

- Quick CRUD (Create-Read-Update-Delete) operation
  - Stroring/retreiving for web app

- in-app user profiles
- Shopping cart only

Key-value: Not suitable for

- Interconnected data
  - Social Media
  - Recommendation system
- Very high consistency
- Require a lot of queries

EXAMPLE

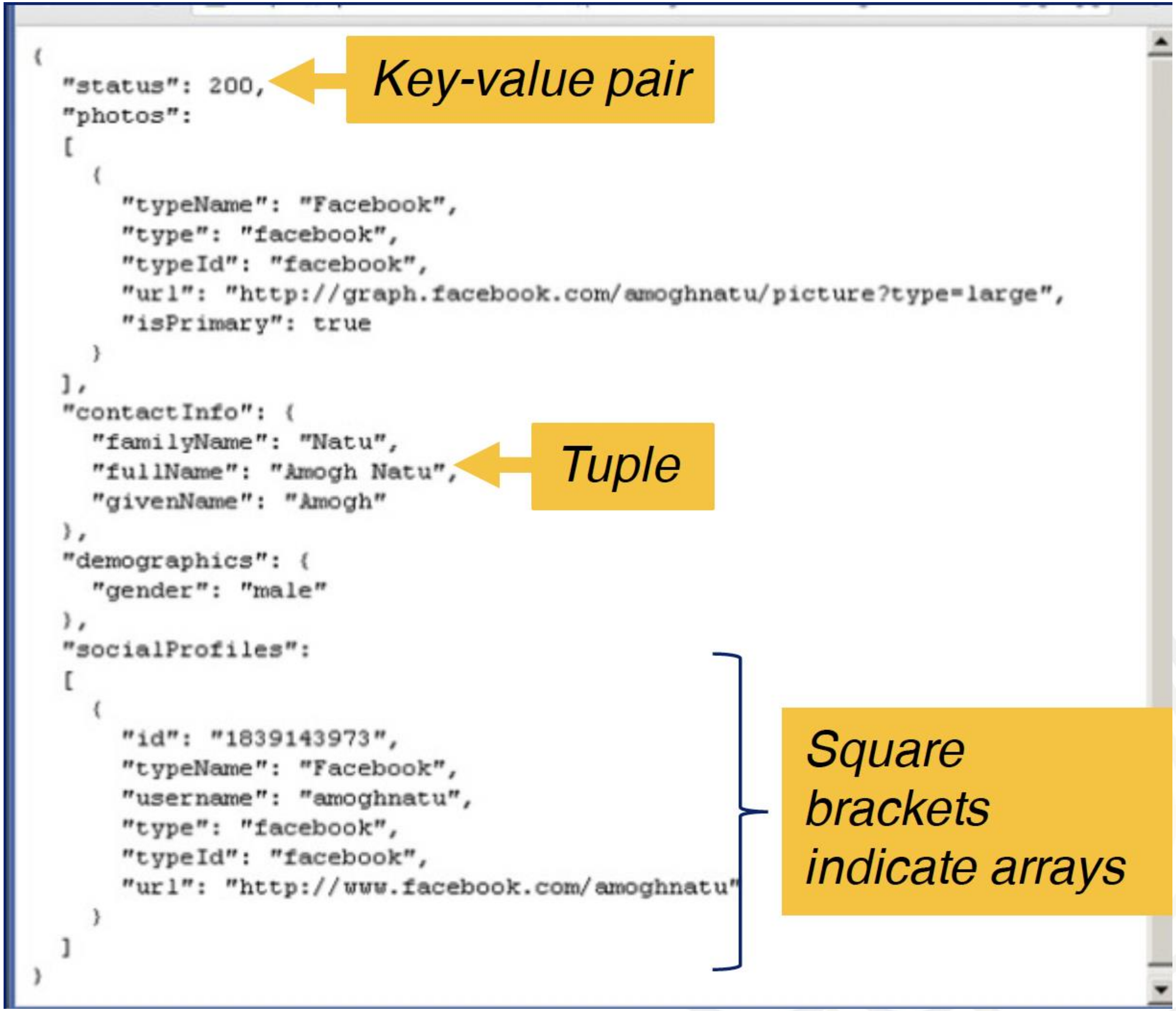
- dbm : unix system
- Sdbm
- GNU dbm
- Berkeley DB
- Amazon DynamoDB
    - Used By: Samsung Cloud, Zoom, Disney+, Nike
- Aerospike - An open-its ultra-low latency, reliability, and ability to handle large load.
- Redis - A multi-purpose database that also acts as memory cache and message broker.
- Riak - Made for developing apps, it works well with other databases and apps.

Document Based

Document-based

- Key-Values → value visible for query
- Each piece of data is considred a document
  - Typically JSON or XML format
- Each document offers a flexible schema
- Content of document ca be indexed and queried

JSON



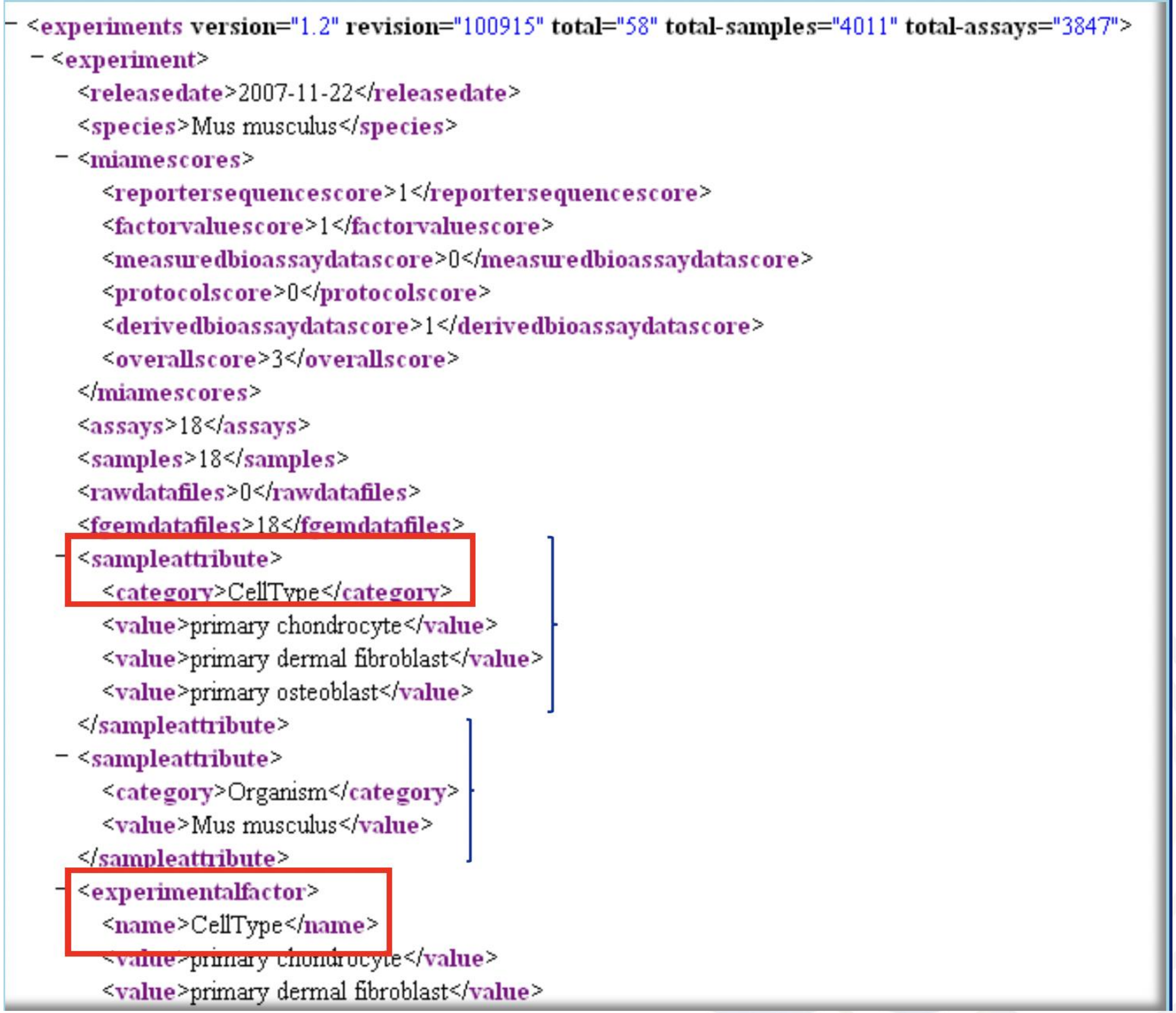
```
{
  "status": 200,
  "photos":
  [
    {
      "typeName": "Facebook",
      "type": "facebook",
      "typeId": "facebook",
      "url": "http://graph.facebook.com/amoghnatu/picture?type=large",
      "isPrimary": true
    }
  ],
  "contactInfo": {
    "familyName": "Natu",
    "fullName": "Amogh Natu",
    "givenName": "Amogh"
  },
  "demographics": {
    "gender": "male"
  },
  "socialProfiles":
  [
    {
      "id": "1839143973",
      "typeName": "Facebook",
      "username": "amoghnatu",
      "type": "facebook",
      "typeId": "facebook",
      "url": "http://www.facebook.com/amoghnatu"
    }
  ]
}
```

Key-value pair

Tuple

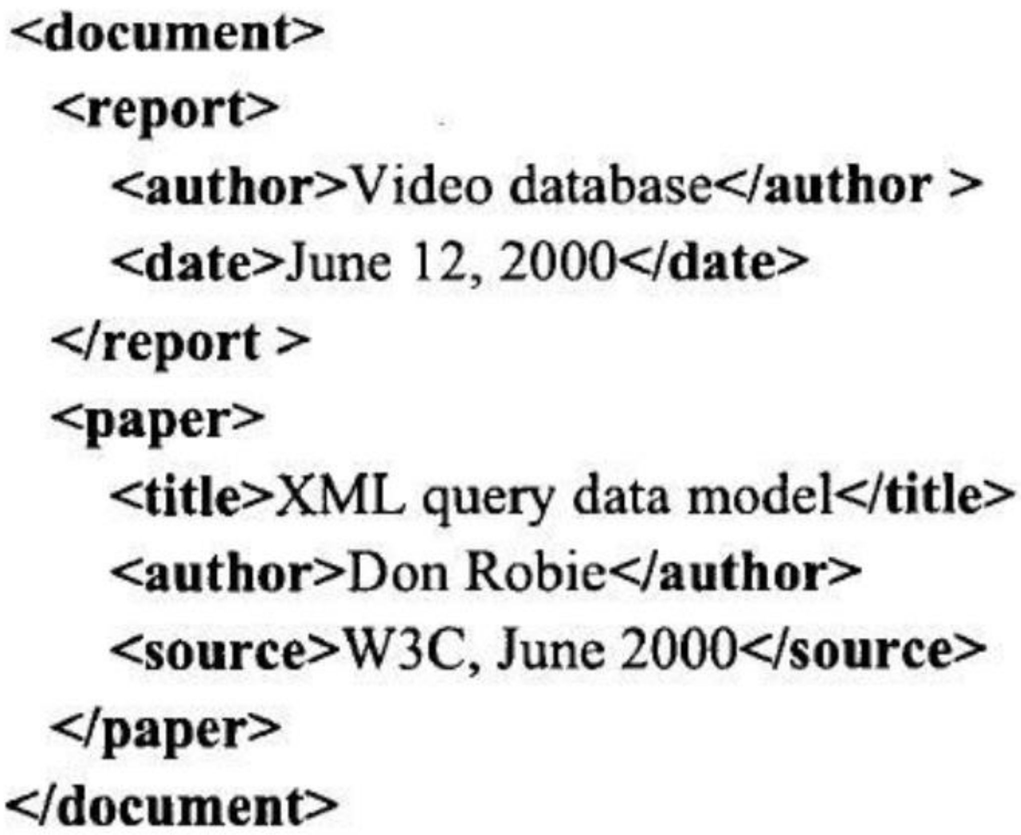
Square brackets indicate arrays

XML

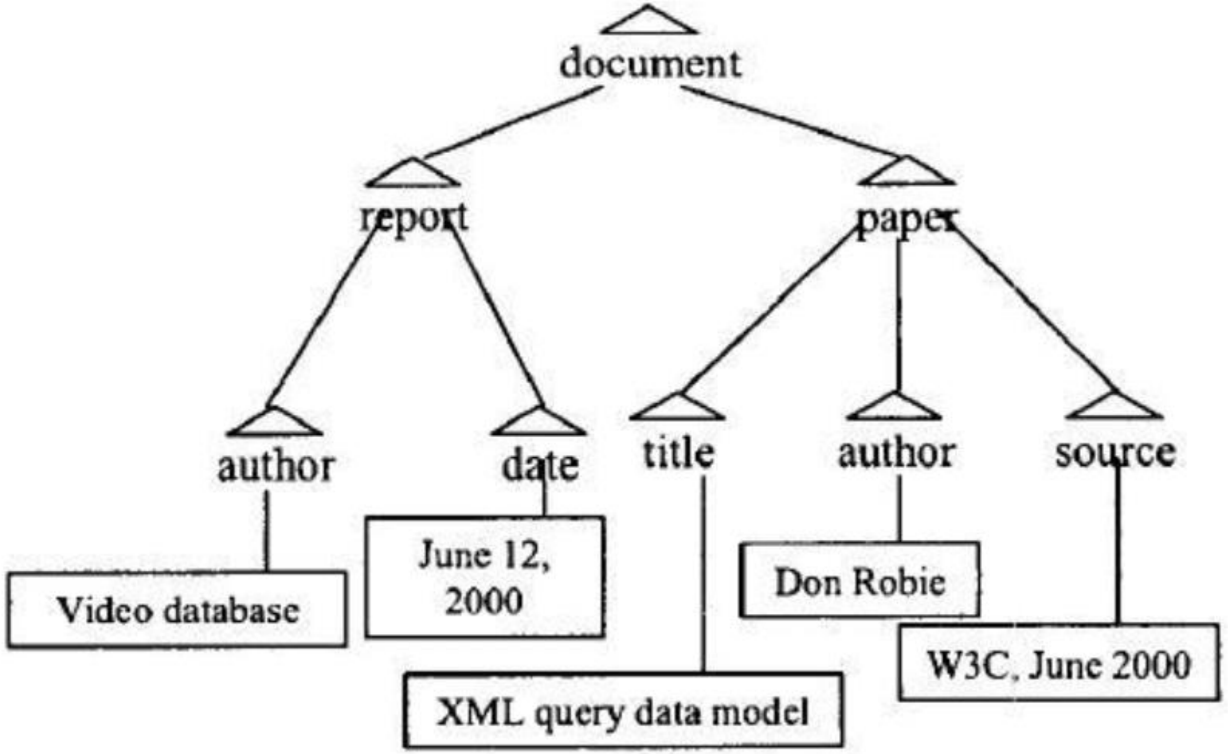


```
- <experiments version="1.2" revision="100915" total="58" total-samples="4011" total-assays="3847">
  - <experiment>
      <releasedate>2007-11-22</releasedate>
      <species>Mus musculus</species>
    - <miamescores>
        <reportersequencescore>1</reportersequencescore>
        <factorvaluescore>1</factorvaluescore>
        <measuredbioassaydatascore>0</measuredbioassaydatascore>
        <protocolscore>0</protocolscore>
        <derivedbioassaydatascore>1</derivedbioassaydatascore>
        <overallscore>3</overallscore>
      </miamescores>
      <assays>18</assays>
      <samples>18</samples>
      <rawdatafiles>0</rawdatafiles>
      <fgemdatafiles>18</fgemdatafiles>
    - <sampleattribute>
        <category>CellType</category>
        <value>primary chondrocyte</value>
        <value>primary dermal fibroblast</value>
        <value>primary osteoblast</value>
      </sampleattribute>
    - <sampleattribute>
        <category>Organism</category>
        <value>Mus musculus</value>
      </sampleattribute>
    - <experimentalfactor>
        <name>CellType</name>
        <value>primary chondrocyte</value>
        <value>primary dermal fibroblast</value>
```

Document-based Query Property

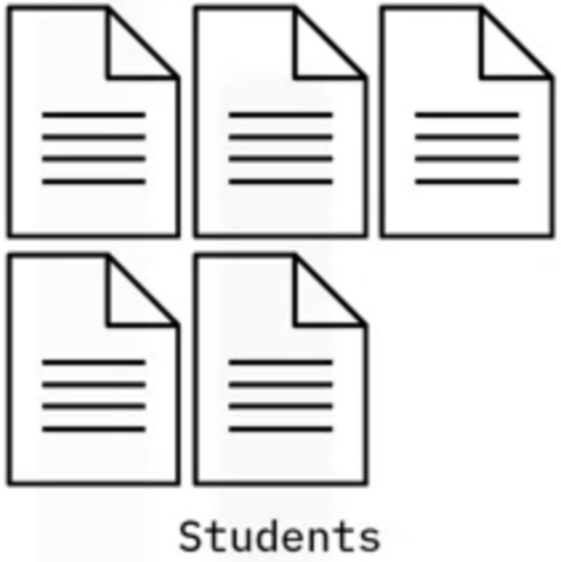


```
<document>
  <report>
    <author>Video database</author >
    <date>June 12, 2000</date>
  </report >
  <paper>
    <title>XML query data model</title>
    <author>Don Robie</author>
    <source>W3C, June 2000</source>
  </paper>
</document>
```



Collection

- A group of stored document
  - All student records → collection
  - Staff records in Employee section

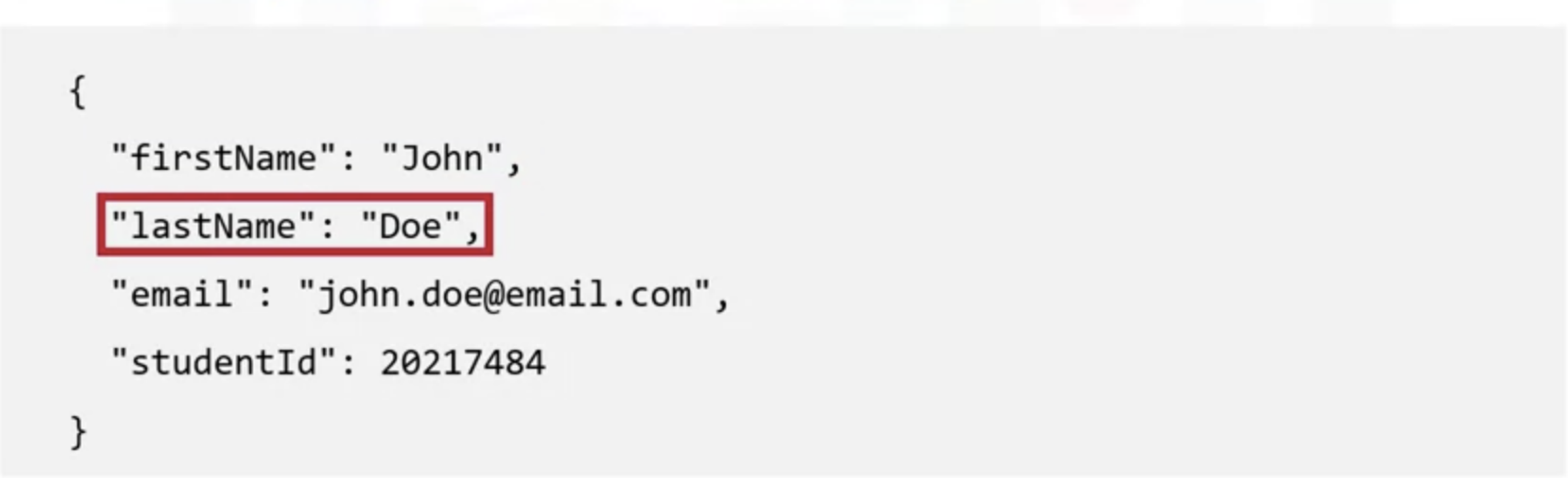


EXAMPLE: MONGO DB

- A document and a NoSQL database

- Document

Kev:value



```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@email.com",
  "studentId": 20217484
}
```

Mongo DB

- Get mongodb account → https://cloud.mongodb.com/

- **https://github.com/AjMing/BigData/blob/main/MongoDB_PyMongo_Example.ipynb**

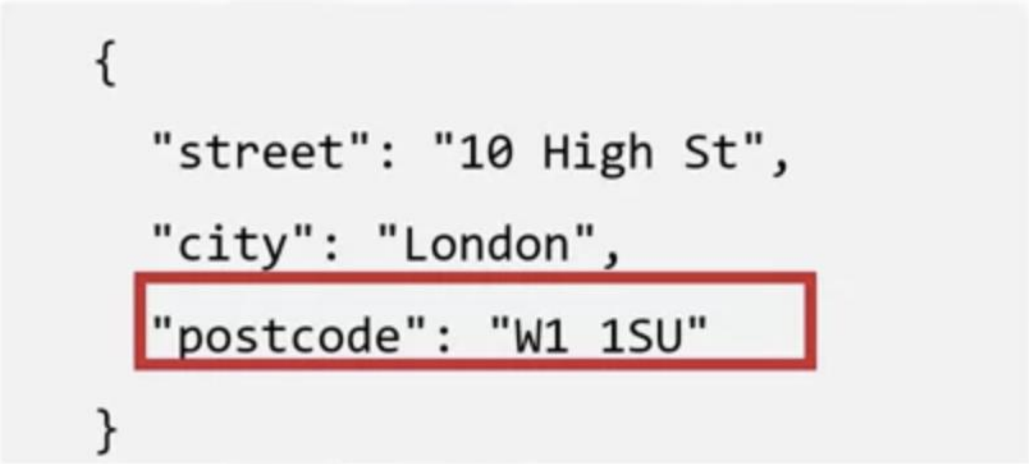
More reference:
https://www.cloudskillsboost.google/course_templates/731
(some are depreciated)

Document in Detail

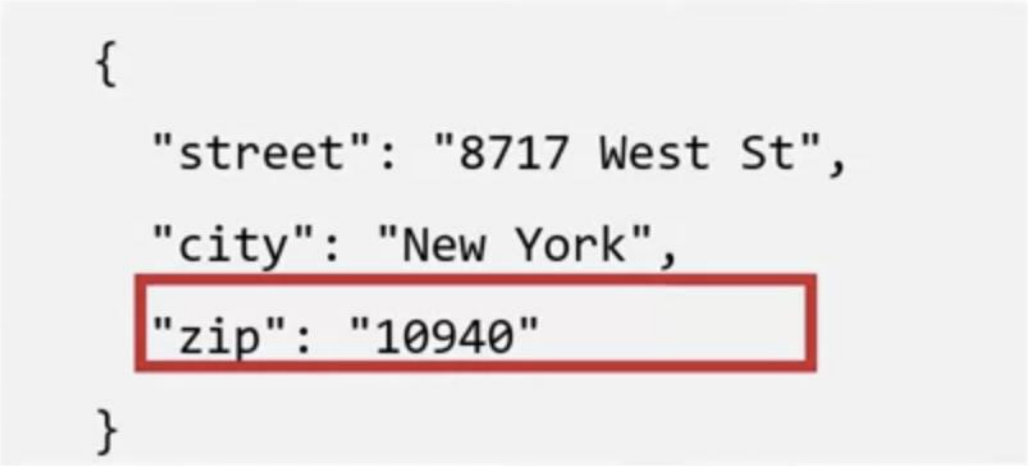
```
{
        “firstName”: “Jaidee”,
        “lastName”: “Deejung”,
        “email”: “Jaidee@gmail.com”,
        “studentId”: 6522111
}
```

```
{
    “firstName”: “Jaidee”,
    “lastName”: “Deejung”,
    “email”: “Jaidee@gmail.com”,
   “studentId”: 6522111
   ”address”:{
          ”province”:”Nakhon Pathom”,
          “district”:”Salaya”
    },
  “Interests”[“football”,”reading”,”café hopping”]
}
```

Flexible Schema



```
{
  "street": "10 High St",
  "city": "London",
  "postcode": "W1 1SU"
}
```

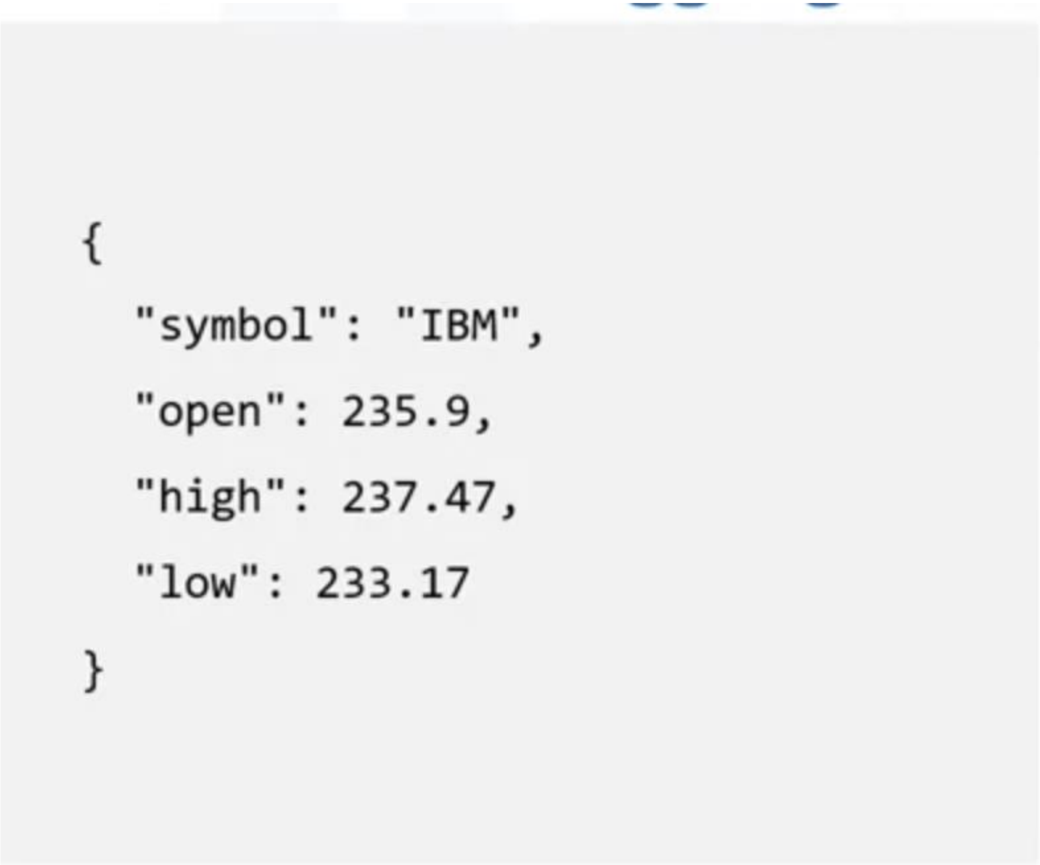


```
{
  "street": "8717 West St",
  "city": "New York",
  "zip": "10940"
}
```

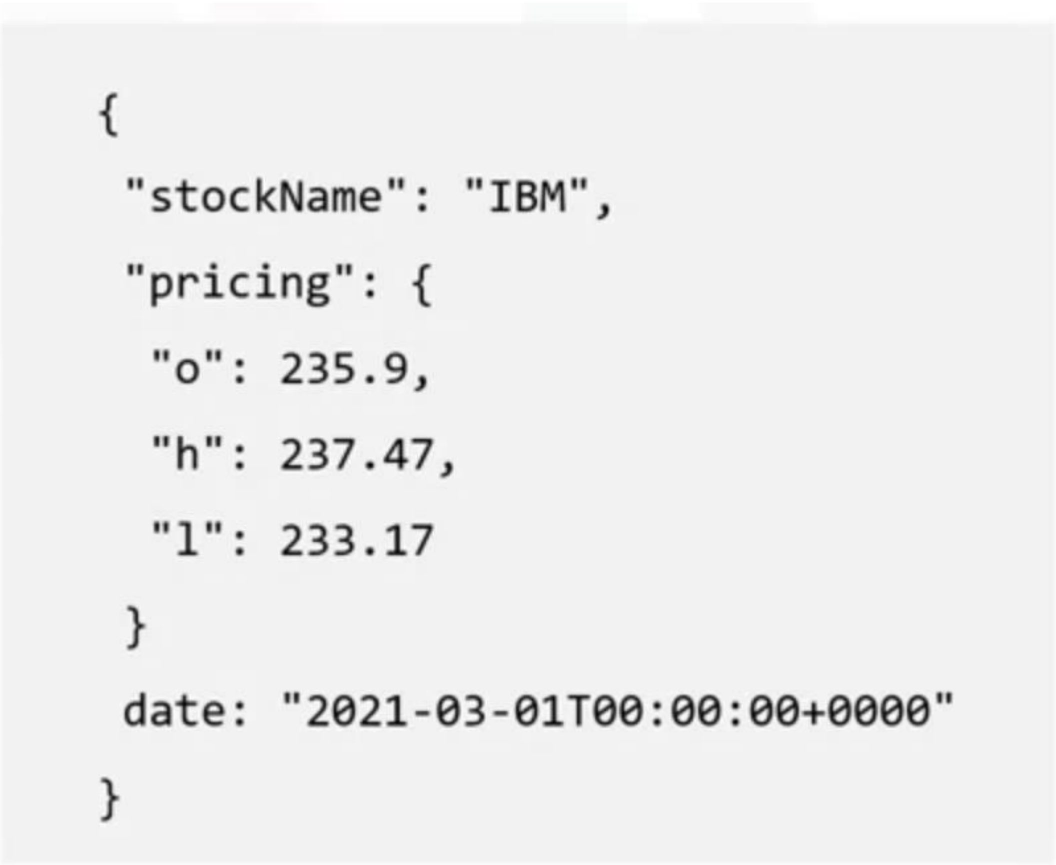
No Schema required

- In RDBMS
  - Create table
  - Insert into Table
- in MongoDB
  - Insert directly
  - easily add field in the database

Combine different type of data



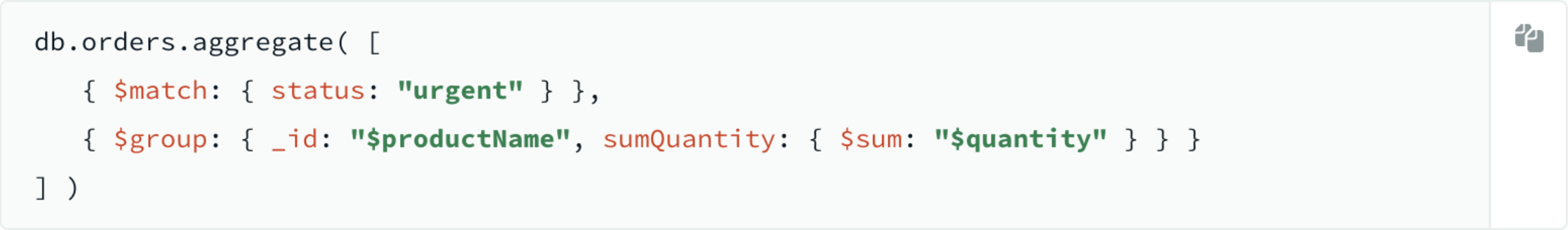
```
{
  "symbol": "IBM",
  "open": 235.9,
  "high": 237.47,
  "low": 233.17
}
```



```
{
 "stockName": "IBM",
 "pricing": {
  "o": 235.9,
  "h": 237.47,
  "l": 233.17
 }
 date: "2021-03-01T00:00:00+0000"
}
```

Query and Analytics

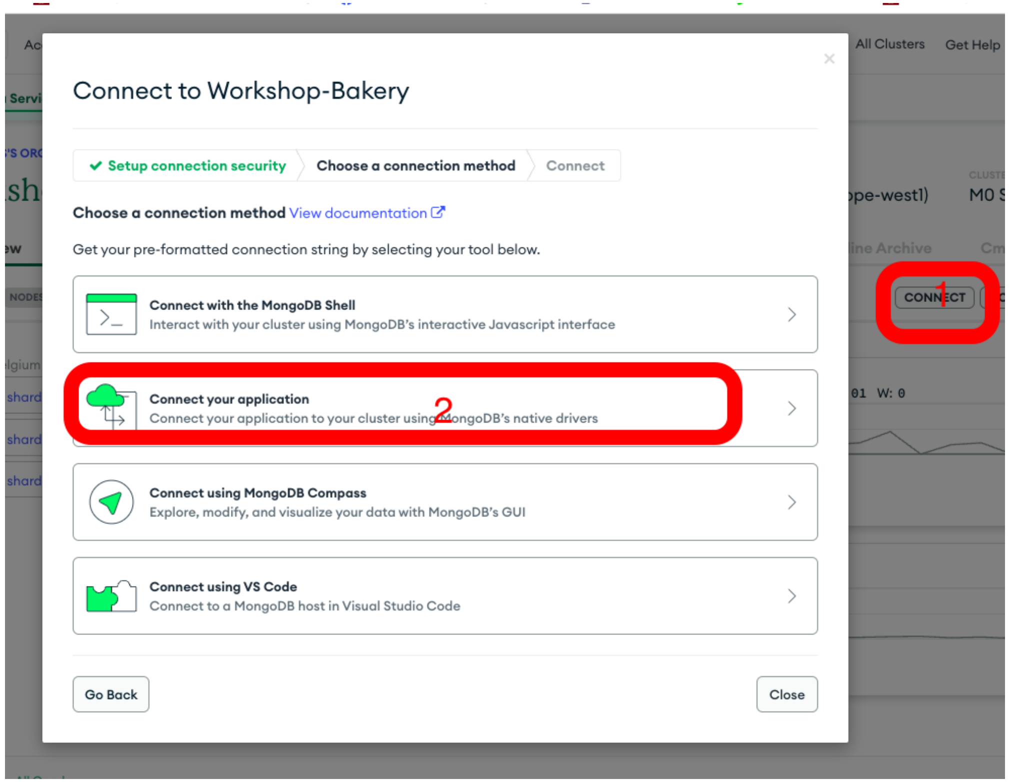
- MongoDb querying system MQL
  - wide range of operations



```
db.orders.aggregate( [
   { $match: { status: "urgent" } },
   { $group: { _id: "$productName", sumQuantity: { $sum: "$quantity" } } }
] )
```

EXAMPLE of pymongo

- Setting up



Connect to Workshop-Bakery

✔ Setup connection security | Choose a connection method | Connect

Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

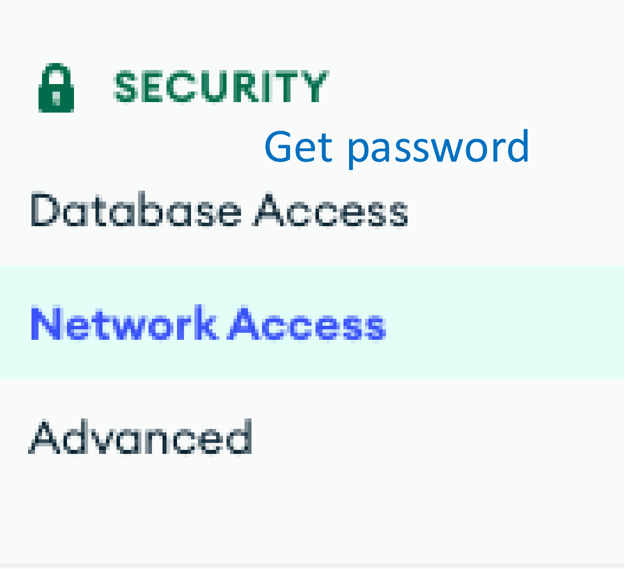
Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Connect using VS Code
Connect to a MongoDB host in Visual Studio Code

Go Back | Close

- white list (remove afterward)

allow all ip address



Get password

EXAMPLE of pymongo

- Pymongo in colab
- MongoDB with Big Query

CRUD operation

- Create
  - use sample
  - db.user.insertOne({“firstName”: “John”,
    “lastName”: “Doe”,
    “email”: “john.doe@mahidol.edu”,
    “id”: 6082111 }
)
    - db.user.insertMany(userlist)
- Read
  - user.findOne({“email”: john.doe@mahidol.edu})
  - user.find({“lastName”: Doe})
  - user.countDocuments({“lastName”: Doe})

CRUD operation

- Replace
  - user=db.user.findOne({”lastName”:”Doe”})
  - db.user[“onlineOnly”]=true
  - db.user[“email”]=“john.doe@mahidol.ac.th”
  - db.user.replaceOne({“lastName”:”Doe”},student)

Update

- change= {“$set”: {“onlineOnly”:true, email: johnd@campus.edu }}
- db.user.updateOne({“lastName”:”Doe”},changes)
- db.user.updateMany({ }, {“$set”: {“onlineOnly”:true}})

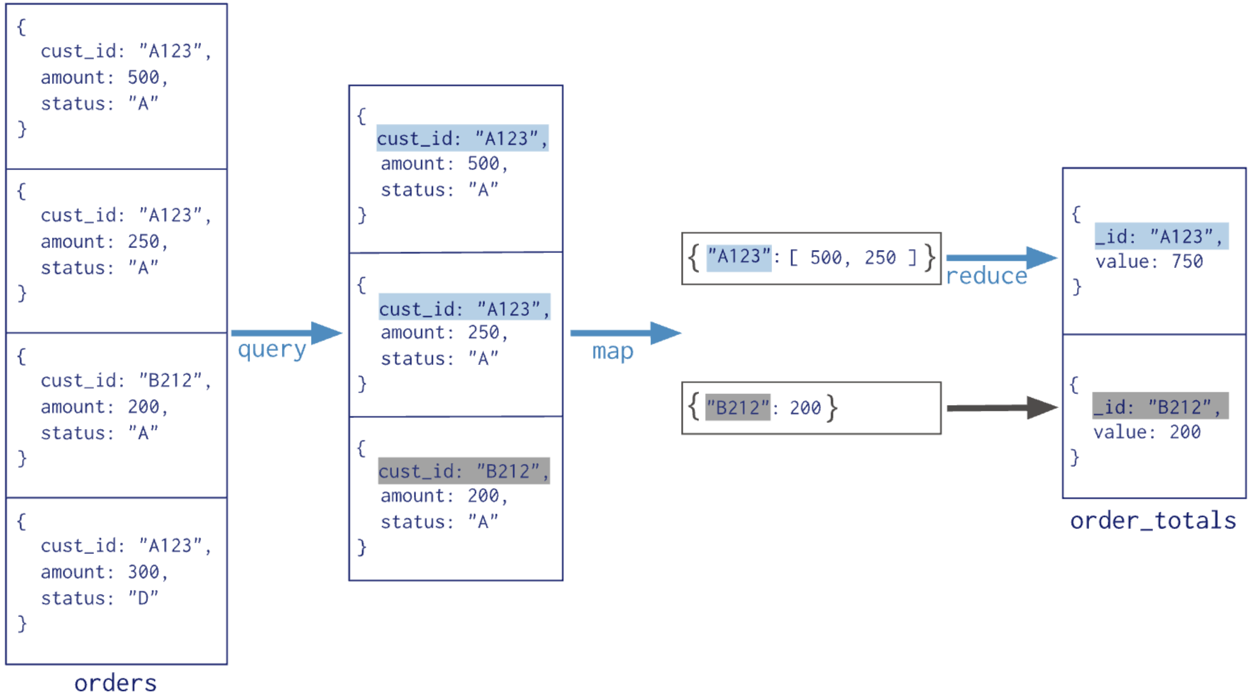
Delete

- db.user.deleteOne({“lastName”:”Doe”})

- db.user.deleteMany({“lastName”:”Doe”})

Map-reduce(deprecated)

```
   Collection
       ↓
db.orders.mapReduce(
          map     →  function() { emit( this.cust_id, this.amount ); },
          reduce  →  function(key, values) { return Array.sum( values ) },
                     {
          query   →    query: { status: "A" },
          output  →    out: "order_totals"
                     }
          )
```



```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

orders

query

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

map

```
{ "A123": [ 500, 250 ] }
{ "B212": 200 }
```

reduce

```
{
  _id: "A123",
  value: 750
}
{
  _id: "B212",
  value: 200
}
```

order_totals

High Availability

- It highly available
- Highly redunduncy
- Typically 3 node replica sets, 1 primilary

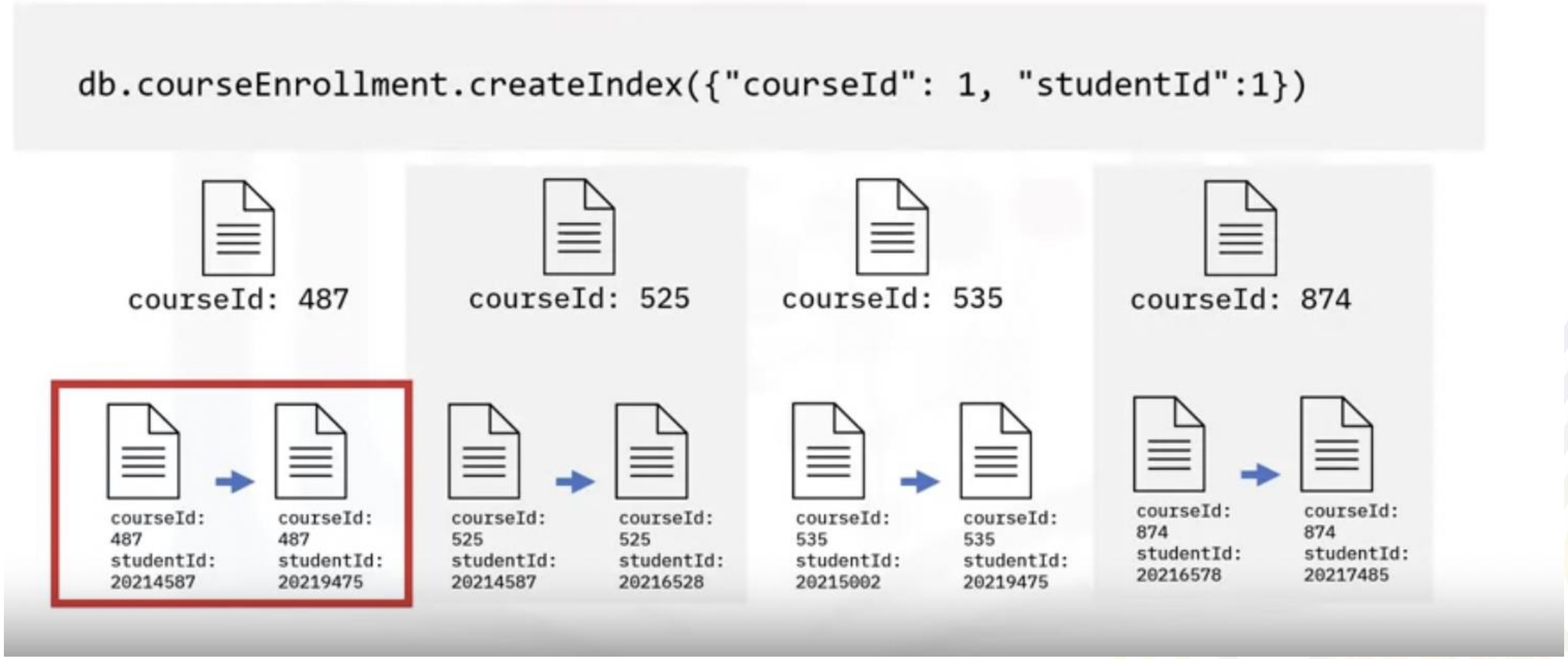
Indexes

- Quickly locate data without looking for it everywhere

- Create for frequent queries
  - db.courseEnrollment.createIndex({“courseId”:1)}
- For sorting
  - db.courseEnrollment.createIndex({“courseId”:1 , “studentId”:1)}

- Index. →location
  - using balance tree

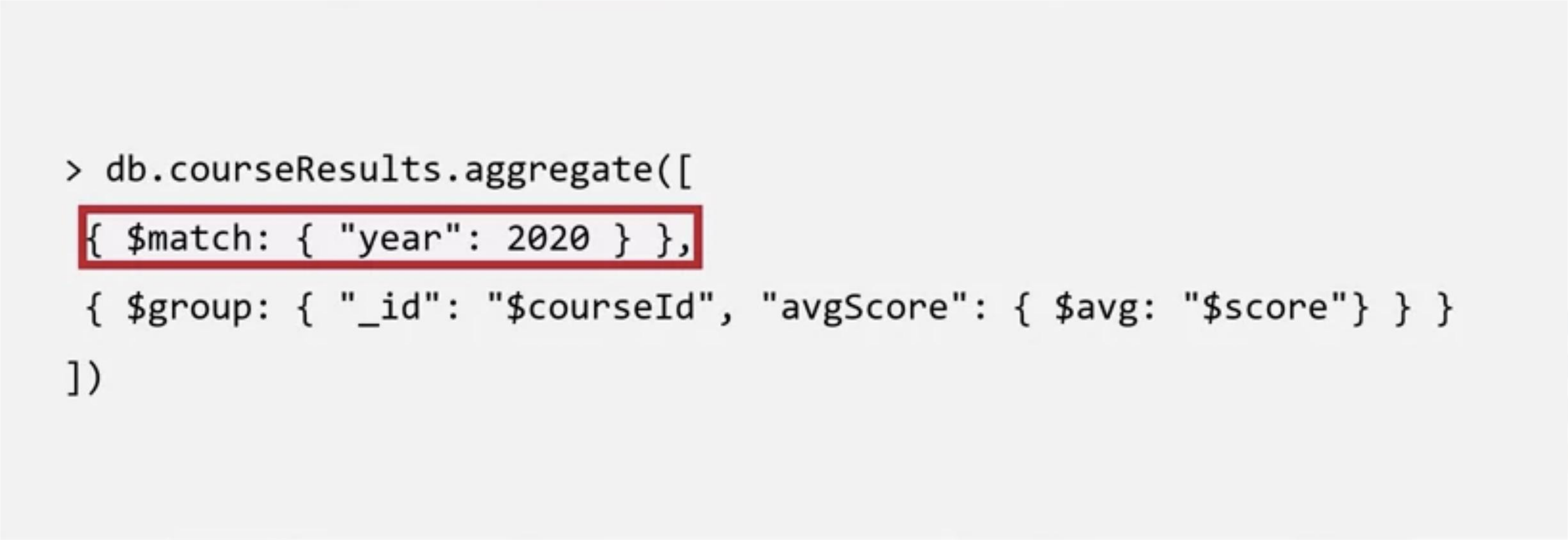
Tree index



```
db.courseEnrollment.createIndex({"courseId": 1, "studentId":1})
```

courseId: 487
courseId: 525
courseId: 535
courseId: 874
courseId: 487 studentId: 20214587
courseId: 487 studentId: 20219475
courseId: 525 studentId: 20214587
courseId: 525 studentId: 20216528
courseId: 535 studentId: 20215002
courseId: 535 studentId: 20219475
courseId: 874 studentId: 20216578
courseId: 874 studentId: 20217485

Aggregation Framework



```
> db.courseResults.aggregate([
 { $match: { "year": 2020 } },
 { $group: { "_id": "$courseId", "avgScore": { $avg: "$score"} } }
])
```

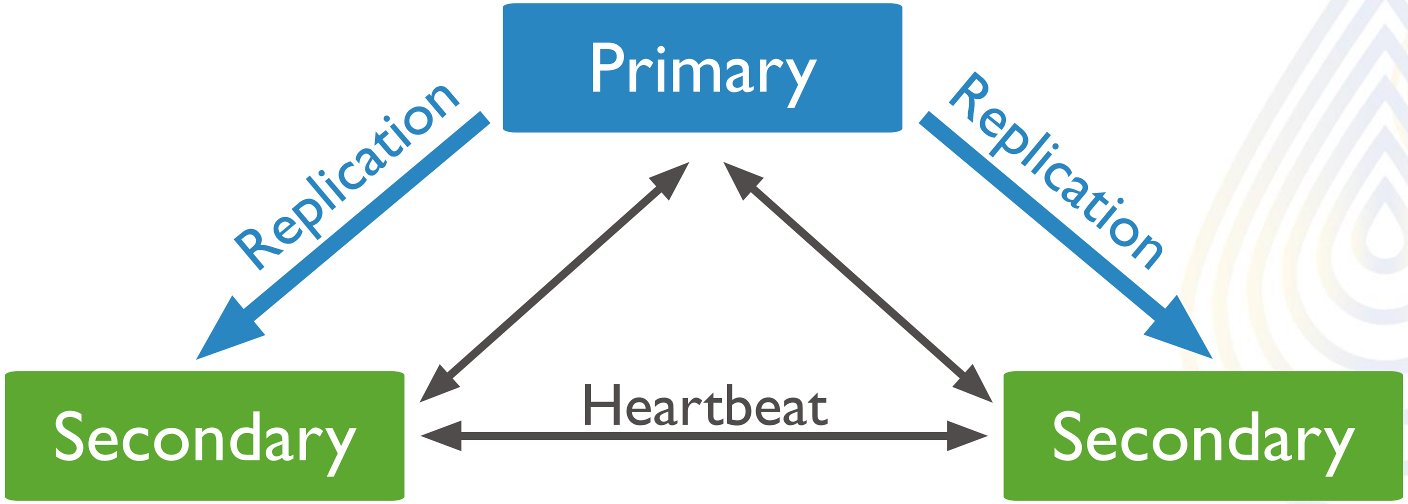
Aggregation stages

- $project
  - change the project
- $Sort

- $ Count
- $merge →into

Replcation & Sharding

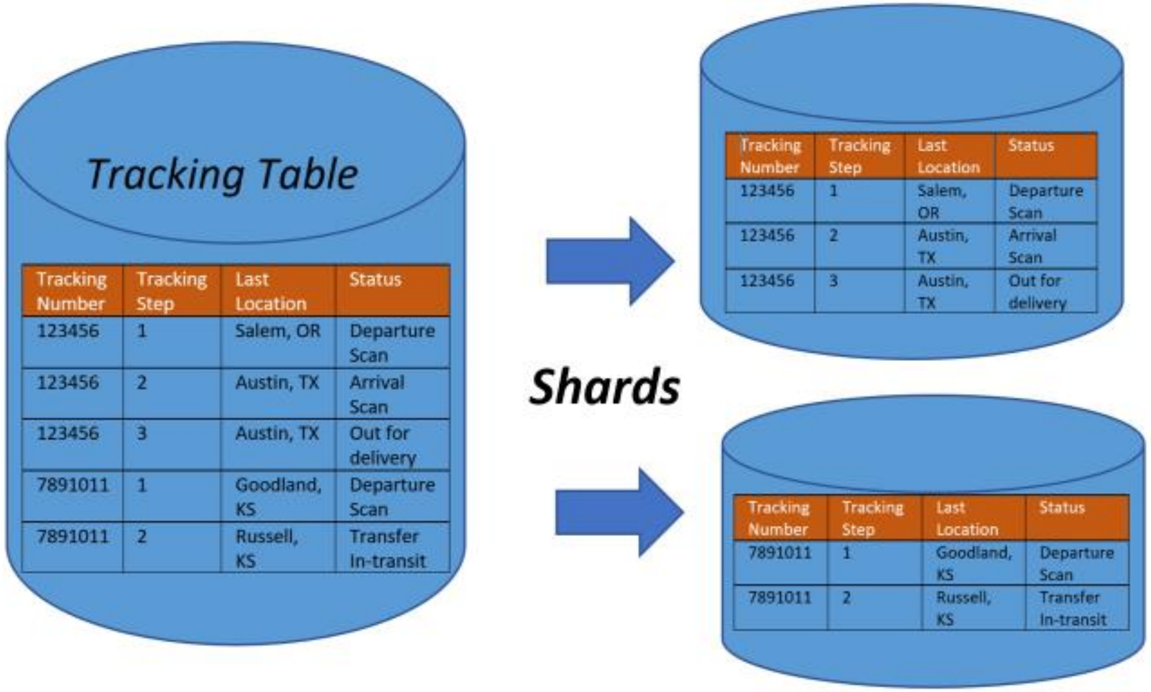
- A MongoDB cluster is made of data bearing nodes
- All nodes contain the same data
- Write is done on Primary node → Replicate to Secondary nodes



Benefits of Replication

- Redundancy
- High availability
- Fault tolerance
- Rely more on back up on stores

Sharding (Horizontal Scaling)



Tracking Table

Tracking Number	Tracking Step	Last Location	Status
123456	1	Salem, OR	Departure Scan
123456	2	Austin, TX	Arrival Scan
123456	3	Austin, TX	Out for delivery
7891011	1	Goodland, KS	Departure Scan
7891011	2	Russell, KS	Transfer In-transit

Shards

Tracking Number	Tracking Step	Last Location	Status
123456	1	Salem, OR	Departure Scan
123456	2	Austin, TX	Arrival Scan
123456	3	Austin, TX	Out for delivery

Tracking Number	Tracking Step	Last Location	Status
7891011	1	Goodland, KS	Departure Scan
7891011	2	Russell, KS	Transfer In-transit

- Increase your throughput by directing to only to relevant shards.
- Store more data that couldn’t fit on a single node.
- Split data across shards based on regions

Examples: IoT devices

- Different types of data
  - different sensors: temp sensor, wind sensor
  - Vast amout of data
  - Large scale

Examples: E-commerce

- Different attributes
  - Clothes: colour , size
  - Shoe: Colour, size
  - Phone: storage, network, color,brand
  - Book: Publisher, Author, pages, year, title

- Optimise for read
  - they can be stored together

Real-time Analytics

- Quick response to changes
- Simplified ETL
- **Real time,** along with opearational data

Document-based Unsuitable

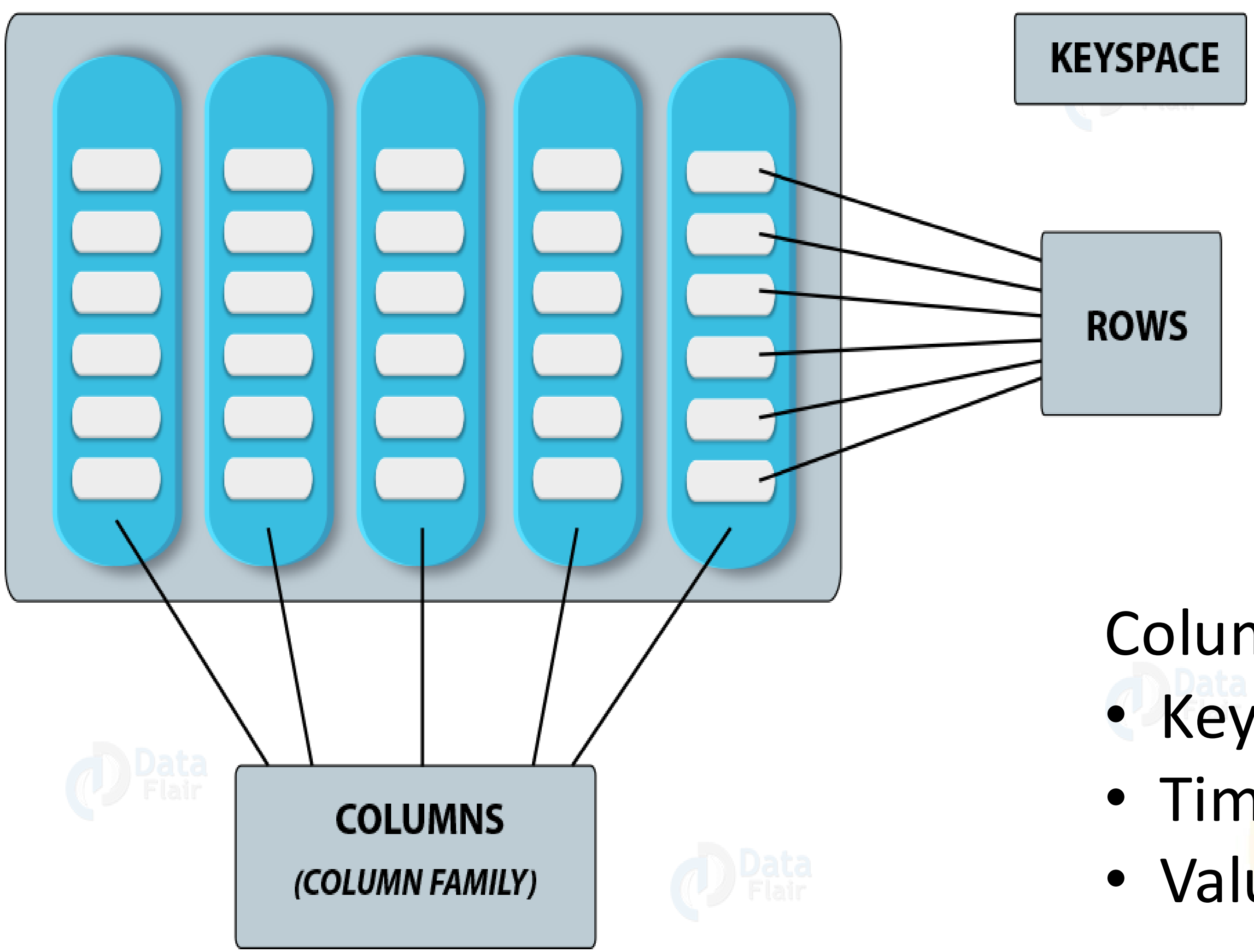
- Cannot handle transaction with multiple documents
- Not for aggregated-oriented model, handling complex join operations
- Not suited for high consistency is required

Column-Based

Column-Based Databases

- Columnar or Wide-Column databases
- Spawned from Google's 'Bigtable'
- Store data in columns
- Each row can has different column

Column-Based Databases



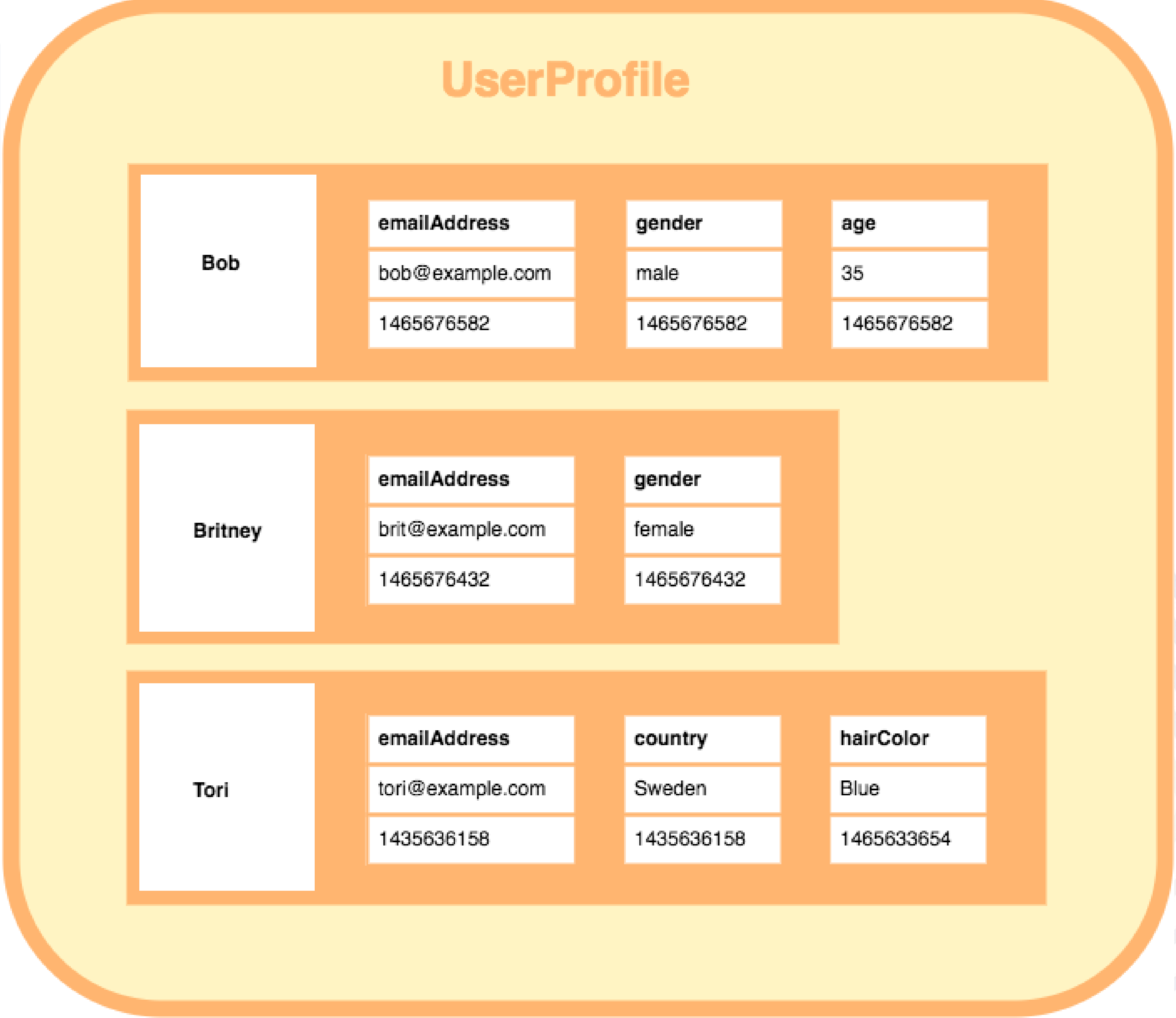
Columns have three values

- Key or columns name
- Timestamp
- Value

Example:

Column Family

User Profile



UserProfile

Bob

emailAddress	gender	age
bob@example.com	male	35
1465676582	1465676582	1465676582

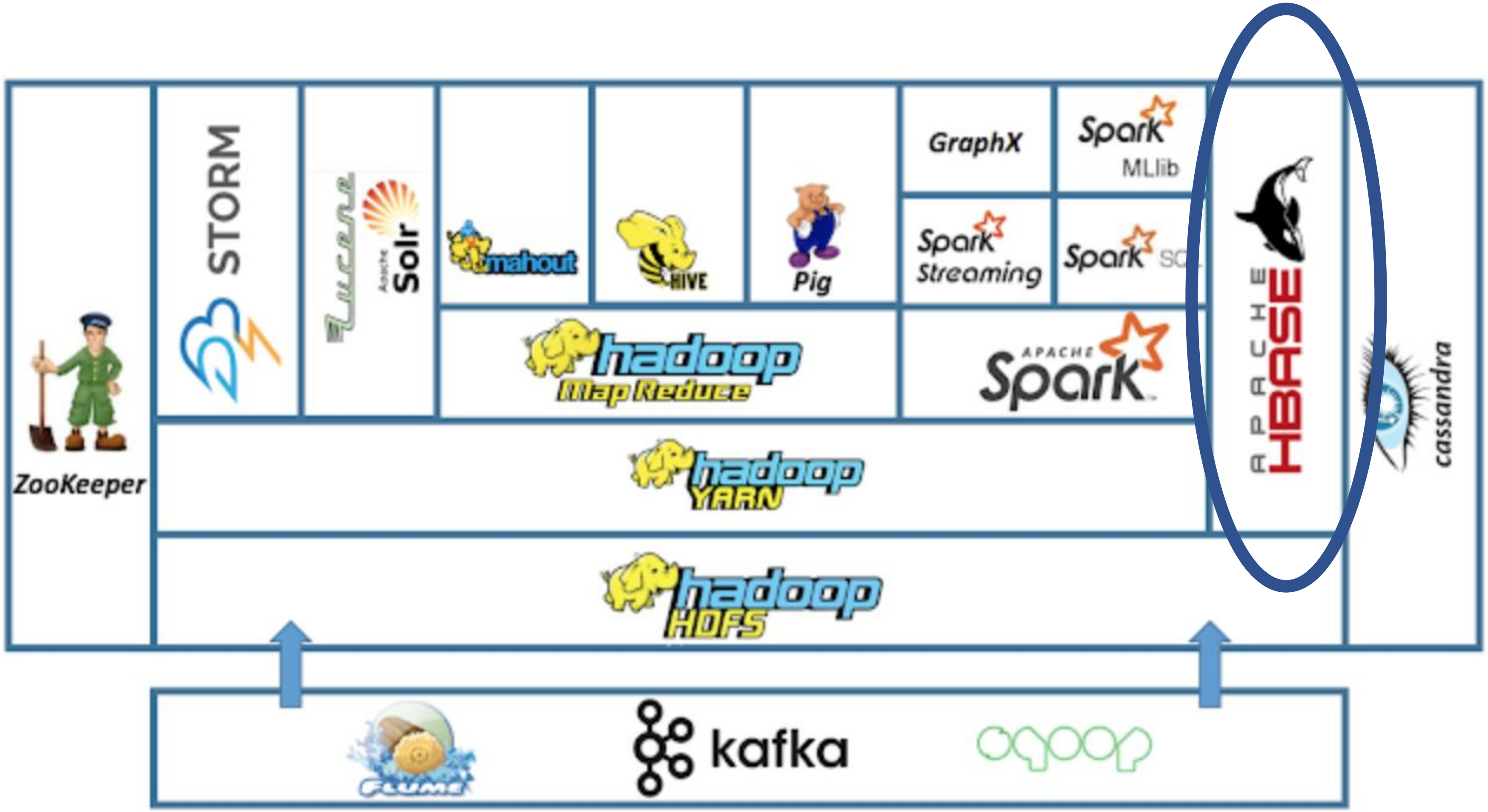
Britney

emailAddress	gender
brit@example.com	female
1465676432	1465676432

Tori

emailAddress	country	hairColor
tori@example.com	Sweden	Blue
1435636158	1435636158	1465633654

Hadoop Layer

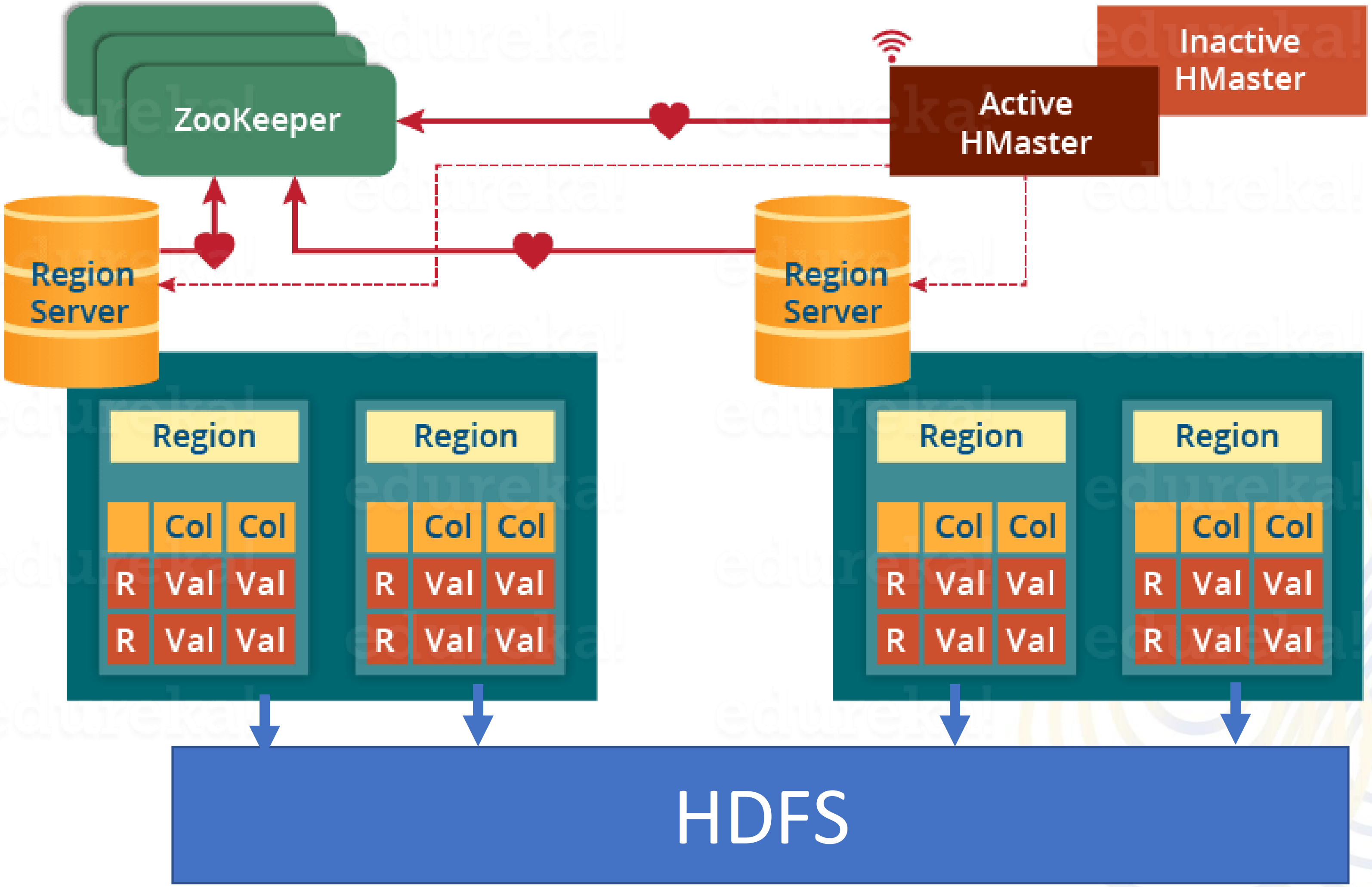


HBASE Architecture

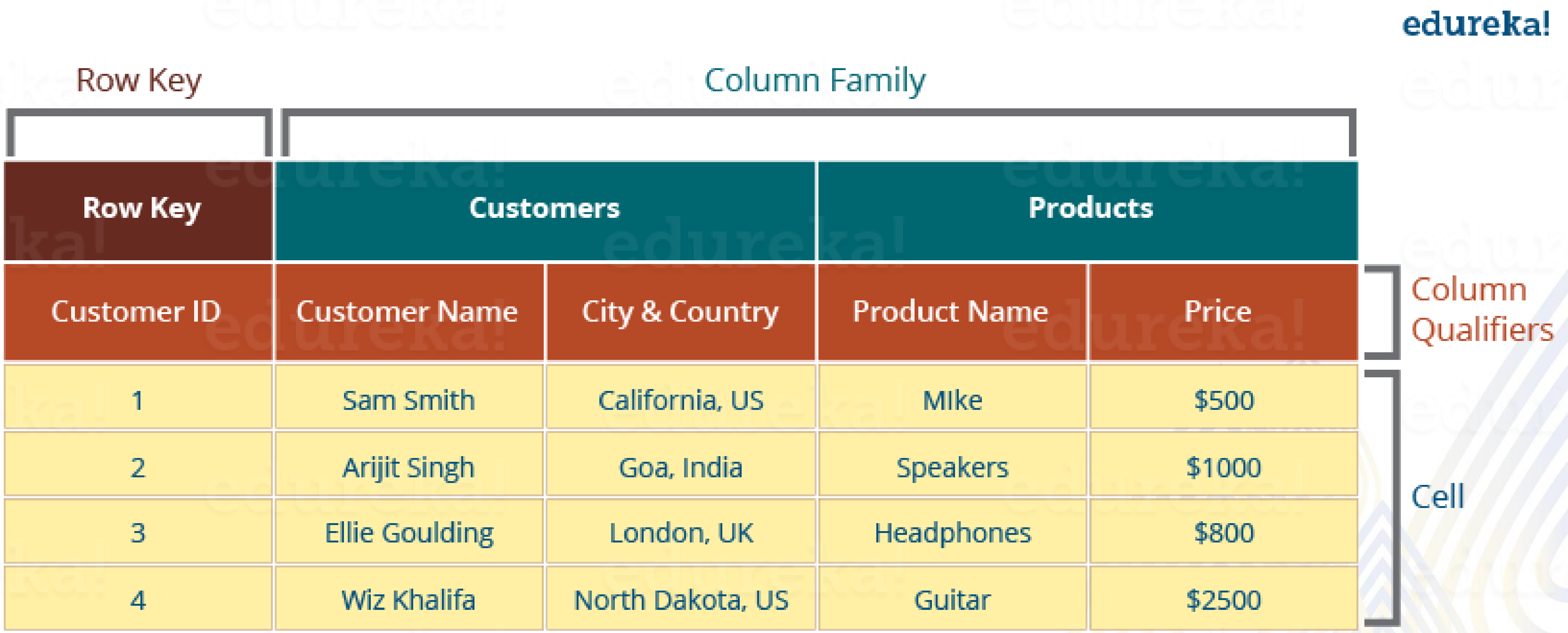
maintain configuration
Track node failure

Distribute/monitor services to region server

Mange the region
&
Communicate with client



HBASE Column Family

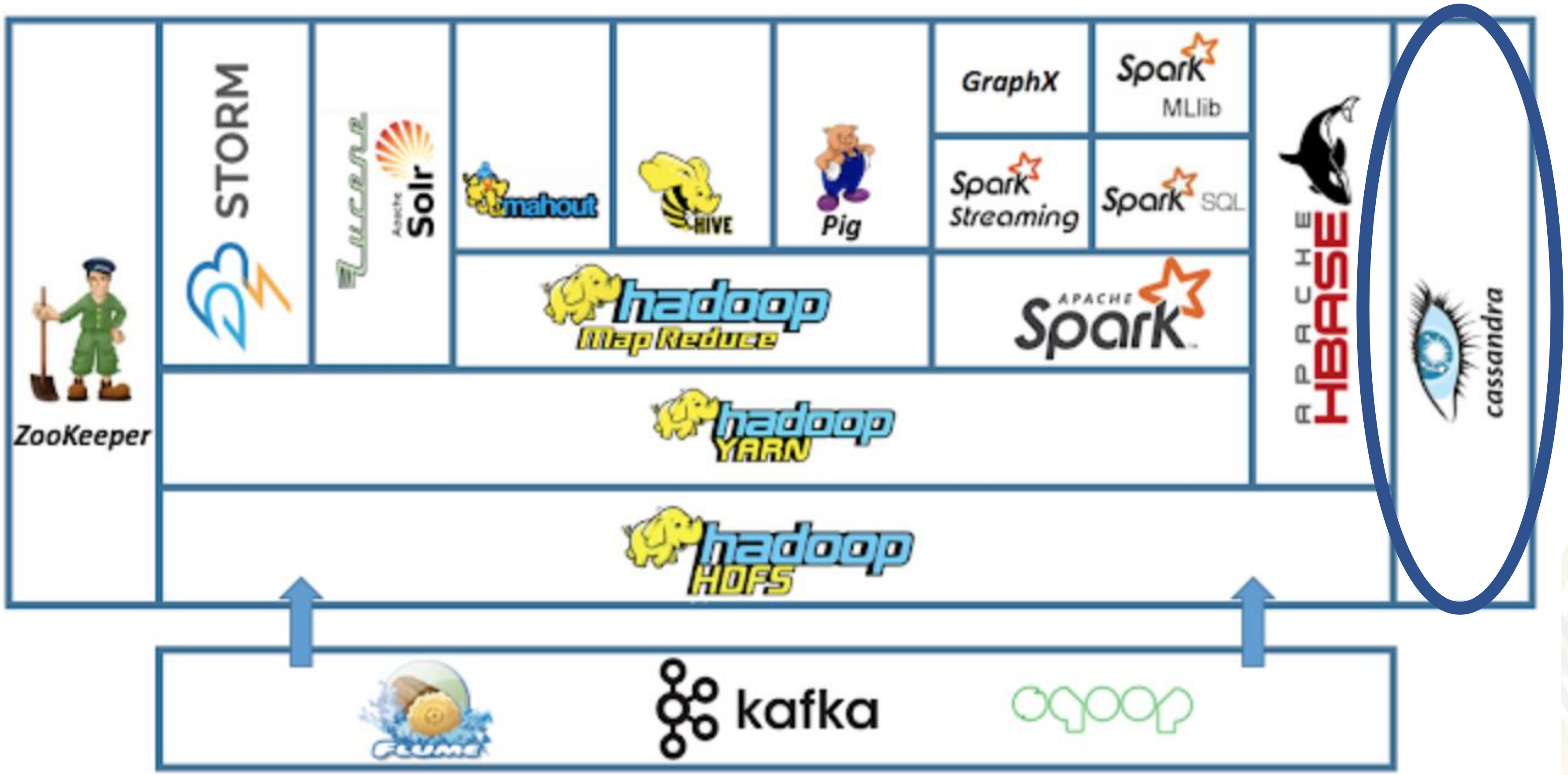

edureka!
Row Key
Column Family
Column Qualifiers
Cell

Row Key	Customers		Products	
Customer ID	Customer Name	City & Country	Product Name	Price
1	Sam Smith	California, US	MIke	$500
2	Arijit Singh	Goa, India	Speakers	$1000
3	Ellie Goulding	London, UK	Headphones	$800
4	Wiz Khalifa	North Dakota, US	Guitar	$2500

Figure: HBase Table

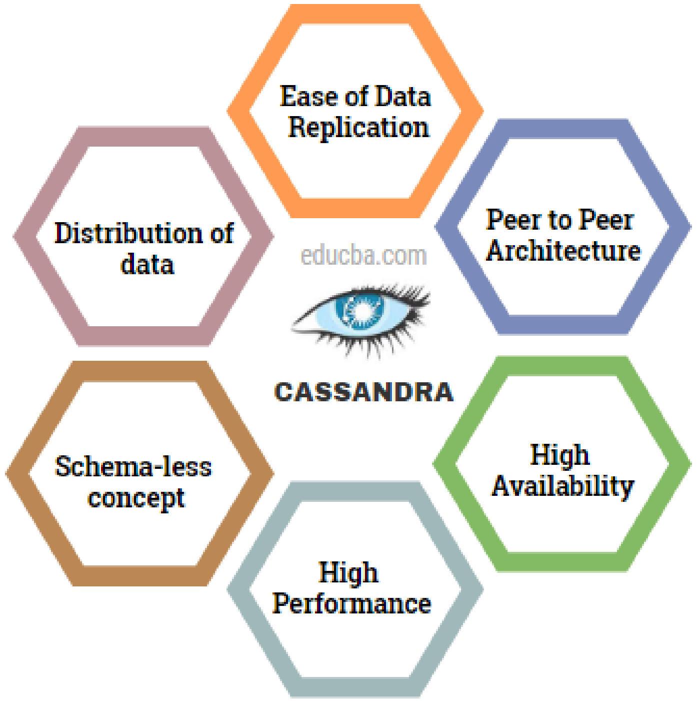
Cassandra

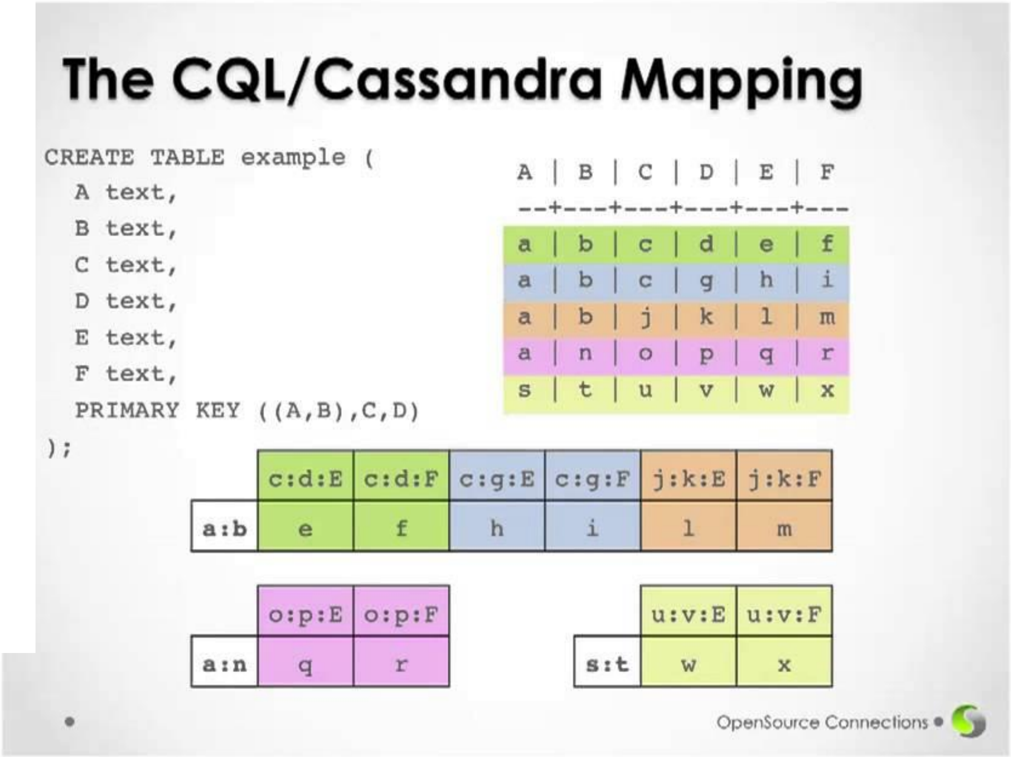
Hadoop Layer



Apache Cassandra

What is Cassandra





The CQL/Cassandra Mapping

```
CREATE TABLE example (
  A text,
  B text,
  C text,
  D text,
  E text,
  F text,
  PRIMARY KEY ((A,B),C,D)
);
```

```
 A | B | C | D | E | F
---+---+---+---+---+---
 a | b | c | d | e | f
 a | b | c | g | h | i
 a | b | j | k | l | m
 a | n | o | p | q | r
 s | t | u | v | w | x
```

	c:d:E	c:d:F	c:g:E	c:g:F	j:k:E	j:k:F
a:b	e	f	h	i	l	m

	o:p:E	o:p:F
a:n	q	r

	u:v:E	u:v:F
s:t	w	x

OpenSource Connections

Apache Cassandra

- Highly available
- Create peer-to-peer architechture
- Not good for a lot of update/delete

Write to main memory and update periodically

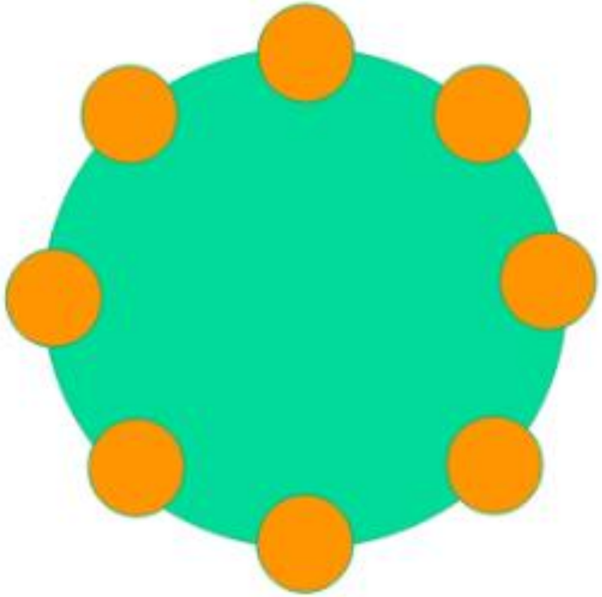
- Good for fast write (append)
- Netflix, Uber, Spotify, Time series applications

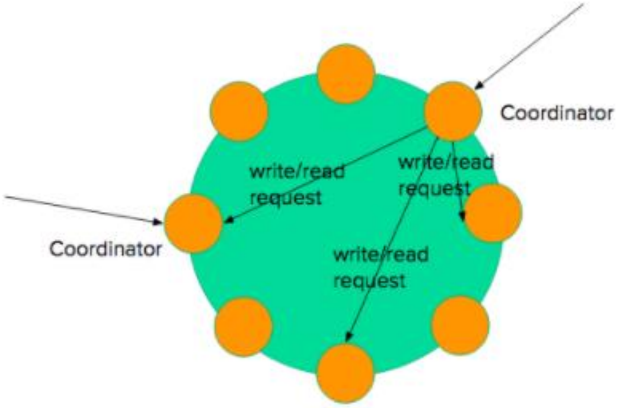
Cassadra Architecuture

Node

- The smallest physical item of a cluster
- No Primary/Secondary
- Equal to all nodes in the cluster
- Also Coordinator

Ring/Cluster





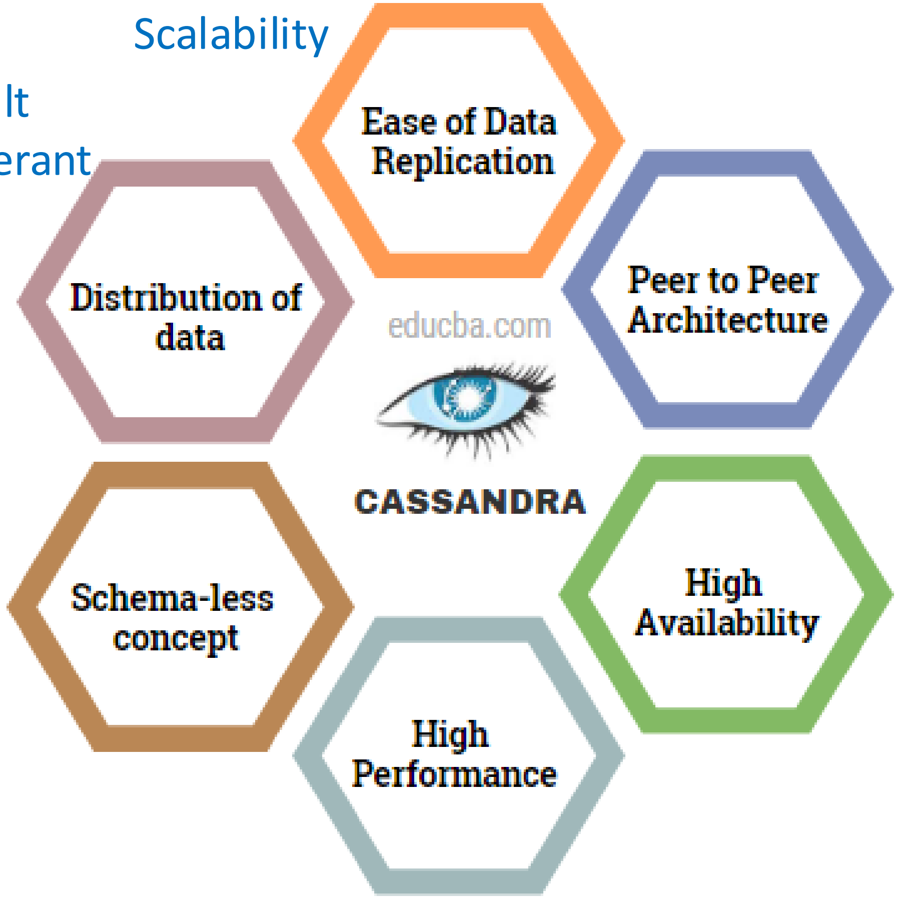
Cassandra Key Feature

Scalability

Fault Tolerant

Distribute& decentralized

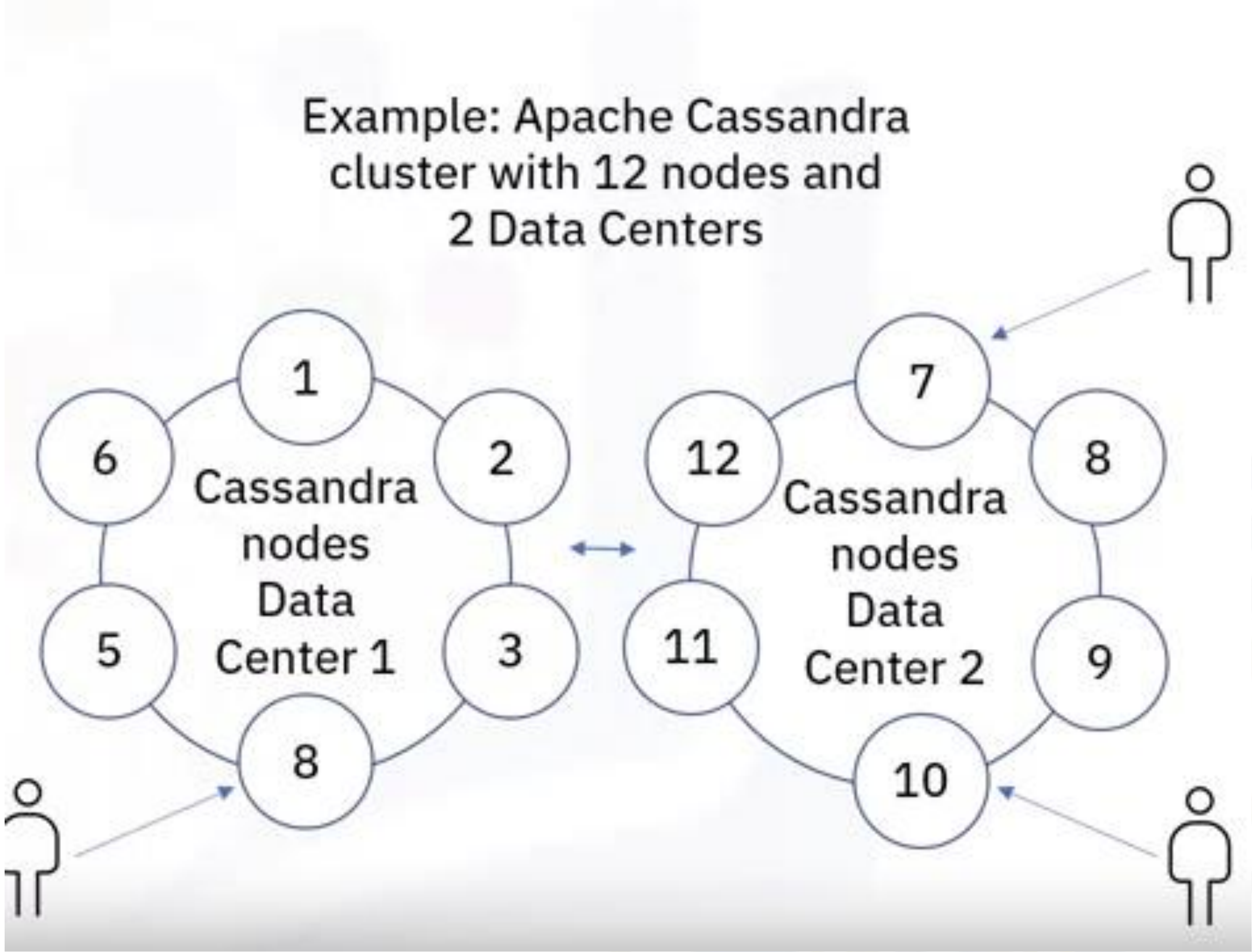
Easy to access



High write Througput

Distributed and decentralized

- Run on multiple distributed machines
- Application can route the user request optimally
- Use peer-to-peer architecture (gossip)
- All cluster are identical


Example: Apache Cassandra cluster with 12 nodes and 2 Data Centers
1
2
3
8
5
6
Cassandra nodes Data Center 1
7
8
9
10
11
12
Cassandra nodes Data Center 2

Data Queries

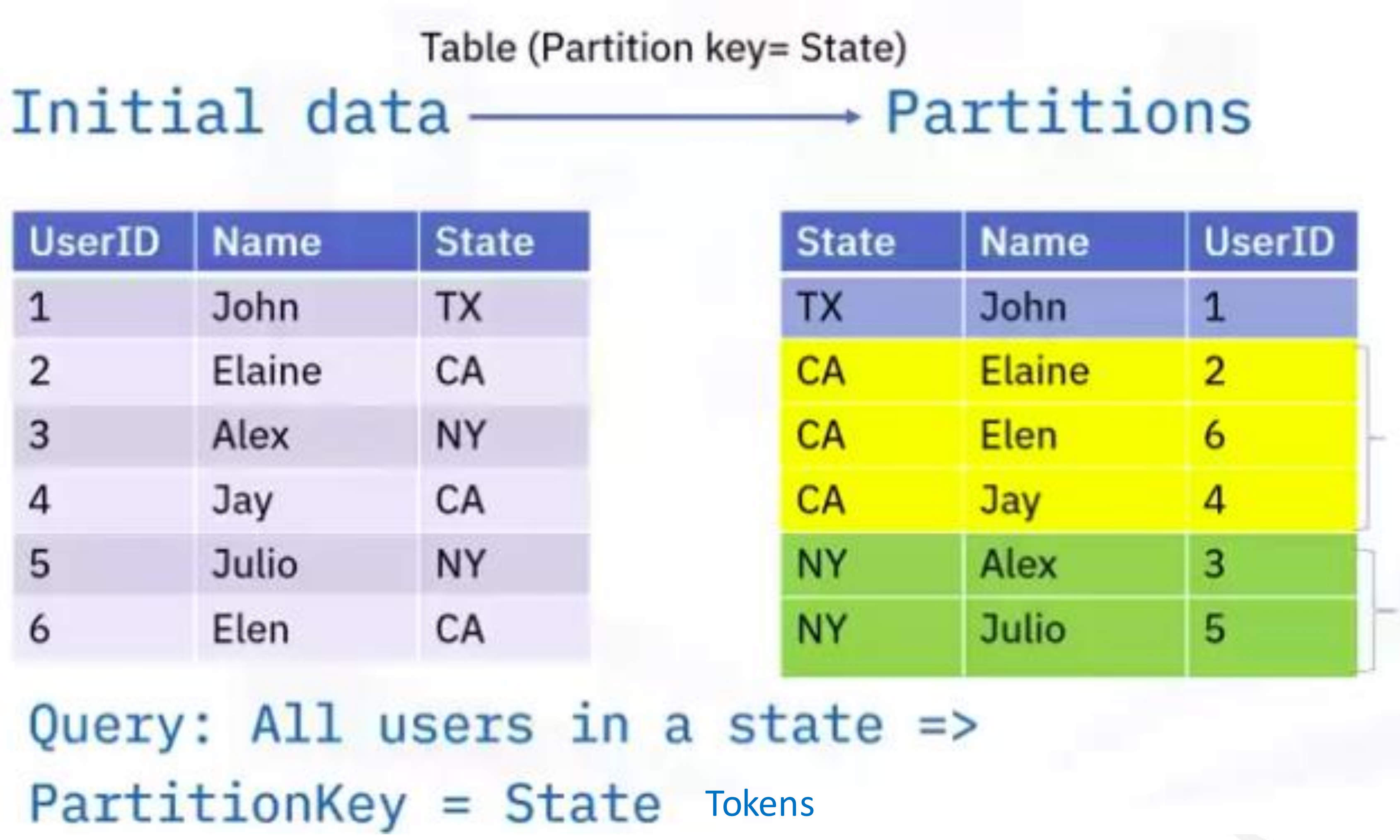


Table (Partition key= State)

Initial data → Partitions

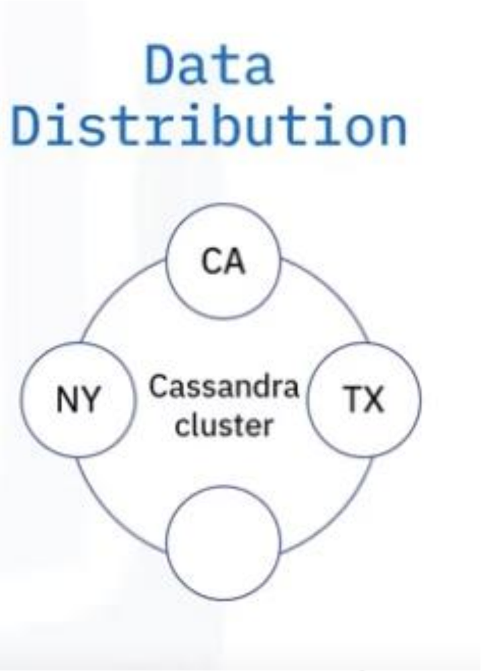
UserID	Name	State
1	John	TX
2	Elaine	CA
3	Alex	NY
4	Jay	CA
5	Julio	NY
6	Elen	CA

State	Name	UserID
TX	John	1
CA	Elaine	2
CA	Elen	6
CA	Jay	4
NY	Alex	3
NY	Julio	5

Query: All users in a state =>

PartitionKey = State

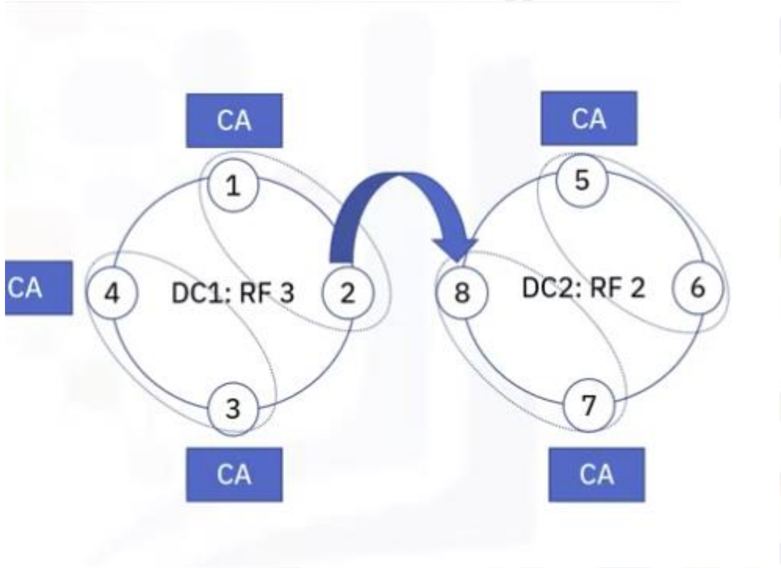
Tokens



Data Distribution

Data replication

- Replicas
  - Tells how many nodes contain the data(partition)
- Done clockwise – based on placement of rack/ data center
- Replication Factor: detemine the number nodes to whole the replication of replicas



Availability vs Consistency

- Always available***

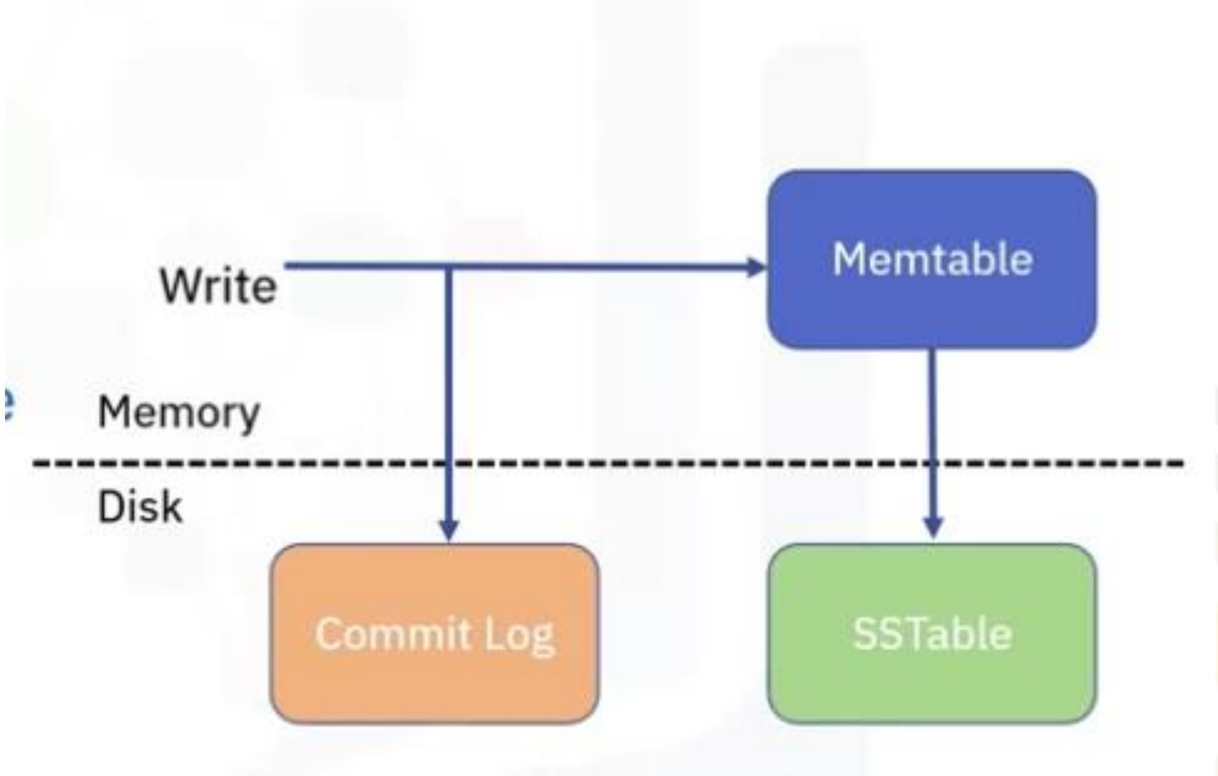
- **Tunable** consistency
  - Per operation set consistency(read/write)
  - Tune →Strong or Eventual
  - Conflict is solved during READ operations

Fast and Linear Scalability

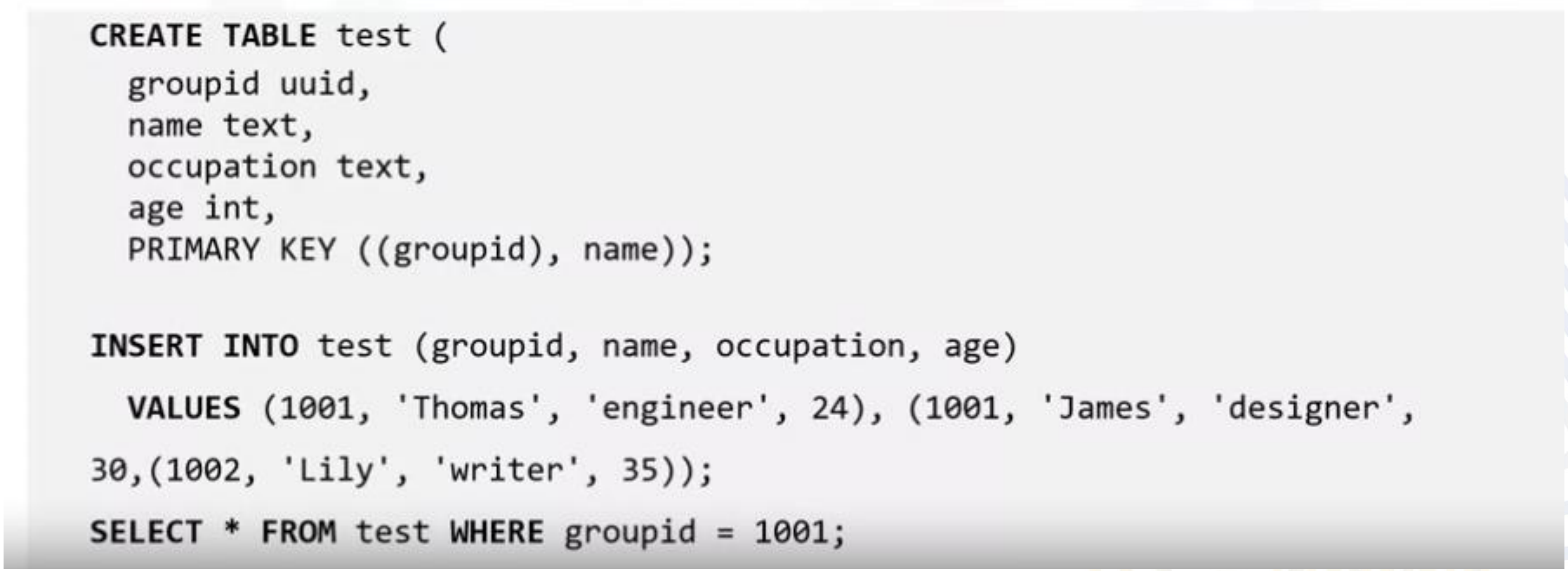
- Scale horizontally by adding new nodes in the cluster
- Performance increases linearly
- New nodes are automatically assinged tokens
- Addition/Reomoving nodes is done seamlessly

High write throughput

- Write ared done in memory nodes
- Write in memory →Flush on disk
- All disk are append sequentially



CQL syntax



```
CREATE TABLE test (
  groupid uuid,
  name text,
  occupation text,
  age int,
  PRIMARY KEY ((groupid), name));

INSERT INTO test (groupid, name, occupation, age)
  VALUES (1001, 'Thomas', 'engineer', 24), (1001, 'James', 'designer',
30,(1002, 'Lily', 'writer', 35));
SELECT * FROM test WHERE groupid = 1001;
```

https://www.datastax.com/try-it-out

Columnar Databases

Suitable for

- Large amounts of sparse data – Good for compression
- Can handle across cluster of nodes
- Best on column-wise data analytic
- *Popular* for counter type database
- Column can have expiry date as a parameter
  - Good for trial period
  - Anything with TTL(Time to live)

Columnar Databases

NOT Suitable for

- Difficult query– require changes to the column to make it easier
- More difficult updates

Use cases

- Log file
- E-commerce

Example The browsing history data might include the following columns for each user interaction:

- user_id (integer)
- timestamp (datetime)
- product_id (integer)
- category (string)
- price (float)
- device (string)
- duration_seconds (integer)
- action (e.g., 'view', 'add_to_cart', 'purchase')

Other types of database

- Redis
    - Enhanced key-value store
    - Store different types of data– hash, bitmap, ziplist
    - Twitter usage
- AsterixDB
    - Semi-structured data
- Solr
    - Text management
    - Search Engine

Cassandra

- Deploy Cassandra using GCP Market place
- Intro to canssandra
- Insert/Update/delete

More Examples

- Order Management
- Sensor data

Graph Database

Graph Database

- Store information in entities(node) and relationship(edges)
- Good for dataset with graph-like structure
- Not scale well horizontally

Graph Database

Highly connected and related

Example

- Social networking
- Routing
- Map application
- Recommendation engines

Summary

RDBMS

  - Consistency
  - Structured data(Fixed Schema)
  - Transaction
  - Join operations

- Non- RDBMS
  - High Performance
  - Unstructured data(Flexible Schema)
  - Availability
  - Easy Scalibility