

Apache Hadoop and NoSQL

Apache Hadoop

Collection

- Of open-source software utilities

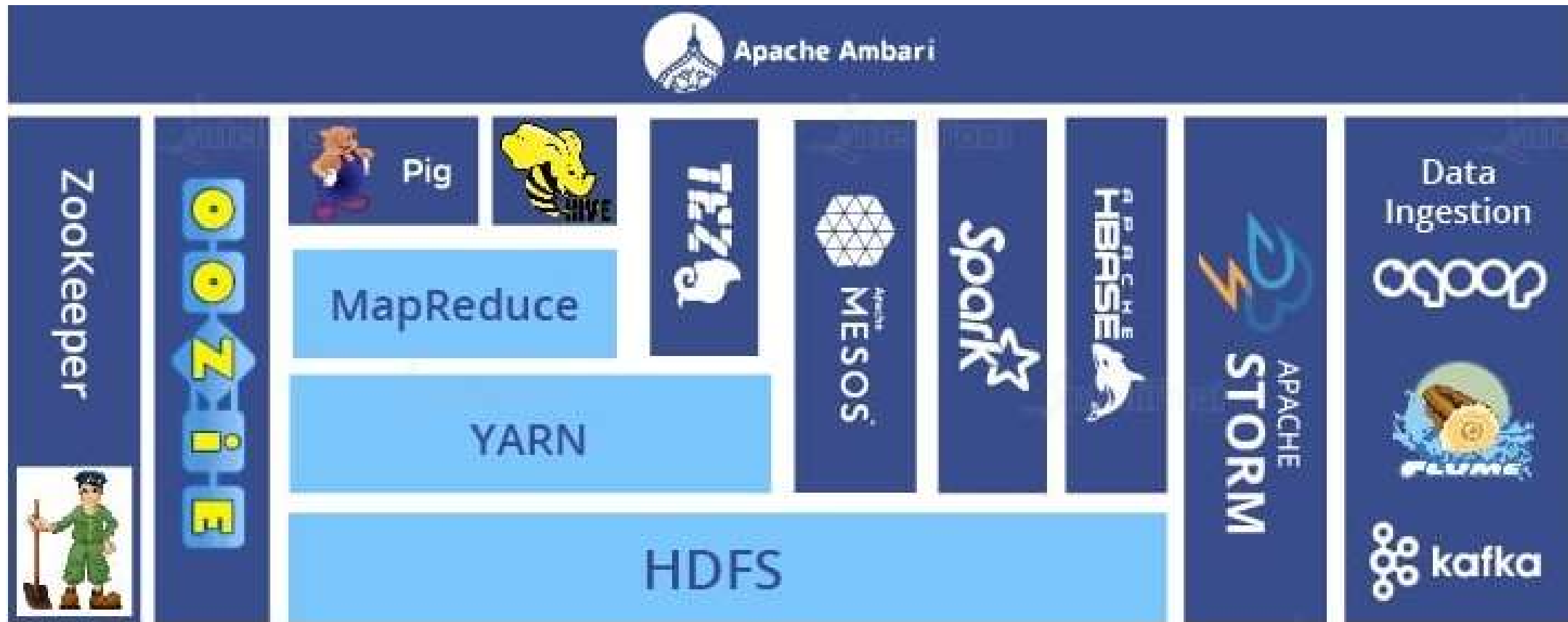
Facilitates using a network of many computers

- To solve problems involving massive amounts of data and computation

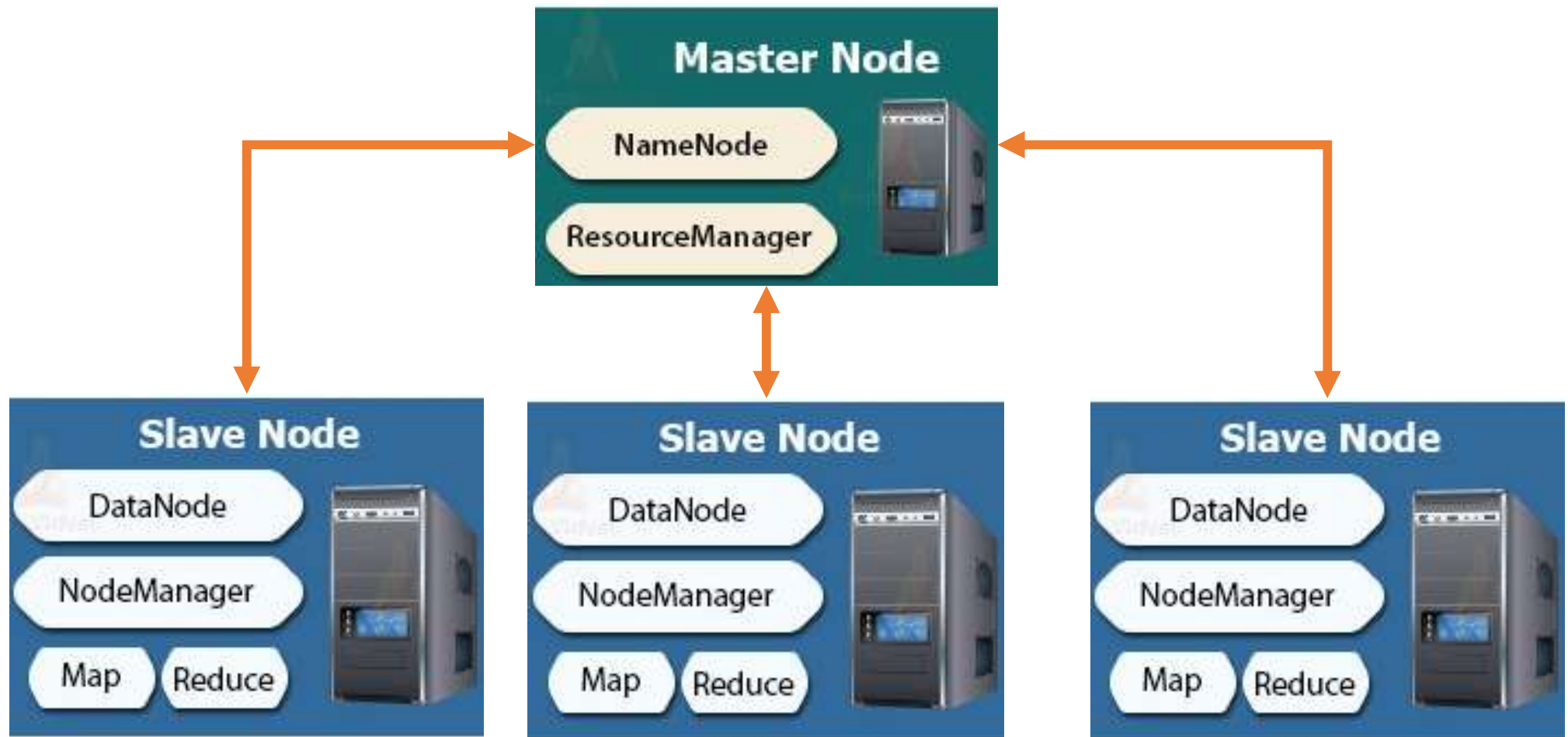
Provides a software framework

- For distributed storage and processing of big data using the MapReduce programming model

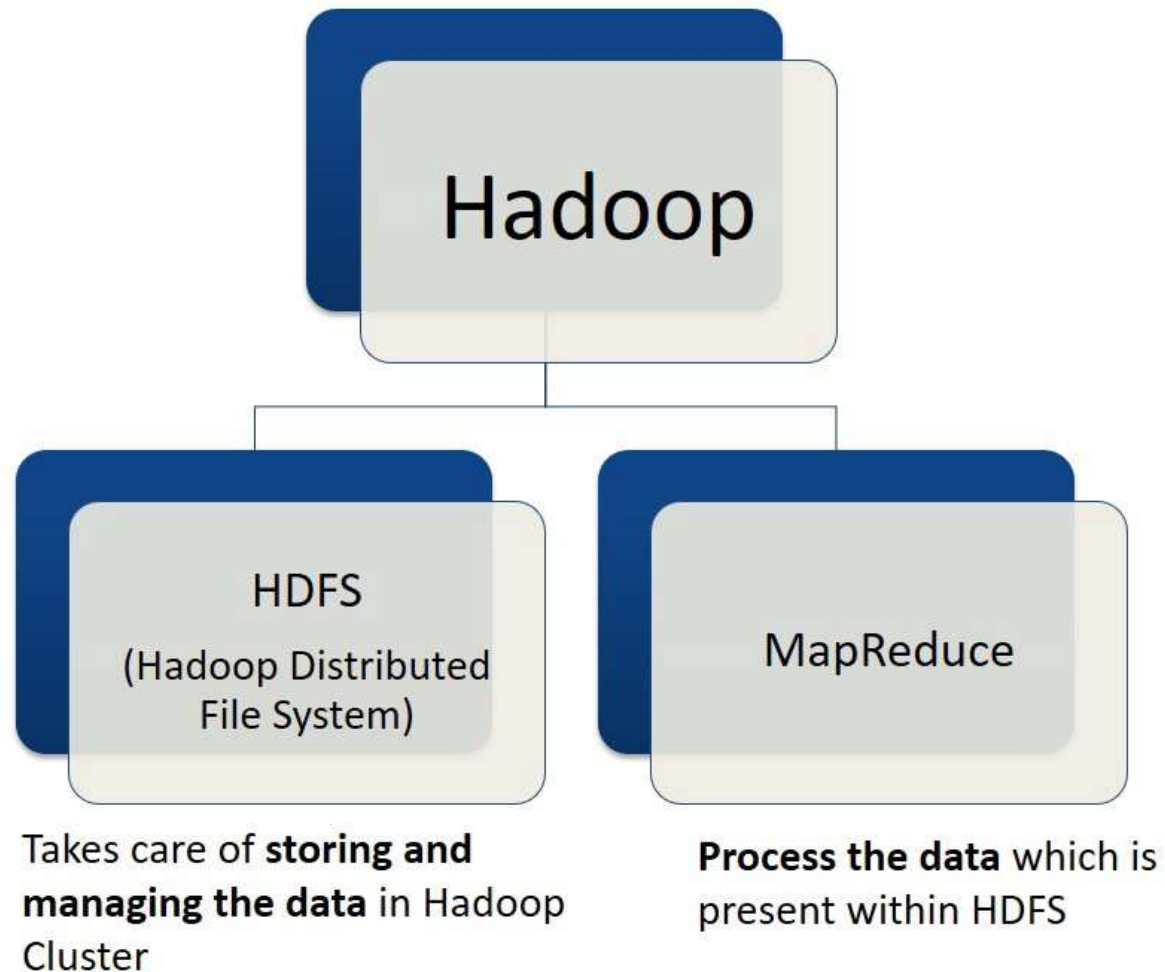
Hadoop ecosystem



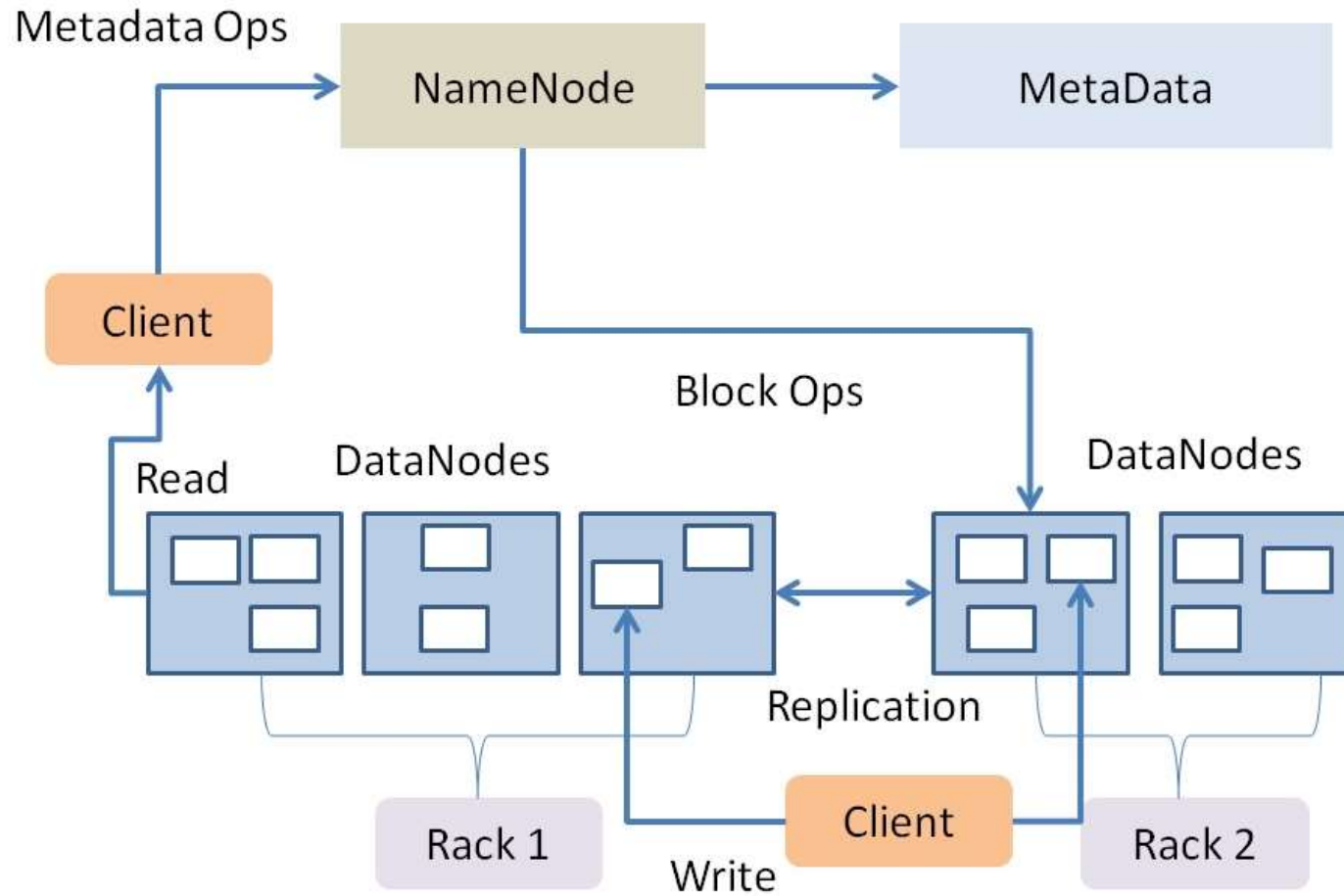
Hadoop Architecture



HDFS and MapReduce

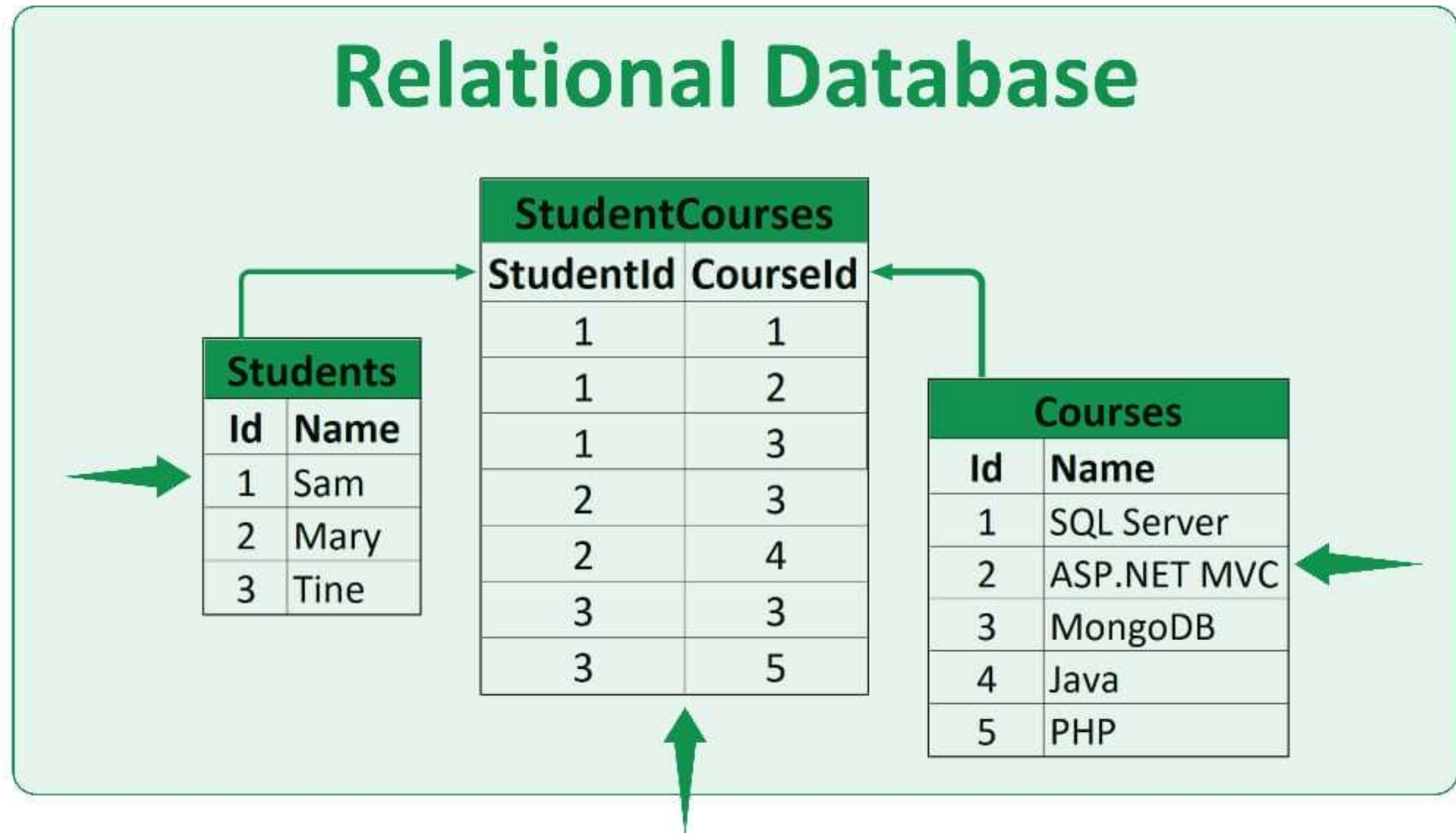


HDFS Architecture

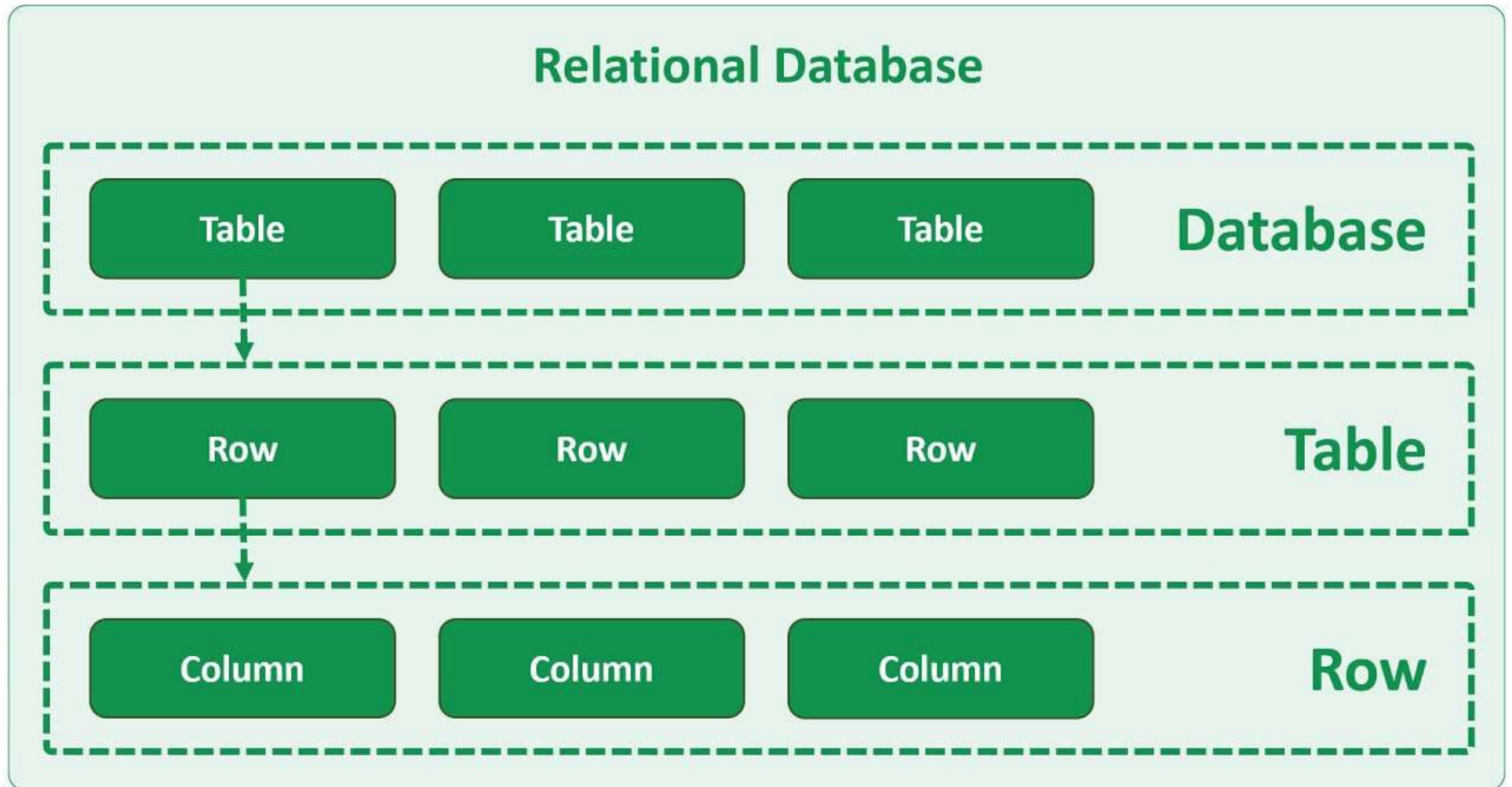


NoSQL

Relational Database



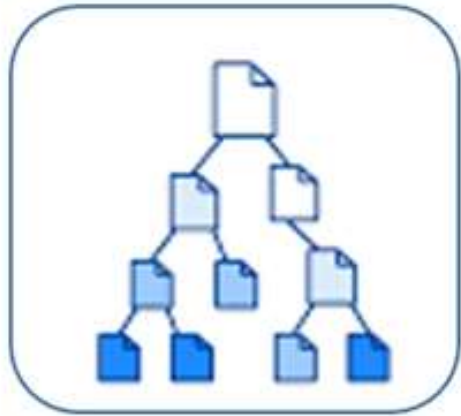
Relational Database



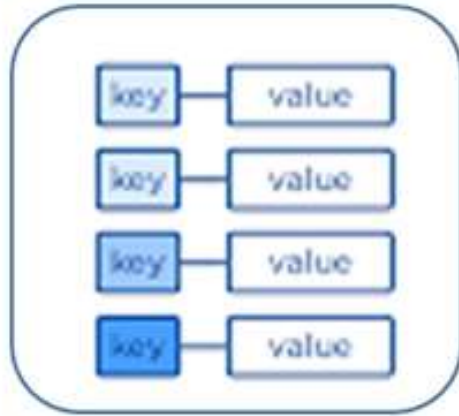
NoSQL Database



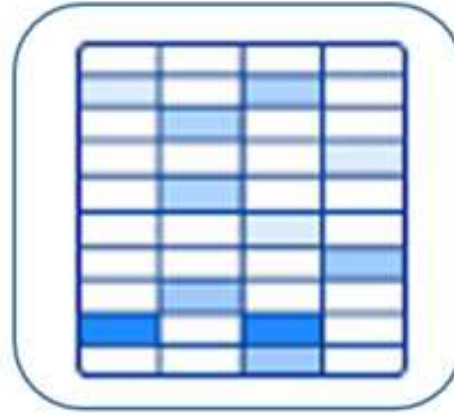
NoSQL



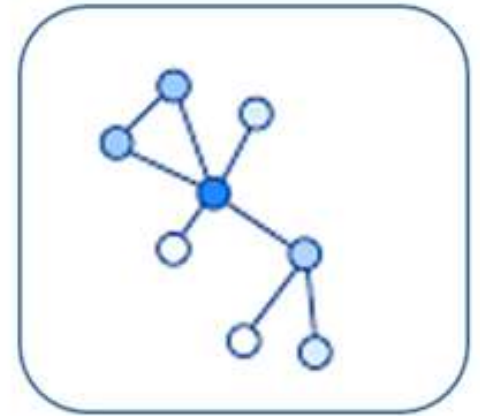
Document
Store



Key-Value
Store

















Wide-Column
Store



Graph
Store

Types of NoSQL Databases

Document Database	Graph Databases
  	 
Wide Column Stores	Key-Value Databases
   	    

Applications on NoSQL databases

NoSQL



Gaming



Social



IoT



Web



Mobile



Enterprise



Key/value store



Document database



Column family store

SQL



Web



Mobile



Enterprise



Data mart



Relational table storage

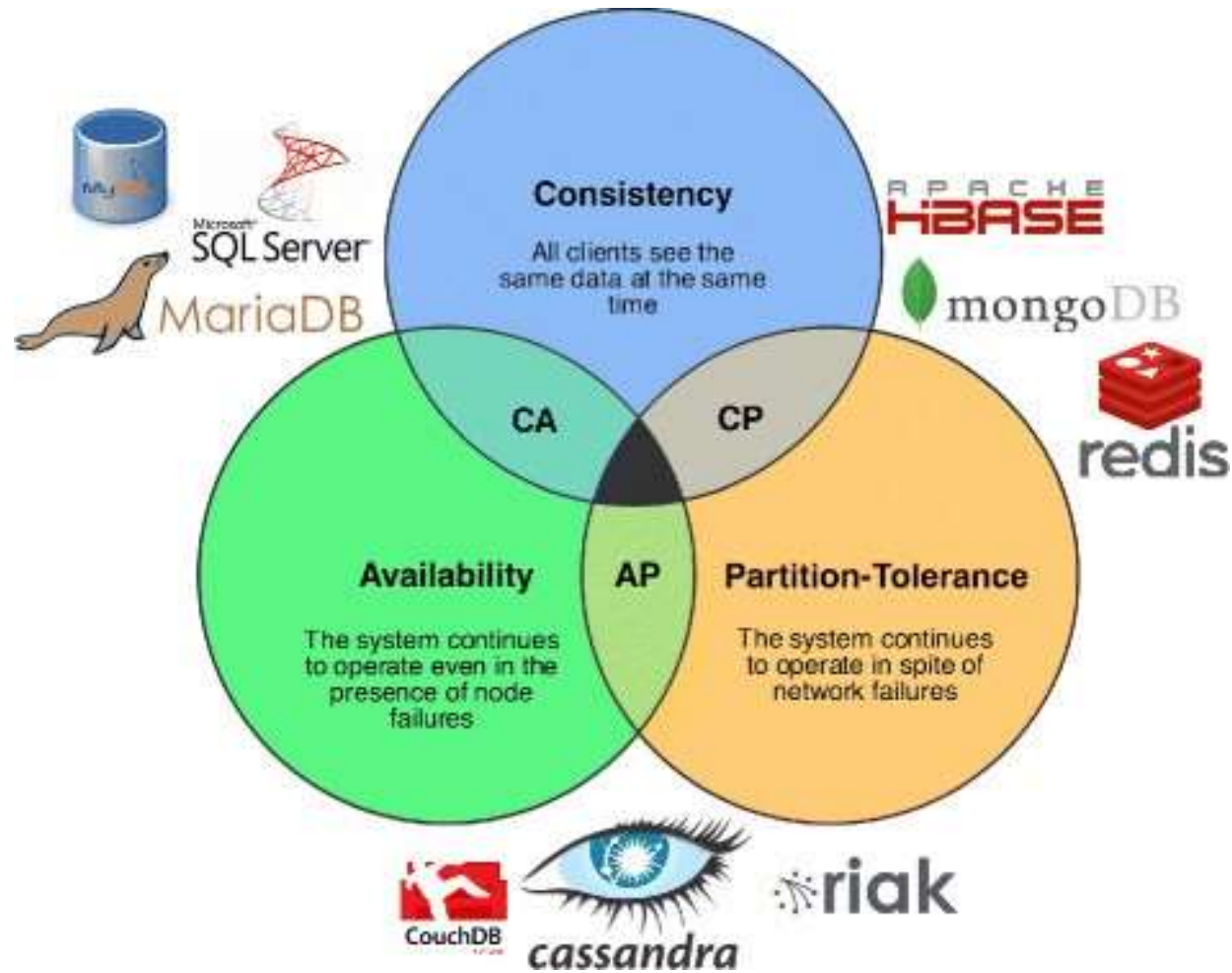


Relationships use joins

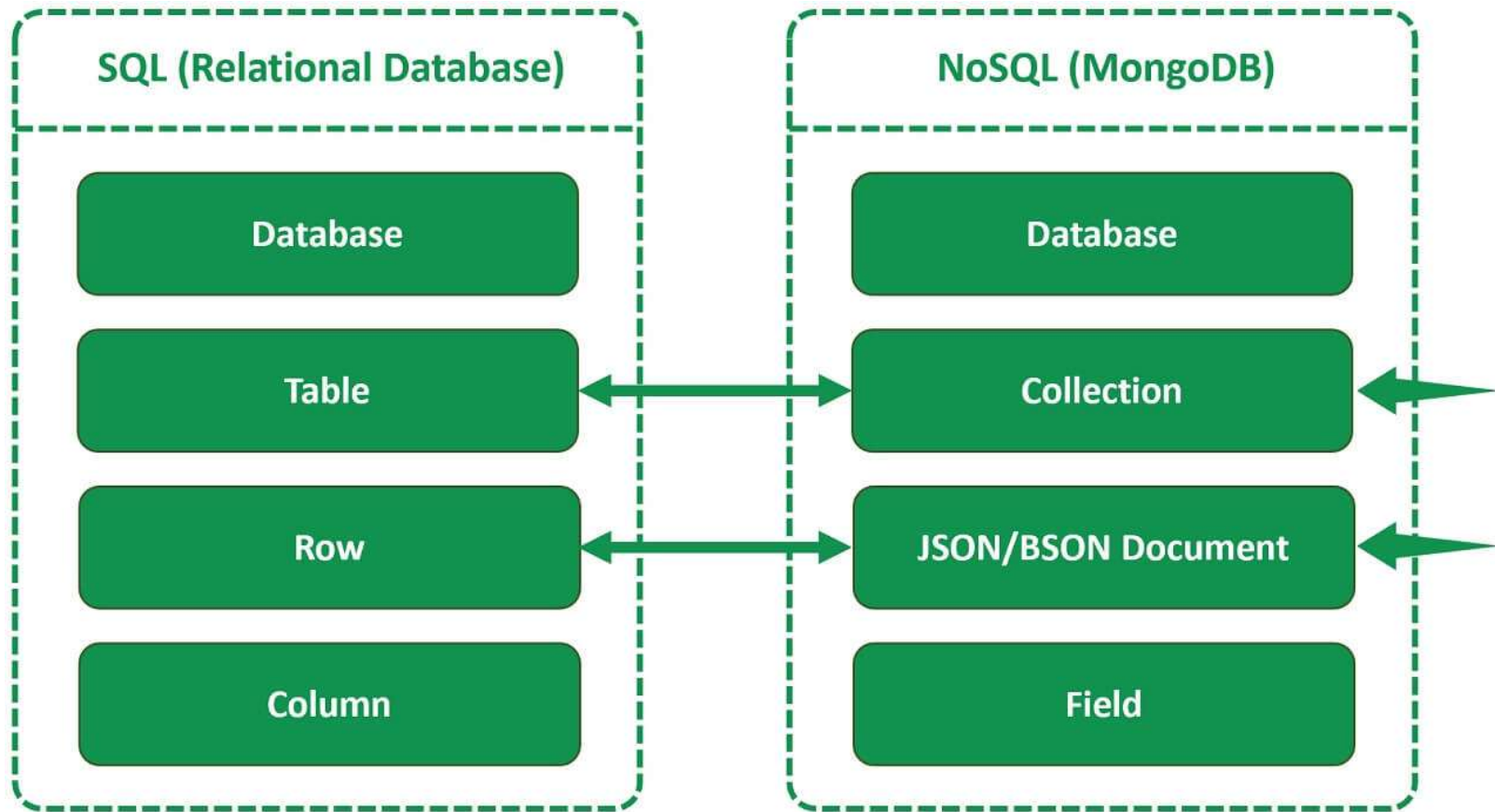
Schema-on-Read vs Schema-on-Write

Schema-on-Write	Schema-on-Read
Feast Reads	Slower Reads
Slower Loads	Fast Loads
Structured	Structured/Semi structured
Fewer Errors	More Errors
SQL	NoSQL

CAP Principal



Relational vs non-relational databases



Nested JSON document

```
{
  "_id": 1,
  "name": "John",
  "age": 25,
  "gender": "Male",
  "graduated": true,
  "address": {
    "street": "123 Main Street",
    "city": "Star City"
  },
  "awards": [ {"award": "Star Student", "year": 2021}, {"award": "Math Master", "year": 2020} ]
}
```



Nested JSON document

Column-oriented DBMS

- Stores data tables by column rather than by row
- Benefits
 - More efficient access to data when only querying a subset of columns (by eliminating the need to read columns that are not relevant)
 - More options for data compression
- Cons:
 - Typically less efficient for inserting new data
- Column oriented databases will have significant benefits when stored on separate disks

Row-oriented			
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented		Column-oriented		Column-oriented	
Name	ID	Grade	ID	GPA	ID
John	001	Senior	001	4.00	001
Karen	002	Freshman	002	3.67	002
Bill	003	Junior	003	3.33	003

Column-oriented DBMS

- A column store database can also be referred to as a:

- Column database
- Column family database
- Column oriented database
- Wide column store database
- Wide column store
- Columnar database
- Columnar store

- Use a concept called a keyspace

Row-oriented			
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented					
Name	ID	Grade	ID	GPA	ID
John	001	Senior	001	4.00	001
Karen	002	Freshman	002	3.67	002
Bill	003	Junior	003	3.33	003

Column-oriented DBMS

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

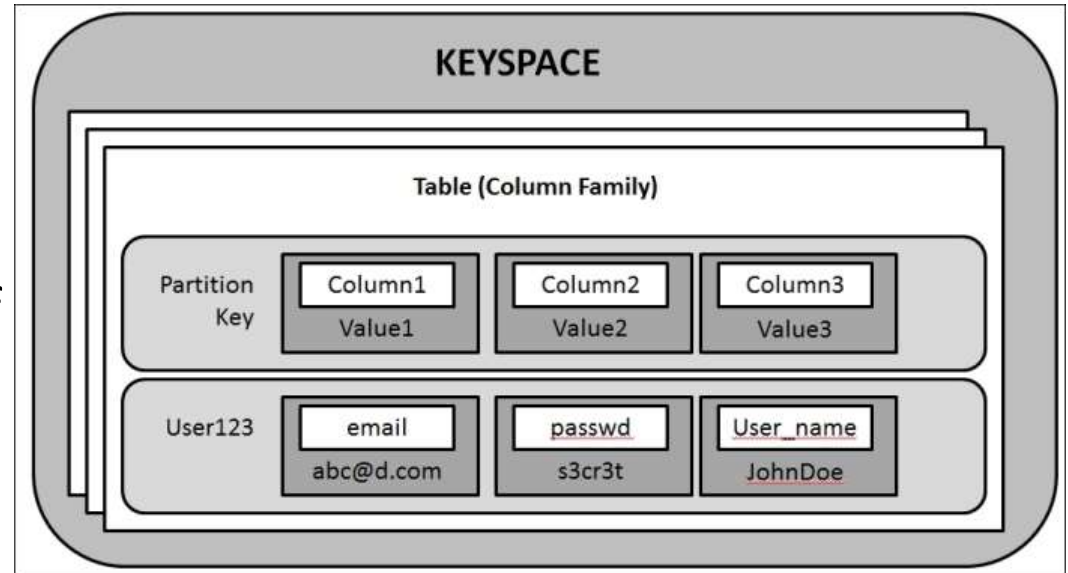
Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

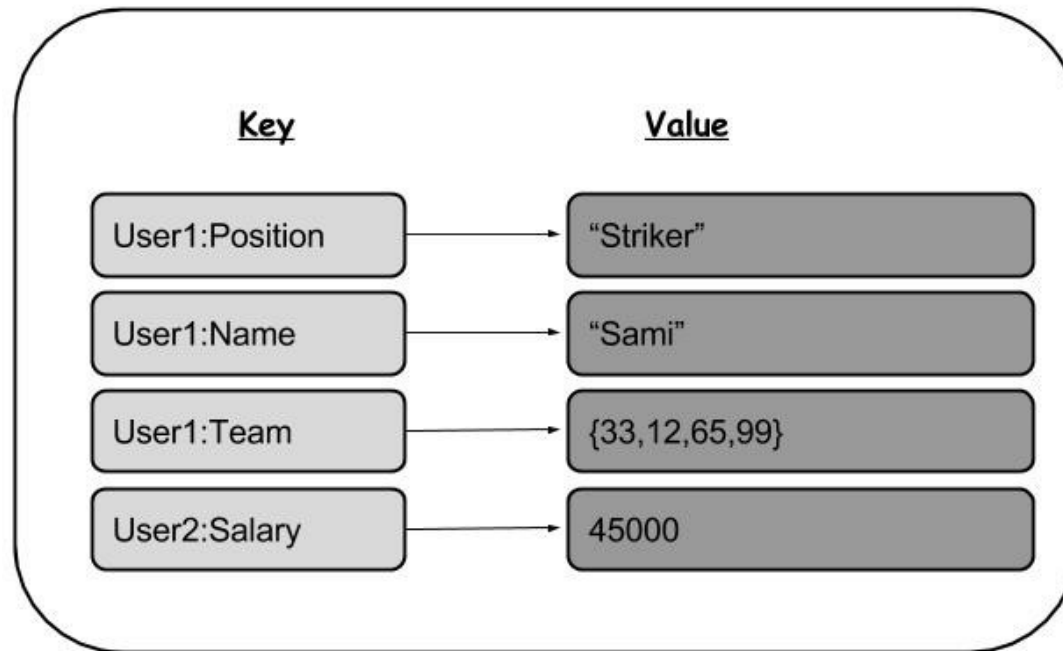
GPA	ID
4.00	001
3.67	002
3.33	003

Keyspace

- Kind of like a schema in the relational model.
- Contains all the column families which contain rows, which contain columns.
- From a user perspective, the metadata of a columnar database looks exactly the same as a RDBMS
- You perform schema management in much the same way as Oracle
- In most cases, it's 100% SQL compliant and 100 ACID compliant (unlike many NoSQL)
- NoSQL databases tend to be either Key Value stores or Document Stores. Columnar is neither



key-value database



Key / Value Database

- Just keys and values

No schema

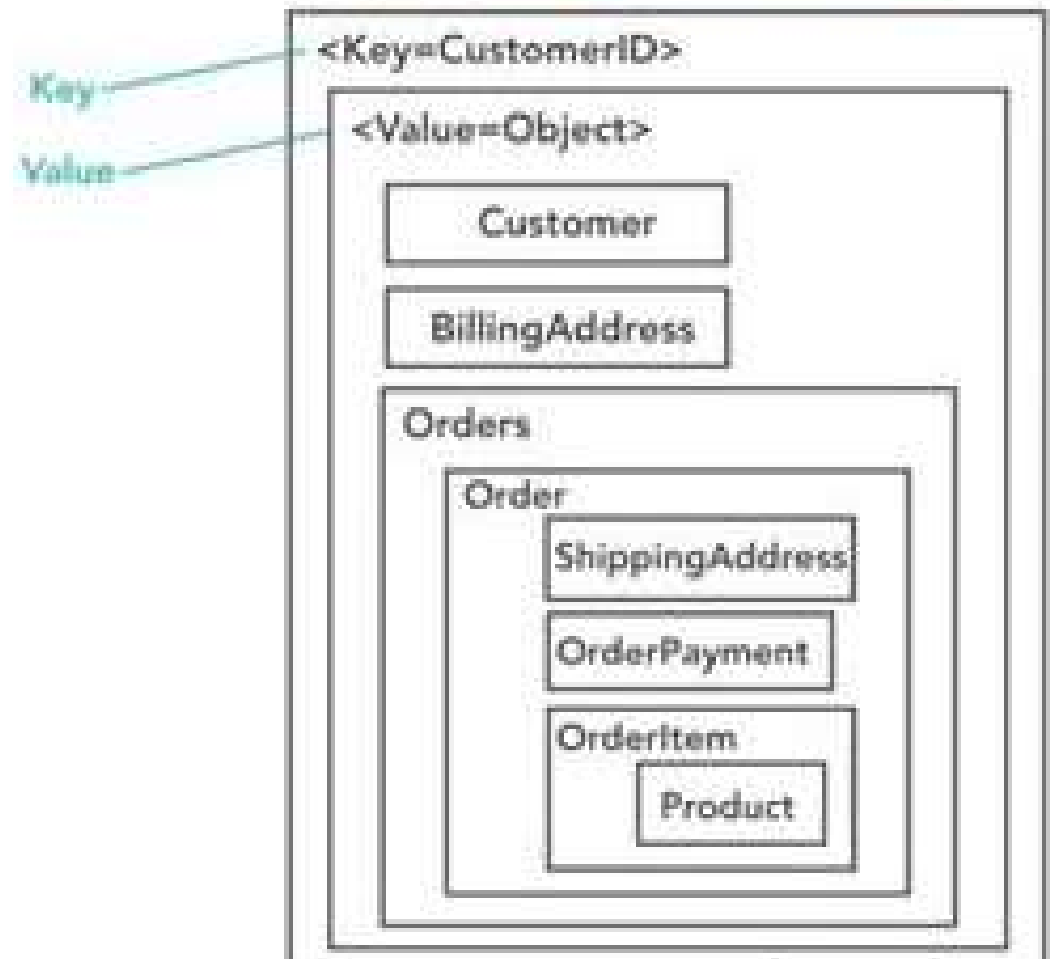
- Persistent or volatile

- Examples

Redis


AWS DynamoDB

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



key-value database

"phone" "(800) 123-4567"

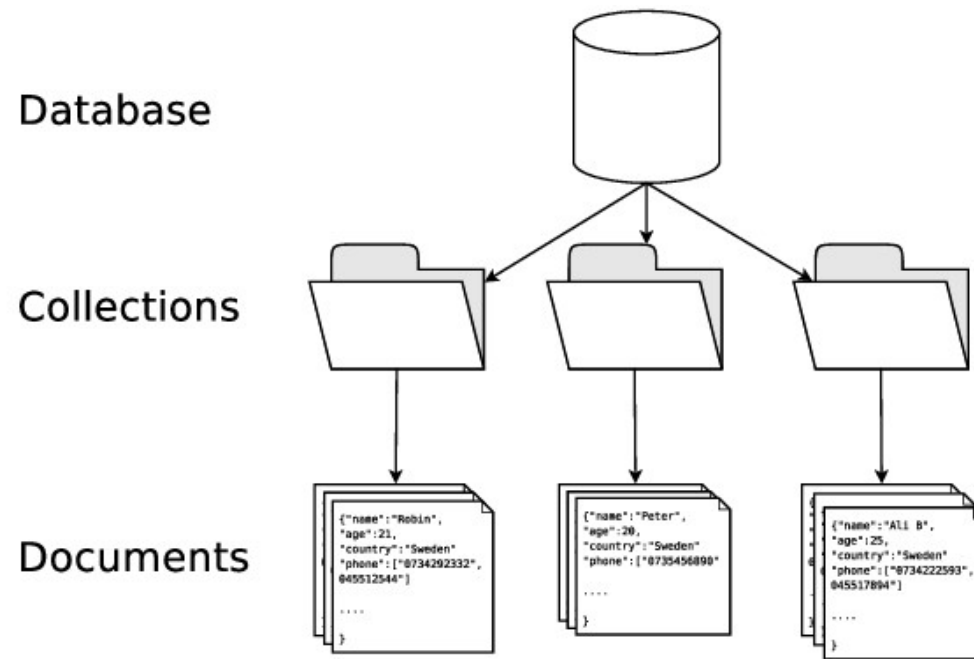
"books"  ,  , 



"address" { street: "...", city: "..." }

"binary" 101010111001

Document database



Document database vs key-value database

- An extension of the key-value database model
- Support more advanced querying capabilities
 - Key-value databases typically have more limited querying capabilities and may not support advanced search or indexing features.
 - In Key value database we can only query by key but in document store, not limited to query by key
- Have built-in support for indexing and searching.
- The term document in NoSQL databases refers to a set of key-value pairs, typically represented in
 - JSON
 - XML or
 - BSON

AWS DynamoDB

- Based upon open-source MongoDB
- Fully managed NoSQL database
- Supports both
 - Key-value and
 - Document data models
- Schemaless
- Does not have a query optimizer
- Secondary index is only used when querying or scanning



AWS Keyspaces

- Can have predefined structure but flexible as well
- Column-family data model



Azure Cosmos DB

- NoSQL document
- Supports the APIs:
 - MongoDB: document
 - PostgreSQL: document
 - Cassandra: Columnar
 - Gremlin: Graph API and
 - Table: Key Value



JSON and BSON

- BSON is the binary encoding of JSON-like documents that MongoDB uses when storing documents in collections
- It adds support for data types like Date and binary that aren't supported in JSON.
- Pros of BSON:
 - Compact
 - In most cases, storing a BSON structure requires less space than its JSON equivalent
 - Data Types
 - BSON provides additional data types not found in regular JSON, such as Date and BinData

JSON and BSON

	JSON	BSON
Encoding	UTF-8 String	Binary
Readability	Humans and Machines	Machines Only
Data Types	String, Boolean, Number, Array	String, Boolean, Number (Integer, Float, Long, Decimal128...), Array, Date, Raw Binary

Relational vs NoSQL

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

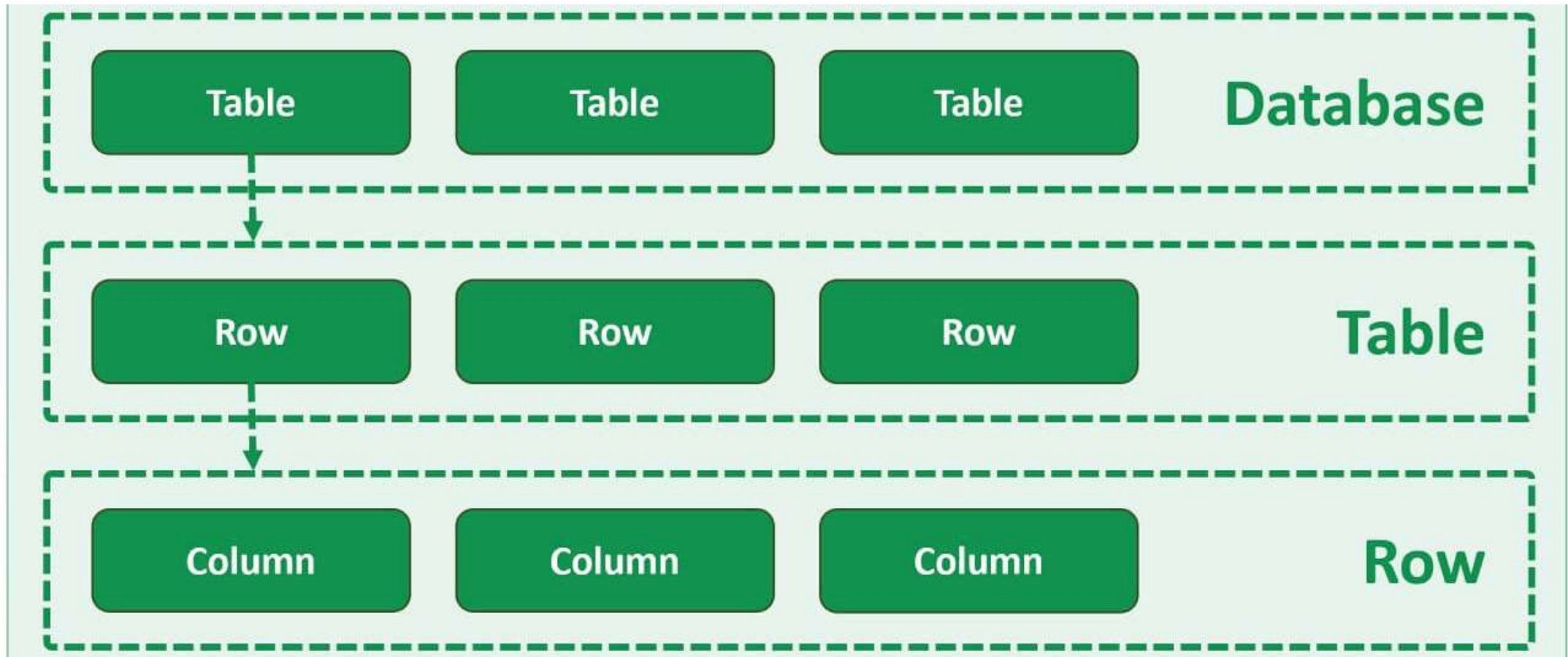
Document 2

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

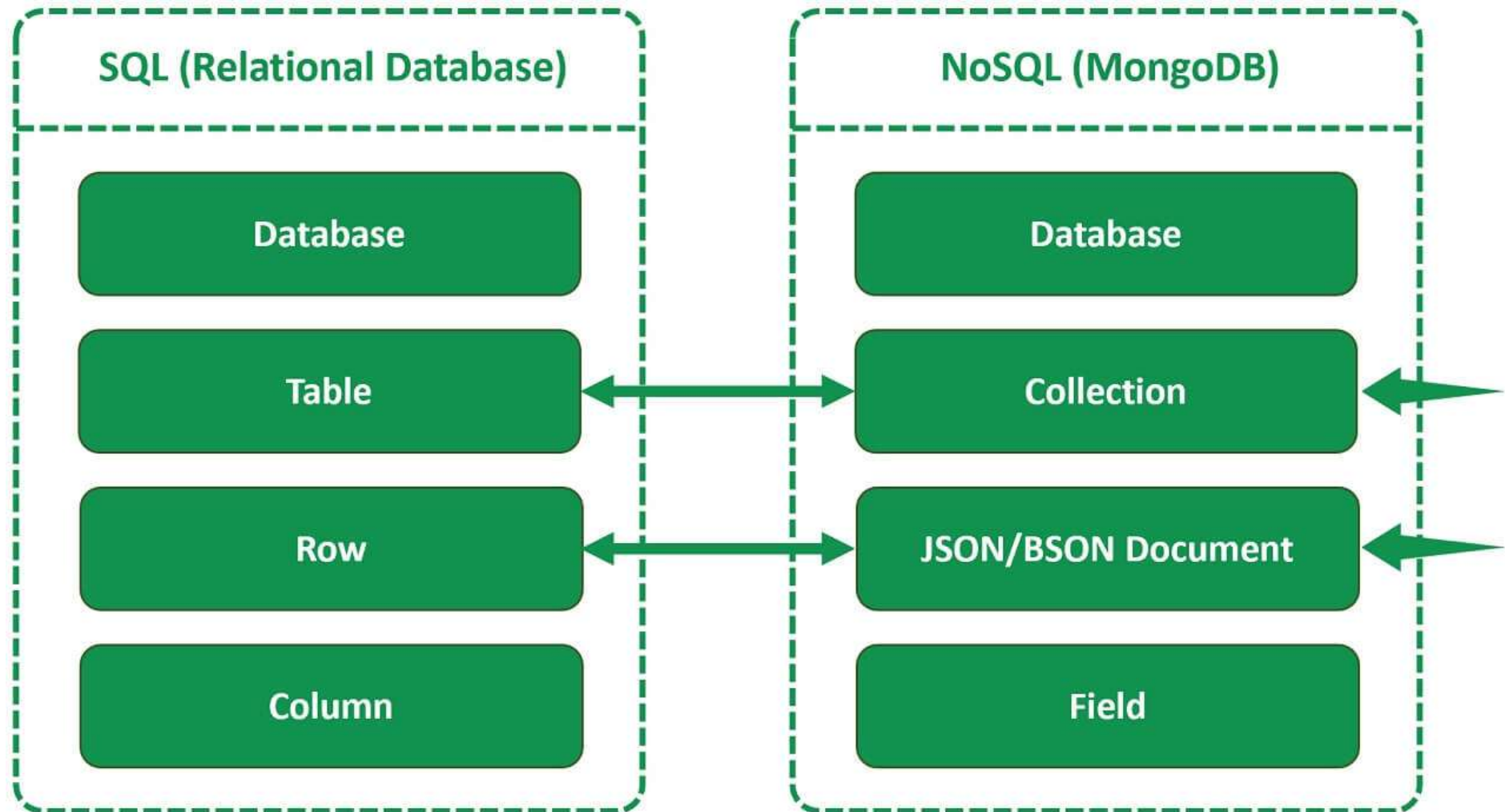
Document 3

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```


Relational Database



SQL vs NoSQL



Simple use case



Name : Sam

Courses : SQL Server, ASP.NET MVC, MongoDB



Name : Mary

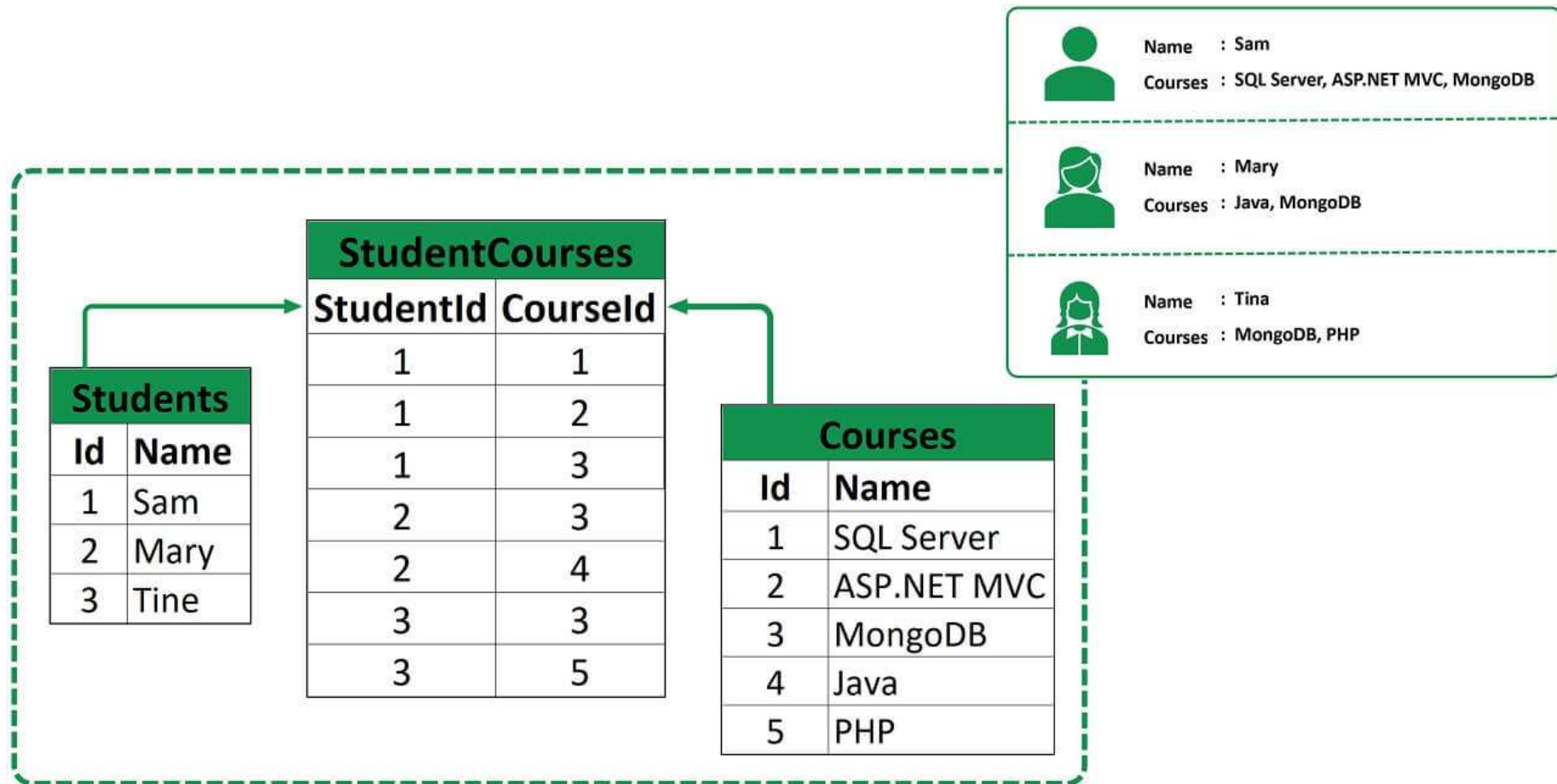
Courses : Java, MongoDB



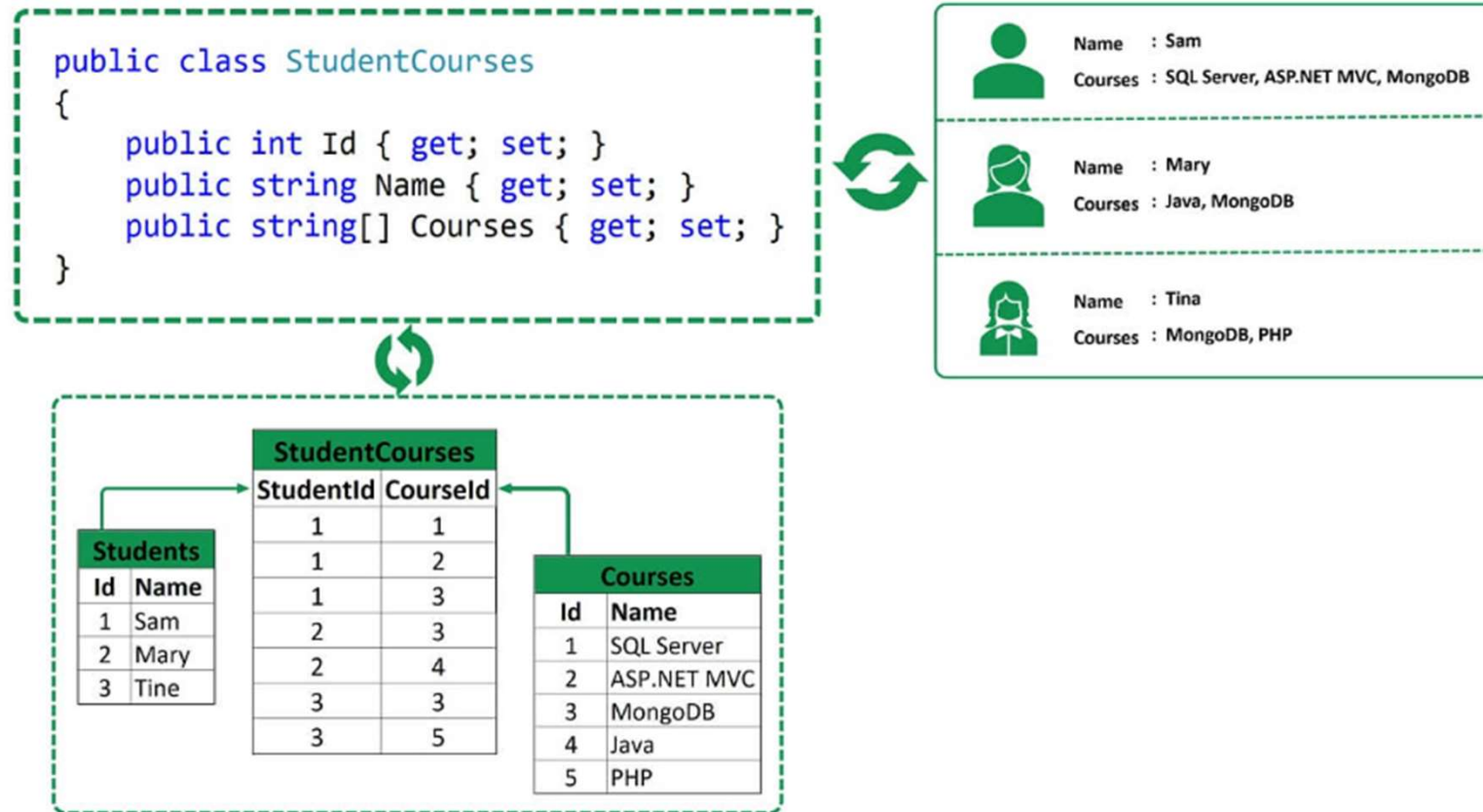
Name : Tina

Courses : MongoDB, PHP

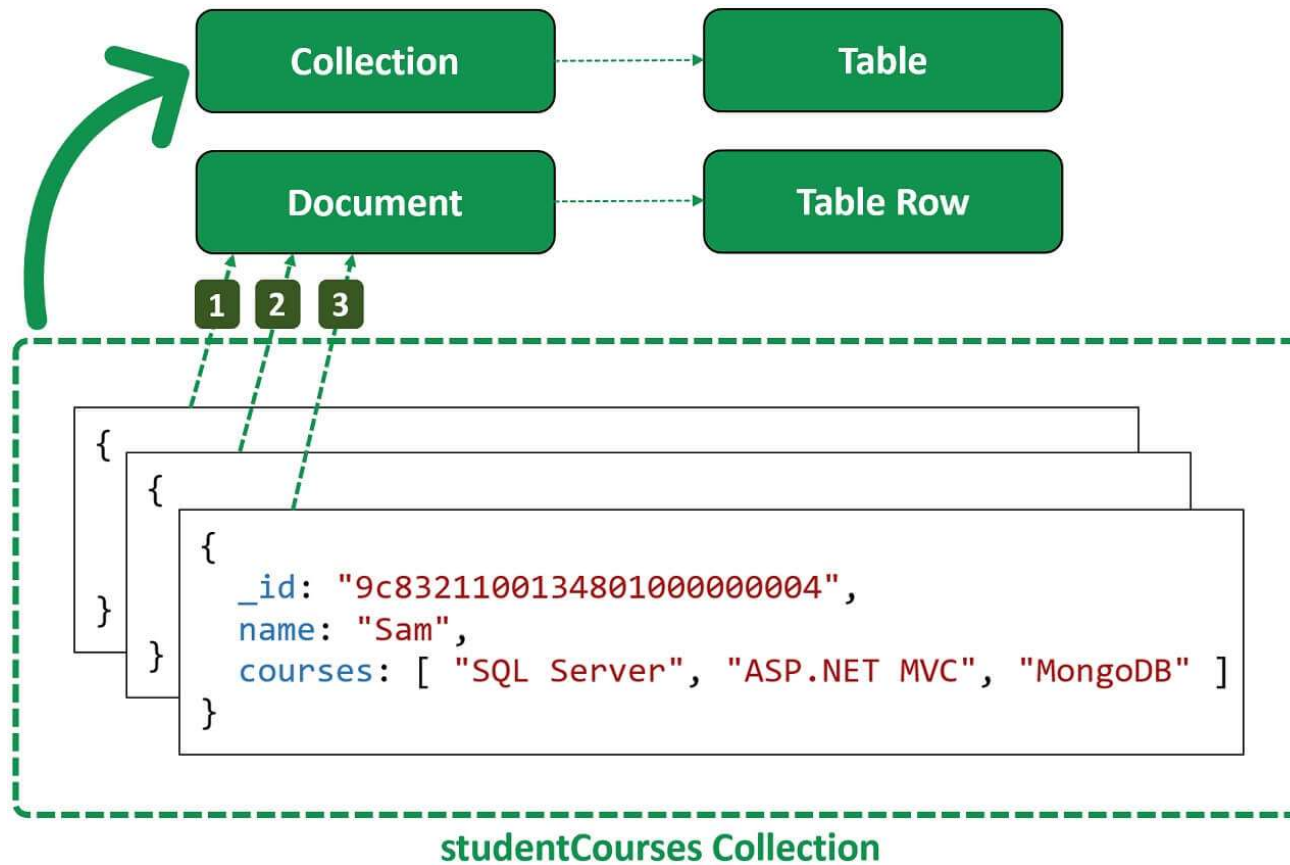
Data Modelling



Application Side



Non-relational Databases



When to use SQL database over non-relational database

- If you want a flexible schema for the data i.e in terms of shape or size, or if it needs to be open to change in the future, then a non-relational database is the answer
- non-relational databases have been designed from the ground up for the cloud, which makes them naturally good for horizontal scaling where lots of smaller servers can be spun up to handle increased load.

Partitions and data distribution

Partitions

- Refers to breaking the data in database into separate partitions.
- An allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across multiple Availability Zones
- Partition management occurs automatically in the background and is transparent to your applications.
- Improve scalability
- Improve availability
- Improve performance

Types of database partitioning

- Vertical partitioning
- Horizontal partitioning and sharding

Vertical partitioning

- Table is split by columns
 - Different columns stored on different partitions

Partition 1

id	username	city
1	theo	london
2	kee	portland
3	julian	new york
4	jasper	boston

Partition 2

id	balance
1	213
2	75444
3	645
4	342

Horizontal partitioning

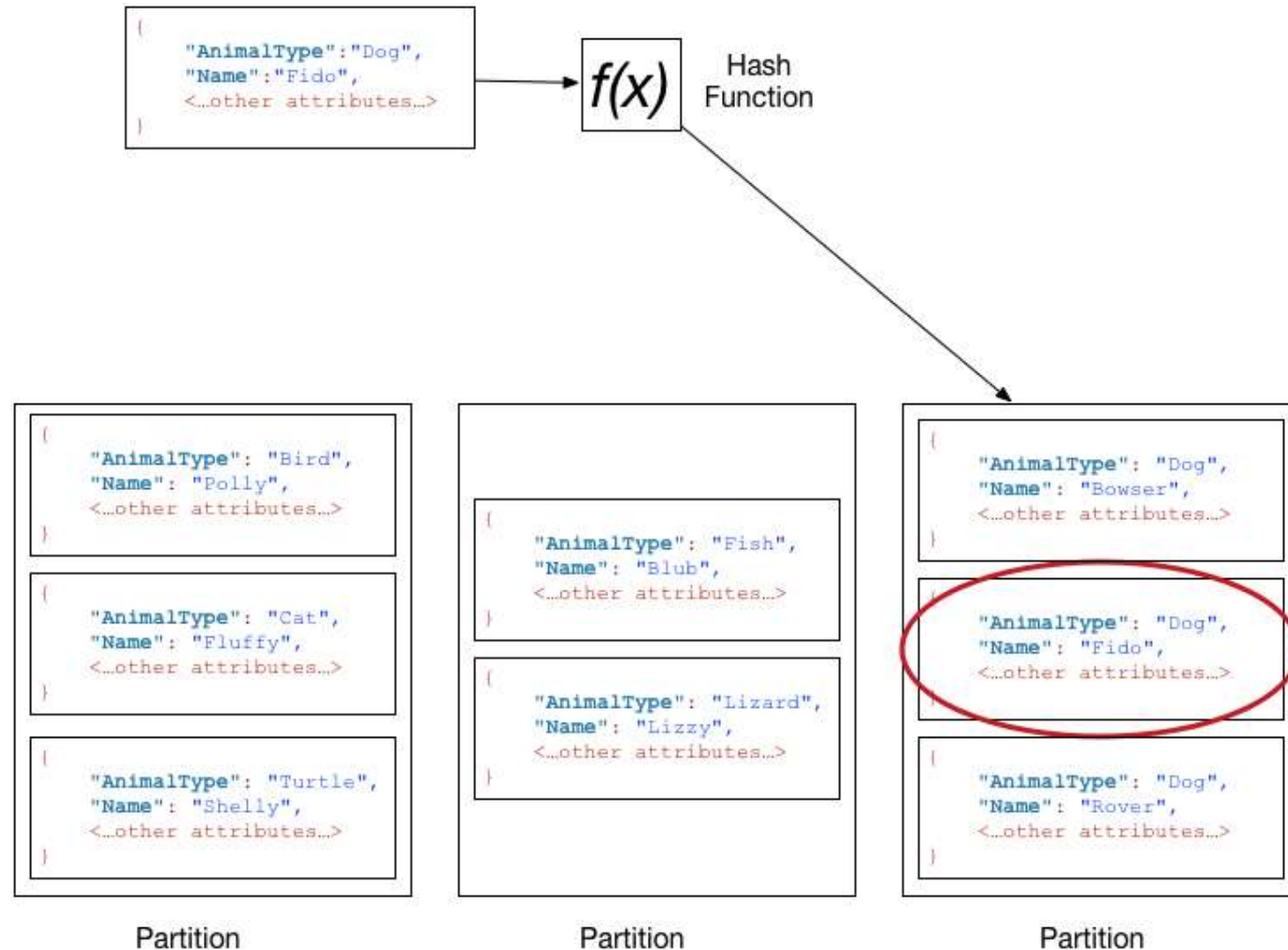
Partition 1

id	username	city	balance
1	theo	london	213
2	kee	portland	75444
3	julian	new york	645
4	jasper	boston	342

Partition 2

id	username	city	balance
501	tim	l.a.	24235
...
9998	syd	shanghai	5145
9999	nigel	santiago	4350

Data distribution: Partition key and sort key





Thanks