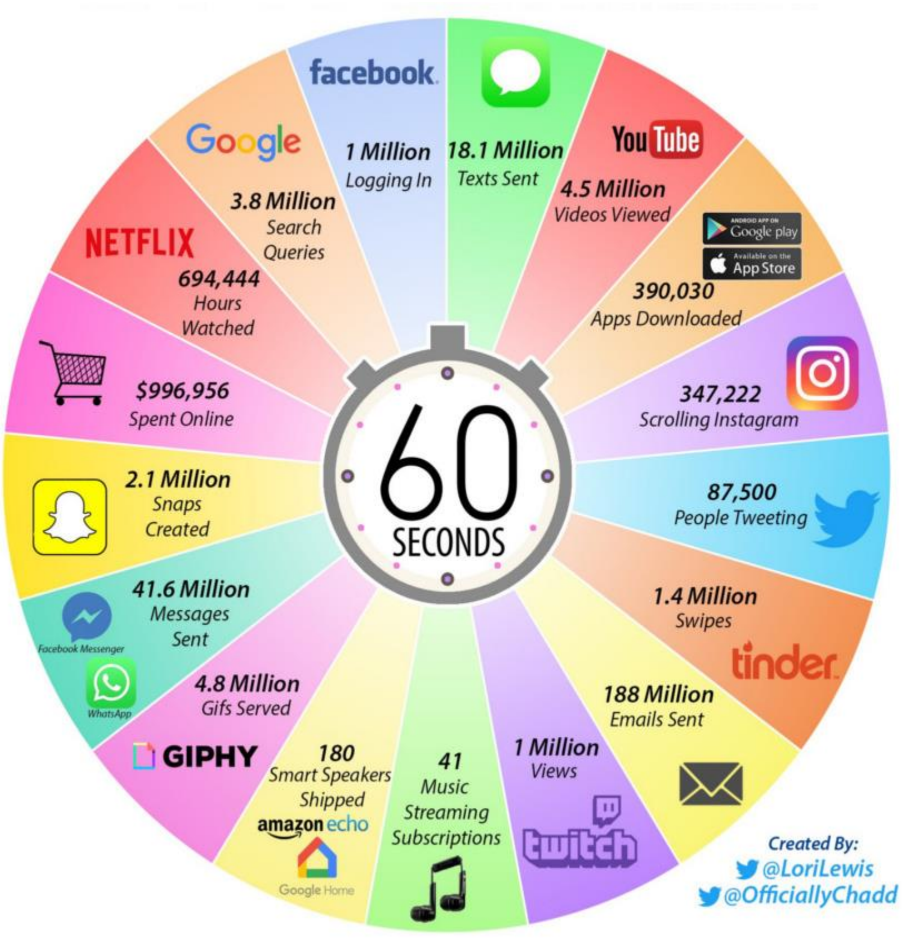
Big Data and its Implications on the Cloud

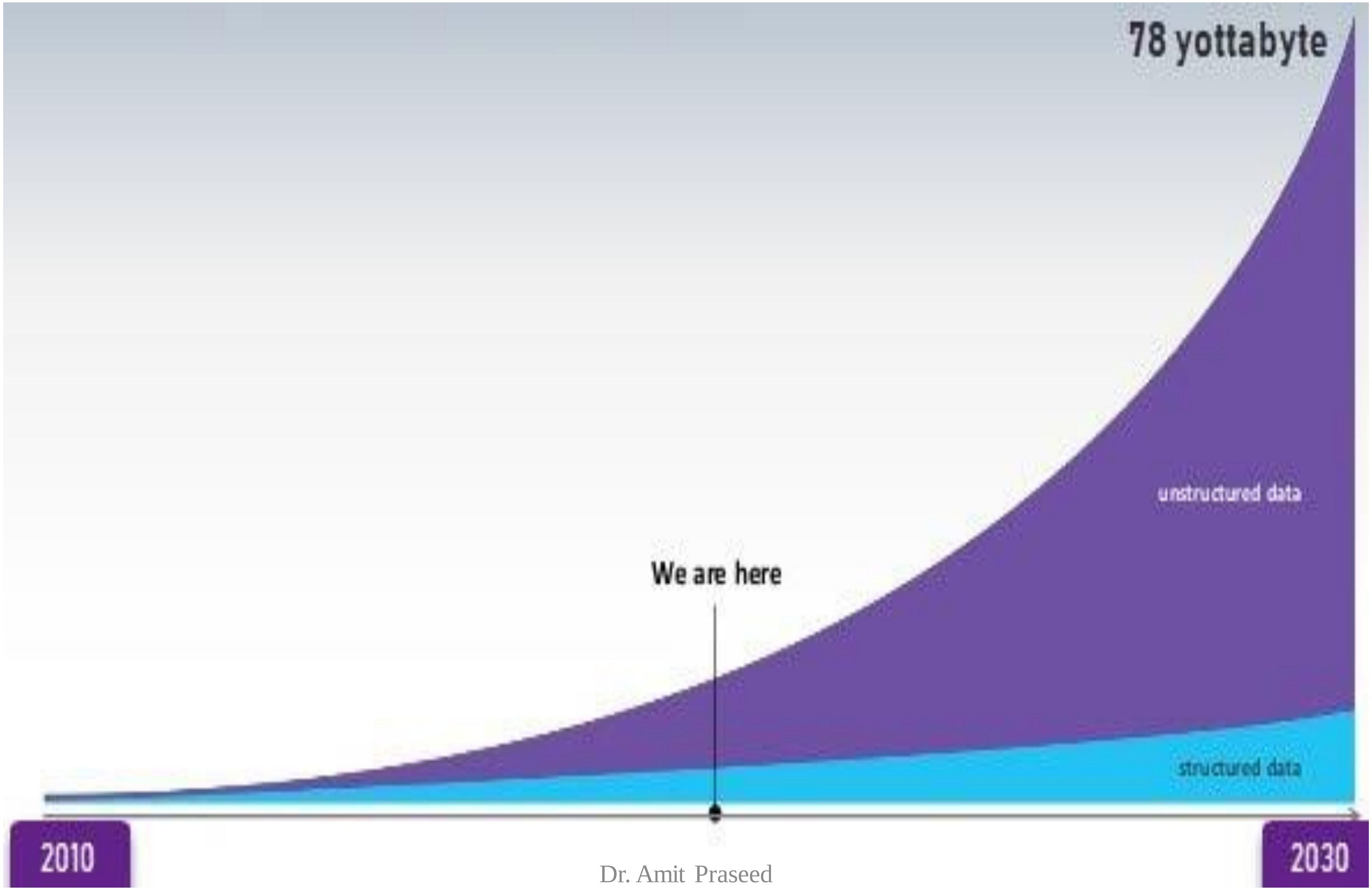
Dr. Amit Praseed

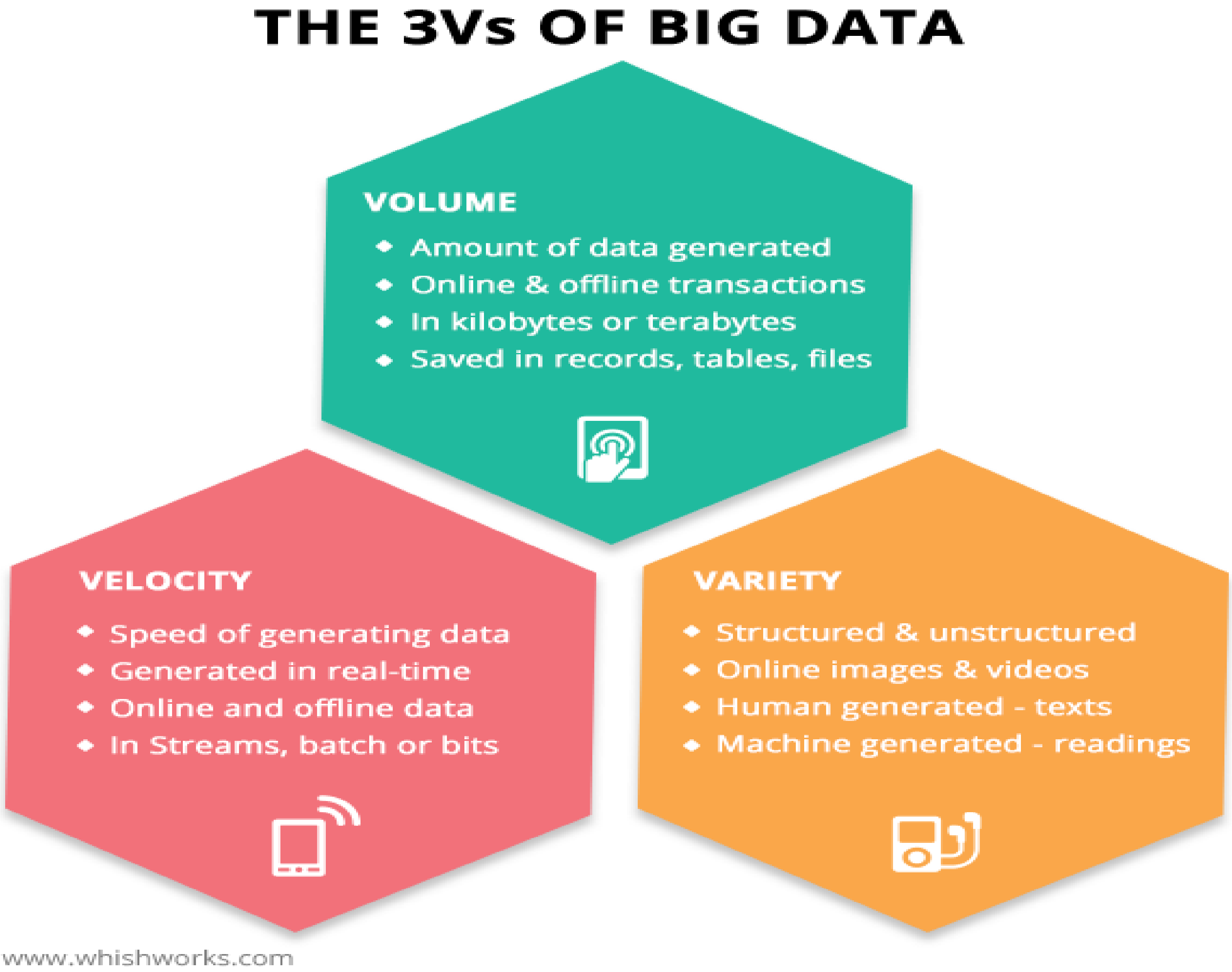
How much data do we generate?

- **Structured Data**
  - Relational Databases
  - Well defined schema
- **Unstructured Data**
  - Videos, audio, images etc.
- **Semi-structured Data**
  - structured in form but not well defined (no schema)
  - XML files



The Rise of Unstructured Data





THE 3Vs OF BIG DATA

VOLUME

- Amount of data generated
- Online & offline transactions
- In kilobytes or terabytes
- Saved in records, tables, files

VELOCITY

- Speed of generating data
- Generated in real-time
- Online and offline data
- In Streams, batch or bits

VARIETY

- Structured & unstructured
- Online images & videos
- Human generated - texts
- Machine generated - readings

www.whishworks.com

The Rise of NoSQL

- NoSQL = Not only SQL
- A fundamental shift or alternative to storing data which does not conform to the relational format
- Features
  - Schema Agnostic
  - Non relational
  - Commodity hardware
  - Highly distributable

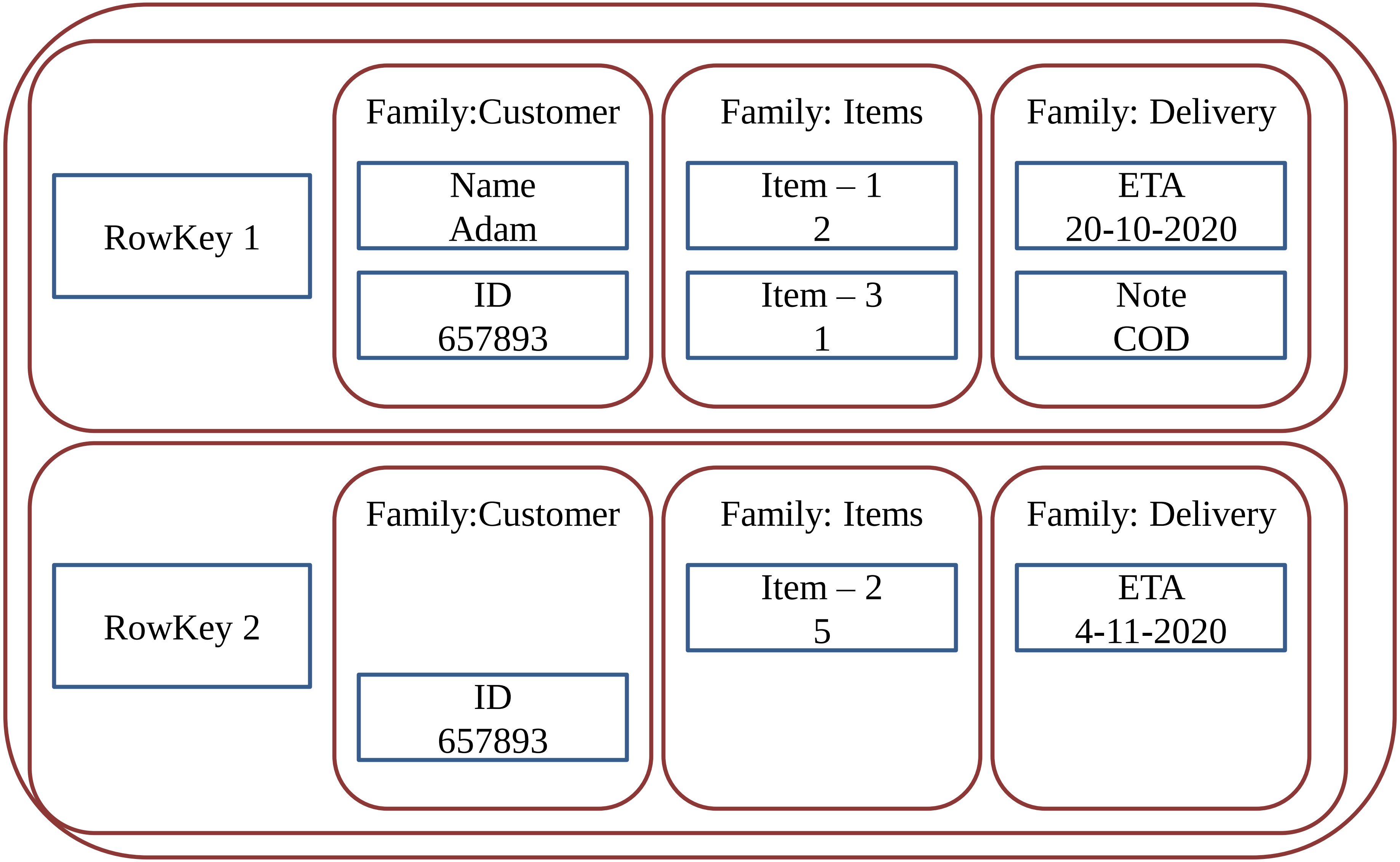
Four Core NoSQL Varieties

- Columnar
- Key-Value
- Triple
- Document

Columnar Databases

- Similar to relational model – concept of rows and columns still exist
- Optimized and designed for faster column access
- Ideal for running aggregate functions or for looking up records that match multiple columns
- A single record may consist of an ID field and multiple column families
- Each one of these column families consists of several fields. One of these column families may have multiple "rows"

Columnar Databases



RowKey 1

Family:Customer

Name
Adam

ID
657893

Family: Items

Item – 1
2

Item – 3
1

Family: Delivery

ETA
20-10-2020

Note
COD

RowKey 2

Family:Customer

ID
657893

Family: Items

Item – 2
5

Family: Delivery

ETA
4-11-2020

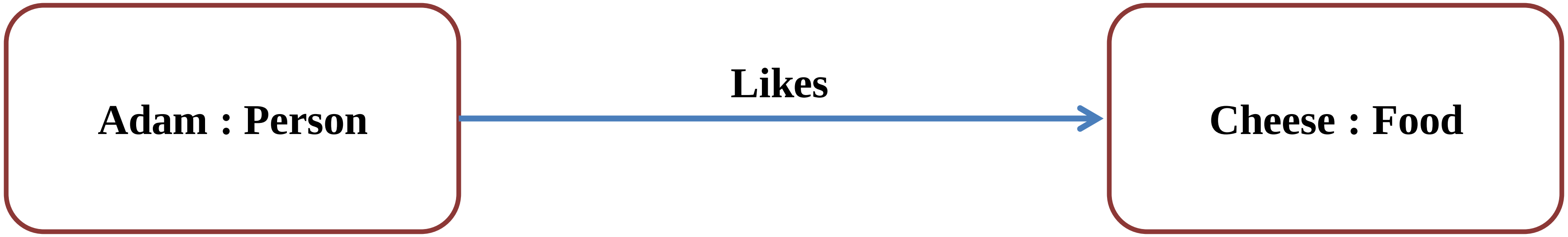
Key – Value Stores

- An ID field — the key in key-value stores — and a set of data
- Some key-value stores support typing (such as integers, strings, and Booleans) and more complex structures for values (such as maps and lists)
- Key-value stores are optimized for speed of ingestion and retrieval

Triple and Graph Stores

- Every *fact* or assertion is described as a triple of subject, predicate, and object:
  - A *subject* is the thing you're describing. It has a unique ID called an IRI. It may also have a type, which could be a physical object (like a person) or a concept (like a meeting).
  - A *predicate* is the property or relationship belonging to the subject. This again is a unique IRI that is used for all subjects with this property.
  - An *object* is the intrinsic value of a property (such as integer or Boolean, text) or another subject IRI for the target of a relationship.

Triple and Graph Stores



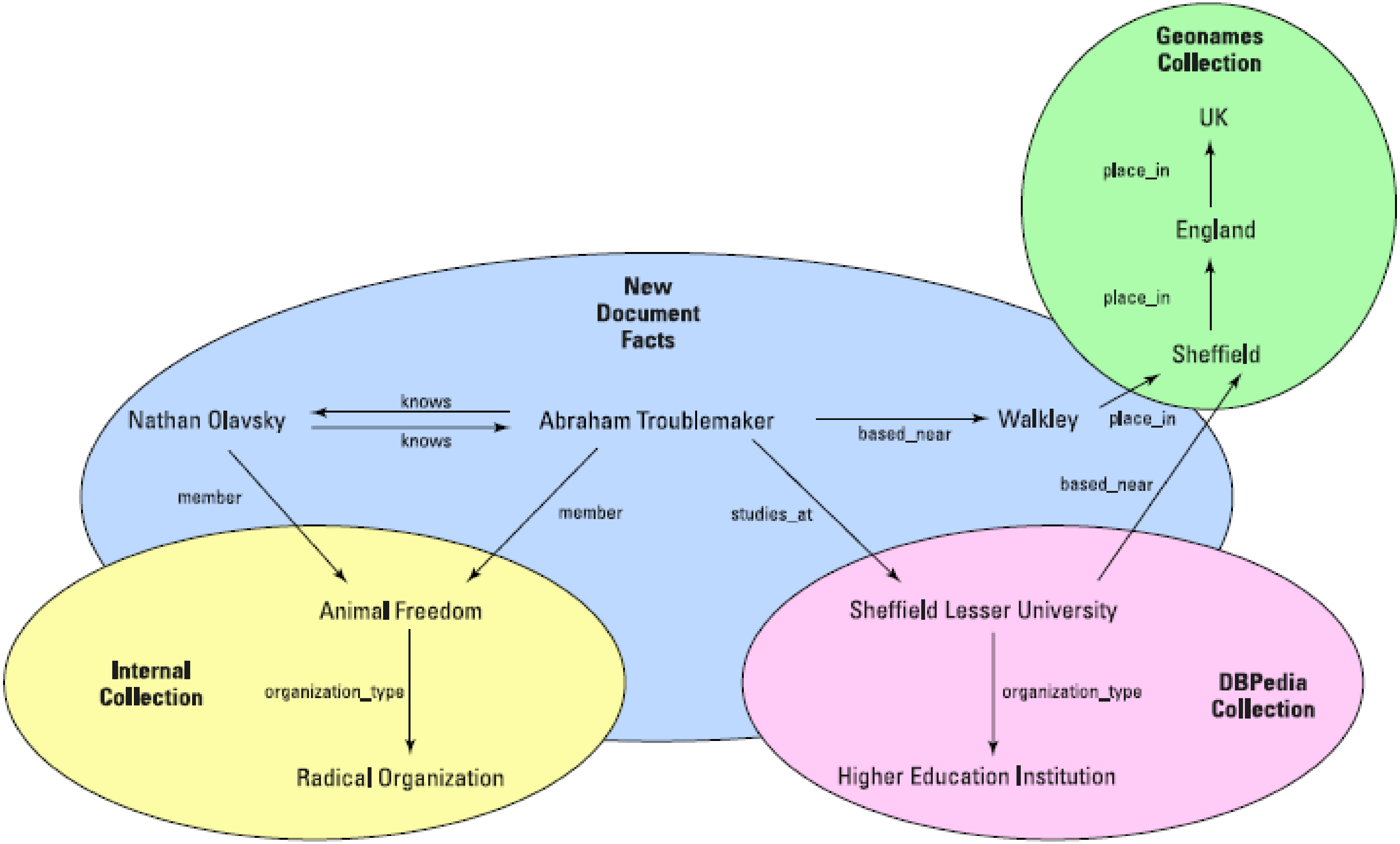
Adam : Person

Likes

Cheese : Food

- Three points of information
  - Adam is a person
  - Cheese is a food item
  - Adam likes cheese

Triple and Graph Stores



Document Stores

- Hold documents that combine information in a single logical unit
- Retrieving all information from a single document is easier with a database and is more logical for applications
- A document is any unstructured or tree-structured piece of information.
  - It could be a recipe, financial services trade, PowerPoint file, PDF, plain text, or JSON or XML document.

Examples of NoSQL Products

- **Columnar:** DataStax, Apache Cassandra, HBase, Apache Accumulo, Hypertable
- **Key-value:** Basho Riak, Redis, Voldemort, Aerospike, Oracle NoSQL
- **Triple/graph:** Neo4j, Ontotext's GraphDB (formerly OWLIM), MarkLogic, OrientDB, AllegroGraph, YarcData
- **Document:** MongoDB, MarkLogic, CouchDB, FoundationDB, IBM Cloudant, Couchbase

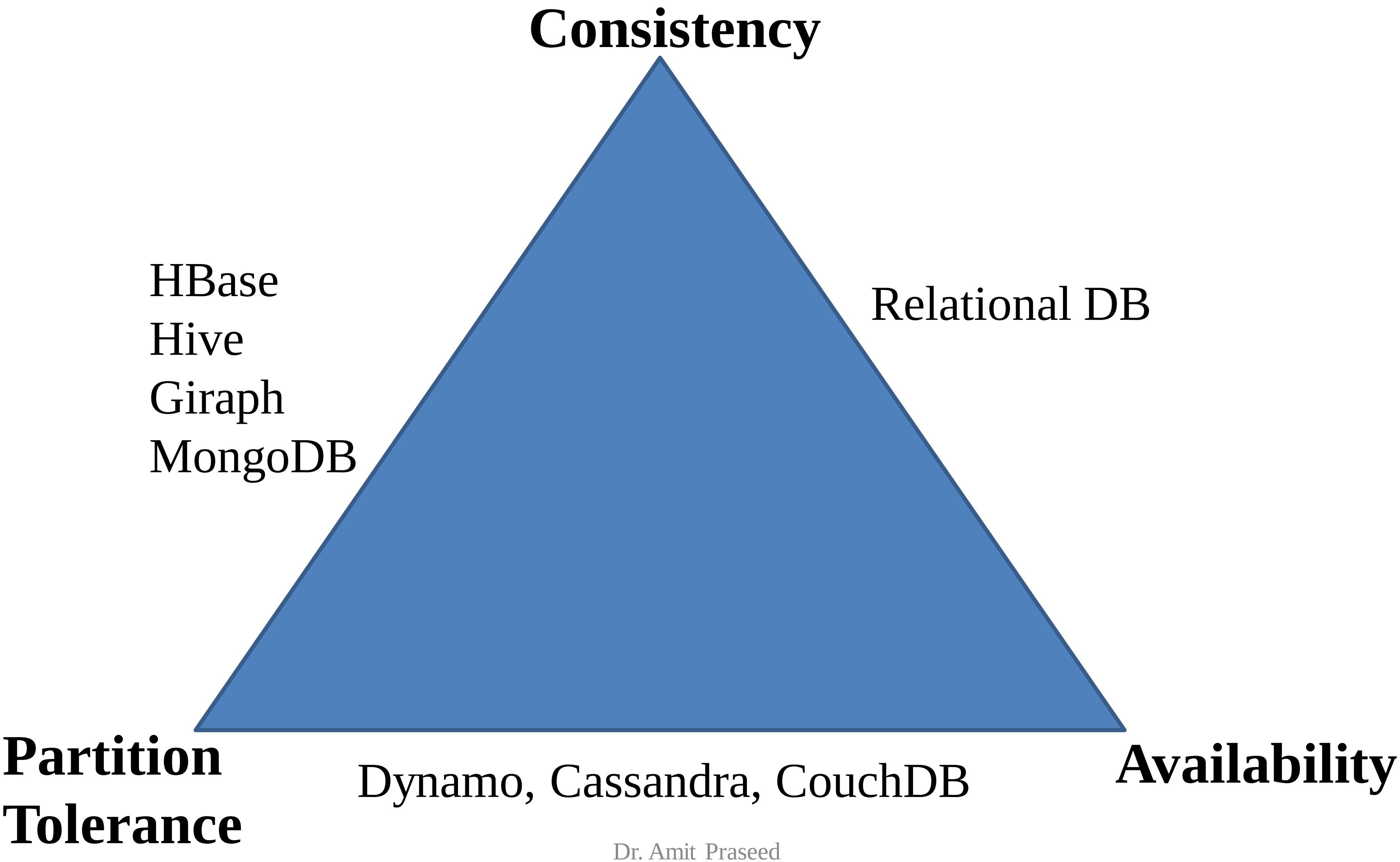
The CAP Theorem

- It is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:
  - **Consistency**: Every read receives the most recent write or an error
  - **Availability**: Every request receives a response, without the guarantee that it contains the most recent write
  - **Partition tolerance**: The system continues to operate despite an arbitrary number of messages being dropped by the network between nodes

Consistency in NoSQL

- NoSQL relaxes the ACID consistency model used in relational databases
- NoSQL uses an eventual consistency model called BASE
  - Basically Available
  - Soft state
  - Eventual consistency

ACID, BASE and the CAP Theorem



Consistency

HBase
Hive
Giraph
MongoDB

Relational DB

Partition Tolerance

Dynamo, Cassandra, CouchDB

Availability

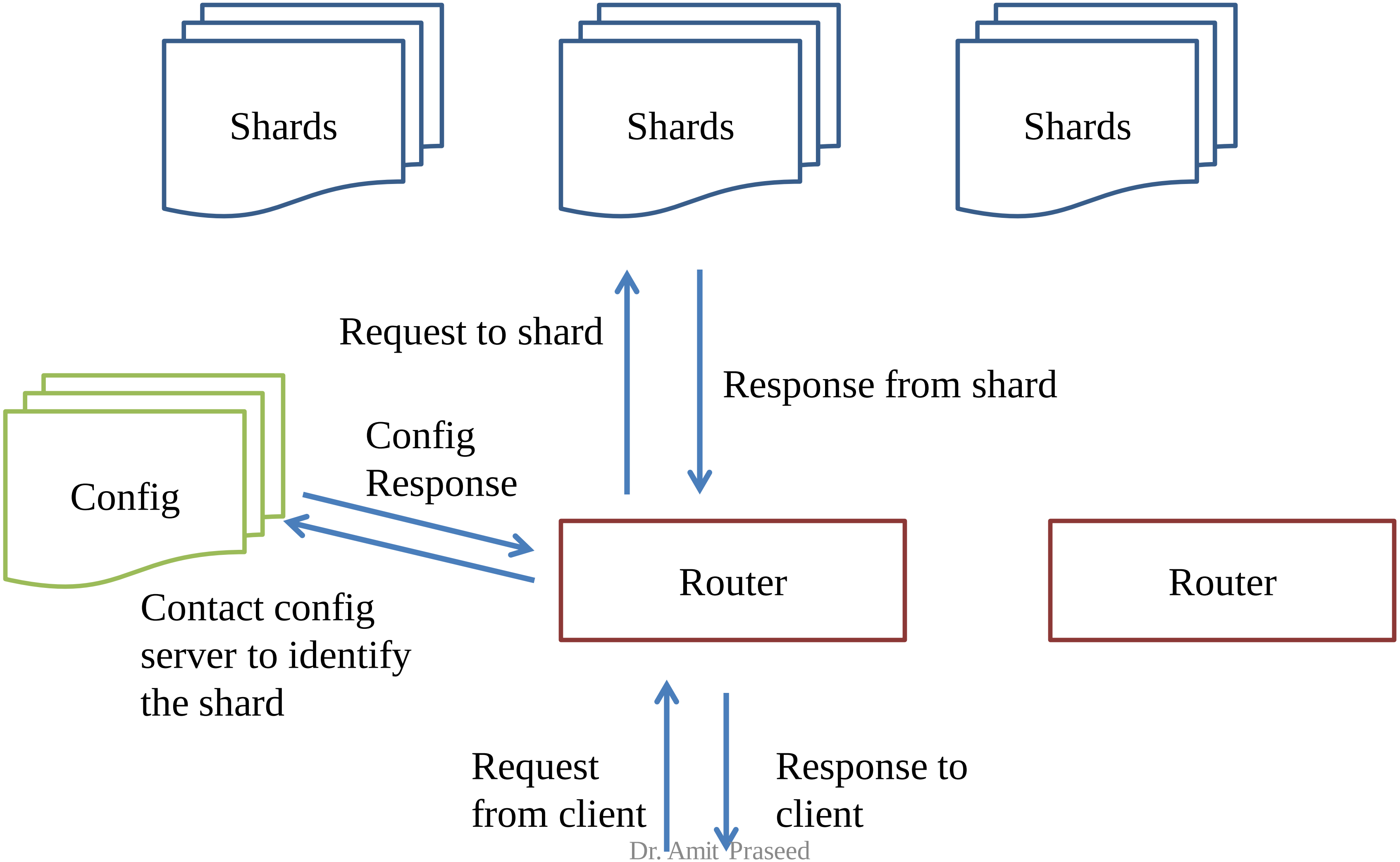
MongoDB

- A document-oriented, NoSQL database
  - Hash-based, schema-less database
  - Atomic writes and fully-consistent reads
  - Master-slave replication with automated failover (replica sets)
  - Built-in horizontal scaling via automated range-based partitioning of data (sharding)
  - No joins nor transactions
  - Consistency + Partition Tolerance

MongoDB Terminology

- A MongoDB instance may have zero or more ‘databases’
- A database may have zero or more ‘collections’
- A collection may have zero or more ‘documents’
- A document may have one or more ‘fields’
- MongoDB ‘Indexes’ function much like their RDBMS counterparts.

MongoDB Architecture



Shards

Shards

Shards

Request to shard

Response from shard

Config

Config Response

Router

Router

Contact config server to identify the shard

Request from client

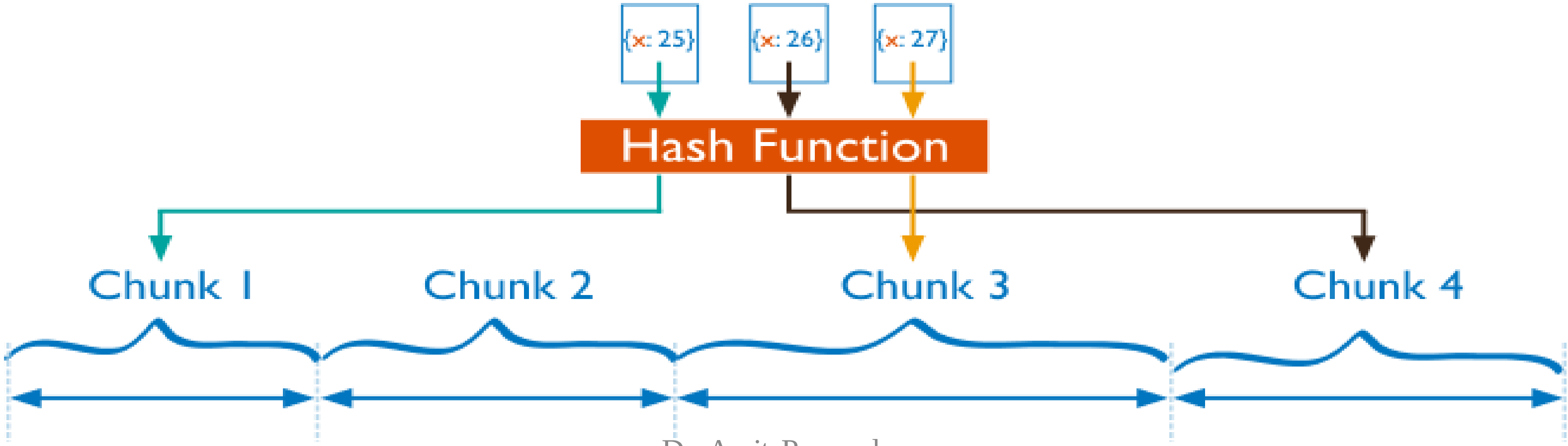
Response to client

MongoDB Replication

- A replica set is a group of mongod instances that maintain the same data set.
  - A replica set contains several data bearing nodes and optionally one arbiter node.
  - Of the data bearing nodes, one member is deemed the primary node, while the other nodes are deemed secondary nodes.
- The primary node receives all write operations.
  - The primary records all changes to its data sets in its operation log, i.e. oplog
  - The secondaries replicate the primary’s oplog and apply the operations to their data sets

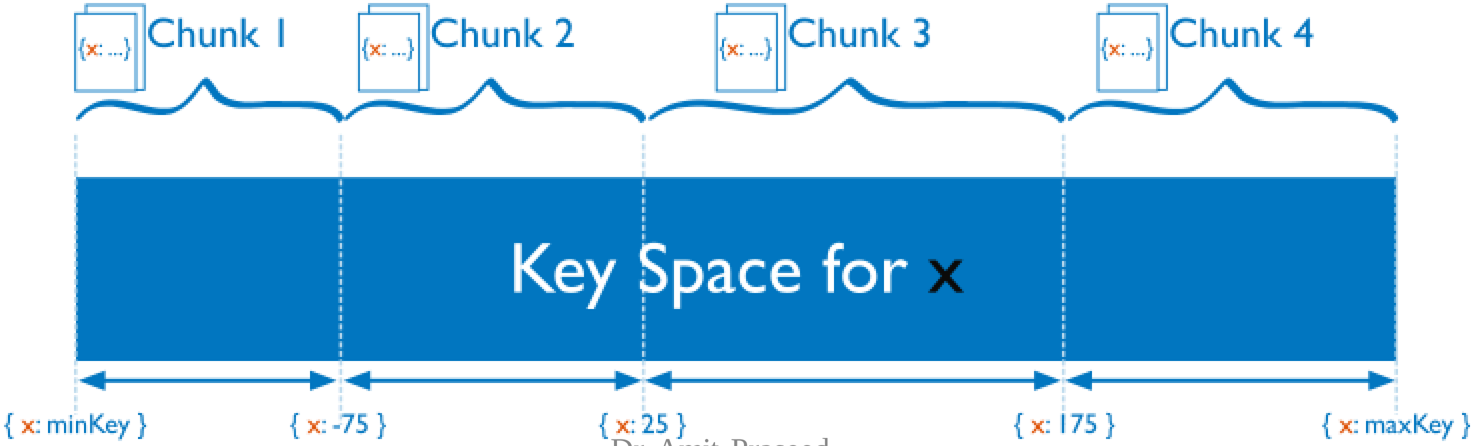
Sharding in MongoDB

- **Hashed Sharding**
  - Involves computing a hash of the shard key field’s value.
  - Each chunk is then assigned a range based on the hashed shard key values.
  - Even data distribution, but may result in more cluster wide broadcast operations



Sharding in MongoDB

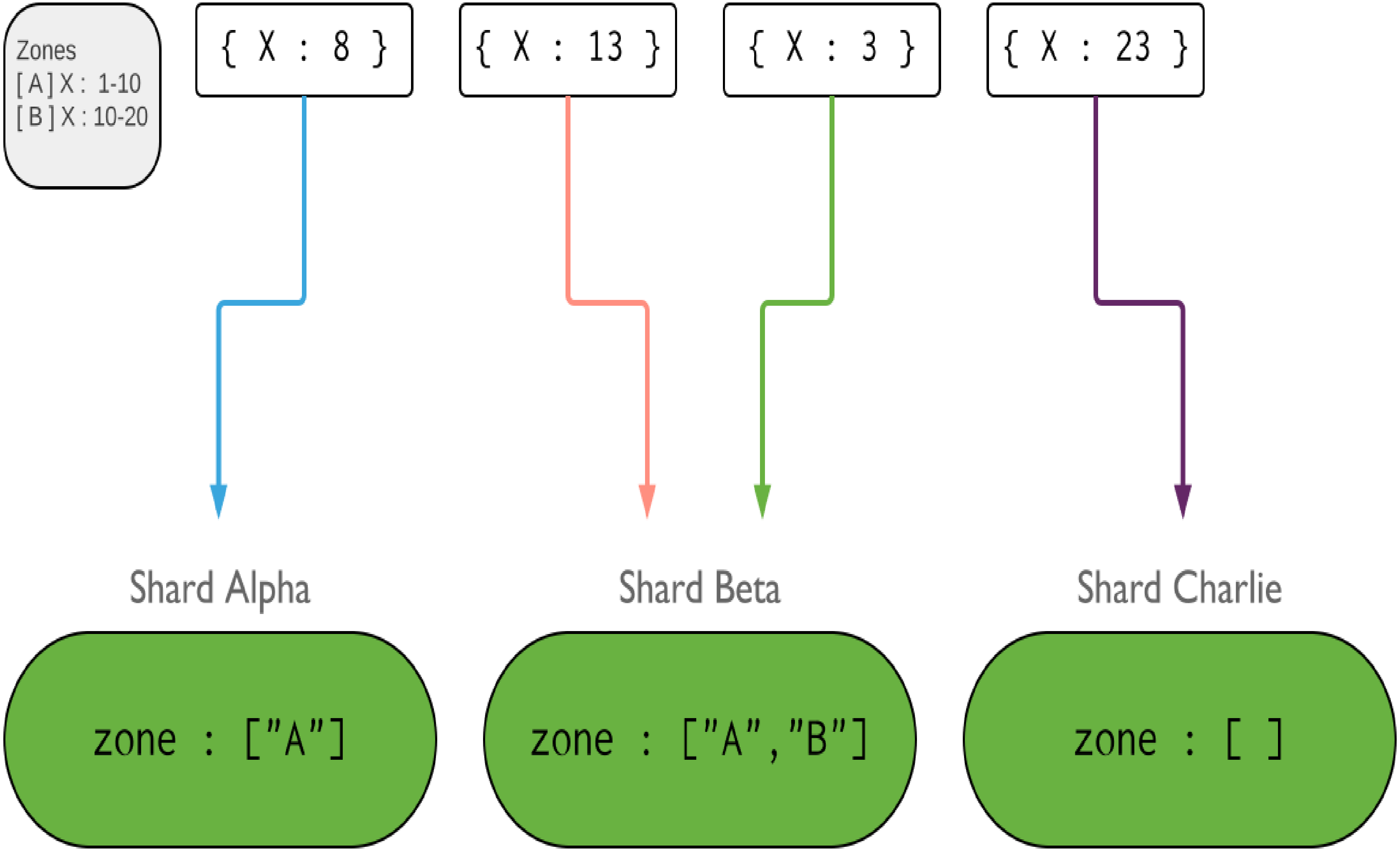
- **Ranged Sharding**
  - Involves dividing data into ranges based on the shard key values.
  - Each chunk is then assigned a range based on the shard key values
  - Allow targeted operations, but poorly chosen shard key may result in uneven data distribution



Sharding in MongoDB

- **Zoned Sharding**
  - Zones can help improve the locality of data for sharded clusters that span multiple data centers.
  - In sharded clusters, you can create zones of sharded data based on the shard key.
  - You can associate each zone with one or more shards in the cluster. A shard can associate with any number of zones.
  - In a balanced cluster, MongoDB migrates chunks covered by a zone only to those shards associated with the zone.
  - Each zone covers one or more ranges of shard key values

Zoned Sharding



Choosing a Shard Key

- The shard key is either an indexed field or indexed compound fields that determines the distribution of the collection’s documents among the cluster’s shards.
- All sharded collections must have an index that supports the shard key; i.e. the index can be an index on the shard key or a compound index where the shard key is a prefix of the index
- The choice of shard key affects the creation and distribution of the chunks across the available shards. This affects the overall efficiency and performance of operations within the sharded cluster.