Newer Data Models

System Analysis & Design

RELATIONAL DATA MODELS: BENEFITS

- Strong mathematical basis
- Declarative syntax
- Well known language SQL
- Adhere to ACID properties

ACID

- An important aspect of relational databases which guarantees the reliability of transactions is their adherence to the ACID properties:
- **Atomicity**: Either all parts of a transaction must be completed or none.
- **Consistency**: The integrity of the database is preserved by all transactions. The database is not left in an invalid state after a transaction.
- **Isolation**: A transaction must be run isolated in order to guarantee that any inconsistency in the data involved does not affect other transactions.
- **Durability**: The changes made by a completed transaction must be preserved or in other words be durable.

RELATIONAL DATA MODELS: LIMITATIONS

- Applications using normalized database schema require join's,
  - which doesn't perform well under lots of data and/or nodes
- Have to pre-define a schema that is fixed and hard to change
  - Harder to handle unstructured data
- Typically best designed to run on a single server in order to maintain integrity of table mappings
  - Existing RDBMS clustering solutions can scale-up (on a single server) which is limited & not really scalable when dealing with exponential growth. The latter requires scale out (multiple servers)

NEW DEVELOPMENTS

Web introduces a new scale for applications

- Concurrent users (millions of reqs/second)
- Data (peta-bytes generated daily)
- Processing (all this data needs massive processing)
- Exponential growth (surging unpredictable demands)

- **Big Data**
  - Volume, variety, velocity
  - Web sites with very large quantity of unstructured data have no way to deal with this using existing RDBMS solutions

THE NEW IDEA

- New DBMS capable of handling data at web scale (distributed)
- Catch: CAP theorem or Brewer's theorem. Distributed DBMS have to manage the following three, but *can only guarantee two out of following three*
  - **Consistency** (system is in a consistent state after an operation – all clients see the same data)
  - **Availability** (system has no downtime, node failure tolerance, upgrade tolerance)
  - **Partition-tolerance** (system continues to function even when split into disconnected subsets, for reads and writes)
- Different types of databases pick a different pair to guarantee

CAP EXAMPLE

- When you have a lot of data which needs to be highly available,
  - you may partition it across machines
  - and also replicate it to be more fault-tolerant
- This means, that when writing a record, all replicas must be updated too
- Now you need to choose between:
  - Lock all relevant replicas during update => be less available
  - Don't lock the replicas => be less consistent

THE SOLUTION: NON RELATIONAL DB

- BigTable (developed at Google)
- Hbase (developed at Yahoo!)
- Dynamo (developed at Amazon)
- Cassandra (developed at FaceBook)
- Voldemort (developed at LinkedIn)
- a few more: Riak, Redis, CouchDB, MongoDB, Hypertable

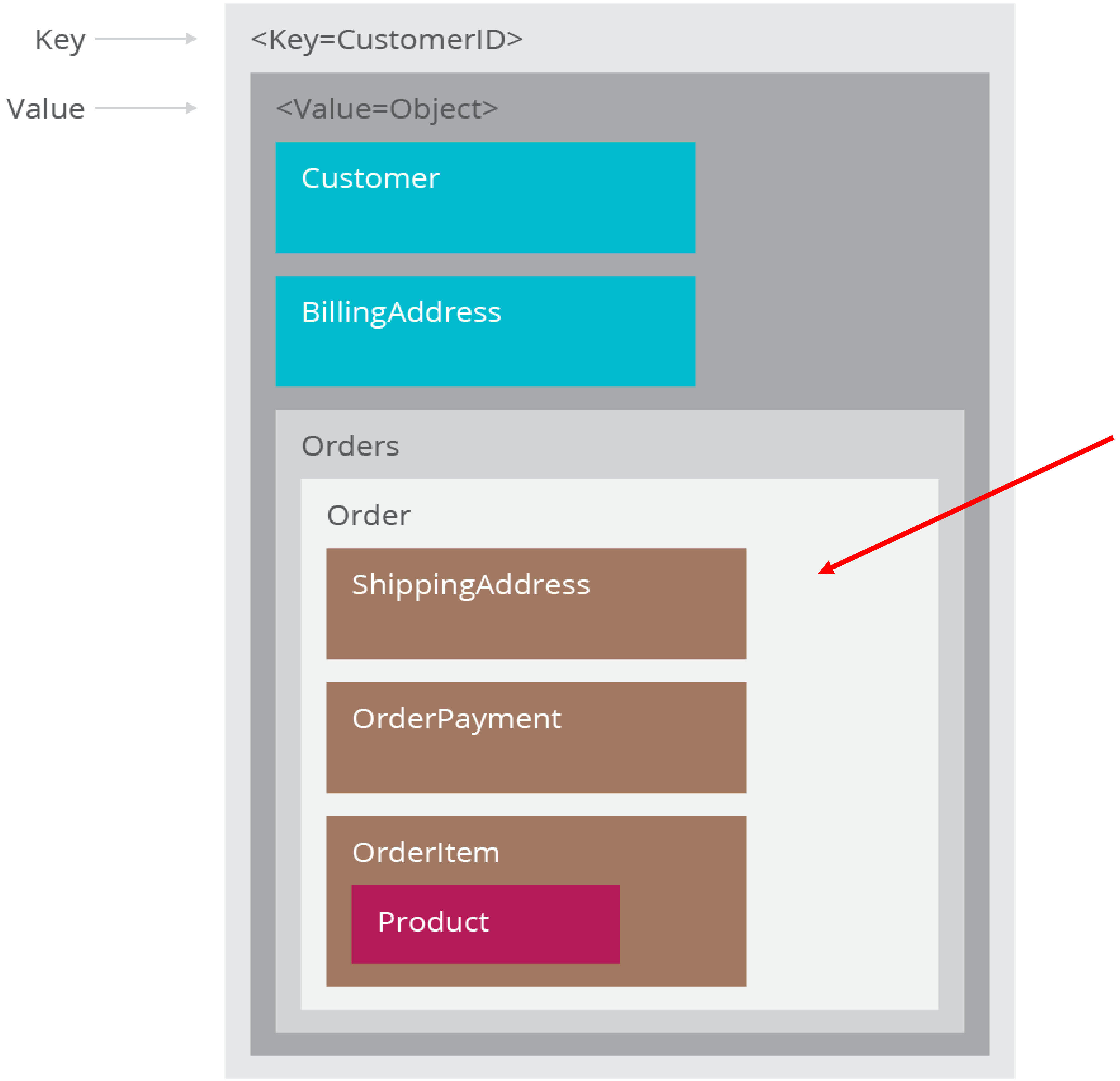
NOSQL DATABASES

- Key- value databases
  - Key is a hash (no duplicates), data is stored as a value that is a binary object BLOB that is non query-able
- Document databases
  - Also Key-value stores, but value is a complex data structure that is query-able (e.g., XML document)
- Columnar databases (BigTable clones)
  - Stores data in columns instead of rows
  - Each key is associated with many attribute /columns
  - Handles structured data
- Graph database
  - Nodes correspond to entities and edges are relationships.
  - Can have multiple types of relationships across nodes
  - SPARQL query language

EXAMPLES OF NOSQL

- Key-value
  - Riak
  - Redis
  - Memcached
  - upscaleDB
  - Amazon DynamoDB
  - Project Voldemort
  - Couchbase
- Column-family
  - Cassandra
  - Hbase
  - HyperTable
  - Amazon DynamoDB
- Document
  - MongoDB
  - CouchDB
  - RavenDB
  - OrientDB
  - Lotus Notes
- Graph
  - Neo4J
  - Infinite Graph
  - OrientDB
  - FlockDB
  - HyperGraphDB

KEY VALUE



A blob of data: where the value is not examinable.

Not good when we want to query the values

Source: https://www.thoughtworks.com

DOCUMENT

XML, JSON, BSON (binary JSON used in MongoDB)

Data stored in a self-describing tree-like structure

<Key=CustomerID>

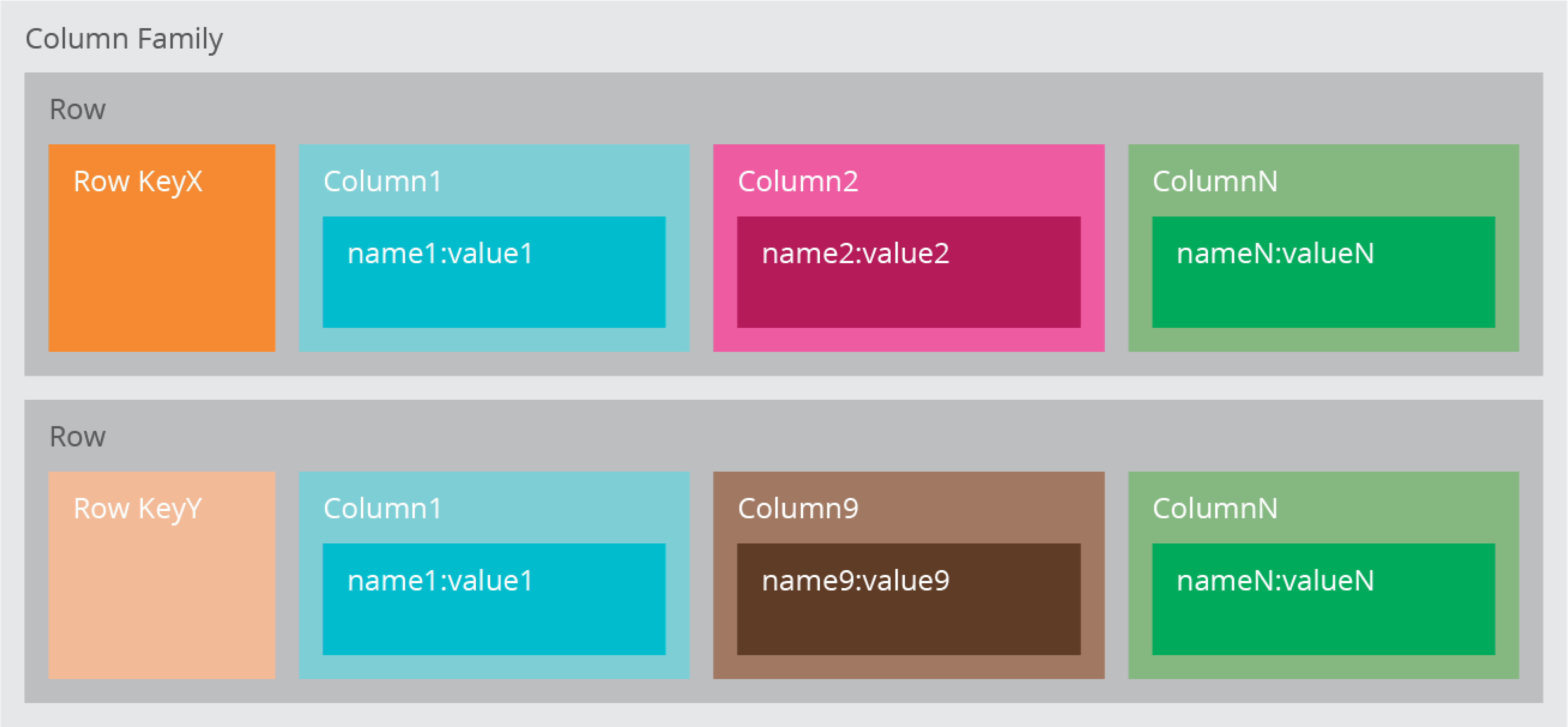
```
{
  "customerid": "fc986e48ca6"
  "customer":
  {
  "firstname": "Pramod",
  "lastname": "Sadalage",
  "company": "ThoughtWorks",
  "likes": [ "Biking","Photography" ]
  }
  "billingaddress":
  { "state": "AK",
    "city": "DILLINGHAM",
    "type": "R"
   }
}
```

Key

The value is examinable.

Source: https://www.thoughtworks.com

COLUMNAR



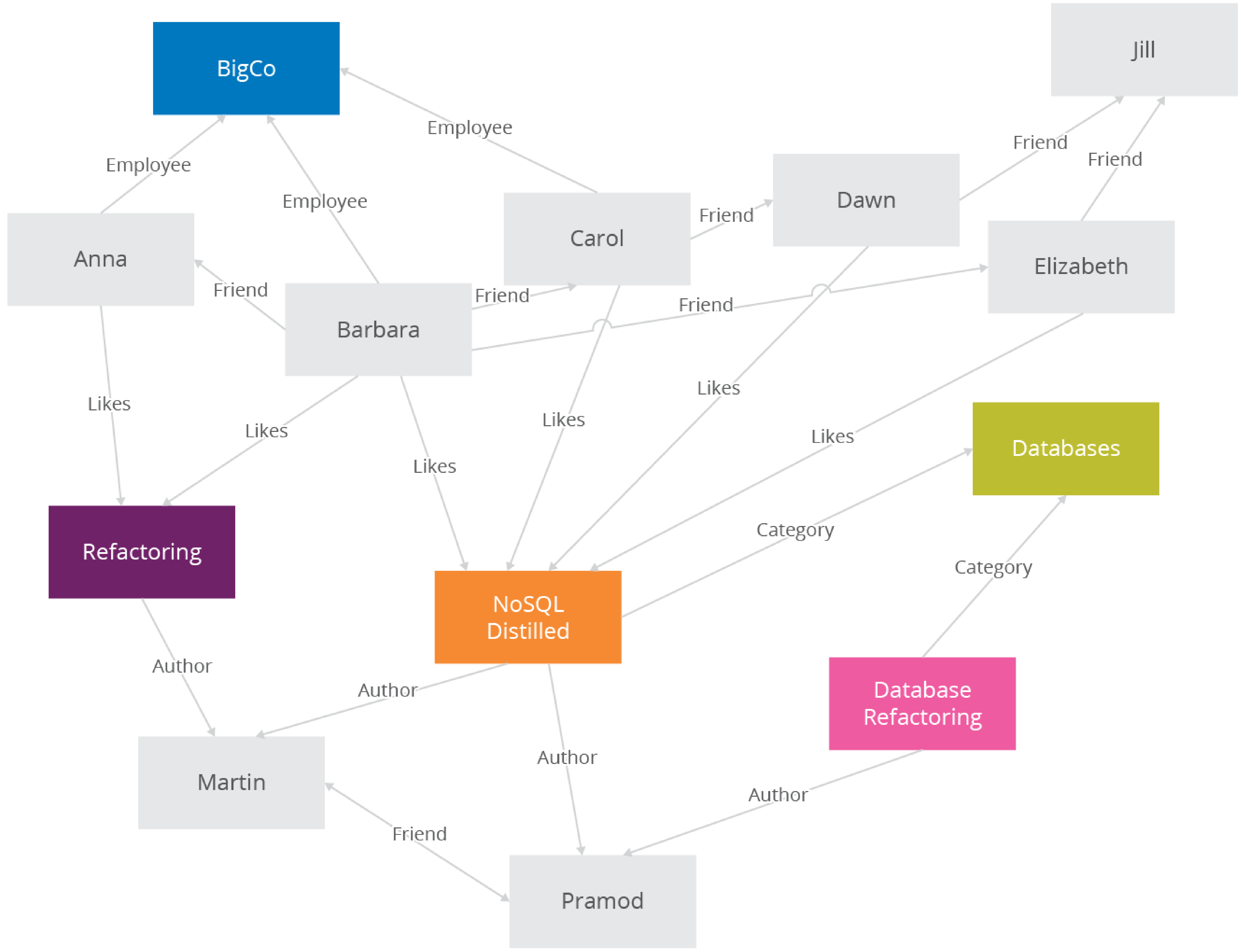
Data that are frequently accessed together in a row, but spread across columns are stored using a common key

For a customer, we would often access all the profile columns together, but not necessarily their orders as well

Each collection of columns can be of varying sizes and columns can be added into any collections

Source: https://www.thoughtworks.com

GRAPH



nodes (objects) and edges (directional relationships)
data can be interpreted in different ways

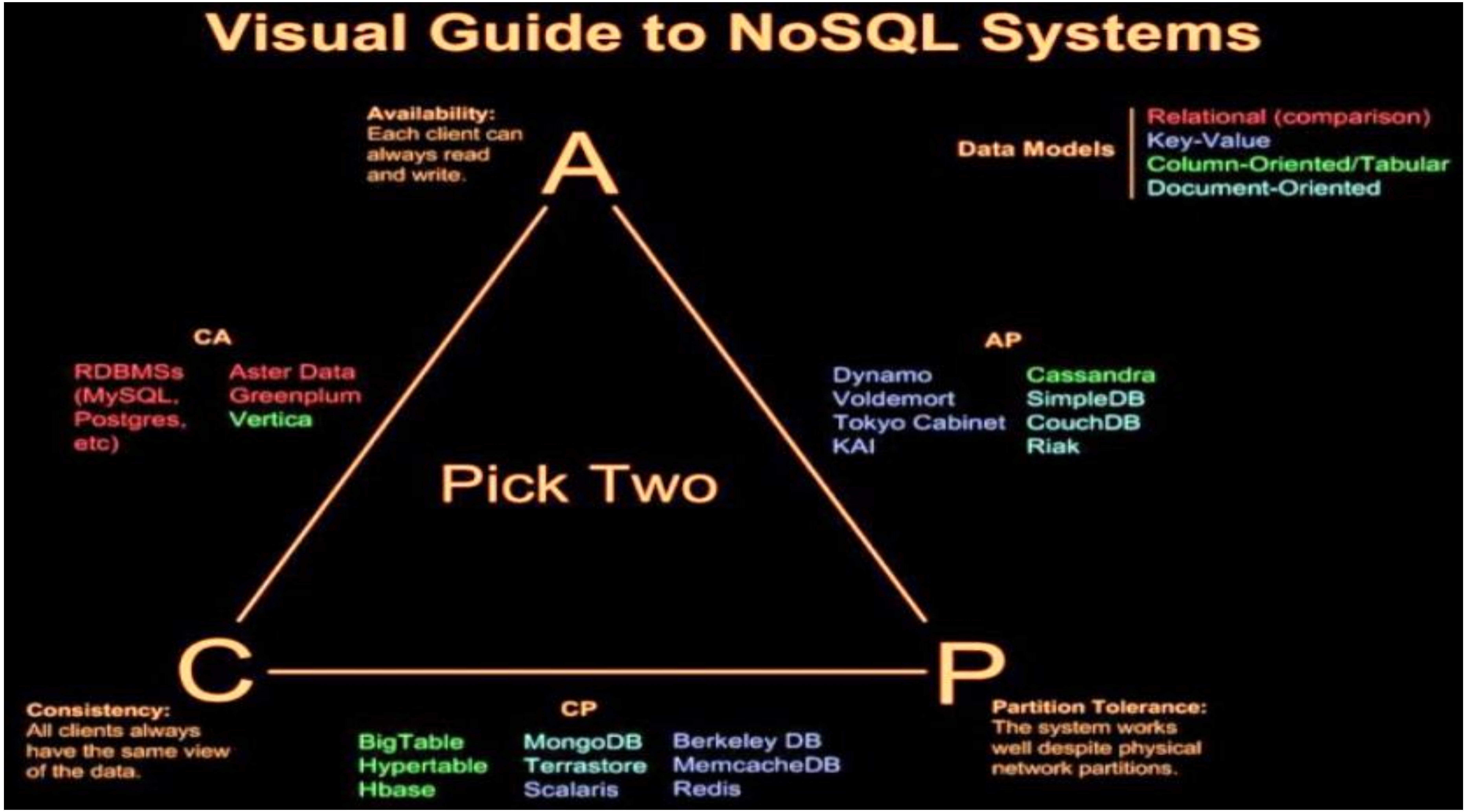
Good for connected data

Source: https://www.thoughtworks.com

PROPERTIES OF NOSQL

- No predefined schema
  - Records can have different fields as necessary
- Data is distributed
  - Needs parallel processing
- Support replication
  - Option of having replicas of a server to provide reliability
- Use some type of Map-Reduce algorithms
  - For performing a specific function on an entire dataset and retrieving the result
  - Map() – filtering and sorting: assign tasks to subnodes
  - Reduce() – summarizing operation – combine results from subnodes
- They do not guarantee ACID properties
  - Replication creates issues with synchronization, which can result in a secondary node becoming primary but not have up-to-date copy of data.

VISUAL GUIDE TO NOSQL



David Eccles School of Business

THE UNIVERSITY OF UTAH