**NOSQL**

**Disadv of RDBMS**

* + Expensive – Storage need to be increased every time basing on requirement
  + Dynamic scalability – Not Easy to increase or decrease capability Dynamically
  + Can not serve Big Users
  + Can not serve BigData – Semi structured and unstructured
    - Product table
    - Mobile product columns – id name price ram camera battery
    - Cloths – id name price length discount
    - So each product requires different columns , where no.of columns may become empty for many products
    - So what ever columns require only insert that data
  + Rigid schemas - schema is not flexible i.e changing the schema not easy
  + At the time of scaling the RDBMS it will causes lot of disturbance to a web Applications that are interacting with that Databases

**Currents Trends in market**

**Big Data** – nature of data is changing as shopping , entertainment , getting data from sensors…

Much of this data is semi structured or unstructured data.

Rigid Schema makes it impossible to incorporate the new types of data and poor fit for

Structured and semi structured data

**Big users -** No. Of Users Accesing the Database at a time

**Cloud Computing** - Resources need to be scaled out horizontally which is dynamic Scalable

Performance

**Maximum resource optimization**

Disadv of RDBMS cause the evolution of **No-Sql**

* No- sql can scale out horizontally , ie working on multiple servers with sharing load
* No-Sql Can operate across hundreds of servers peta bytes of data and billions of docs and still

manage to process 10000 transcns per second

* Can do all things inexpensive commodity servers which has cheaper hardware operating on any environment
* We can quickly add or remove Servers quickly ,
* Data is automatically rebalanced when nodes are added or removed

**No Sql –**

No Sql is a Database mechanism that used to store and retrieve the data except Tabular Format

Generally NO-SQL Databases Compromise in the concept of Consistency by considering CAP theorem

They consider mainly Availability and Partition Tolerance

**Can use No-Sql in cases**

Flexible Schema required

Not Required ACID Properties

Does not require ORM

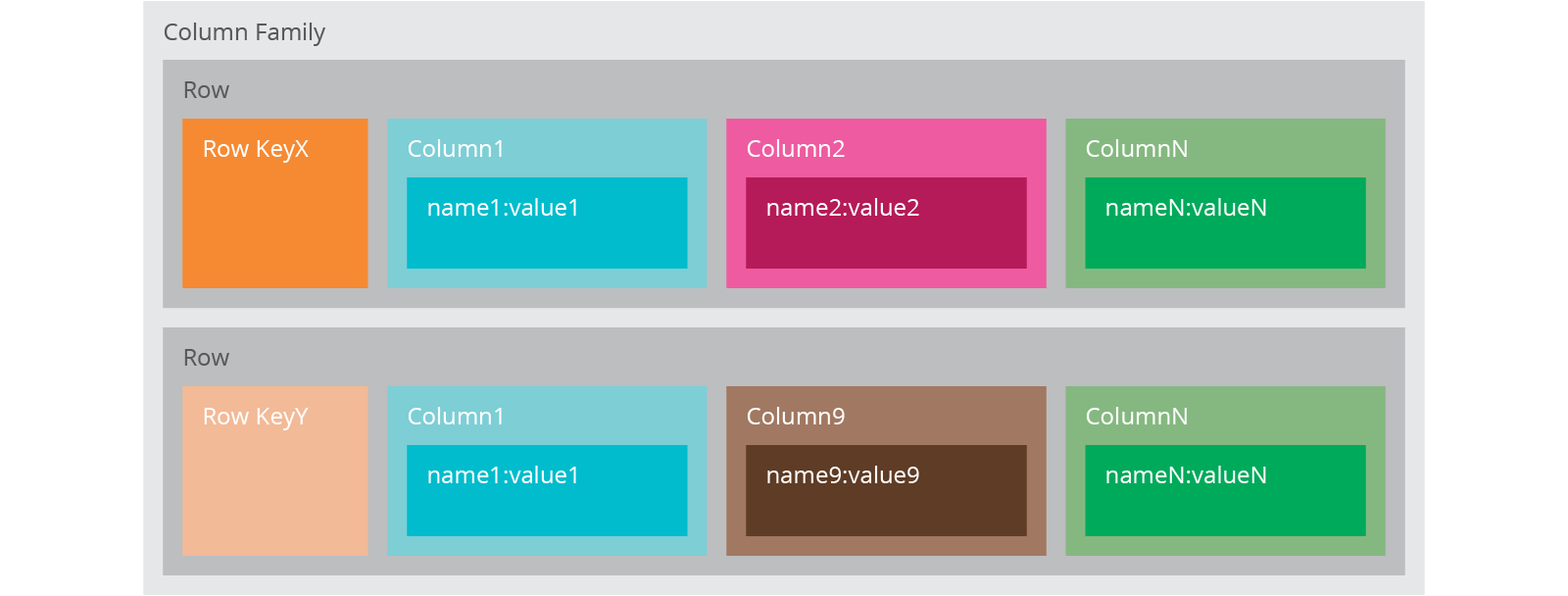
Not a Financial Data

**Data Model Types**

**Wide Column Store**

Column-family databases store data in column families as rows that have many columns associated with a row key

Column families are groups of related data that is often accessed together.



Column family databases are generally useful for Maintaining Historical Data , CMS, blogging platforms, maintaining counters, expiring usage, heavy write volume such as log aggregation.

We would avoid using column family databases for systems that are in early development, changing query patterns.

Ex:**Hadoop, Hypertable, Cassandra,Scylla**

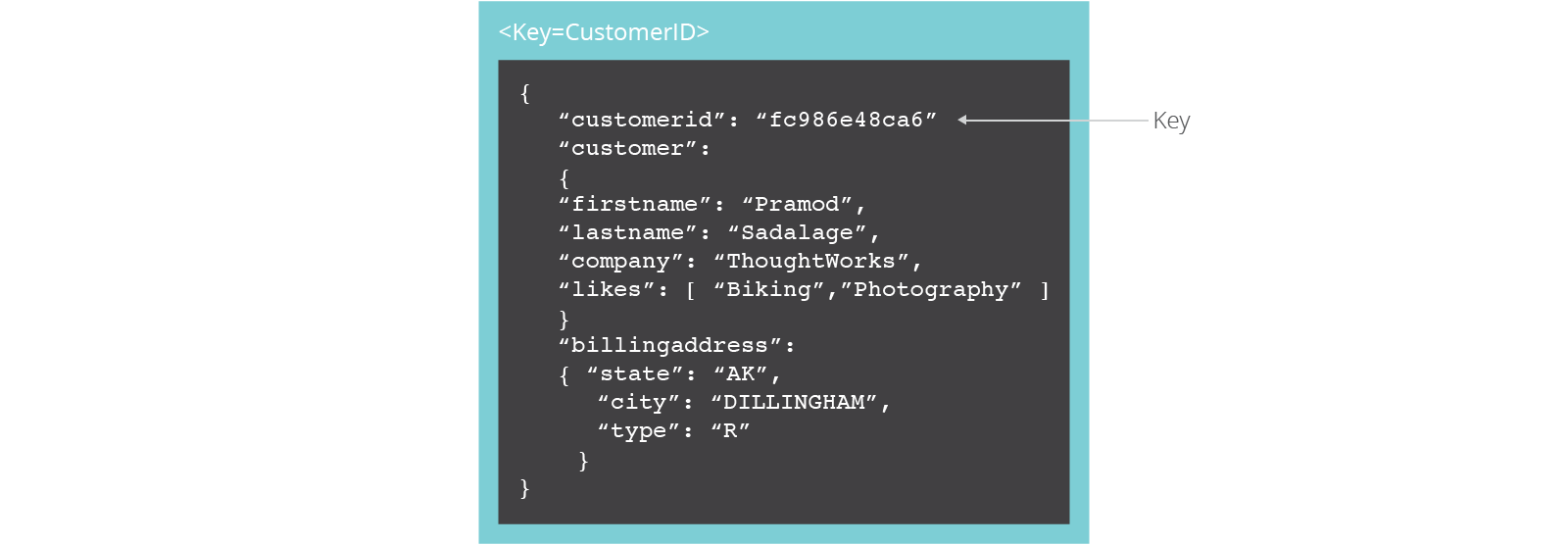
**Document Data Model**

The database stores and retrieve data in Document format which can be XML, JSON, BSON, and so on.

These documents are self-describing, hierarchical tree data structures which can consist of maps, collections, and scalar values.

The documents stored are similar to each other but do not have to be exactly the same.

Document databases store documents in the value part of the key-value store; think about document databases as key-value stores where the value is examinable.



Document databases are useful for CMS, blogging platforms, web analytics, real-time analytics, ecommerce-applications.

We would avoid using document databases for systems that need complex transactions spanning multiple operations or queries against varying aggregate structures.

Ex: **ArangoDB, MongoDB, AzureDocumentDB, CouchDB, RavneDB**

**Key Value Data Model**

It has a Big Hash Table of keys & values

The client can get the value for key, put a value for key , delete a key from the data store.

Used to store Large Objects ie BLOB irrespective of whats in side the object . Canbe vedio , audio, Images ,Text and etc

Since key-value stores use primary-key access, gives great performance and scaled.

Key-value databases are generally useful for storing session information, user profiles, preferences, shopping cart data.

We would avoid using Key-value databases when we need to query by data

Ex: **DynamoDB, BangDB, Scalaris, GenieDB**

**Graph Databases**

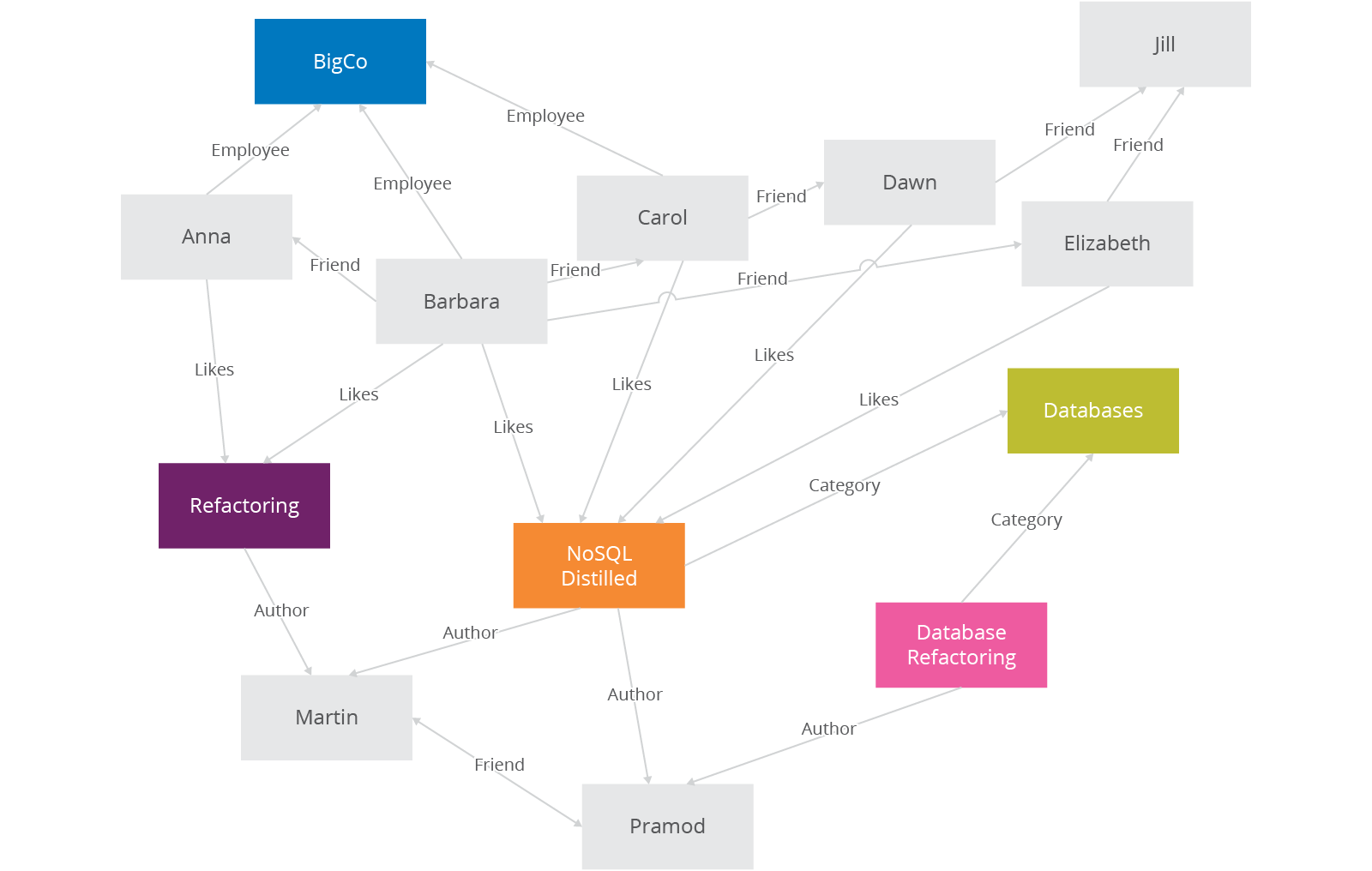
Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties.

Think of a node as an instance of an object in the application.

Relations are known as edges that can have properties.

Edges have directional significance;

The organization of the graph lets the data to be stored once and then interpreted in different ways based on relationships.



Graph databases are very well suited to problem spaces where we have connected data, such as social networks, spatial data, routing information for goods and money, recommendation engines

Ex: **ArangoDB, Neo4J, GraphBase**

Note : One NoSql DBMS may support more than one Data model, example Arangodb is graphBased , DocumnetBased, keyValueBased