What is a NoSQL Database?

In general terms, Databases, which store data in a format different from relational databases, are known as NoSQL databases. NoSQL stands for "not only SQL," which pertains to the fact that either the database can store and manage data using "no SQL" at all or it can work in a combination that combines the flexibility of the newer approach (NoSQL) with the power of the traditional relational system (SQL).
Unlike relational databases, related data doesn't have to be split up between multiple tables; instead, it can be nested within the same data structure.
Let's understand the workings by taking an example of a bank.

SQL :

In relational databases, data is normalized (to avoid redundancy) and stored in multiple tables with relationships defined by constraints.
For example, the Bank Database contains two tables whose attributes are listed as follows

Customer	Loan
Account Number (PK)	Account Number
Name	Loan_ID (primary key)
Phone Number	Loan Amount
	Loan Status

NoSQL:

Since NoSql is classified into four types, let's discuss this example by taking a Document Database.
The customer's entire record, including his loan details, can be stored within a JSON document in the document database. All the attributes, including Account Number, Name, Phone Number, Loan_Id, Loan Amount, and Loan status, can be stored inside a single document for a particular customer.


```
{
  "Acc_No":1234,
  "Name":"Rahul",
  "Phone_No":34567832,
  "Loan_Id":12,
  "Loan_Amt":320000,
  "Loan_Status":"Pending"
}
```

NoSQL Database Management Systems are mainly used to deal with Big data that are not necessarily structured or related.

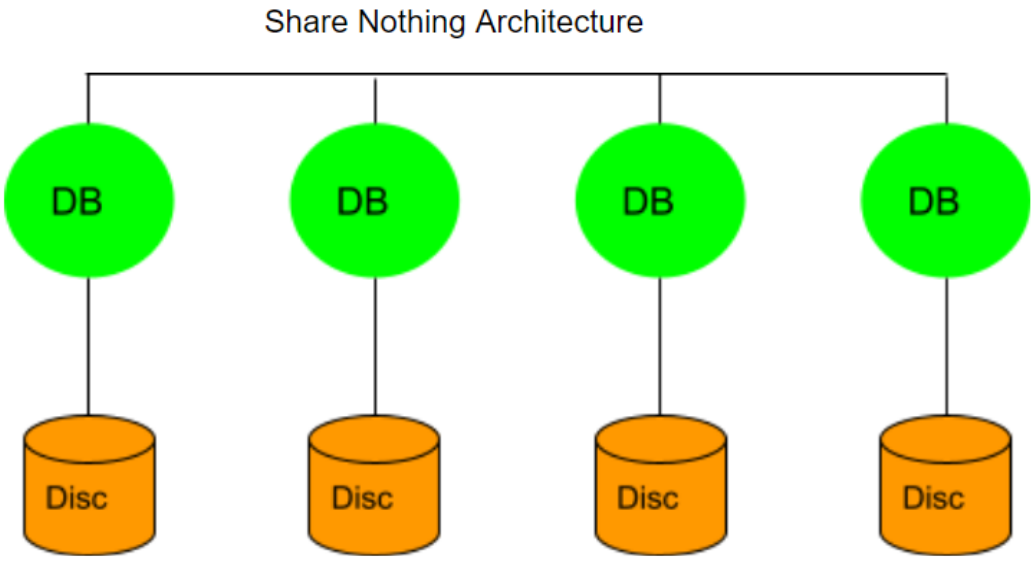
Features of NoSQL

1. Multi-Model:

Unlike relational databases, where data is stored in relations, different data models in NoSQL databases make them flexible to manage data. Each data model is designed for specific requirements.
Examples of data models include document, graph, wide-column, and key-value.
The concept is to allow multiple data models in a single database. By doing so, the need for deploying and managing different databases for the same data cancels out.

2. Distributed

NoSQL databases use the shared-nothing architecture, implying that the database has no single control unit or storage. The advantage of using a distributed database is that data is continuously available because data remains distributed between multiple copies. On the contrary, Relational Databases use a centralized application that depends on the location.



3. Flexible Schema

Unlike relational databases where data is organized in a fixed schema, NoSQL databases are quite flexible while managing data. While relational databases were built typically to manage

structured data, NoSQL databases can process structured, semi-structured or unstructured data with the same ease, thereby increasing performance.

4. Eliminated Downtime

One of the essential features is the eliminated downtime. Since the data is maintained at various nodes owing to its architecture, the failure of one node will not affect the entire system.

5. High Scalability

One of the reasons for preferring NoSQL databases over relational databases is their high scalability. Since the data is clustered onto a single node in a relational database, scaling up poses a considerable problem. On the other hand, NoSQL databases use horizontal scaling, and thus the data remains accessible even when one or more nodes go down.

What do NoSQL databases have in common?

1. They use distributed databases, which are based on shared-nothing architecture.
2. NoSQL databases can easily be scaled out horizontally, depending on the volumes of data.
3. All of the NoSQL databases have a flexible schema.
4. Furthermore, they process: structured, semi-structured as well as non-structured data.
5. Lastly, the format of storing data is different from relational databases hence are non-relational.

What are the Different Types of NoSQL Data Stores ??

In my last post, I gave an overview of NoSQL databases and typical use cases where one can use NoSQL data stores.

As mentioned in the last post, NoSQL databases can be classified into four types,

1. Key-Value (KV) Stores
2. Document Stores
3. Column Family Data stores or Wide column data stores
4. Graph Databases

Here is an explanation for each of these types.

1. Key-Value (KV) Stores

This is the simplest type of NoSQL database.Under this type, the data is stored in the form of key/value pairs. For each Key, there is a value assigned to it. Each Key is unique and accepts only strings, whereas the value corresponding to the particular Key can accept String, JSON, XML, etc. Owing to this behavior, it is capable of dealing with massive loads of data.
Key Value Stores maintain data as pair consisting of an index key and a value. KV stores query Values using the index Key. Every item in the database is stored in the pairs of Keys (Indexes) and Values. KV stores resemble a relational database but with each table having only two columns.

Some KV stores may even allow basic joins to help you scan through if there are composite joins, they may not be a suitable options.

There are multiple KV Stores available, each differing mainly in their adaption of the CAP theorem and their configurations of memory v/s storage usage.

KV stores have fast query performance and are best suited for applications that require content caching, e.g. a gaming website that constantly updates the top 10 scores & players.

Key	Value
Name	Raman Sharma
Account Number	21657243
Amount	567543

Key-Value Pairs Database: Features:

1. Consistency
2. Transactions
3. Query Features
4. Data Structure
5. Scaling

Pros:

– Simple Data model

– Scalable

– Value can include JSON, XML, flexible schemas

– Extremely Fast Owing to it's simplicity

– Best fit for cases where data is not highly related

Cons:

– No relationships, create your own foreign keys

– Not suitable for complex data

– Lacks Scanning Capabilities

– Not ideal for operations rather than CRUD (create, read, update Delete)

Key-Value Pairs Database: Use Case:

These kinds of databases are best suited for the following cases:

1. Storing session information: offers to save and restore sessions.
2. User preferences: Specific Data for a particular user
3. Shopping carts: easily handle the loss of storage nodes and quickly scale Big data during a holiday/sale on an e-commerce application.
4. Product recommendations: offering recommendations based on the person's data.

Popular KV Stores would include Dynamo DB, Redis, BerkleyDB.

2. Document Stores

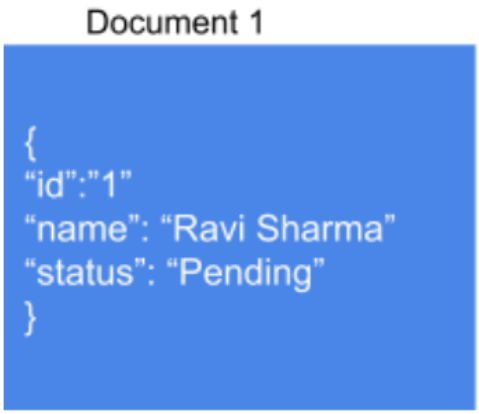
Document Stores are an extension of the simplicity of Key Value stores, where the values are stored in structured documents like XML or JSON. Document stores make it easy to map Objects in the object- oriented software.

A document database is schema free, you don't have to define a schema beforehand and adhere to it. It allows us to store complex data in document formats (JSON, XML etc.).

Document databases do not support relations. Each document in the document store is independent and there is no relational integrity.

Document stores can be used for all use cases of a KV store database, but it also has additional advantages like there is no limitation of querying just by the key but even querying attributes within a document, also data in each document can be in a different format. E.g. A product review website where zero or many users can review each product and each review can be commented on by other users and can be liked or disliked by zero to many users.

E.g, A product review website where zero or many users can review each product, and each review can be commented on by other users and can be liked or disliked by zero to many users.



Document Stores: Features:

Features of document databases

1. Faster Querying
2. A large amount of data can be easily handled owing to its structure
3. Flexible Indexing

Pros:

– Simple & Powerful Data model

– Scalable

– Open Formats

– No foreign Keys

Cons:

– Not suitable for relational data

– Querying limited to keys & indexes

– Map Reduce for more significant queries

Document Stores: Use Case:

1. User Profiles: Since they have a flexible Schema, the document can store different attributes and values. This enables the users to store different types of information.
2. Management of Content: Since it has a flexible schema, collection and storing any data has been made possible. This allows the creation of new types of content, including images, videos, comments, etc. Everyday use is seen in blogging platforms.
   Some popular Document stores are MongoDB, CouchDB, Lotus Notes.

3. Column Family Data stores or Wide column data stores

Wide column data stores take a hybrid approach mixing the declarative characteristics game of relational databases with the key-value pair based and totally variables schema of key-value stores. Wide Column databases stores data tables as sections of columns of data rather than as rows of data.

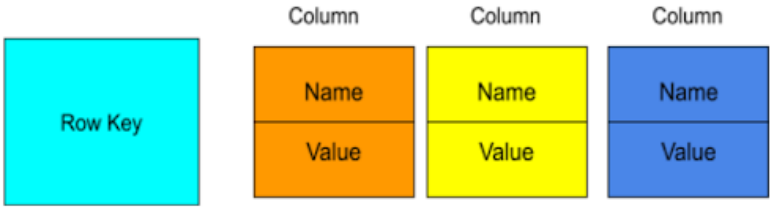
Columnar Family databases have their origins in Google's Bigtable. According to Google's paper on Bigtable, "A Bigtable is a sparse, distributed, persistent multidimensional sorted map." This definition might leave you confused, just as I was, it was all greek to my RDBMS oriented mind.

Here is a more simplified explanation, a column family data store is a multi-dimensional key value store (map or associative array) which is persistent (values persist after creation or access), distributed (data is distributed across multiple computing & storage nodes), sorted (sorted keys) and sparse (values for certain dimensions may not be populated, similar to sparsely populated rows in RDBMS).
The multi-dimensional aspect of column stores brings in another concept of column families.

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.

There are two types of column families:

1. Standard Column family: Standard Column family consists of a key-value pair, where the key is mapped to a value that is a set of columns. In analogy with relational databases, a standard column family is as a "table", each key-value pair being a "row".
2. Super Column family: Super Column family consists of a key-value pair, where the key is mapped to a value that are column families. In analogy with relational databases, a super column family is something like a "view" on a number of tables. It can also be seen as a map of tables.



Column Oriented Database: Features:

Features of Wide column stores:

1. Multidimensional key store
2. Persistent in nature
3. Distributed
4. High flexibility

Pros:

– Supports semi-structured data

– Naturally indexed

– Scalable

Cons:

– Not suitable for relational data

Column Based Databases: Use Case:

1. User Preferences
2. Business Intelligence
3. Managing data warehouses
4. Reporting Systems

Some of the popular Wide column data stores include Google's Bigtable, Cassandra, HBase.

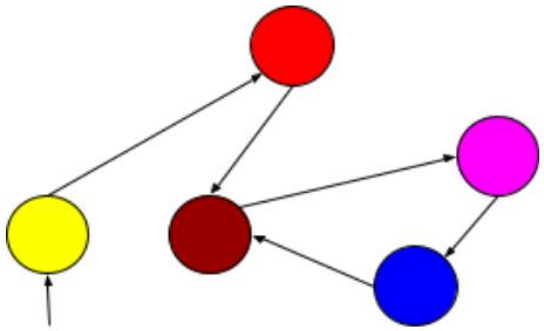
Note, wide column data stores are not to be confused with column-oriented databases, I'll may be cover this in a separate post.

4. Graph Databases

Graph Databases specific purpose is the storage of graph-oriented data structures. A graph database is any storage system that provides index-free adjacency. This means that every

node contains a direct pointer to its adjacent element and no index lookups are necessary. As the number of nodes increases, the cost of a hop remains the same.
Graph databases are optimized for traversing through connected data, e.g. traversing through a list of contacts on your social network to find out the degree of connections.

Graph databases usually come with a flexible data model, which means there is no need to define the types of edges and vertices.



Graph Databases: Features:

Features of graph databases

1. Flexibility
2. Agility
3. Improved performance, even with huge volumes of data.
   Typical use cases for graph databases would include social networking site, recommendation engine.

Pros:

– Extremely powerful

– Connected data is locally indexed

– Can provide ACID

– Results in real-time

– Agile Structure

Cons:

– Difficult to scale out, though can scale up

Graph Databases: Use Case:

Typical use cases for graph databases would include

1. Social networking site
2. Recommendation engine
3. Logistics
4. Risk assessment
5. Fraud detection

Some of the popular graph databases are Neo4j, OrientDB, Allegrograph.

Conclusion:

In crux, we can say that there are four types of NoSQL Databases: Key-Value (KV) Stores, Document Stores, Column Family Data stores, and Graph Databases.
NoSQL data stores provide an alternative to the traditional RDBMS, and you might be not be sure of the NoSQL databases you want to select. The ideal way of identifying the best suitable NoSQL database for your application is to figure out the requirement that is not met by RDBMS. If all you requirements are fulfilled by a RDBMS, you may not want a NoSQL data store.

If you have a requirement for managing large, unstructured data sources feel free to contact us. I will also appreciate your feedback to this post.