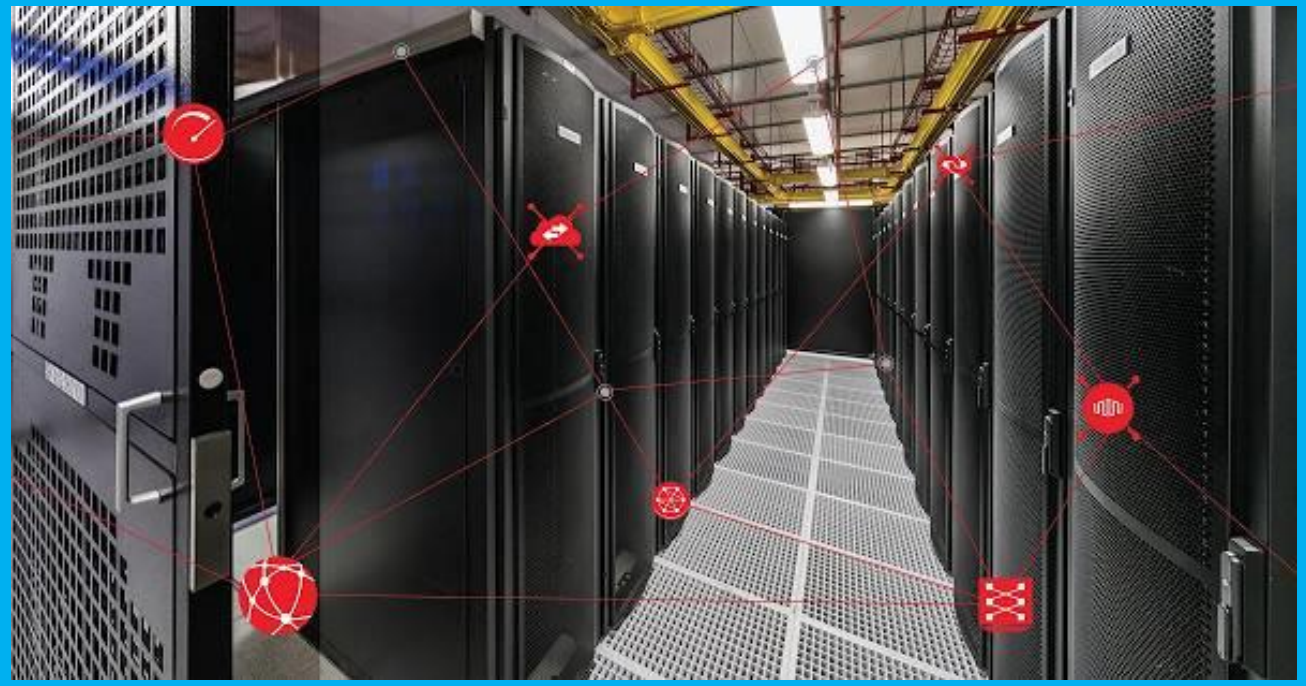
IT1768 - Database Systems



Topic 9 :

NoSQL Database

NoSQL Database – A next big trend in databases

Reference source: https://www.tutorialspoint.com/mongodb/index.htm

Unit 10 : NoSQL Database

Objective :

- At the end of this unit, you should be able to:
  - Summarize different types of NoSQL databases.
  - classify the characteristics of NoSQL databases.
  - List examples of application of NOSQL databases.

What is NoSQL Database ?

- **NoSQL** database referred to "non SQL“ database or "non relational" database.
- NoSQL is a mechanism for storage and retrieval of data that is not modelled in relational databases.
- NoSQL databases have existed since the late 1960s. It gained popularity triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com.
- NoSQL databases are increasingly used in big data and real-time web applications.
- Many NoSQL stores compromise consistency in favour of availability, partition tolerance, and speed.

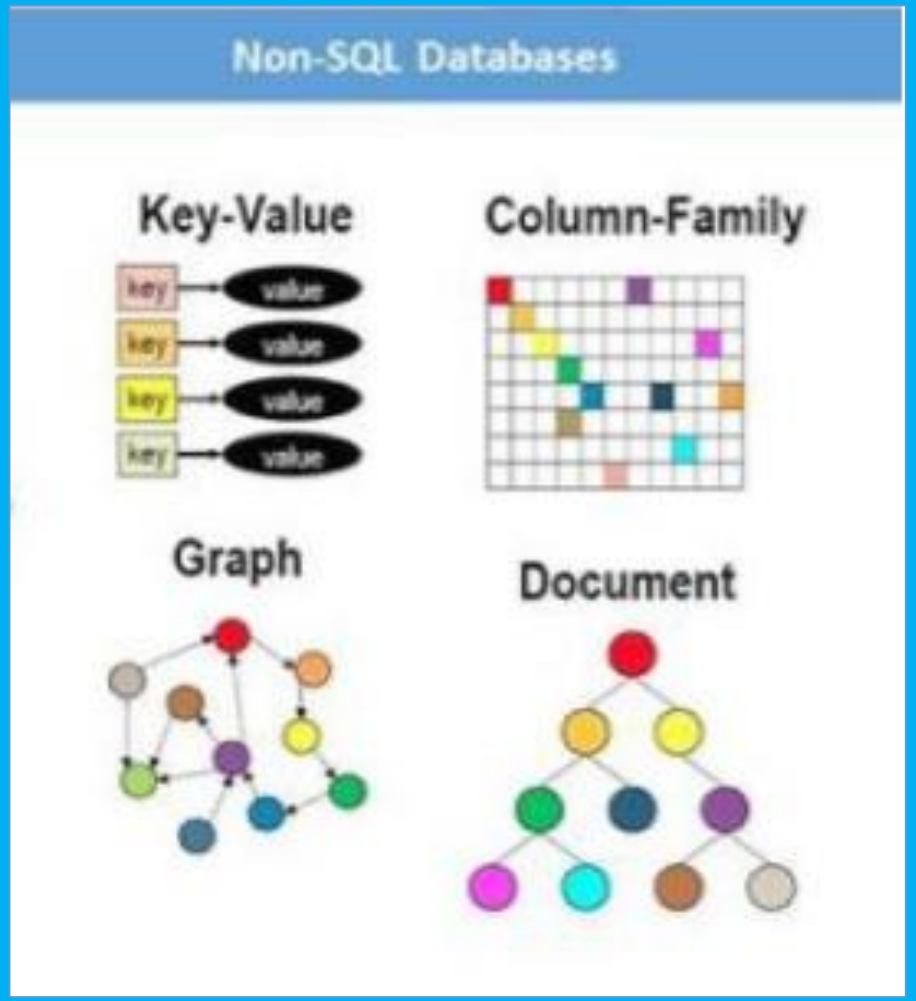
View video on introduction of NoSQL database:
(focus on segment from 7:50 min to 20 min)
Video on NoSQL – Reference source: https://youtu.be/qI_g07C_Q5I

4 Types of NoSQL databases

4 types of NoSQL databases:

- Key-Value databases
- Document store databases
- Column-Family databases
- Graph databases.

NoSQL databases and database engines:	
▪ Key-value:	Apache, Ignite, ArangoDB, Couchbase, DynamoDB
	Oracle NoSQL, OrientDB, Oracle Berkeley DB, Redis
▪ Document :	Apache, CouchDB, ArangoDB, BaseX, Couchbase,
	MongoDB, OrientDB, RethinkDB
▪ Column :	Apache Accumulo, Cassandra, Hbase, Big table
▪ Graph :	AllegroGraph, Neo4j, ArangoDB, Apache Giraph,
	OrientDB, MarkLogic



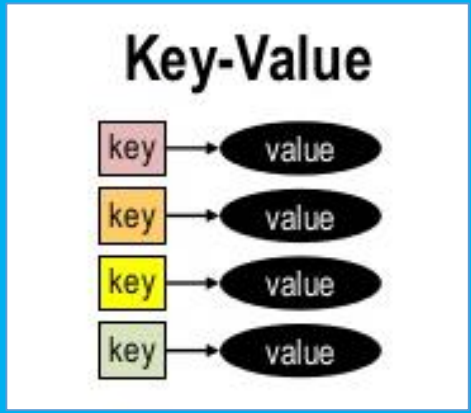
Reference source : https://www.thoughtworks.com/insights/blog/nosql-no-problem-intro-nosql-databases

Reference source: https://www.tutorialspoint.com/mongodb/index.htm

Types of NoSQL Databases

1) Key Value Databases

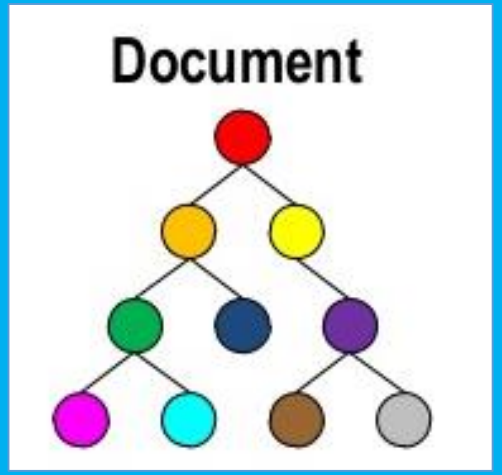
Key-value stores are the simplest NoSQL databases. Every single key-value pair stores a unique key which can be used to look up for a value (e.g. text, integer, currency etc.), each value could have different meta data. Examples of key-value stores are Riak and Berkeley DB.



2) Document Oriented Databases

Document databases pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents

- Document databases allow indexing of documents on the basis of not only its primary identifier but also its properties. The most prominent among the available options are MongoDB and CouchDB.

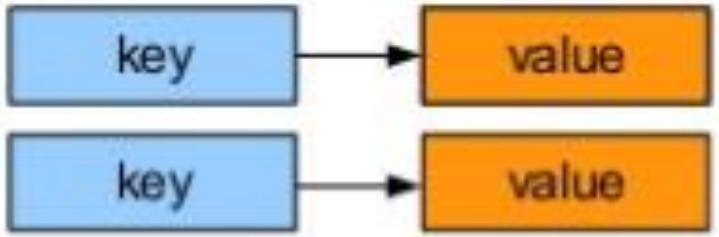


https://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases

NoSQL Database : (1) Key Value Store (KV)

Key / value stores (opaque)

- Keys are mapped to values
- Values are treated as BLOBs (opaque data)
- No type information is stored
- Values can be heterogenous (string, integer, or the like)



Key	Value
"India"	{"B-25, Sector-58, Noida, India – 201301"
"Romania"	{"IMPS Moara Business Center, Buftea No. 1, Cluj-Napoca, 400606",City Business Center, Coriolan Brediceanu No. 10, Building B, Timisoara, 300011"}
"US"	{"3975 Fair Ridge Drive. Suite 200 South, Fairfax, VA 22033"}

The key can be synthetic or auto-generated while the value can be String, JSON, BLOB (basic large object) etc.

This key/value type database allow clients to read and write values using a key as follows:

- Get(key), returns the value associated with the provided key.
- Put(key, value), associates the value with the key.
- Multi-get(key1, key2, .., keyN), returns the list of values associated with the list of keys.
- Delete(key), removes the entry for the key from the data store.

Reference source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

NoSQL Database : (1) Key Value Store (K-V) ….. No schema involved

- key-value databases store all key-value pairs together in a single namespace. The key in a key-value pair must be unique.
- Key-value databases are the simplest of the NoSQL databases. It can store a value, such as an integer, string, a JSON structure, or an array, along with a key used to reference that value.
- The value in a key-value store can be anything, such as text (long or short), a number, HTML, XML or JSON, programming code such as PHP, an image, short video etc. It could also be a list, or even a nested key-value pair.
- If you usually retrieve data by key or ID value and don't need to support complex queries, a key-value database is a good option.
- If query patterns and data structures are fairly simple, key-value databases are a good choice. As the complexity of queries and entities increase, document databases become a better option.

Keys	Values
ID	SLP1234A
Brand	Audi
Model	A3
Capacity	1600
Color	Black
Errormsg	Brake light warning

Keys	Values
ID	Group1
Brand	Audi
Size	25, [1,2,3]

(value can be in a set)

Poll a node to match a value; e.g.

SLP1234A_Brand >> ‘Audi’

SLP1234A_Color >> Black.

SLP1234A_errormsg>>’Brake light warning’

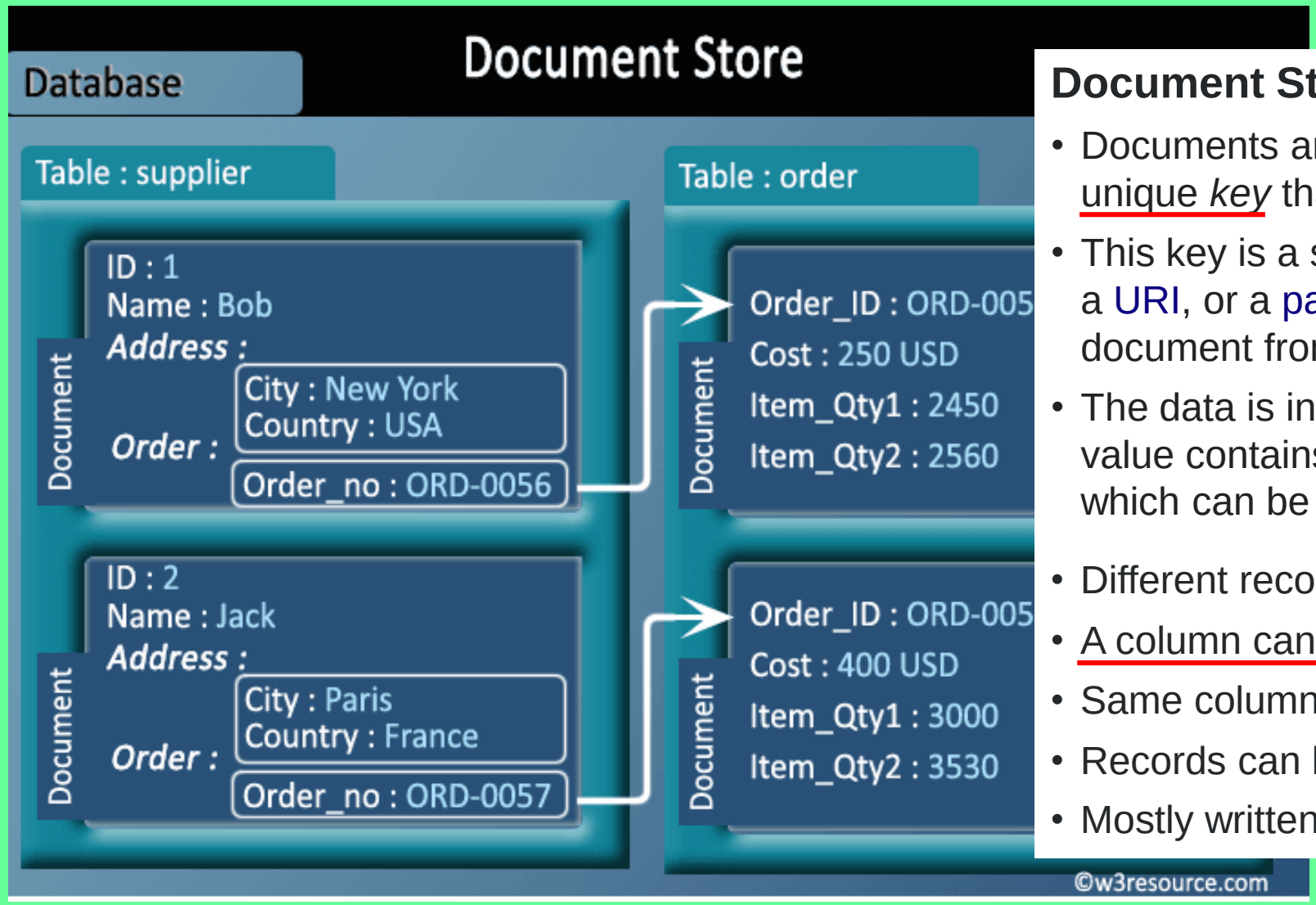
Reference source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

NoSQL Database : (1) Key – Value Store (KV)

Why adopt Key-Value DB

- Key-value architecture can be more performant than relational databases in many scenarios because there is no need to perform lock, join, union, or other operations when working with objects.
- Unlike traditional relational databases, a key-value store doesn't need to search through columns or tables to find an object. Knowing the key will enable very fast location of an object.

NoSQL Database : (2) Document Store (Document-oriented database)



Document Store:

- Documents are addressed in the database via a unique *key* that represents that document.
- This key is a simple identifier (or ID), typically a string, a URI, or a path. The key can be used to retrieve the document from the database.
- The data is in a collection of key value pairs. The value contains structured or semi-structured data which can be in XML, JSON or BSON format.
- Different records can have different columns.
- A column can have more than one values (array).
- Same column can have different types of values.
- Records can have a nested structure.
- Mostly written in JSON language

NoSQL Database : Document Structure of a blog site

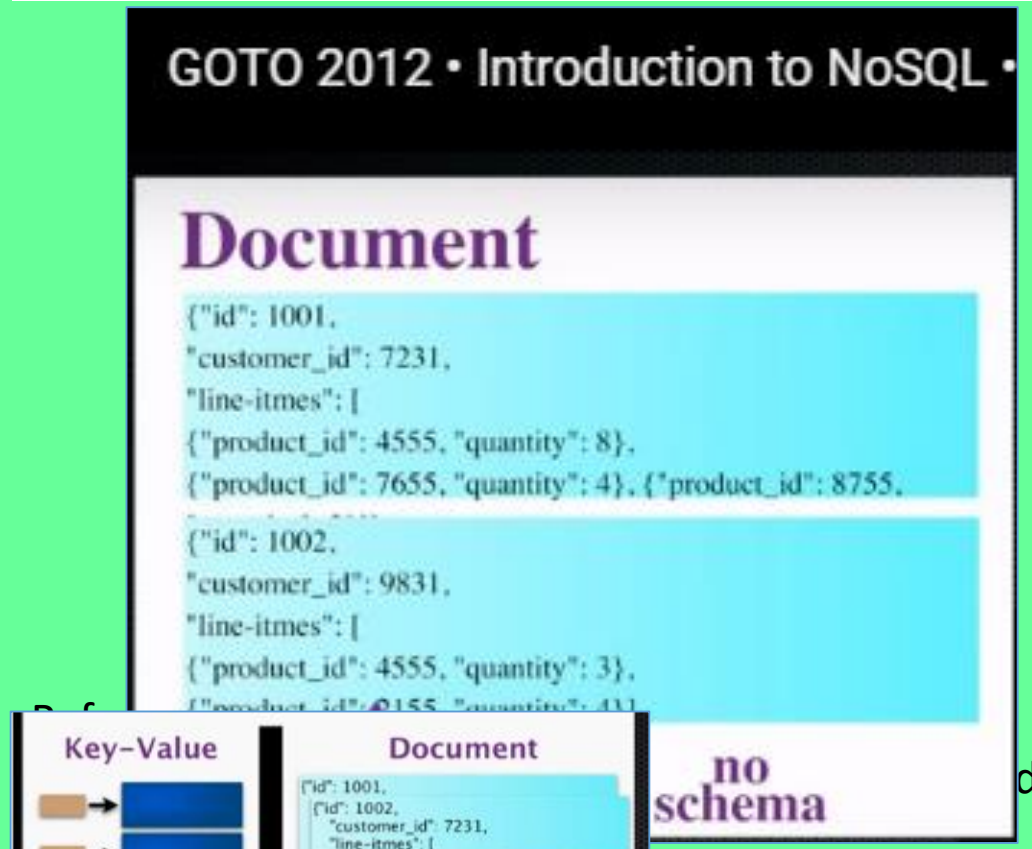
Examples of Document Store DB:
- Mongo DB

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100,
   comments: [
      {
         user:'user1',
         message: 'My first comment',
         dateCreated: new Date(2011,1,20,2,15),
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

Key-value pairs

Considered as one document

- Identifier : _id is a 12 bytes hexadecimal number
- MangoDB can provide a unique id for every document.



db

NoSQL Database : (2) Document Store

Examples of Document Store DB:
- Mongo DB, CouchDB
- Amazon DynamoDB
- Microsoft Azure Cosmos DB.

Written in JSON

```
{
    '_id' : 1,
    'artistName' : { 'Iron Maiden' },
    'albums' : [
        {
            'albumname' : 'The Book of Souls',
            'datereleased' : 2015,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Killers',
            'datereleased' : 1981,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Powerslave',
            'datereleased' : 1984,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Somewhere in Time',
            'datereleased' : 1986,
            'genre' : 'Hard Rock'
        }
    ]
}
```

Written in XML

```
<artist>
  <artistname>Iron Maiden</<artistname>
  <albums>
    <album>
      <albumname>The Book of Souls</albumname>
      <datereleased>2015</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Killers</albu
      <datereleased>1981</date
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Powerslave</a
      <datereleased>1984</date
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Somewhere in
      <datereleased>1986</date
      <genre>Hard Rock</genre>
    </album>
  </albums>
</artist>
```

If by relational database:
- It needs 3 tables

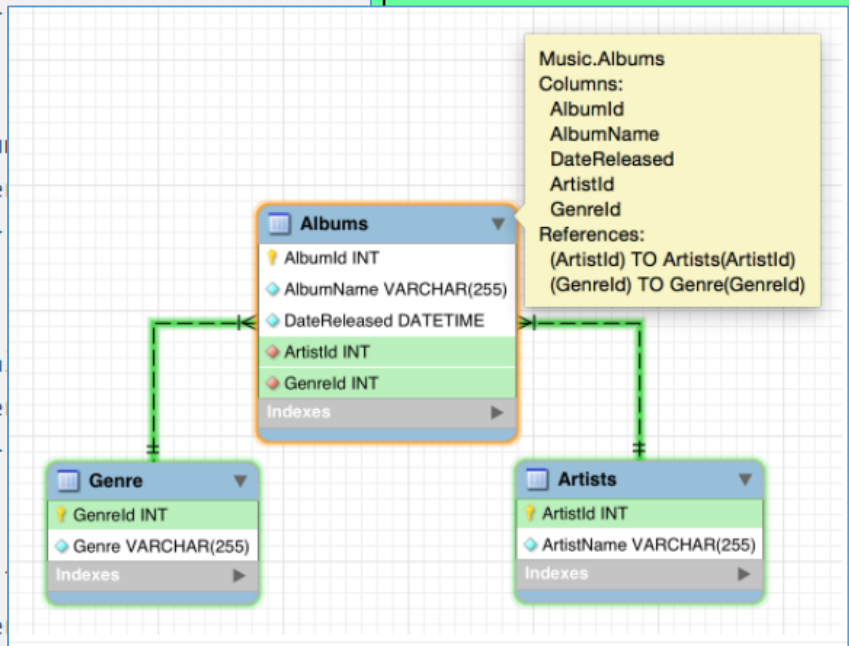


Diagram of a relationship between three tables. The primary key and foreign key fields have been highlighted.

Why use Document Store Database ?

BENEFITS OF A DOCUMENT DATABASE

Document stores offer important advantages when specific characteristics are required, including:

- **Flexible data modeling:** As web, mobile, social, and IoT-based applications change the nature of application data models, document databases eliminate the need to force-fit relational data models to support new types of application data models.
- **Fast write performance:** Unlike traditional relational databases, some document databases prioritize write availability over strict data consistency. This ensures that writes will always be fast even if a failure in one portion of the hardware or network results in a small delay in data replication and consistency across the environment.
- **Fast query performance:** Many document databases have powerful query engines and indexing features that provide fast and efficient querying capabilities.

Reference source: http://basho.com/resources/document-databases/

Examples of DB:

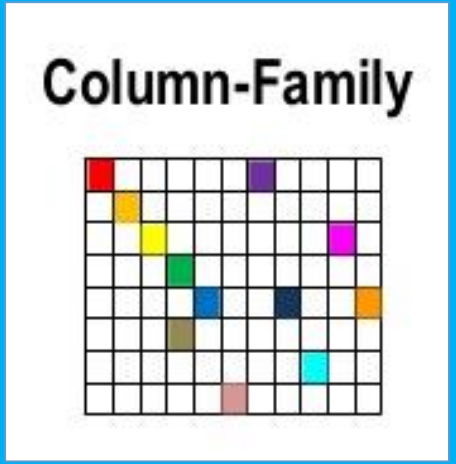
- Mongo DB, CouchDB
- Amazon DynamoDB
- Microsoft Azure Cosmos DB.

Types of NoSQL Databases

3) Column Databases

The data is stored in rows and multiple columns with data being organized by columns. Each unit of data can be thought of as a set of key/value pairs. Retrieval of data bases on the single row-key together with the common column name. Column-oriented storage allows data to be stored effectively.

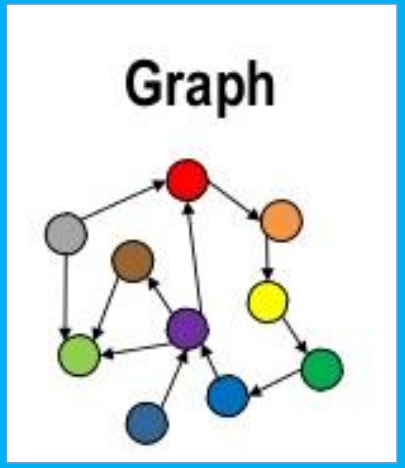
Wide-column stores such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows



4) Graph Based Databases

A graph database uses graph structures with nodes, edges, and properties to represent and store data. A graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjacent element and no index lookups are necessary.

Graph stores are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.



https://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases

NoSQL Database : (3) Column-Family (Column-Oriented DB, Columnar DB)

Relational Database

City	Pincode	Strength	Project
Noida	201301	250	20
Cluj	400606	200	15
Timisoara	300011	150	10
Fairfax	VA 22033	100	5

Column Store Database

Row ID	Columns...		
1	Name	Website	
	Preston	www.example.com	
2	Name	Website	
	Julia	www.example.com	
3	Name	Email	Website
	Alice	example@example.com	www.example.com

Each row in the column family has a unique identifier key, and then as many columns as it needs to hold the information. There is no requirement for every row to have the same columns

Column Store NoSQL Database:

- In column store database, it stores data tables by column rather than by row. Each row has a unique key.
- Data is stored in cells grouped in columns of data rather than as rows of data.
- Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns.
- Each row can contain a different number of columns to the other rows. And the columns don't have to match the columns in the other rows.
- It can use traditional database query languages like SQL to load data and perform queries.
- The benefit of storing data in columns is fast search/ access and data aggregation. Columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster.

Reference source:
https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases/

Column-oriented databases : Reference source : https://youtu.be/mRvkikVuojU

How does Column-Family appear like:

Relational Database

City	Pincode	Strength	Project
Noida	201301	250	20
Cluj	400606	200	15
Timisoara	300011	150	10
Fairfax	VA 22033	100	5

Column Store Database	
Row (the key)	3PillarNoida, 3PillarCluj, 3PillarTimisoara, 3PillarFairfax
Column families	Address and details
Column family has columns ‘city’ & ‘pincode’	
Column family has columns ‘strength’ & ‘projects’	

Column Store Database Example

```
{  3PillarNoida :  {
                        city : Noida
                        pincode : 201301 } ,
                        details : {
                        strength : 250
                        project : 20 }                    }
{  3PillarCluj : {
                    address : {
                            city : Cluj
                            pincode : 400606 } ,
                    details : {
                            strength : 200
                            projects : 15 } }             }
{  3PillarTimisoara : {
                    address : {
                            city : Timisoara
                            pincode : 300011 },
                    details : {
                            strength : 150
                            projects : 10 }               }
```

Reference source: https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases/

Performance of Column-Family NoSQL Database

1) Faster speed of reading the same number of column field values

- Row-based system is not efficient at performing set-wide operations on the whole table, the DBMS would have to fully scan through the entire table looking for matching records. Using columnar storage, each data block holds column field values for as many as three times as many records as row-based storage. This means that reading the same number of column field values for the same number of records requires a third of the I/O operations compared to row-wise storage.
- The database can more precisely access the data it needs to answer a query rather than scanning and discarding unwanted data in rows. Query performance is often increased.
- It reduces the amount of data you need to load from disk. Even with index, if it is heavily used, it can dramatically reduce the time for common operations.
- A column-oriented database serializes all of the values of a column together, it operates on one or a small number of columns at a time, thus speeding up the retrieval process.

2) Fast storing and retrieval of data in memories. (optimize analytic query performance)

Since many database operations only need to access or operate on one or a small number of columns at a time, you can save memory space by only retrieving blocks for columns you actually need for a query.It performs particularly well with aggregation queries.

Why use Column-Family Database ?

Benefits of Column Store Databases

Some key benefits of columnar databases include:

- **Compression**. Column stores are very efficient at data compression and/or partitioning.
- **Aggregation queries**. Due to their structure, columnar databases perform particularly well with aggregation queries (such as SUM, COUNT, AVG, etc).
- **Scalability**. Columnar databases are very scalable. They are well suited to massively parallel processing (MPP), which involves having data spread across a large cluster of machines – often thousands of machines.
- **Fast to load and query**. Columnar stores can be loaded extremely fast. A billion row table could be loaded within a few seconds. You can start querying and analysing almost immediately.

Examples of Column Store Databases

- Bigtable
- Cassandra
- HBase
- Vertica
- Druid
- Accumulo
- Hypertable

http://database.guide/what-is-a-column-store-database/

Summary on Column-Family NoSQL Database

Column-oriented databases

- Column-oriented databases primarily work on columns and every column is treated individually.
- Values of a single column are stored contiguously. (adjacently).:
- Column stores data in column specific files.
- In Column stores, query processors work on columns too.
- All data within each column datafile have the same type which makes it ideal for compression.
- Column stores can improve the performance of queries as it can access specific column data.
- High performance on aggregation queries (e.g. COUNT, SUM, AVG, MIN, MAX).
- Works on data warehouses and business intelligence, customer relationship management (CRM), Library card catalogs etc.

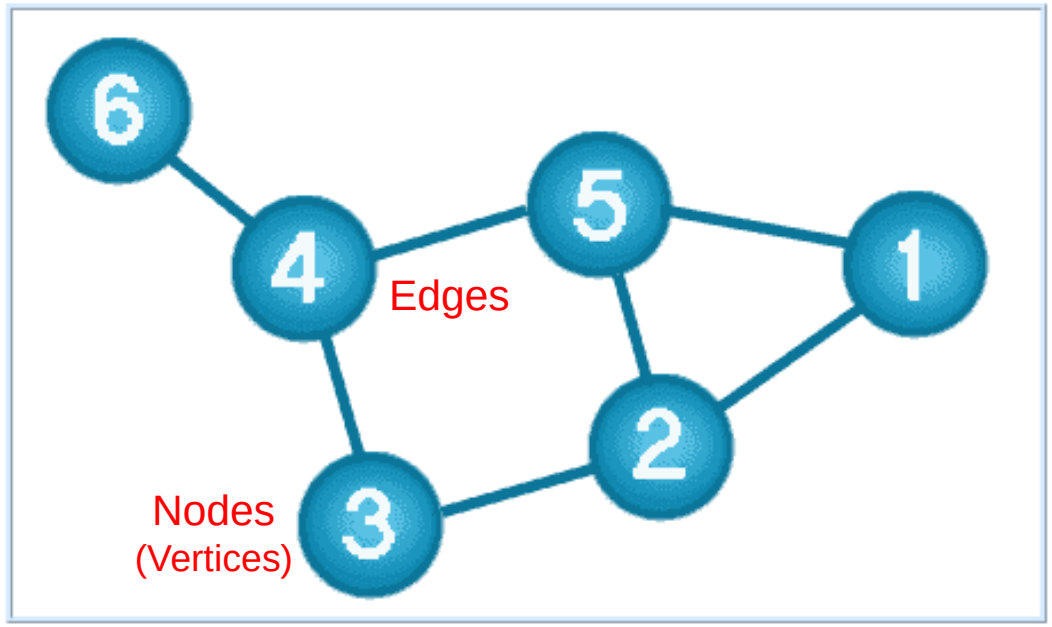
Example of Column-oriented databases : BigTable, Cassandra, SimpleDB etc.

NoSQL Database : (4) Graph Database

Graph databases

A graph data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices.

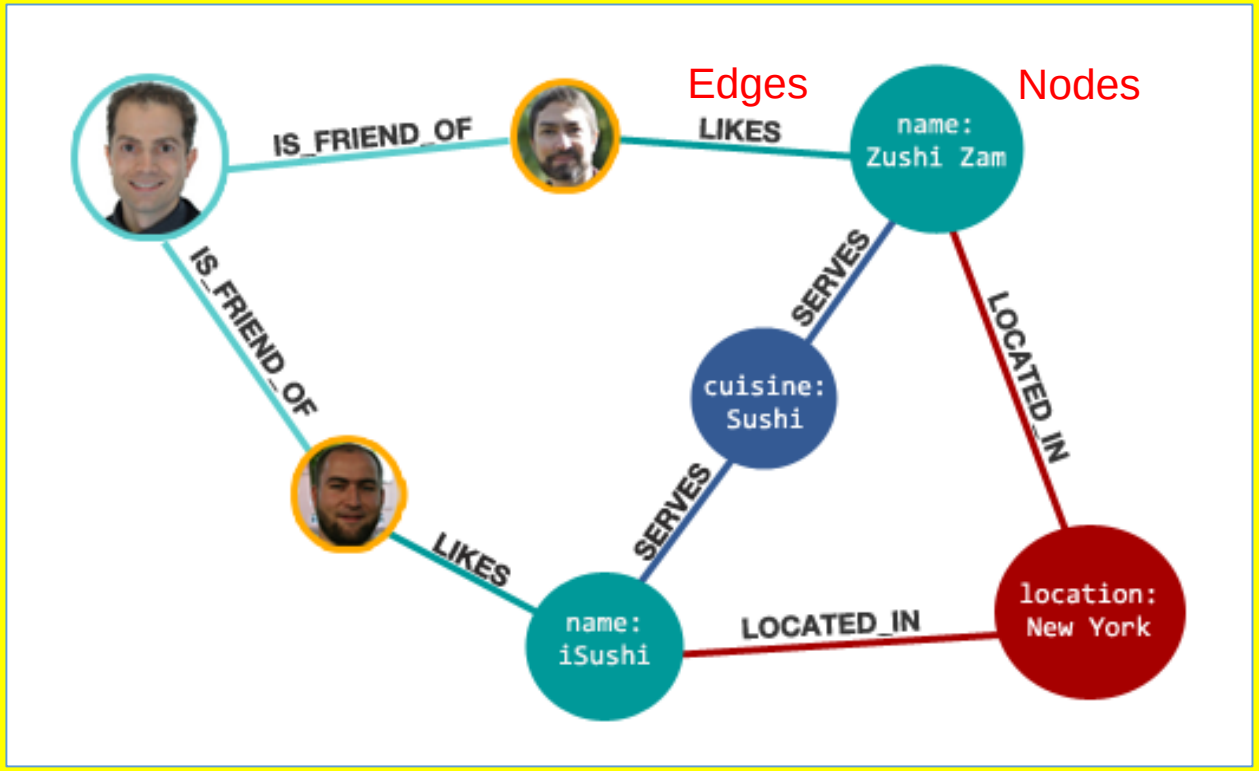
The following picture presents a labeled graph of 6 vertices and 7 edges.

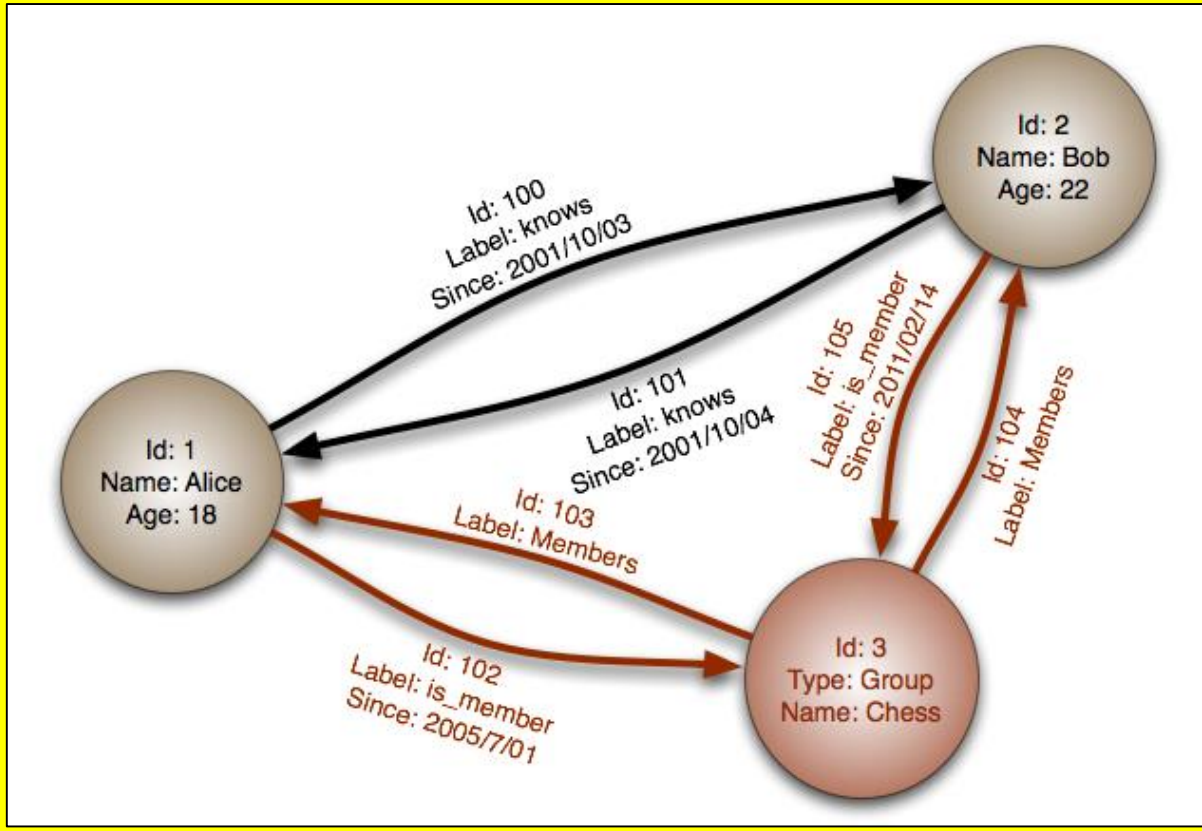


Graph databases are basically built upon the Node, edge and property to represent and store data. Nodes contain properties. Edges represent relationships exist between nodes. An edge allows data in the store to be linked together and, a property is simply the node on the opposite end of the relationship.

Reference source:
https://neo4j.com/blog/why-the-most-important-part-of-facebook-graph-search-is-graph/

NoSQL Database : (4) Graph Database

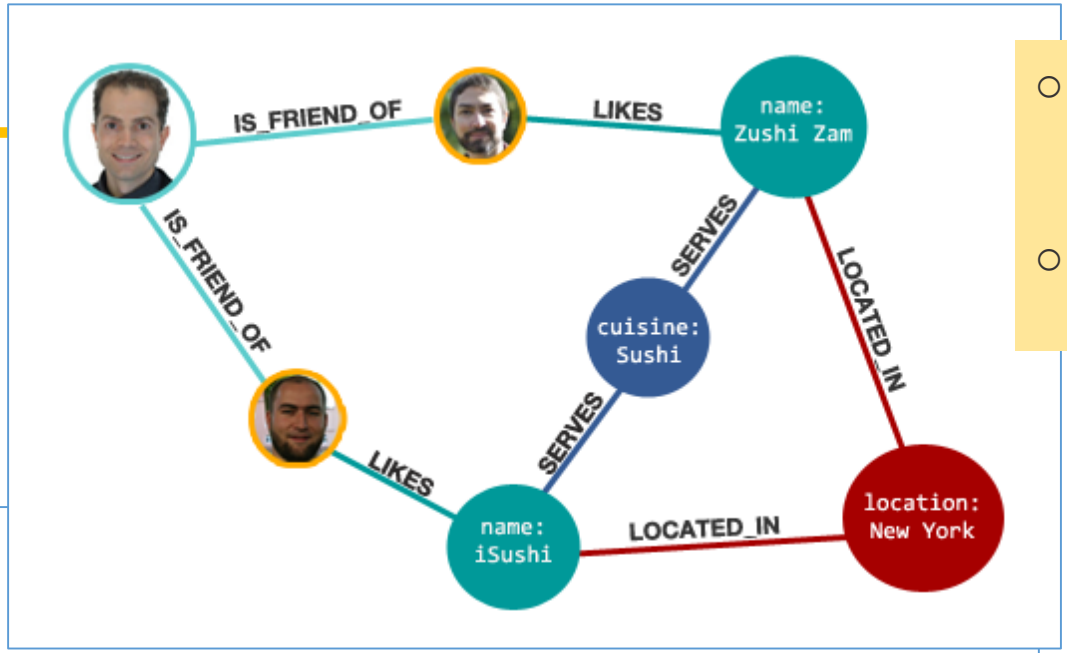




Reference source: https://neo4j.com/blog/why-the-most-important-part-of-facebook-graph-search-is-graph/

Sushi restaurants in New York that my friends like

NoSQL Database :



- Graph databases model relationships between nodes.
- Neo4j database is an example

neo4j

```
START me=node:person(name = 'Philip'),
      location=node:location(location='New York'),
      cuisine=node:cuisine(cuisine='Sushi')

MATCH (me)-[:IS_FRIEND_OF]->(friend)-[:LIKES]->(restaurant)
      -[:LOCATED_IN]->(location),(restaurant)-[:SERVES]->(cuisine)

RETURN restaurant
```

Cypher Query Language Example: *Sushi restaurants in New York that my friends like*

The benefits of NoSQL Database

- When compared to relational databases, NoSQL databases are
- More flexible, scalable and of higher productivity : NoSQL databases have good tendency to grow dynamically with changing requirements. It can handle structured, semi-structured and unstructured data. This issue cannot be addressed by the relational model.
- Improved data comprehension : Large volumes of rapidly changing structured, semi-structured, and unstructured data.
- Schema-less : Not defined by strict data structure, Agile sprints, quick schema iteration, and frequent code pushes.
- Object-oriented : Object-oriented programming that is easy to use and flexible.
- Geographically distributed : It scales-out architecture instead of expensive, monolithic architecture.
- Better performance : NoSQL databases deliver better and faster performance.

NoSQL will be the next big trend in databases, why ?

Over decades and decades of software development, we have been using databases in form of SQL (Structured Query Language) where we store our data in relational tables. However, in recent years with the tremendous rise in use of internet and Web 2.0 applications, the databases have grown into thousands and thousands of terabytes. Applications such as Facebook, Google, Amazon, Whatsapp, etc. gave rise to an entire new era of database management which follows approach of simple design, speed and faster scaling than the traditional databases. Such databases are used in big data, massive real time applications and analytics.

As an example, consider that you have a blogging application that stores user blogs. Now suppose that you have to incorporate some new features in your application such as users liking these blog posts or commenting on them or liking these comments. With a typical RDBMS implementation, this will need a complete overhaul to your existing database design. However, if you use NoSQL in such scenarios, you can easily modify your data structure to match these agile requirements. With NoSQL you can directly start inserting this new data in your existing structure without creating any new pre-defined columns or pre-defined structure.

Reference source : https://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases

Are people really using NoSQL database ?

- If you have ever used Amazon, Yahoo or Google then you have had your data served via a NoSQL solution.
- If you have used eBay or Twitter you have indirectly used data stores that bear little resemblance to traditional databases, (for example eBay does not use transactions and Twitter uses a custom graph database to track who follows whom).
- Microsoft.NET Framework has been written on MongoDB (mongodb.org) and CouchDB (couchdb.apache.org)—as well as RavenDB (ravendb.net).
- Usually the question really means, are people like to use it?. The answer is that if you are facing issues dealing with certain types of data then there is potential competitive advantage to be gained by looking at a NoSQL solution. The area is new enough that most businesses would feel uncomfortable running critical work anywhere other than in mature relational data stores, even if those relational stores cause a lot of issues in their own right.

Performance of NoSQL Database

- Ben Scofield rated different categories of NoSQL databases as follows:

Data model	Performance	Scalability	Flexibility	Complexity	Functionality
Key–value store	high	high	high	none	variable (none)
Column-oriented store	high	high	moderate	low	minimal
Document-oriented store	high	variable (high)	high	low	variable (low)
Graph database	variable	variable	high	high	graph theory
Relational database	variable	variable	low	moderate	relational algebra

End of Topic 9