Unit VI

Structured/Unstructured Data

Structured Data

- Data that resides in a fixed field within a record or file is called structured data. This includes data contained in relational databases.

- Structured data first depends on creating a data model – a model of the types of business data that will be recorded and how they will be stored, processed and accessed.

- This includes defining what fields of data will be stored and how that data will be stored: data type (numeric, alphabetic, name, date, address) and any restrictions on the data input (number of characters; restricted to certain terms such as Mr., Ms. or Dr.; M or F).

- Structured data has the advantage of being easily entered, stored, queried and analyzed.

- **Structured data is** often managed using Structured Query Language (SQL) – a programming language created for managing and querying data in relational database management systems. Originally developed by IBM in the early 1970s and later developed commercially by Relational Software, Inc. (now Oracle Corporation).

- **Unstructured data** is all those things that can't be so readily classified : photos and graphic images, videos, streaming instrument data, webpages, pdf files, PowerPoint presentations, emails, blog entries, wikis and word processing documents, books, journals, documents, metadata, health records, audio, analog **data**, images, files, body of an e-mail message.

- **Semi-Structured data** is a cross between the two. It is a type of structured data, but **lacks the strict data model structure.** With semi-structured data, tags or other types of markers are used to identify certain elements within the data, but the data doesn't have a strict structure.

- For example, word processing software now can include metadata showing the author's name and the date created, with the bulk of the document just being unstructured text.

- **Emails** have the sender, recipient, date, time and other fixed fields added to the unstructured data of the email message content and any attachments.

- Photos or other graphics can be tagged with keywords such as the creator, date, location and keywords, making it possible to organize and locate graphics. **XML and other markup languages** are often used to manage semi-structured data.

- **Structured Data**
- data is organized in semantic chunks (entities)
- similar entities are grouped together (relations or classes)
- entities in the same group have the same descriptions (attributes)
- descriptions for all entities in a group (schema)
  - have the same defined format
  - have a predefined length
  - are all present
  - and follow the same order

- **Semi-Structured Data**
- data is available electronically in
  - database systems
  - file systems, e.g., bibliographic data, Web data
- semi-structured data
  - similar entities are grouped together
  - entities in same group may not have same attributes
  - order of attributes not necessarily important
  - not all attributes may be required
  - size of same attributes in a group may differ
  - type of same attributes in a group may differ

- **Unstructured Data**

-data can be of any type

-not necessarily following any format or sequence

-does not follow any rules

-is not predictable

-examples include

  - text
  - video
  - sound
  - images

Structured, Semi-structured, and Unstructured data

- **Structured data**
  - Information stored DB
  - Strict format
  - **Example-Databases, DataWarehouse,Enterprise systems(ERP)**

- **Semi-structured data**
  - Data may have certain structure but not all information collected has identical structure
  - Some attributes may exist in some of the entities of a particular type but not in others
  - **Example:** XML,E-Mail

- **Unstructured data**
  - Very limited indication of data type
    - **Example** a simple text document, Analog data,GPS Tracking information,Audio/video data.

- **Limitations for SQL database**
- **Scalability**: Users have to scale relational database on powerful servers that are expensive and difficult to handle. To scale relational database it has to be distributed on to multiple servers.

- **Complexity**: In SQL server's data has to fit into tables anyhow. If your data doesn't fit into tables, then you need to design your database structure that will be complex and again difficult to handle.

- Relational Database Management Systems that use SQL are Schema –Oriented i.e. the structure of the data should be known in advance ensuring that the data adheres to the schema.
- Examples of such predefined schema based applications that use SQL include Payroll Management System, Order Processing, and Flight Reservations.
- It is not possible for SQL to process unpredictable and unstructured information. However, Big Data applications, demand for an occurrence-oriented database which is highly flexible and operates on a schema less data model.
- SQL Databases are vertically scalable – this means that they can only be scaled by enhancing the horse power of the implementation hardware, thereby making it a costly deal for processing large batches of data.
- IT enterprises need to increase the RAM, SSD, CPU, etc., on a single server in order to manage the increasing load on the RDBMS.
- With increasing size of the database or increasing number of users, Relational Database Management Systems using SQL suffer from serious performance bottlenecks -making real time unstructured data processing a hard row to hoe.
- With Relational Database Management Systems, built-in clustering is difficult due to the ACID properties of transactions.

NOSQL

- A **NoSQL** or **Not Only SQL** database provides a mechanism for storage and retrieval of data that is modeled other than the tabular relations used in relational databases.

- NoSQL = "Not Only SQL"

  Not every data management/analysis problem
  is best solved exclusively using a traditional DBMS

- The data structure (e.g. key-value, graph, or document) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS.

- Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability.

What is NoSQL database

- Relational databases, on the other hand, were not designed to cope with the scale and the challenges that face modern applications.

- The basic quality of NoSQL is that, it **may not require fixed table schemas, usually avoid join operations**, and typically scale horizontally.

- NoSQL includes a wide variety of different database technologies and were developed in response to a rise in the volume of data stored about users, objects and products, the frequency in which this data is accessed, and performance and processing needs.

❑ Four main types of NoSQL databases/Data Models

1)Key / Value databases:

- the model is reduced to a simple hash table which consists of key / value pairs.
- It is often easily distributed across multiple servers. As the name implies, a key-value store is a system that stores values indexed for retrieval by keys.
- These systems can hold structured or unstructured data.
- The most famous products of this group include Redis, Dynamo, and Riak,BerkeleyDB.
- **Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or "key"), together with its value.

2) Column-oriented databases:

- The data are stored in sections of columns which offers more flexibility and easy aggregation.
- Rather than store sets of information in a heavily structured table of columns and rows with uniform sized fields for each record, as is the case with relational databases, column-oriented databases contain one extendable column of closely related data.
- i.e. columns are logically grouped into column families.
- RDBMS stores a single column as a continous disk entry.
- Different rows are stored in different places on disk while columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes search/access faster.
- Facebook's Cassandra, BigTable from Google, and Amazon's SimpleDB ,HBase are the examples which belongs to this group.

3)Document databases:

- The data model consists of document collections where individual documents can have multiple fields, without necessarily defining a schema.
- A document store is similar to a key value store in that stored objects are character string keys.
- The difference is that the values being stored are referred to as **documents** provide some encodings like XML, JSON,BSON
- **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.
- The best known and used are **MongoDB** and CouchDB.

4)Graph databases:

- The domain model consists of vertices interconnected by edges which creates a rich graph structure.
- **Graph stores** are used to store information about networks, such as social connections. Graph stores include Neo4J, OrientDB and HyperGraphDB.
- Scalability concerns are perfectly address by Graph store.

• Categories of NoSQL database

Category	Description	Name of the database
Document Oriented Data is stored as documents.	An example format may be like - FirstName="XYZ", Address="St. Xavier's Road", Spouse=[{Name:"Kiran"}], Children=[{Name:"Rihit", Age:8}]	MongoDB, CouchDB, RethinkDB, RavenDB etc.
XML database	Data is stored in XML format	BaseX, eXist, MarkLogic Server etc.
Graph databases	Data is stored as a collection of nodes, where nodes are analogous to objects in a programming language. Nodes are connected using edges.	Allegro, Neo4J, OrientDB, Virtuoso.

Key-value store	**In Key-value-store category of NoSQL database, an user can store data in schema-less way. A key may be strings, hashes, lists, sets, sorted sets and values are stored against these keys**	**Dynamo, FoundationDB, MemcacheDB, Redis, Riak. etc. -**
Column store	A column is a key value pair, where the key is an identifier and the value stores values related to the key (identifier).	Accumulo, Cassandra, HBase etc.

✔ There is a large number of companies using NoSQL. To name a few :

- Google
- Facebook
- Mozilla
- Adobe
- Foursquare
- LinkedIn
- McGraw-Hill Education
- Vermont Public Radio

❖ The Benefits of NoSQL

- When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address.
- Large volumes of structured, semi-structured, and unstructured data.

- Object-oriented programming that is easy to use and flexible .

- Efficient, scale-out architecture.

- NoSQL database also trades off “ACID” (atomicity, consistency, isolation and durability).

- **No schema required**: Data can be inserted in a NoSQL database without first defining a rigid database schema. This provides immense application flexibility.

- **Auto elasticity:** NoSQL automatically spreads your data onto multiple servers without requiring application assistance. Servers can be added or removed from the data layer automatically.

❖ Advantages of NoSQL database

1.) NoSQL databases generally process data faster than relational databases.

2.) NoSQL databases are also often faster because their data models are simpler.

3.) Major NoSQL systems are flexible enough to better enable developers to use the applications in ways that meet their needs.

❖ SQL vs NoSQL: High-Level Differences

- SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL database are primarily called as non-relational or distributed database.

- SQL databases are table based databases whereas NoSQL databases are **document based, key-value pairs, graph databases or wide-column stores.**

- This means that SQL databases represent data in form of tables which consists of n number of rows of data whereas NoSQL databases are the collection of **key-value pair, documents, graph databases or wide-column stores** which do not have standard schema definitions.

- SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data.

- SQL databases are **vertically scalable** whereas the NoSQL databases are **horizontally scalable**.

- SQL databases uses SQL (structured query language) for defining and manipulating the data, which is very powerful.

- In NoSQL database, queries are focused on collection of documents. Sometimes it is also called as UnQL (Unstructured Query Language). The syntax of using UnQL varies from database to database.

- **SQL database examples:** MySql, Oracle,Postgres,Sqlite and MS-SQL.

- **NoSQL database examples:** MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb

- **For the type of data to be stored:** SQL databases are not best fit for **hierarchical data storage.** But, NoSQL database fits better for the hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data.

- NoSQL database are highly preferred for large data set (i.e for big data). HBase is an example.

- **For scalability:** In most typical situations, SQL databases are vertically scalable. You can manage increasing load by increasing the CPU, RAM, etc, on a single server.

- On the other hand, NoSQL databases are horizontally scalable. You can just add few more servers easily in your NoSQL database infrastructure to handle the large traffic.

- **For properties:** SQL databases emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance)

- **For DB types:** On a high-level, we can classify SQL databases as either open-source or close-sourced from commercial vendors. NoSQL databases can be classified on the basis of way of storing data as graph databases, key-value store databases, document store databases, column store databases and XML databases.

Comparative Study of SQL and NOSQL

Types	One type (SQL database) with minor variations	Many different types including key-value stores, document databases, wide-column stores, and graph databases
Development History	Developed in 1970s to deal with first wave of data storage applications	Developed in 2000s to deal with limitations of SQL databases, particularly concerning scale, replication and unstructured data storage
Examples	MySQL, Postgres, Oracle Database	MongoDB, Cassandra, HBase, Neo4j

Data Storage Model	Individual records (e.g., "employees") are stored as rows in tables, with each column storing a specific piece of data about that record (e.g., "manager," "date hired," etc.), much like a spreadsheet. Separate data types are stored in separate tables, and then joined together when more complex queries are executed. For example, "offices" might be stored in one table, and "employees" in another. When a user wants to find the work address of an employee, the database engine joins the "employee" and "office" tables together to get all the information necessary.	Varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ("key" and "value"), with more complex information sometimes stored within the "value" columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single "document" in JSON, XML, or another format, which can nest values hierarchically.

Schemas	**Structure and data types are fixed in advance. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.**	**Typically dynamic. Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.**
Scaling	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required.	Horizontally, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary

Development Model	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
Consistency	Can be configured for strong consistency	Depends on product. Some provide strong consistency (e.g., MongoDB)

Advantages of NoSQL

1: Elastic scaling

2: Big data

Today, the volumes of "big data" that can be handled by NoSQL systems, such as Hadoop.

3: Economics

NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data and transaction volumes, while RDBMS tends to rely on expensive proprietary servers and storage systems.

The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing you to store and process more data at a much lower price point.

4 Flexible data models

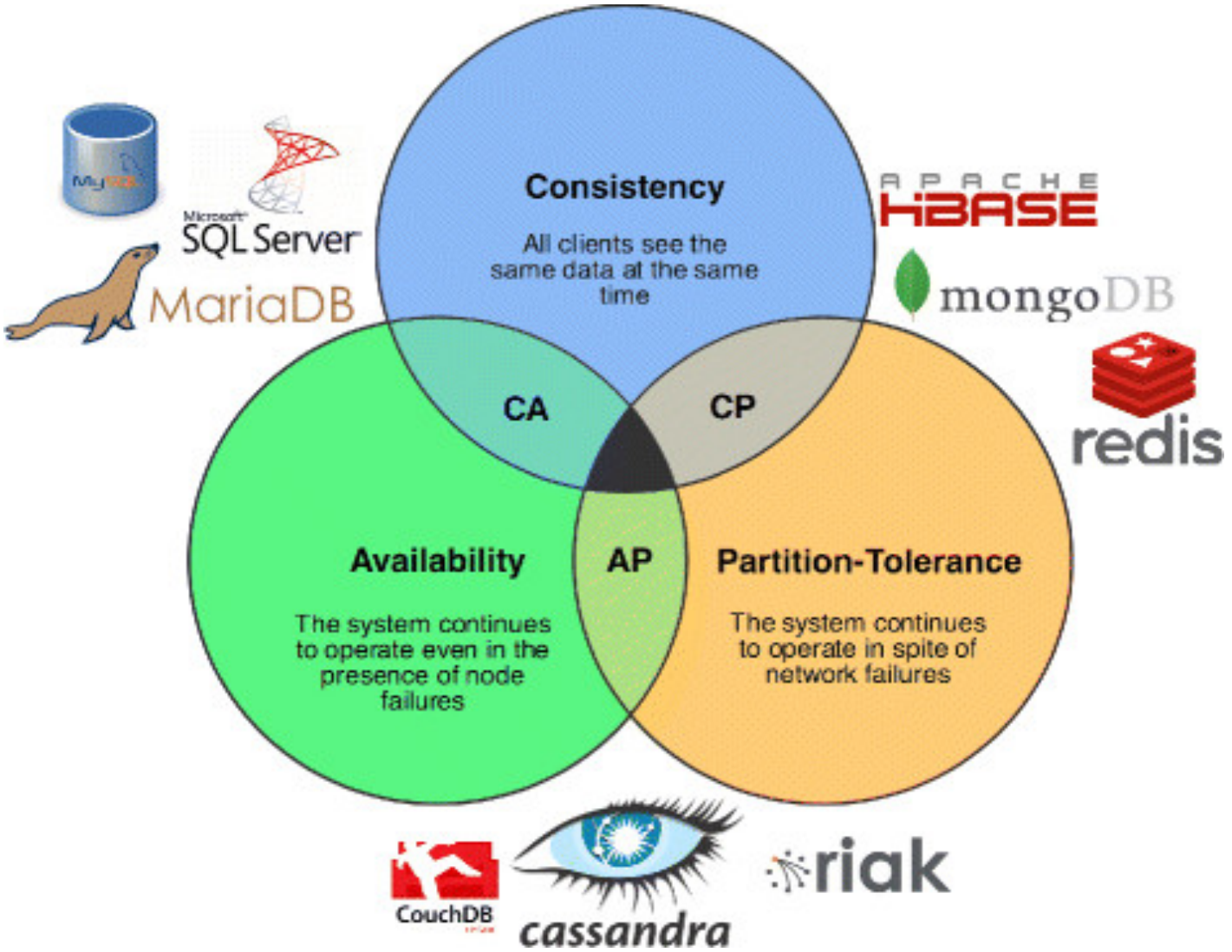
- Minor changes to the data model of an RDBMS have to be carefully managed.
- NoSQL databases have far more relaxed -- or even nonexistent -- data model restrictions.

CAP Theorem

- While designing applications for a distributed architecture, some basic requirements should be present in relations.

- **C-**Consistency
- **A-**Availability
- **P-**Partition tolerance

CAP Theorem (Brewer's Theorem)



CAP Theorem (Brewer's Theorem)

- it is impossible for a ***distributed*** computer system to simultaneously provide all three of the following guarantees:
  - *Consistency*: all nodes see the same data at the same time
  - *Availability*: Node failures do not prevent other survivors from continuing to operate (a guarantee that every request receives a response about whether it succeeded or failed)
  - *Partition tolerance*: the system continues to operate despite arbitrary partitioning due to network failures (e.g., message loss)
- A distributed system can satisfy any two of these guarantees at the same time but not all three.

- **C-**Consistency:-After update operation every client should see the same data- **data consistency.**
- **A-**Availability:-system should always on so service guarantee availability.
- **P-**Partition tolerance:-The system continues to function even the communication among the servers is unreliable.

- **CA-**RDBMS
- **CP-**MongoDB,HBase,Redis
- **AP-**Cassandra,CouchDB,DynamoDB,Riak

BASE

(Basically Available, Soft-State, Eventually Consistent)

- **Basic Availability:** fulfill request, even in partial consistency. Database should appears to work.
- **Soft State:** data storage may not contain the write consistent state.

Different replicas on different shards have to be mutually consistent at all the time.

- **Eventual Consistency:** at some point in the future, data will converge to a consistent state; delayed consistency, as opposed to immediate consistency of the ACID properties.
- A BASE model normally focuses on availability as it is important for scaling.
- But it does not guranteed for data consistency.

References

- http://bigdata.black/infrastructure/storage/sql-nosql-differences/

END