Big Data | Meerza Ahmed

Tuesday, October 31, 2023 1:00 PM

Lecture 11: Link | Syllabus

Class Week 11/15

Database Systems by Coronel & Morris

Chapter 14: Big Data Analytics & NoSQL

* 14.1 *Big Data*

Big Data generally refers to a set of data that displays the characteristics of volume, velocity, and variety (the "3 Vs") to an extent that makes the data unsuitable for management by a relational database management system. Although the term Big Data lacks a consistent definition, there is a set of characteristics generally associated with it.

These characteristics can be defined as follows :

- *Volume* - refers to the amounts of data being stored
- *Velocity* - refers not only to the speed with which data grows but also to the need to process this data quickly in order to generate insights
- *Variety* - refers to the fact that the data being collected comes in multiple different data formats

* Other characteristics have also debated as being equally important as the 3 Vs, some of them also keeping in the spirit of additional Vs include; Variability referring to the changes in the meaning of data based on the context, Veracity which refers to the trustworthiness of the data, Value also known as Viability refers to the degree to which the data can be analyzed to provide meaningful information that can add value to the organization, and Visualization which is the ability graphically represent the data in way that makes it understandable to non-data folk.

* 14.2 *Hadoop*

Scaling out into clusters based on low-cost, commodity servers is the dominant approach that organizations are currently pursuing for Big Data management. As a result, new technologies not based on the relational model have been developed. Hadoop is a Java-based framework for distributing and processing very large data sets across clusters of computers, it has become the de facto standard for most Big Data storage and processing While the Hadoop framework includes many parts, the two most important components are the Hadoop Distributed File System (HDFS) and MapReduce.

* *Impala* was the first SQL-on-Hadoop application. It was produced by Cloudera as a query engine that supports SQL queries that pull data directly from HDFS. Prior to Impala, if an organization needed to make data from Hadoop available to analysts through an SQL interface, data would be extracted from HDFS and imported into a relational database. With Impala, analysts can write SQL queries directly against the data while it is still in HDFS. Impala makes heavy use of in-memory caching on data nodes. It is generally considered an appropriate tool for processing large amounts of data into a relatively small result set.

* 14.2-a *HDFS*

Hadoop Distributed File System (HDFS) is a low-level distributed file processing system, this means that it is used for data storage.

(HDFS) approach to distributing data is based on several key assumptions :

- High Volume - The volume of data in Big Data applications is expected to be in tera bytes, petabytes, or larger. Hadoop assumes that files in the HDFS will be extremely large. Data in the HDFS is organized into physical blocks, just as in other file storage.

- Write - Once, read - many - Using a write-once, read-many model simplifies concurrency issues and improves overall data throughput. Using this model, a file is created, writ ten to the file system, and then closed. Once the file is closed, changes cannot be made to its contents. This improves overall system performance and works well for the types of tasks performed by many Big Data applications

- Streaming access - Unlike transaction processing systems where queries often retrieve small pieces of data from several different tables, Big Data applications typically process entire files. Instead of optimizing the file system to randomly access individual data elements, Hadoop is optimized for batch processing of entire files as a continuous stream of data.

- Fault tolerance - Hadoop is designed to be distributed across thousands of low-cost, commodity computers. It is assumed that with thousands of such devices, at any point in time, some will experience hardware errors. Therefore, the HDFS is designed to replicate data across many different devices so that when one device fails, the data is still available from another device.

* 14.2-b *MapReduce*

MapReduce provides data processing to complement data storage of HDFS, MapReduce is the computing framework used to process large data sets across clusters. *Conceptually,* MapReduce follows the principle of divide and conquer. MapReduce takes a complex task, breaks it down into a collection of smaller subtasks, performs the subtasks all at the same time, and then combines the result of each subtask to produce a final result for the original task.

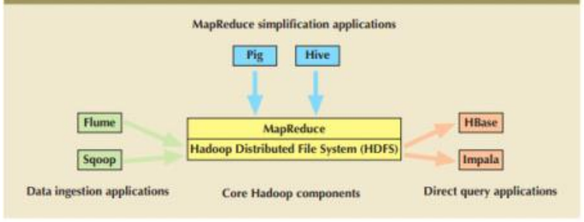
* A map function takes a collection of data and sorts and filters the data into a set of key-value pairs, it is performed by a program called a mapper. A reduce function takes a collection of key-value pairs, all with the same key value, and summarizes them into a single result, it is performed by a program called a reducer.

* MapReduce works like this :

1. Big data is split into file segments, held in a compute cluster made up of nodes (aka partitions)
2. A mapper task is run in parallel on all the segments (ie. in each node/partition, in each of its segments); each mapper produces output in the form of multiple (key,value) pairs
3. Key/value output pairs from all mappers are forwarded to a shuffler, which consolidates each key's values into a list (and associates it with that key)
4. The shuffler forwards keys and their value lists, to multiple reducer tasks; each reducer processes incoming key-value lists, and emits a single value for each key

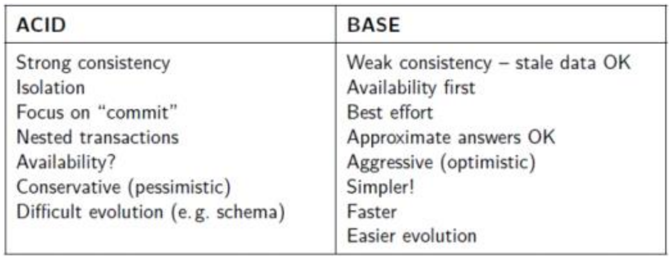
Optionally, before forwarding to shufflers, a 'combiner' operation in each node can be set up to perform a local per-key reduction - if specified, this would be 'step 1.5', in the above workflow.
The cluster user (programmer) only needs to supply a mapper task and a reducer task, the rest is automatically handled!

FIGURE 14.6 A SAMPLE OF THE HADOOP ECOSYSTEM


MapReduce simplification applications
Pig
Hive
Flume
Sqoop
MapReduce
Hadoop Distributed File System (HDFS)
HBase
Impala
Data ingestion applications
Core Hadoop components
Direct query applications

*** 14.3** *NoSQL*

NoSQL is the unfortunate name given to a broad array of nonrelational database technologies that have developed to address the challenges represented by Big Data. The name is unfortunate in that it does not describe what the NoSQL technologies are, but rather what they are not. There are literally hundreds of products that can be considered as being under the broadly defined term NoSQL. Most of these fit roughly into one of four categories : key value data stores, document databases, column-oriented databases, and graph databases.



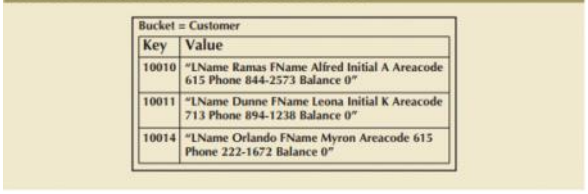
ACID	BASE
Strong consistency	Weak consistency – stale data OK
Isolation	Availability first
Focus on "commit"	Best effort
Nested transactions	Approximate answers OK
Availability?	Aggressive (optimistic)
Conservative (pessimistic)	Simpler!
Difficult evolution (e.g. schema)	Faster
	Easier evolution

* The NoSQL DBs are characterized by their 'BASE' property, in contrast with RDBMS' 'ACID' property.
BASE stands for BAsic availability (db is up most of the time), Soft state (of consistency - consistency is not guaranteed while data is written to a node, or between replicas), Eventual consistency (at a later point in time, by push or pull, data will become consistent across nodes and replicas).

*** 14.3-a** *Key-Value Databases*

Key-value (KV) databases are conceptually the simplest of the NoSQL data models. A KV database is a NoSQL database that stores data as a collection of key-value pairs. The key acts as an identifier for the value. The value can be anything such as text, an XML document, or an image. Key-value pairs are typically organized into "buckets." A bucket can roughly be thought of as the KV database equivalent of a table. A bucket is a logical grouping of keys. Key values must be unique within a bucket, but they can be duplicated across buckets. All data operations are based on the bucket plus the key

FIGURE 14.7 KEY-VALUE DATABASE STORAGE



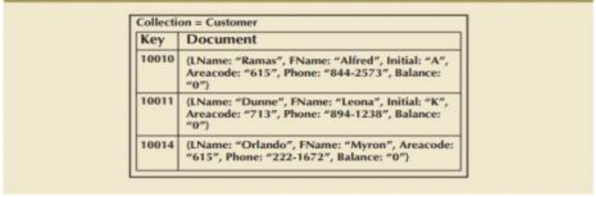
Bucket = Customer

Key	Value
10010	"LName Ramas FName Alfred Initial A Areacode 615 Phone 844-2573 Balance 0"
10011	"LName Dunne FName Leona Initial K Areacode 713 Phone 894-1238 Balance 0"
10014	"LName Orlando FName Myron Areacode 615 Phone 222-1672 Balance 0"

*** 14.3-b** *Document Databases*

Document databases are conceptually similar to key-value databases, and they can almost be considered a subtype of KV databases. A document database is a NoSQL database that stores data in tagged documents in key-value pairs. Unlike a KV database where the value component can contain any type of data, a document database always stores a document in the value component. The document can be in any encoded format, such as XML, JSON (JavaScript Object Notation), or BSON (Binary JSON). Another important difference is that while KV databases do not attempt to understand the content of the value component, document databases do. Tags are named portions of a document. For example, a document may have tags to identify which text in the document represents the title, author, and body of the document. Within the body of the document, there may be additional tags to indicate chapters and sections. Despite the use of tags in documents, document databases are considered schema-less, that is, they do not impose a predefined structure on the data that is stored.
* Just as KV databases group key-value pairs into logical groups called buckets, document databases group documents into logical groups called collections. While a document may be retrieved by specifying the collection and key, it is also possible to query based on the contents of tags

FIGURE 14.8 DOCUMENT DATABASE TAGGED FORMAT



Collection = Customer

Key	Document
10010	{LName: "Ramas", FName: "Alfred", Initial: "A", Areacode: "615", Phone: "844-2573", Balance: "0"}
10011	{LName: "Dunne", FName: "Leona", Initial: "K", Areacode: "713", Phone: "894-1238", Balance: "0"}
10014	{LName: "Orlando", FName: "Myron", Areacode: "615", Phone: "222-1672", Balance: "0"}

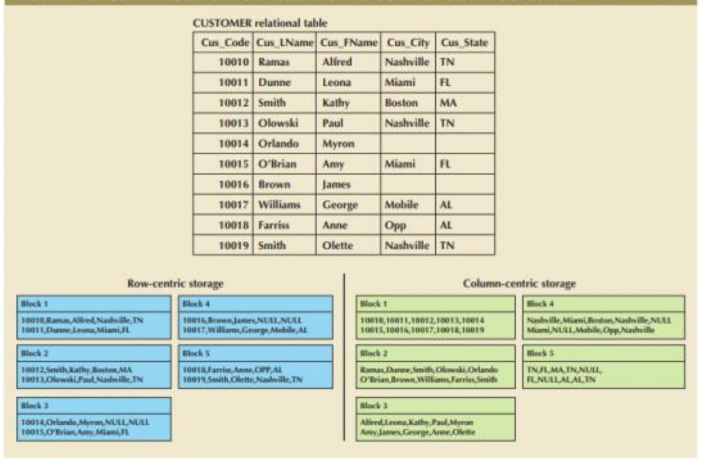
*** 14.3-c** *Column-Oriented Databases*

The term Column-oriented database can refer to two different sets of technologies that are often confused with each other. In one sense, column-oriented database or columnar database can refer to traditional, relational database technologies that use column-centric storage instead of row-centric storage. Relational databases present data in logical tables; however, the data is actually stored in data blocks containing rows of data. All of the data for a given row is stored together in sequence with many rows in the same data block.

* The other use of the term *column-oriented database*, also called column family database, is to describe a type of NoSQL database that takes the concept of column-centric storage beyond the confines of the relational model. As NoSQL databases, these products do not require the data to conform to predefined structures nor do they support SQL for queries.

* As more columns are added, it becomes clear that some columns form natural groups , these groupings are used to create super columns. A super column is a group of columns that are logically related.

FIGURE 14.9 COMPARISON OF ROW-CENTRIC AND COLUMN-CENTRIC STORAGE

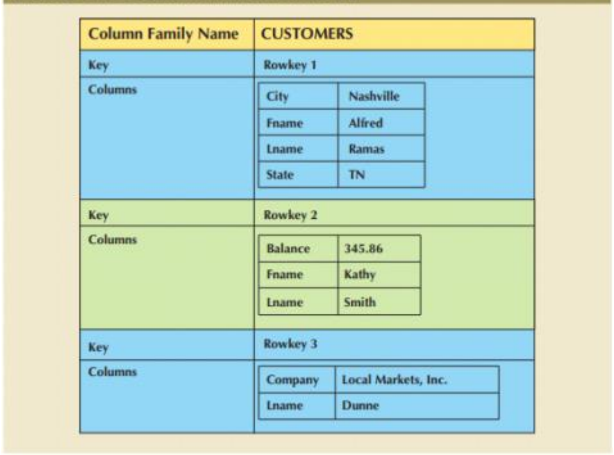


CUSTOMER relational table

Cus_Code	Cus_LName	Cus_FName	Cus_City	Cus_State
10010	Ramas	Alfred	Nashville	TN
10011	Dunne	Leona	Miami	FL
10012	Smith	Kathy	Boston	MA
10013	Olowski	Paul	Nashville	TN
10014	Orlando	Myron		
10015	O'Brian	Amy	Miami	FL
10016	Brown	James		
10017	Williams	George	Mobile	AL
10018	Farriss	Anne	Opp	AL
10019	Smith	Olette	Nashville	TN

Row-centric storage
Block 1
10010,Ramas,Alfred,Nashville,TN
10011,Dunne,Leona,Miami,FL
Block 2
10012,Smith,Kathy,Boston,MA
10013,Olowski,Paul,Nashville,TN
Block 3
10014,Orlando,Myron,NULL,NULL
10015,O'Brian,Amy,Miami,FL
Block 4
10016,Brown,James,NULL,NULL
10017,Williams,George,Mobile,AL
Block 5
10018,Farriss,Anne,OPP,AL
10019,Smith,Olette,Nashville,TN
Column-centric storage
Block 1
10010,10011,10012,10013,10014
10015,10016,10017,10018,10019
Block 2
Ramas,Dunne,Smith,Olowski,Orlando
O'Brian,Brown,Williams,Farriss,Smith
Block 3
Alfred,Leona,Kathy,Paul,Myron
Amy,James,George,Anne,Olette
Block 4
Nashville,Miami,Boston,Nashville,NULL
Miami,NULL,Mobile,Opp,Nashville
Block 5
TN,FL,MA,TN,NULL,
FL,NULL,AL,AL,TN

FIGURE 14.10 COLUMN FAMILY DATABASE

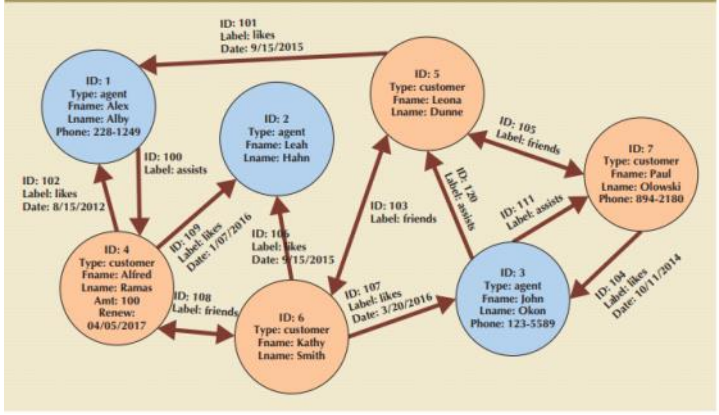


*** 14.3-c** *Graph Databases*

Graph database is a NoSQL database based on graph theory to store data about relationship-rich environments. Graph theory is a mathematical and computer science field that models relationships, or edges, between objects called nodes. Modeling and storing data about relationships is the focus of graph databases. The concept of graph theory immediately provides a rich source for algorithms and applications that have helped graph databases gain in sophistication very quickly.

* The primary components of graph databases are nodes, edges, and properties; A node corresponds to the idea of a relational entity instance. The node is a specific instance of something we want to keep data about. Properties are like attributes; they are the data that we need to store about the node. All agent nodes might have properties like first name and last name, but all nodes are not required to have the same properties. An edge is a relationship between nodes. Edges (shown as arrows in Figure 14.10) can be in one direction, or they can be bidirectional. Note that edges can also have properties. A query in a graph database is called a traversal. Instead of querying the database, the correct terminology would be traversing the graph.

FIGURE 14.11 GRAPH DATABASE REPRESENTATION



* 14.4 *Data Analytics*

Data analytics is a subset of business intelligence (BI) functionality that encompasses a wide range of mathematical, statistical, and modeling techniques with the purpose of extracting knowledge from data. Data analytics is used at all levels within the BI framework, including queries and reporting, monitoring and alerting, and data visualization. Data analytics represents what business managers really want from BI: the ability to extract actionable business insight from current events and foresee future problems or opportunities
In practice, data analytics is better understood as a continuous spectrum of knowledge acquisition that goes from discovery to explanation to prediction.
Data analytics tools can be grouped into two separate (but closely related and often overlapping) areas :

- Explanatory analytics focuses on discovering and explaining data characteristics and relationships based on existing data. Explanatory analytics uses statistical tools to formulate hypotheses, test them, and answer the how and why of such relationships-for example, how do past sales relate to previous customer promotions?
- Predictive analytics focuses on predicting future data outcomes with a high degree of accuracy. Predictive analytics uses sophisticated statistical tools to help the end user create advanced models that answer questions about future data occurrences-for example, what would next month's sales be based on a given customer promotion?

* You can think of explanatory analytics as explaining the past and present, while predictive analytics forecasts the future. However, you need to understand that both sciences work together; predictive analytics uses explanatory analytics as a stepping stone to create predictive models

* 14.4-a *Data Mining*

Data mining refers to analyzing massive amounts of data to uncover hidden trends, patterns, and relationships; to form computer models to simulate and explain the findings; and then to use such models to support business decision making. In other words, data mining focuses on the discovery and explanation stages of knowledge acquisition.
How is ML different from DM?
Machine learning is the process of TRAINING an algorithm on an EXISTING dataset in order to have it discover relationships (so as to create a model/pattern/trend), and USING the result to analyze NEW data.

* Data mining consists of four general phases :

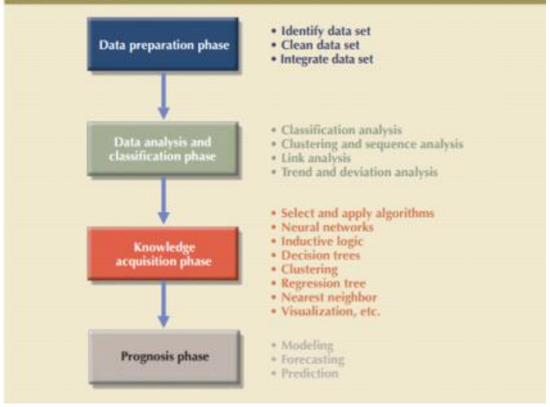
1. Data Preparation - In this phase the main data sets to be used by the data-mining operation are identified and cleansed of any data impurities. Because the data in the data warehouse is already integrated and filtered, the data warehouse usually is the target set for data-mining operations

2. Data Analysis & Classification - The data analysis and classification phase studies the data to identify common data characteristics or patterns.
   During this phase, the data-mining tool applies specific algorithms to find :
   - Data groupings, classifications, clusters, or sequences
   - Data dependencies, links, or relationships
   - Data patterns, trends, and deviations

3. Knowledge Acquisition - this phase uses the results of the data analysis and classification phase. During the knowledge acquisition phase, the data-mining tool (with possible intervention by the end user) selects the appropriate modeling or knowledge acquisition algorithms

4. Prognosis - Although many data-mining tools focus on the knowledge–discovery phase, others continue to the prognosis phase. In that phase the data-mining findings are used to predict future behavior and forecast business outcomes.

FIGURE 14.13 DATA-MINING PHASES



* *Data mining can be run in two modes:*

1. Guided. The end user guides the data-mining tool step by step to explore and explain known patterns or relationships. In this mode, the end user decides what techniques to apply to the data.
2. Automated. In this mode, the end user sets up the data-mining tool to run automatically and uncover hidden patterns, trends, and relationships. The data-mining tool applies multiple techniques to find significant relationships