**Assignment3.9**

**Problem Statement**:

Explain the below concepts with an example in brief.

● Nosql Databases

In the computing system (web and business applications), there are enormous data that comes out every day from the web. A large section of these data is handled by Relational database management systems (RDBMS). The idea of relational model came with E.F.Codd’s 1970 paper "A relational model of data for large shared data banks" which made data modeling and application programming much easier. Beyond the intended benefits, the relational model is well-suited to client-server programming and today it is predominant technology for storing structured data in web and business applications.

Classical relation database follow the ACID Rules

A database transaction, must be atomic, consistent, isolated and durable. Below we have discussed these four points.

**Atomic :** A transaction is a logical unit of work which must be either completed with all of its data modifications, or none of them is performed.

**Consistent :** At the end of the transaction, all data must be left in a consistent state.

**Isolated :** Modifications of data performed by a transaction must be independent of another transaction. Unless this happens, the outcome of a transaction may be erroneous.

**Durable :** When the transaction is completed, effects of the modifications performed by the transaction must be permanent in the system.

Often these four properties of a transaction are acronymed as **ACID**.

Distributed Systems

A distributed system consists of multiple computers and software components that communicate through a computer network (a local network or by a wide area network). A distributed system can consist of any number of possible configurations, such as mainframes, workstations, personal computers, and so on.The computers interact with each other and share the resources of the system to achieve a common goal.

Advantages of Distributed Computing

**Reliability (fault tolerance) :**  
The important advantage of distributed computing system is reliability. If some of the machines within the system crash, the rest of the computers remain unaffected and work does not stop.

**Scalability :**  
In distributed computing the system can easily be expanded by adding more machines as needed.

**Sharing of Resources :**  
Shared data is essential to many applications such as banking, reservation system. As data or resources are shared in distributed system, other resources can be also shared (e.g. expensive printers).

**Flexibility :**  
As the system is very flexible, it is very easy to install, implement and debug new services.

**Speed :**  
A distributed computing system can have more computing power and it's speed makes it different than other systems.

**Open system :**  
As it is open system, every service is equally accessible to every client i.e. local or remote.

**Performance :**  
The collection of processors in the system can provide higher performance (and better price/performance ratio) than a centralized computer.

Disadvantages of Distributed Computing

**Troubleshooting :**  
Troubleshooting and diagnosing problems.

**Software :**  
Less software support is the main disadvantage of distributed computing system.

**Networking :**  
The network infrastructure can create several problems such as transmission problem, overloading, loss of messages.

**Security :**  
Easy access in distributed computing system increases the risk of security and sharing of data generates the problem of data security

Scalability

In electronics (including hardware, communication and software), scalability is the ability of a system to expand to meet your business needs. For example scaling a web application is all about allowing more people to use your application. You scale a system by upgrading the existing hardware without changing much of the application or by adding extra hardware.   
There are two ways of scaling horizontal and vertical scaling :

**Vertical scaling**  
To scale vertically (or scale up) means to add resources within the same logical unit to increase capacity. For example to add CPUs to an existing server, increase memory in the system or expanding storage by adding hard drive.

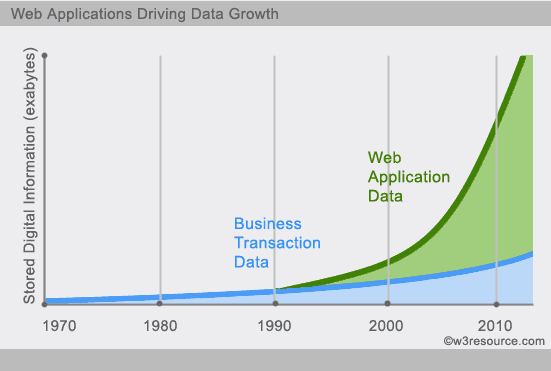
**Horizontal scaling**  
To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. In NoSQL system, data store can be much faster as it takes advantage of “scaling out” which means to add more nodes to a system and distribute the load over those nodes.

What is NoSQL?

NoSQL is a non-relational database management systems, different from traditional relational database management systems in some significant ways. It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users). These type of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

Why NoSQL?

In today’s time data is becoming easier to access and capture through third parties such as Facebook, Google+ and others. Personal user information, social graphs, geo location data, user-generated content and machine logging data are just a few examples where the data has been increasing exponentially. To avail the above service properly, it is required to process huge amount of data. Which SQL databases were never designed. The evolution of NoSql databases is to handle these huge data properly.



Example :

**Social-network graph:**

Each record: UserID1, UserID2

Separate records: UserID, first\_name,last\_name, age, gender,...

Task: Find all friends of friends of friends of ... friends of a given user.

**Wikipedia pages :**

Large collection of documents

Combination of structured and unstructured data

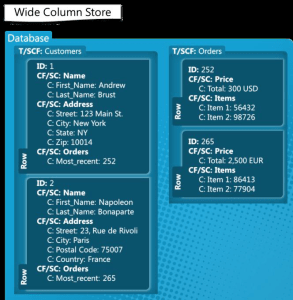
Task: Retrieve all pages regarding athletics of Summer Olympic before 1950.

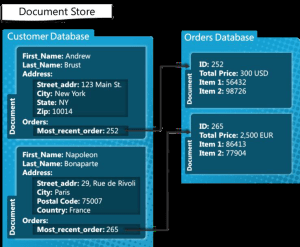
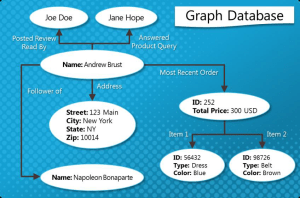
● Types of Nosql Databases

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.  NoSQL is often interpreted as Not-only-SQL to emphasize that they may also support [SQL](http://en.wikipedia.org/wiki/SQL)-like query languages.  Most NoSQL databases are designed to store large quantities of data in a [fault-tolerant](https://en.wikipedia.org/wiki/Fault_tolerance) way.

NoSQL is simply the term that is used to describe a family of databases that are all non-relational.  While the technologies, data types, and use cases vary wildly amount them, it is generally agreed that there are four types of NoSQL databases:

* Key-value stores – These databases pair keys to values.  An analogy is a files system where the path acts as the key and the contents act as the file.  There are usually no fields to update, instead, the entire value other than the key must be updated if changes are to be made.  The simplicity of this scales well but it can limit the complexity of the queries and other advanced features.  Examples are: [Dynamo](http://en.wikipedia.org/wiki/Dynamo_(storage_system)), [MemcacheDB](http://en.wikipedia.org/wiki/MemcacheDB), [Redis](http://en.wikipedia.org/wiki/Redis), [Riak](http://en.wikipedia.org/wiki/Riak), FairCom [c-treeACE](http://en.wikipedia.org/wiki/C-treeACE), [Aerospike](http://en.wikipedia.org/wiki/Aerospike_database), [OrientDB](http://en.wikipedia.org/wiki/OrientDB), [MUMPS](http://en.wikipedia.org/wiki/MUMPS), [HyperDex](http://en.wikipedia.org/wiki/HyperDex), [Azure Table Storage](http://azure.microsoft.com/en-us/services/storage/tables/) (see [Redis vs Azure](http://www.cunningplanning.com/?p=317))
* Graph stores – These excel at dealing with interconnected data.  Graph databases consist of connections, or edges, between nodes.  Both nodes and their edges can store additional properties such as key-value pairs.  The strength of a graph database is in traversing the connections between the nodes.  But they generally require all data to fit on one machine, limiting their scalability.  Examples include: [Allegro](http://en.wikipedia.org/wiki/AllegroGraph), [Neo4J](http://en.wikipedia.org/wiki/Neo4J), [InfiniteGraph](http://en.wikipedia.org/wiki/InfiniteGraph), [OrientDB](http://en.wikipedia.org/wiki/OrientDB), [Virtuoso](http://en.wikipedia.org/wiki/Virtuoso_Universal_Server), [Stardog](http://en.wikipedia.org/wiki/Stardog), Sesame
* Column stores – Relational databases store all the data in a particular table’s rows together on-disk, making retrieval of a particular row fast.  Column-family databases generally serialize all the values of a particular column together on-disk, which makes retrieval of a large amount of a specific attribute fast.  This approach lends itself well to aggregate queries and analytics scenarios where you might run range queries over a specific field.  Examples include: [Accumulo](http://en.wikipedia.org/wiki/Accumulo), [Cassandra](http://en.wikipedia.org/wiki/Apache_Cassandra), [Druid](http://en.wikipedia.org/wiki/Druid_(open-source_data_store)), [HBase](http://en.wikipedia.org/wiki/HBase), [Vertica](http://en.wikipedia.org/wiki/Vertica)
* Document stores – These databases store records as “documents” where a document can generally be thought of as a grouping of key-value pairs (it has nothing to do with storing actual documents such as a Word document).  Keys are always strings, and values can be stored as strings, numeric, Booleans, arrays, and other nested key-value pairs.  Values can be nested to arbitrary depths.  In a document database, each document carries its own schema — unlike an RDBMS, in which every row in a given table must have the same columns.  Examples include: [Lotus Notes](http://en.wikipedia.org/wiki/Lotus_Notes), [Clusterpoint](http://en.wikipedia.org/wiki/Clusterpoint), [Apache CouchDB](http://en.wikipedia.org/wiki/Apache_CouchDB), [Couchbase](http://en.wikipedia.org/wiki/Couchbase), [MarkLogic](http://en.wikipedia.org/wiki/MarkLogic), [MongoDB](http://en.wikipedia.org/wiki/MongoDB), [OrientDB](http://en.wikipedia.org/wiki/OrientDB), [Qizx](http://en.wikipedia.org/wiki/Qizx), [Cloudant](http://en.wikipedia.org/wiki/Cloudant), Azure DocumentDB (see [MongoDB vs. Azure DocumentDB](http://justazure.com/mongodb-vs-azure-documentdb/) and [An Overview of Microsoft Azure DocumentDB](https://msdn.microsoft.com/en-us/magazine/mt147238.aspx))

[](https://i0.wp.com/www.jamesserra.com/wp-content/uploads/2015/04/0247_image_090D4983.png)[](https://i1.wp.com/www.jamesserra.com/wp-content/uploads/2015/04/4477_image_523FC8B1.png)

[](https://i2.wp.com/www.jamesserra.com/wp-content/uploads/2015/04/2311_image_42DC40D5.png)

● CAP Theorem

For any distributed system, CAP Theorem reiterates the need to find balance between Consistency, Availability and Partition tolerance. Consistency means all the nodes see the same data at the same time. Availability implies that every request receives a response about whether it was successful or failed. It’s more of a handshaking mechanism in computer network methodology.

Coming to partition tolerance, the system continues to operate despite arbitrary message loss or failure of part of the system. Systems with partition tolerance feature works well despite physical network partitions.

According to CAP Theorem distributed systems can satisfy any two features at the same time but not all three features. Traditional systems like RDBMS provide consistency and availability. Column oriented databases like MongoDB, Hbase and Big Table provide features consistency and partition tolerance.



You must understand the CAP theorem when you talk about NoSQL databases or in fact when designing any distributed system. CAP theorem states that there are three basic requirements which exist in a special relation when designing applications for a distributed architecture.

**Consistency** - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation all clients see the same data.

**Availability** - This means that the system is always on (service guarantee availability), no downtime.

**Partition Tolerance** - This means that the system continues to function even the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

In theoretically it is impossible to fulfill all 3 requirements. CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements. Therefore all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem. Here is the brief description of three combinations CA, CP, AP :

**CA -** Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.   
**CP -**Some data may not be accessible, but the rest is still consistent/accurate.   
**AP -** System is still available under partitioning, but some of the data returned may be inaccurate.



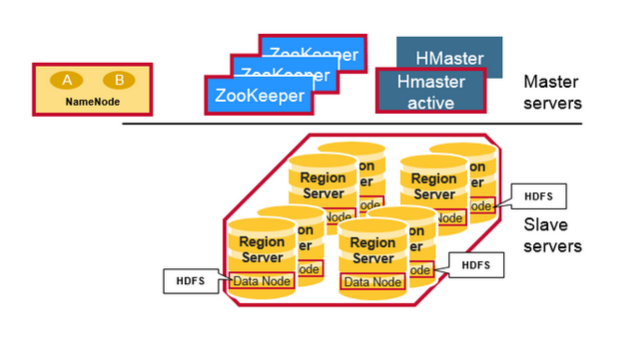
● HBase Architecture

**HBase Architectural Components**

Physically, HBase is composed of three types of servers in a master slave type of architecture. Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state.

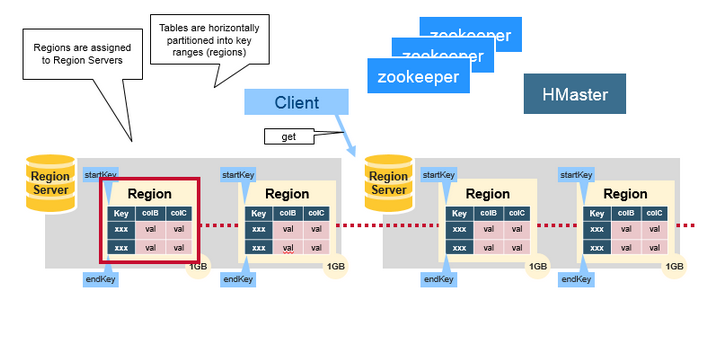
The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the RegionServers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.

The NameNode maintains metadata information for all the physical data blocks that comprise the files.



**Regions**

HBase Tables are divided horizontally by row key range into “Regions.” A region contains all rows in the table between the region’s start key and end key. Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes. A region server can serve about 1,000 regions.



**HBase HMaster**

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

A master is responsible for:

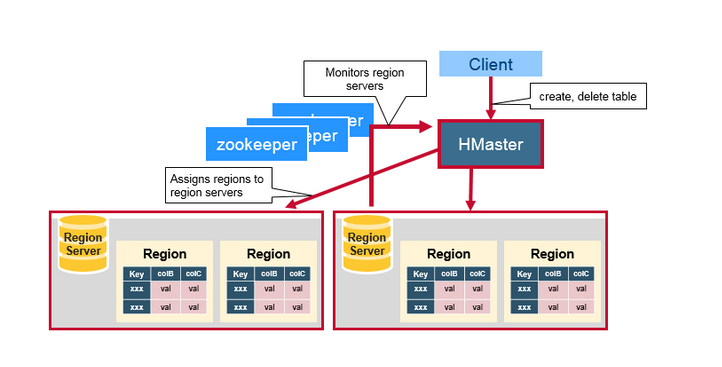
* Coordinating the region servers

- Assigning regions on startup , re-assigning regions for recovery or load balancing

- Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)

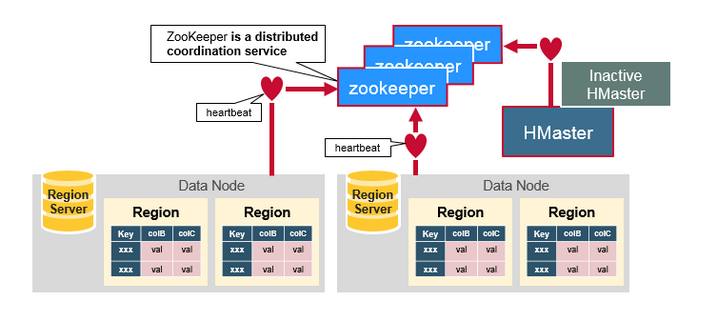
* Admin functions

- Interface for creating, deleting, updating tables



**ZooKeeper: The Coordinator**

HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.



● HBase vs RDBMS

