CS802 (B) Cloud Computing
Subject Notes

Unit-3 **CO3**

Data in the cloud: Relational databases, Cloud file systems: GFS and HDFS, Features and comparisons among GFS, HDFS, Big Table, H Base and Dynamo. Map-Reduce and extensions: Parallel computing, the Map-Reduce model: Parallel efficiencyofMap-Reduce, Relational operations, Enterprise batchprocessing, Example/Application of Map-Reduce.

DATA IN THE CLOUD: RELATIONAL DATABASES

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute.

The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys.

Important Aspects of Relational Databases:

SQL (Structured Query Language):

- SQL or Structured Query Language is the primary interface used to communicate with Relational Databases. SQL became a standard of the American National Standards Institute (ANSI) in 1986.
- The standard ANSI SQL is supported by all popular relational database engines, and some of these engines also have an extension to ANSI SQL to support functionality that is specific to that engine.
- SQL is used to add, update or delete rows of data, retrieving subsets of data for transaction processing and analytics applications, and to manage all aspects of the database.

Data Integrity:

- Data integrity is the overall completeness, accuracy, and consistency of data. Relational databases use a set of constraints to enforce data integrity in the database.
- These include primary Keys, Foreign Keys, ‘Not NULL’ constraint, ‘Unique’ constraint, ‘Default’ constraint, and ‘Check’ constraints. These integrity constraints help enforce business rules on data in the tables to ensure the accuracy and reliability of the data.

Transactions:

- A database transaction is one or more SQL statements that are executed as a sequence of operations that form a single logical unit of work.
- Transactions provide an "all-or-nothing" proposition, meaning that the entire transaction must complete as a single unit and be written to the database or none of the individual components of the transaction should go through.
- In the relation database terminology, a transaction results in a COMMIT or a ROLLBACK. Each transaction is treated coherently and reliably independent of other transactions.

ACID Compliance:

• Atomicity requires that either transaction be successfully executed or if a part of the transaction fails, then the entire transaction be invalidated.

• Consistency mandates the data written to the database as part of the transaction must adhere to all defined rules, and restrictions including constraints, cascades, and triggers.

• Isolation is critical to achieving concurrency control and makes sure each transaction is independent unto itself.

• Durability requires that all changes made to the database be permanent once a transaction is successfully completed.

CLOUD FILE SYSTEMS: GFS AND HDFS

Google File System (GFS):

• The Google File System (GFS) is a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware and it delivers high aggregate performance to a large number of clients.

• GFS provides a familiar file system interface, though it does not implement a standard API such as POSIX.

• Files are organized hierarchically in directories and identified by path-names. GFS supports the usual operations such as create, delete, open, close, read, and write files.

• GFS has snapshot and record appends operations. Snapshot creates a copy of a file or a directory tree at a low cost.

The architecture of GFS:

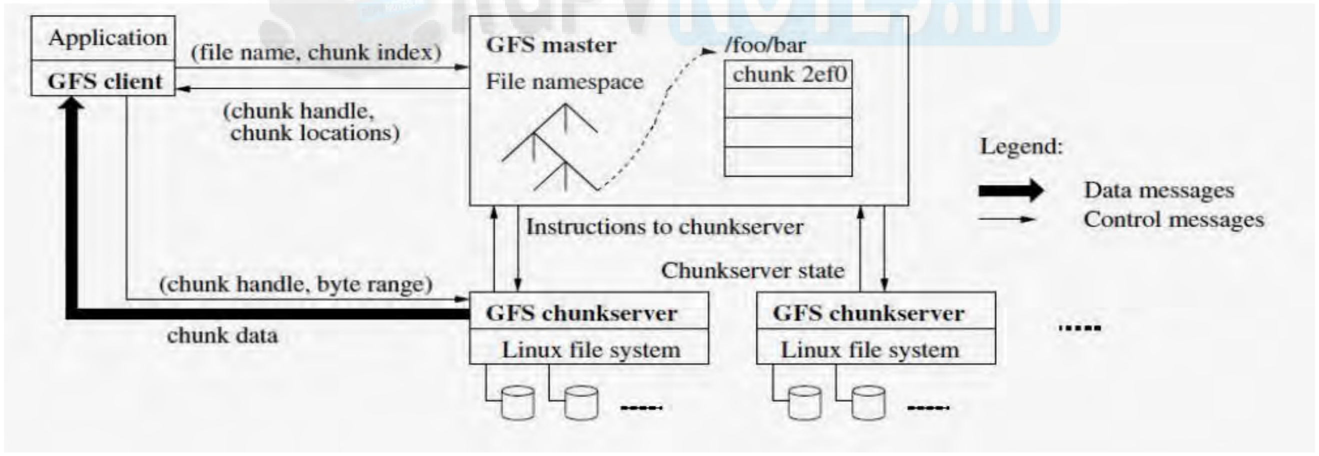


Figure 3.1: Google File System Architecture

• A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients.

• Each of these is typically a commodity Linux machine running a user-level server process.

• Files are divided into fixed-size chunks. Each chunk is identified by a fixed and globally unique 64-bit chunk handle assigned by the master at the time of chunk creation. Chunk servers store chunks on local disks as Linux files.

• The master maintains the metadata, which includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks.

• GFS client code linked into each application to implement the file system API and communicates with the master & chunk servers to read or write data on behalf of the application.

• Clients interact with the master for metadata operations, but all data-bearing communication goes directly to the chunk servers.

Hadoop Distributed File System (HDFS):

Hadoop File System was developed using distributed file system design.It runs on commodity hardware. HDFS holds a very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored redundantly to rescue the system from possible data losses in case of failure.

The architecture of HDFS:

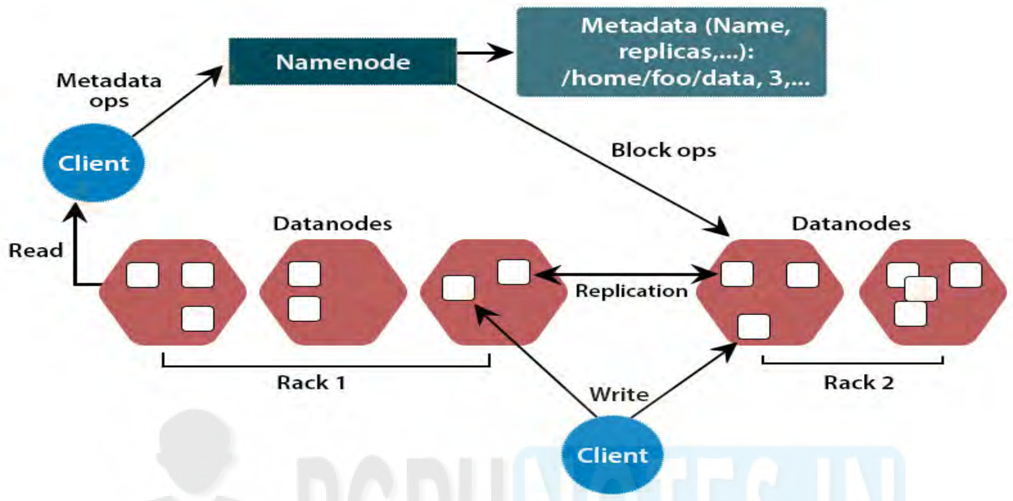


Figure 3.2: HDFS Architecture

HDFS follows the master-slave architecture and it has the following elements.

NameNode:

The NameNode is the commodity hardware that contains the GNU/Linux operating system and the NameNode software. It is software that can be run on commodity hardware. The system having the NameNode acts as the master server and it does the following tasks −

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

DataNode:

The DataNode is commodity hardware, having the GNU/Linux operating system and DataNode software. For every node (Commodity hardware/System) in a cluster, there will be a DataNode. These nodes manage the data storage of their system.

- DataNode perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the NameNode.

Block:

The user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and stored in individual data nodes. These file segments are called blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

Features of GFS:

- GFS was designed for high fault tolerance.
- Master and chunk servers can be restarted in a few seconds, each chunk is replicated at least three places and can tolerate at least two data crashes for a single chunk of data.
- The shadow master handles the failure of the GFS master.
- For data integrity, GFS makes checksums on every 64KB block in each chunk.
- GFS can achieve the goals of high availability, high performance, and implementation.

Features of HDFS:

- **Cost-effective:**

In HDFS architecture, the DataNodes, which store the actual data are inexpensive commodity hardware, thus reduces storage costs.

- **Large Datasets/ Variety and volume of data:**

HDFS can store data in any sizeand format (structured, unstructured).

- **Replication:**

Data Replication is one of the most important and unique features of HDFS. In HDFS, replication of data solves the problem of data loss in unfavourable conditions.The data is replicated across several machines in the cluster by creating replicas of blocks. The process of replication is maintained at regular intervals by HDFS.

- **High Availability:**

The High availability feature of Hadoop ensures the availability of data even during NameNode or DataNode failure.

- **High Throughput:**

Hadoop HDFS stores data in a distributed fashion, which allows data to be processed parallel on a cluster of nodes. This decreases the processing time and thus provides high throughput.

COMPARISON BETWEEN GFS AND HDFS

Properties	**GFS**	**HDFS**
Design Goals	• The main goal of GFS is to support large files. • Used for data-intensive computing. • Store data reliably, even when failures occur within chunk servers, master, or network partitions. • GFS is designed more for batch processing rather than interactive use by users.	• One of the main goals of HDFS is to support large files. • Used for data-intensive computing. • Store data reliably, even when failures occur within name nodes, data nodes, or network partitions. • HDFS is designed more for batch processing rather than interactive use by users.
Processes	• Master and chunk server	• Name node and Data node
File Management	• Files are organized hierarchically in directories and identified by path names. • GFS is exclusively for Google only.	• HDFS supports a traditional hierarchical file organization. • HDFS also supports third-party file systems such as Cloud Store and Amazon Simple Storage Service.
Security	• Google has dozens of datacenters for redundancy. These datacenters are in undisclosed locations and most are unmarked for protection. • Access is allowed to authorized employees and vendors only.	• HDFS security is based on the POSIX model of users and groups. • Security is limited to simple file permissions.

Database Files	• Bigtable is the database used by GFS. Bigtable is a proprietary distributed database of Google Inc.	• HBase provides Bigtable (Google)-like capabilities on top of Hadoop Core.
File Serving	• A file in GFS is comprised of fixed-sized chunks. The size of the chunk is 64MB.	• HDFS is divided into large blocks for storage and access, typically 64MB in size.
Cache Management	• Clients do cache metadata. • Neither the server nor the client caches the file data. • Chunks are stored as local files in a Linux system.	• HDFS uses distributed cache • It is a facility provided by the Map-Reduce frameworkto cache application-specific, large, read-only files (text, archives, jars, and so on) • Private and Public Distributed Cache Files.
Communication	• TCP connections are used for communication. Pipelining is used for data transfer over TCP connections.	• RPC based protocol on top of TCP/IP

BIG TABLE

- Cloud Bigtable is a sparsely populated table that can scale to billions of rows and thousands of columns, enabling you to store terabytes or even pentabytes of data. A single value in each row is indexed; this value is known as the row key.
- Cloud Bigtable is ideal for storing very large amounts of single-keyed data with very low latency. It supports high read and writes throughput at low latency, and it is an ideal data source for Map-Reduce operations.
- Cloud Bigtable is exposed to applications through multiple client libraries, including a supported extension to the Apache HBase library for Java.

Cloud Bigtable's powerful back-end servers offer several key advantages over a self-managed HBase installation:

- **Incredible scalability -** Cloud Bigtable scales in direct proportion to the number of machines in your cluster. A self-managed HBase installation has a design bottleneck that limits the performance after a certain threshold is reached. Cloud Bigtable does not have this bottleneck to scale the cluster up to handle more reads and writes.
- **Simple administration -** Cloud Bigtable handles upgrades and restarts transparently. It automatically maintains high data durability. To replicate data, simply add a second cluster to an instance, and replication starts automatically. No more managing replicas or regions; just design the table schemas, and Cloud Bigtable will handle.
- **Cluster resizing without downtime -** Increase the size of a Cloud Bigtable cluster for a few hours to handle a large load, and then reduce the cluster's size again—all without any downtime. After changing a cluster's size, it typically takes just a few minutes under load for Cloud Bigtable to balance performance across all of the nodes in the cluster.

HBASE AND DYNAMO

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data.

Architecture:

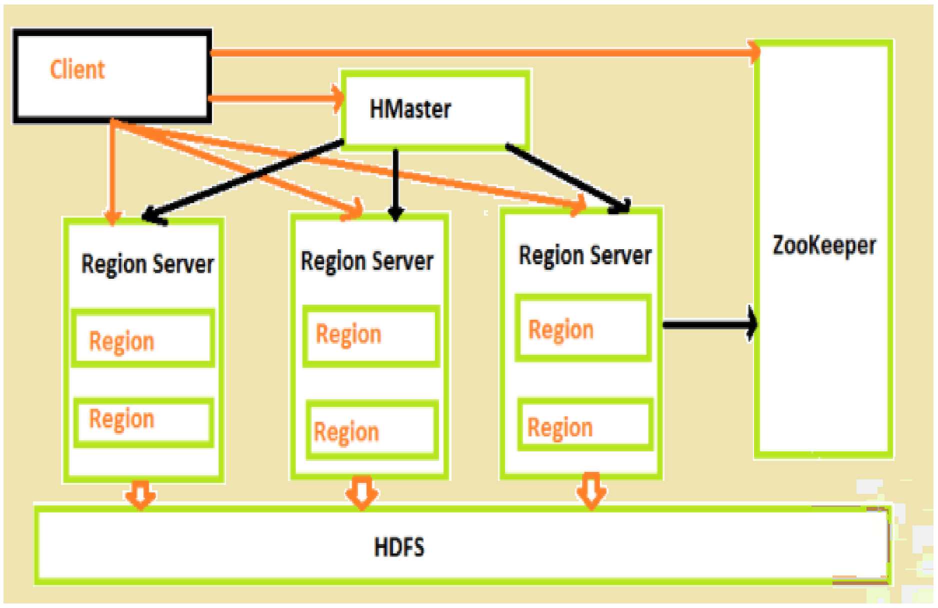


Figure 3.3: HBase Architecture

Master Server:

The master server -

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is also responsible for schema changes and other metadata operations such as creation of tables and column families.

Regions:

Regions are nothing but tables that are split up and spread across the region servers.

Region server:

The region servers have regions that -

- Communicate with the client and handle data-related operations.
- Handle read and write requests for all the regions under it.
- Decide the size of the region by following the region size thresholds.

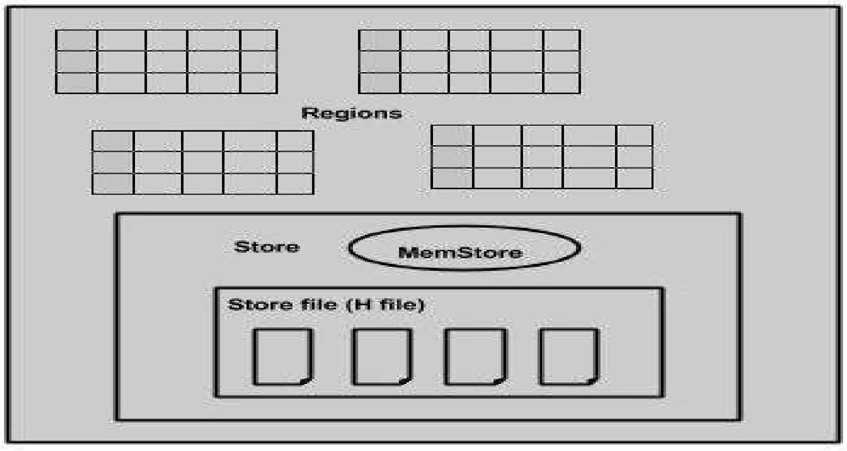


Figure 3.4: Regions and Stores

The store contains a memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

Zookeeper

• Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.

• Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.

• In addition to availability, the nodes are also used to track server failures or network partitions.

• Clients communicate with region servers via zookeeper.

• In pseudo and standalone modes, HBase itself will take care of the zookeeper.

MAP-REDUCE AND EXTENSIONS: PARALLEL COMPUTING

• The Map-Reduce programming model was developed at Google in the process of implementing large-scale search and text processing tasks on massive collections of web data stored using BigTable and the GFS distributed file system.

• The Map-Reduce programming model is designed for processing and generating large volumes of data via massively parallel computations.

• The underlying infrastructure to support this model needs to assume that processors and networks will fail, even during a particular computation, and build in support for handling such failures while ensuring the progress of the computations being performed.

Parallel Computing

Parallel computing is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource that has been applied to work is working at the same time.

Advantages of Parallel Computing:

• It saves time and money as many resources working together will reduce the time and cut potential costs.

• It can be impractical to solve larger problems on Serial Computing.

• It can take advantage of non-local resources when the local resources are finite.

• Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of the hardware.

Types of Parallelism:

• **Bit-level parallelism:** It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute to perform a task on large-sized data.

• **Instruction-level parallelism:** A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program.

• **Task Parallelism:** Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.

Applications of Parallel Computing:
• Databases and Data mining.
• Real-time simulation of systems.
• Science and Engineering.
• Advanced graphics, augmented reality, and virtual reality.

Limitations of Parallel Computing:
• It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
• The algorithms must be managed in such a way that they can be handled in the parallel mechanism.
• The algorithms or program must have low coupling and high cohesion. But it's difficult to create such programs.
• More technically skilled and expert programmers can code a parallelism based program well.

THE MAP-REDUCE MODEL
Map-Reduce is a processing technique and a program model for distributed computing based on java. The Map-Reduce algorithm contains two important tasks, namely Map and Reduce. The map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples.

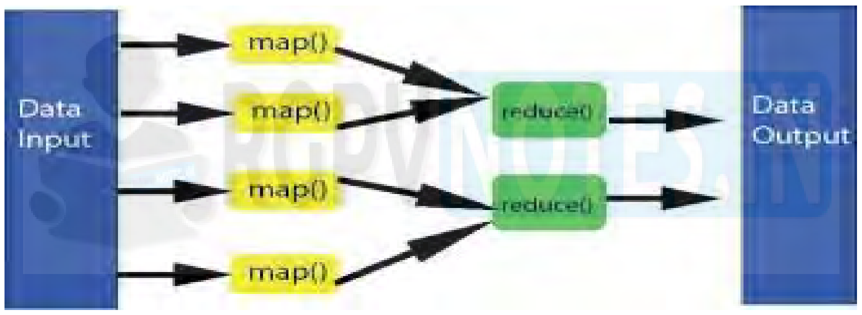
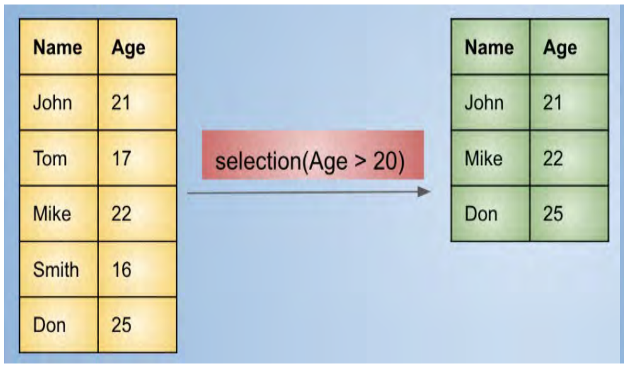


Figure 3.5: Map-Reduce Model

• **Parallel efficiency of map-reduce:**
Map-Reduce is the preferred cloud computing framework used in large data analysis and application processing. Map-Reduce frameworks currently in place suffer performance degradation due to the adoption of sequential processing approaches with little modification and thus exhibit underutilization of cloud resources.

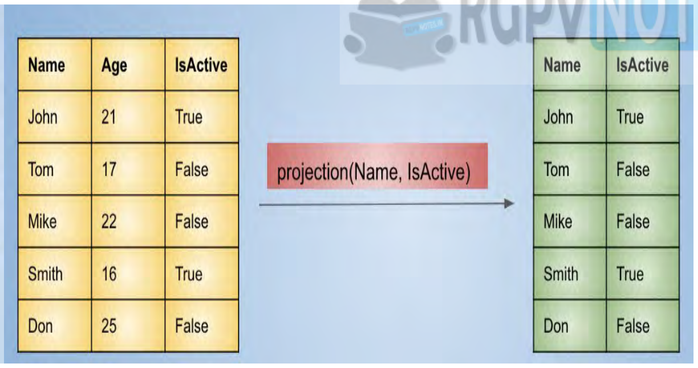
RELATIONALOPERATIONS
In relational algebra first need to know what a relation represents. As familiar with SQL there is no point in putting a long description here. A relation represents a database table. A major point that distinguishes SQL and relational algebra are that in relational algebra duplicate rows are implicitly eliminated which is not the case with SQL implementations.
• **Selection:** The SELECT operation is used for selecting a subset of the tuples according to a given selection condition.
• **Projection:** The projection operator helps to select some columns only in given table. It's analogous to SELECT in SQL.



Name	Age
John	21
Tom	17
Mike	22
Smith	16
Don	25

Name	Age
John	21
Mike	22
Don	25

Figure 3.6: Selection operations



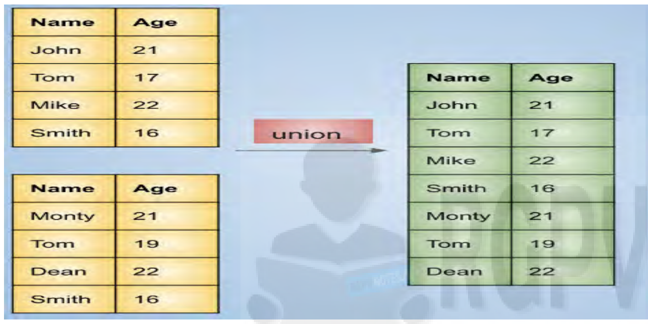
Name	Age	IsActive
John	21	True
Tom	17	False
Mike	22	False
Smith	16	True
Don	25	False

Name	IsActive
John	True
Tom	False
Mike	False
Smith	True
Don	False

Figure 3.7: Projection operations

- **Union:** It concatenates two tables vertically. Similar to UNION in SQL, but the duplicate rows are removed implicitly. The point to note in the below output table is that (Smith, 16) was a duplicate row so it appears only once in the output whereas (Tom, 17), (Tom, 19) appears as two, as those are not identical rows.

- **Intersection:** Same as INTERSECT in SQL. It intersects two tables and selects only the common rows.

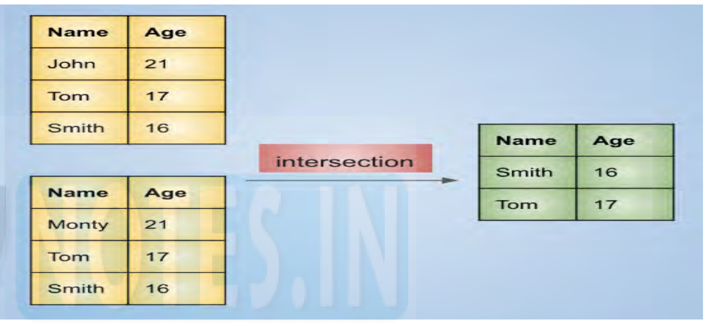


Name	Age
John	21
Tom	17
Mike	22
Smith	16

Name	Age
Monty	21
Tom	19
Dean	22
Smith	16

Name	Age
John	21
Tom	17
Mike	22
Smith	16
Monty	21
Tom	19
Dean	22

Figure 3.8: Union the two tables



Name	Age
John	21
Tom	17
Smith	16

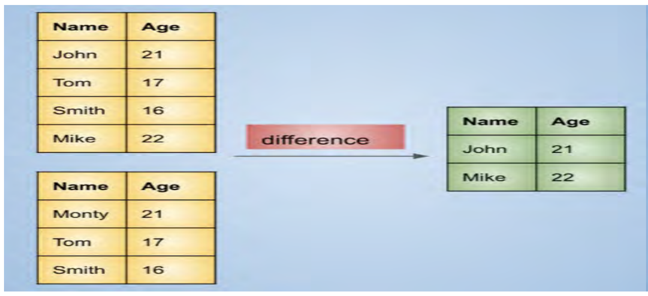
Name	Age
Monty	21
Tom	17
Smith	16

Name	Age
Smith	16
Tom	17

Figure 3.9: Intersection of two tables

- **Difference:** The rows that are in the first table but not in second are selected for output. That (Monty, 21) is not considered in the output as its present in the second table but not in first.

- **Natural Join:** Merge two tables based on some common column. It represents the INNER JOIN in SQL. But the condition is implicit on the column that is common in both tables. The output will only contain rows for which the values in the common column match. It will generate one row for every time the column value across two tables match as is the case below for Tom.

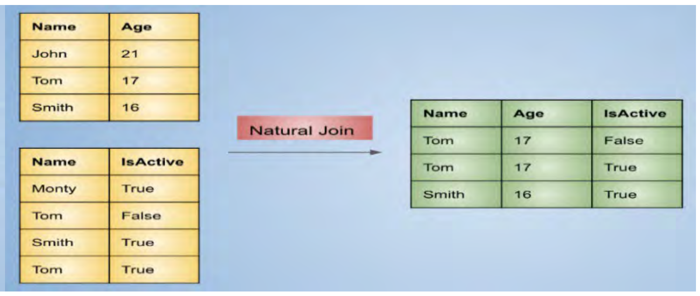


Name	Age
John	21
Tom	17
Smith	16
Mike	22

Name	Age
Monty	21
Tom	17
Smith	16

Name	Age
John	21
Mike	22

Figure 3.10: Difference between the two tables



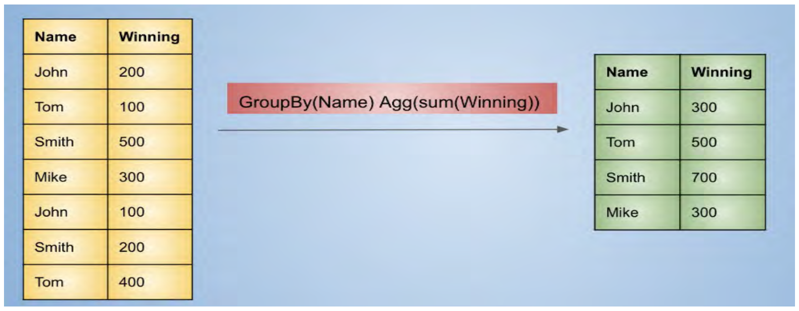
Name	Age
John	21
Tom	17
Smith	16

Name	IsActive
Monty	True
Tom	False
Smith	True
Tom	True

Name	Age	IsActive
Tom	17	False
Tom	17	True
Smith	16	True

Figure 3.11: Natural Join

- **Grouping and Aggregation:** Group rows based on some set of columns and apply some aggregation (sum, count, max, min, etc.) on some columns of the small groups that are formed. This corresponds to GROUP BY in SQL.



Name	Winning
John	200
Tom	100
Smith	500
Mike	300
John	100
Smith	200
Tom	400

Name	Winning
John	300
Tom	500
Smith	700
Mike	300

Figure 3.12: Group by Name and take sum of the Winning

ENTERPRISE BATCH PROCESSING

Enterprise applications contain tasks that can be executed without user interaction. These tasks are executed periodically or when resource usage is low, and they often process large amounts of information such as- log files, database records, or images. Examples include billing, report generation, data format conversion, and image processing.

Batch processing refers to running batch jobs on a computer system. Java EE includes a batch processing framework that provides the batch execution infrastructure common to all batch applications, enabling developers to concentrate on the business logic of their batch applications.

EXAMPLE/APPLICATIONS OF MAP REDUCE

- **Social Networks:**

Social networking users like Facebook, Twitter, and LinkedIn connect with friends and the community.

- **Entertainment:**

To discover the most popular movies Netflix uses Hadoop and Map-Reduce. This is the best solution and suggestions for registered users taking into account their interests.

Map-Reduce can determine how users are watching movies, analysing their logs and clicks.

- **Electronic Commerce:**

E-commerce providers, such as Amazon, Wal-Mart, and eBay, use the Map-Reduce programming model to identify favourite products based on users' interests or buying behaviour.

- **Fraud Detection:**

Hadoop and Map-Reduce are used in the financial industries, including companies such as banks, insurance providers, and payment locations for fraud detection, trend identification, or business metrics through transaction analysis.

- **Search and Advertisement Mechanisms:**

Advertisement mechanisms utilize it to analyze and understand search behavior, trends, and missing results for specific keywords. Google and Yahoo use Map-Reduce to understand users' behavior, such as popular searches for an event such as presidential elections.