Cassandra

Extension to Databases

Professor: Pablo Ramos

pablo.ramos@u-tad.com

INTRODUCTION

- Using columns to store information:
  - Key: Unique name that references the column
  - Value: contents of the column.
  - Timestamp : Date on which the document was inserted or updated. Used to check the validity of data in distributed systems.

Clue	Id	Name
Worth	12345	Pepe
Timestamp	2013-10-30 05:02:45.004000	2013-10-30 05:02:50.004000

INTRODUCTION

- Cassandra is a database designed to support large volumes of data distributed across hundreds of servers, guaranteeing high data availability and ensuring the integrity of the database against massive failures.

- Its main features are
  - Decentralization (high availability)
  - Replication
  - Fault tolerance
  - Scalability

ARCHITECTURE: Main elements

- Items:
  - **Node:** Cassandra
    - 1 or more on a server (Development)
    - 1 per server (Production)
  - **Partition:** Basic unit of information. Of vital importance for data replication and organization.
  - **Rack:** logical set of nodes.
  - **Data center:** logical set of racks
  - **Cluster :** Complete set of nodes. (Full token ring)

ARCHITECTURE: Nodes

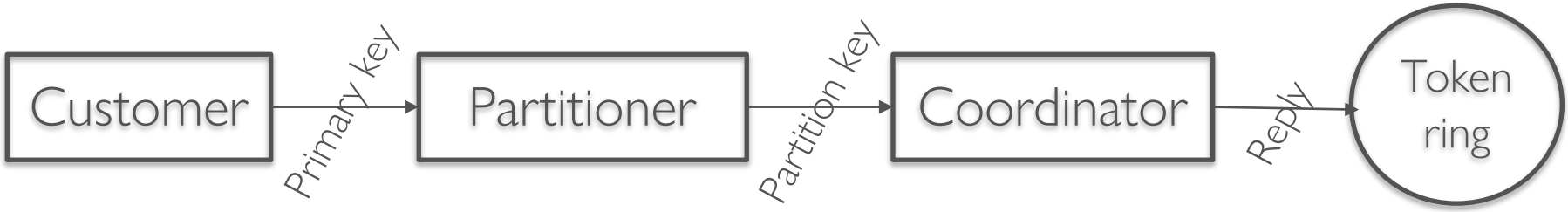
- **Node:** Cassandra instance .
  - **Seed node:** Node that a new node first contacts when joining a cluster to learn about the cluster structure and its role.
  - **Coordinator:** Each of the nodes in the cluster that receives a request from a client.
    - It is responsible for coordinating the request through the nodes and ensuring that it is carried out and the result is returned through it.
    - It is usually chosen randomly by the client based on availability.
    - If a node goes down, the next node will take over coordinating the requests.

ARCHITECTURE: Nodes

- **Node:** Cassandra instance .
  - **Seed node**
  - **Coordinator**
  - **Driver:** Connection interface for the client.
    - It is responsible for deciding which node coordinates each request.
    - -robin pattern is implemented . It directs each new request to a new node.

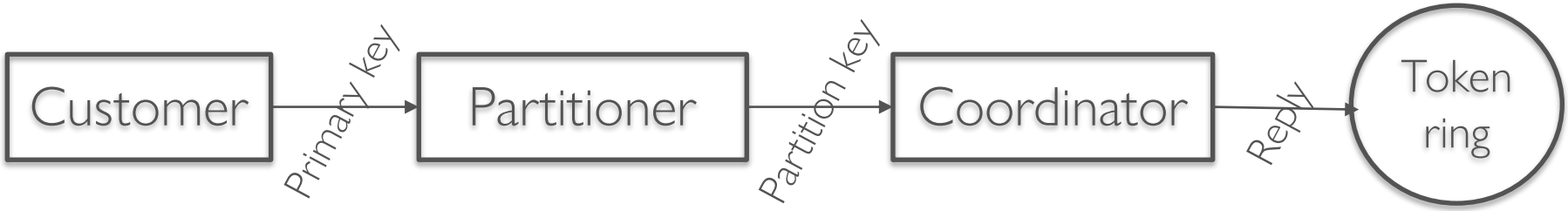
ARCHITECTURE: Partition

- Partition is the basic unit of storage.
  - **Partitioning process:** it is responsible for assigning a token (partition key) to each data unit from its primary key (primary key). This token will uniquely identify the partitions.
  - Partition keys are used to determine the location of data in the database. Each node is assigned a range of partitions.
  - **Partitioner :** responsible for carrying out the partitioning process.



ARCHITECTURE: Partition

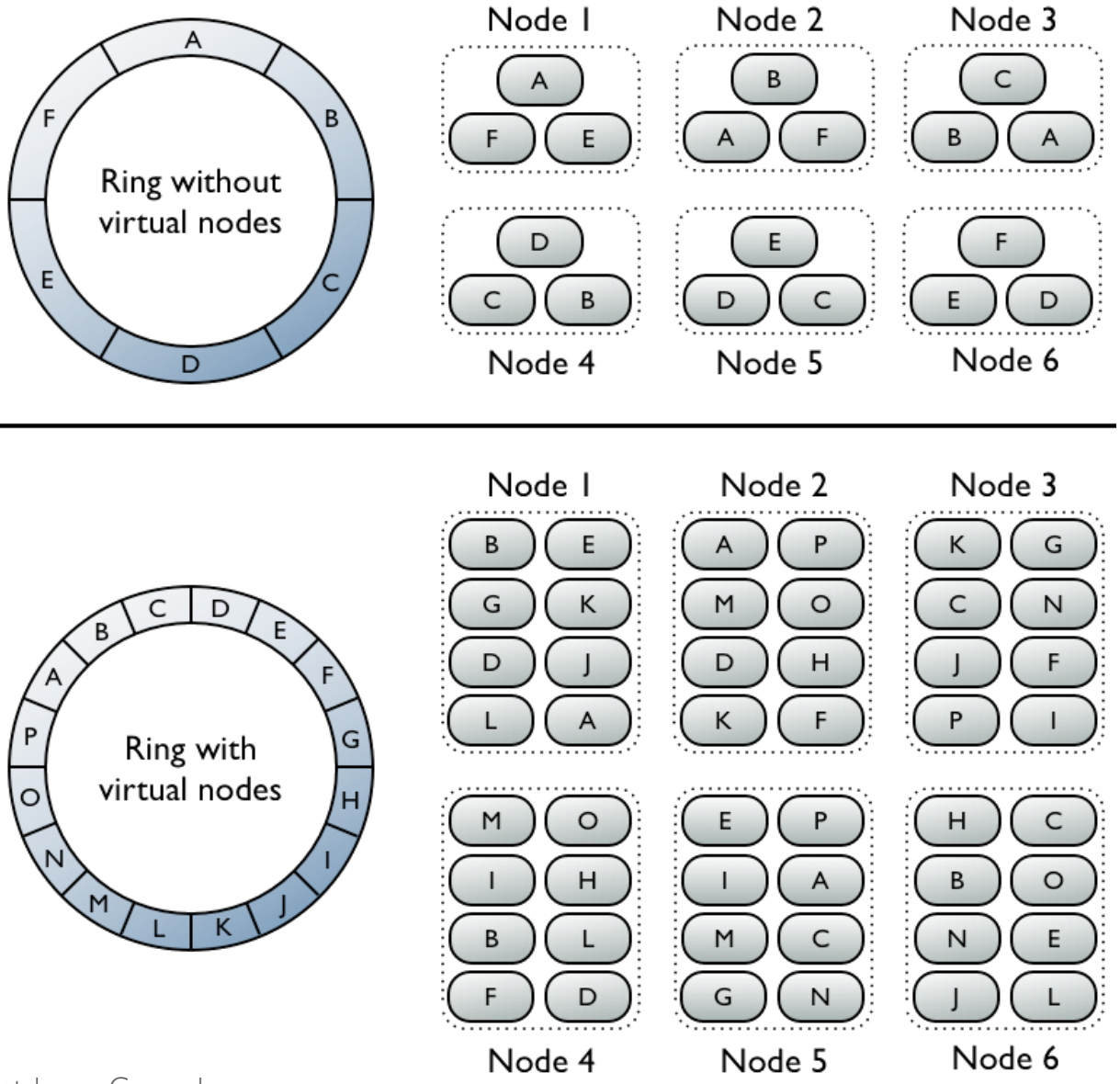
- Partition is the basic unit of storage.
  - **Token ring:** This is the set of partition keys existing in a cluster. This set is subdivided into segments and each segment is assigned to a node. Depending on the partition key generated by the partitioner , the replica of a partition will be directed by the coordinator to one node or another.



ARCHITECTURE: Virtual nodes

- Since version 2.0, each node is subdivided into virtual subnodes whose token ring segment is discontinuous.
- Virtual nodes behave like normal nodes.
- The main feature of virtual nodes is that they can be dynamically and automatically assigned their respective token ring segments.
  - Bootstrap : Process carried out when a new node is launched. Part of the token ring is automatically reassigned to the new node and its replicas.
  - Decommissioning : Process carried out when a node is removed. Its share of the token ring is automatically reassigned to the remaining nodes and their replicas.

ARCHITECTURE: Virtual nodes



https://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archDataDistributeDistribute.html

ARCHITECTURE: Replication

- Replication enables Cassandra to provide high availability and fault tolerance. The basic unit of replication is the partition. There are no originals and/or copies of a partition, they are all replicas .
  - Each partition has one or more replicas depending on the replication settings specified in its keyspace .
  - **Replication factor:** Determines how many nodes should perform a write request. At least one piece of data must be on one node and at most on all nodes in the cluster.
  - **Replication Strategy:** Determines how replicas are distributed. If there are multiple racks in a data center, replicas are distributed evenly across the racks.
    - SimpleStrategy : Replicas are distributed sequentially.
    - NetworkTopologyStrategy : Each data center has its own replication factor. Each data center will have its remote coordinator to perform the replication.

ARCHITECTURE: Level of consistency

- **consistency** level determines how many nodes must respond to a request before the coordinator returns the response to the client.
  - **Writing:** How many have stored the partition.
  - **Reading:** How many have returned the partition.
  - Different levels of consistency can be defined depending on how many replica nodes must send " acknowledged "

ONE, TWO, THREE	The first node , the first two …
ANY	Any node .
ALL	All nodes .
QUORUM	At least 51% of the
LOCAL_ONE	The first node of the data center
LOCAL_QUORUM	Quorum in the data center
EACH_QUORUM	Quorum of each data center

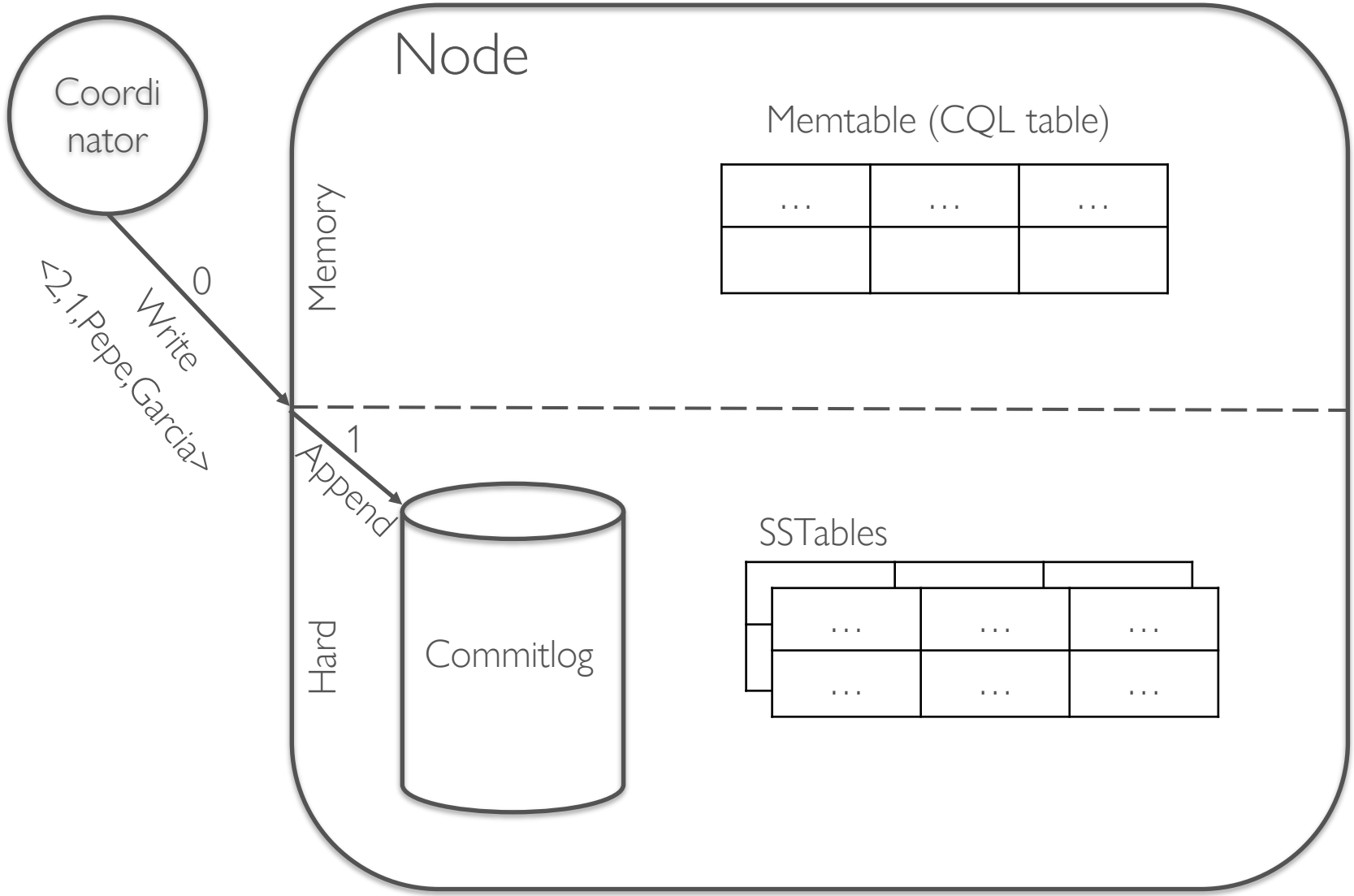
ARCHITECTURE: Hinted Handoff

- **Records crashes and failures** on **nodes.**
  - Allows you to manage write operations on nodes that are down.
    - In case a node is down, the **coordinator** saves a " **hint** " that specifies the replica to write to the down node.
    - When the node becomes available again, the coordinator performs the write operation again.
    - If a node is down for more than three hours, it is understood that there is a major problem and no " hint " is saved.

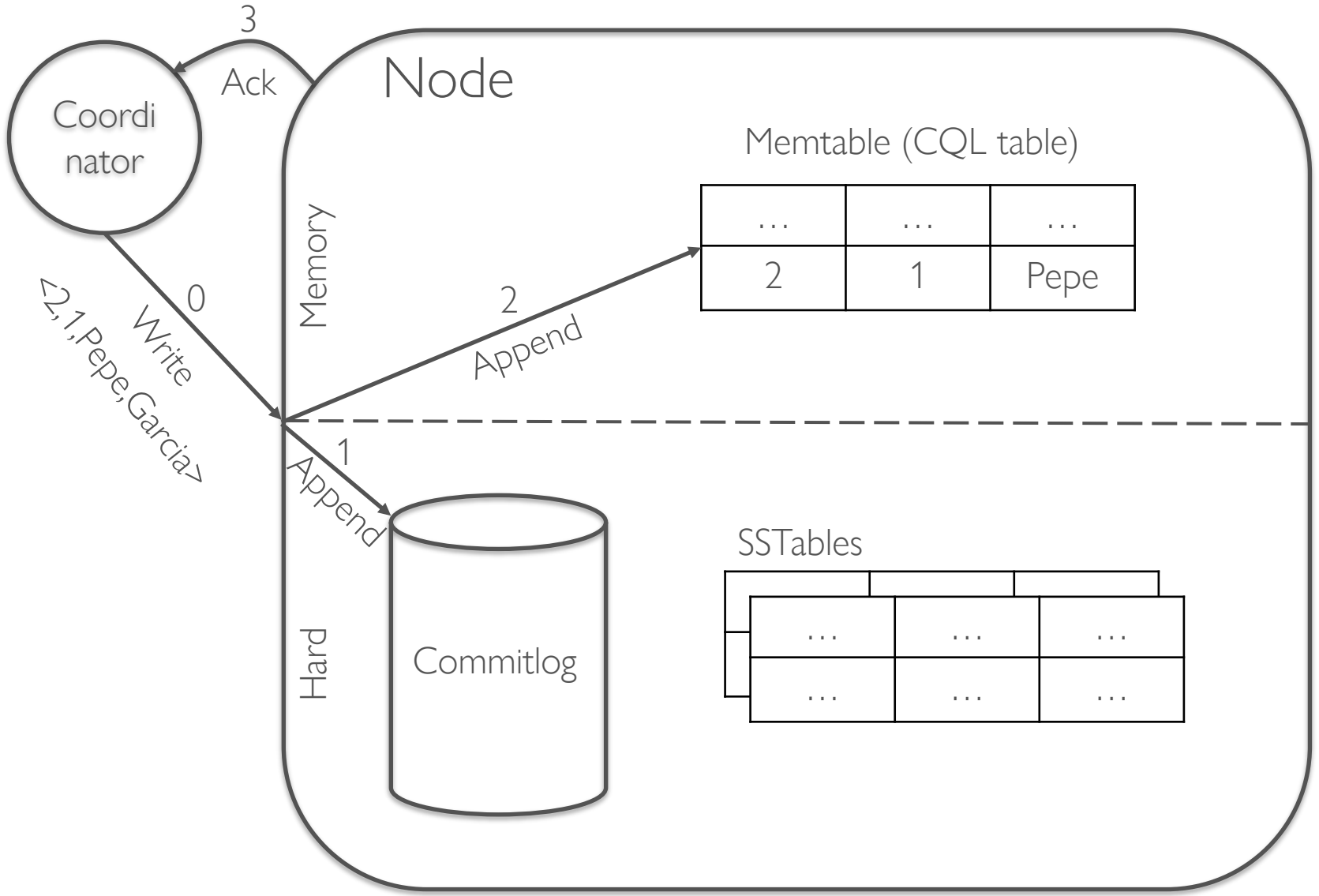
ARCHITECTURE: Write path flow

- The data writing process in Cassandra (insert , update , delete) is composed of four stages that ensure the durability and consistency of the data as well as the efficiency of the queries.
  - Writing to Memtables (CQL tables)
  - Writing in the Commitlog
  - Writing to SSTables (String Sorted Tables)
  - Compacting data in SSTables

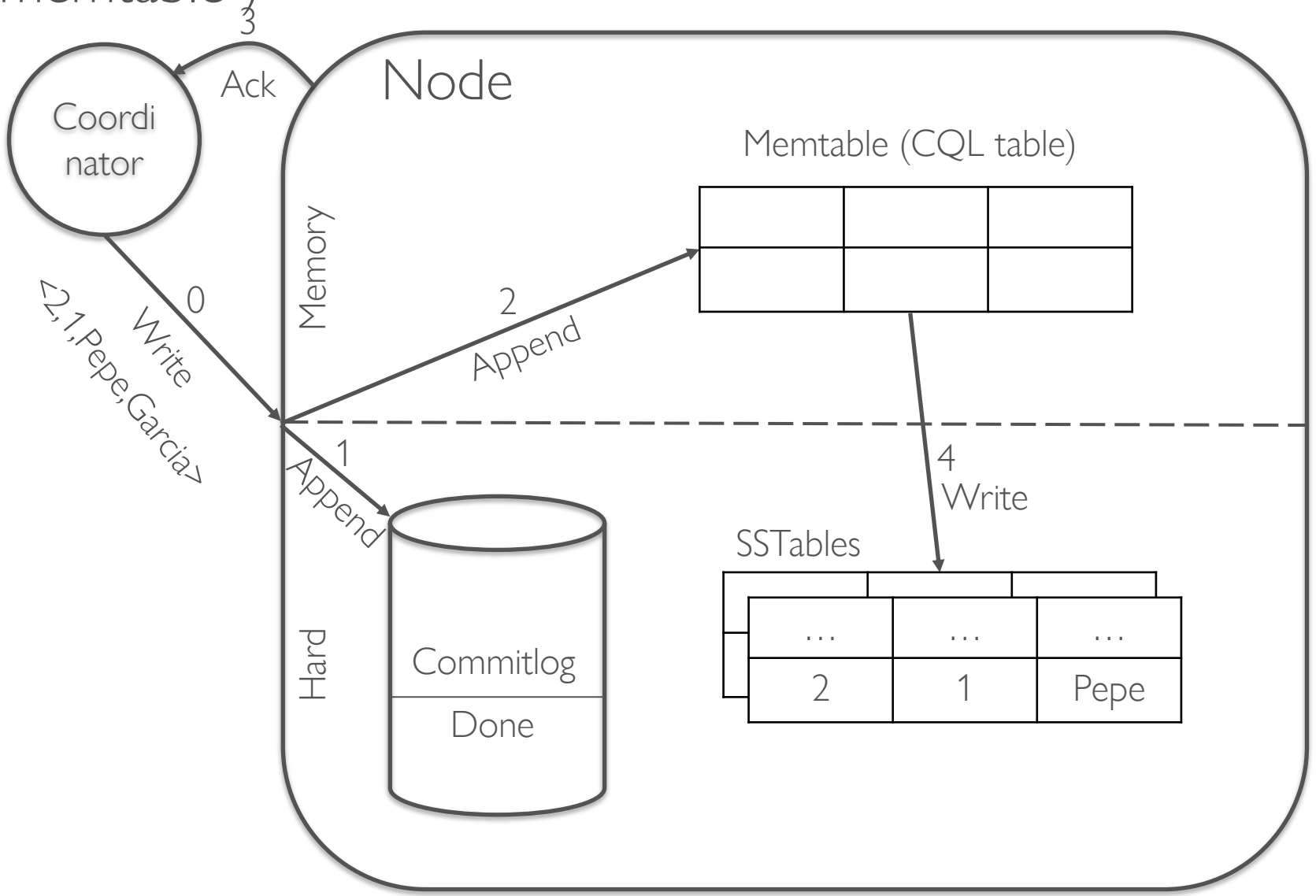
ARCHITECTURE: Write path flow



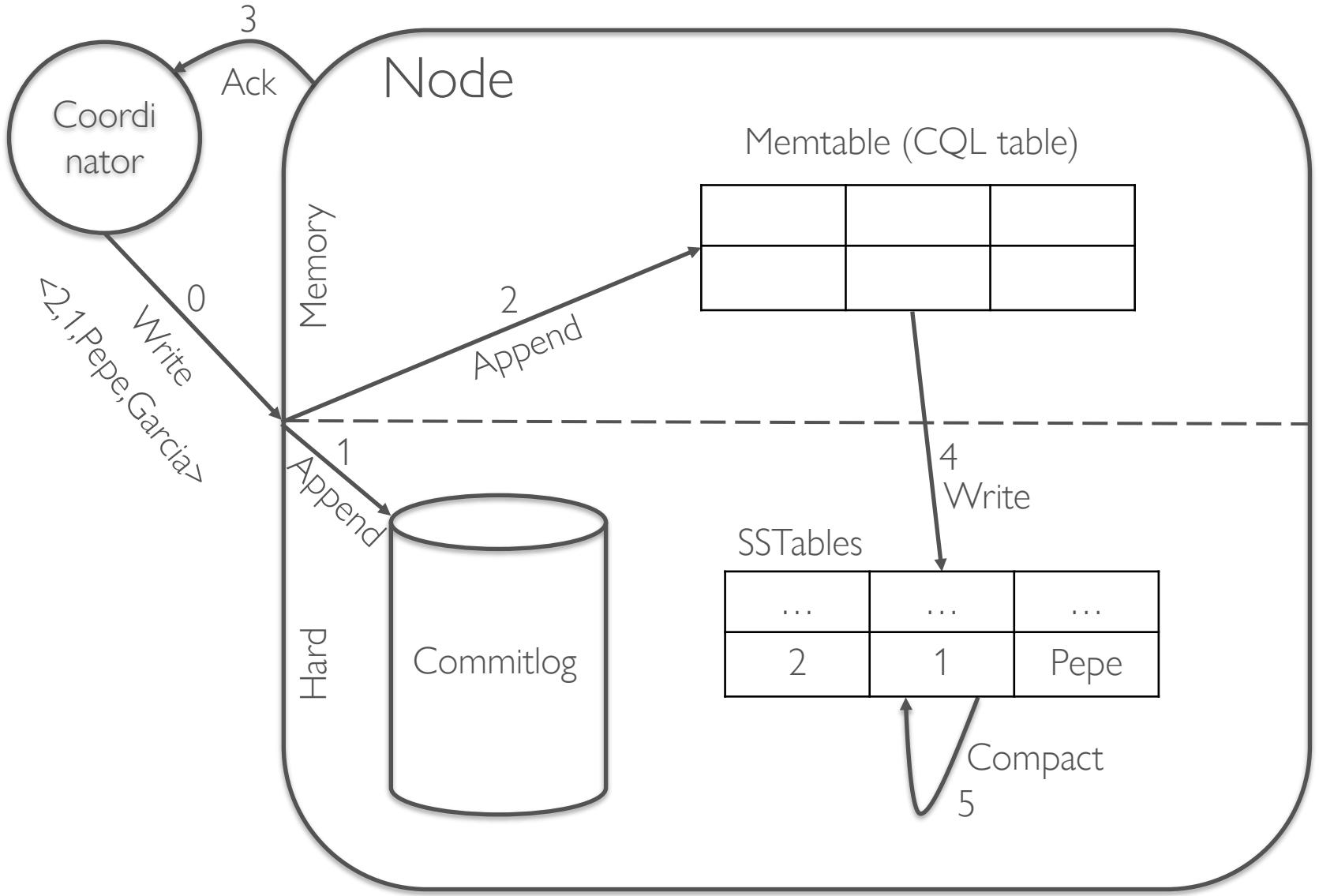
ARCHITECTURE: Write path flow



ARCHITECTURE: Write path flow (Flush of the memtable)



ARCHITECTURE: Write path flow (Compaction)



ARCHITECTURE: Write path flow

- Commitlog
  - It introduces each new write query to the hard disk so that if there is any server failure, it is able to update the Memtables with the pending queries to be executed.
  - Writing queries to disk will be performed when the queries recorded in the commitlog without writing to SSTables (disk) reach a certain size.
  - When queries are written to SSTables they are marked flushed .
  - In reality, queries are not written directly to the hard disk. Until they are written to the hard disk, the Acknowledge is not returned to the coordinator.

ARCHITECTURE: Write path flow

- Memorable
  - Memtable is created for each CQL table in each KeySpace .
  - In the Memtables, requests for writing to databases are accumulated. Therefore, it is possible that we have several entries for the same data.
  - Read queries are also allowed for data that has not yet been written to the hard disk.
  - Periodically all the contents of the Memtables are moved to new SSTables on hard disk and their corresponding Commitlog entries are marked as flushed .

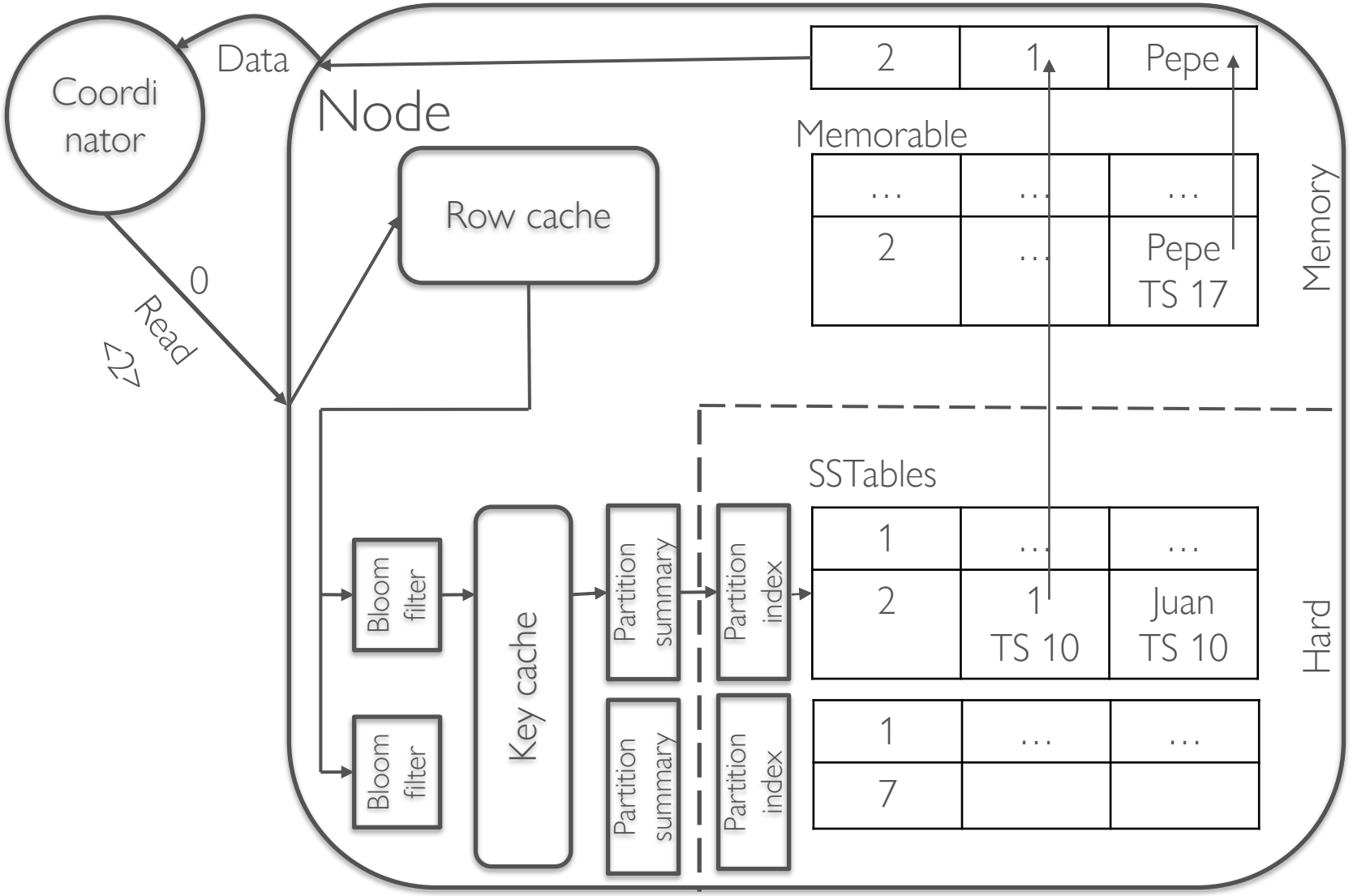
ARCHITECTURE: Write path flow

- SSTable
  - SSTables are immutable . Information simply accumulates in the tables.
  - SSTable is created for each Memtable each time the flush process is performed . Therefore, there will be multiple SSTables per CQL Table .
  - Periodically, all the information contained in the SSTables is compacted so that there is only one SSTable per CQL table . From this point on, there will be only one entry updated for the same data in the SSTable .

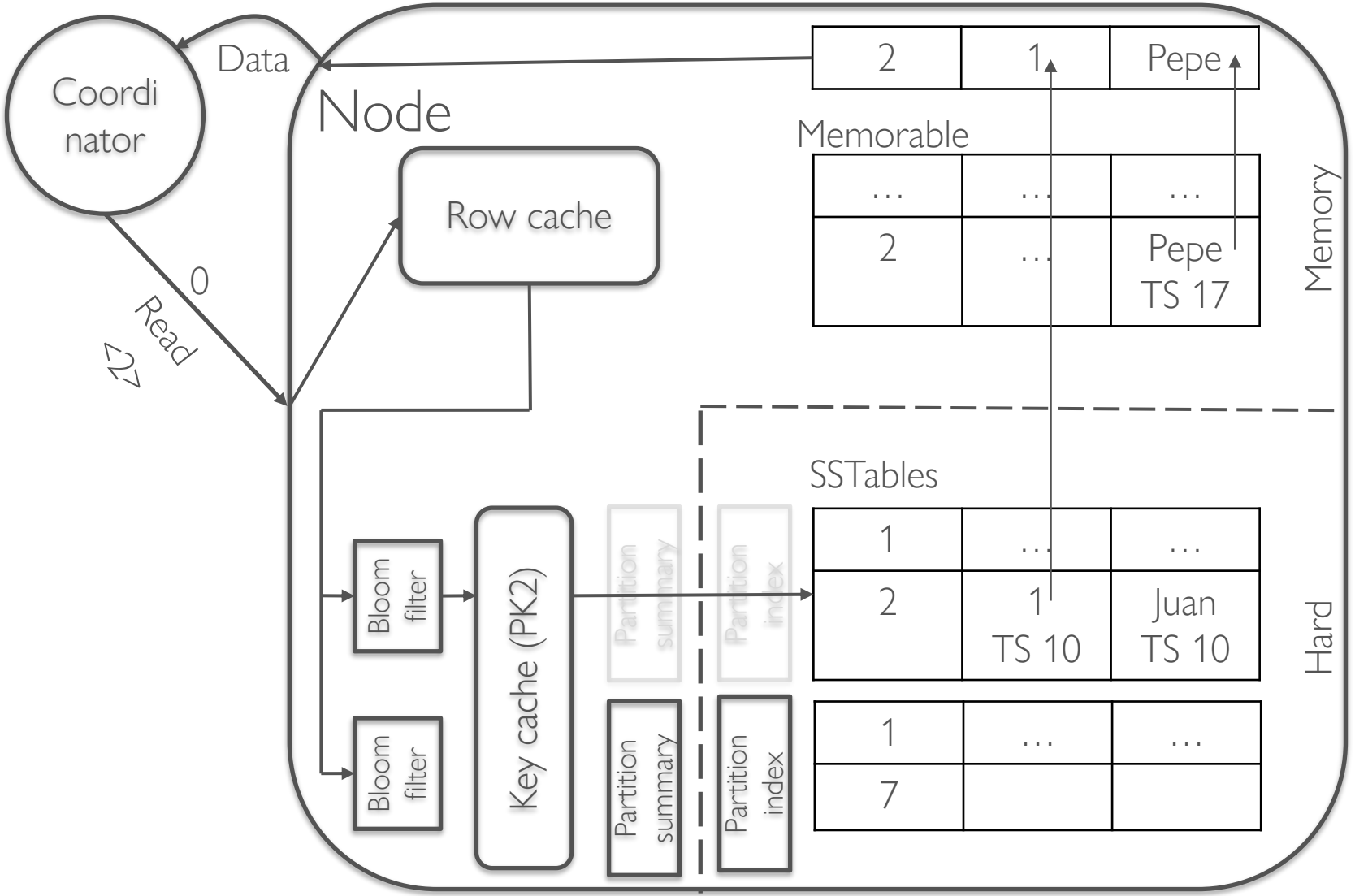
ARCHITECTURE: Read path flow

- The reading process in Cassandra searches both the **Memtables** and the **SSTables** for entries for the partition key specified.
- Once these entries have been located, the set of partition fields is returned to the coordinator through a process called " merge ". key with the most recent timestamp .
- There are filters and intermediate caches that allow the query process to be accelerated.
- **Read_repair_chance** (10%): Probability with which a data consistency

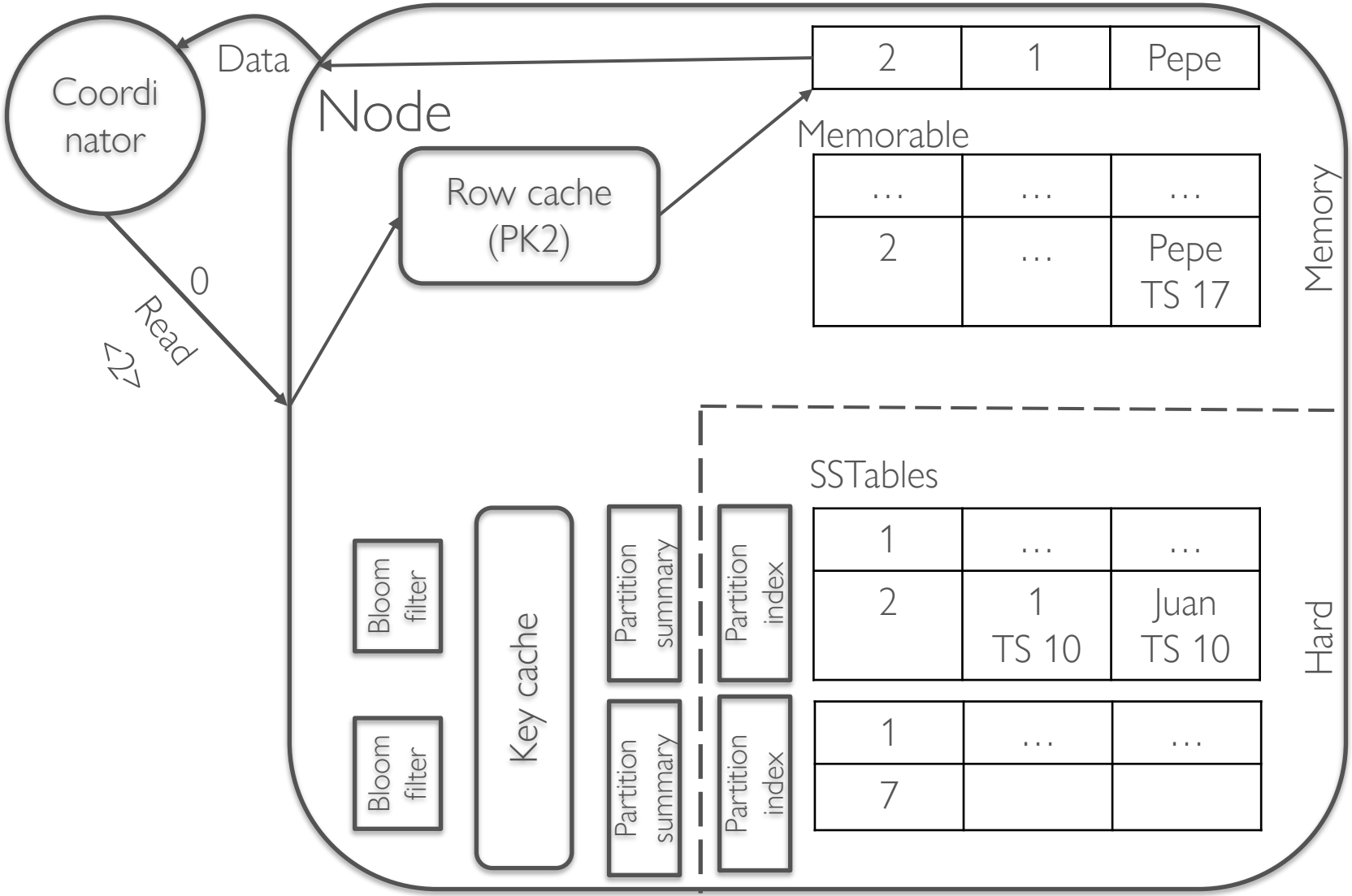
ARCHITECTURE: Read path flow (No cache)



ARCHITECTURE: Read path flow (Key cache)



ARCHITECTURE: Read path flow (Row cache)



ARCHITECTURE: Read path flow

- Row cache:
  - It allows you to store the results of the last queries performed so that if that data is consulted again, it directly returns the data without having to perform a merge .
  - Row cache is optional. By default it is disabled.
  - The contents of the cache are periodically saved to disk so that if the node is rebooted the cache can be quickly recovered.

ARCHITECTURE: Read path flow

- Bloom filter :
  - It is a probabilistic data structure that indicates whether a primary key is or is not in its associated SSTable .
  - Allows you to reduce search costs.
  - Boom filter will test positive if it thinks the primary key is in your SSTable and negative if you know your SSTable does not contain the primary key .
    - There are false positives.
    - There are no false negatives.
  - The percentage of false positives can be regulated. The lower the percentage of false positives, the higher the memory usage.

ARCHITECTURE: Read path flow

- Key cache:
  - It allows storing the positions of the last queries performed so that if that data is consulted again it is not necessary to consult the partition tables to find its position in the SSTables .
  - The contents of the cache are periodically saved to disk so that if the node is rebooted the cache can be quickly recovered.

ARCHITECTURE: Read path flow

- Partition summary :
  - It is an index of the partition index that allows direct access to the segment where the position is located in the partition 's SSTable key searched.
  - While the partition index is located on hard disk, the partition summary is located in memory so that the performance of the query process is improved.
- Partition index
  - Contains the complete set of positions for each partition key in its associated SSTable .