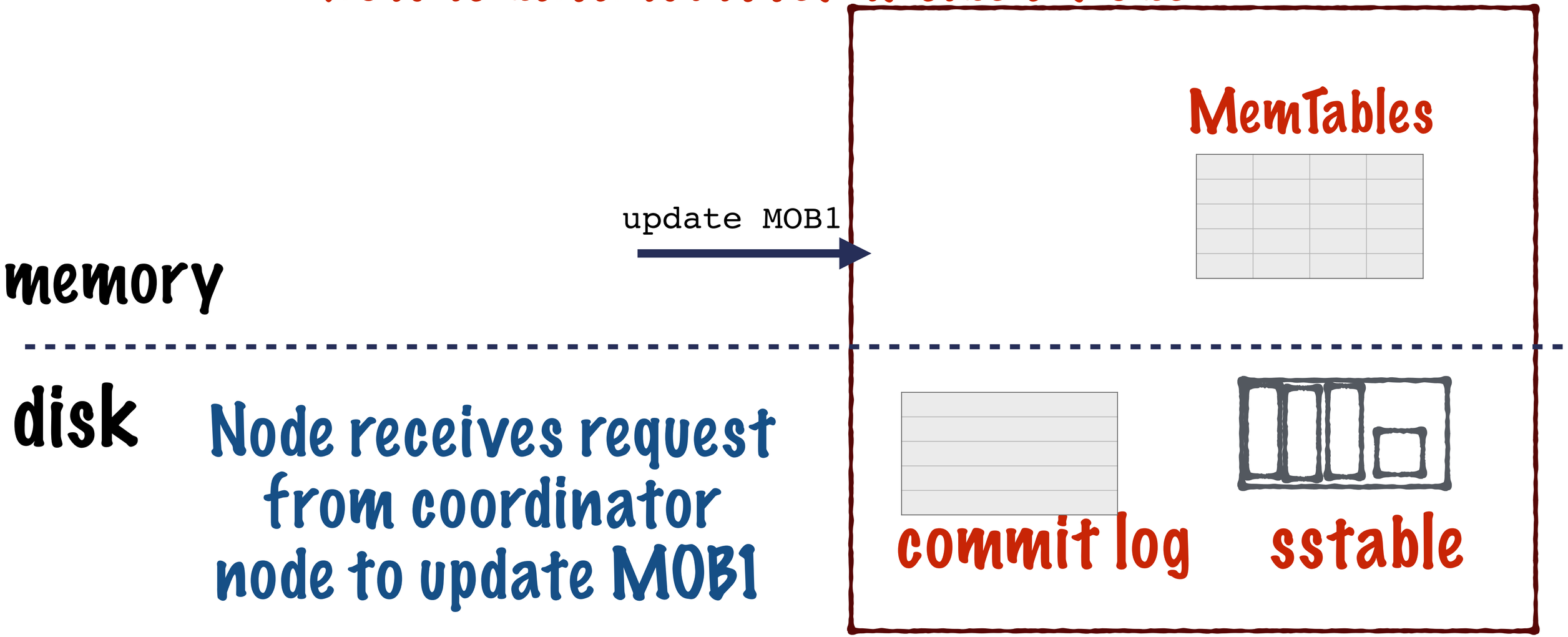
Storage components in Cassandra

Node

Let's look at a brief overview of how is data written inside a node



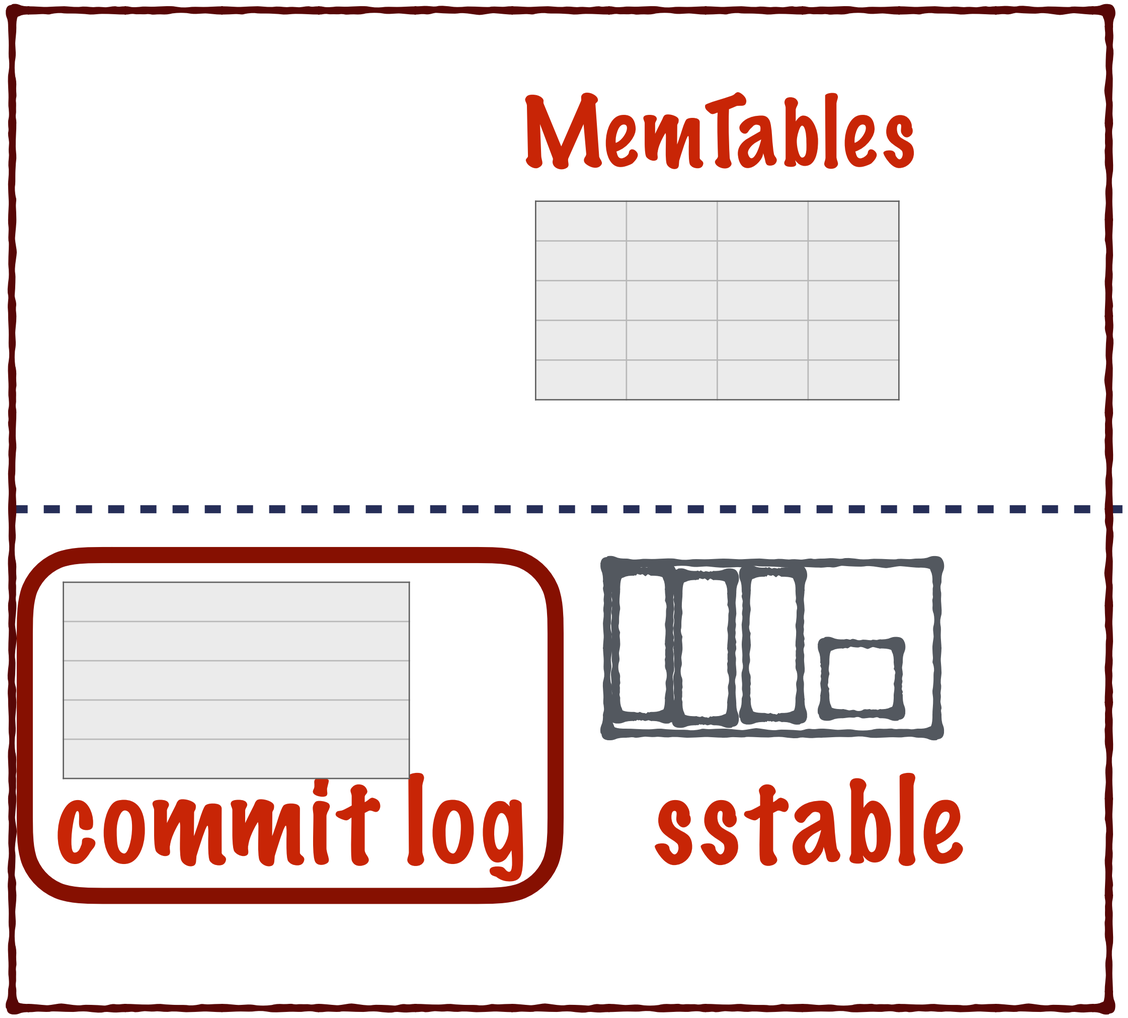
Node receives request from coordinator node to update MOB1

Node

Data is first updated in the commit log

memory

disk

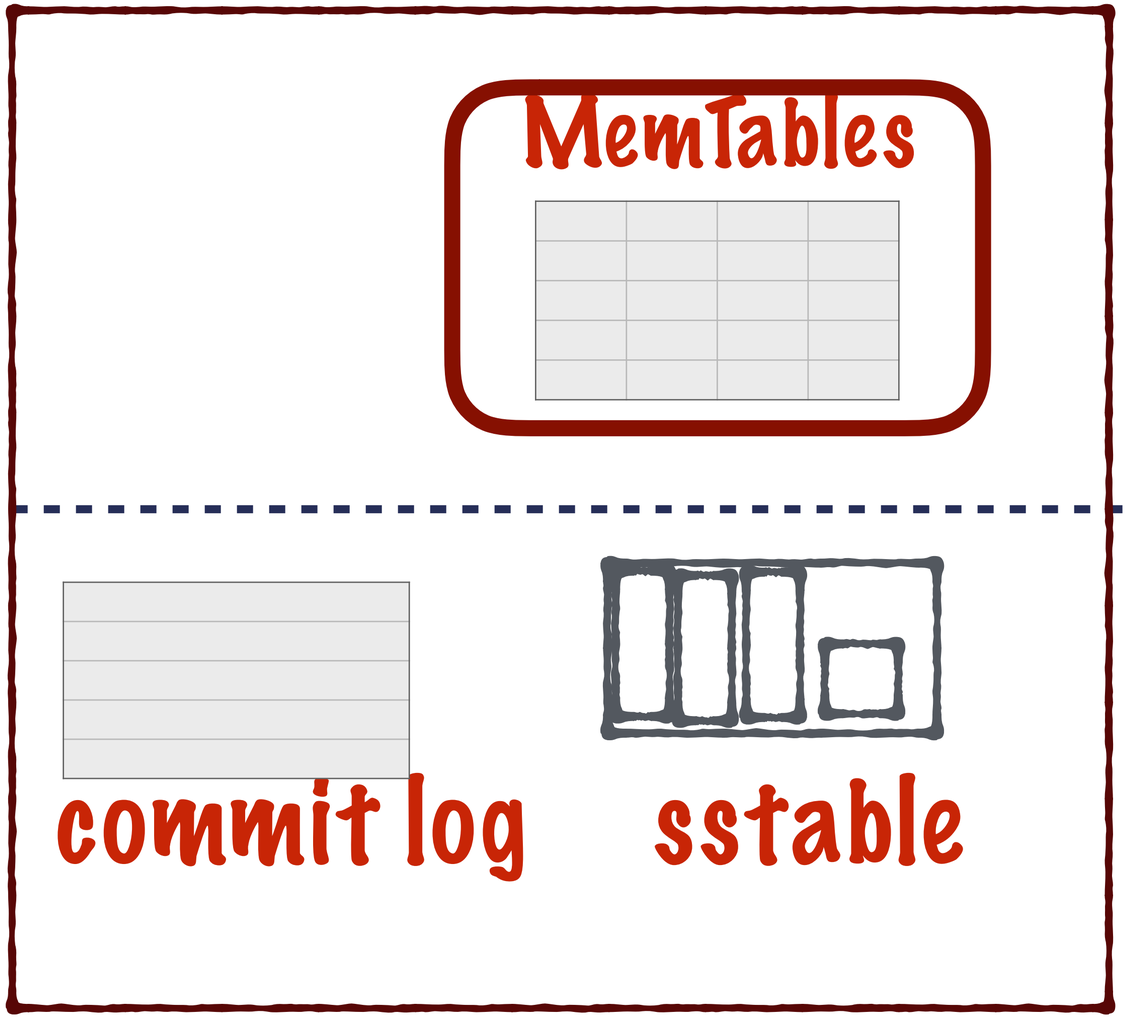


Node

Data is then updated in the memtable

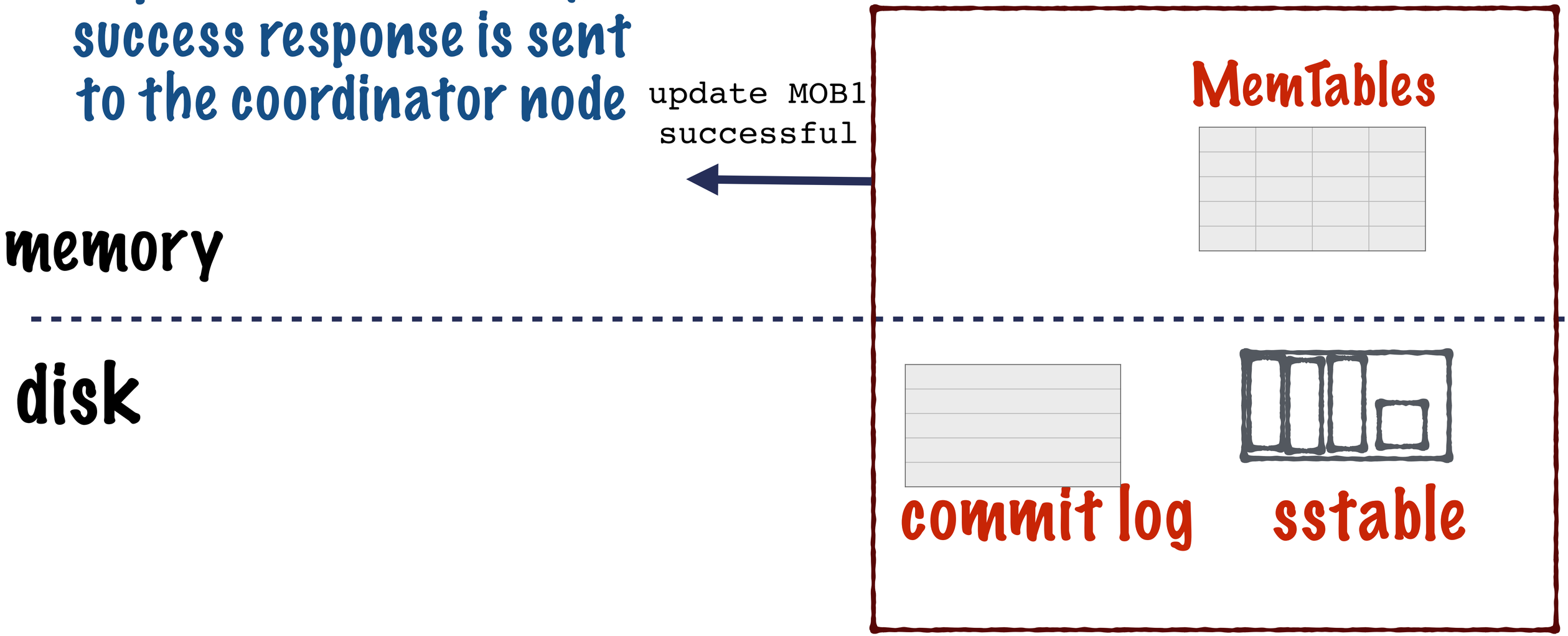
memory

disk



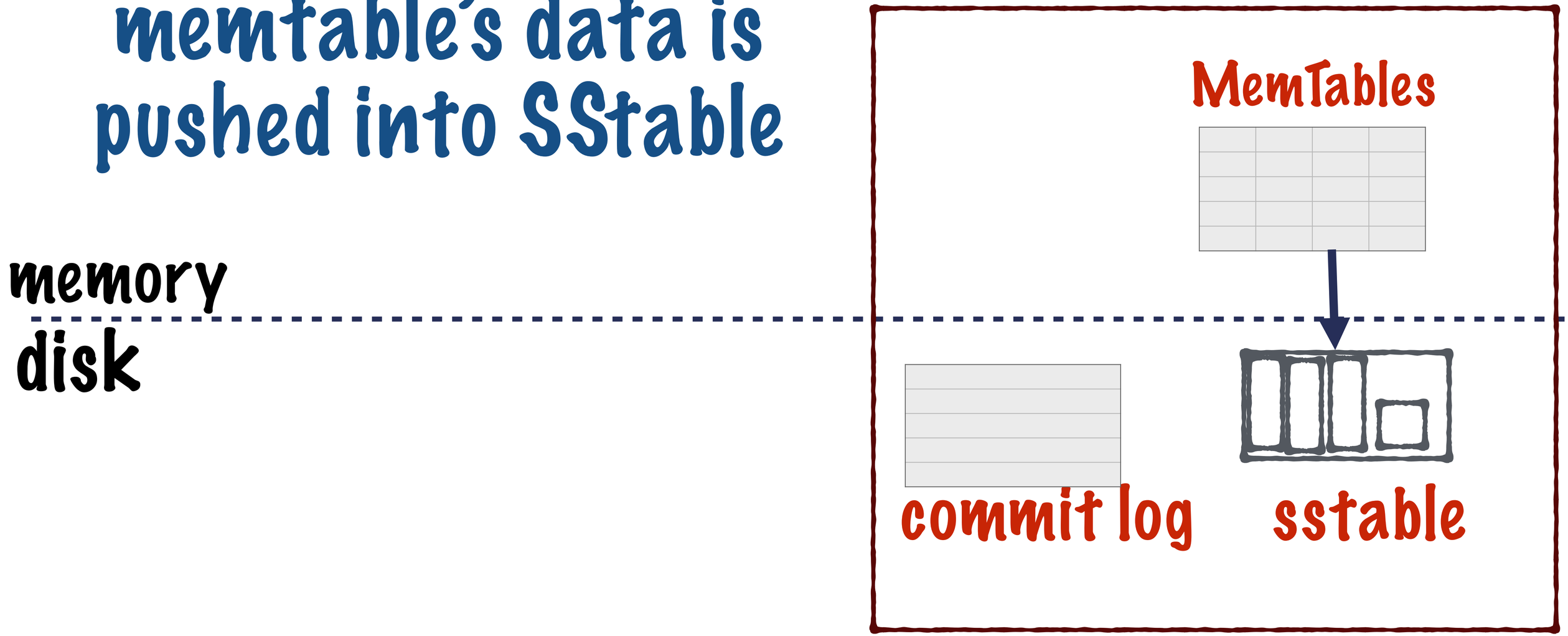
Node

after memtable is updated successfully, success response is sent to the coordinator node



Node

At some point memtable's data is pushed into SStable



Node

Now lets understand all the storage components

Memtables

in-memory hash tables

A CF can have multiple Memtables

in-memory hash tables

Memtables

A CF can have multiple Memtables

Data in memtable

{k1: [{c1, v1 },{c2, v2}, {c3, v3}]}

in-memory hash tables

Memtables

A CF can have multiple Memtables

{k1: [{c1, v1},{c2, v2}, {c3, v3}]}

A row key

in-memory hash tables

Memtables

A CF can have multiple Memtables

{k1: [{c1, v1}, {c2, v2}, {c3, v3}]}

A list of column-value pairs

Commit Log

It is a memory mapped log file on disk

A segment of virtual memory that has a file resource present on disk

Data is synced to disk every 10s

This makes writes very fast

Commit Log

This is also used for recovering data after a node crash

Data is first written to the commit log

If node crashes after data is written

memtable can be re-built by replaying the commit log

SSTable

Sorted String Table

Index File

Summary File

Data File

Bloom Filter

SSTable

Index File

This holds the location where a row is located in the data file

partition key1	location of row in dataFile
partition key23	location of row in dataFile
partition key12	location of row in dataFile
partition key125	location of row in dataFile
…	…

SSTable

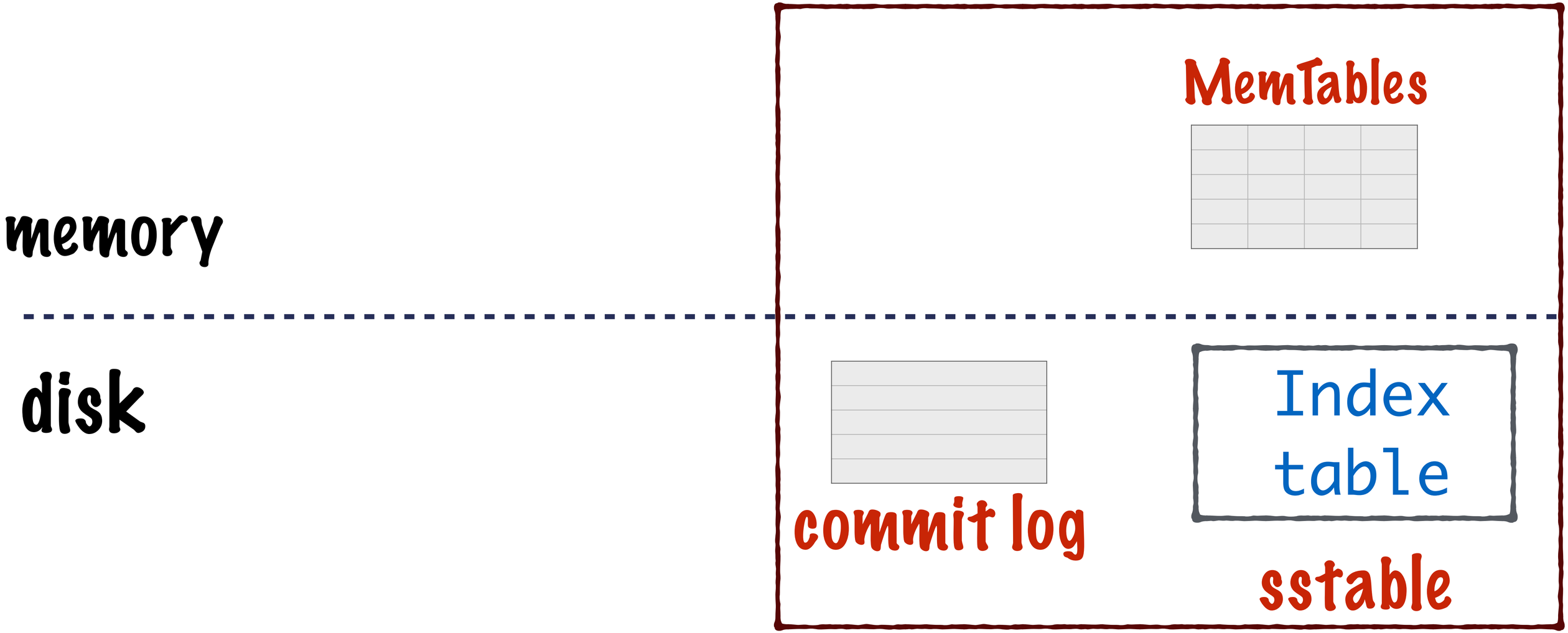
Index File

partition key1	location of row in dataFile
partition key23	location of row in dataFile
partition key12	location of row in dataFile
partition key125	location of row in dataFile
…	…

We have 1 index table per SSTable

Entries are sorted by token (partition key)

Node



SSTable

Sorted String Table

Index File

Summary File

Data File

Bloom Filter

SSTable Summary File

This is the index file

partition key1	location of row in dataFile
partition key23	location of row in dataFile
partition key12	location of row in dataFile
partition key125	location of row in dataFile
…	…

Sample keys in this at a certain interval

This interval is a configuration parameter

SSTable Summary File

This is the index file

partition key1	location of row in dataFile
partition key23	location of row in dataFile
partition key12	location of row in dataFile
partition key125	location of row in dataFile
…	…

This is the summary file

start:partition key1 end: partiionkey125	index location
start:partition key128 end: partiionkey256	index location
start:partition key256 end:partitionkey512	index location
..	…
…	…

SSTable
Summary File

start:partition key1 end: partiionkey125	index location
start:partition key128 end: partiionkey256	index location
start:partition key256 end:partitionkey512	index location
..	…
…	…

Sampled range of keys from the index file

SSTable
Summary File

start:partition key1 end: partiionkey125	index location
start:partition key128 end: partiionkey256	index location
start:partition key256 end:partitionkey512	index location
..	…
…	…

The corresponding location in the index file for each range

SSTable
Summary File

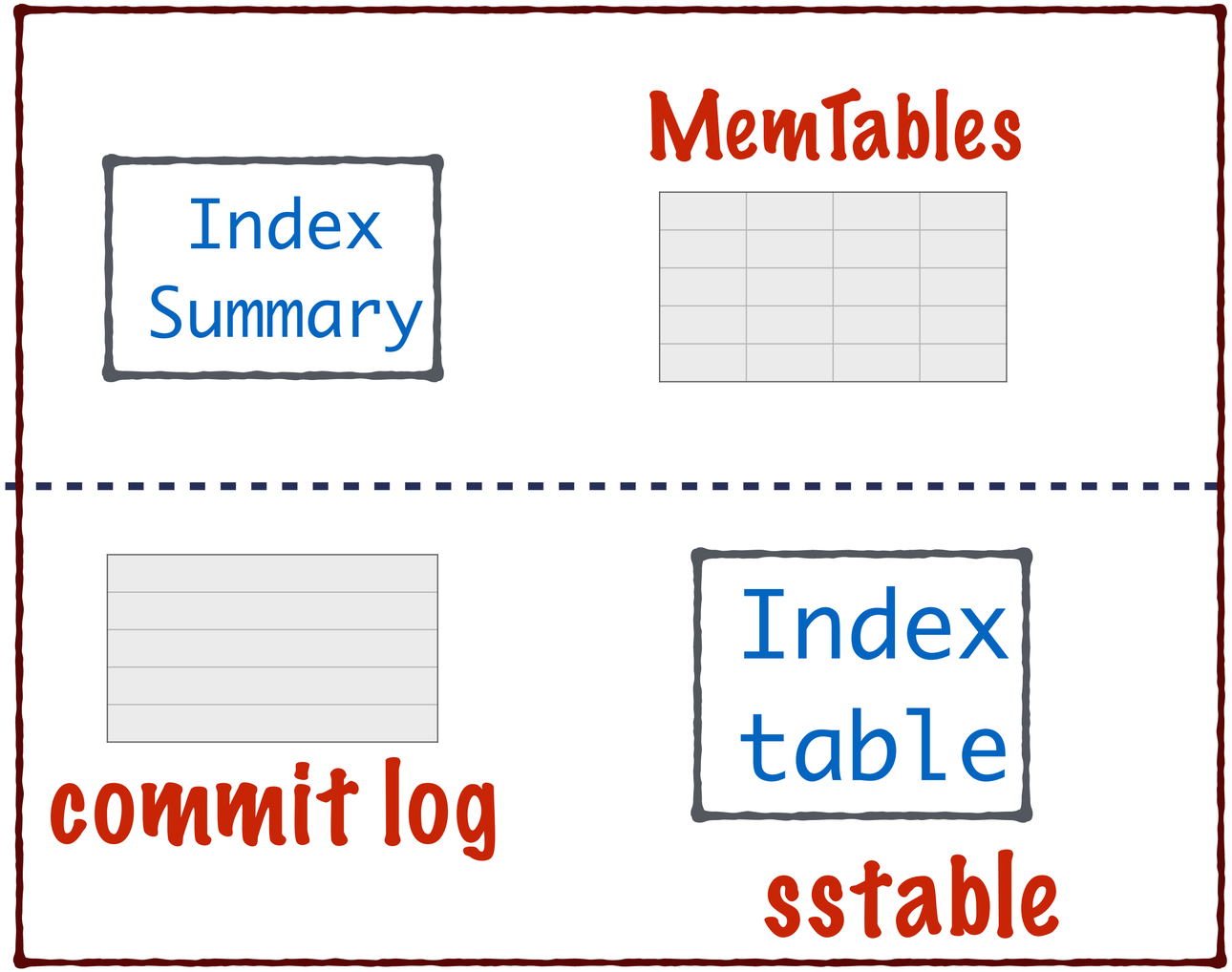
start:partition key1 end: partiionkey125	index location
start:partition key128 end: partiionkey256	index location
start:partition key256 end:partitionkey512	index location
..	…
…	…

The sampling interval is defined by the index_interval property of a columnfamily

Node

memory

disk



SSTable

Sorted String Table

Index File

Summary File

Data File

Bloom Filter

SSTable
BloomFilter

A bit vector used to determine whether a key is present on disk

A value of 1 at a position indicates that a key is present 0 means the key is not present

SSTable
BloomFilter

A bit vector used to determine
whether a key is present on disk

it is a probabilistic data structure
which can have false positives

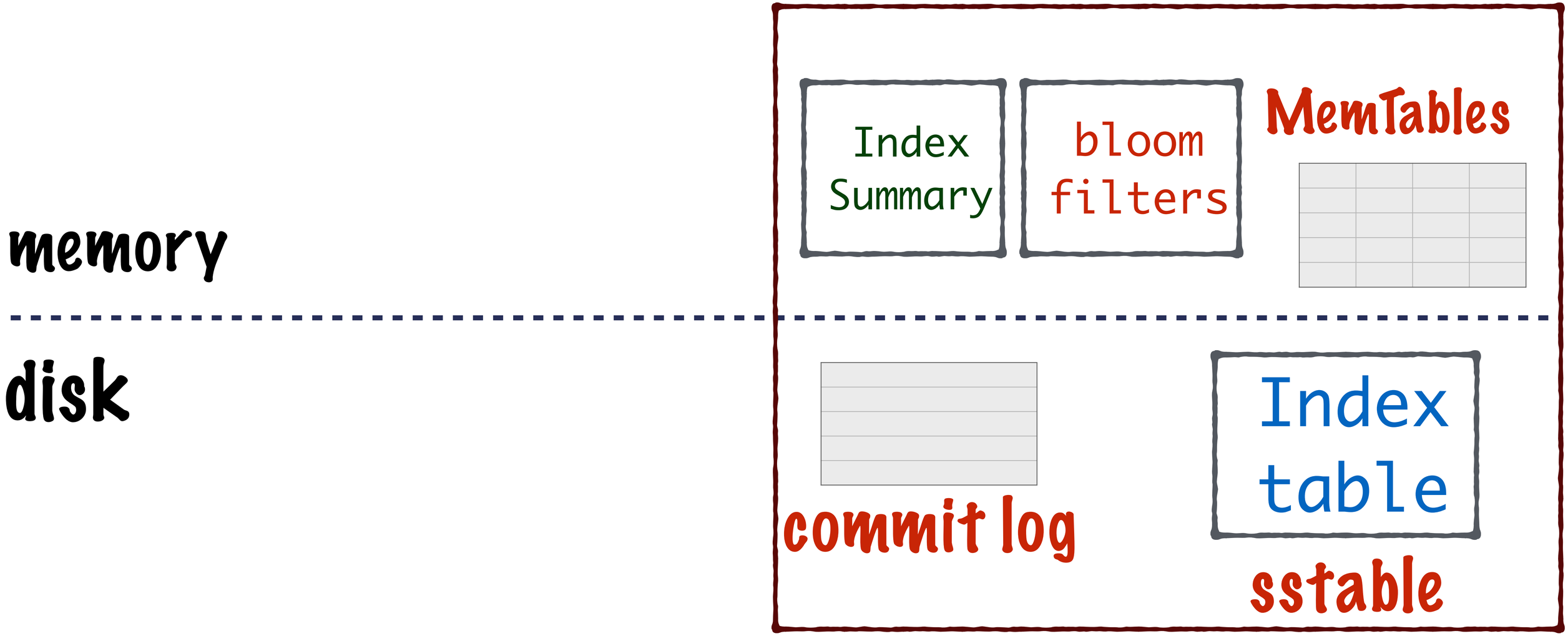
For any key the bloom filter can indicate
that it is present - when it is not!

SSTable
BloomFilter

The probability of false positives is configurable for every column family

Higher this probability - less Cassandra will rely on the filter

Node


memory
disk
Index Summary
bloom filters
MemTables
commit log
Index table
sstable

SSTable

Sorted String Table

Index File

Summary File

Data File

Bloom Filter

SSTable Datafile

Index

partition key1	location of row in dataFile
partition key23	location of row in dataFile
partition key12	location of row in dataFile
partition key125	location of row in dataFile
…	…

Data file

partitionkey1
local deletion time
markedForDeleteAt
sotred list of columns
partition key 23
local deletion time
markedForDeleteAt
sotred list of columns

The data file is where the actual data values are stored

SSTable
Datafile

Data file

partitionkey1
local deletion time
markedForDeleteAt
sotred list of columns
partition key 23
local deletion time
markedForDeleteAt
sotred list of columns

column name
column value
column timestamp

column name
column value
column timestamp

column name
column value
column timestamp

The columns have the actual data in the form of column-value pairs

SSTable
Datafile

Data file

partitionkey1
local deletion time
markedForDeleteAt
sotred list of columns
partition key 23
local deletion time
markedForDeleteAt
sotred list of columns

column name
column value
column timestamp

column name
column value
column timestamp

column name
column value
column timestamp

All columns are stored in the data file

SSTable
Datafile

Reading from a DataFile has 2 steps

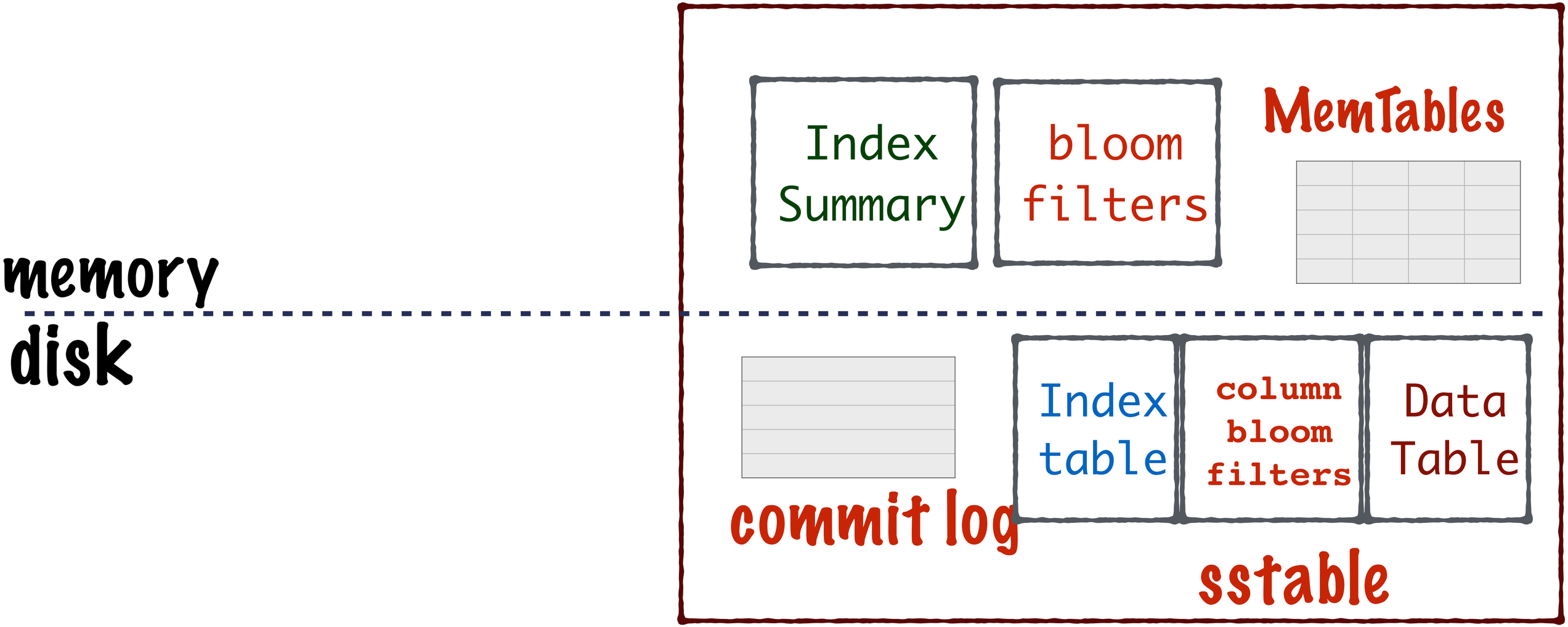
1. A disk seek to access the right row

2. Sequential read to access the columns in that row

SSTable
Datafile

Each row has a bloom filter associated with it which indicates whether a column is present or not

Node


memory
disk
Index Summary
bloom filters
MemTables
commit log
Index table
column bloom filters
Data Table
sstable

SSTable

Sorted String Table

Index File

Summary File

Data File

Bloom Filter

There are some more storage structures which are involved in Cassandra

Rowcache and Keycache

Rowcache

It contains row data based on the property 'rows_per_partition'

ALL, NONE or N

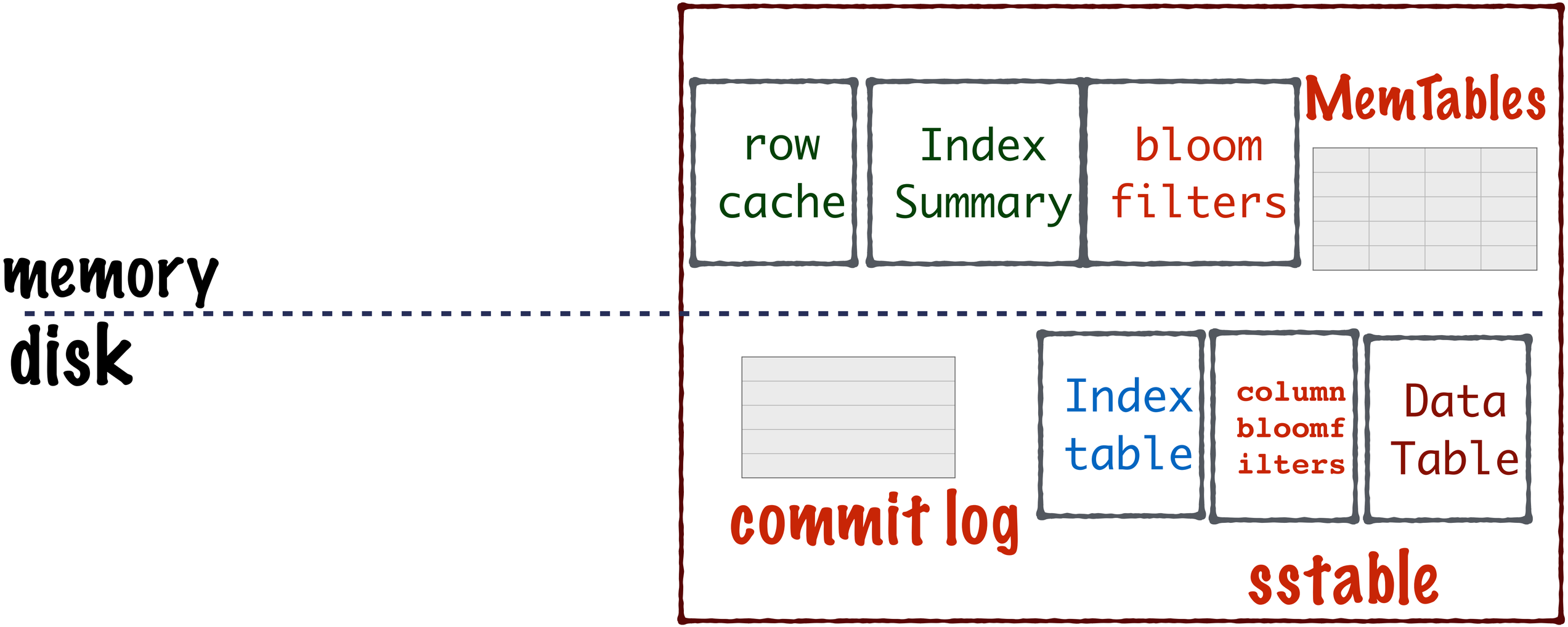
All rows are cached, no rows are cached or N rows are cache

Rowcache

Row cache is a read through cache

If value is not present in cache, then in the next read operation, it will be set in cache with the latest data

Node


memory
disk
row cache
Index Summary
bloom filters
MemTables
commit log
Index table
column bloomf ilters
Data Table
sstable

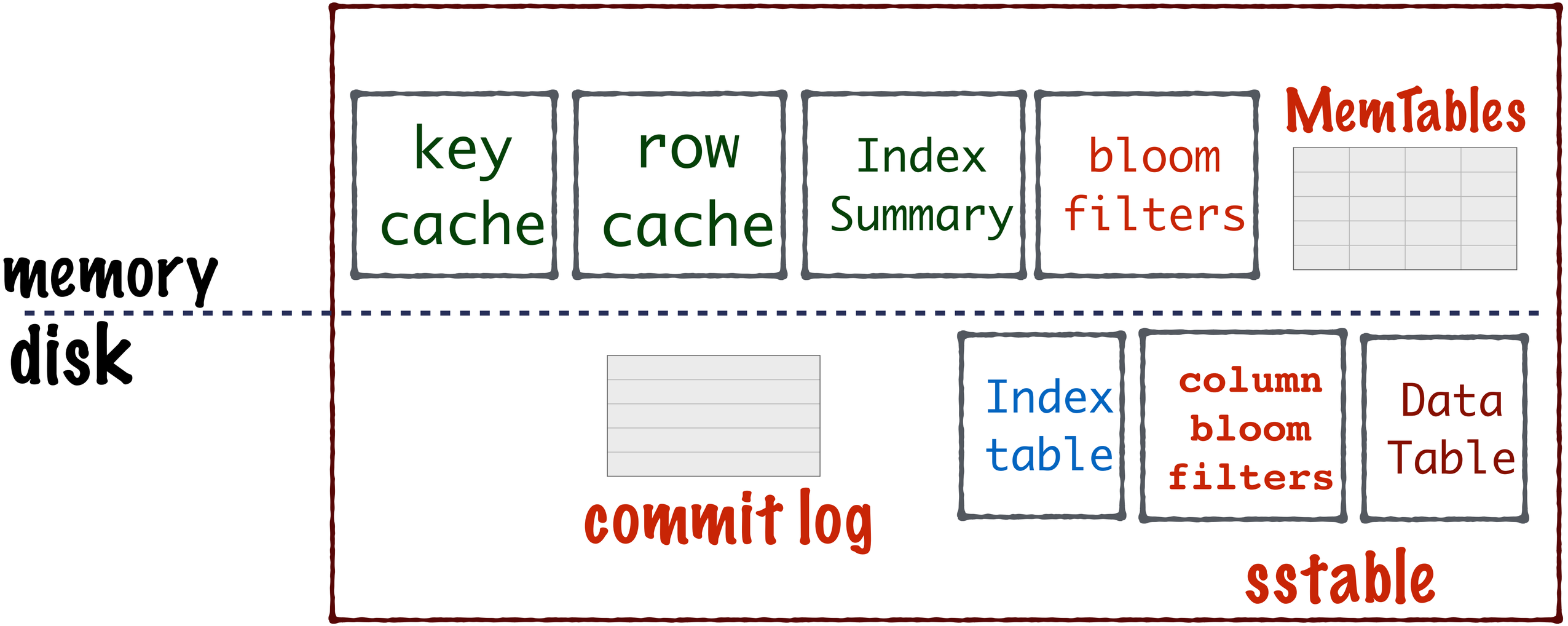
KeyCache

Its a key/value cache

Key is the primary key

value is the location offset in the SSTable where the row is located

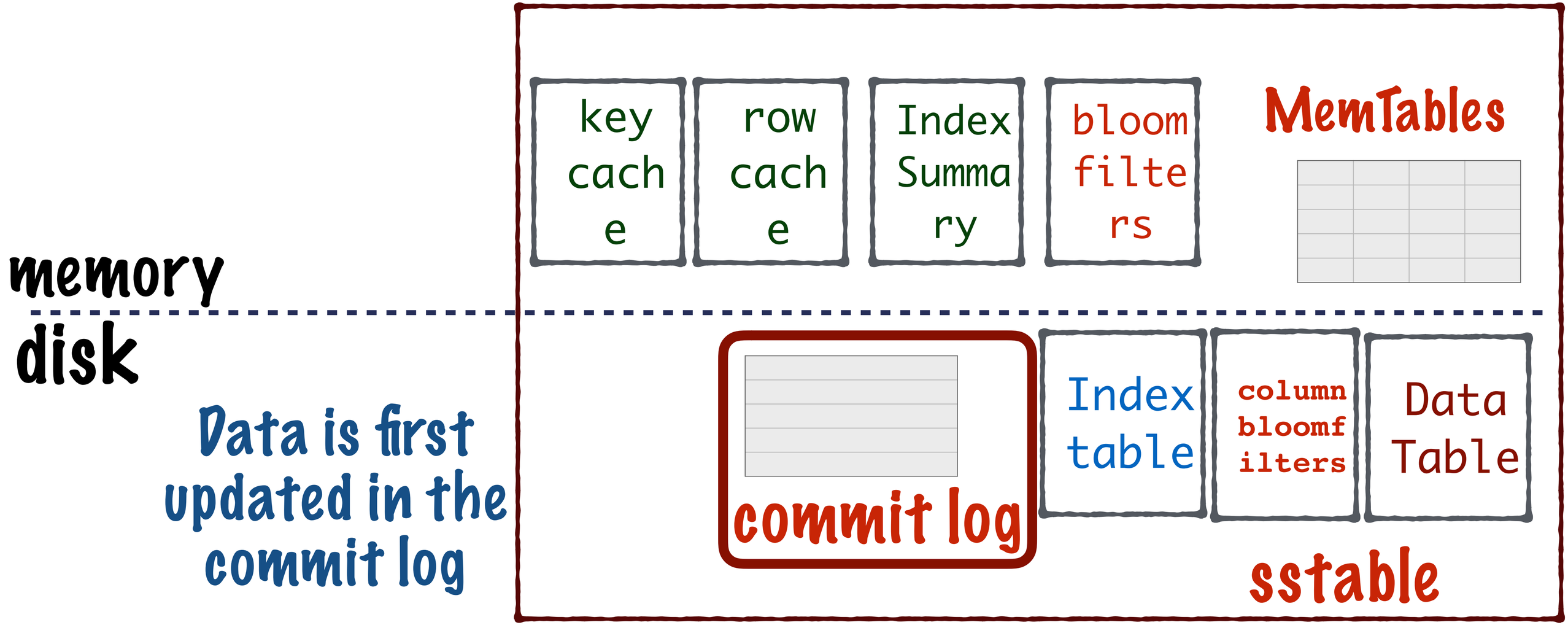
Node


memory
disk
key cache
row cache
Index Summary
bloom filters
MemTables
commit log
Index table
column bloom filters
Data Table
sstable

Let's see the write data path in detail

Node

Write data in a node

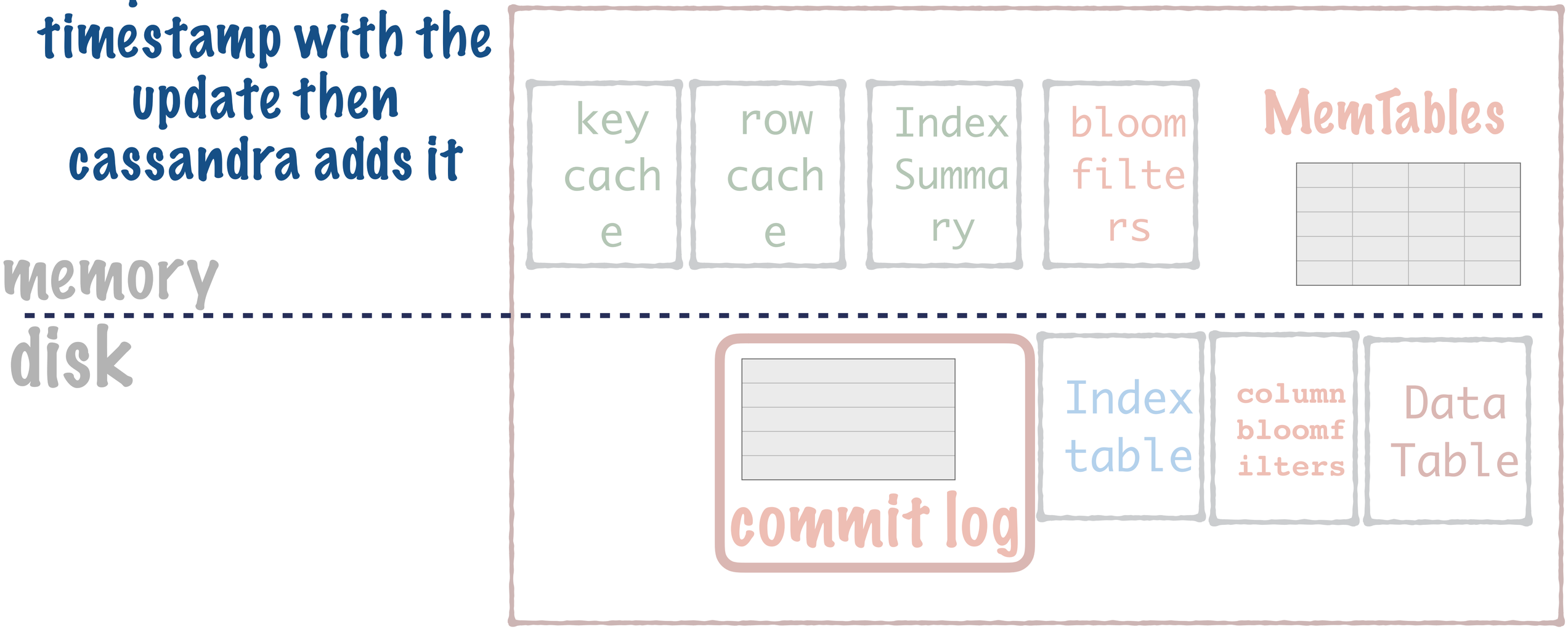


Data is first updated in the commit log

Node

Write data in a node

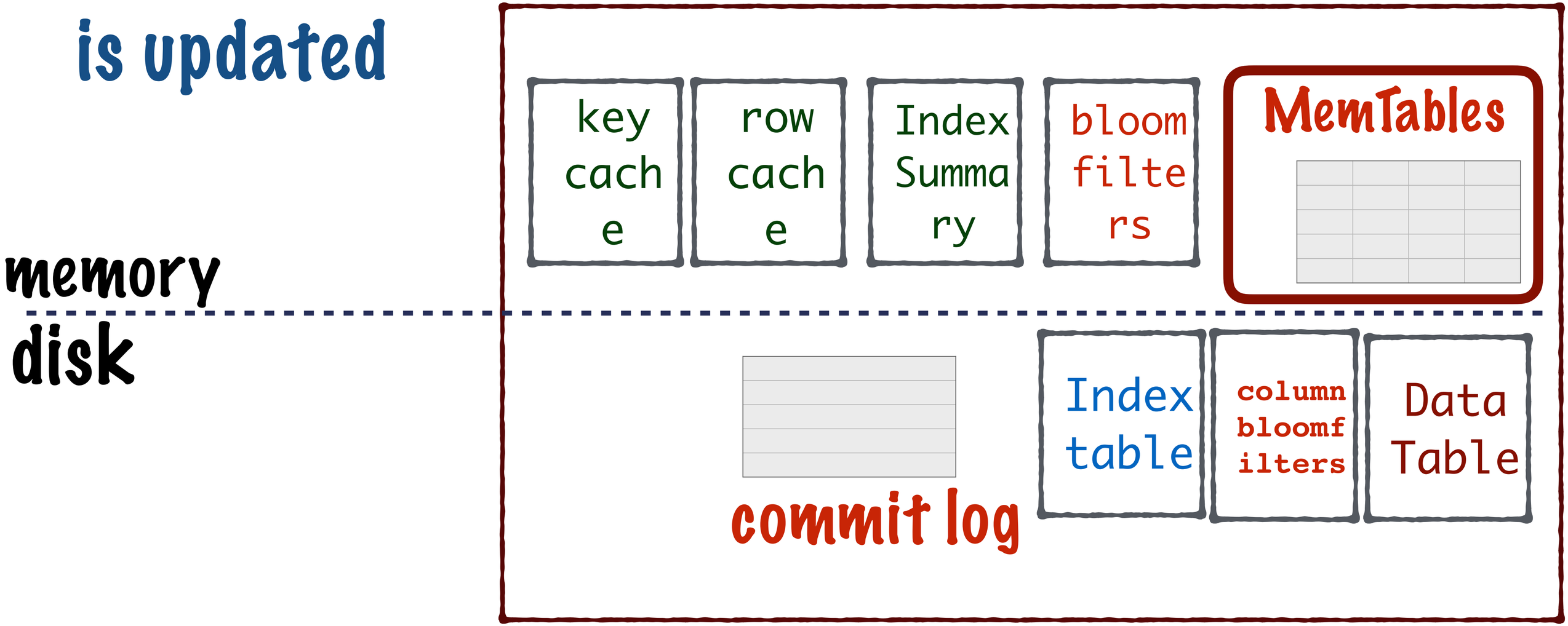
if client hasn't provided the timestamp with the update then cassandra adds it



Node

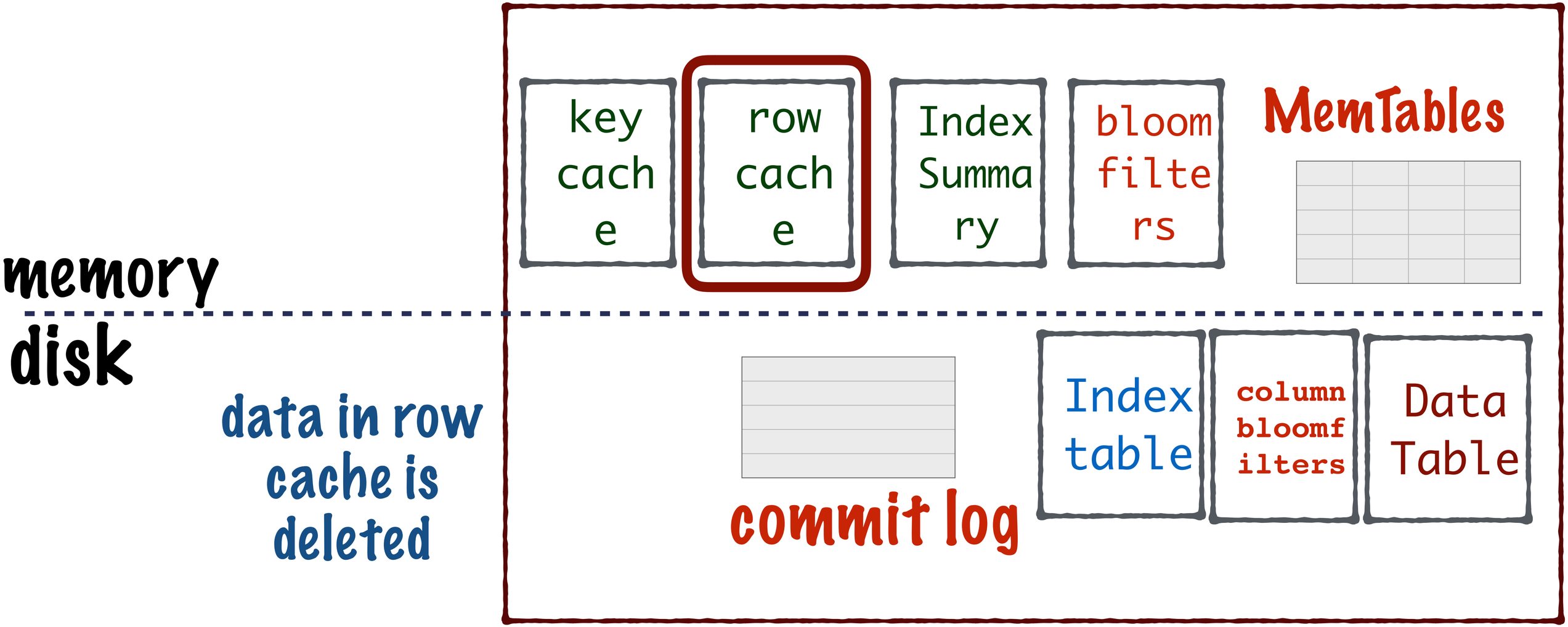
Write data in a node

MemTable is updated



Node

Write data in a node



TOMBSTONES

Cassandra doesn't delete data directly, it marks it to be deleted with a tombstone

How long tombstones live is dictated by gc_grace_seconds

SSTable
Datafile

tombstone information

Index

Data file

TOMBSTONES

Why do we not delete the data?

TOMBSTONES

Why?

Let's say we delete Mobl right away
rather than marking it with a tombstone

and a replica node missed
an update as it was down

TOMBSTONES

and a replica node missed
an update as it was down

Why?

now if a read occurs on
the node that was down

we'll have 'ghost data' in
the cluster

TOMBSTONES

we set gc_grace_seconds long enough to allow nodes which are down to recover and receive the update

default value of this is 10 days but it can be configured according to your requirements

Flushing memtable

Memtable data is flushed to sstables

- if memtable is full or
- commitlog is full or
- an explicit flush is triggered

Flushing memtable

data is sorted in Memtable by token

a new SSTable is created and data is set in the bloomfilter, index, summary tables

data is written to the datafile sequentially

At the end commit log is purged

Node

How many SSTables
are allowed on a node?

Node

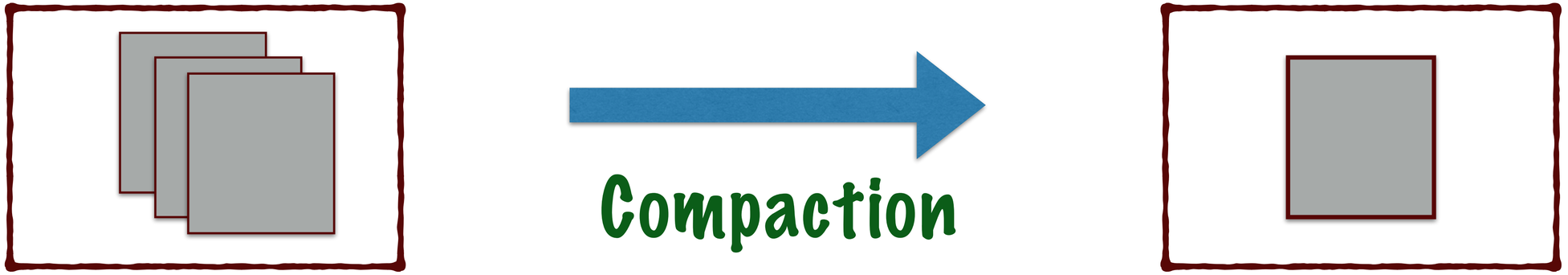
min_threshold property is the maximum number of SSTables allowed on a node

when the number of SStables becomes greater than min_threshold

compaction is triggered on SSTables

Node

Compaction



multiple SSTables

single SSTable

Node Compaction

merges data from all sstables by partition key

selects the data with the latest timestamp for the final table

removes tombstones and deletes associated row /column

Node

Compaction

deletes old sstables

Compaction is incremental

Incoming reads and write requests are served from the new SSTable while compaction is going on

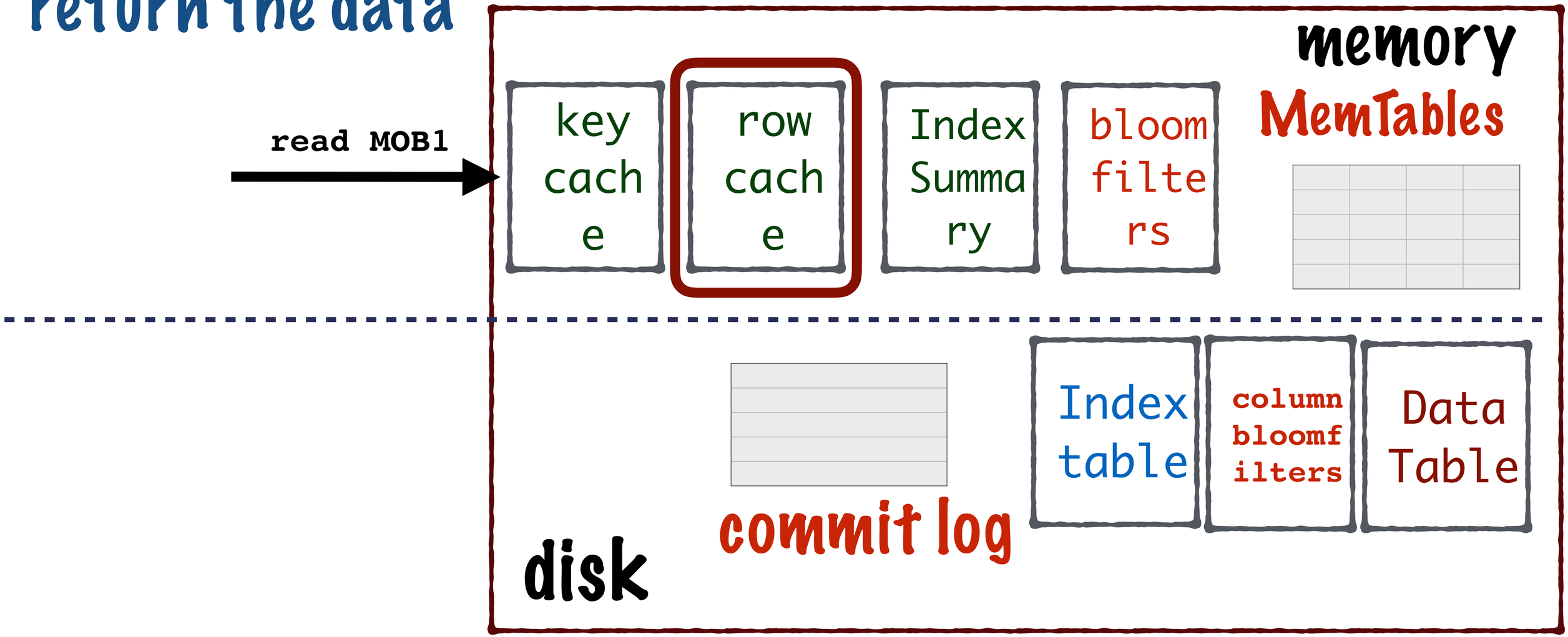
Node

Let's understand
how data is read

Node

Read data in a node

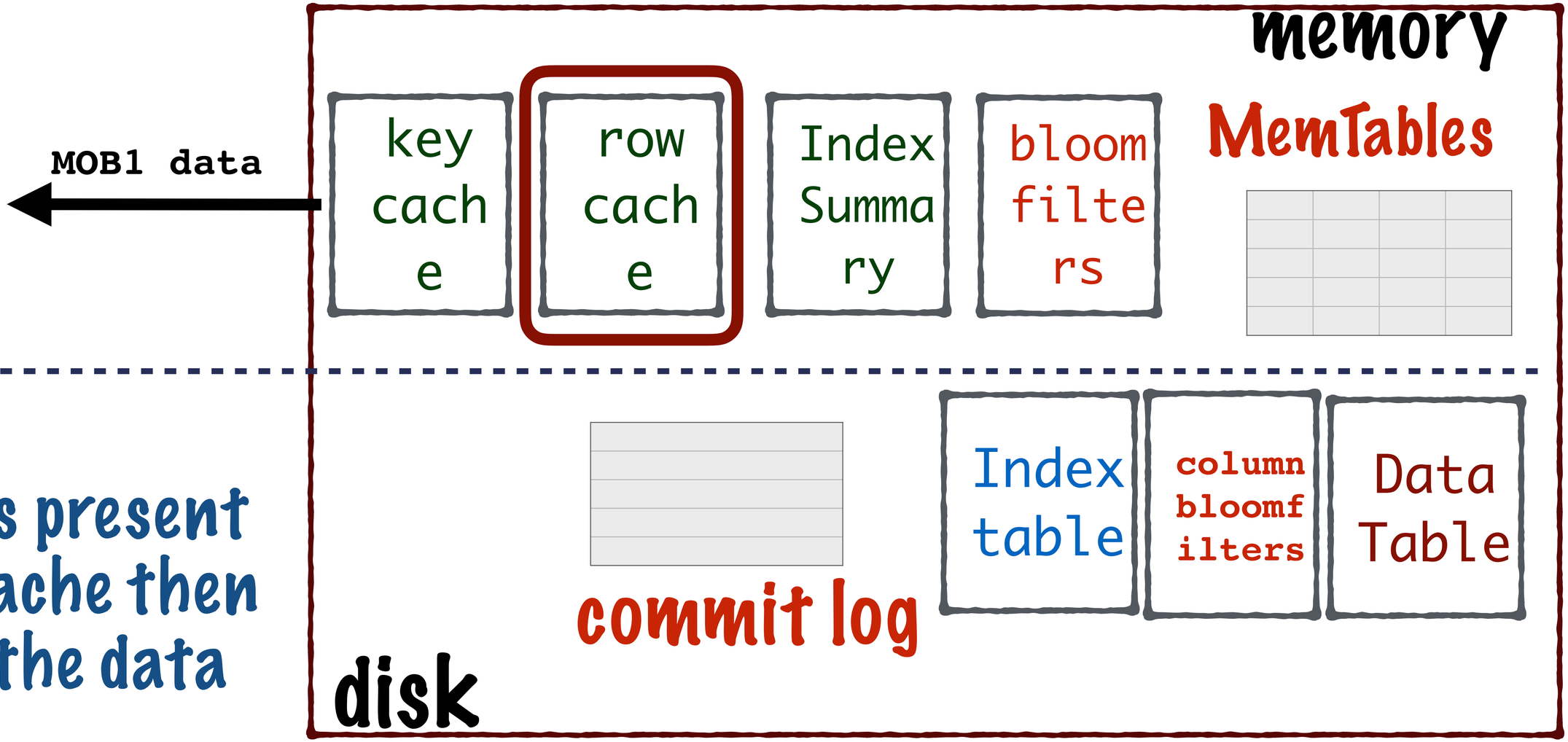
If data is present in row cache then return the data



Node

Read data in a node

If row cache is enabled

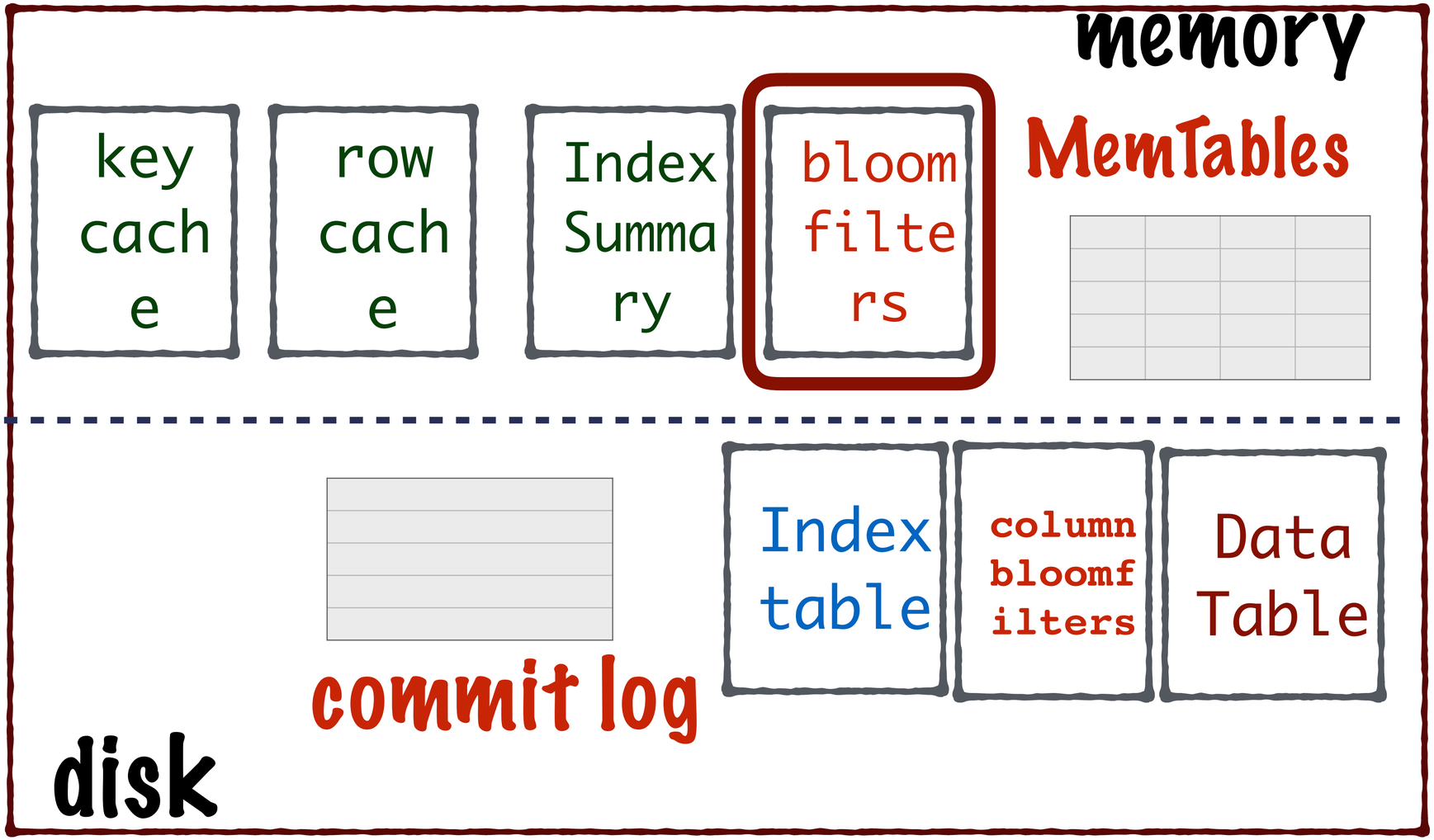


If data is present in row cache then return the data

Node

Read data in a node

If data is not in the row cache
OR
row cache is not enabled



Check all bloom filters to see whether the row exists in SSTables

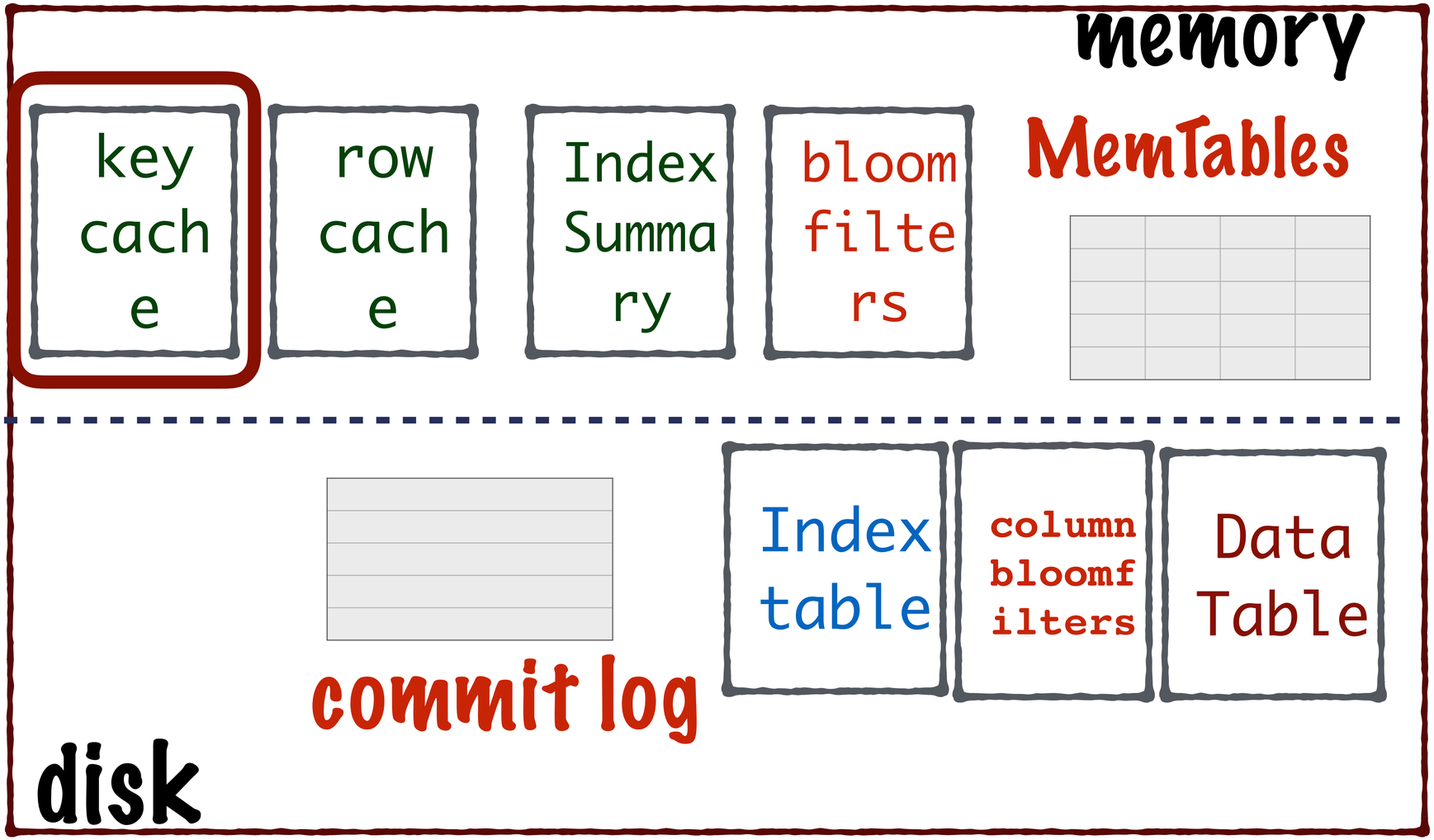
Node

Read data in a node

if key cache is enabled

Keys for which bloom filter returned positive

check key cache if key is present



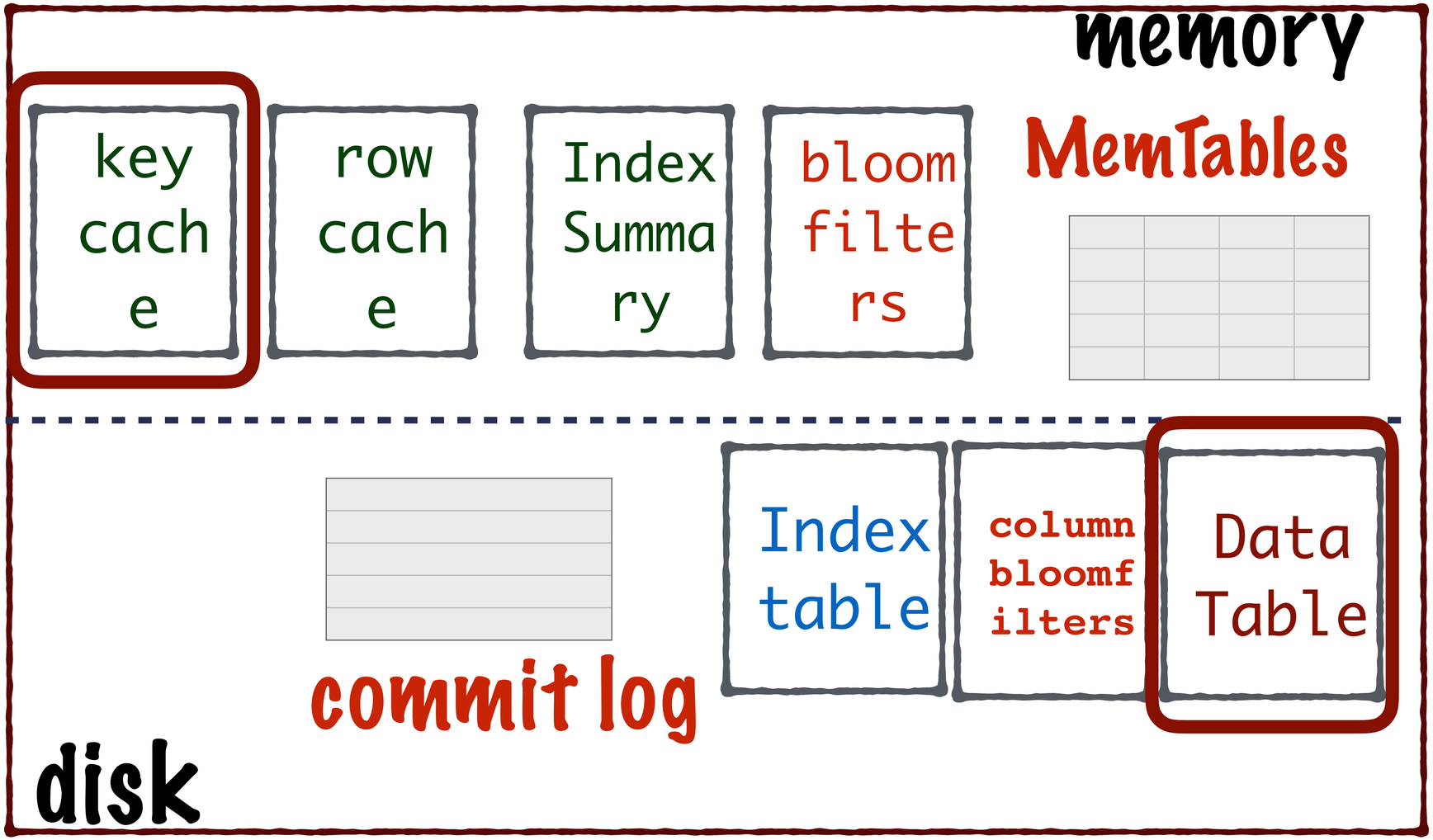
Node

Read data in a node

if key cache is enabled

if keycache has multiple entries of the key then multiple sstables have data

read data from SSTables

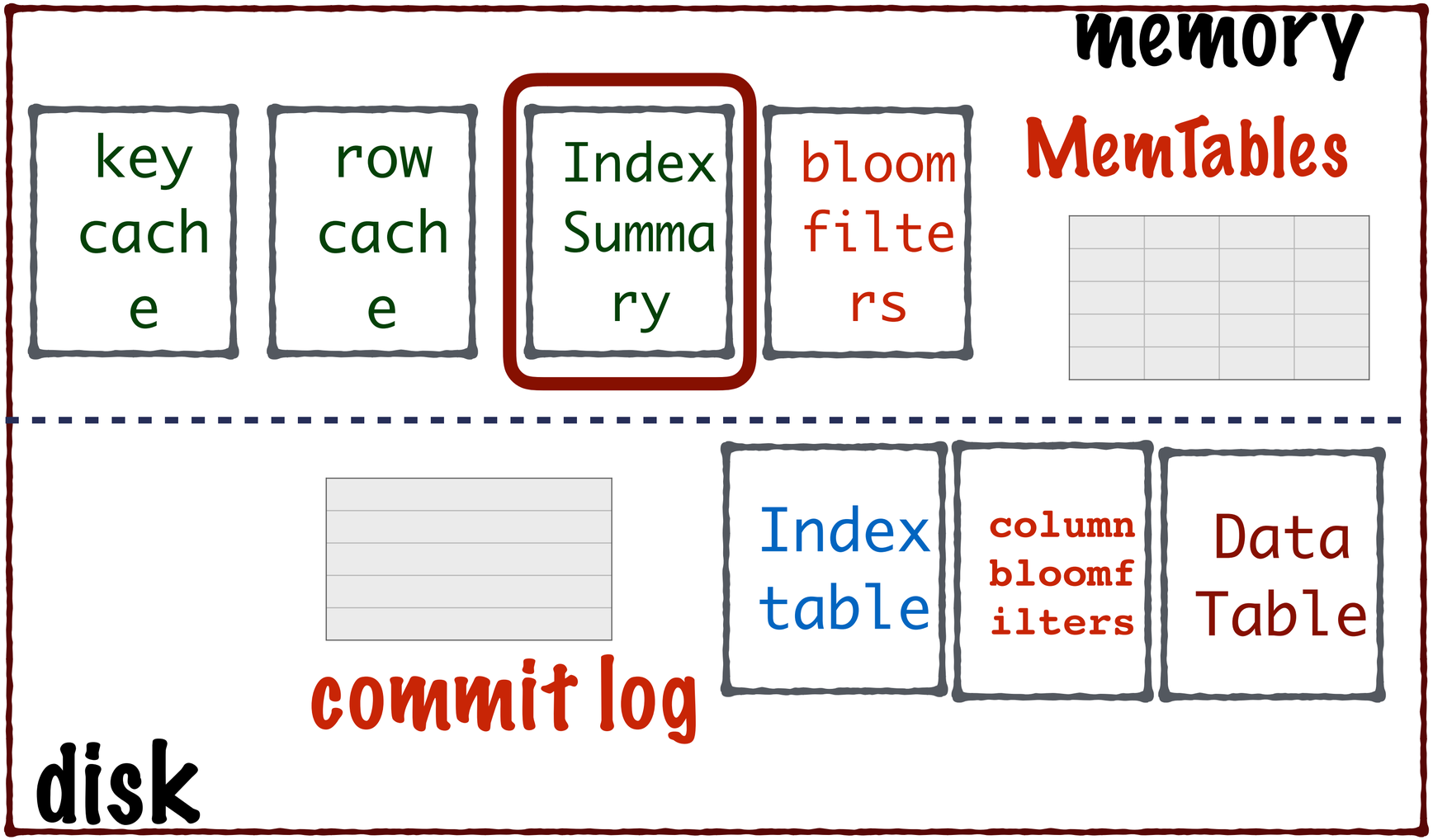


Node

Read data in a node

if key cache is not enabled or key cache doesn't have key entry

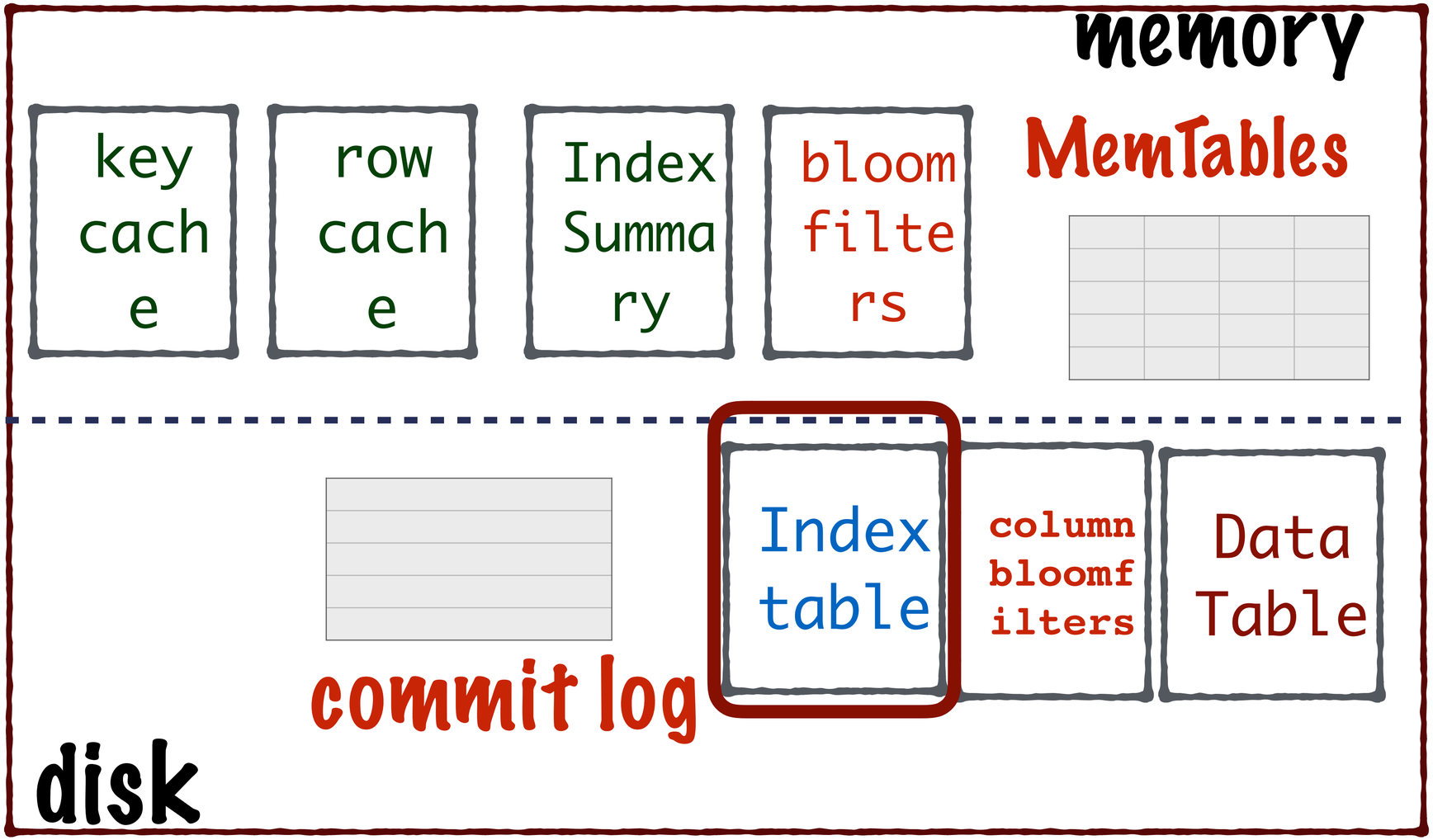
read index summary and get the offset of the partition key in the index table



Node

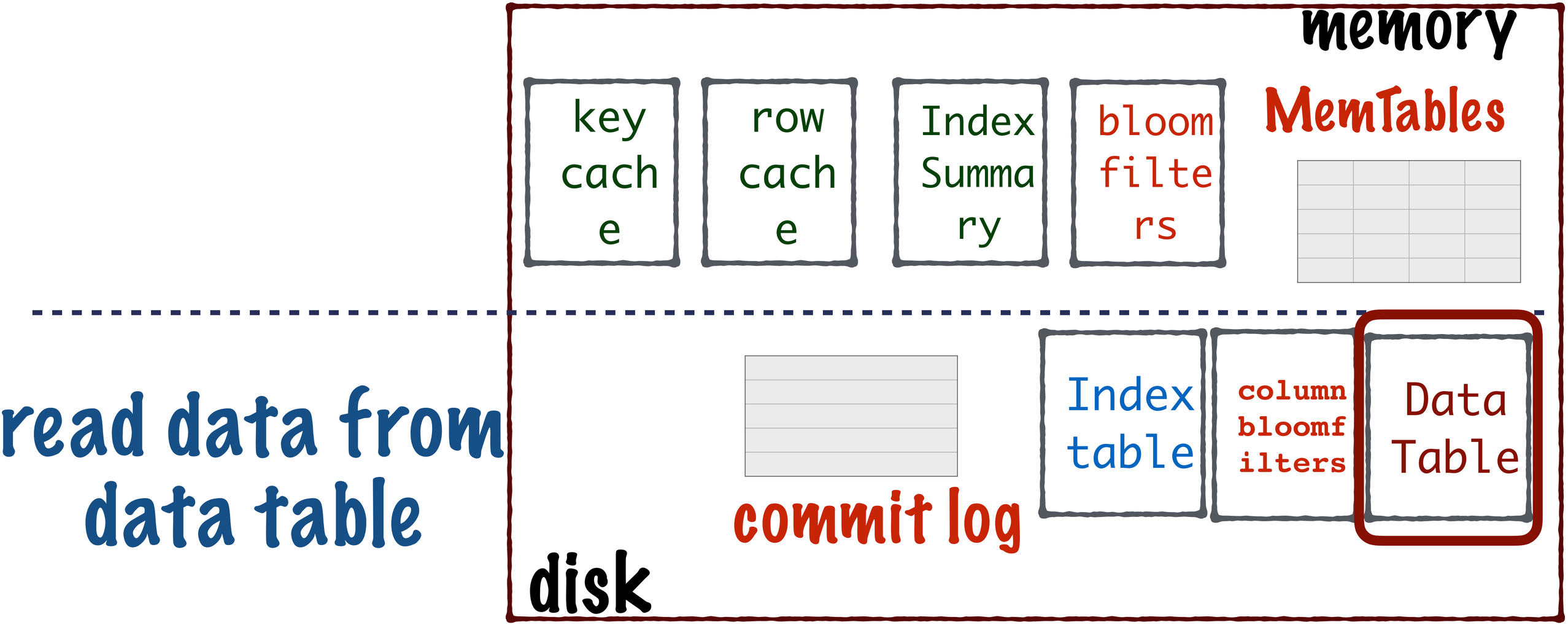
Read data in a node

scan through the index table from the offset sequentially and get the location of data in data table



Node

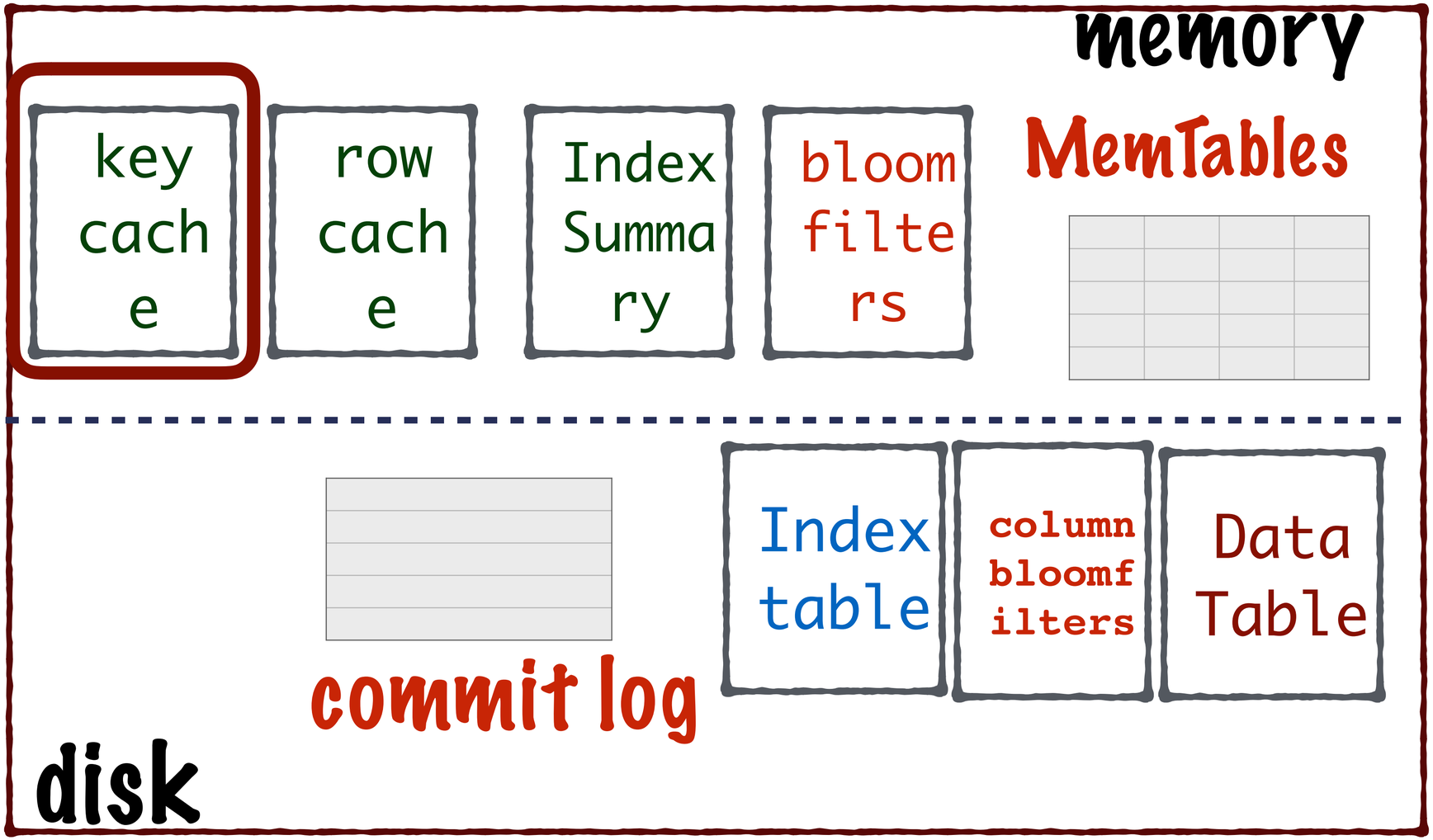
Read data in a node



Node

If key cache is enabled

Read data in a node



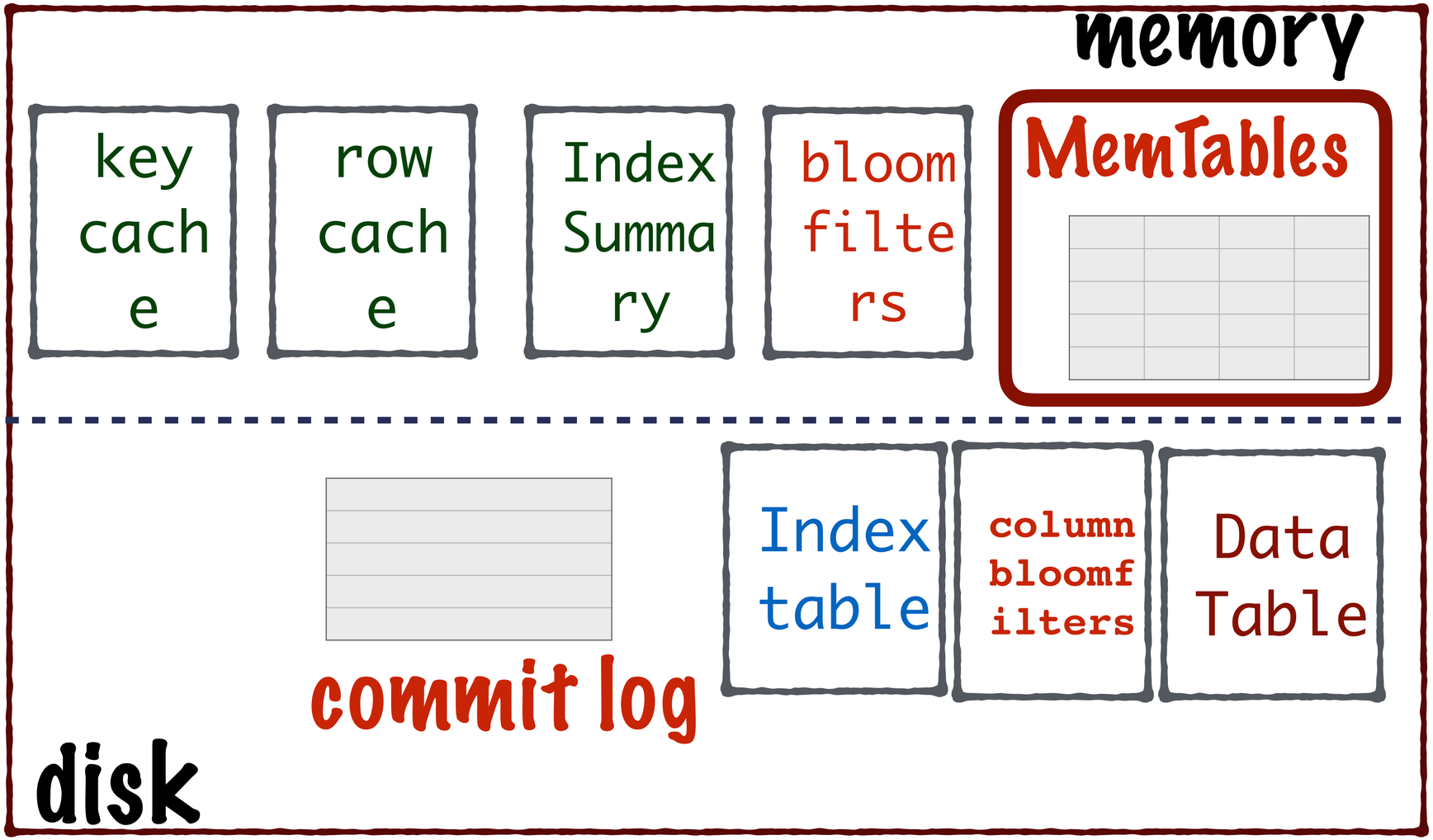
update key cache

Node

Read data in a node

Read data from MemTables

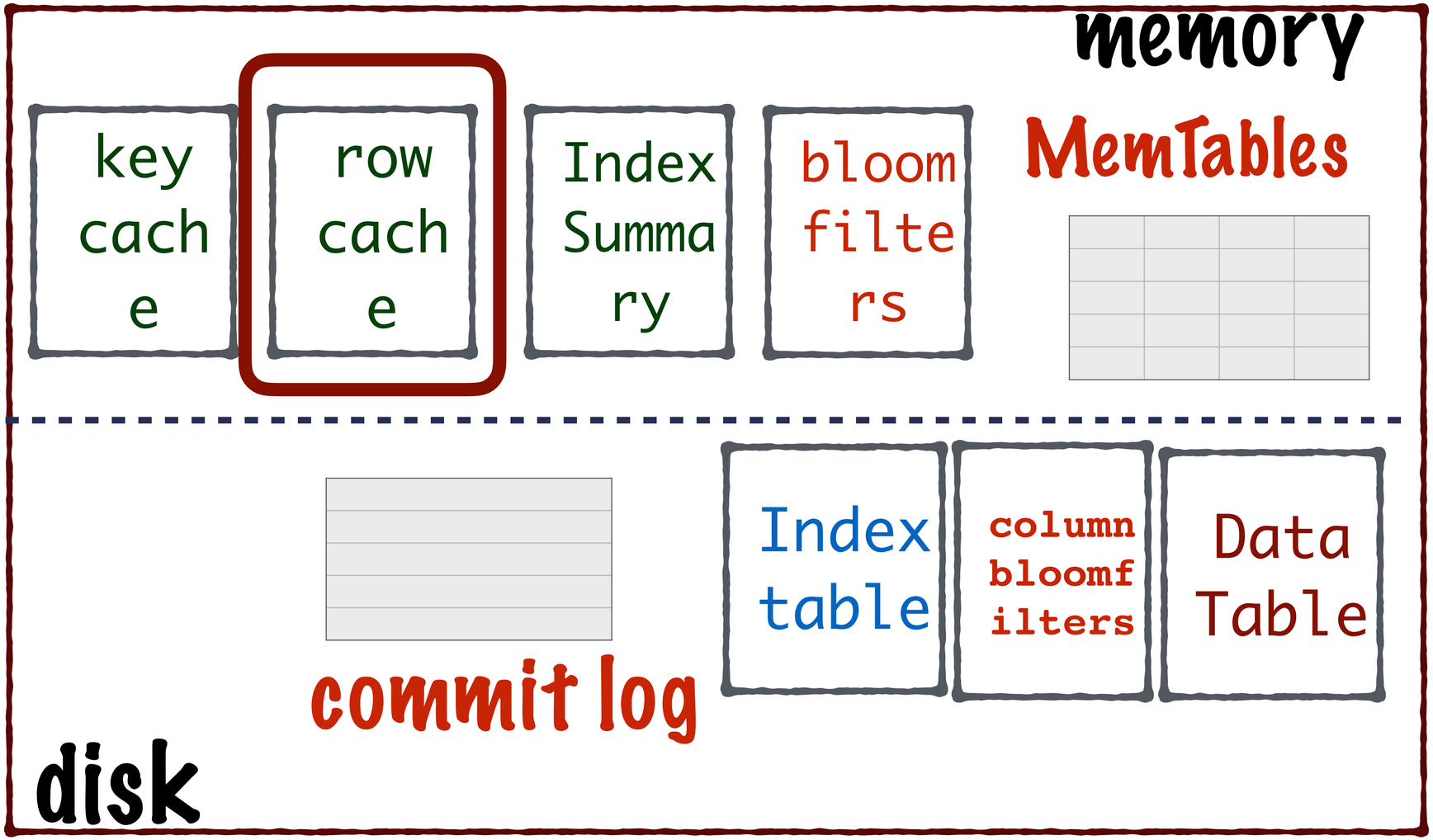
Merge the cells from SSTable and MemTable by timestamp



Node

Read data in a node

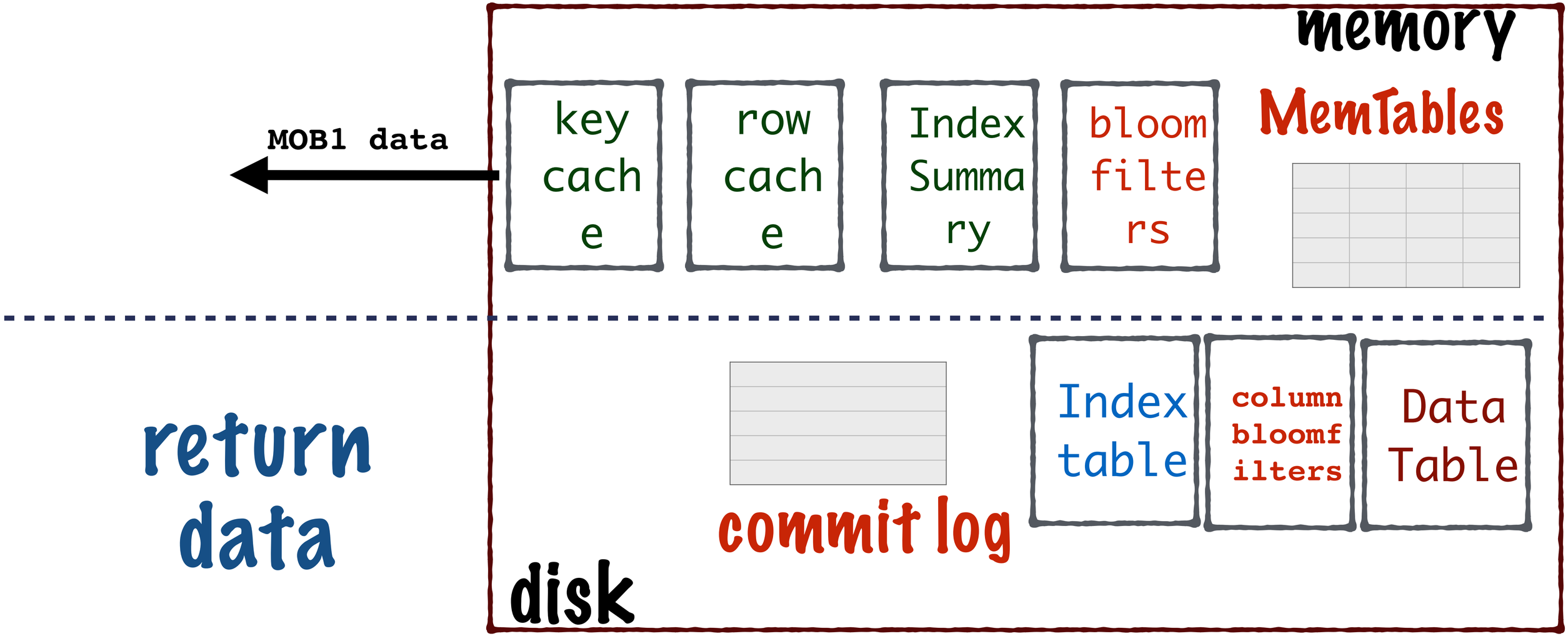
If row cache is enabled



Update the row cache

Node

Read data in a node



This is all within a node

But what about internode communication?

Gossip Protocol

is used for inter
node communication

Gossip Protocol

nodes exchange state
information about themselves

if a node has state information about
any other node, it shares that too

the mechanism runs
periodically (1sec)

Gossip Protocol

Soon all nodes know the
each others state

the information ('gossip') has
a version associated with it

As information exchange goes on, the older
version is overwritten with new data

Gossip Protocol

Hinted Handoff is trigged by gossip

A node notices whether the node for which it has a hint file has recovered or not using gossip

If the node has recovered it relays the writes from each hint file