BBDD distribuidas
Cassandra

Ampliación a Bases de Datos

Profesor: Pablo Ramos

pablo.ramos@u-tad.com

INTRODUCCIÓN

- Uso de columnas para almacenar la información:
  - Clave: nombre único que referencia la columna
  - Valor: contenido de la columna.
  - Timestamp: fecha en la que el documento fue insertado o actualizado. Utilizado para comprobar la validez del dato en sistemas distribuidos.

Clave	Id	Nombre
Valor	12345	Pepe
Timestamp	2013-10-30 05:02:45.004000	2013-10-30 05:02:50.004000

INTRODUCCIÓN

- Cassandra es una base de datos diseñada para soportar grandes volúmenes datos distribuidos en cientos de servidores garantizando la alta disponibilidad de los datos y asegurando la integridad de la base de datos frente a fallos masivos.

- Sus características principales son
  - Descentralización (alta disponibilidad)
  - Replicación
  - Tolerancia a fallos
  - Escalabilidad

ARQUITECTURA: Elementos principales

- Elementos:
  - **Nodo:** Instancia de Cassandra
    - 1 o más en un servidor (Desarrollo)
    - 1 por servidor (Producción)
  - **Partición:** Unidad básica de información. De vital importancia para la replicación y la ordenación de datos.
  - **Rack:** conjunto lógico de nodos.
  - **Data center:** conjunto lógico de racks
  - **Cluster:** conjunto completo de nodos. (Token ring completo)

ARQUITECTURA: Nodos

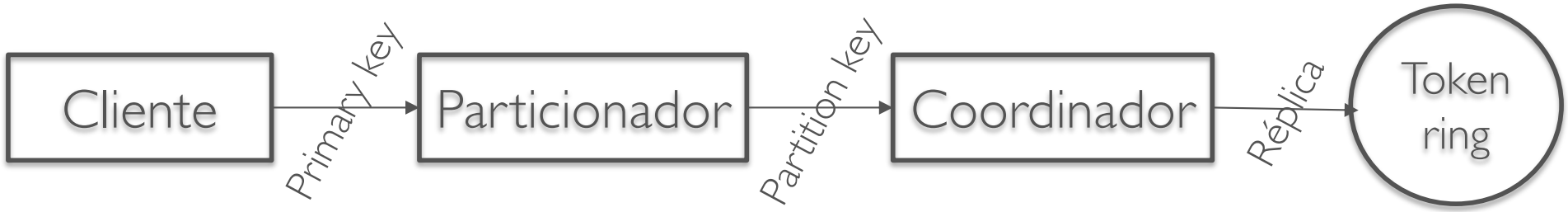
- **Nodo:** Instancia de Cassandra.
  - **Nodo semilla:** Nodo al que un nodo nuevo se comunica por primera vez cuando se conecta a un clúster para conocer la estructura del clúster y su papel.
  - **Coordinador:** Cada uno de los nodos del clúster que recibe una petición de un cliente.
    - Se encarga de coordinar la petición a través de los nodos y de que esta se realice y se devuelva el resultado a través de él.
    - Normalmente es elegido de forma aleatoria por el cliente en función de la disponibilidad.
    - Si un nodo cae, el siguiente nodo se encargará de coordinar las peticiones.

ARQUITECTURA: Nodos

- **Nodo:** Instancia de Cassandra.
  - **Nodo semilla**
  - **Coordinador**
  - **Driver:** Interfaz de conexión para el cliente.
    - Se encarga de decidir que nodo coordina cada petición.
    - Por defecto se implementa el patrón Round-robin. Direcciona cada nueva petición a un nuevo nodo.

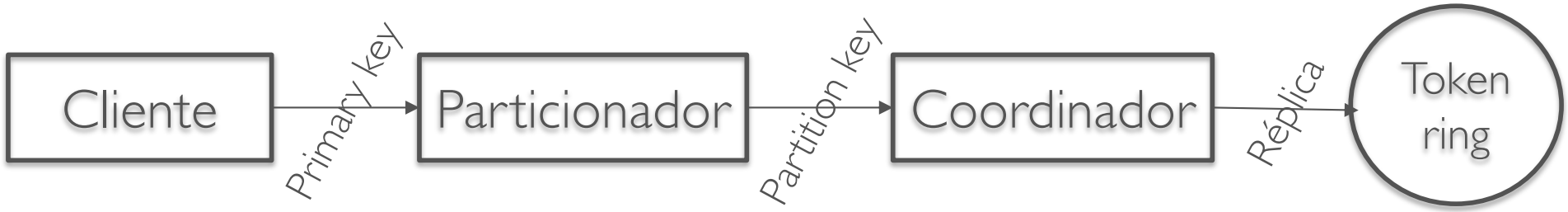
ARQUITECTURA: Partición

- La partición es la unidad básica de almacenamiento.
  - **Proceso de particionado:** se encarga de asignar un token (partition key) a cada unidad de datos a partir de su clave primaria (primary key). Dicho token identificará de forma unívoca las particiones.
  - Las claves de partición son utilizadas para determinar el emplazamiento de los datos en la base de datos. Cada nodo tiene asignado un rango de particiones.
  - **Particionador:** se encarga de realizar el proceso de particionado.



ARQUITECTURA: Partición

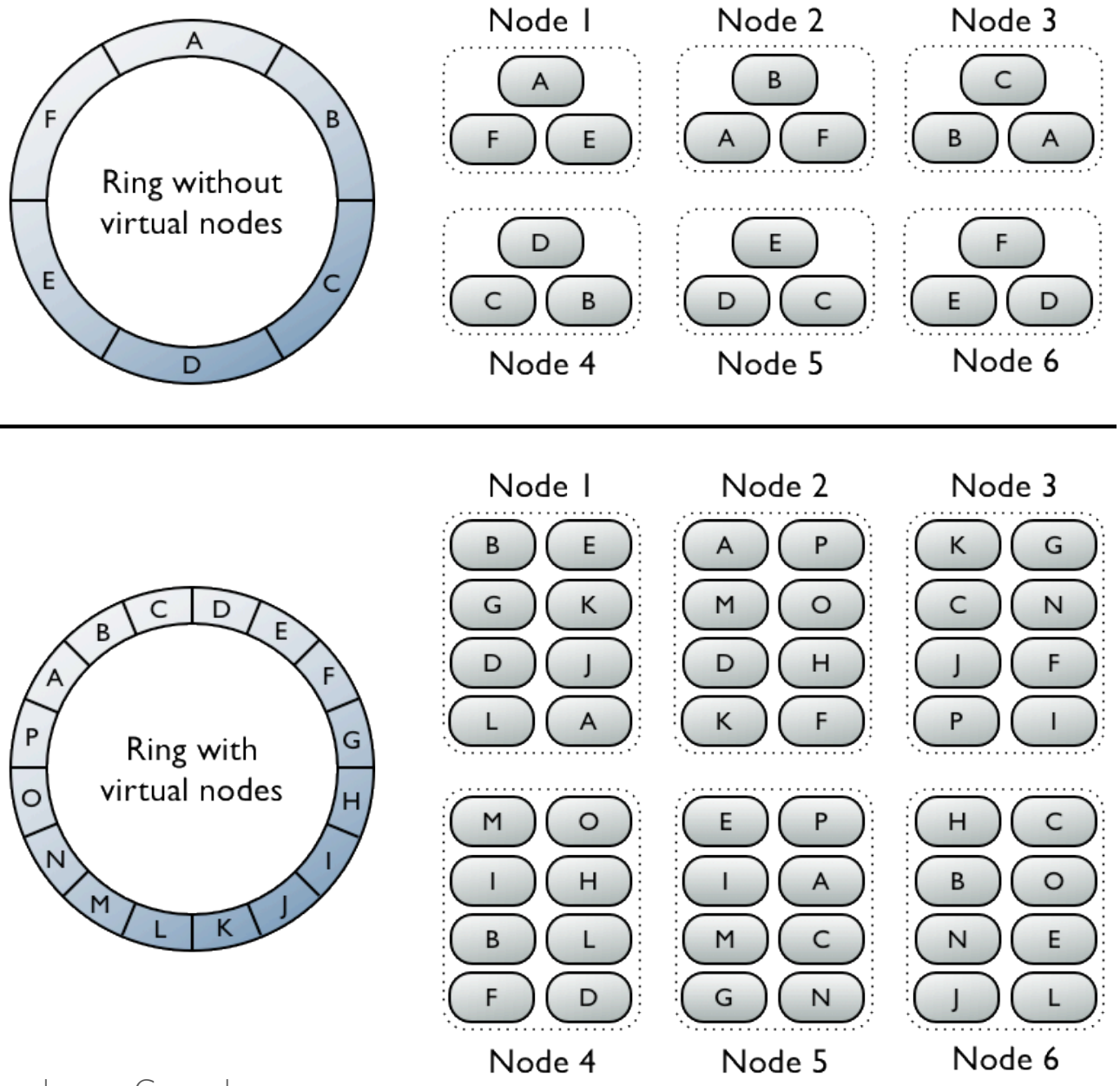
- La partición es la unidad básica de almacenamiento.
  - **Token ring:** Es el conjunto de claves de partición existentes en un clúster. Este conjunto esta subdividido en segmentos y cada segmento está asignado a un nodo. En función de la clave de partición generada por el particionador, la réplica de una partición será dirigida por el coordinador a un nodo u otro.



ARQUITECTURA: Nodos virtuales

- Desde la versión 2.0, cada nodo es subdividido en subnodos virtuales cuyo segmento del token ring es discontinuo.
- Los **nodos virtuales** se comportan como nodos normales.
- La principal característica de los nodos virtuales es que posible asignarles de forma dinámica y automática sus respectivos segmentos del token ring.
  - **Bootstrap:** Proceso llevado a cabo cuando un nuevo nodo es levantado. Se reasignada de manera automática parte del token ring al nuevo nodo y sus replicas.
  - **Decommision:** Proceso llevado a cabo cuando un nodo es eliminado. Se reasignada de manera automática su parte del token ring al resto de nodos y sus replicas.

ARQUITECTURA: Nodos virtuales



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

ARQUITECTURA: Replicación

- La **replicación** permite a Cassandra ofrecer una alta disponibilidad y tolerancia a fallos. La unidad básica de replicación es la partición. No existen originales y/o copias de una partición, todos son replicas.
  - Cada partición tiene una o varias replicas en función de la configuración de replicación especificada en su keyspace.
  - **Factor de replicación:** Determina cuantos nodos deben realizar una petición de escritura. Como mínimo un dato debe estar en un nodo y como máximo en la totalidad de los nodos del clúster.
  - **Estrategia de replicación:** Determina como se distribuyen las replicas. Si existen varios racks en un data center, las replicas se distribuyen de forma equitativa en los racks
    - SimpleStrategy: Se distribuyen las replicas de forma secuencial.
    - NetworkTopologyStrategy: Cada data center tiene su propio factor de replicación. Cada data center tendrá su coordinador remoto para realizar la replicación.

ARQUITECTURA: Nivel de consistencia

- El nivel de **consistencia** determina cuantos nodos deben responder a una petición antes de que el coordinador devuelva la respuesta al cliente.
  - **Escribiendo:** Cuantos han almacenado la partición.
  - **Leyendo:** Cuantos han devuelto la partición.
  - Se puede definir distintos niveles de consistencia en función de cuantos nodos con replica deben enviar "acknowledged"

ONE, TWO, THREE	El primero nodo, los dos primeros…
ANY	Cualquier nodo.
ALL	Todos los nodos.
QUORUM	Al menos el 51% del factor de replicación
LOCAL_ONE	El primer nodo del data center
LOCAL_QUORUM	Quorum en el data center
EACH_QUORUM	Quorum de cada data center

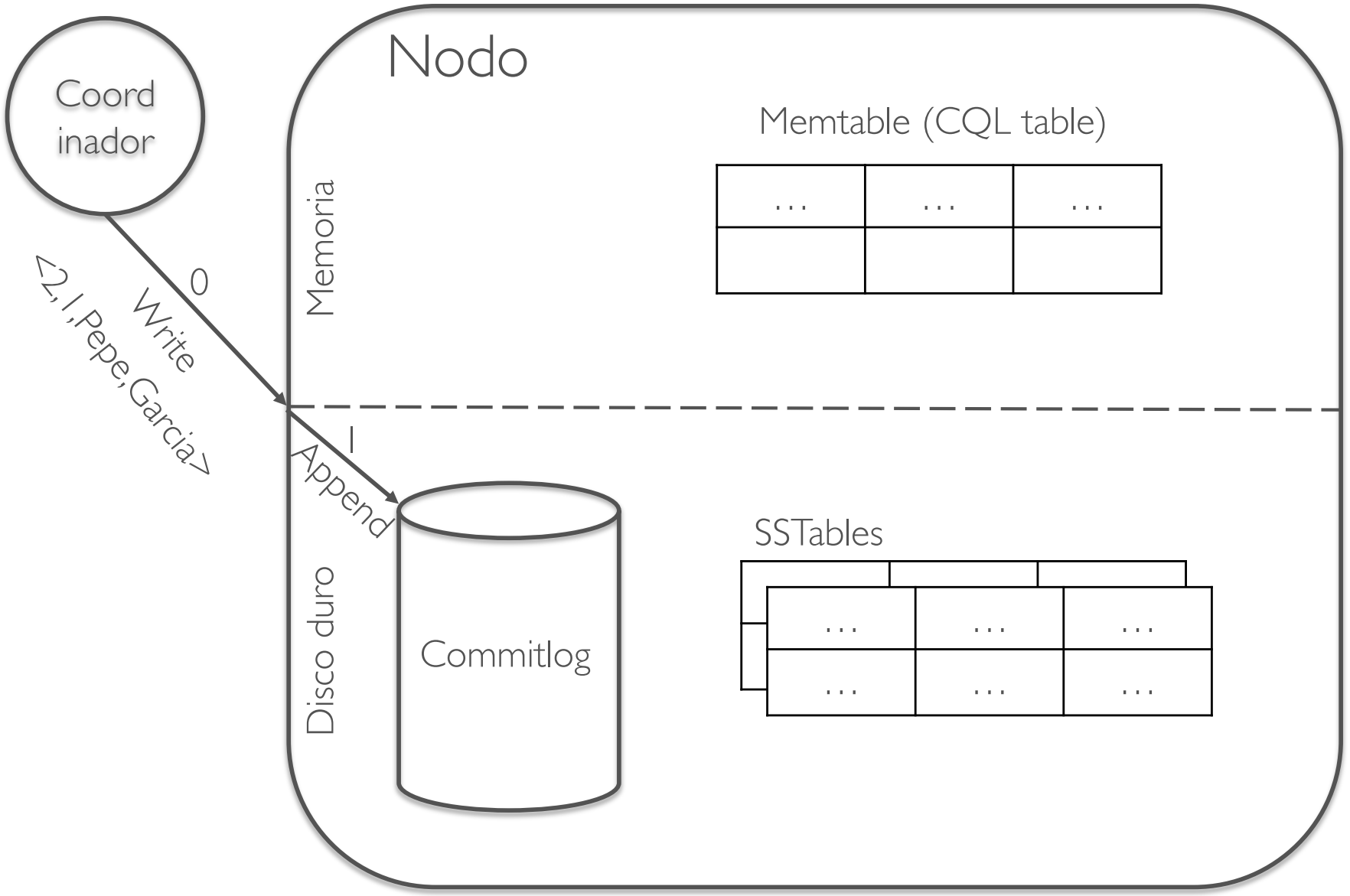
ARQUITECTURA: Hinted Handoff

- **Registra caídas y fallos en los nodos.**
  - Permita gestionar las operaciones de escrituras en nodos que están caídos.
    - En caso de que un nodo este caído, el **coordinador** guarda un "**hint**" que especifica la replica a escribir en el nodo caído.
    - Cuando el nodo vuelve a estar disponible el coordinador realiza de nuevo la operación de escritura.
    - Si un nodo esta caído por más de tres horas, se entiende que tiene un problema mayor y no se guardan "hints"

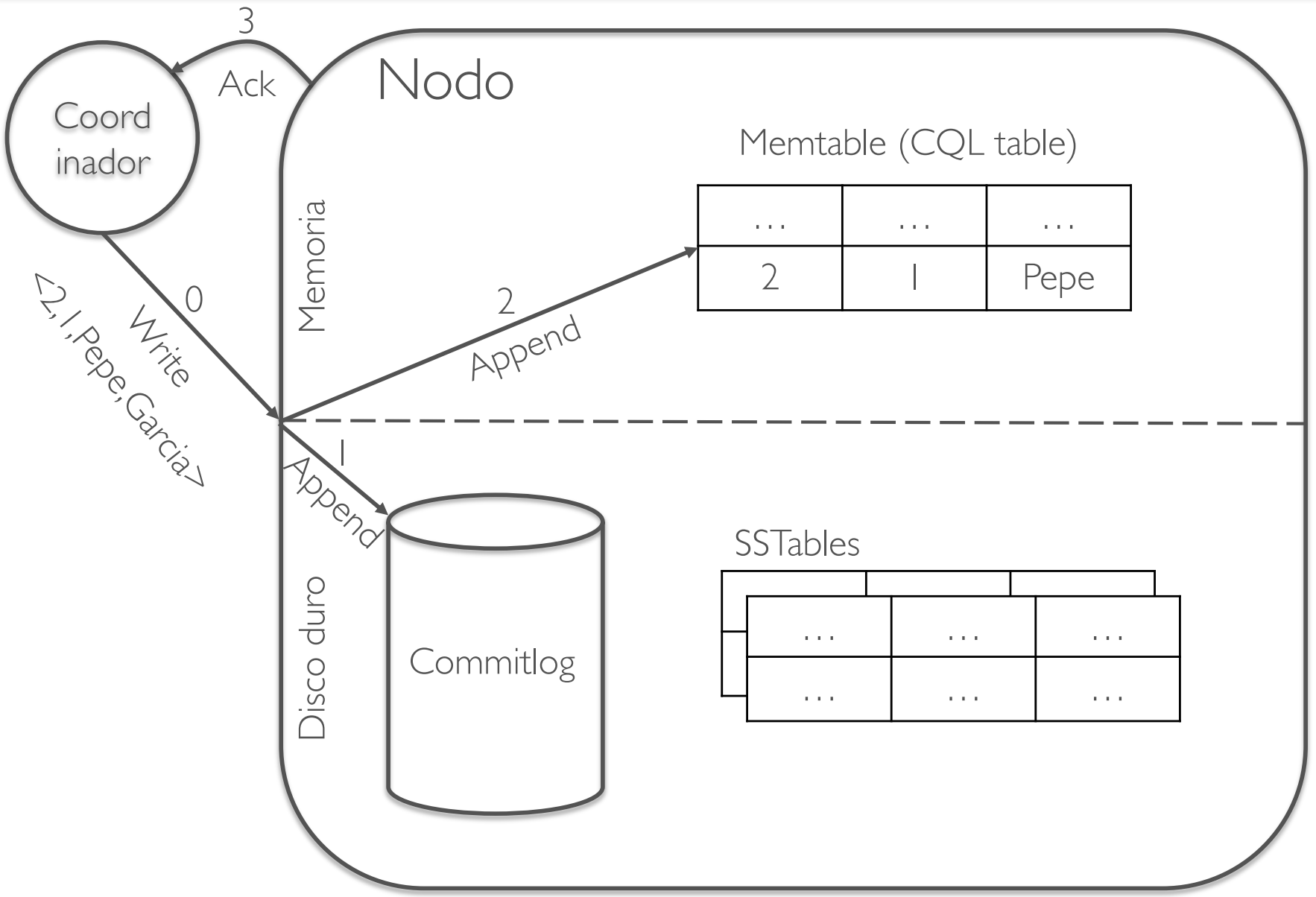
ARQUITECTURA: Write path flow

- El proceso de escritura de datos en cassandra (insert, update, delete) está compuesto por cuatro etapas que aseguran la durabilidad y consistencia de los datos así como la eficiencia de las consultas.
  - Escritura en los Memtables (CQL tables)
  - Escritura en el Commitlog
  - Escritura en las SSTables (String Sorted Tables)
  - Compactación de los datos en las SSTables

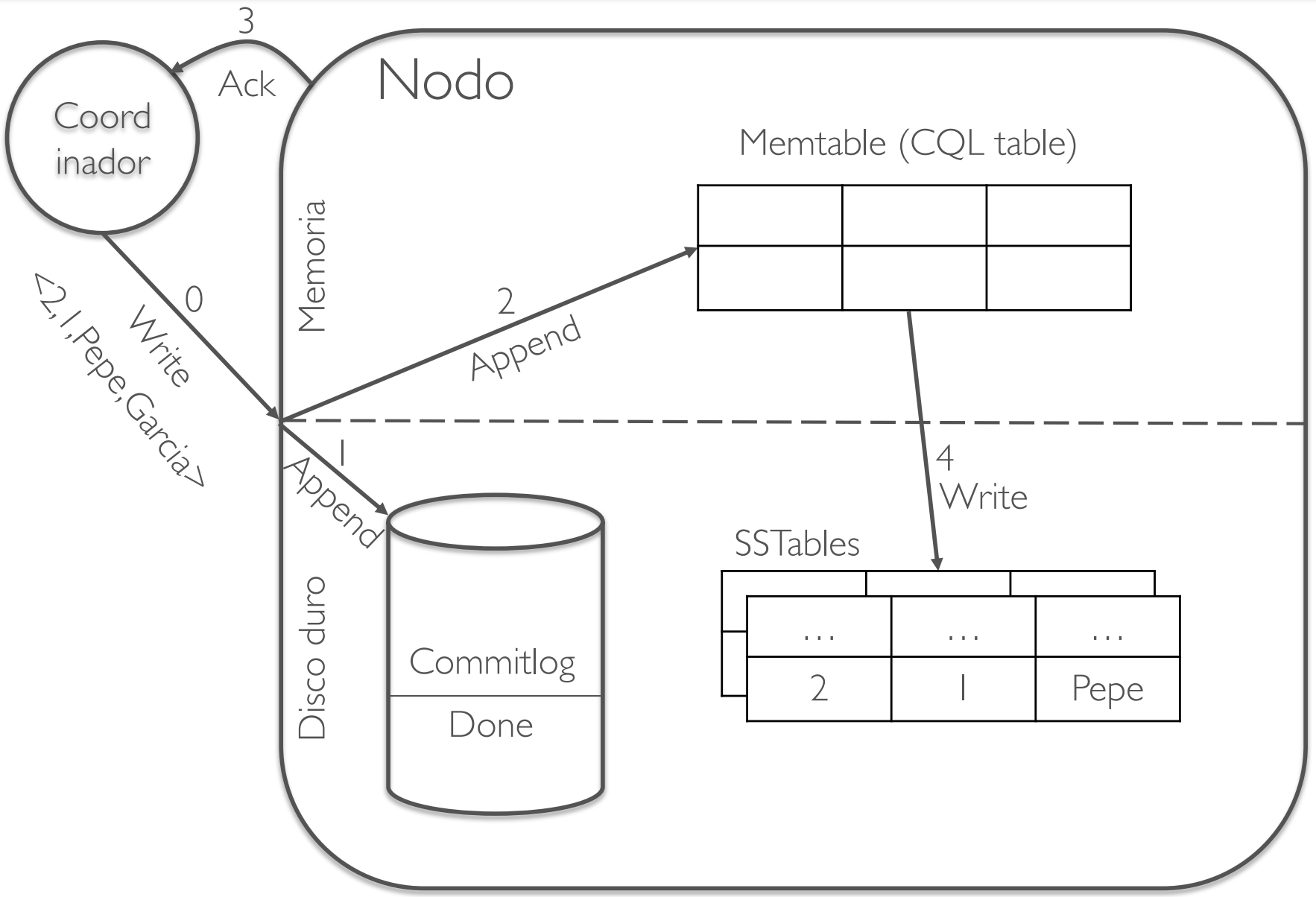
ARQUITECTURA: Write path flow



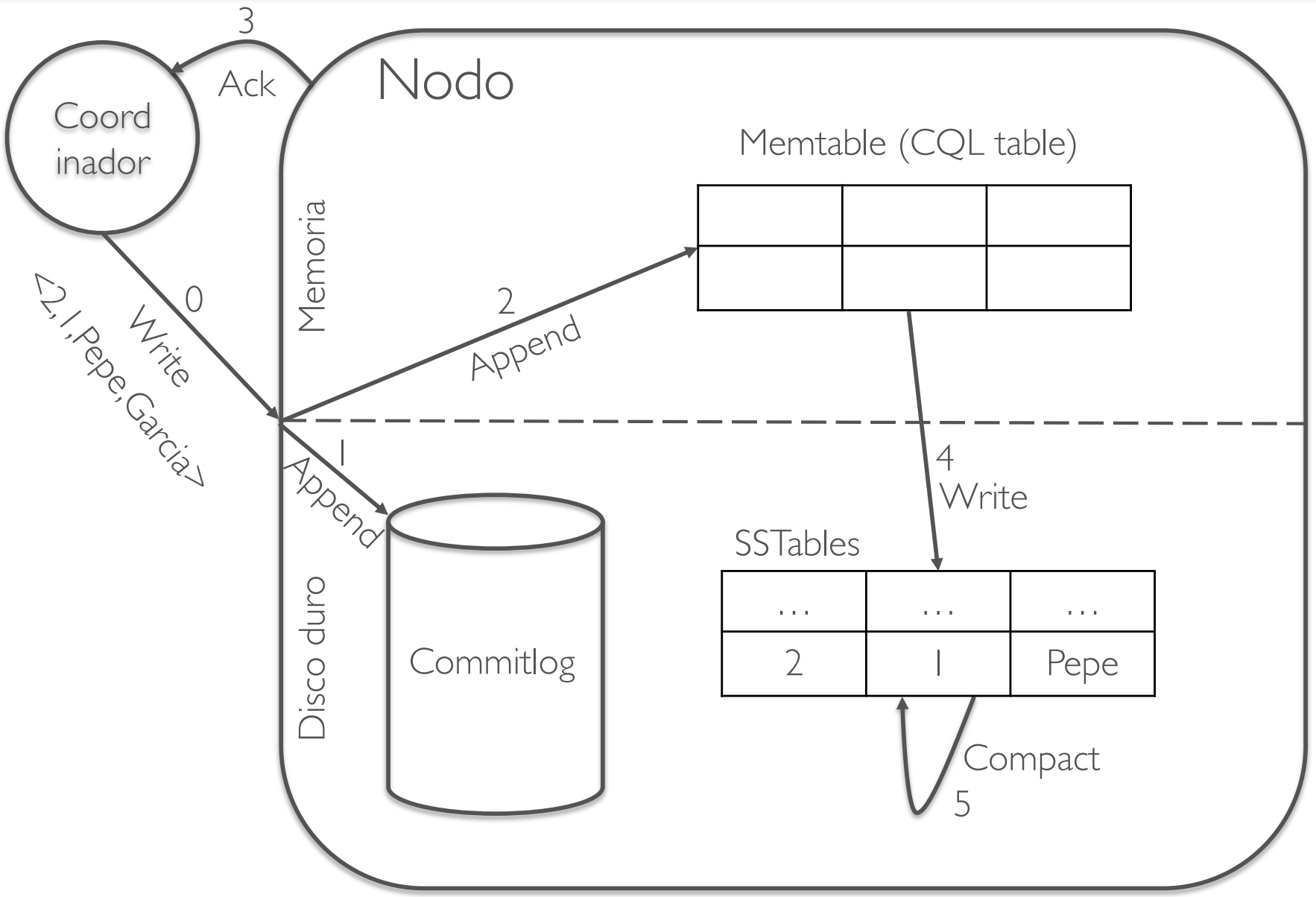
ARQUITECTURA: Write path flow



ARQUITECTURA: Write path flow (Flush del memtable)



ARQUITECTURA: Write path flow (Compactación)



ARQUITECTURA: Write path flow

- Commitlog
  - Introduce cada nueva consulta de escritura en disco duro de modo que si existe cualquier caída del servidor, este es capaz de actualizar las Memtables con las consultas pendientes de ejecutar.
  - La escritura de las consultas en disco se llevarán a cabo cuando las consultas registradas en el commitlog sin escribir en las SSTables (disco) alcance un tamaño determinado.
  - Cuando las consultas son escritas en las SSTables son marcadas con flushed.
  - En realidad las consultas no son escritas directamente en el disco duro. Hasta que no están escritas en disco duro no se devuelve el Acknowledge al coordinador.

ARQUITECTURA: Write path flow

- Memtable
  - Se crea un Memtable por cada tabla CQL de cada KeySpace.
  - En los Memtables se van acumulando las peticiones de escritura en BBDD. Por tanto es posible que tengamos varias entradas para un mismo dato.
  - También se permite las consultas de lectura para aquellos datos que aún no han sido escritos en disco duro.
  - Periódicamente todo el contenido de las Memtables es trasladado a SSTables nuevas en disco duro y sus correspondientes entradas en el Commitlog son marcadas como flushed.

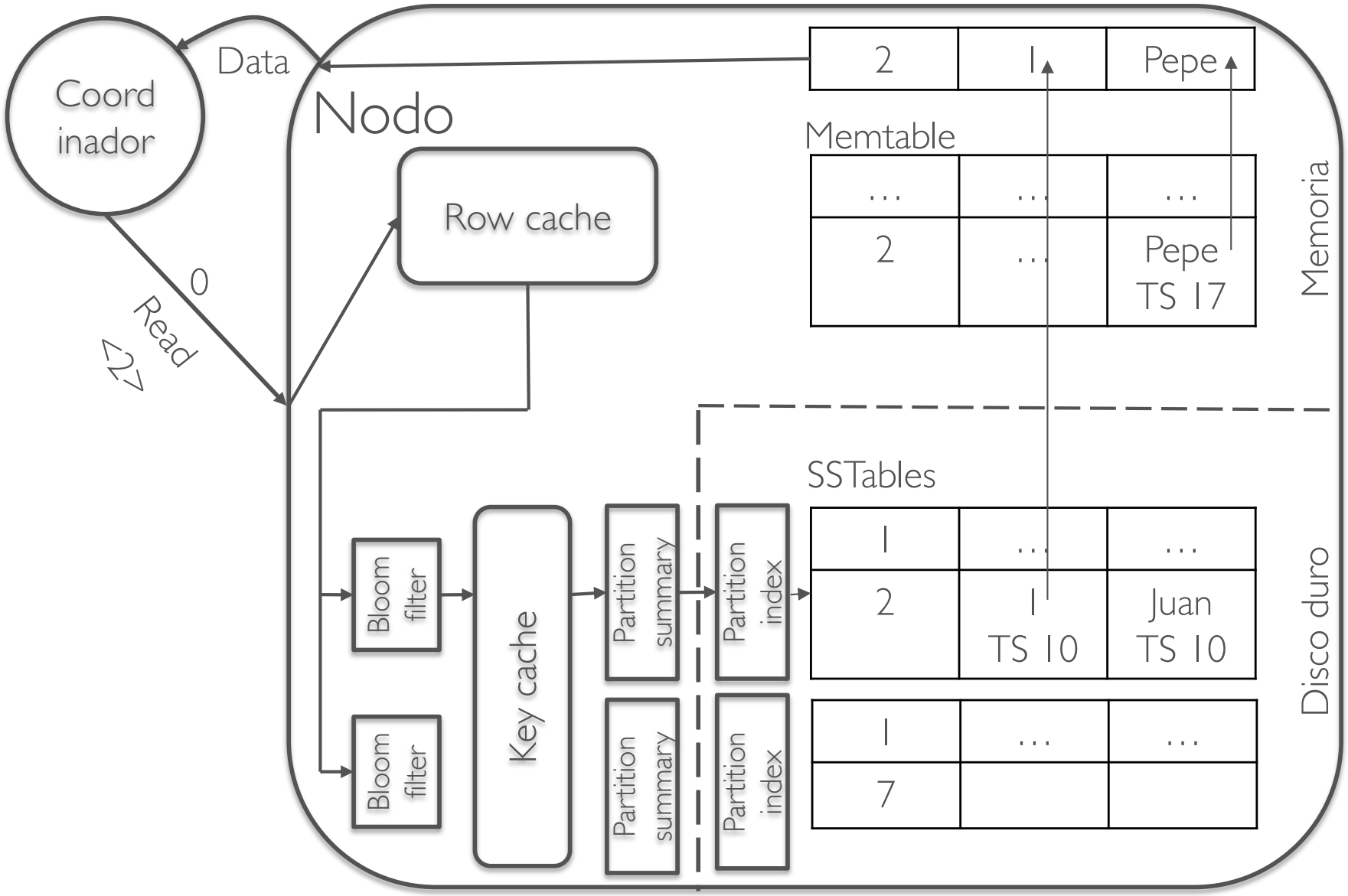
ARQUITECTURA: Write path flow

- SSTable
  - Las tablas SSTables son inmutables. La información simplemente se acumula en las tablas.
  - Se crea una SSTable nueva por cada Memtable cada vez que se realiza el proceso flush. Por tanto, exisitirán varias SSTable por CQL Table.
  - Periódicamente se compacta toda la información contenida en los SSTables de modo que solo exista un SSTable por CQL table. A partir de este momento, existirá una sola entrada actualizada por para un mismo dato en la SSTable.

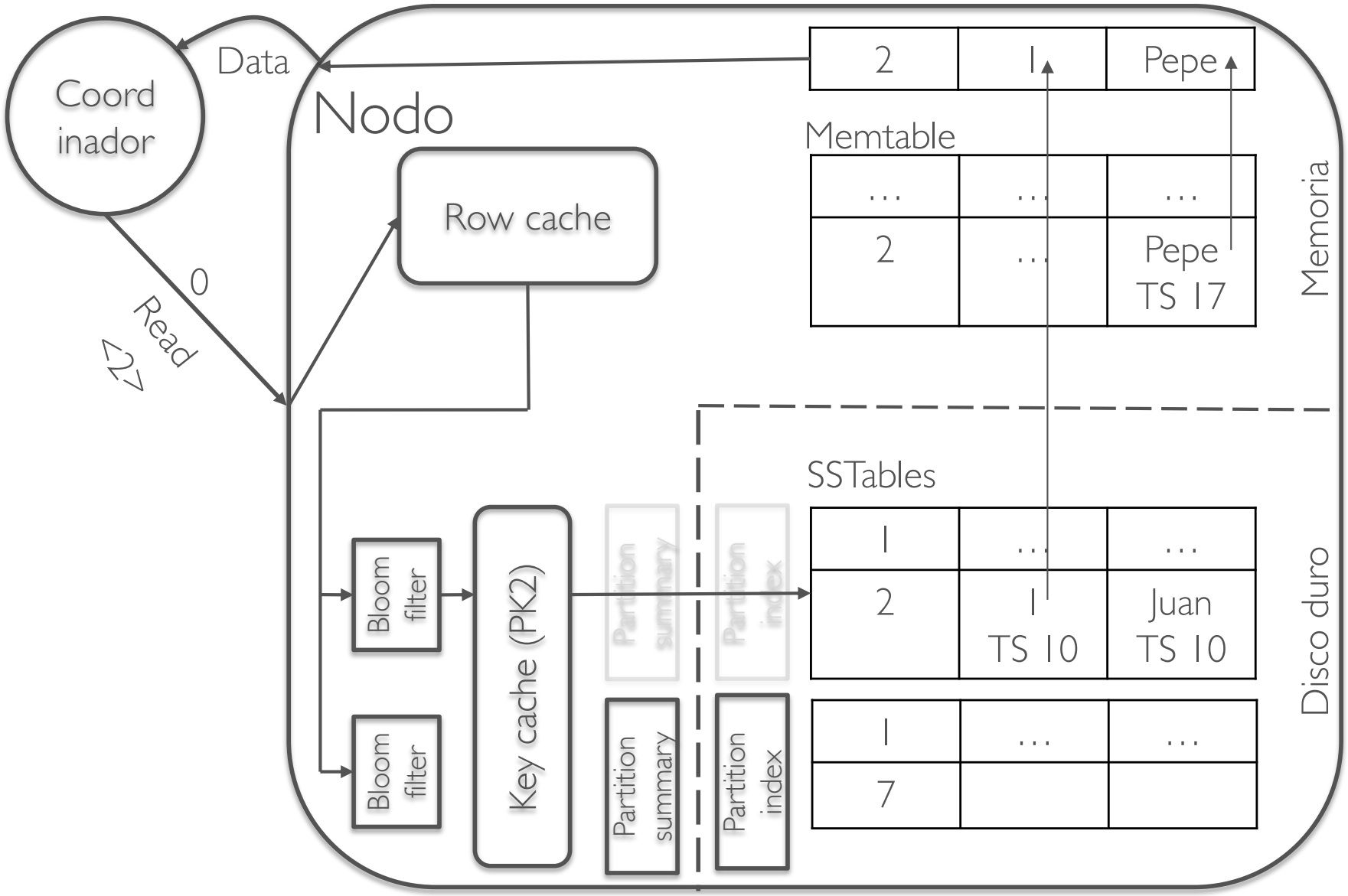
ARQUITECTURA: Read path flow

- El proceso de lectura en cassandra busca tanto en las **Memtables** como en las **SSTables** las entradas para la partition key especificada.
- Una vez localizadas dichas entradas, se devuelve al coordinador mediante un proceso llamado "merge" el conjunto de los campos del partition key con el timestamp más reciente.
- Existen filtros y caches intermedias que permiten acelerar el proceso de consulta.
- **Read_repair_chance** (10%): probabilidad con la que se lanza un proceso de comprobación de consistencia del dato en todos los nodos cuando se lanza un comando de lectura.

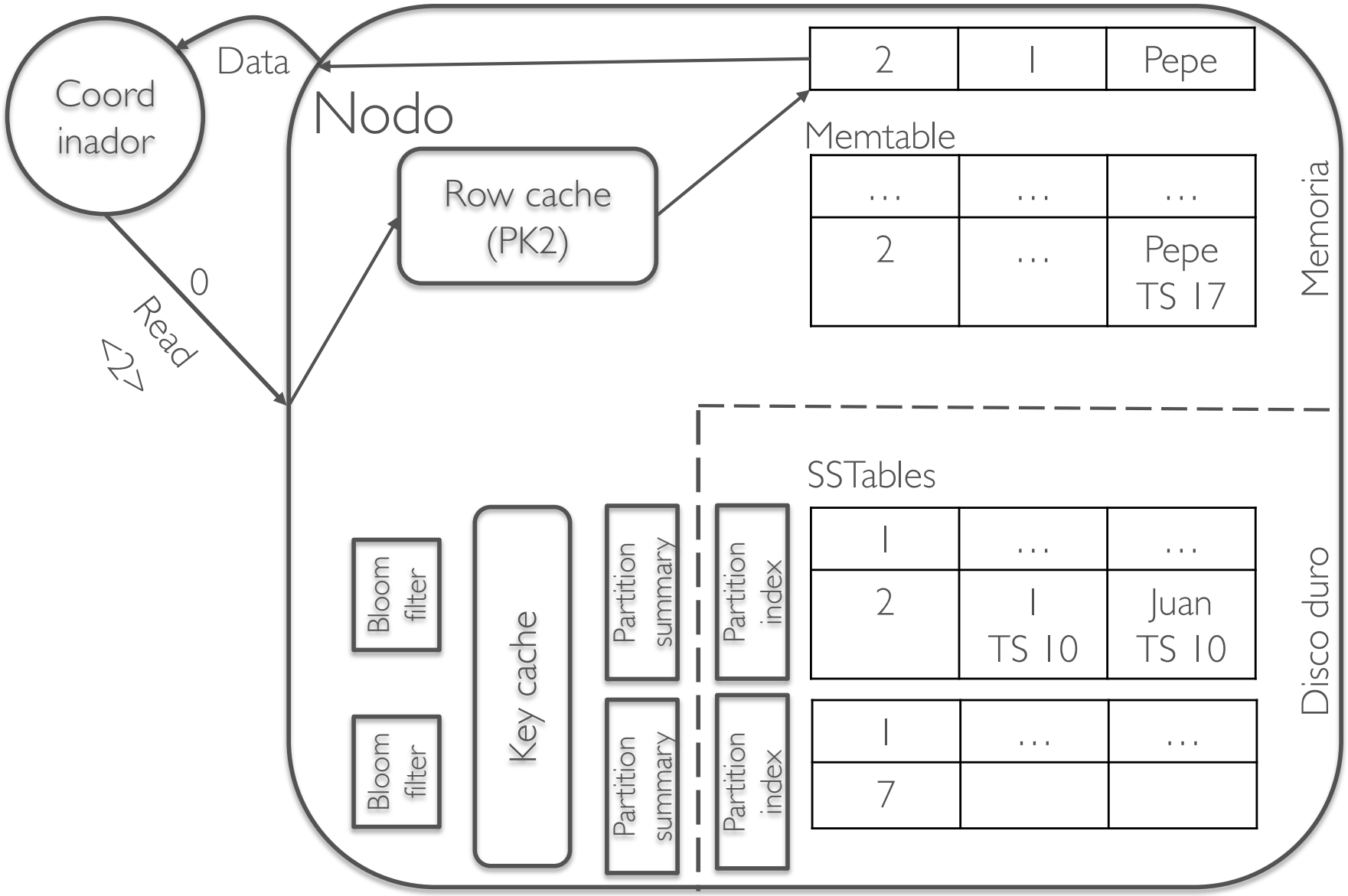
ARQUITECTURA: Read path flow (No cache)



ARQUITECTURA: Read path flow (Key cache)



ARQUITECTURA: Read path flow (Row cache)



ARQUITECTURA: Read path flow

- Row cache:
  - Permite almacenar los resultados de las últimas consultas realizadas de modo que si se consulta de nuevo ese dato devuelve directamente el dato sin necesidad de realizar un merge.
  - Row cache es opcional. Por defecto está desactivado.
  - Se guarda periódicamente el contenido de la cache a disco de modo que si el nodo se reinicia la cache se puede recuperar rápidamente

ARQUITECTURA: Read path flow

- Bloom filter:
  - Es una estructura de datos probabilística que indica si un primary key está o no en su SSTable asociada.
  - Permite reducir el coste de búsqueda.
  - Boom filter dará positivo si cree que la primary key está en su SSTable y negativo si sabe que su SSTable no contiene la primary key.
    - Hay falsos positivos.
    - No hay falsos negativos.
  - Se puede regular el porcentaje de falsos positivos. A menor porcentaje de falsos positivos mayor uso de memoria.

ARQUITECTURA: Read path flow

- Key cache:
  - Permite almacenar las posiciones de las últimas consultas realizadas de modo que si se consulta de nuevo ese dato no es necesario consultar las tablas de particiones para encontrar su posición en las SSTables.
  - Se guarda periódicamente el contenido de la cache a disco de modo que si el nodo se reinicia la cache se puede recuperar rápidamente

ARQUITECTURA: Read path flow

- Partition summary:
  - Es un índice del partition index que permite acceder directamente al segmento donde se encuentra la posición en la SSTable del partition key buscado.
  - Mientras que el partition index se encuentra en disco duro, el partition summary se encuentra en memoria de modo que se mejora el rendimiento del proceso de consulta.
- Partition index
  - Contiene el conjunto completo de posiciones para cada partition key en su SSTable asociada.