Actividad 1. Bases de datos en memoria

DANIELA VIGNAU, Tecnológico de Monterrey, México
CRISTOPHER CEJUDO, Tecnológico de Monterrey, México
HÉCTOR REYES, Tecnológico de Monterrey, México

Este reporte busca profundizar en varios aspectos de las bases de datos en memoria, teniendo como foco el sistema de administración de bases de datos (DBMS) Kinetica. También se analiza el rendimiento de la base de datos en cuestión para las operaciones de inserción y consulta.

ACM Reference Format:
Daniela Vignau, Cristopher Cejudo, and Héctor Reyes. 2021. Actividad 1. Bases de datos en memoria. *J. ACM* 1, 1 (January 2021), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCCIÓN

Kinetica es una base de datos distribuida en memoria que permite ingerir, analizar y visualizar datos simultáneamente. Es un DBMS único en el sentido de que aprovecha el rendimiento de la unidad de procesamiento gráfico (GPU) para realizar operaciones con más rapidez que las bases de datos tradicionales.

Al tratarse de una base de datos orientada a columnas diseñada para el procesamiento analítico (OLAP), Kinetica está optimizada para manejar grandes volúmenes de datos de alta cardinalidad. Por tanto, no es adecuada como sistema de uso transaccional (OLTP). Kinetica organiza los datos de forma estructurada, similar a otras bases de datos relacionales, y los almacena en la memoria RAM o vRAM, en el caso de las GPUs.

2 DESARROLLO

2.1 Arquitectura del DBMS

Kinetica cuenta con una arquitectura distribuida diseñada así para el procesamiento de datos a escala. Un clúster de Kinetica, conformado por nodos idénticos, es capaz de ejecutarse en hardware básico, así como en aquellos equipados con GPU. Uno de esos nodos es seleccionado para ser el nodo de agregación principal. En la Fig. 1 se muestra un diagrama de la arquitectura de Kinetica.

2.2 Tipo de almacenamiento utilizado

La interfaz API nativa a Kinetica para el almacenamiento de datos es de tipo objeto; cada uno de ellos es una fila en la tabla.

Authors' addresses: Daniela Vignau, Tecnológico de Monterrey, México; Cristopher Cejudo, Tecnológico de Monterrey, México; Héctor Reyes, Tecnológico de Monterrey, México.

0004-5411/2021/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

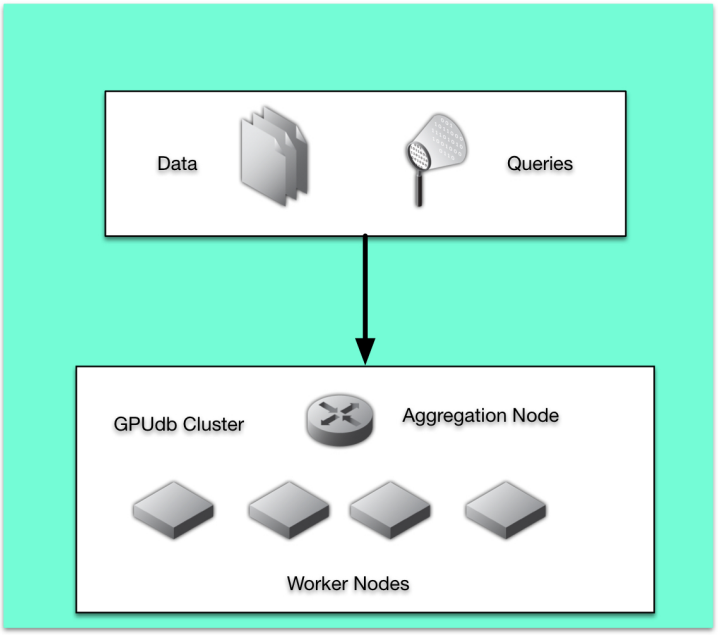

Fig. 1. Arquitectura del DBMS

2.3 Representación en memoria

Dado que el tipo de almacenamiento utilizado por Kinetica es por columnas, la representación en memoria es secuencial. Así, cada registro de una columna va después del otro, e inmediatamente después de terminar esa columna, comienza la siguiente.

2.4 Mecanismos de compresión

Es posible aplicar diferentes mecanismos de compresión a las columnas de las tablas de Kinetica. De manera predeterminada, las columnas son almacenadas sin compresión. En el caso de que se compriman, permanecen en ese estado hasta que son utilizadas. Cuando se recuperan los datos, se realiza una copia temporal de los datos descomprimidos, la cual es descartada una vez que ya no se utiliza.

Kinetica admite cuatro configuraciones de compresión diferentes que varían en velocidad:

- Ninguna (predeterminado)
- `snappy`: Altas velocidades de compresión y descompresión.
- `lz4`: Alta velocidad de compresión, mayor velocidad de descompresión.
- `lz4hc`: Menores velocidades de compresión y descompresión.

2.5 Particionamiento

Nota: ¿Será necesario escribir más al respecto?

Los datos de una tabla pueden ser sometidos a un particionamiento para optimizar el rendimiento y el manejo de datos. Los esquemas de particionamiento permitidos en Kinetica son los siguientes:

- Rango
- Intervalo

- Lista
- Hash
- Series

2.6 Operaciones DML

2.7 Buffer diferencial y el proceso de mezcla

2.8 Reconstrucción de tuplas

2.9 Tipos de *join*

Nota: No estoy muy segura de si está bien esto

Con Kinetica, es posible conectar datos relacionados entre dos o más tablas a través de *join views*. Los tipos de *join* permitidos son:

- INNER
- LEFT
- RIGHT
- FULL OUTER
- Cross

De igual manera, hay dos tipos de ejecución para los *joins*:

- Native joins: Realiza un aislamiento de la operación join y crea una join view nativa de Kinética.
- SQL Joins: Se realiza de manera automática cuando una base de datos recibe una query de SQL con la cláusula JOIN.

2.10 Logging/Recovery

Kinética cuenta con un registro el cual almacena información como: las interacciones de la base de datos, startup/shutdown, información de errores y más. Este registro puede ser configurado para que se almacene otra información que no está por default.

2.11 Respaldos

Nota: No está completo pero YO me encargo de terminarlo

Existe una herramienta para la administración, configuración e instalación de Kinética llamada KAgent, la cual también permite simplificar el proceso de respaldo y

2.12 Manejo de transacciones

2.13 Cold/hot store

2.14 Manejo de datos históricos