# Actividad 1. Bases de datos en memoria

DANIELA VIGNAU (A01021698), Tecnológico de Monterrey, México
CRISTOPHER CEJUDO (A01021698), Tecnológico de Monterrey, México
HÉCTOR REYES (A01339607), 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 *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 (A01021698), Cristopher Cejudo (A01021698), and Héctor Reyes (A01339607). 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 sistema de administración de bases de datos (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

Kinética cuenta con una arquitectura distribuida diseñada así para el procesamiento de datos a escala. Un clúster, compuesto por nodos idénticos, es capaz de correr en hardware básico así como también en aquellos equipados con una unidad de procesamiento de gráficos (GPU, por su siglas en inglés). Uno de esos nodos, es seleccionado para ser el nodo de agregación principal. En la figura 1 se muestra un diagrama sobre ella.

### 2.2 Tipo de almacenamiento utilizado

La interfaz API nativa al sistema para el almacenamiento de datos, es de tipo objetos, con cada uno de estos siendo una fila en la tabla.

Authors' addresses: Daniela Vignau (A01021698), Tecnológico de Monterrey, México; Cristopher Cejudo (A01021698), Tecnológico de Monterrey, México; Héctor Reyes (A01339607), Tecnológico de Monterrey, México.


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

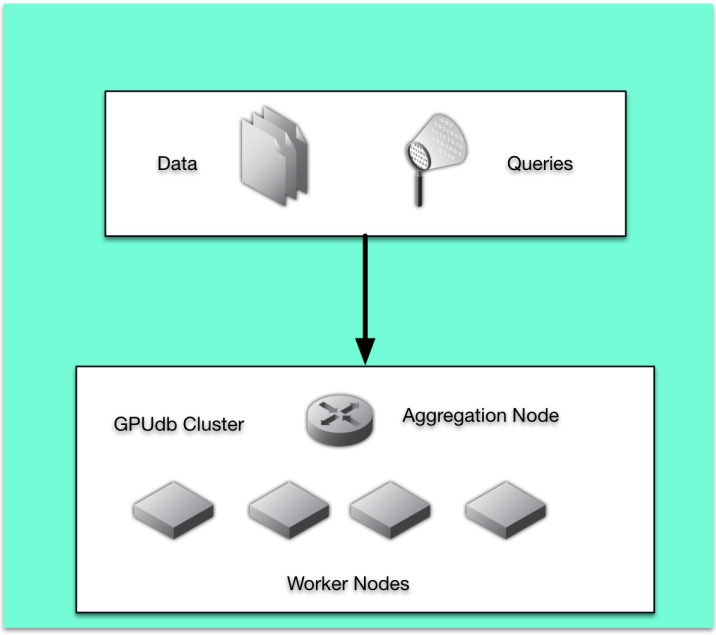


Fig. 1. Arquitectura del DBMS

### 2.3 Representación en memoria

Dado que el tipo de almacenamiento es por columnas, la representación en memoria es de la secuencial, todos los datos de una columna están después del otro, e inmediatamente después de terminar dicha columna, comienza la siguiente.

### 2.4 Mecanismos de compresión

*Nota: No estoy muy segura de si está bien esto*

La compresión puede ser aplicada a una columna de una tabla. Por default, la columna es almacenada sin compresión y una vez que se comprime, se mantiene en ese estado hasta que es usada. Cuando los datos son recuperados, se realiza una copia temporal de los datos descomprimidos y se descarta una vez que se deje de utilizar. Los mecanismos de compresión disponibles en Kinética no están claramente especificados en la documentación, por lo que se puede llegar asumir que se permite cualquier algoritmo.

### 2.5 Particionamiento

*Nota: ¿Será necesario escribir más al respecto?*

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

- 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 joins

*Nota: No estoy muy segura de si está bien esto*

Kinética 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

Existen 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