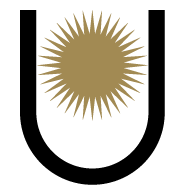
Universidad Nacional del Nordeste



Facultad de Ciencias Exactas y Naturales y Agrimensura

Licenciatura en Sistemas de Información

Base de datos l

Año: 2023

**Índices columnares SQL**

Comisión 1 grupo 5

Alumno: Bournisent, Matias

L.U. N°: 50.434

D.N.I. N°: 39.126.075

Alumno: Comba, Carlos Alfredo

L.U. N°: 44.055

D.N.I. N°: 32.335.478

Alumno:Ramirez Gonzalo Daniel

L.U. N°: 56.838

D.N.I. N°: 44.543.439

Alumno: Zini, Franco Joaquin

L.U. N°: 50.717

D.N.I. N°: 40.049.028

**Índice**

1. Capítulo I: Introducción 3 pág.

2. Marco Teórico 4 pág.

3. Armado de script SQL x pág.

4. Resultados de Pruebas x pág.

**Capítulo I: INTRODUCCIÓN**

En el ámbito de la materia Base de Datos I, se nos asignó la tarea de investigar y experimentar con un componente fundamental del diseño de bases de datos: los índices columnares. Nuestro enfoque se centró en analizar cómo estos índices impactan en el rendimiento del sistema, utilizando un caso de estudio concreto.

Partiendo de un modelo de base de datos proporcionado por la cátedra, llamado "base\_consorcio", nuestra tarea fue duplicar la tabla 'gasto', creando así una nueva tabla denominada 'gastoNew'. La clave de nuestro estudio fue la implementación de índices columnares en esta nueva tabla, lo que nos permitió investigar cómo estos índices aceleran la recuperación de datos.

Ambas tablas se cargaron con un millón de registros para crear un escenario realista de carga de trabajo. Nuestro objetivo era evaluar y comparar las diferencias de rendimiento entre realizar búsquedas en una tabla con índices columnares ('gastoNew') y una sin ellos ('gasto'), para entender la mejora cuantitativa y cualitativa que ofrecen estos índices.

Este informe proporcionará una visión detallada de nuestra metodología, incluyendo los fundamentos teóricos de los índices columnares, los scripts utilizados para modificar las tablas y realizar las inserciones, así como los scripts de búsqueda. Además, se detallará el entorno técnico en el que se llevaron a cabo las pruebas, incluyendo el motor y gestor de la base de datos utilizados, así como las especificaciones de la máquina que alojó nuestras pruebas.

Los resultados de nuestras pruebas incluyen estadísticas precisas sobre la duración de las consultas, el consumo de memoria, el uso de la CPU y el número de filas recuperadas. Estos datos ofrecerán una comprensión profunda y cuantificable del impacto de los índices columnares en el rendimiento del sistema, proporcionando una base sólida para nuestras conclusiones finales.

**Capítulo II: Marco teórico**

En la gestión de bases de datos, el uso de índices columnares optimiza el rendimiento. Son estructuras de datos especializadas que permiten una recuperación eficiente de información en grandes conjuntos de datos. En lugar de buscar a través de todas las filas de una tabla, los índices columnares almacenan y ordenan los valores de una columna específica, actuando como rutas de acceso rápidas hacia los datos que los usuarios buscan.

Cuando se ejecutan consultas en una base de datos, estos índices permiten una búsqueda mucho más rápida y, por ende, reducen significativamente el tiempo necesario para recuperar resultados. Esto se traduce en un rendimiento general del sistema notablemente mejorado, especialmente cuando se trata de operaciones que implican grandes volúmenes de datos.

Cuando se crea un índice columnar para una columna específica, la base de datos organiza los valores de esa columna en una estructura de datos optimizada:

* **Árboles B:** Los índices columnares basados en árboles B son estructuras de datos de tipo árbol que permiten búsquedas eficientes, inserciones y eliminaciones. Estos árboles están diseñados para minimizar la altura del árbol, lo que resulta en tiempos de búsqueda rápidos.
* **Árboles B+:** Son una variante de los árboles B que mejora aún más la eficiencia en las operaciones de búsqueda. Los nodos internos de un árbol B+ contienen solo claves para la navegación, lo que aumenta la capacidad de almacenamiento de las claves y mejora la velocidad de búsqueda.
* **Estructuras de Bitmap:** Los índices columnares basados en estructuras de bitmap son eficaces para columnas que tienen un número limitado de valores únicos (por ejemplo, género o estado civil). Cada valor único se asigna a un conjunto de bits, y las operaciones de búsqueda implican operaciones lógicas en estos conjuntos de bits, lo que permite búsquedas rápidas y eficientes.
* **Índices Hash:** Los índices hash se basan en funciones hash que asignan valores de columna a ubicaciones específicas en una estructura de datos hash. Esto permite búsquedas rápidas cuando se conoce el valor exacto de la columna. Sin embargo, los índices hash no son tan eficientes para operaciones como el rango o las comparaciones de patrones.
* **Árboles Trie:** Estos árboles son útiles para índices columnares en datos de texto o cadenas. Los árboles Trie almacenan información de texto de manera jerárquica, lo que facilita la búsqueda y comparación de palabras o patrones de texto.

La elección de la estructura asociada a un índice columnar depende del tipo de datos en la columna y del tipo de operaciones que se realizarán con esa columna (por ejemplo, búsquedas exactas, búsquedas de rango, etc.). Cada estructura tiene sus ventajas y desventajas, por lo que se selecciona según los requisitos específicos de la aplicación. El motor de base de datos toma la decisión de qué estructura de índice utilizar. Los DBMS están diseñados para ser sistemas inteligentes y autónomos que optimizan el rendimiento de las consultas y operaciones en función de la estructura de los datos y el tipo de consultas que se realizan.

**Armado de Script**

Sobre la base del modelo que entrego la catedra. La estructura de base de datos y dos lotes de carga.

Donde se ejecuta:

1. ModeloDatos\_Consorcio.sql esta script crea la base de datos y sus tablas.
2. BDI\_loteDatosConsorcios.sql permite cargar los datos de provincias, localidad, zona, conserje, administracion, consorcio, tipogasto, gasto
3. BDI\_tabla\_inmueble\_completo.sql, crea y carga la tabla inmueble
4. columnstore\_index\_script.sql crea la tabla gastonew, crea los indice columnares, se insertan 1 millón de registros, y se carga la tabla gasto con los registros faltante para llegar al millón y poder hacer las pruebas comparativas.
5. script\_busqueda.sql tiene los select de busqueda para ambas tablas.