Organización de Archivos y Control de Concurrencia sobre una base de datos relacional de crímenes en Boston

Sebastián Hurtado Diego Linares Piero Marini

Septiembre 26, 2019

1 Introducción

1.1 Objetivo del Proyecto

La meta del presente trabajo es, en una base de datos relacional, desarrollar y probar el funcionamiento de algoritmos de almacenamiento de archivos en memoria secundaria y manejo de concurrencia. Para ello, se hizo uso de un set de data de crímenes en la ciudad de Boston, Massachusets. Sobre esta se implementaron las siguientes tres técnicas de almacenamiento:

- Random File Access
- Static Hashing
- Dynamic Hashing

Se realizaron, por medio de transacciones, pruebas del performance de cada una de las implementaciones para observar las mejoras tanto en tiempo de retribución, como en cantidad de accesos a disco. Se emulo así mismo, el funcionamiento de concurrencia a través de threads.

1.2 Definiciones previas

1.2.1 Organización de Archivos

Definimos a un archivo como una colección de registros que se encuentra almacenado en algún tipo de memoria secundaria (discos magnéticos, cintas, etc.). Debido a que la velocidad de acceso a esta memoria se encuentra limitado, es necesario una forma de recuperar estos datos de forma eficiente. La organización de archivos se refiere a la relación entre los registros que los conforman. El objetivo de este es aumentar la eficiencia en términos de la identificación y acceso a un registro en particular. Dependiendo de la **estructura** en particular utilizada para controlar los registros en forma lógica, es que se obtienen

diferentes beneficios para las operaciones de búsqueda, inserción, actualización y borrado.

1.2.2 Acceso a Archivos

Como resultado de la cantidad de información, esta no se puede encontrar en memoria principal para ser trabajada completamente en simultáneo. Por lo tanto, se tiene que acceder archivos individuales, y esto puede ser realizado en una variedad de maneras:

- **Acceso Secuencial**: forma más común de acceso a memoria en la que se accede a los registros uno detrás de otro. Sin embargo, posee altos costos de lectura debido a que se tiene que recorrer linearmente el archivo. La escritura se puede hacer de forma más sencilla si es que se trabaja con una estructura de tipo *heap*.
- **Acceso Directo**: permite la lectura y escritura rápida de los archivos que no se encuentran en un orden particular. Se accede de forma directa a un bloque en el disco del cual se extrae la data. No tiene restricciones de orden para sus operaciones.
- **Secuencial Indexado**: se construye un índice para el archivo de forma que se puede acceder a cada uno de los bloques de este. Para encontrar un archivo, entramos en el índice a través del cual se puede acceder directamente.

En el proceso de este trabajo repasaremos cada uno de estos métodos en las diferentes organizaciones implementadas.

1.2.3 Hashing

Se refiere al uso de una función de hash para transformar un registro de tamaño arbitrario a un campo de tamaño fijo. La función en cuestión recibe una llave como input que sirve para identificar al registro. El output de la función es un código hash que permite ubicar la data en una tabla de hash que contiene la información correspondiente. Las ventajas que provee se caracterizan tanto por ser en espacio de almacenamiento y en recuperación de data. Por un lado, evita el crecimiento exponencial de espacio de almacenamiento necesario conforme aumenta el número de llaves, mientras que por otro lado los datoos pueden ser obtenidos en un tiempo reducido casi constante - comparado con los tiempos lineares que requieren otras formas de almacenamiento -.

1.3 Descripción del dominio de datos y planteamiento del problema

El dominio de datos que se seleccionó para este proyecto es un resumen de dominio público sobre crímenes ocurridos en la ciudad de Boston entre 2015 y 2018. De acuerdo a la descripción del mismo: *"este contiene registros del nuevo sistema de reporte de incidentes, que incluye un set reducido de campos enfocado en capturar el tipo de incidente, así como cuando y donde ocurrió"*. Con el objetivo de facilitar el trabajo, se ha reducido la tabla para usar los siguientes campos:

Campo	**Tipo de Dato**
Incident Number	`char[10]`
Offense Code	`char[5]`
District	`char[4]`
Reporting Area	`int`
Shooting	`char`
Year	`int`
Month	`int`
Day of Week	`char[10]`
Hour	`int`
UCR Part	`char[20]`
Street	`char[20]`

El problema presente en la base de datos a trabajar es el mal planteamiento de algunos de los campos y las formas de acceso a este a través de un archivo `.csv`. Los datos originales no se encontraban limpios y poseían campos redudantes y de difícil *parsing* que podían ser mejorados. Además, la llave primaria original se encuentra asignada a un valor binario, "shooting", que únicamente podía ser positivo o negativo. Por otra parte, el tener los archivos almacenados en un único `.csv` permitía únicamente técnicas de acceso secuenciales. A pesar de la simplicidad de la implementación, la búsqueda e inserción de nueva data era lenta y requería cargar todo el archivo en memoria primaria.

1.4 Resultados que se esperan obtener

Se esperan obtener mejoras en la performance de las instrucciones de inserción y búsqueda en la base de datos relacional y un control de concurrencia mínimo. Para ello, se están implementando 3 diferentes formas de organización de datos las cuales van a ser comparadas con el mismo proceso en la base de datos original. El control de concurrencia será probado a través de transacciones trabajadas en simultáneo utilizando thread control de UNIX. Se espera que estas realicen una organización serializable que pueda ser aplicada en los diferentes sistemas de organización de archivos.

2 Fundamente y Describa las técnicas