udp Escuela de Informática y Telecomunicaciones

Bases de Datos Avanzada

Actividad 2: Recuperación de Direcciones

Autores:
Nicolás Correa
Joaquín Fernández
David Pazán

Profesor:
Juan Ricardo Giadach

14 de Abril del 2021

Índice

1. Introducción: **2**

2. Desarrollo: **2**
2.1. Diseño y herramientas: 2
2.2. Recuperación de los datos: 3

3. Resultados **4**

4. Análisis generales **4**
4.1. Personas1: 5
4.2. Personas2: 6
4.3. Comparación y/o Contraste de resultados: 8

5. Conclusión **9**

6. Bibliografía **10**

7. Anexado **11**

1. Introducción:

Durante la presente actividad se desarrolla la idea de trabajar con consultas SQL, con el objetivo principal de medir tiempos de búsqueda de las direcciones y encapsulamiento de estas en un nuevo archivo. Todo esto utilizando el mismo hardware que la actividad pasada, con la intención de cuidar la coherencia de los datos entregados.

2. Desarrollo:

La actividad constó de los siguientes pasos:

- Eliminar el índice de la tabla PERSONAS1.
- Recuperar las direcciones de 10 mil datos de dicha tabla, en ruts.
- Repetir el paso anterior ahora en PERSONAS2.
- Una vez obtenidos los tiempos de ejecución, se procede al análisis
- Cálculo de los parámetros de acceso a la base de datos.

2.1. Diseño y herramientas:

El computador que se usó para la actividad tiene las siguientes características y componentes:

- Acer Aspire E-15-575G-76P4
- Sistema Operativo: Elementary Os versión Hera
- Intel core i7-7500U 4 cpus.
- SSD KINGSTON SA400 con capacidad 480gb
- Velocidad para lectura de 550 MB/s y escritura 450 MB/s.

2.2. Recuperación de los datos:

Para utilizar los archivos en la base de datos se necesita que no existan elementos repetidos que correspondan a una llave primaria. En este caso la variable RUT corresponde a una llave primaria, es por esto que para conformar la tabla necesitamos filtrar los datos con los siguientes códigos.

```
#comando para eliminar los datos repetidos del archivo
cat diezmil.txt | sort | uniq > diezmil_fixed.txt
```

Además, al hacer ingreso al archivo se logro percibir un dato que difería del formato planteado para la variable Rut, es por esto que se utilizo el siguiente comando para eliminar el dato. Es importante tener en cuenta que este procedimiento fue netamente posible debido a la cantidad de datos que existían, ya que para un archivo mucho mas grande, no hubiese sido posible este actuar, por lo que buscar una variante para limpiar el archivo seria lo adecuado.

```
#comando que elimina el rut correspondiente al carácter ``V''
#realizando un salto de linea al encontrarlo.
sed '/V/d' -i diezmil
```

Comando que da los permisos necesarios para que todo usuario pueda trabajar con dicho archivo, esto debido a que postgres se toma como un usuario aparte.

```
# comando para cambiar permisos del archivo
chmod 746 savep1.txt
```

En este caso el comando antes descrito utiliza el comando 746, este numero hace referencia a los permisos que otorgara chmod a los diferente usuarios, en este caso:

- Propietario: lectura, escritura y ejecución (rwx)
- Grupo: lectura (r - -)
- Todos: lectura, escritura (rw-)

Se realizó la recuperación de datos, se dio uso de los siguientes comandos (en PERSONAS1 y PERSONAS2 se aplicaron los mismos comandos; solamente variaría el número que acompaña a cada entidad).

```
1 #Creacion de la tabla temporal, para los ruts que se han de ingresar.
2 create temporary table ruts (rut numeric(10));
3 #Se copian los valores del archivo 10mil a la tabla temporal
4 #creada anteriormente.
5 copy ruts from '/home/joaquin/BDDA/BDDA2/diezmil';
6 #Por ultimo se recuperan los valores que han de corresponder
7 #con respecto a PERSONAS1
8 \copy (select personas1.direccion from personas1, ruts
9 where personas1.rut = ruts.rut) to '/tmp/savep1.txt';
```

Por otra parte, se trabajo con el documento de 50 millones de datos, anteriormente usado para rellenar las tablas PERSONAS1 y PERSONAS2, junto a esto, se hizo uso de un nuevo archivo, 10 mil datos, los cuales contienen ruts. En el archivo, hay existencia de ruts no numéricos, estos no son tomados en cuenta para la actividad.

3. Resultados

Teniendo en consideración el uso del filtro descrito en el punto 2.2, se obtuvo como resultado que solo 9999 de los 10000 datos se identificaron dentro de la sentencia "inner join" entre ambas entidades. Los tiempos de ejecución correspondiente a la búsqueda de los 9999 ruts son presentados en forma de promedio continuación.

	Primera Busqueda	Segunda Busqueda	Tercera Busqueda	Promedio
Pesonas 1	120438.030 [ms]	161412.105 [ms]	174094.552 [ms]	151981.5623 [ms]
Pesonas 2	165.452 [ms]	145.598 [ms]	165.981 [ms]	159.0103333 [ms]

Cuadro 1: Valores asociados a la búsqueda en [ms] de Personas1 y Personas2 (referencia Anexada páginas 11 y 12).

4. Análisis generales

Un punto a tener en cuenta previo al desarrollo del análisis, es que tanto el sistema operativo como postgres tienen métodos de optimización de procesos, los cuales provocan que exista una disparidad o variación con las mediciones de tiempo. Cabe mencionar que los valores obtenidos en la tabla (ver Resultados pág 4, Cuadro 1) fueron extraídos tras reiniciar el ordenador reiteradas veces, con la intención de mantener limpio los caché asociados tanto al sistema operativo como la base de datos postgres. Para efectos del presente trabajo se omitirán estas optimizaciones del sistema, con el objetivo de trabajar de mejor manera con los valores empíricos.

Por otra parte, para realizar los análisis generales se utilizo una herramienta de postgres llamada "EXPLAIN", el cual es un comando que muestra el plan de ejecución que genera el planificador de PostgreSQL para la declaración proporcionada. El plan de ejecución muestra cómo se escanearán las tablas a las que hace referencia la declaración (mediante escaneo secuencial simple, escaneo de índice, etc.) y si se hace referencia a varias tablas, qué algoritmos de unión se usarán para reunir las filas requeridas de cada tabla de entrada.

La parte más crítica de la pantalla es el costo estimado de ejecución de la declaración, que es la conjetura del planificador sobre cuánto tiempo tomará ejecutar la declaración.

4.1. Personas1:

Tomando en cuenta el siguiente plan de ejecución:

```
tablas=# explain select personas1.direccion from personas1, ruts where personas1.rut
 = ruts.rut;
                                  QUERY PLAN
------------------------------------------------------------------------------
 Hash Join  (cost=3166687.11..3995700.26 rows=9990 width=101)
   Hash Cond: (ruts.rut = personas1.rut)
   ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=16)
   ->  Hash  (cost=1715231.16..1715231.16 rows=49875516 width=109)
         ->  Seq Scan on personas1  (cost=0.00..1715231.16 rows=49875516 width=109)
(5 filas)

Duración: 5,430 ms
```

Figura 1: Plan de Ejecución en Personas1

Mediante el uso de tablas temporales, el factor de bloqueo se obtuvo con las siguientes expresiones matemáticas:

Se sabe que:

$$T = \frac{F \cdot T_a}{Fb}$$

Para realizar este apartado se considero el "diseño fisico" asociado al apartado de las diapositivas entregadas en cátedra (documento: "BDatosAva01.pdf", página 9); diapositiva en la cual existe un cuadro explicativo referente a los métodos de búsqueda y su uso; en este caso se utilizó por bloques, puesto que no hay un índice asociado a la entidad Personas1. Además se debió tener en cuenta el plan de ejecución, en primer lugar la obtención de accesos, para luego dar con el tiempo total muy similar a lo ya visto en clases. Siguiendo esa lógica, se procede a multiplicar el tiempo de acceso (varía según el tipo de disco que ejecute el proceso SSD 1[ms] o HDD 10[ms]) por el número de filas de la tabla Personas1 y este valor dividirlo por las filas por bloque o factor de bloqueo. Y mediante este valor se busca identificar las coincidencias pertenecientes a la relación entre Personas y ruts.

Despejando Fb se obtiene:

$$Fb = \frac{F \cdot T_a}{T}$$

En donde se tendrían las variables:

- Fb: Factor de Bloqueo.
- F: Numero de filas de personas1 = 49875576
- T_a: Tiempo promedio de acceso = 1 [ms]
- T: Tiempo total de la consulta = 151981.5623 [ms] aproximado a 2 minutos con 53 segundos.

$$Fb = 328{,}1685965$$

Para tener una cifra más exacta se opto por redondear por exceso al valor a 329.

Luego se tiene para un T_n:

$$T_n = \frac{n \cdot T_a}{Fb}$$

Con n correspondiente al número de filas que posee el archivo en caso de este tener otra extensión.

4.2. Personas2:

Tomando en cuenta el siguiente plan de ejecución:

```
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
Nested Loop  (cost=0.56..84953.08 rows=9990 width=101)
  ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=16)
  ->  Index Scan using personas2_pkey on personas2  (cost=0.56..8.49 rows=1 width=109)
        Index Cond: (rut = ruts.rut)
(4 filas)
```

Figura 2: Plan de Ejecución en Personas2

En el caso de Personas2 no fue necesario obtener un factor de bloqueo ya que la tabla que contiene los datos y que actuará como herramienta para la búsqueda, posee índices. Lo que facilitaría el acceso a los datos obtenidos por la consulta SQL.

$$T = (\log_M(F) + 1) \cdot N \cdot T_a$$

Esta expresión viene dada por el mismo documento mencionado en el apartado anterior de Personas1 ("BDatosAva01.pdf", página 9). Basándose en este documento, se utilizo la búsqueda por índices; esta se representa como el logaritmo que tiene como base el orden del árbol y como argumento la cantidad de filas de los datos guardados en Personas2, sumado un 1 haciendo de acceso para la entidad temporal ruts y multiplicada ya la operación por el número de accesos totales que tendría que realizar, estos, siendo el número de filas guardadas en ruts producto con el tiempo de acceso entregado por el disco SSD (1 [ms]).

Luego se debe de despejar M, el cual representa al orden del árbol, quedando la ecuación de la forma:

$$M = \sqrt[\frac{T-N \cdot T_a}{N \cdot T_a}]{F}$$

Donde:

- M: Orden del Árbol.
- F: Numero de filas de personas1 = 49875576
- N: Número total de veces que se repetirá la consulta = 9999
- T_a: Tiempo promedio de acceso = 1 [ms]

- T: Tiempo total de la consulta = 159.0103333 [ms] aproximado a 0.2 segundos.

$$M = 1{,}505626526 \cdot 10^{-8}$$

Aproximado por exceso a $1{,}6 \cdot 10^{-8}$

Ahora para el caso de T_n en donde n pertenece al número de filas de personas2, Se podría estimar un valor en caso de tener otro largo o tamaño, quedando de la siguiente forma:

$$T_n = (\log_{1{,}6 \cdot 10^{-8}}(n) + 1) \cdot 1$$

4.3. Comparación y/o Contraste de resultados:

Tras las cifras obtenidas en la tabla ya mencionada (ver en 3. Resultados. Cuadro 1) se puede apreciar, que la búsqueda por bloques es funcional pero no óptima esto se ve claramente al usar la tabla Personas1 con un valor de tiempo promedio igual a 72107.722 ms (1 minuto con 12 segundos aprox); mientras que en el caso de Personas2 se aplicó una búsqueda asociada a los índices llegando a un valor promedio de 161.4103333 ms (0.16 segundos); puesto que tiene asignada una llave primaria en ruts, en donde la primary key jugará el papel de identificador para recorrer el árbol comprendido por la asociación entre ambas tablas (Personas2 y ruts).

Un caso a tomar en cuenta es el de índices redundantes, donde existen muy escasos datos por ejemplo 1 o 2. En estos casos la búsqueda por índice podría demorar más que la búsqueda por bloque, pero es un caso borde que no debería ocurrir a no ser de ser forzado.

Teniendo esto en cuenta solo queda agregar que en el caso que hubiese sido un archivo con más de 10000 y que todos estos se encuentren asociados a la sentencia "Join"para la entidad Personas1, llegaría a ser mucho más tardío al momento de esperar la respuesta del traspaso al archivo. Por lo que nuevamente no todos los escenarios le favorecen a la búsqueda por bloques, ya que, sería más conveniente usar una asociada a los índices.

5. Conclusión

Tras el análisis de los resultados obtenidos en el laboratorio se puede concluir que aunque el factor de bloqueo mediante búsqueda por bloques es mayor al factor de bloqueo por índices, el método implementado por la tabla indexada logra ser más rápido que su contra parte. También se demostró lo mencionado por el docente al momento de usar el bash de la terminal en contraste a usar un programa externo ya que en el laboratorio anterior el tiempo de filtrado de datos y demases fue mucho mayor de lo esperado; De esta forma se puede inferir que en caso de haber desarrollado un programa externo el tiempo de demora debe de multiplicarse por el número de filas que se desean buscar y a la vez el método solución llega a ser mas demoroso respecto al tiempo dedicado, haciéndolo así menos eficaz.

Cabe mencionar que cumplir con los objetivos de la actividad; no implica que este haya estado exento de dificultades o complicaciones, tal es el caso de los permisos asociados a los usuarios postgres y el usuario del equipo; Por lo que se optó por usar el directorio tmp, ya que al almacenar archivos temporales no tiene asociados usuarios predeterminados de acceso. Pero de todas maneras se le debió conceder el permiso de lectura, escritura y ejecución al archivo para poder trabajarlo.

Por ultimo fue de suma dificultad estimar el cálculo asociado a cada uno de los métodos de búsqueda, los comandos para el filtrado del documento e identificación de atributos repetidos. Pese a los problemas que surgieron, el laboratorio es terminado con éxito y así mismo la actividad.

6. Bibliografía

- Documentación asociada a comando "explain"
  https://www.postgresql.org/docs/9.1/sql-explain.html

- Comandos para permisos de archivos en kernel de linux
  https://www.sololinux.es/uso-del-comando-chmod/

- Factor de Bloqueo información adicional diapositivas: 4, 5, 6.
  http://exa.unne.edu.ar/informatica/programacion1/public_html/archivos/clase_archivo1.pdf

7. Anexado

Anexo relacionado a personas 1:

```
tablas=# \copy (select personas1.direccion from personas1, ruts where personas1.rut
= ruts.rut) to '/tmp/savep1.txt';
COPY 9999
Duración: 120438,030 ms (02:00,438)
```

Figura 3: Primera recuperación de datos Personas1

```
tablas=# \copy (select personas1.direccion from personas1, ruts where personas1.rut
= ruts.rut) to '/tmp/savep1.txt';
COPY 9999
Duración: 161412,105 ms (02:41,412)
```

Figura 4: Segunda recuperación de datos Personas1

```
tablas=# \copy (select personas1.direccion from personas1, ruts where personas1.rut
= ruts.rut) to '/tmp/savep1.txt';
COPY 9999
Duración: 174094,552 ms (02:54,095)
```

Figura 5: Tercera recuperación de datos Personas1

Anexo relacionado a personas 2:

```
tablas=# \copy (select personas2.direccion from personas2, ruts where personas2.rut
= ruts.rut) to '/tmp/savep2.txt';
COPY 9999
Duración: 165,452 ms
```

Figura 6: Primera recuperación de datos Personas2

```
tablas=# \copy (select personas2.direccion from personas2, ruts where personas2.rut
= ruts.rut) to '/tmp/savep2.txt';
COPY 9999
Duración: 145,598 ms
```

Figura 7: Segunda recuperación de datos Personas2

```
tablas=# \copy (select personas2.direccion from personas2, ruts where personas2.rut
= ruts.rut) to '/tmp/savep2.txt';
COPY 9999
Duración: 165,981 ms
```

Figura 8: Tercera recuperación de datos Personas2