UNIVERSIDAD DIEGO PORTALES

FACULTAD DE INGENIERÍA Y CIENCIAS

BASE DE DATOS AVANZADAS

Actividad 1

Nombres: Iván Cáceres
Sebastián Cornejo
Tobías Guerrero
Marcelo Muñoz

Profesor: Juan Giadach
Sección 1

Fase Inicial:

De acuerdo a lo pedido en la actividad 1 de la tarea, a continuación se mostrarán los resultados del ítem 1:

1. Crear la tabla PERSONAS1 con los atributos

RUT	**10 dígitos enteros**
Nombre	**50 caracteres**
Edad	**2 dígitos enteros**
Dirección	**100 caracteres**

USAMOS: *CREATE TABLE PERSONAS1 (RUT numeric(10), Nombre char(50), Edad numeric(2), Direccion char(100));*

```
bddavz=# CREATE TABLE PERSONAS1 (RUT numeric(10), Nombre char(50), Edad numeric(2), Direccion char(100));
CREATE TABLE
Duración: 3.530 ms
```

Figura n°1. Creación de tabla PERSONAS1, con sus atributos.

Se crea la tabla "PERSONAS1" y a continuación se sigue con el ítem 2:

2. Insertar en PERSONAS1 los registros contenidos en el archivo "diezmil" (provisto) y tomar el tiempo de ejecución. Corregir los problemas que haya encontrado.

USAMOS: COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil.csv' DELIMITER '|';

Error encontrado:

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil.csv' DELIMITER '|';
ERROR:  el valor es demasiado largo para el tipo character(50)
CONTEXTO:  COPY personas1, línea 1, columna nombre: « VSISESDCCWHGONGXGUHOPWLUKJNLLFQDPELXDGOHWJILIPKXBC »
```

Figura n°2. Insertar en la tabla PERSONAS1 los datos de diezmil.

Esto se debe a que los campos contienen caracteres ' ' que se encuentran a cada lado del delimitador '|', la solución a esto es modificar el archivo retirando esos caracteres con un programa de Python.

C: > Varios > U > 5TO SEMESTRE > BDDAVZ > bdd_avz.py > ...

```python
f = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil','r')
#f = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50','r')
g = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv','w')
#g = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50_corregido.csv','w')

for linea in f:
    temp = linea.split(' | ')
    escribir = str(temp[0])+'|'+str(temp[1])+'|'+str(temp[2])+'|'+str(temp[3])
    g.write(escribir)
f.close()
g.close()
```

Figura n°3. Código python para arreglar el csv de diezmil.

Una vez corregido este problema se realiza la inserción de las diez mil filas a la tabla PERSONAS1

USAMOS: COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';

1ERA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';
COPY 10000
Duración: 47,827 ms
```

Figura n°4. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

2DA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITE
R '|';
COPY 10000
Duración: 47,928 ms
```

Figura n°5. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

3ERA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';
COPY 10000
Duración: 41,953 ms
```

Figura n°6. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

3. Indexar PERSONAS1 por RUT y tomar el tiempo requerido

USAMOS: CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);

1ERA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 31,736 ms
```

Figura n°7.Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

2DA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 11,450 ms
```

Figura n°8.Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

3ERA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 21,695 ms
```

Figura n°9.Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

4. Crear PERSONAS2 igual que (1) pero con Primary key RUT

USAMOS: CREATE TABLE PERSONAS2 (RUT numeric(10) PRIMARY KEY, Nombre char(50), Edad numeric(2), Direccion char(100));

```
bddavz=# CREATE TABLE PERSONAS2 (RUT numeric(10) PRIMARY KEY, Nombre char(50), Edad numeric(2), Direccion char(100));
CREATE TABLE
Duración: 18,748 ms
```

Figura n°10. Creación de tabla PERSONAS2, con sus atributos, y con su primary key (RUT).

5. Repetir el paso (2) en PERSONAS2

USAMOS: COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';
COPY veintemil(rut) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/veintemil';

1ERA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';
COPY 10000
Duración: 71,959 ms
```

Figura n°11. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

2DA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITE
R '|';
COPY 10000
Duración: 64,703 ms
```

Figura n°12. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

3ERA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv' DELIMITER '|';
COPY 10000
Duración: 59,777 ms
```

Figura n°13. Insertado de diezmil_corregido.csv con su respectivo tiempo de ejecución.

6. Verificar que ambas tablas hayan quedado con el mismo número de filas

USAMOS: select count(*) from personas1;
select count(*) from personas2;

1ERA VEZ

```
bddavz=# select count(*) from personas1;
 count
-------
 10000
(1 fila)
```

Figura n°14. Conteo de datos en la tabla de PERSONAS1.

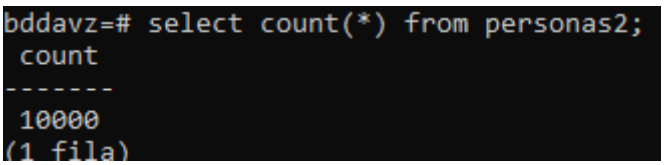


Figura n°15.Conteo de datos en la tabla de PERSONAS2.

2DA VEZ

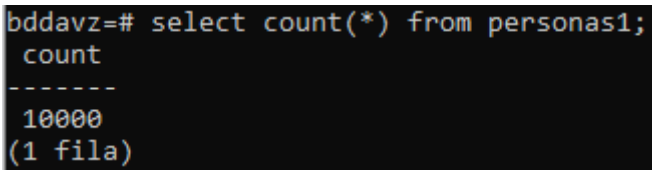


Figura n°16. Conteo de datos en la tabla de PERSONAS1.

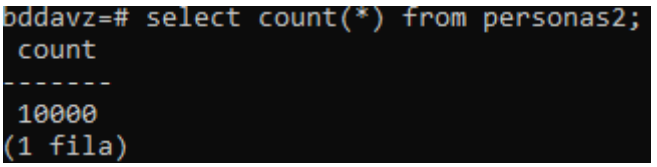


Figura n°17. Conteo de datos en la tabla de PERSONAS2.

3ERA VEZ

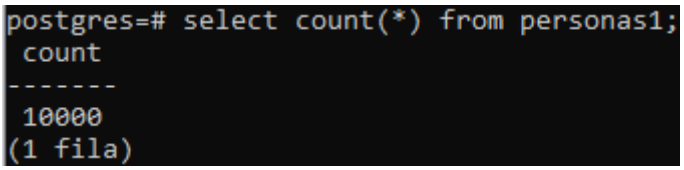


Figura n°18. Conteo de datos en la tabla de PERSONAS1.

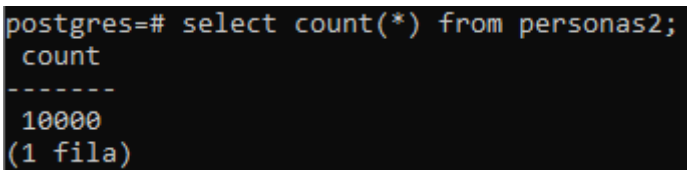


Figura n°19. Conteo de datos en la tabla de PERSONAS2.

7. Comparar los tiempos de ejecución de (2), (3) y (5)

Los tiempos de ejecución son calculados como el promedio de las tres veces que se realizó el proceso, entre cada proceso se borran las tablas personas1 y personas2 y se reinició el pc para liberar la memoria.

EN MS	1era vez	2da vez	3ra vez	PROMEDIO
Tiempo(2)	47,827	47,928	41,953	45,90
Tiempo(3)	31,736	11,45	21,695	21,627
Tiempo(5)	71,959	64,703	59,777	65,48

Tabla n°1. Tiempos de ejecución del Ítem I, III y V, con su promedio respectivo.

Fase Final:

8. Eliminar PERSONAS1 Y PERSONAS2 y repetir todo el proceso para el archivo "millones50"

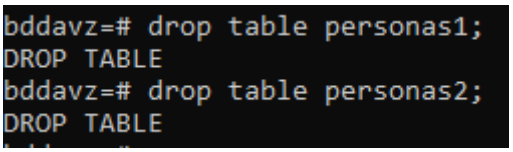


Figura n°20. Borrado de ambas tablas.

Este archivo se generó mediante el comando:
docker run -e name=millones50 -e lines=50000000 -v %cd%:/app/mount/ ensena/bdd-generator, ya que se realizó en windows.

8.1 Crear la tabla PERSONAS1 con los atributos

RUT	**10 dígitos enteros**
Nombre	**50 caracteres**
Edad	**2 dígitos enteros**
Dirección	**100 caracteres**

USAMOS: *CREATE TABLE PERSONAS1 (RUT numeric(10), Nombre char(50), Edad numeric(2), Direccion char(100));*

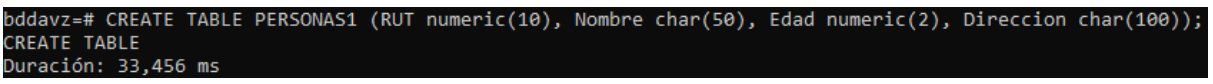


Figura n°21. Creación de tabla PERSONAS1 y con sus atributos.

8.2. Insertar en PERSONAS1 los registros contenidos en el archivo "millones50" y tomar el tiempo de ejecución. Corregir los problemas que haya encontrado.

Aplicamos la misma corrección de la parte 1 pero para este archivo.

C: > Varios > U > 5TO SEMESTRE > BDDAVZ > bdd_avz.py > ...

```python
#f = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil','r')
f = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50','r')
#g = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/diezmil_corregido.csv','w')
g = open ('C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50_corregido.csv','w')

for linea in f:
    temp = linea.split(' | ')
    escribir = str(temp[0])+'|'+str(temp[1])+'|'+str(temp[2])+'|'+str(temp[3])
    g.write(escribir)
f.close()
g.close()
```

Figura n°22.Código python para arreglar el archivo de millones50.

USAMOS: COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50_corregido.csv' DELIMITER '|';

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/millones50_corregido.csv' DELIMITER '|';
COPY 50000000
Duración: 221277,968 ms (03:41,278)
```

Figura n°23. Insertado de millones50_corregido.csv con su respectivo tiempo de ejecución.

8.3. Indexar PERSONAS1 por RUT y tomar el tiempo requerido

USAMOS: CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
ERROR:  no se pudo crear el índice único «indice»
DETALLE:  La llave (rut)=(7888089434) está duplicada.
```

Figura n°24. Indexado de tabla PERSONAS1 por RUT.

Al observar ruts duplicados, se utilizó **python** con una biblioteca llamada "**pandas**" que básicamente se utiliza para manipular y utilizar datos. En este caso se utilizó para tomar el archivo millones50 y se eliminaron las filas cuyos ruts estuvieran duplicados.
También para medir el tiempo en el que este algoritmo se demoraba en tomar, arreglar y recrear los datos se utilizó la biblioteca **time**.

```python
import pandas as pd
import csv

import time

start = time.time()

colsnames = ['rut','nombre','edad','direccion']
data = pd.read_csv('millones50.csv',sep='|', names=colsnames)
data['nombre'] = data['nombre'].str.strip()
data['direccion'] = data['direccion'].str.strip()

print(len(data))
data.drop_duplicates(subset='rut',inplace = True ,keep = 'first')
print(len(data))

data.to_csv('new_archive_50.csv',sep='|',index=False, header=False)

end = time.time()

print("The time of execution of above program is :", end-start)

time.sleep(10)
```

Figura n°25. Código python 2 para arreglar el archivo de millones50.

```
50000000
49875490
The time of execution of above program is : 3629.7368915081024
```

Figura n°26. Número de filas antes/después y tiempo de ejecución.

En total el proceso de arreglo y creación tomó aproximadamente 1 Hora 30 segundos.

Por lo que repetimos desde el paso 1 con la corrección realizada, por lo que se borra la tabla personas 1.

```
bddavz=# drop table personas1;
DROP TABLE
Duración: 230,409 ms
```

Figura n°27. Borrado de tabla PERSONAS1.

8.1.Crear la tabla PERSONAS1

USAMOS: *CREATE TABLE PERSONAS1 (RUT numeric(10), Nombre char(50), Edad numeric(2), Direccion char(100));*

```
bddavz=# CREATE TABLE PERSONAS1 (RUT numeric(10), Nombre char(50), Edad numeric(2), Direccion char(100));
CREATE TABLE
Duración: 13,904 ms
```

Figura n°28. Creación de tabla PERSONAS1, con sus atributos.

8.2. Insertar en PERSONAS1 los registros contenidos en el archivo "millones50" y tomar el tiempo de ejecución.

USAMOS: COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';

1ERA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';
COPY 49875490
Duración: 224846,600 ms (03:44,847)
```

Figura n°29. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

2DA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';
COPY 49875490
Duración: 218395,777 ms (03:38,396)
```

Figura n°30. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

3ERA VEZ

```
bddavz=# COPY personas1(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';
COPY 49875490
Duración: 234496,093 ms (03:54,496)
```

Figura n°31. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

8.3. Indexar PERSONAS1 por RUT y tomar el tiempo requerido

USAMOS: CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);

1ERA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 69883,278 ms (01:09,883)
```

Figura n°32. Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

2DA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 73329,249 ms (01:13,329)
```

Figura n°33. Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

3ERA VEZ

```
bddavz=# CREATE UNIQUE INDEX indice ON PERSONAS1 (rut);
CREATE INDEX
Duración: 74029,505 ms (01:14,030)
```

Figura n°34. Indexado de tabla PERSONAS1 por RUT y con su tiempo de ejecución.

8.4. Crear PERSONAS2 igual que (1) pero con Primary key RUT

USAMOS: CREATE TABLE PERSONAS2 (RUT numeric(10) PRIMARY KEY, Nombre char(50), Edad numeric(2), Direccion char(100));

```
bddavz=# CREATE TABLE PERSONAS2 (RUT numeric(10) PRIMARY KEY, Nombre char(50), Edad numeric(2), Direccion char(100));
CREATE TABLE
Duración: 7,198 ms
```

Figura n°35. Creación de tabla PERSONAS2, con sus atributos, y con su primary key (RUT).

8.5. Repetir el paso (8.2) en PERSONAS2

USAMOS: COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';

1ERA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';
COPY 49875490
Duración: 1835572,438 ms (30:35,572)
```

Figura n°36. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

2DA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|'
COPY 49875490
Duración: 1808695,384 ms (30:08,695)
```

Figura n°37. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

3ERA VEZ

```
bddavz=# COPY personas2(rut,nombre,edad,direccion) FROM 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/new_archive_50.csv' DELIMITER '|';
COPY 49875490
Duración: 1829470,604 ms (30:29,471)
```

Figura n°38. Insertado de new_archivo_50.csv con su respectivo tiempo de ejecución.

8.6. Verificar que ambas tablas hayan quedado con el mismo número de filas

```
bddavz=# select count(*) from personas1;
  count
----------
 49875490
(1 fila)
```

Figura n°39. Conteo de datos en la tabla de PERSONAS1.

```
bddavz=# select count(*) from personas2;
  count
----------
 49875490
(1 fila)
```

Figura n°40. Conteo de datos en la tabla de PERSONAS2.

8.7. Comparar los tiempos de ejecución de (2), (3) y (5)

EN MS	1era vez	2da vez	3ra vez	PROMEDIO
Tiempo (2):	224846,6	218395,777	234496,093	225912,823
Tiempo (3):	69883,278	73329,249	74029,505	72414,01
Tiempo (5):	1835572,438	1808695,384	1829470,604	1824579,475

Tabla n°2. Tiempos de ejecución del Ítem I, III y V, con su promedio respectivo.

Conclusión:

Crear y luego Indexar	Crear con Primary Key
En este proceso en resumen lo que se hace es crear la tabla y posterior a esto se indexa por “rut” según lo pedido. 225912.823+72414.01=298326.833 ms ***aproximadamente 4.9 minutos*** **Se demoró esto porque al ya tener todos los índices en la tabla, haría algún algoritmo para ingresar los índices al B-tree hacerlo de una manera más eficiente.**	En este proceso en resumen lo que se hace es crear la tabla y mientras se crea se añade el parámetro primary key que en este caso es “rut”. 1824579.475 ms ***aproximadamente 30.4 minutos*** **Se demoró esto porque tiene que ingresar mientras se crean los índices al B-tree.** **Por cada dato ingresado se revisan las restricciones asociadas a la primary key, estas son: el campo no sea nulo y que no se encuentre duplicado.**

Se puede observar que para esta cantidad de datos el primer proceso es 6 veces más rápido que el segundo, también es posible observar que en el caso de los 10 mil, la diferencia entre métodos (ms) es mínima.
En conclusión es posible observar que crear y luego indexar por rut es mucho más rápido que crear la tabla con primary key.

tarea 2

quitamos el índice de personas1

```
bddavz=# drop index indice;
DROP INDEX
```

Creamos la tabla veintemil.
Usamos: CREATE TABLE veintemil(rut numeric(10) primary key);

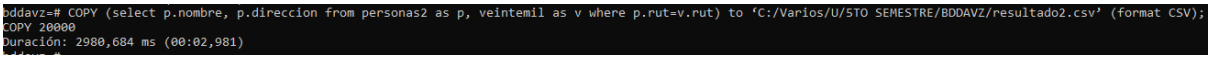
Usamos: COPY veintemil(rut) from ‘C:/Varios/U/5TO SEMESTRE/BDDAVZ/veintemil’;

```
bddavz=# COPY veintemil(rut) FROM ‘C:/Varios/U/5TO SEMESTRE/BDDAVZ/veintemil’;
COPY 20000
Duración: 84,536 ms
```

REALIZAMOS LA CONSULTA PARA PERSONAS1.
USAMOS: COPY (select p.nombre, p.direccion from personas1 as p, veintemil as v where p.rut=v.rut) to ‘C:/Varios/U/5TO SEMESTRE/BDDAVZ/resultado.csv’ (format CSV);

```
bddavz=# COPY (select p.nombre, p.direccion from personas1 as p, veintemil as v where p.rut=v.rut) to ‘C:/Varios/U/5TO SEMESTRE/BDDAVZ/resultado.csv’ (format CSV);
COPY 20000
Duración: 9810,274 ms (00:09,810)
```

A CONTINUACIÓN PARA PERSONAS2.
USAMOS: COPY (select p.nombre, p.direccion from personas2 as p, veintemil as v where p.rut=v.rut) to 'C:/Varios/U/5TO SEMESTRE/BDDAVZ/resultado2.csv' (format CSV);



TIEMPO PARA PERSONAS1: 9810,274 ms = 9,810 segundos
TIEMPO PARA PERSONAS2: 2980,684 ms = 2,981 segundos

EXPLICACIÓN DE LOS TIEMPOS, PORQUE UNO ES MÁS RÁPIDO….

CALCULAR FILAS POR BLOQUE PARA PERSONAS1 Y ORDEN DEL ÁRBOL PARA PERSONAS2

t1 = (n/fb) * ta = (49875490/fb) * 1ms = 9810.274ms
=> 49875490 = 9810.274 * fb
=> fb = 5084 filas por bloque
t2= (log m 49875490 + 1)*20000*1ms = 2980,684 ms
=>m=1.12719 -> =2?

(formulas que puso el profe en clase xd)

6. Calcular el tiempo de respuesta de las dos consultas realizadas.
Asuma que los parámetros son fb = 50 y m = 300. Para el tiempo
de acceso asuma 10 ms en el caso de un HDD y 1 ms para un SSD.

T1: (N/fb)*ta = (49875490/50)*1ms=997509.8 ms = 997.5098s
T2: (log300 49875490 + 1)*20000*1ms = 82151.88022ms = 82.151s

(formulas que puso el profe en clase xd)

7. Finalmente, compare los resultados obtenidos empíricamente
con los calculados teóricamente.

Resultado t1 teórico:
Resultado t1 empírico:
Resultado t2 teórico:
Resultado t2 empírico:

Conclusiones.

Es posible observar que