udp Escuela de Informática y Telecomunicaciones

Bases de Datos Avanzada

Actividad 3: Eliminación de Datos

Autores:
Nicolás Correa
Joaquín Fernández
David Pazán

Profesor:
Juan Ricardo Giadach

4 de mayo del 2021

Índice

1. Introducción:

En el presente informe se desarrollara la entrega numero 4 de la asignatura bases de datos avanzadas, donde tomando en cuanta el desarrollo de las actividades pasadas se procederá a la eliminación de datos en las diferentes tablas. Esto con la intención de captar tiempos de ejecución para un posterior análisis comparativo.

2. Desarrollo:

2.1. Herramientas

El computador que se utilizo tanto para las actividades pasadas como para esta posee las siguientes características y componentes:

- Acer Aspire E-15-575G-76P4
- Sistema Operativo: Linux, con distribución 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.

Dicho hardware se mantiene en el desarrollo de los trabajos con la intención de tener datos relacionados entre trabajos, ya que el tiempo de las consultas medidas depende del equipo en el cual se ejecuta.

Para el desarrollo de la actividad se debe utilizar distintas herramientas, como los comandos proporcionadas por Postgres, llamados BEGIN y ROLLBACK. Para definir estos comandos es importante manejar el concepto en bases de datos llamado transacción, esta se refiere a funciones que empaquetan varios pasos en una operación, de forma que se cumplan todos o ninguno. En el caso de que ocurra algún fallo que impida que se complete la transacción, ninguno de los pasos se ejecutan y no afectan a los objetos de la base de datos.

Con esto presente, BEGIN permite comenzar una transacción, mientras que con ROLLBACK después de haber realizado y confirmado una transacción, PostgreSQL permite anular dicha transacción de forma que no se modifique los datos de nuestra base de datos.

Por otro lado el comando EXPLAIN será quien da a conocer el plan de ejecución de Postgres, para las consultas que se le proporcione. Este 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.

¿Por qué es necesario usar "explain" en la actividad?

Ya que entre sus utilidades esta, la entrega de un costo estimado adherido a un plan de ejecución lo que es la suposición del planificador sobre el tiempo que tomará correr la consulta (medido en páginas de disco). Además ofrece una descripción del vinculo que comparten las entidades involucradas en las consultas tal es el caso de "Hashz "Hash Join". O también el tipo de recorrido que tendría que aplicar respecto al número de iteraciones incluyendo así la cantidad de filas inspeccionadas.

2.2. Plan de ejecución

La actividad consta de los siguientes puntos:

1. Eliminar el índice de PERSONAS1 (si aun lo tuviera) y confirmar que PERSONAS2 tenga su llave primaria
2. En PERSONAS1, eliminar los registros asociados a los RUT indicados midiendo el tiempo que demora el proceso
3. Repetir el mismo proceso en PERSONAS2
4. Analizar los tiempos obtenidos e indicar si hay algún método más eficiente para realizar la eliminación

Con respecto al punto numero 1 de la actividad, de antemano se sabe que Personas 1 no tiene índice y de que Personas 2 tiene su llave primaria, esto debido al procedimiento llevado en las anteriores actividades. De todas maneras es posible revisar el estado de las tablas con el comando $\backslash d$, el cual proporciona la descripción de una tabla, siendo esta información compuesta por las columnas, sus tipos, el espacio de tabla (si no es el predeterminado) y cualquier atributo especial como puede ser una variable definida como NO NULO o valores predeterminados. También se muestran índices, restricciones, reglas y activadores asociados.

```
tablas=# \d personas1
                    Tabla «public.personas1»
  Columna  |      Tipo      | Ordenamiento | Nulable | Por omisión
-----------+----------------+--------------+---------+-------------
 rut       | numeric(10,0)  |              |         |
 nombre    | character(50)  |              |         |
 edad      | numeric(2,0)   |              |         |
 direccion | character(100) |              |         |
```

Figura 1: Comprobación de la NO existencia de índice para Personas 1.

```
tablas=# \d personas2
                    Tabla «public.personas2»
  Columna  |      Tipo      | Ordenamiento | Nulable  | Por omisión
-----------+----------------+--------------+----------+-------------
 rut       | numeric(10,0)  |              | not null |
 nombre    | character(50)  |              |          |
 edad      | numeric(2,0)   |              |          |
 direccion | character(100) |              |          |
Índices:
    "personas2_pkey" PRIMARY KEY, btree (rut)
```

Figura 2: Comprobación de la existencia de llave primaria para Personas 2.

Los puntos 2, 3 y 4 se llevan a cabo utilizando los comandos BEGIN y ROLLBACK para evitar eliminar datos, y así generar otras querys sin necesidad de volver a cargar dichos datos. Además evaluando cada query 3 veces sera calculado un promedio de estas con la intención de tener datos lo más íntegros posibles.

En esta ecuación se evaluaran 3 sentencias distintas conceptualmente, pero que tienen el mismo fin que es eliminar los datos exigidos, esto para tener otra perspectiva de la consulta.

Luego de calcular los promedios para cada una de las 3 querys, se realizara un análisis comparativo.

Un punto a tener en cuenta es que para realizar cada una de las consultas se debe reiniciar el ordenador del usuario, esto como alternativa para vaciar las tablas de cache que tienen tanto postgres como el sistema operativo. Se debe recordar de la actividad pasada que tanto Postgres como el sistema operativo tienen sus propias formas de acelerar las ejecuciones de las consultas, lo cual se escapa de las manos del usuario, es por esto que con esta practica se espera tener datos un poco más íntegros.

2.2.1. Plan de ejecución para cada consulta

Las consultas siguen un patrón de ejecución perteneciente a la aplicación de bases de datos. Estas ejecuciones se evaluaran a continuación con el comando EXPLAIN:

```
tablas=# explain delete from personas1 using ruts where personas1.rut = ruts.rut;
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
 Delete on personas1  (cost=2583244.29..2827896.44 rows=9990 width=12)
   ->  Hash Join  (cost=2583244.29..2827896.44 rows=9990 width=12)
         Hash Cond: (ruts.rut = personas1.rut)
         ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
         ->  Hash  (cost=1715919.24..1715919.24 rows=49895524 width=14)
               ->  Seq Scan on personas1  (cost=0.00..1715919.24 rows=49895524 width=14)
(6 filas)

Duración: 5,886 ms
```

Figura 3: EXPLAIN a consulta numero 1 para Personas 1

```
tablas=# explain delete from personas1 where personas1.rut in (select rut from ruts);
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
 Delete on personas1  (cost=183.38..2124622.22 rows=24947762 width=12)
   ->  Hash Join  (cost=183.38..2124622.22 rows=24947762 width=12)
         Hash Cond: (personas1.rut = ruts.rut)
         ->  Seq Scan on personas1  (cost=0.00..1715919.24 rows=49895524 width=14)
         ->  Hash  (cost=180.88..180.88 rows=200 width=22)
               ->  HashAggregate  (cost=178.88..180.88 rows=200 width=22)
                     Group Key: ruts.rut
                     ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
(8 filas)

Duración: 0,572 ms
```

Figura 4: EXPLAIN a consulta numero 2 para Personas 1

```
tablas=# explain delete from personas1 where exists (select from ruts where personas1.rut=ruts.rut);
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
 Delete on personas1  (cost=183.38..2124622.22 rows=24947762 width=12)
   ->  Hash Join  (cost=183.38..2124622.22 rows=24947762 width=12)
         Hash Cond: (personas1.rut = ruts.rut)
         ->  Seq Scan on personas1  (cost=0.00..1715919.24 rows=49895524 width=14)
         ->  Hash  (cost=180.88..180.88 rows=200 width=22)
               ->  HashAggregate  (cost=178.88..180.88 rows=200 width=22)
                     Group Key: ruts.rut
                     ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
(8 filas)

Duración: 0,687 ms
```

Figura 5: EXPLAIN a consulta numero 3 para Personas 1

Como se puede apreciar en los planes de acción en las figuras 3, 4 y 5, Aunque la query tengan el mismo fin (eliminan los mismos datos), el plan de ejecución que genera el gestor de bases de datos es ligeramente distinto. Sabiendo que el plan de acción del método 1 comenzará

con un Seq Scan, esto traería como consecuencia una menor cantidad de datos a manejar en pasos siguientes (sabiendo de que ruts tiene menor cantidad de filas que personas1), explicando así el menor tiempo de ejecución.

```
tablas=# explain delete from personas2 using ruts where personas2.rut = ruts.rut;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------
 Delete on personas2  (cost=0.56..84953.08 rows=9990 width=12)
   ->  Nested Loop  (cost=0.56..84953.08 rows=9990 width=12)
         ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
         ->  Index Scan using personas2_pkey on personas2  (cost=0.56..8.49 rows=1 width=14)
               Index Cond: (rut = ruts.rut)
(5 filas)

Duración: 5,461 ms
```

Figura 6: EXPLAIN a consulta numero 1 para Personas 2

```
tablas=# explain delete from personas2 where personas2.rut in (select rut from ruts);
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------
 Delete on personas2  (cost=179.44..1630.79 rows=24937758 width=12)
   ->  Nested Loop  (cost=179.44..1630.79 rows=24937758 width=12)
         ->  HashAggregate  (cost=178.88..180.88 rows=200 width=22)
               Group Key: ruts.rut
               ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
         ->  Index Scan using personas2_pkey on personas2  (cost=0.56..8.58 rows=1 width=14)
               Index Cond: (rut = ruts.rut)
(7 filas)

Duración: 1,253 ms
```

Figura 7: EXPLAIN a consulta numero 2 para Personas 2

```
tablas=# explain delete from personas2 where exists (select from ruts where personas2.rut=ruts.rut);
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------
 Delete on personas2  (cost=179.44..1630.79 rows=24937758 width=12)
   ->  Nested Loop  (cost=179.44..1630.79 rows=24937758 width=12)
         ->  HashAggregate  (cost=178.88..180.88 rows=200 width=22)
               Group Key: ruts.rut
               ->  Seq Scan on ruts  (cost=0.00..153.90 rows=9990 width=22)
         ->  Index Scan using personas2_pkey on personas2  (cost=0.56..8.58 rows=1 width=14)
               Index Cond: (rut = ruts.rut)
(7 filas)

Duración: 0,875 ms
```

Figura 8: EXPLAIN a consulta numero 3 para Personas 2

Como se puede ver en el segundo recuadro, perteneciente a figuras 6, 7 y 8, el proceso que ocurre en la tabla Personas 2, con llave primaria en rut y consultas similares, ha de generar un procedimiento ligeramente distinto, es decir, para la primera consulta, no hace uso de

HashAggregate, el cual se encarga para las siguientes consultas de generar una tabla de hash temporal, la cual ha de agrupar los registros y facilitar la eliminación de estos de forma rápida a través de la aplicación de índices.

2.3. Borrado de datos

2.3.1. Personas 1

Query 1: Se evalúa la siguiente consulta encargada de borrar registros a partir de las igualdades encontradas en la tabla ruts; con los atributos implicados en la relación. Personas 1 no posee una llave primaria.

```
begin;
delete from personas1 using ruts where personas1.rut = ruts.rut;
rollback;
```

Query 2: Se realiza otro borrado mediante la siguiente consulta anidada, la cual ha de comparar los ruts seleccionados en la tabla principal, Personas 1, y posteriormente los rut de la tabla ruts

```
begin;
delete from personas1 where personas1.rut in (select rut from ruts);
rollback;
```

Query 3: El ultimo borrado se realiza si existe la relación entre los atributos rut de Personas1 y rut de la tabla ruts. A través de una consulta anidada que gestiona la búsqueda.

```
begin;
delete from personas1 where exists (select from ruts where personas1.rut = ruts.
    rut);
rollback;
```

2.3.2. Personas 2

Query 1: Se evalúa la siguiente consulta con la intención de borrar registros a partir de las igualdades encontradas en la tabla ruts; con los atributos implicados en la relación. Personas 2 posee a rut como llave primaria.

```
begin;
delete from personas2 using ruts where personas2.rut = ruts.rut;
rollback;
```

Query 2: Se realiza otro borrado, mediante la siguiente consulta anidada, la cual ha de comparar los ruts seleccionados en la tabla principal Personas 2, y posteriormente los rut de la tabla ruts.

```
begin;
delete from personas2 where personas2.rut in (select rut from ruts);
rollback;
```

Query 3: El ultimo borrado se realiza si existe la relación entre los atributos rut de Personas2 y rut de la tabla ruts. A través de una consulta anidada que gestiona la búsqueda.

```
1 begin;
2 delete from personas2 where exists (select from ruts where personas2.rut = ruts.
      rut);
3 rollback;
```

3. Resultados

Query 1	Primer intento	Segundo intento	Tercer intento	Promedio
Pesonas 1	50549,172 [ms]	47444,824 [ms]	45908,953 [ms]	47.9676 [s]
Pesonas 2	8392,544[ms]	8746,464 [ms]	8617,267[ms]	8.5854 [s]

Cuadro 1: Cuadro comparativo de tiempos, Query 1

Query 2	Primer intento	Segundo intento	Tercer intento	Promedio
Pesonas 1	34911,708 [ms]	34456,149 [ms]	34584,965 [ms]	34.6509 [s]
Pesonas 2	9516,084[ms]	7650,657 [ms]	10115,839[ms]	9.0942 [s]

Cuadro 2: Cuadro comparativo de tiempos, Query 2

Query 3	Primer intento	Segundo intento	Tercer intento	Promedio
Pesonas 1	34716,665 [ms]	34412,028 [ms]	34387,494 [ms]	34.5054[s]
Pesonas 2	9856,999 [ms]	8881,540 [ms]	9233,072[ms]	9.3239 [s]

Cuadro 3: Cuadro comparativo de tiempos, Query 3

El tiempo asociado a las consultas se trabaja en milisegundos mientras que los promedios están en horas, minutos y segundos según corresponda.

4. Análisis generales

4.1. Tabla comparativa

Dentro de esta actividad fueron implementados tres métodos distintos para llevar al cabo la eliminación de datos, estos datos serán ingresados en una tabla comparativa a continuación.

	Query 1	Query 2	Query 3
Pesonas 1	47.9676 [s]	34.6509 [s]	34.5054 [s]
Pesonas 2	8.5854 [s]	9.0942 [s]	9.3239 [s]

Cuadro 4: Cuadro comparativo respecto a los promedios de las Query

La tabla logra evidenciar por una parte los tiempos de las diferentes querys(1, 2 y 3) para una tabla con y sin índices como lo son Personas 1 y Personas 2 respectivamente.

Es una constante la diferencia de tiempo, en Personas 1 la eliminación es mucho más lenta que en Personas 2, sin importar la query que se haya utilizado.

Recordando experiencias anteriores, Personas1 tiene una búsqueda secuencial de datos, mientras que personas2, puede generar búsquedas mucho más rápidas gracias al árbol creado a partir del índice. Esta es la principal razón por la cual las eliminaciones en Personas2 son más rápidas, el dato debe ser buscado primero antes de ser eliminado.

A través de las capturas obtenidas en la actividad de planes de ejecución (ver Plan de Ejecución páginas 6 y 7), se puede ver que la Query2 y la Query3 tienen resultados en tiempos y estructuras muy similares, por lo que ya se encuentran optimizando ambas los recursos dados por el join hecho de forma implícita en la Consulta 1. Ya que el par al ser del tipo anidada o subquery reemplazan la complejidad de la consulta inicial a través de una combinación entre dos consultas, simplificando la complejidad de ejecución de estas, haciendo más eficiente el tiempo de recuperación de la información en la bases de datos.

Esto se puede identificar, observando los hash que se vinculan así mismos con la query interna (anidada) y esta a la vez realiza un hash de forma secuencial de los resultados generando un valor de filas por bloque o factor de bloqueo (ver Plan de Ejecución páginas 6 y 7, Figuras 4 y 5) lo cual optimiza el numero de accesos, disminuyendo la carga al servidor base de datos, disminuyendo así el tiempo de búsqueda de los datos para su posterior eliminación.

5. Conclusión

De acuerdo con los datos proporcionados y demostrados, se puede apreciar que las consultas realizadas en personas 2 fueron siempre más óptimas en tiempo de espera que en personas 1, la característica principal por lo que esto sucedió fue por la existencia de índices en personas 2.

Además, tomando en cuenta los tiempos de ejecución vistos en las tablas, la query que proporciono los menores tiempos al momento de trabajar la eliminación de datos, ha de ser la tercera query teniendo muy de cerca a la segunda. Mientras que la peor es la primera, demostrando que en este caso el uso de una sentencia anidada puede agilizar el proceso de búsqueda de la sentencia para estos escenarios.

Finalmente teniendo presente que los datos se eliminaron de forma correcta para todos los casos propuestos, se concluye con la realización exitosa de los objetivos de la actividad.

6. Bibliografía

- Comandos postgres, BEGIN y ROLLBACK
  https://www.geeksforgeeks.org/postgresql-rollback/

- Comandos postgres, para información de tablas
  https://www.todopostgresql.com/meta-commands-de-psql-mas-utilizados-en-postgresql/

- Comandos postgres, EXPLAIN
  https://www.postgresql.org/docs/9.1/sql-explain.html

- Información sobre plan de ejecución
  https://www.gpsos.es/2020/04/plan-de-ejecucion-en-postgresql-como-se-obtiene/

- Querys y Subquerys, ventajas y desventajas
  https://www.geeksforgeeks.org/sql-join-vs-subquery/

7. Anexado

7.1. Anexado de tiempos para Personas 1:

Query 1:

```
tablas=# begin;
BEGIN
Duración: 0,300 ms
tablas=*# delete from personas1 using ruts where personas1.rut = ruts.rut;
DELETE 9999
Duración: 50549,172 ms (00:50,549)
tablas=*# rollback;
ROLLBACK
Duración: 0,650 ms
```

Figura 9: Primera ejecución de Query 1 para personas 1

```
BEGIN
Duración: 0,283 ms
tablas=*# delete from personas1 using ruts where personas1.rut = ruts.rut;
DELETE 9999
Duración: 47444,824 ms (00:47,445)
tablas=*# rollback;
ROLLBACK
Duración: 0,316 ms
```

Figura 10: Segunda ejecución de Query 1 para personas 1

```
tablas=# begin;
BEGIN
Duración: 0,420 ms
tablas=*# delete from personas1 using ruts where personas1.rut = ruts.rut;
DELETE 9999
Duración: 45908,953 ms (00:45,909)
tablas=*# rollback;
ROLLBACK
Duración: 0,580 ms
```

Figura 11: Tercera ejecución de Query 1 para personas 1

Query 2:

```
BEGIN
Duración: 0,322 ms
tablas=*# delete from personas1 where personas1.rut in (select rut from ruts);
DELETE 9999
Duración: 34911,708 ms (00:34,912)
tablas=*# rollback;
ROLLBACK
Duración: 0,733 ms
```

Figura 12: Primera ejecución de Query 2 para personas 1

```
BEGIN
Duración: 0,334 ms
tablas=*# delete from personas1 where personas1.rut in (select rut from ruts);
DELETE 9999
Duración: 34456,149 ms (00:34,456)
tablas=*# rollback;
ROLLBACK
Duración: 0,826 ms
```

Figura 13: Segunda ejecución de Query 2 para personas 1

```
BEGIN
Duración: 0,319 ms
tablas=*# delete from personas1 where personas1.rut in (select rut from ruts);
DELETE 9999
Duración: 34584,965 ms (00:34,585)
tablas=*# rollback;
ROLLBACK
Duración: 0,668 ms
```

Figura 14: Tercera ejecución de Query 2 para personas 1

Query 3:

```
BEGIN
Duración: 0,305 ms
tablas=*# delete from personas1 where exists (select from ruts where personas1.rut = ruts
.rut);
DELETE 9999
Duración: 34716,665 ms (00:34,717)
tablas=*# rollback;
ROLLBACK
Duración: 0,674 ms
```

Figura 15: Primera ejecución de Query 3 para personas 1

```
BEGIN
Duración: 0,318 ms
tablas=*# delete from personas1 where exists (select from ruts where personas1.rut = ruts
.rut);
DELETE 9999
Duración: 34412,028 ms (00:34,412)
tablas=*# rollback;
ROLLBACK
Duración: 0,644 ms
```

Figura 16: Segunda ejecución de Query 3 para personas 1

```
BEGIN
Duración: 0,375 ms
tablas=*# delete from personas1 where exists (select from ruts where personas1.rut = ruts
.rut);
DELETE 9999
Duración: 34387,494 ms (00:34,387)
tablas=*# rollback;
ROLLBACK
Duración: 0,646 ms
```

Figura 17: Tercera ejecución de Query 3 para personas 1

Anexado de tiempos para Personas 2: Query 1:

```
BEGIN
Duración: 0,380 ms
tablas=*# delete from personas2 using ruts where personas2.rut = ruts.rut;
DELETE 9999
Duración: 8392,544 ms (00:08,393)
tablas=*# rollback;
ROLLBACK
Duración: 0,244 ms
```

Figura 18: Primera ejecución de Query 1 para personas 2

```
BEGIN
Duración: 0,442 ms
tablas=*# delete from personas2 using ruts where personas2.rut = ruts.rut;
DELETE 9999
Duración: 8746,464 ms (00:08,746)
tablas=*# rollback;
ROLLBACK
Duración: 0,514 ms
```

Figura 19: Segunda ejecución de Query 1 para personas 2

```
BEGIN
Duración: 0,315 ms
tablas=*# delete from personas2 using ruts where personas2.rut = ruts.rut;
DELETE 9999
Duración: 8617,267 ms (00:08,617)
tablas=*# rollback;
ROLLBACK
Duración: 0,638 ms
```

Figura 20: Tercera ejecución de Query 1 para personas 2

Query 2:

```
BEGIN
Duración: 0,319 ms
tablas=*# delete from personas2 where personas2.rut in (select rut from ruts);
DELETE 9999
Duración: 9516,084 ms (00:09,516)
tablas=*# rollback;
ROLLBACK
Duración: 0,603 ms
```

Figura 21: Primera ejecución de Query 2 para personas 2

```
BEGIN
Duración: 0,317 ms
tablas=*# delete from personas2 where personas2.rut in (select rut from ruts);
DELETE 9999
Duración: 7650,657 ms (00:07,651)
tablas=*# rollback;
ROLLBACK
Duración: 0,555 ms
```

Figura 22: Segunda ejecución de Query 2 para personas 2

```
BEGIN
Duración: 0,293 ms
tablas=*# delete from personas2 where personas2.rut in (select rut from ruts);
DELETE 9999
Duración: 10115,839 ms (00:10,116)
tablas=*# rollback;
ROLLBACK
Duración: 0,612 ms
```

Figura 23: Tercera ejecución de Query 2 para personas 2

Query 3:

```
BEGIN
Duración: 0,321 ms
tablas=*# delete from personas2 where exists (select from ruts where personas2.rut = ruts
.rut);
DELETE 9999
Duración: 9856,999 ms (00:09,857)
tablas=*# rollback;
ROLLBACK
Duración: 0,656 ms
```

Figura 24: Primera ejecución de Query 3 para personas 2

```
BEGIN
Duración: 0,167 ms
tablas=*# delete from personas2 where exists (select from ruts where personas2.rut = ruts
.rut);
DELETE 9999
Duración: 8881,540 ms (00:08,882)
tablas=*# rollback;
ROLLBACK
Duración: 0,633 ms
```

Figura 25: Segunda ejecución de Query 3 para personas 2

```
BEGIN
Duración: 0,320 ms
tablas=*# delete from personas2 where exists (select from ruts where personas2.rut = ruts
.rut);
DELETE 9999
Duración: 9233,072 ms (00:09,233)
tablas=*# rollback;
ROLLBACK
Duración: 0,624 ms
```

Figura 26: Tercera ejecución de Query 3 para personas 2