udp Escuela de Informática y Telecomunicaciones

Bases de Datos Avanzada

Actividad 8: Optimización de Bases de Datos

Autores:
Nicolás Correa
Joaquín Fernández
David Pazán

Profesor:
Juan Ricardo Giadach

junio 8 del 2021

Índice

1. Introducción

En las de bases de datos relacionales existen herramientas que ayudan al usuario a manejar de mejor manera su sistema, logrando mayor seguridad, menores tiempos de ejecución y mayor orden, entre otras características. PostgreSQL al ser una base de datos relacional posee estas herramientas.

En el presente informe es posible visualizar de manera practica la selección y el uso de herramientas en PostgreSQL con la intención de mejorar los tiempos de ejecución en consultas.

1.1. Herramientas

El ordenador que se utiliza para esta actividad es el mismo que se utilizo para las anteriores, siendo poseedor de las siguientes especificaciones.

- 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 ellos.

2. Desarrollo:

2.1. Creación del entorno

```
--CLIENTES
create sequence codigo start 1;

create table clientes as (select nextval('codigo'), rut,nombre,direccion from
    personas1 limit(5000000));

alter table clientes rename nextval to codigo;
```

Para crear la tabla clientes es necesario instanciar una secuencia, la que itera de 1 hasta el limite dado en la consulta, en este caso 5000000. De esta forma se ingresarían en la nueva entidad los campos o atributos código (la secuencia), rut, nombre y dirección de *Personas*.

El último *ALTER TABLE* es necesario para cambiar el nombre de la columna a código, ya que al usar la clausula nextval esta queda con el nombre de "nextval".

```
-- PRODUCTOS
create table productos as (select generate_series(1,1000) as codigo,md5(random()
    ::text) as nombre, floor(random()*(1000000-1)+1) AS precio);
```

Para la creación de productos fue necesario utilizar una consulta *SELECT* que genera datos, esta puede ser secuencial o aleatoria dependiendo de la columna a tratar. La query contiene los atributos "codigo", el que se genera mediante una serie de 1 a mil, esta última también permitirá designar un final para el *SELECT* en cuestión. Por otra parte la variable "nombre" contiene cadenas de caracteres generadas de forma aleatoria, por último a "precio" se le asigna un valor con tope inferior 1 y superior de 1000000.

```
#Generar data ventas
import random
Cantidad_Cliente=[0 for x in range(1,5000002)]
id_clientes=list(range(1,5000001))
id_productos=list(range(1,1001))
count=0
id_cliente=0
id_producto=0
linea=""
ventas=[]
F = open("ventas.txt","w")
while count<200000000:
    id_cliente=random.choice(id_clientes)
    try:
        if Cantidad_Cliente[id_cliente]<40:
            id_producto=random.choice(id_productos)
            cantProd = random.randint(1,10)
            linea=str(id_cliente)+"|"+str(id_producto)+"|"+str(cantProd)+"\n"
            F.write(linea)
            Cantidad_Cliente[id_cliente]=Cantidad_Cliente[id_cliente]+1
```

```
            count=count+1
    except IndexError as e:
        print(e)
        print(id_cliente)
F.close()
```

Para la generación de datos que serán ingresados a la tabla ventas es necesario realizar un programa, en este caso el código esta escrito en Python un lenguaje de alto nivel. La lógica a la cual se acopla el código es la siguiente.

Como se utilizan números random para completar información es necesario importar la librería "random", luego teniendo presente los atributos que debe tener la tabla ventas es necesario crear variables, en este caso las variables son llamadas "id_cliente, id_producto, Cantidad_Cliente".

Una vez definidas las variables es necesario ejecutar un archivo en modo escritura y generar un ciclo para los doscientos millones de filas de datos que necesitamos incorporar a este. Cada una de estas filas cumple con la condición de que a cada cliente se le haya vendido uniformemente hasta un máximo de 40 productos diferentes y cantidades variables de entre 1 y 10. Cumpliendo esta condición se escribirá una linea en el archivo la cual se ingresa partiendo por el string "id_producto", seguido de "id_producto" y terminado por "cantProd".

```
-- ventas
create table ventas(
    codigocliente serial,
    codigoproducto serial,
    cantidad INTEGER
);
copy ventas from '/home/joaquin/BDDA/BDDA8/ventas.txt' delimiter '|';
```

Una vez generado el archivo ventas.txt, se crea la tabla ventas con sus atributos correspondientes "codigocliente, codigoproducto y cantidad", para luego realizar un "copy" del archivo en esta tabla.

Por último y no olvidar que para ciertos campos de tablas que se encuentran en la base de datos al momento de crearlos los $SELECT$'s que generaban los datos aleatorios los hacían en formatos de variables que ocasionaban conflictos con las columnas en las que se buscaban relacionar. De esta forma previo al hacer las querys se tuvo que hacer $ALTER\ TABLE$ para cambiar el tipo de atributo procedente a las columnas.
La sintaxis vendría a ser la siguiente:

```
alter table productos alter column codigo type numeric;
```

2.2. Consultas en un entorno no optimizado

A continuación se encuentran las consultas realizadas para los puntos 1, 2 y 3 de la actividad.

```
--Query1
select clientes.nombre from clientes,(select ventas.codigocliente,
sum(ventas.cantidad * productos.precio) from productos, ventas
     where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigocliente
    ) as monto
     WHERE monto.sum = (select max(mayor.sum) from (select
     sum(ventas.cantidad * productos.precio) from productos, ventas
     where ventas.codigoproducto = productos.codigo
     GROUP BY ventas.codigocliente) as mayor);
```

Primero se identifica la columna nombre de la tabla $CLIENTES$. Luego se crea la tabla $MONTO$, dicha tabla contiene los valores de cada producto y las veces que el cliente realizó una compra de algún tipo mercancía, dicha asociación vincula el código del producto de la tabla ventas ("codigoproducto") con el código de $PRODUCTOS$ ("codigo"). Es útil usar la columna "codigocliente" de ventas en $MONTO$ puesto que es posible agrupar todas las compras realizadas por un determinado cliente. Finalmente a través de una condición con una estructura similar a la de $MONTO$ se encuentra el máximo y se compara a través de un $WHERE$ cada fila perteneciente a $MONTO$.

```
-- Query2
select productos.nombre from productos, (select ventas.codigoproducto, sum(ventas
    .cantidad)
from productos, ventas where ventas.codigoproducto = productos.codigo GROUP BY
    ventas.codigoproducto)
    as numero WHERE numero.sum = (select min(menor.sum) from (select
     sum(ventas.cantidad) from productos, ventas
     where ventas.codigoproducto = productos.codigo
     GROUP BY ventas.codigoproducto) as menor);
```

En primera instancia se localiza el o los nombres de los productos, que a parte de ubicarse en la tabla $PRODUCTOS$, cumplan con una cantidad mínima en función de ventas realizadas, es decir, que se encuentren en la tabla $VENTAS$ mediante la relación creada entre los atributos *ventas.codigoproducto* y *productos.codigo*. Así mismo se agrupan por *ventas.codigoproducto* permitiendo usar la sentencia *sum* para sumar la cantidad asociada a cada producto en bloques.

Por último se realiza una subconsulta similar a la empleada en la Query 1, pero en este caso para buscar el menor, comprobando por cada fila si la cantidad corresponde a la mínima; A través de la conexión entre *ventas.codigoproducto* y *productos.codigo*.

```
-- Query 3
select productos.nombre from (select ventas.codigoproducto as newcode from
    clientes,ventas where
clientes.codigo = ventas.codigocliente and clientes.rut = 5360294326) as tabla,
    productos
where productos.codigo = tabla.newcode;
```

La tercera query se encarga de buscar todos los nombres de los productos que estén registrados en ventas, a través de un rut que servirá como un verificador o confirmación de su compra o transacción a demás de la condición que *clientes.codigo* sea igual en *ventas.codigocliente*.

En la presente consulta se vincular las llaves foráneas tanto de "productos" como de "clientes" con las de "ventas", para ello se utiliza la palabra clave "JOIN", la condición con la que debe cumplir la consulta es que el rut en clientes sea 5360294326 dando como respuesta los nombres de los productos comprados pro el cliente con este rut.

2.3. Consultas en un entorno optimizado

En esta ocasión y con el objetivo de aminorar los tiempos de ejecución para las consultas es necesario realizar cambios tanto en las mismas consultas como en las tablas, utilizando las herramientas del motor de base de datos.

```
1 create index on clientes(rut);
2
3 alter table ventas add constraint codigocliente_fk foreign key(codigocliente)
    references clientes(codigo);
4
5 alter table ventas add constraint codigoproducto_fk foreign key(codigoproducto)
    references productos(codigo);
6
7 alter table productos add constraint productos_pk primary key (codigo);
```

Se crea un index en rut de clientes, variable que se utiliza como referencia al momento de realizar las consultas. Además se agregan llaves foráneas a la tabla ventas con la intención de disminuir tiempos de ejecución al relacionar tablas.

En la tabla "ventas" se realizan dos llaves, una para referenciar el código de los clientes y otra para referenciar el código de los productos. Por otra parte, en productos se genera una llave primaria para el atributo código ya que es la variable que se utiliza para buscar en su tabla (Figura 10, Anexo. Cambios en tablas para optimización).

A continuación se presentan las querys para el entorno optimizado:

La siguientes dos query presenta una complejidad mucho menor que su contra parte del entorno no optimizado, esto se debe al uso de las "vistas". En este caso las vistas tienen como característica de generar un entorno con índices normalizado, lo cual agiliza el proceso de búsqueda.

Al normalizar la base de datos los datos presentan una dependencia respecto a la creación de la nueva tabla contenedora de la información, además los índices agilizan el proceso de ejecución y el tiempo de respuesta de la query al momento de encontrarse trabajando con "vistas".

```sql
-- Query1
create view monto as (select ventas.codigocliente,
sum(ventas.cantidad * productos.precio) from productos, ventas
     where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigocliente
    );

create view MaxNum as (select max(mayor.sum) from (select
     sum(ventas.cantidad * productos.precio) from productos, ventas
     where ventas.codigoproducto = productos.codigo
     GROUP BY ventas.codigocliente)as mayor);

select clientes.nombre from clientes, monto, MaxNum
     WHERE monto.sum = MaxNum.max;
```

```sql
-- Query2
create view ElPepe as (select ventas.codigoproducto, sum(ventas.cantidad)
from productos, ventas where ventas.codigoproducto = productos.codigo GROUP BY
    ventas.codigoproducto);

create view minimum as (select
     sum(ventas.cantidad) from productos, ventas
     where ventas.codigoproducto = productos.codigo
     GROUP BY ventas.codigoproducto);

select productos.nombre from productos, ElPepe, minimum
    WHERE ElPepe.sum = minimum.sum;
```

En cuanto a la optimización de la tercera consulta, el código del producto a de tomar el rol de índice, como se estableció en la creación de entorno respecto a las "foreing key".

```sql
-- Query3
create view findrut as (select productos.nombre from (select ventas.
    codigoproducto
as newcode from clientes,ventas where clientes.codigo = ventas.codigocliente and
    clientes.rut = 5360294326) as tabla, productos
where productos.codigo = tabla.newcode);
select * from findrut;
```

La optimización realizada a la tercera consulta fue realizada mediante vistas materializadas,las cuales se encuentran anexadas al final del informe (Figura 17, 18 y 19). Esta optimización es la mejor en tiempo de ejecución al realizar las consultas, este cambio se da debido a que el optimizador de consultas selecciona la vista si determina que ésta puede sustituirse por parte o por toda la consulta del plan de ejecución si es de un coste menor (La sintaxis para la creación de vistas materializadas vendría a ser la misma que la de una vista común y corriente; Solo que se debe de anteponer antes que la sentencia $VIEW$ la instrucción $MATERIALIZED$).

2.4. Resultados y análisis de resultados

Los resultados obtenidos en tiempos de ejecución para cada una de las consultas realizadas se encuentran en la siguiente tabla comparativa.

	Query 1	Query 2	Query 3
Sin optimizar	10:24,410 [min]	7:27,623 [min]	1:00,218 [min]
Optimización 1	6:42,277 [min]	1:32,851 [min]	00:54,069 [s]
Optimización 2	8:13,947 [min]	1:59,439 [min]	00:52,389 [s]
Optimización 3	00:04,944[s]	320,971 [ms]	6,194 [ms]

Cuadro 1: Tabla comparativa de tiempos de ejecución

Teniendo en cuenta la tabla con sus respectivos resultados de optimización con 3 métodos distintos y ejecutados sobre la misma query (dependiendo de lo que se quiere consultar en la actividad), se puede concluir que el método asociado a vistas materializadas es el más rápido y efectivo en nuestro caso. Pero cabe hacer mención que la vista materializada se encuentra indizada, esto quiere decir, que la estructura de la figura 10 (ver página 16, 4.3 Creación entorno con Optimizar) fue de suma importancia para mantener índices, llaves foráneas y llaves primarias. También se debe considerar que es mejor desplazar u ordenar las filas a través de bloques, ya que resulta un poco más óptimo en contraste al usar la sentencia $ORDER\ BY$, puesto que se estarían desplazando los índices con $GROUP\ BY$ y además se desplazarían según un orden gestipulado con la instrucción ya mencionada $ORDER\ BY$ que no solo desplazará los valores de las filas sinó que también hará esto mismo con los índices añadiendo más "peso" y decrementando el rendimiento de la base de datos y afectando también con la velocidad del disco duro en este caso "SSD". En si no es del todo la mejor solución puesto que a pesar de que entregue los resultados de forma ágil no se están usando todas las posibilidades de herramientas u opciones que entrega la estructura, ya que en sí no se puede actualizar contenido como una tabla y en caso que se deseé aplicar algún tipo de "update" se tendría que usar alguna función de carácter disparador o al momento de declararse usar un *constraint* que permita la actualización de esta. Y como es una vista la actualización de esta, puede demorar bastante dependiendo de la cantidad de filas, ya que los datos se van a borrar e insertar nuevamente. De esta forma se estarían aplicando dos querys por detrás un $DELETE$ y un $INSERT$. La forma de contrarestar esto es al momento de instanciarla la Vista Materializada, colocarle limitaciones para los casos de actualización.

En el caso de usar únicamente Vistas indizadas tienden a aportar beneficios pero a la vez contras, ya que la indexación masiva de una tabla transaccional con inserciones o modificaciones masivas, puede generar Vistas con mayor consumo de recursos, poca consistencia de datos a consultar con la vista asociada, etc; pero no olvidar que simplifica el tipeo o escritura; Tal como se puede ver en los escenarios con consultas de gran tamaño incluyendo la comunicación entre diversas tablas tales son productos, clientes y ventas. De esta forma permite un proceso similar a la normalización de bases de datos, pero usando $VIEWS$. No obstante, no se debe pasar por alto las limitaciones que ofrece una vista, tal como se mencionó en el

párrafo anterior con la actualización de datos en tablas. Representado esto último en los detalles asociados a las compras que generaron un mayor ingreso ($VIEW\ MONTO$ y $VIEW\ MAYOR$). Cabe recalcar que la optimización es efectuada únicamente por los índices aplicados en las columnas de cada tabla, generando así una mejora en el rendimiento de la base de datos. En resumen las vistas indizadas pueden ofrecer grandes mejoras de rendimiento pero siempre en entornos adecuados y por último evitar su uso en entornos transaccionales de gran tamaño que incluyan actualizaciones ya que, independiente de tener una composición similar a una tabla, no tiene esa libertad propia para cambiar atributos, borrar elementos, etc. Tal como dice su nombre es una Vista.

Por último el entorno de indexación aplicado únicamente(ver página 16, 4.3 Creación entorno con Optimizar, figura 10) y sin ningún tipo de herramienta añadida también llega a ser en muchos aspectos muy útil y con resultados muy prometedores (ver Cuadro 1, 2.4 Resultados y análisis de resultados, página 9) sobre todo si se busca simplificar o reducir el tiempo de ejecución de los $SELECT$'s solicitados. Pero como se mencionó en párrafos anteriores, si no se tienen identificados los campos a indexar y se crean índices sin ningún tipo de orden, ya sea con atributos que no pertenezcan a la relaciones entre tablas sería un desperdicio de espacio en el disco y a la vez puede relentizar procesos de lectura y escritura. Esto último se puede ver al momento de usarlos con vistas al momento de cargarlas con datos puesto que los datos ahora también comprenden $CONSTRAINT$'s de la misma naturaleza que los indices; y cuando se ejecutan consultas como $ALTER\ TABLE$ ya que si la columna que se busca cambiar tiene función de índice puede llegar a demorar bastante dependiendo del volumen de la tabla.

3. Conclusión

De acuerdo a los datos proporcionados en el presente informe, al utilizar las herramientas proporcionadas por la base de datos de manera efectiva, se puede apreciar la optimización en tiempos de ejecución para las distintas consultas "Select", lo que valida el correcto cumplimiento de los objetivos propuestos para esta actividad.

Como otra observación, no es recomendable trabajar con este estilo de base de datos, sin antes haber hecho un tratamiento a esta, tal es el caso de realizar el uso de índices, vistas y otras herramientas.

Como último y no menos importantes podemos decir que el tratamiento y análisis de base de datos es fundamental debido a que en base a pequeñas mejoras y modificaciones podemos llegar a tiempo ampliamente mejores, debido a la correcta implementación de "index", "foreing key" y vistas, como se puede apreciar en el recuadro de resultados que son tiempos menores al realizar esta optimización.

4. Anexado

4.1. Creación de Entorno sin Optimizar

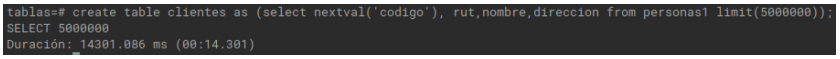

Figura 1: Creación tablas *Clientes*

Figura 2: Creación tablas *Productos*

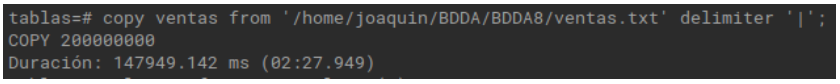

Figura 3: Creación tablas *Ventas*

4.2. Desarrollo de la Actividad

```
tablas=# select clientes.nombre from clientes,(select ventas.codigocliente,
tablas(# sum(ventas.cantidad * productos.precio) from productos, ventas
tablas(#      where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigocliente) as monto
tablas-#      WHERE monto.sum = (select max(mayor.sum) from (select
tablas(#      sum(ventas.cantidad * productos.precio) from productos, ventas
tablas(#      where ventas.codigoproducto = productos.codigo
tablas(#      GROUP BY ventas.codigocliente) as mayor);
Duración: 624410.401 ms (10:24.410)
```

Figura 4: Ejecución de la $QUERY1$

```
                                  nombre
-------------------------------------------------------------------
 BNYGFBPTLBLAXMMQGWFSJARNLYATHBOLRLTDEQVIOYYVGMVRWL
 GKSUCNISYWIBKTBTNDYMTNRLXKLNMKYYWULTGDEDYBWDWGAVXT
 RQOUMUQFUBRPDNTWDQIJNHPDLJNDWVUTXJJOMKRVRIEUFCEXKQ
 PJDNHDTNWFYPGIGRIWYRHVLIFDOGBAHSVCCWYWXCQUWGRXSNBC
 KYPQQDHFXGIDITVNWSLPJDCXMKPTGNDFDGITRRMSMLHSAYMUCO
 GOXGNKPWYAWHBVSISCOHWQUOPBMFYRJJUKDPJMVCVNQCUCWXLG
 YSCUQIXHWHBMHUWSLKAFWSYVEABVIIJWIKVOPOJPHKQMNEOWJW
 VAYMUFAVHXFYKBWFXNBCHXEFVMYMQSVYYQXIIAJHTJCLIONBXT
 WXCMSYTHVSQYGLTOQUNKWMKJIJBVCEBHCLXEKNHGRQGTASSLRC
 QYXNSAXKHQSFKDXHEOCSPLSRREKQLVXGJINGIKKPTVUPIAHIHR
 CGJPNKUFNVEOQYLFYSKNUQNVIWYTUGFISBFTCWBEBTBNRHAYUQ
 GKWJOJIUNYVFTHFGWIUYEPNRBOGBHDFLQGAWIAUGASLNLKIGOX
 JXQXVYHAARTKAOWSWKERNFEOGXUMLEHGANUUJPIULDUTOLTHTS
 KOVTYHLDYUXVRMTOXSTLRPEVXFVLEEPRREASLRGAHHQTPMDAPB
 PLFKGFXWTSTMSATSCGITCKBUXGNHAQSCYMHOFLTSTTVPQLKHQA
 CQNOEVMWBGNKCXTCKMBSLFNFKSTFFWVFJOYPBIHVUWILHWSBQO
 CQNFOONVNNGWFPFHSVUKJPFVCCKVKGSAWORHJSKQCPYTQUTCDJ
 UGKYYOOISTTDXCMSULDOPXOSCQSUBYDDXNVXIUGGMKOQWTMGBQ
 ODCAPHPFMWEUUNQXJMDXRYQFPYRCCHDSHAJUTAKJEKXTPAKLWD
 ULECWXTIOCURRSYXTEJWLMKOKNSYRUCTFDXNWQSKHDOBMNIJSA
 JXFXMIVAALJDMWKFJTQDSNKGVJQRTAKEGWBKIFCPEYPUOGCFJO
 DSFWCLAMGDGPFYGEWVRQKIMICHWWUBWYOFPSBXSXLJRVEBULPO
 HIXQUXKOOLTNIGHNUNOWAJMYDEVKDCBCWATUYDYELLMIRYLUGE
 NSINSXJHCGDKREAQQYMCQRGOFEGSXPFQLWBKPCYAKOUDMMQHAY
 AFYIEWDBRURAIEHXPWJBKWBWSQPYOFAPTDMKNNFWYTSKWNNFMW
 JSVGXPRGDTHBCTTNGTSGGPQCKESCIHOIRRXCJNHFQUIBKXMEED
 BRWHJXEUKUHEQECIVBVUHTXDYBSHIHIETDGBKETQCNECKNMSGF
 DBIQASVKMWFAHKPDFQHWLLECHVKFTVNXRRSHOIUXPQKVHKCOVL
 KULKYQGMOWNMGRLCRPHRSYQRAMUAYWWCDHIFXKFXXNHUXTAWGU
 QOXTAIPARWTPRBYLJQUOOLIUYJGYDEXDEXKGJVVJLQVSEXVAOV
 WDRTGEHSVIRACDFVBCPRPACCHYIXDMEROPCNOYGXBXDAYPOEXG
 GMPNAHIKGTTEGTDETYVONYRDCXCWNYPVLTIREMJCCHDATSTQKQ
 NDCBEXNUVRMMTKUHWHAIVBNTEPMHUVAVBEDPBTDCEADIRSEGLT
 NKPKBVMBCYWVEUGEKIFDTMAUYTKRNFGIDOLPJLQKTYPLMGXXEH
 BUXLWLMYMWDFOWQGLEMMLGWAEMVSETAQQRGKOCJQTFMILQIGBS
 XUDNXUXAFNFXHSKUWBVWWBUUNQFUONOEGTMSEFAHOOSMOMTOVA
:
```

Figura 5: Resultado de la $QUERY1$

```
tablas=# select productos.nombre from productos, (select ventas.codigoproducto, sum(ventas.cantidad)
tablas(# from productos, ventas where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigoproducto)
tablas-#     as numero WHERE numero.sum = (select min(menor.sum) from (select
tablas(#      sum(ventas.cantidad) from productos, ventas
tablas(#      where ventas.codigoproducto = productos.codigo
tablas(#      GROUP BY ventas.codigoproducto) as menor);
Duración: 447622.531 ms (07:27.623)
```

Figura 6: Ejecución de la $QUERY2$

```
                nombre
----------------------------------
 d1f1c5d045d2066c6aea49b1a0075b65
 c9142c693e16298d10939742bf3fde33
 95cda82382799c9acbac78ce5cb14870
 ef73d8462f34493572458514b687cdf4
 74804987797c17a48acb0c73a4d1273f
 362d5fd3cb709a0463de70a9775a9a23
 d98c1238dee972e180050d511a2eba0b
 d7370fa2680882d03c2e86f74a5bd766
 c16dfa9e16f46c0a42148fa4d988caa9
 ef65ed7a341196d8dc9c871361c92b79
 a7d22471baafef19ff7b83c8549df368
 b004a0457cef3d424e037b6bf7f0182b
 e97b029bad3c9217aa2bfbd5577f63ef
 fa894f77b7d55573abfb91531d1da9bf
 9ffaf8fd5caa97eb88839d57e6b0ee85
 46b5dc0f5a85850d72191aa0f6dd1808
 8df86a262047684926af24064ec67b63
 7dec4ac4ffca9b19082a038bdd8a531c
 0c4855e282f06c81675a4ef750e8e8fb
 fffa06c85ead264f40dcaad5ad713793
 40fe8f6dc2f96c8d3de698dae3329061
 c62ee66a03bf25f78f94d7e7462238c4
 9b99e1096a4372b13d1cf1b2665276e1
 6c67276064fb9f2fac056d4afc87e053
 b6ba1f4e6662fd7794fd419c6426ec56
 213c0c49de4191dda9c68ca2e2c304ed
 2f913879bb129d1ec8988d2df746a7b1
 4cf891df3728b422e9566f926d600478
 f5449f2175f9e8a01967924516cda94a
 f2dc759bfaa0f27fcf7e736f62247389
 fa07b6b053f76c91e1a7c8fe60b13391
 5a4bca32133a045225c420d0c3eea4f3
 f341a75ec417a10a5676fa697f4b92bc
 d2afc211a5f058e10c69eddf5ab81d23
 a6c061666f799963d2e1e4172f6e72c1
 55cf94fd6a31a7789fd360ed7442f699
:
```

Figura 7: Resultado de la $QUERY2$

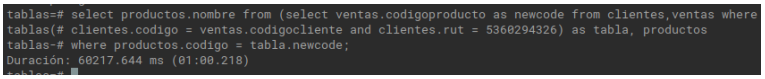

Figura 8: Ejecución de la $QUERY3$

Figura 9: Resultado de la $QUERY3$

4.3. Creación entorno con Optimizar

```
tablas=# \d ventas
                        Tabla «public.ventas»
    Columna     |     Tipo      | Ordenamiento | Nulable | Por omisión
----------------+---------------+--------------+---------+-------------
 codigocliente  | numeric(10,0) |              |         |
 codigoproducto | numeric(10,0) |              |         |
 cantidad       | integer       |              |         |
Restricciones de llave foránea:
    "codigocliente_fk" FOREIGN KEY (codigocliente) REFERENCES clientes(codigo)
    "codigoproducto_fk" FOREIGN KEY (codigoproducto) REFERENCES productos(codigo)

tablas=# \d clientes
                        Tabla «public.clientes»
  Columna  |      Tipo      | Ordenamiento | Nulable  | Por omisión
-----------+----------------+--------------+----------+-------------
 codigo    | numeric        |              | not null |
 rut       | numeric(10,0)  |              |          |
 nombre    | character(50)  |              |          |
 direccion | character(100) |              |          |
Indices:
    "clientes_pkey" PRIMARY KEY, btree (codigo)
    "clientes_rut_idx" btree (rut)
Referenciada por:
    TABLE "ventas" CONSTRAINT "codigocliente_fk" FOREIGN KEY (codigocliente) REFERENCES clientes(codigo)

tablas=# \d productos
                        Tabla «public.productos»
 Columna |       Tipo       | Ordenamiento | Nulable  | Por omisión
---------+------------------+--------------+----------+-------------
 codigo  | numeric          |              | not null |
 nombre  | text             |              |          |
 precio  | double precision |              |          |
Indices:
    "productos_pk" PRIMARY KEY, btree (codigo)
Referenciada por:
    TABLE "ventas" CONSTRAINT "codigoproducto_fk" FOREIGN KEY (codigoproducto) REFERENCES productos(codigo)
```

Figura 10: Cambios en tablas para optimización

```
tablas=# select clientes.nombre from clientes,(select ventas.codigocliente,
tablas(# sum(ventas.cantidad * productos.precio) from productos, ventas
tablas(#       where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigocliente) as monto
tablas-#       WHERE monto.sum = (select max(mayor.sum) from (select
tablas(#       sum(ventas.cantidad * productos.precio) from productos, ventas
tablas(#       where ventas.codigoproducto = productos.codigo
tablas(#       GROUP BY ventas.codigocliente) as mayor);
Duración: 402276.611 ms (06:42.277)
```

Figura 11: Query 1 para la primera optimización y su resultado

```
tablas=# select productos.nombre from productos, (select ventas.codigoproducto, sum(ventas.cantidad)
tablas(# from productos, ventas where ventas.codigoproducto = productos.codigo GROUP BY ventas.codigoproducto)
tablas-#      as numero WHERE numero.sum = (select min(menor.sum) from (select
tablas(#       sum(ventas.cantidad) from productos, ventas
tablas(#       where ventas.codigoproducto = productos.codigo
tablas(#       GROUP BY ventas.codigoproducto) as menor);
Duración: 92850.762 ms (01:32.851)
```

Figura 12: Query 2 para la primera optimización y su resultado

```
tablas=# select productos.nombre from (select ventas.codigoproducto as newcode from clientes,ventas where
tablas(# clientes.codigo = ventas.codigocliente and clientes.rut = 5360294326) as tabla, productos
tablas-# where productos.codigo = tabla.newcode;
Duración: 54069.250 ms (00:54.069)
```

Figura 13: Query 3 para la primera optimización y su resultado

```
tablas=# select clientes.nombre from clientes, monto, MaxNum
tablas-#      WHERE monto.sum = MaxNum.max;
Duración: 493946.938 ms (08:13.947)
```

Figura 14: Query 1 para la segunda optimización y su resultado

```
tablas=# select productos.nombre from productos, ElPepe, minimum
tablas-#     WHERE ElPepe.sum = minimum.sum;
Duración: 119438.550 ms (01:59.439)
```

Figura 15: Query 2 para la segunda optimización y su resultado

```
tablas=# select * from findrut;
Duración: 52388.827 ms (00:52.389)
tablas=#
```

Figura 16: Query 3 para la segunda optimización y su resultado

```
tablas=# select clientes.nombre from clientes, monto, MaxNum
tablas-#      WHERE monto.sum = MaxNum.max;
Duración: 4944.018 ms (00:04.944)
```

Figura 17: Query 1 para la tercera optimización y su resultado

```
tablas=# select productos.nombre from productos, ElPepe, minimum
tablas-#     WHERE ElPepe.sum = minimum.sum;
Duración: 320.471 ms
```

Figura 18: Query 2 para la tercera optimización y su resultado

```
tablas=# select * from findrut;
Duración: 6.194 ms
tablas=#
```

Figura 19: Query 3 para la tercera optimización y su resultado

5. Bibliografía

- Documentación de Views para PostgreSQL
  https://www.postgresql.org/docs/9.2/sql-createview.html
  Página consultada el 20 de Junio del 2021.
- Documentación de Vistas Materializadas para PostgreSQL
  https://www.postgresql.org/docs/12/rules-materializedviews.html
  Página consultada el 21 de Junio del 2021.
- Documentación de Índices Geeks for Geeks
  https://www.geeksforgeeks.org/indexing-in-databases-set-1/
  Página concultada el 21 de Junio del 2021.
- Documentación de Vistas Indizadas
  https://www.dataprix.com/es/blog-it/ilmasacratore/sql-server-vistas-indizadas-y-porque-usarlas-cargas-dwh#:~:text=Una%20vista%20indizada%20se%20diferencia,hacer%20con%20una%20vista%20simple.
  Página consultada el 21 de Junio del 2021.