udp Escuela de Informática y Telecomunicaciones

Bases de Datos Avanzada

Actividad 6: Trabajando con Vistas

Autores:
Nicolás Correa
Joaquín Fernández
David Pazán

Profesor:
Juan Ricardo Giadach

junio 8 del 2021

Índice

1. Introducción

En los sistemas de bases de datos actuales, existen herramientas que ayudan al usuario en distintas áreas tales como la seguridad, el acceso a los datos, rendimiento, etc. Una de estas herramientas son las $VIEWS$, con las cuales se puede revisar el resultado de una consulta preestablecida tal como si fuera una tabla interior de la base de datos o así mismo virtual. Durante este documento se analizará el funcionamiento de las $VIEWS$ en distintos casos, con el objetivo de ver cuales son las ventajas y desventajas de ocupar $VIEWS$ y en que situaciones ocurren.

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. Conceptos importantes

2.1.1. Vistas en Bases de datos

Herramienta incorporada en bases de datos tradicionales como puede ser PostgreSQL o MySQL.

Como su nombre indica una *View* es una vista de una consulta a la cual se le asigna un nombre con la intención de ser utilizada como tabla a nivel de sintaxis, estas pueden ser temporal o no temporal. Son comúnmente llamadas tablas virtuales, ya que a diferencia de las tablas tradicionales estas no se manifiestan de manera física en el disco duro, es decir, solo muestran los datos que están almacenados en otras tablas (que sí son reales).

La consulta se ejecuta cada vez que se hace referencia a la vista. Las cuales por defecto son solo de lectura, no se permite insertar, actualizar ni eliminar, aunque existen métodos para actualizar datos a través de otras herramientas, pero esto no es recomendado.

La creación de una vista es bastante flexible, es posible crearlas con un conjunto de tablas, otras vistas, unión de tablas y vistas, un conjunto de otras vistas, entre otros.

Una vista es una alternativa para mostrar datos de varias tablas, esto nos permite una serie de características:

- Limitar información: El usuario solo puede consultar la vista, por lo que uno de sus principales usos es limitar la información a la que puede acceder un usuario de la base de datos.
- Permisos de usuario: Es posible dar al usuario permisos para acceder a los datos a través de vistas, en lugar de conceder permisos a otros campos, así se protegen las tablas base (originales) de cambios en su estructura.
- Mayor facilidad en escritura de consultas: Se puede evitar escribir instrucciones complejas repetidamente almacenando en una vista el resultado de una consulta que incluya información de varias tablas.

2.2. Desarrollo de la actividad

Al inicio de la presente actividad es necesario implementar el siguiente script para la generación de datos con "edad" del 1 al 100 y con su respectiva "descripción", la cual es la edad escrita en palabras. Y así almacenar los resultados en un archivo de texto en este caso "*demofile.txt*".

```
def convert_to_words(num):
  line = ""
  l = len(num)
  if (l == 0):
    return ("String vacio")

  if (l > 4):
    return ("Largo mayor a 4 no soportado")

  single_digits = ["cero", "uno", "dos", "tres",
          "cuatro", "cinco", "seis", "siete",
          "ocho", "nueve"]
  two_digits = ["", "diez", "once", "doce",
        "trece", "catorce", "quince",
        "dieciseis", "diecisiete", "dieciocho",
        "diecinueve"]
  tens_multiple = ["", "", "veinte", "treinta", "cuarenta",
          "cincuenta", "sesenta", "setenta", "ochenta",
          "noventa"]
  if (l == 1):
    line = line + single_digits[ord(num[0]) - 48]
    return line
  x = 0
  while (x < len(num)):
    if (l >= 3):
      if (ord(num[x]) - 48 != 0):
        line = line + "cien"
      l -= 1
    else:
      if (ord(num[x]) - 48 == 1):
        sum = (ord(num[x]) - 48 + ord(num[x+1]) - 48)
        line = line + two_digits[sum]
        return line
      elif (ord(num[x]) - 48 == 2 and ord(num[x + 1]) - 48 == 0):
        line = line + ("veinte")
        return line
      else:
        i = ord(num[x]) - 48
        if(i > 0):
          line = line +  (tens_multiple[i])
        else:
          line = line + ""
        x += 1
        if(ord(num[x]) - 48 != 0):
          if (tens_multiple[i] == "veinte"):
                line = "veinti"
```

```
                line = line + single_digits[ord(num[x]) - 48]
            else:
                line = line + " y " + single_digits[ord(num[x]) - 48]
        x += 1

    return line

if __name__ == '__main__':
    f = open("demofile.txt", "w")
    endl = "\n"
    for i in range(1,101):
        if i == 100:
            f.write(str(i)+","+convert_to_words(str(i)))
            break
        f.write(str(i)+","+convert_to_words(str(i))+endl)
    f.close()
```

Posteriormente se comprueba el largo máximo que poseen los caracteres de la descripción.

```
if __name__ == '__main__':
    f = open ('demofile.txt','r')
    arr = []; seguir = True
    while seguir:
        mensaje = f.readline()
        if len(mensaje) == 0:
            seguir = False
        x = mensaje.split(",")
        if(x[0] == ""):
            break
        arr.append(x[1])
    valor_maximo = len(arr[0])
    for i in range(len(arr)):
        if (len(arr[i]) > valor_maximo):
            valor_maximo = len(arr[i]);
    print(f"La valor_maximoima cantidad de caracteres asociados a los enteros en
    el archivo es: {valor_maximo}")
```

Una vez obtenido este dato, se procede a crear las respectivas tablas para trabajar.

Las tablas edades son creadas con un tamaño máximo de 3 dígitos para edades, y de 19 para sus descripciones, posteriormente los datos del archivo "demofile.txt" son cargados a la base de datos.

```
--Creando Tabla edades
create table edades (edad numeric(3), descripcion char(19));

-- Cargando Datos
copy edades from '/home/joaquin/BDDA/BDDA6/demofile.txt' delimiter ',';
```

Tras cargar los datos, se definen las vistas (CREATE VIEW) respectiva para cada categoria: jovenes, adultos y tercera edad.

```
--Creando View Jovenes
create view jovenes as
    select personas1.rut,
           personas1.nombre,
           personas1.edad,
           personas1.direccion,
           edades.descripcion
    from personas1, edades
    where personas1.edad = edades.edad and
    personas1.edad < 30;

--Creando Views Adultos
create view adultos as
    select personas1.rut,
           personas1.nombre,
           personas1.edad,
           personas1.direccion,
           edades.descripcion
    from personas1, edades
    where personas1.edad = edades.edad and
    personas1.edad >= 30 and personas1.edad <= 60;

--Creando Views Tercera
create view TerceraEdad as
    select personas1.rut,
           personas1.nombre,
           personas1.edad,
           personas1.direccion,
           edades.descripcion
    from personas1, edades
    where personas1.edad = edades.edad and
    personas1.edad > 60;
```

Se realiza la agrupación de los datos para cada vista creada mediante la descripción.

```

--Agrupar jovenes contando cuantas filas hay por descripci n sin indice
select count(*) from jovenes group by descripcion;
```

```
4
5  --Agrupar adultos contando cuantas filas hay por descripci n sin indice
6  select count(*) from adultos group by descripcion;
7
8  --Agrupar TerceraEdad contando cuantas filas hay por descripci n sin indice
9  select count(*) from TerceraEdad group by descripcion;
```

Se indexa la tabla Edades mediante edad y se agrupan los datos para cada una de las vistas creadas en un principio (Jovenes, adultos y tercera edad), ahora estando bajo efecto de un índice.

```
1
2  --Indexar la tabla Edades por edad
3  create index on edades(edad);
4
5  --Agrupar jovenes contando cuantas filas hay por descripci n con indice
6  select count(*) from jovenes group by descripcion;
7
8  --Agrupar adultos contando cuantas filas hay por descripci n con indice
9  select count(*) from adultos group by descripcion;
10
11 --Agrupar TerceraEdad contando cuantas filas hay por descripci n con indice
12 select count(*) from TerceraEdad group by descripcion;
```

En el paso siguiente, se desecha el index creado anteriormente, y se realiza la agrupación de forma similar al paso tres, es decir toman la tablas de Personas y Edades sin índices.

```
1  --Agrupaciones con tablas Personas y Edades similar al paso 3 sin indice
2  drop index edades_edad_idx;
3
4  --Jovenes
5  select count(*)
6  from (select personas1.rut,
7              personas1.nombre,
8              personas1.edad,
9              edades.descripcion,
10             personas1.direccion
11             from personas1, edades
12             where personas1.edad = edades.edad
13             and personas1.edad < 30) as t1
14             group by t1.descripcion;
15
16 --Adultos
17 select count(*)
18 from (select personas1.rut,
19             personas1.nombre,
20             personas1.edad,
21             edades.descripcion,
22             personas1.direccion
23             from personas1, edades
24             where personas1.edad = edades.edad
25             and personas1.edad >=30 and personas1.edad <=60) as t2
```

```
            group by t2.descripcion;

--TerceraEdad
select count(*)
from (select personas1.rut,
            personas1.nombre,
            personas1.edad,
            edades.descripcion,
            personas1.direccion
            from personas1, edades
            where personas1.edad = edades.edad
            and personas1.edad > 60) as t3
            group by t3.descripcion;
```

Ahora se crean nuevas tablas con datos tomados de su vista respectiva y los rangos de edad.

```
--Se desecha el index
drop index edades_edad_idx;
--Crear tablas con datos de las vistas y de los rangos de edad.
--Joven
create table Joven as (select personas1.rut,
            personas1.nombre,
            personas1.edad,
           edades.descripcion,
            personas1.direccion
            from personas1, edades
            where personas1.edad = edades.edad
            and personas1.edad < 30);

--Adulto
create table Adulto as (select personas1.rut,
            personas1.nombre,
            personas1.edad,
           edades.descripcion,
            personas1.direccion
            from personas1, edades
            where personas1.edad = edades.edad
            and personas1.edad >=30 and personas1.edad <=60);

--Tercera
create table Tercera as (select personas1.rut,
            personas1.nombre,
            personas1.edad,
           edades.descripcion,
            personas1.direccion
            from personas1, edades
            where personas1.edad = edades.edad
            and personas1.edad > 60);
```

Se realizan consultas SELECT respecto a las tablas nuevas agrupadas por descripción.

```

--Joven
```

```
3 select count(*) from Joven group by Joven.descripcion;
4
5 --Adultos
6 select count(*) from Adulto group by Adulto.descripcion;
7
8 --Tercera
9 select count(*) from Tercera group by Tercera.descripcion;
```

2.3. Resultados

	Jovenes	Adultos	Tercera Edad
Vistas (Sin Índice)	1:12,266 [min]	1:12,506 [min]	1:13,321 [min]
Vistas (Con Índice)	1:09,544 [min]	1:11,254 [min]	1:13,066 [min]
Tabla Edades y Personas (Sin Índice)	1:13,837 [min]	1:13,871 [min]	1:12,427 [min]
Tablas Nuevas (Sin Índice)	34,346 [s]	28,131 [s]	46,264 [s]

Cuadro 1: Valores pasos 3, 4, 5 y 7, ver anexado figuras: 4, 5, 6, 8, 9, 10, 11, 12, 13, 17, 18 y 19

3. Análisis de resultados

Teniendo presente los datos del apartado de resultados, es posible resaltar diferencias entre los valores obtenidos.

De todas las opciones propuestas para agrupar a través de un *SELECT* los valores de jóvenes, adultos, y tercera edad, la mas óptima es mantener el 100 % de los datos necesarios en una tabla aparte para cada uno (Sin índice, tablas nuevas, ver cuadro 1 página 10 2.3 Resultados). Esto es debido a que los tiempos de búsqueda de la sentencia se ven reducidos a la cantidad de valores que posee la tabla, por ejemplo al agrupar los valores de la tabla *Jóvenes*, no sera necesario pasar por los valores de *Adultos* ni de *TerceraEdad* ya que no serán participes de esta tabla. Sera lo mismo al agrupar los valores tanto de Adultos como Tercera edad.

Es importante tener presente que las vistas no se especializan en aumentar los tiempos de búsqueda, por lo que era de esperar este resultado.

Por otra parte se observan tiempos de ejecución parecidos para la agrupación de las vistas sin y con índices además de la ejecutada directamente en las tablas *Edades* y *Personas*.

Es posible dar respuesta a esto volviendo al apartado de Conceptos importantes, Vistas en Bases de datos (ver 2.1.1 Vistas en Bases de datos, página 4), ya que por definición una vista solo guarda una query con un nombre referencial para ser ejecutada como tabla, por lo que es de esperarse que los tiempos de ejecución para la agrupación en vistas sin índice y para las ejecutadas directamente en las tablas sean muy parecidas, ya que no deberían tener mayor diferencia en su proceso, ejecutando ambas la misma query en su consulta.

Con respecto a la vista ejecutada con índices en la tabla, al ser el índice creado en la tabla de origen (*edades*) teóricamente la única deferencia seria la forma de búsqueda por fila, debido a que la búsqueda por índices es logarítmica y la sin índices es secuencial.

4. Conclusión

Teniendo presente que las ejecuciones de los script mencionados en este informe fueron intercaladas con el reinicio del ordenador con la intención de eliminar el cache que generan las bases de datos, se puede concluir lo siguiente.

Las vistas no son un método para minimizar los tiempos de ejecución, tiene una mayor relación con el control de acceso a los usuarios a sectores de las diferentes tablas, minimizando además el tamaño y la complejidad de la consulta a utilizar.

Aún así es posible rescatar algunos aspectos que tienen que ver con los tiempos de ejecución utilizando $Views$.

Las vistas "sí" generan un impacto en los recursos del sistema, debido a que cada vez que se referencia las $Views$ dicha consulta debe ser ejecutada. Al ser una "tabla virtual" no guarda información en el disco, solo hace referencia a otras tablas al ejecutarse.

Además al momento de comparar las agrupaciones entre las $Views$ con y sin índice, y las tablas $PERSONAS$ y $EDAD$, es posible observar que al momento de realizar las agrupaciones se consideran el total de los datos, debido a que como bien se especifico, las vistas no son un método que mejore los tiempos de búsqueda para las sentencias. Posteriormente al generar nuevas tablas respecto a $PERSONAS$, esta ha de presentar un menor tiempo, debido a que toma los datos específicos que cumplen las características correspondiente a $JOVEN$, $ADULTO$ y $TERCERAEDAD$.

En resumen, el uso de $Views$ en una base de datos facilita al momento de realizar consultas, siempre y cuando las consultas se establezcan bajo un mismo sector respecto a las $Views$. De esta forma se evita la entrega de información importante sobre la estructura de la base de datos, la redundancia de la información y organizando de mejor manera la información dentro de la base de datos, siempre y cuando se esté de acuerdo en sacrificar recursos del sistema para su implementación.

Como conclusión adicional se debe hacer mención de que la actividad en primera instancia no se hizo con el cuidado del reinicio del equipo y con ello trajo así mismo irregularidades con los tiempos obtenidos; ya que podría decirse que las querys quedan almacenadas en un cache y la ejecución posterior de estas contemplan también la optimización del sistema operativo y el motor de bases de datos, esto desemboca en que los tiempos de ejecución en las querys posteriores se ejecutaron en tiempos menores a lo esperado.

Otra información adicional, es que es posible actualizar datos utilizando vistas. Cada vez que se desee editar una vista, el sistema ha de traducir esta acción en una petición de actualización a la tabla de origen de las $Views$ y teniendo presente el tamaño de las $Views$, puede causar problemas y evitar que las actualizaciones se realicen. Es por esto que las vistas podrían utilizarse para lectura de datos.

5. Anexado

Views (paso 2):

```
tablas=# create view jovenes as
tablas-#     select personas1.rut,
tablas-#            personas1.nombre,
tablas-#            personas1.edad,
tablas-#            personas1.direccion,
tablas-#            edades.descripcion
tablas-#     from personas1, edades
tablas-#     where personas1.edad = edades.edad and
tablas-#     personas1.edad < 30;
CREATE VIEW
Duración: 43.328 ms
```

Figura 1: Views para Jovenes

```
tablas=# create view adultos as
tablas-#     select personas1.rut,
tablas-#            personas1.nombre,
tablas-#            personas1.edad,
tablas-#            personas1.direccion,
tablas-#            edades.descripcion
tablas-#     from personas1, edades
tablas-#     where personas1.edad = edades.edad and
tablas-#     personas1.edad >= 30 and personas1.edad <= 60;
CREATE VIEW
Duración: 54.237 ms
```

Figura 2: Views para Adultos

```
tablas=# create view TerceraEdad as
tablas-#     select personas1.rut,
tablas-#            personas1.nombre,
tablas-#            personas1.edad,
tablas-#            personas1.direccion,
tablas-#            edades.descripcion
tablas-#     from personas1, edades
tablas-#     where personas1.edad = edades.edad and
tablas-#     personas1.edad > 60;
CREATE VIEW
Duración: 67.573 ms
```

Figura 3: Views para TerceraEdad

Agrupación según vista (paso 3):

```
tablas=# select count(*) from jovenes group by descripcion;
 count
--------
 498936
 497980
 497825
 499159
 499195
 499301
 498400
 498487
 498192
 498592
 497450
 499264
 499442
 498354
 498740
 498491
 499073
 499028
 498514
 499134
 498187
 498090
 497952
 499986
 499170
 499661
 499020
 499161
 498076
(29 filas)

Duración: 72265.548 ms (01:12.266)
```

Figura 4: Agrupación para Jovenes

```
tablas=# select count(*) from adultos group by descripcion;
Duración: 72505.985 ms (01:12.506)
```

Figura 5: Agrupación para Adultos

```
tablas=# select count(*) from TerceraEdad group by descripcion;
Duración: 73321.331 ms (01:13.321)
```

Figura 6: Agrupación para TerceraEdad

Indexación de tablas (paso 4):

```
tablas=# create index on edades(edad);
CREATE INDEX
Duración: 28.794 ms
```

Figura 7: Indexación en tabla edades

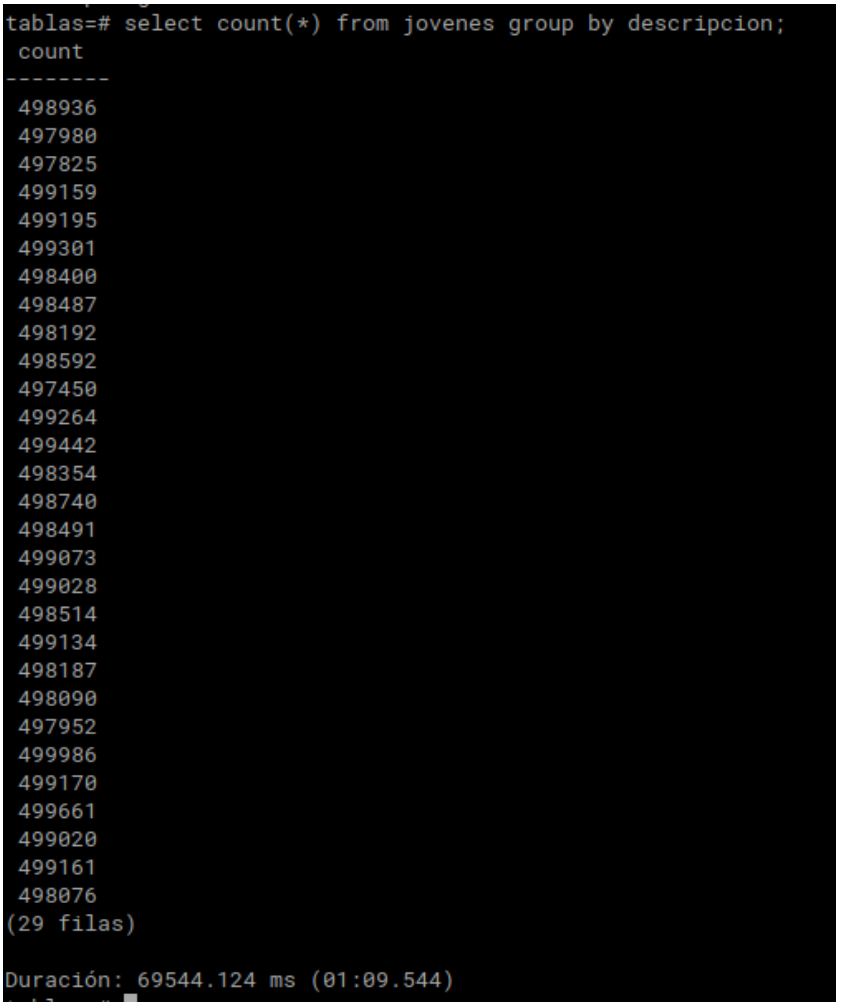

Figura 8: Rescatar datos para Jovenes

```
tablas=# select count(*) from adultos group by descripcion;
 count
--------
 500120
 499482
 499133
 499244
 498821
 498492
 498907
 497390
 498910
 498979
 497198
 499213
 499468
 498930
 499989
 498757
 499171
 499590
 499155
 498419
 499514
 499628
 498483
 498002
 497651
 498302
 497844
 498168
 499493
 498861
 498595
(31 filas)

Duración: 71254.295 ms (01:11.254)
```

Figura 9: Rescatar datos para Adultos

```
tablas=# select count(*) from TerceraEdad group by descripcion;
Duración: 73065.852 ms (01:13.066)
tablas=#
```

Figura 10: Rescatar datos para TerceraEdad

Agrupación utilizando Personas y Edades (paso 5):

```
tablas=# select count(*)
tablas-# from (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#              edades.edad,
tablas(#                edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad < 30) as t1
tablas-#              group by t1.descripcion;
 count
--------
 498936
 497980
 497825
 499159
 499195
 499301
 498400
 498487
 498192
 498592
 497450
 499264
 499442
 498354
 498740
 498491
 499073
 499028
 498514
 499134
 498187
 498090
 497952
 499986
 499170
 499661
 499020
 499161
 498076
(29 filas)

Duración: 73836.739 ms (01:13.837)
tablas=#
```

Figura 11: Agrupación utilizando Personas y Edades para Jóvenes

```
tablas=# select count(*)
tablas-# from (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#              edades.edad,edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad >=30 and personas1.edad <=60) as t2
tablas-#              group by t2.descripcion;
 count
--------
 500120
 499482
 499133
 499244
 498821
 498492
 498907
 497390
 498910
 498979
 497198
 499213
 499468
 498930
 499989
 498757
 499171
 499590
 499155
 498419
 499514
 499628
 498483
 498002
 497651
 498302
 497844
 498168
 499493
 498861
 498595
(31 filas)

Duración: 73870.890 ms (01:13.871)
tablas=#
```

Figura 12: Agrupación utilizando Personas y Edades para Adultos

```
tablas=# select count(*)
tablas-# from (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#              edades.edad,edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad > 60) as t3
tablas-#              group by t3.descripcion;
Duración: 72426.775 ms (01:12.427)
```

Figura 13: Agrupación utilizando Personas y Edades para Tercera Edad

Creación de las tablas JOVEN, ADULTO y TERCERA (Paso 6):

```
tablas=# create table Joven as (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#             edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad < 30);
SELECT 14462860
Duración: 70423.624 ms (01:10.424)
tablas-#
```

Figura 14: Creación tabla Jovenes

```
tablas=# create table Adulto as (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#             edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad >=30 and personas1.edad <=60);
SELECT 15463909
Duración: 77953.855 ms (01:17.954)
```

Figura 15: Creación tabla Adultos

```
tablas=# create table Tercera as (select personas1.rut,
tablas(#              personas1.nombre,
tablas(#              personas1.edad,
tablas(#             edades.descripcion,
tablas(#              personas1.direccion
tablas(#              from personas1, edades
tablas(#              where personas1.edad = edades.edad
tablas(#              and personas1.edad > 60);
SELECT 19450616
Duración: 93840.715 ms (01:33.841)
```

Figura 16: Creación tabla Tercera Edad

Agrupación con las nuevas tablas (Paso 7):

```
tablas=# select count(*) from Joven group by Joven.descripcion;
 count
--------
 498936
 497980
 497825
 499159
 499195
 499301
 498400
 498487
 498192
 498592
 497450
 499264
 499442
 498354
 498740
 498491
 499073
 499028
 498514
 499134
 498187
 498090
 497952
 499986
 499170
 499661
 499020
 499161
 498076
(29 filas)

Duración: 34345.896 ms (00:34.346)
```

Figura 17: Agrupar por descripción Jóvenes

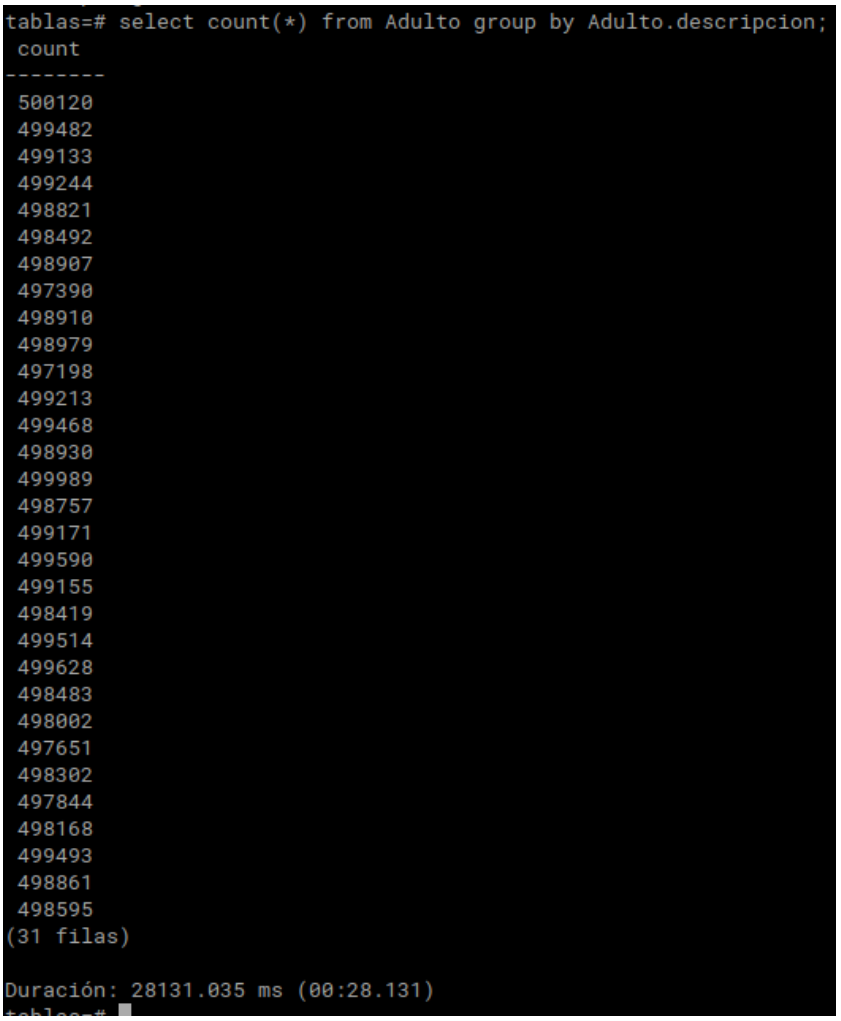

Figura 18: Agrupar por descripción Adultos

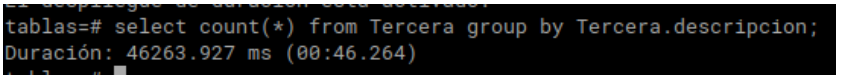

Figura 19: Agrupar por descripción Tercera Edad

6. Bibliografía

- Documentación de Views para PostgreSQL
  `https://www.postgresql.org/docs/9.2/sql-createview.html`
  Página consultada el 04 de Junio del 2021.
- Documentación de Views para SQL Server
  `https://www.sqlshack.com/sql-server-indexed-views/`
  Página consultada el 05 de Junio del 2021.
- Documentación de Views indexadas para SQL Server
  `https://docs.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-ver15`
  Página consultada el 05 de Junio del 2021.
- SQL Views Geeks for Geeks
  `https://www.geeksforgeeks.org/sql-views/`
  Página consultada el 04 de Junio del 2021.