udp Escuela de Informática y Telecomunicaciones

# Bases de Datos Avanzada

## Actividad 1: Inserción de datos

*Autores:*
Nicolás Correa
Joaquín Fernández
David Pazán

*Profesor:*
Juan Ricardo Giadach

3 de Abril del 2021

# Índice

# 1. Introducción:

En palabras simples, un sistema de bases de datos es un repositorio de datos. Compuesto por equipo físico y múltiples programas, esta herramienta le garantiza al usuario las características de independencia, seguridad, confidencialidad, concurrencia y consistencia de los datos; De esta forma se encarga de estructurar y agrupar la data relevante con la finalidad de futuro uso de esta.

Ahora teniendo en cuenta lo descrito en el párrafo anterior se procede a resumir en breve la tarea a desempeñar. La actividad presente ha de tener el objetivo de analizar el tiempo que conlleva la inserción de datos mediante un filtrado de estos, según un uso de comandos o elaboración de programas que se encarguen de dicho labor. Estas cantidades son generadas por sentencias entregadas por el docente; Teniendo así un volumen correspondiente al archivo 5 mil y otro de 50 millones. A la vez se pide almacenar estos en un orden asociado a 2 entidades o tablas siendo estas: Personas1 y Personas2; ya con sus respectivos atributos y claves. Luego se habría de realizar un estudio comparativo respecto al tiempo de ejecución producto del volcado y relleno de estos. Según los escenarios a los que se sujeten las entidades mediante una instrucción de elaboración o pasos a seguir.

Al final del documento se tendrán imágenes que comprueban el funcionamiento y un esquema de lo solicitado.

# 2. Desarrollo:

## 2.1. Diseño y herramientas:

En dos tablas respectivas PERSONAS1 y PERSONAS2, la primera tabla no tiene índices respectivos para los datos, por lo cual, será tomado el tiempo que demora el hacerlo sin índices y con índices. Por otro lado, la segunda tabla que ha de poseer llave primaria por lo que se tomará en cuenta la acción de ingreso de datos. Estos tres casos (insertar el archivo en PERSONAS1 con y sin índices; Y además de insertarlos en la tabla PERSONAS2) se han de repetir, posteriormente, con un archivo de cincuenta millones de datos. Una vez realizada la actividad, se procede al análisis de los resultados obtenidos.

Por último, el computador que se usó para la actividad tiene las siguientes características:

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

## 2.2. Creación de entorno de bases de datos:

Como primer paso de la actividad, es la creación de la base de datos y las entidades que la componen; teniendo esto en cuenta se usaron los siguientes scripts:

```
CREATE DATABASE tablas;
create table personas1(rut numeric(10),
                       nombre char(50),
                        edad numeric(2),
                        direccion char(100)
                       );
create table personas2(rut numeric(10) primary key,
                       nombre char(50),
                        edad numeric(2),
                        direccion char(100)
                       );
```

## 2.3. Filtrado de datos

Si bien en las bases de datos relacionales existen sistemas para trabajar los datos duplicados, en esta ocasión optaremos por filtrar los datos previos a la utilización de la base de datos, esto se hizo con un código en el lenguaje de programación "Python", debido tanto a las librerías, como a la facilidad que tiene este lenguaje para trabajar con lectura y escritura de datos.

```
import sys, os
from time import time

def clean(name, savein):
    Old = open(name,'r')
    New = open(savein+'/'+name+'fixed.csv','w')
    clientes = dict()

    for i in Old:
        curr = i.split(' | ')
        if ((len(curr[0]) <= 10) and (len(curr[1])<=50)
        and (len(curr[2]) <= 2) and (len(curr[3]) <= 101)):
            if (curr[0] not in clientes):
                clientes[curr[0]]=""
                New.write(",".join(curr))

    Old.close()
    New.close()

if __name__ == '__main__':
    while(len(os.listdir("./"))<5):
        if (len(os.listdir("./")) == 3):
            clean("millones50", os.getcwd())
        elif (len(os.listdir("./")) == 4):
            clean("cincomil",os.getcwd())
```

```
        '''star_time = time()
        clean("millones50", os.getcwd())
```

```
        elapsed_time = time() - star_time
        print(elapsed_time)'''
        '''Funcion usada para calcular el tiempo de filtrado (50 millones)
        para el caso de 5 mil, basta cambiar el nombre de la funcion clean)'''
```

Para realizar el filtrado de datos se utilizaron dos paquetes:

- sys: Este modulo permite trabajar con funciones y variables que se utilizan para manipular diferentes partes del entorno de ejecución de Python. Nos permite acceder a parámetros y funciones específicos del sistema.
- Os: Este modulo permite realizar acciones dependiente del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc.

Con estos paquetes se crea una función "main()"que básicamente es la encargada de llamar y enviar parámetros a la función "clean()" estos parámetros tienen relación con la ruta del documento. En este caso debería ejecutar dos veces la función, una vez para cada archivo que tenga la intención de filtrar.

La función "clean()" esta encargada recibir la ruta de un archivo, buscarlo y filtrarlo para posteriormente crear un archivo con los datos nuevos. Para ello se debe generar 2 instancias una para la escritura del nuevo archivo y otra para leer el archivo de la ruta que llego. Los datos serán leídos del archivo generado por el comando linea por línea, siendo comparadas entre ellas con la intención de evitar los repetidos y finalmente estas lineas serian ingresadas en el nuevo archivo.

Con esto se debería tener en el proyecto, 2 archivos nuevos con los datos listos para ser ingresados a la base de datos.

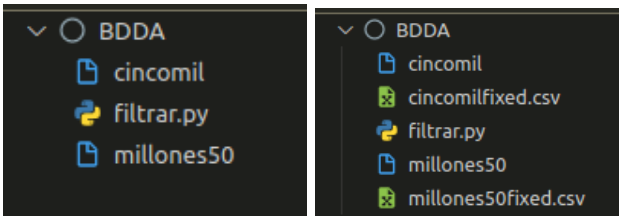


Figura 1: Antes y después de ser ejecutada la función

## 2.4. Archivo 5 mil

Apartado centrado en desarrollar lo respectivo al archivo con los 5 mil datos para las operaciones establecidas en el enunciado.

**Creación tablas Personas1 y Personas2:**
Para las tablas creadas, se tiene los siguientes atributos:

- RUT 10 Dígitos enteros.
- Nombre 50 Caracteres.
- Edad 2 Dígitos enteros.
- Dirección 100 Caracteres.

La diferencia que caracteriza a las tablas creadas, en donde Personas2 ha de tener el atributo RUT como Primary Key.

Proceso de Inserción Personas1 y Personas2:

Una vez realizada la creación de las tablas y teniendo ya los datos filtrados; como se mencionó en el punto anterior; los archivos en formato csv han de ser usados para la inserción de estos últimos, todo esto mediante la terminal en la instancia psql.

```
    #Inserci n para Personas1
    copy personas1 from '/home/joaquin/BDDA/cincomilfixed.csv' delimiter ',';
    #Inserci n para Personas2
    copy personas2 from '/home/joaquin/BDDA/millones50fixed.csv' delimiter ',';

```

Cabe mencionar que la ruta o path cambia dependiendo del equipo y sesión de usuario registrada dentro del dispositivo.

**Indexado Personas1:**

A continuación se realiza el proceso de indexado. En donde se crea una tabla ïndex.ªuxiliar que permite acceder con mayor velocidad a los datos consultados, siempre y cuando hagan referencia a la columna indexada.

```
    create index id_rut on personas1(rut);

```

| 5 mil | Ejecución 1 | Ejecución 2 | Ejecución 3 | Promedio |
|---|---|---|---|---|
| Personas1 Inserción | 29.960 ms | 23.233 ms | 82.874 ms | 45.355 ms |
| Personas1 Indexar | 17.438 ms | 21.415 ms | 34.103 ms | 24.318 ms |
| Personas2 | 46.396 ms | 92.454 ms | 56.516 ms | 65.112 ms |

Esta tabla muestra el promedio de los valores asociados a los procesos hechos con el archivo "5 mil", donde se identificó que los valores tienen cierta coherencia con la comparación teórica que debiese existir entre ellos. Al mismo tiempo se notó que la inserción, a la tabla personas1, al ser sin llave primaria, necesita menos procesos para ser ejecutado, mientras que en personas2 al tener la primary key se genera un incremento en el tiempo de ejecución el cual en este caso no es tan significativo, ya que sigue siendo del valor de los ms.

Por otra parte el indexado como solo está asociado a la tabla personas1 no obtuvo una comparación de este proceso, ya que no se puede comparar con una inserción. Hay que recordar que este proceso trabaja con todos los valores de la tabla relacionando sus valores y creando así un árbol, lo cual se aleja de los pasos con lo que trabaja la inserción.

## 2.5. Archivo 50 Millones

Una vez concluidos los procesos sobre el archivo 5 mil, se procede ha realizar las mismas acciones sobre el de 50 millones.

**Limpiar e ingresar datos a Personas1 y Personas2:**
Las tablas han de ser vaciadas, y al igual que en el documento anterior, se ingresan los datos mediante los comando:

```
1     #Inserci n para Personas1
2    copy personas1 from '/home/joaquin/BDDA/millones50fixed.csv' delimiter ',';
3    #Inserci n para Personas2
4    copy personas2 from '/home/joaquin/BDDA/millones50fixed.csv' delimiter ',';
5
```

Y se obtienen los tiempos respectivos a estos. **Indexar Personas1:**
A continuación se realiza el proceso de indexado. En donde se crea una tabla ïndex.ªuxiliar que permite acceder con mayor velocidad a los datos consultados, siempre y cuando hagan referencia a la columna indexada.

| 50 millones | Ejecución 1 | Ejecución 2 | Ejecución 3 | Promedio |
|---|---|---|---|---|
| Personas1 Inserción | 502137.516 ms | 418676.636 ms | 494020.697 ms | 471611.616 ms |
| Personas1 Indexar | 326264.806 ms | 358572.155 ms | 337523.232 ms | 340.786.731 ms |
| Personas2 | 7942408,482 ms | 7213405.859 ms | 7233136.526 ms | 7462983.622 ms |

Esta tabla muestra los valores asociados a los procesos hechos con el archivo "50 millones", donde se generaron varias ejecuciones para evitar errores, mostrando finalmente un promedio de estos.

Se pudo ver que los valores tienen cierta coherencia con la comparación teórica que debiese existir entre ellos. En específico, la inserción a la tabla personas1 al ser sin llave primaria necesitó menos procesos para ser ejecutada, mientras que en "personas2.ªl tener valga la redundancia llave primaria se genera un incremento en el tiempo de ejecución los cuales en esta ocasión tienen un valor muy elevado.

Por otra parte el indexado tiene un valor propio no comparable con los tiempos de ingreso, esto debido a que el proceso de indexación es distinto al de inserción.

## 3. Análisis generales

El tiempo de filtrado que aconteció en el archivo de 5 mil datos fue de 0.01 ms aproximado, mientras que para el archivo de 50 millones de datos fue 3.58 minutos. Para trabajar en el análisis general e los datos, se utilizará la siguiente tabla:

| Tablas | Cinco Mil | Cinco Millones |
|---|---|---|
| PERSONA 1 (Sin indexar) | 45.355 ms | 471611.616 ms |
| PERSONA 1 (Con indexación) | 24.318 ms | 340.786.731 ms |
| PERSONA 2 | 65.122 ms | 7462983.622 ms |

Como comparación general, se identifica que el promedio de ejecución para la inserción en "Personas1.[en] el archivo de 5 mil datos fue de 45.355 ms mientras que en el de 5 millones de datos fue de 471611.616 ms como promedio y en el caso de "Personas2.[en] el primero fue 65.122 ms y en el segundo 7462983.622 ms (2:04:22 hrs) como promedio. También se notó un parentesco en tiempo de ejecución para los procesos hechos en la misma tabla con cantidades de datos distintos, mientras que una diferencia de tiempo al momento efectuar el mismo proceso en tablas distintas, ya que en "Personas2"los valores son ampliamente mayores que en "Personas1". De esto se puede concluir que tener una Primary key puede generar eficiencia en temas de comunicación con otras entidades y modelamiento; pero la inserción de muchos datos sobre esta y además no hechos con un algoritmo de forma eficiente; Puede afectar no solo en temas de rendimiento sino que hasta uno puede intuir que el tiempo que demora la consulta puede ser secuencial y en el peor de los casos de extensión n; Siendo este último el peor de los estos.

Ahora bien para el indexado, a pesar que agrega tiempo a la inserción en la tabla de "Personas1"no quita que al momento de generar alguna consulta a la base de datos, esta sea mucho más rápida en contraste a la misma sin indexar. Además, al comparar los tiempos, estos son similares para ambos tamaños de archivos, donde tendría que variar ya que se deben de indexar más datos. Se pudo apreciar una ventaja (hasta el momento teórica) con el uso de índices o "index", ya que se intuye que al trabajar con B-tree. Guardando lo que esté en la columna del indexado en su correspondiente dirección de memoria, el árbol crecerá y se contraerá a partir de la raíz de este. Este proceso ha de generar una mayor demora en el proceso de inserción, pero al mismo tiempo mejora considerablemente la eficiencia de búsqueda.

# 4. Conclusión

Muchas personas asumen que una consulta lenta es una señal de que es necesario introducir tecnologías adicionales a SQL, especialmente aquellas enfocadas en el manejo de Big-Data. La realidad es que son pocos y muy específicos los casos en que la información supera las capacidades de un buen motor de base de datos, en la gran mayoría de los casos es más práctico y de mayor beneficio solucionar la lógica de la llamada SQL.

Hoy en día se manejan grandes cantidades de datos por todos lados, por ende un profesional de calidad y que entiende las necesidades que puede cubrir esta habilidad, necesariamente tendrá que ser alguien que propicie buenas practicas, además de eficiencia y eso es lo que se espera desarrollar con el pasar de los contenidos del presente curso.

Por ahora, los objetivos propuestos para el desarrollo de este informe fueron abarcados en su totalidad, tanto el trabajo con los archivos, el filtrado y la inserción, fueron realizados con éxito. Siendo este ultimo una de las conclusiones más importantes relacionadas directamente con los tiempos de ejecución de los programas, observamos que la inserción en la tabla sin llave primara fue notablemente más rápida que en la que la poseía, se entiende esto como un dato de suma importancia a la hora de trabajar con las tablas, debido a que los procesos muy lentos pueden llegar a ser perjudiciales. Llámese esto un costo beneficio, debido a que el uso de este recurso, puede proporcionar eficiencia en otros apartados como búsqueda de datos.

# 5. Anexado

## 5.1. Archivo 5 mil datos:

### Inserciones a personas 1:


Figura 2: Primera inserción, la tabla Personas1

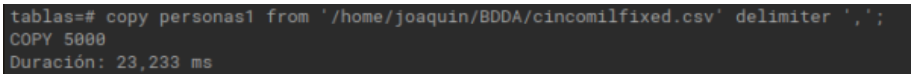


Figura 3: Segunda inserción, la tabla Personas1


Figura 4: Tercera inserción, la tabla Personas1

### Indexación a personas 1:


Figura 5: Primera indexación

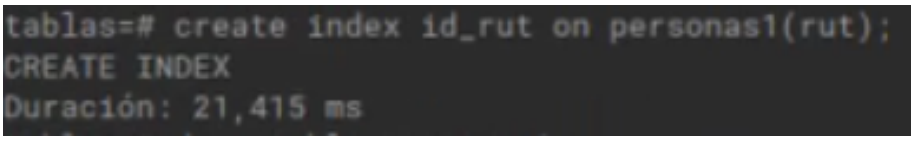


Figura 6: Segunda indexación

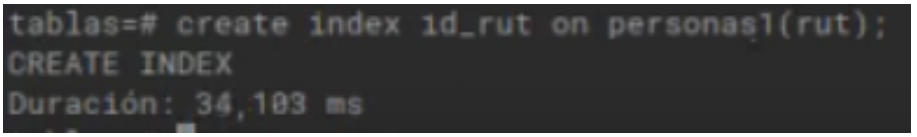


Figura 7: Tercera indexación

**Inserciones a personas 2:**

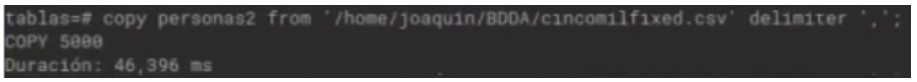


Figura 8: Primera inserción, la tabla Personas2


Figura 9: Segunda inserción, la tabla Personas2

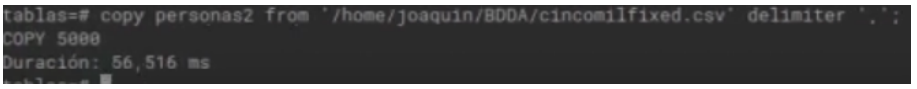


Figura 10: Tercera inserción, la tabla Personas2

## 5.2. Archivo 50 millones de datos:

**Inserciones a personas 1:**

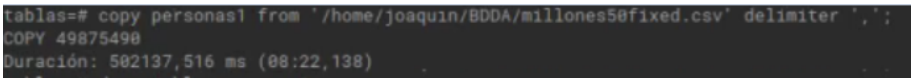


Figura 11: Primera inserción, la tabla Personas1

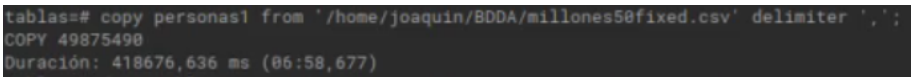


Figura 12: Segunda inserción, la tabla Personas1

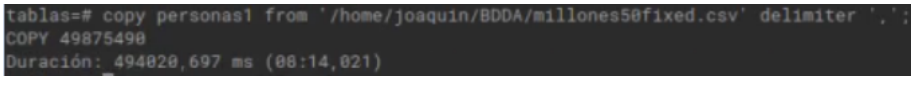


Figura 13: Tercera inserción, la tabla Personas1

**Indexación a personas 1:**

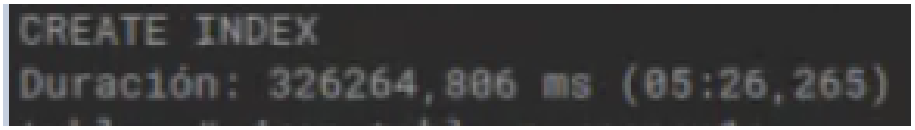


Figura 14: Primera indexación

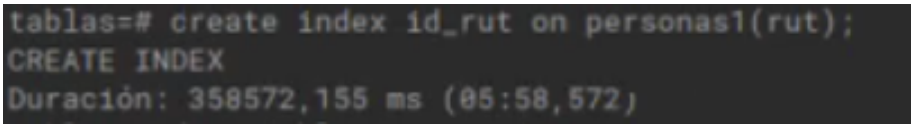


Figura 15: Segunda indexación

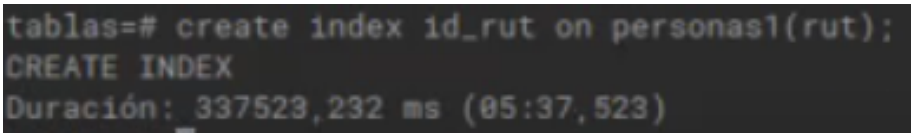


Figura 16: Tercera indexación

**Inserciones a personas 2:**


Figura 17: Primera inserción, la tabla Personas2


Figura 18: Segunda inserción, la tabla Personas2


Figura 19: Tercera inserción, la tabla Personas2