Universidad Nacional Autónoma de México
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Bases de Datos No Estructuradas
Semestre 2020-II
López Sánchez Misael
Tapia López José de Jesús
29 de Febrero del 2020

March 11, 2020

1 Objetivos

Los objetivos de esta práctica son:

1. Desarrollar un sistema de reducción de URLs.
2. El sistema debe soportar el manejo de usuarios.
3. La reducción de URLs debe ser por usuario y permitir que se tengan ligas publicas y ligas privadas (solo distinguirlas).
4. Cada usuario debe tener una wishlist de los sitios que desean visitar en el futuro.
5. Categorizar las URLs.
6. Ser capaces de hacer consultas sobre las intersecciones entre las categorías de dos usuarios.

2 Introducción

Redis es un almacén de estructura de datos en memoria de código abierto, que se utiliza como agente de base de datos, caché y mensaje. Admite estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, etc. Redis tiene replicación incorporada, secuencias de comandos Lua, desalojo de LRU, transacciones y diferentes niveles de persistencia en el disco, y proporciona alta disponibilidad a través de Redis Sentinel y particionamiento automático con Redis Cluster.

Podemos ejecutar operaciones atómicas en estos tipos, como agregar a una cadena; incrementar el valor en un hash; poner un elemento a una lista; conjunto de cómputo intersección, unión y diferencia; o conseguir el miembro con la clasificación más alta en un conjunto ordenado.

Para lograr su rendimiento sobresaliente, Redis trabaja con un conjunto de datos en memoria. Dependiendo de su caso de uso, puede conservarlo volcando el conjunto de datos al disco de vez en cuando o agregando cada comando a un registro. La persistencia se puede deshabilitar opcionalmente, si solo necesita una memoria caché en red rica en funciones y en red.

Redis también admite la replicación asincrónica maestro-esclavo trivial a configuración, con una primera sincronización muy rápida sin bloqueo, reconexión automática con resincronización parcial en división de red.

Podemos mencionar algunas ventajas de utilizar Redis:

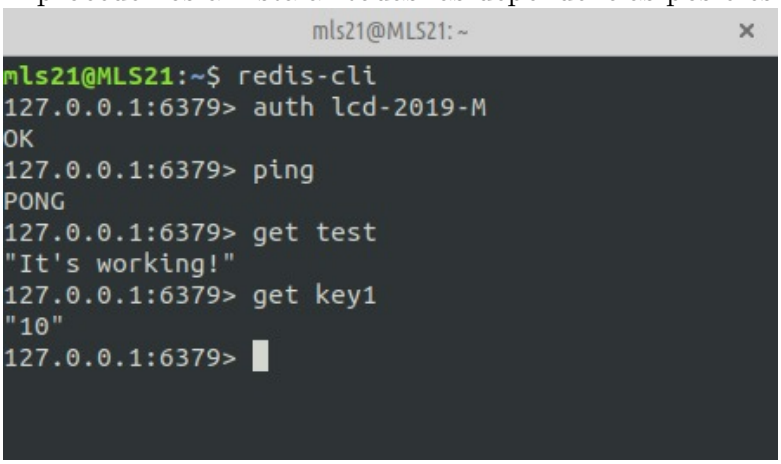
1. Permite almacenar pares de clave y valor de hasta 512 MB. Puede tener enormes claves y

valores de objetos de hasta 512 MB, lo que significa que Redis admitirá hasta 1 GB de datos para una sola entrada.
2. Utiliza su propio mecanismo de hash llamado Redis Hashing. Redis almacena datos en forma de una clave y un mapa, es decir, campos de cadena y valores de cadena.
3. Ofrece replicación de datos. La replicación es el proceso de configurar nodos de caché maestro-esclavo. Los nodos esclavos siempre escuchan el nodo maestro, lo que significa que cuando se actualiza el nodo maestro, los esclavos también se actualizarán automáticamente. Redis también puede actualizar esclavos de forma asincrónica.
4. El caché de Redis puede soportar fallas y proporcionar un servicio ininterrumpido. Dado que Redis se puede usar para configurar una replicación eficiente, en cualquier momento, el servicio de caché estará en funcionamiento, incluso si alguno de los nodos esclavos está inactivo. Sin embargo, los nodos son resistentes y superarán la falla y continuarán brindando servicio.
5. Tiene clientes en todos los lenguajes de programación populares. Redis tiene API de cliente desarrolladas en todos los lenguajes populares como C, Ruby, Java, JavaScript y Python.
6. Admite transacciones, lo que significa que los comandos pueden ejecutarse como una cola en lugar de ejecutarse uno a la vez.

3 Desarollo

Lo primero que hicimos fue instalar Redis a Python en nuestro ordenador con las siguentes instrucciones en la línea de comandos de Ubuntu:

Y procedemos a instalar todas las dependencias posibles y ver que funcionaba adecuadamente:



El programa desarrollado consiste en una aplicación realizada entre Python y Redis por medio de la librería: *Redis PyPI* la cual nos permite poder realizar la conexión entre la base de datos Redis y Python. Hacemos uso únicamente de dos bibliotecas de Python para trabajar bajo esta consola y registrar datos dentro de nuestra base de datos en Redis:

- Redis - Para la conexión Python-Redis.

- os - Para el flujo de datos por consola.

La aplicación empieza mostrando un menú en el cual como usuario podemos ingresar la operación que nosotros queremos realizar; siendo los parámetros del menú los siguientes:

0.- Ingresar nuevo usuario.
1.- Editar datos de usuario.
2.- Buscar Usuario
3.- Cerrar Sesión de un usuario
4.- Ingresar Registros
5.- Comparar Registros
6.- Salir

Los primeras 4 opciones del menú nos permiten trabajar con la parte operacional de los usuarios (dar de alta nuevos usuarios, modificar su información, eliminar usuarios y sus registros, localizar los datos de un usuario, etc). El resto de opciones del menú nos permite trabajar con la inserción de los datos dentro de la base de datos así como realizar consultas básicas sobre conocer aquellos registros comunes registrados entre diferentes usuarios.

Mostramos el código que realizamos para la práctica:

(NOTA: Para mostrar el código, omitimos acentos y los volvamos a incorporar en las imágenes que mostraremos posteriormente.)

```
# Importamos bibliotecas que posteriormente vamos a utilizar
import redis as rds
import os
```

Creamos la conexión entre Python y Redis.

```
# Checamos de que la conexion con redis este funcionando correctamente.
redis1 = rds.Redis(host='localhost',port=6379)
redis2 = rds.Redis(host='localhost',port=6379)

# Comprobamos la conexion con la base de datos
ping = redis1.ping
ping
<bound method Redis.ping of Redis<ConnectionPool<Connection
<host=localhost,port=6379,db=0>>>>

ping2 = redis2.ping
ping2
<bound method Redis.ping of Redis<ConnectionPool<Connection
<host=localhost,port=6379,db=0>>>>

# Hacemos un ejemplo previo
redis1.set('test','examen')
redis1.get('test')
```

Lo cual arroja como salida:

```
b'examen'
```

```
# Estas funciones sirven como auxiliares, la principal se encuentra
# mas adelante

os.system('cls')
clear = lambda:os.system('cls')

def menu_inputs():
    """
    @Author: MLS y JJTL
    summary: Menu que imprime la interfaz grafica para ser un puente
    entre Python y Redis.
    Parameters: NULL
    Returns: Pantalla de la interfaz grafica
    """
    print("")
    print ('+'*20," Opciones ", '+'*20, '\n')
    print("0.- Agregar Registro")
    print("1.- Borrar Registro")
    print("2.- Ver todos los registros de un usuario")
    print("3.- Ver URL o Nickname de un usuario")
    print("4.- Ver la Wishlist")
    print("5.- Salir")

def menu_inputs_4_3():
    """
    @Author: MLS y JJTL
    summary: Menu para ver URL o Nickname de un usuario.
    Parameters: NULL
    Returns: Pantalla de la interfaz grafica
    """
    print("")
    print ('+'*20," Opciones ", '+'*20, '\n')
    print("0.- Ingrese el URL para ver su Nickname")
    print("1.- Ingrese el Nickname para ver su URL")
    print("2.- Salir")

def op4_3():
    while True:
        menu_inputs_4_3()
        optionM = input("Que desea hacer? >> ")
        #clear()
        if optionM == '0': #Agregar Registro
```

```
            id = int(input("Ingrese el identificador: "))
            url = str(input("Ingrese el URL: "))
            print("El nickname del URL de este usuario es: ",
            redis2.hmget(id,'URL'))

        elif optionM == '1':
            id = int(input("Ingrese el identificador: "))
            nickname = str(input("Ingrese el Nickname: "))
            print("El URL del nickname de este usuario es: ",
            redis2.hmget(id,'Nickname'))

        elif optionM == '2':
            break;

        else:
            print("Opcion no valida. Por favor, Vuelva a intentarlo.")


def op4():
    while True:
        menu_inputs()
        optionM = input(" Que  desea hacer? >> ")
        #clear()
        if optionM == '0': #Agregar Registro

            id = str(input("Ingrese el identificador: "))
            url = input("Ingrese la URL: ")
            nickname = input("Ingrese un nickname al URL: ")
            clasificacion = input("Ingrese la clasificacion: ")
            categoria = input("Ingrese la categoria: ")

            #Creamos los conjuntos para cada uno de los datos
            de los usuarios.
            redis1.sadd('URL:'+id,url)
            redis1.sadd('Nickname:'+id,nickname)
            redis1.sadd('Clasificacion:'+id,clasificacion)
            redis1.sadd('Categoria:'+id,categoria)
            # Creamos Hashes solo para obtener posteriormente el url,
            dado el
            # nickname o viceversa (redis2 solo se encarga de esto)
            # Esto es, el valor de la llave URL es el nickname y el
            # valor de la llave nickname es el URL
            redis2.hmset(id,{'URL':nickname,'Nickname':url})

             #Datos posibles de ingreso: Si - para agregar a wishlist;
             # No - caso Contrario
```

```
        wish = str(input("Desea agregar el registro a su Wishlist?\n"))

        if(wish == 'Si'):
            redis1.sadd('URLW:'+id,url)
            redis1.sadd('NicknameW:'+id,nickname)
            redis1.sadd('ClasificacionW:'+id,clasificacion)
            redis1.sadd('CategoriaW:'+id,categoria)


    elif optionM == '1': #Borrar registro
        id = input("Ingrese el identificador: ")
        url = input("Ingrese la URL: ")
        nickname = input("Ingrese un nickname al URL: ")
        clasificacion = input("Ingrese la clasificacion: ")
        categoria = input("Ingrese la categoria: ")

        redis1.srem('URL:'+id,url)
        redis1.srem('Nickname:'+id,nickname)
        redis1.srem('Clasificacion:'+id,clasificacion)
        redis1.srem('Categoria:'+id,categoria)

        #Eliminacion de los elementos de la wishlist
        wish = str(input("Desea removerlos tambien de la wishlist?\n"))

        if(wish == "Si"):
            redis1.srem('URLW:'+id,url)
            redis1.srem('NicknameW:'+id,nickname)
            redis1.srem('ClasificacionW:'+id,clasificacion)
            redis1.srem('CategoriaW:'+id,categoria)

    elif optionM == '2': #Ver registros, los cuales se van asignando
    # en diferentes variables dependiendo el id asignado.
        id = int(input("Ingrese el identificador: "))

        print("Los datos para este usuario son: ")
        sURL = redis1.smembers(f'URL:{id}')
        sNickname = redis1.smembers(f'Nickname:{id}')
        sClasificacion = redis1.smembers(f'Clasificacion:{id}')
        sCategoria = redis1.smembers(f'Categoria:{id}')

        #Imprimimos los registros conforme se fueron insertando
        print(f'Los URLs son: {sURL} ')
        print(f'Los nickname son: {sNickname}')
        print(f'Las clasificaciones son: {sClasificacion}')
        print(f'Las categorias son: {sCategoria}')
```

```
        elif optionM == '3': # Ver URL o Nickname de un usuario
            op4_3()
        elif optionM == '4': #Presentar todos los registros de la wishlist

            id = int(input("Ingrese el identificador: "))

            print("Los datos para este usuario son: ")
            sURL = redis1.smembers(f'URLW:{id}')
            sNickname = redis1.smembers(f'NicknameW:{id}')
            sClasificacion = redis1.smembers(f'ClasificacionW:{id}')
            sCategoria = redis1.smembers(f'CategoriaW:{id}')

            #Imprimimos los registros con forme se fueron insertando
            print(f'Los URLs son: {sURL} ')
            print(f'Los nickname son: {sNickname}')
            print(f'Las clasificaciones son: {sClasificacion}')
            print(f'Las categorias son: {sCategoria}')


        elif optionM == '5':
            break;

        else:
            print("Opcion no valida. Por favor, Vuelva a intentarlo.")



        input("\n\n Presione ENTER para continuar.")
        clear()
```

Cabe mencionar que usa estructuras de datos diferentes:

- Usuarios.
- URL, nickname, clasificación y categoría son guardados en conjuntos, porque en ellos es fácil poder realizar operaciones clásicas de conjuntos.

Esta aplicación funciona como un menú y está conformado por 6 opciones; recibe los datos de los usuarios y los guarda en datos de la siguiente manera:

```
# Funciones auxiliares para ingresar datos por consola en Python.
os.system('cls')
clear = lambda:os.system('cls')

# Observacion: La aplicacion no permite acentos ni comas, por lo que
# al registrarlas en Redis se deforman.

def menu_aplication():
    """
    @Author: MSL y JJTL
```

```
    summary: Menu que imprime la interfaz grafica para ser un puente
    #entre Python y Redis.
    Parameters: NULL
    Returns: Pantalla de la interfaz grafica
    """
    print("")
    print ('+'*20," MENU ", '+'*20, '\n')
    print ("0.- Ingresar un nuevo usuario")
    print ("1.- Editar datos de un usuario")
    print ("2.- Buscar usuario")
    print ("3.- Cerrar sesion de un usuario")
    print ("4.- Registros")
    print ("5.- Comparar registros")
    print ("6.- Salir")
    print ("")


# Empezamos con la interfaz grafica y todo su desarrollo
while True:
    # Inicializamos llamando al menu y eligiendo una opcion
    menu_aplication()
    opMenu = input("Ingrese una opcion >> ")
    clear()

    if opMenu == '0': # Ingresar un nuevo usuario
        print()
        print('-'*5," Ingrese el usuario:", '-'*5, '\n')
        # Ingresar un id para identificar; Numero de cuenta
        # para hacerlo practico
        id = str(input("Identificador: "))
        # Nombre del usuario
        usr = input("Usuario: ")
        # Ingresamos una ciudad: Nada mas para hacer bulto
        ciudad = input("Ciudad: ")
        redis1.hmset(id, {'user':usr, 'ciudad':ciudad})

    elif opMenu == '1': # Editar datos de usuario
        print()
        print('-'*5, "Editar datos de usuarios", '-'*5, '\n')
        id = str(input("Ingrese el identificador del usuario existente: "))
        # Ingresamos los comandos para ver si es que el usuario existe
        if (redis1.scan(0,id)):
            usr = input("Nuevo usuario: ")
            ciudad = input("Nueva ciudad: ")
            redis1.hmset(id, {'user':usr, 'ciudad':ciudad})
        else:
```

```
            print('El usuario no existe.')

    elif opMenu == '2': # Buscar Usuario.
        # En caso de hayamos cerrado la sesion de un usuario
        # o que el identificador no exista, usamos un try -
        # except para indicar que no se encuentra el deter-
        # minado identificador.
        try:
            print()
            print('-'*10, 'Buscador', '-'*10, '\n')
            id = str(input('Ingrese el identificador: '))
            print(redis1.hgetall(id), '\n')
            print('Usuario: ', redis1.hget(id, 'user').decode())
            print('Ciudad: ', redis1.hget(id, 'ciudad').decode())
        except AttributeError:
            print("El identificador que usted ingreso no se encuentra.")

    elif opMenu == '3': # Cerrar sesion de usuario
        print()
        print('-'*5, "Cerrar sesion de un usuario", '-'*5, '\n')
        id = input("Identificador: ")
        if (redis1.scan(0, id)):
            #borramos todos los urls de este usuario y asi sucesivamente.
            redis1.delete(id)
            redis1.delete('URL:'+id)
            redis1.delete('Nickname:'+id)
            redis1.delete('Categoria:'+id)
            redis1.delete('Clasificacion:'+id)
        else:
            print("El usuario no existe.")

    elif opMenu == '4': # Ingresar registros
        # Empezamos a trabajar con los registros los cuales los
        # guardaremos en conjuntos porque son manipulables
        op5()

    elif opMenu == '5': # Comparamos los registros para encontrar
    # matches comunes de la wishlist
        print()
        usuario1 = str(input("Ingrese el identificador del usuario1: "))
        usuario2 = str(input("Ingrese el identificador del usuario2: "))
        # Creamos un nuevo conjunto en el cual guardaremos los
        # datos de la interseccion
        int_dato = 'Interseccion' + usuario1 + usuario2

        # Definimos los conjuntos delas intersecciones
```

```
        url_inter = redis1.sinterstore('URLW'+int_dato,'URLW:'+usuario1,
        'URLW:'+usuario2)
        nick_inter = redis1.sinterstore('NicknameW'+int_dato,
        'NicknameW:'+usuario1,'NicknameW:'+usuario2)
        cate_inter = redis1.sinterstore('CategoriaW'+int_dato,
        'CategoriaW:'+usuario1, 'CategoriaW:'+usuario2)
        clasif_inter = redis1.sinterstore('ClasificacionW'+int_dato,
        'ClasificacionW:'+usuario1, 'ClasificacionW:' + usuario2)

        # Mandamos a llamar el nuevo conjunto donde se almacenaron
        # las intersecciones
        sURL = redis1.smembers(f'URLW{int_dato}')
        sNickname = redis1.smembers(f'NicknameW{int_dato}')
        sClasificacion = redis1.smembers(f'ClasificacionW{int_dato}')
        sCategoria = redis1.smembers(f'CategoriaW{int_dato}')

        # Imprimimos los registros comunes para todos los registrados
        # por cada uno de los usuarios
        print(f'Los URLs comunes son: {sURL} ')
        print(f'Los nicknames comunes son: {sNickname}')
        print(f'Las clasificaciones comunes son: {sClasificacion}')
        print(f'Las categorias comunes son: {sCategoria}')

    elif opMenu == '6':
        break;

    else:
        print("Opcion no valida. Por favor, Vuelva a intentarlo.")
        menu_aplication()


    input("\n\n Presione ENTER para continuar.")
    clear()


    input("\n\n Presione ENTER para continuar.")
    clear()
```

4 Funcionamiento del programa

Inicializamos el programa donde se puede mostrar como es que el programa pide al usuario una opción para trabajar. Para mostrar el funcionamiento en orden, seleccionamos la primera opción:

Figure 1: Ingresar un nuevo usuario.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 0

-----  Ingrese el usuario: -----

Identificador: 1712
Usuario: jesus tapia
Ciudad: tulancingo


 Presione ENTER para continuar.
```

Figure 2: Buscar usuario.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 2

---------- Buscador ----------

Ingrese el identificador: 1712
{b'user': b'jesus tapia', b'ciudad': b'tulancingo'}

Usuario:  jesus tapia
Ciudad:  tulancingo


 Presione ENTER para continuar.
```

Editamos la información del usuario:
Mostramos la información (actualizada) del usuario:

Figure 3: Editar datos de un usuario.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 1

----- Editar datos de usuarios -----

Ingrese el identificador del usuario existente: 1712
Nuevo usuario: jose de jesus tapia lopez
Nueva ciudad: pachuca


 Presione ENTER para continuar.
```

Figure 4: Buscar usuario (actualizado).

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 2

---------- Buscador ----------

Ingrese el identificador: 1712
{b'user': b'jose de jesus tapia lopez', b'ciudad': b'pachuca'}

Usuario:  jose de jesus tapia lopez
Ciudad:  pachuca
```

Para la opción 3, creamos un nuevo usuario para posteriormente cerrar su sesión:

Figure 5: Creamos un nuevo usuario

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 0

-----  Ingrese el usuario: -----

Identificador: 1112
Usuario: samy rodriguez
Ciudad: santiago
```

Figure 6: Cerramos la sesión del usuario recién creado.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

 0.- Ingresar un nuevo usuario
 1.- Editar datos de un usuario
 2.- Buscar usuario
 3.- Cerrar sesión de un usuario
 4.- Registros
 5.- Comparar registros
 6.- Salir

 Ingrese una opción >> 3

 ----- Cerrar Sesión de un usuario -----

 Identificador: 1112


  Presione ENTER para continuar.
```

Y verificamos que ya no se encuentra:

Figure 7: Verificamos que sí cerramos su sesión.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 2

---------- Buscador ----------

Ingrese el identificador: 1112
{}

El identificador que usted ingresó no se encuentra.
```

En la opción 4 podemos modificar (agregar, borrar, revisar, etc) los registros de los URL y sus Nickname conforme al usuario; y vamos revisando en orden comenzando entonces con la primera opción:

Figure 8: Agregamos registro a un usuario.

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 4

++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 0


Ingrese el identificador: 1712
Ingrese la URL: pilarang.unam.mx
Ingrese un nickname al URL: pilar
Ingrese la clasificación: maestros
Ingrese la categoría: publico
¿Desea agregar el registro a su Wishlist?
Si


 Presione ENTER para continuar.
```

Figure 9: Agregamos otro registro a un usuario.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 0
Ingrese el identificador: 1712
Ingrese la URL: alejandropimentel.iimas.unam.mx
Ingrese un nickname al URL: profpimentel
Ingrese la clasificación: privado
Ingrese la categoría: profesores
¿Desea agregar el registro a su Wishlist?
Si


 Presione ENTER para continuar.
```

Figure 10: Vemos los registros de un usuario.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 2
Ingrese el identificador: 1712
Los datos para este usuario son:
Los URLs son: {b'pilarang.unam.mx', b'alejandropimentel.iimas.unam.mx'}
Los nickname son: {b'profpimentel', b'pilar'}
Las clasificaciones son: {b'maestros', b'privado'}
Las categorías son: {b'profesores', b'publico'}


 Presione ENTER para continuar.
```

Figure 11: Borramos un registro a un usuario.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 1
Ingrese el identificador: 1712
Ingrese la URL: pilarang.unam.mx
Ingrese un nickname al URL: pilar
Ingrese la clasificación: maestros
Ingrese la categoría: publico
¿Desea removerlos también de la wishlist?
Si
```

Figure 12: Verificamos que el registro sí se borró.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 2
Ingrese el identificador: 1712
Los datos para este usuario son:
Los URLs son: {b'alejandropimentel.iimas.unam.mx'}
Los nickname son: {b'profpimentel'}
Las clasificaciones son: {b'privado'}
Las categorías son: {b'profesores'}


 Presione ENTER para continuar.
```

Figure 13: URL o Nickname de un usuario.

```
 Presione ENTER para continuar.

++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 3
```

Figure 14: Ver Nickname, dado el URL.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Ingrese el URL para ver su Nickname
1.- Ingrese el Nickname para ver su URL
2.- Salir
¿Qué desea hacer? >> 0
Ingrese el identificador: 1712
Ingrese el URL: alejandropimentel.iimas.unam.mx
El nickname del URL de este usuario es:  [b'profpimentel']
```

Figure 15: Ver URL, dado el Nickname.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Ingrese el URL para ver su Nickname
1.- Ingrese el Nickname para ver su URL
2.- Salir
¿Qué desea hacer? >> 1
Ingrese el identificador: 1712
Ingrese el Nickname: profpimentel
El URL del nickname de este usuario es:  [b'alejandropimentel.iimas.unam.mx']
```

Agregamos un nuevo registro al usuario con el que hemos estado ejemplificando esta práctica para revisar su Wishlist:

Figure 16: Agregando otro registro al usuario.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 0
Ingrese el identificador: 1712
Ingrese la URL: actuaria.ciencias.unam.mx
Ingrese un nickname al URL: fc_actuaria
Ingrese la clasificación: publico
Ingrese la categoría: educacion
¿Desea agregar el registro a su Wishlist?
Si


 Presione ENTER para continuar.
```

Figure 17: Viendo su Wishlist.

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 4
Ingrese el identificador: 1712
Los datos para este usuario son:
Los URLs son: {b'alejandropimentel.iimas.unam.mx', b'actuaria.ciencias.unam.mx'}
Los nickname son: {b'profpimentel', b'fc_actuaria'}
Las clasificaciones son: {b'publico', b'privado'}
Las categorías son: {b'profesores', b'educacion'}


 Presione ENTER para continuar.
```

Agregamos un nuevo usuario y así verificar la opción 5 del menú principal para comparar registros entre usuarios:

Figure 18: Agregando otro usuario

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 0

-----  Ingrese el usuario: -----

Identificador: 0911
Usuario: mary rico
Ciudad: san luis


 Presione ENTER para continuar.
```

Figure 19: Agregando registros a este usuario

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 4
```

```
++++++++++++++++++++  Opciones  ++++++++++++++++++++

0.- Agregar Registro
1.- Borrar Registro
2.- Ver todos los registros de un usuario
3.- Ver URL o Nickname de un usuario
4.- Ver la Wishlist
5.- Salir
¿Qué desea hacer? >> 0
Ingrese el identificador: 0911
Ingrese la URL: alejandropimentel.iimas.unam.mx
Ingrese un nickname al URL: profpimentel
Ingrese la clasificación: privado
Ingrese la categoría: profesores
¿Desea agregar el registro a su Wishlist?
Si
```

Figure 20: Comparando registros de dos usuarios

```
++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 5

Ingrese el identificador del usuario1: 1712
Ingrese el identificador del usuario2: 0911
Los URLs comunes son: {b'alejandropimentel.iimas.unam.mx'}
Los nicknames comunes son: {b'profpimentel'}
Las clasificaciones comunes son: {b'privado'}
Las categorías comunes son: {b'profesores'}
```

Finalmente, concluimos esta sección saliendo de la aplicación:

Figure 21: Fin de la aplicación

```
 Presione ENTER para continuar.

++++++++++++++++++++  MENU  ++++++++++++++++++++

0.- Ingresar un nuevo usuario
1.- Editar datos de un usuario
2.- Buscar usuario
3.- Cerrar sesión de un usuario
4.- Registros
5.- Comparar registros
6.- Salir

Ingrese una opción >> 6
```

5 Conclusión

Al trabajar con Redis como una base de datos no relacional de tipo llave-valor pudimos encontrar varias ventajas y desventajas de este tipo de bases de datos como las siguientes:

Ventajas

- Al ser una base de datos que corre en RAM, el escribir y consultar información en estas bases de datos se vuelve una tarea muy rápida lo cual las vuelven ideales para tareas como procesar información en tiempo real.
- Al ser bases de datos ligeras, organizar su estructura en forma de clúster es relativamente fácil y es precisamente gracias a la estructura de anillo que presentan, y a los registros en tablas hash se puede escalar este tipo de bases de datos para manejar un gran flujo de información de manera dinámica.
- Redis utiliza una arquitectura maestro-esclavo y admite la replicación asíncrona mediante la que los datos se replican en numerosos servidores esclavos. De este modo, se logra una mejora en el desempeño de lectura (ya que las lecturas se pueden repartir entre servidores) y de recuperación cuando el servidor principal sufre un fallo.

Desventajas

- Redis es una tienda en memoria: todos sus datos deben caber en la memoria, por lo que para almacenamiento de cantidades muy grande de información Redis debe de consumir mucha memoria RAM lo cual hace que a mayor sea la información procesada Redis vaya perdiendo su principal atractivo de velocidad debido a la RAM ocupada.
- Redis es un servidor de estructura de datos. No hay lenguaje de consulta (solo comandos) y no hay soporte para un álgebra relacional. No puede enviar consultas ad-hoc (como puede usar SQL en un RDBMS). El desarrollador debe anticipar todos los accesos a los datos, y deben diseñarse rutas de acceso a los datos adecuadas. Se pierde mucha flexibilidad.
- Las estructuras de datos que presenta Redis pueden ser no precisamente las más adecuadas debido a que para guardar cierto tipo de datos para poder después realizar una consulta necesitamos almacenar las consultas en otro tipo de registro, como en el caso de los datos tipo SET que al realizar una consulta de datos en común; esta consulta se tiene que guardar en una nueva llave así que al realizar muchas consultas nos gastaremos muchisima memoria RAM en esto lo cual presenta una gran desventaja.
- Una instancia única de Redis no es escalable. Solo se ejecuta en un núcleo de CPU en modo de subproceso único. Para obtener escalabilidad, se deben implementar e iniciar varias instancias de Redis. La distribución y el fragmentación se realizan en el lado del cliente (es decir, el desarrollador tiene que ocuparse de ellos). Si los compara con una instancia única de Redis, la mayoría de los RDBMS proporcionan más escalabilidad (por lo general, proporcionan paralelismo en el nivel de conexión). Son multiprocesos (Oracle, PostgreSQL, ...) o multiprocesos (MySQL, Microsoft SQL Server, ...), aprovechando las ventajas de las máquinas con múltiples núcleos.

Redis incorpora varias herramientas que facilitan y aceleran el desarrollo y las operaciones, incluidas Pub/Sub, para publicar mensajes en canales, que se entregan a suscriptores, lo que es ideal para sistemas de chat y mensajería; las claves TTL, que indican un tiempo de vida determinado, tras el que se eliminan a sí mismas, lo que resulta útil para evitar llenar las bases de datos de datos no necesarios; por lo que brinda mucha versatilidad.
En general redis presenta muy buenas caracteristicas para trabajar sobre ella como base de datos. Gracias a su velocidad y facilidad de uso, Redis es una opción popular para aplicaciones web, móviles, de juegos, de tecnología publicitaria y de IoT que requieren el mejor desempeño de su clase, en ella se puede trabajar con datos en tiempo real pues la necesidad de consultar información a gran velocidad es una caracteristica de estas bases tipo clave-valor, por sus carácteristicas redis por lo general es utilizado como un componente que trabaja en conjunto con otro tipo de bases dedatos donde necesitamos llamar información al instante (como memoria cache), entre las multiples aplicaciones que utilizan redis por ejemplo podemos encontrar a Twitter o Amazon sólo por mencionar unos ejemplos, por lo que redis es una excelente opción para trabajar con datos que presentan una gran volatilidad.

6 Bibliografía

https://realpython.com/python-redis/
https://redis.io/commands#set