Estructuras de Datos y Base de Datos Relacionales

Brian Anco[1], Ericka Martínez[2], and José Contreras[3]

[1]Universidad Privada de Tacna, Perú. Email: briancoc@upt.pe
[2]Universidad Privada de Tacna, Perú. Email: erimartinezy@upt.pe
[3]Universidad Privada de Tacna, Perú. Email: joscontrerasm@upt.pe

Resumen

Las bases de datos son el método preferido para el almacenamiento estructurado de datos. Por Ello en este artículo se va Analizar de distintas bibliografías, tratando las estructuras de Datos, BD relacionales, Tipo de Estructuras de Datos, dentro de ello también se va tomar en cuenta: pila, árbol, lista enlazada, etc.
Un sistema de gestión de bases de datos moderno es un sistema de software complejo que aprovecha muchos algoritmos sofisticados, por ejemplo, para evaluar operaciones relacionales, para proporcionar eficientes acceso a los datos. Viendo este tema extenso sobre la Estructura de Datos en base de Datos Relacionales, vamos definir su amplia estructura y que tipos de datos tiene.

Abstract

Databases are the preferred method for structured data storage. For this reason, this article will analyze different bibliographies, dealing with Data structures, relational DB, Type of Data Structures, within this it will also take into account: stack, tree, linked list, etc.
A modern database management system is a complex software system that takes advantage of many sophisticated algorithms, for example, to evaluate relational operations, to provide efficient access to data. Seeing this extensive topic about the Data Structure in Relational Data, we are going to define its broad structure and what types of data it has.

1 INTRODUCCIÓN

Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base.

Una base de datos relacional consiste en un conjunto de tablas, a cada una de las cuales se le asigna un nombre exclusivo. Cada fila de la tabla representa una relación entre un conjunto de valores. Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación, del que toma su nombre el modelo de datos relacional.

2 DESARROLLO

2.1 Estructuras de Datos

Se puede definir a la estructura de datos como una forma de representar la información de manera organizada. Estas no se limitan a únicamente mostrar la información, sino también poseen un comportamiento interno y cumplen con reglas específicas en cada estructura. Buscar la eficiencia de los programas sigue siendo una de las características principales que deben tomar en cuenta los desarrolladores durante su elaboración, es por ello que el uso de las estructuras de datos en función a las necesidades del programa, representan una parte fundamental para alcanzar el objetivo de un programa eficiente.

2.2 Bases de Datos Relacionales

Una base de datos relacional nos permite almacenar y tener acceso a diferentes puntos de datos que están relacionados entre sí. Basada en el modelo relacional, cuenta con una manera directa e intuitiva de representar los datos en las tablas. Son altamente dinámicas, y la información se organiza en pequeñas partes integradas mediante identificadores, es por ello que tienen una mayor capacidad de almacenamiento, y son menos vulnerables a posibles fallas.

2.3 Tipos de Estructuras de Datos

2.3.1 Arrays

Un array o arreglol (lista o tabla) es una secuencia de datos del mismo tipor Los datos se llaman elementos del array y se numeran consecutivamente 0, 1, 2, 3 . . . El tipo de elementos almacei nados en el array puede ser cualquier dato simple de Java o de un tipo previamente declarado como una clase. Normalmente, el array se utiliza para almacenar tipos tales como char, int: o float. (Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=87.)

Los datos almacenados de un array son denominados elementos, y el número de elementos de un array son denominados tamaño o rango del vector. Para acceder a cada uno de los elementos de manera individual, se emplea un índice que definirá la posición del elemento dentro del vector, estos índices deben ser números enteros positivos. Gracias a ello, esta estructura nos permite acceder a sus elementos sin tener que consultar a elementos anteriores o posteriores, por lo que es más adecuada cuando se requiera acceder a los datos de manera aleatoria e impredecible.

```java
public static void main(String[] args)throws IOException {
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Ingresa la cantidad de notas a registrar: ");
    int n=0;
    n=Integer.parseInt(reader.readLine());
    int numeros[]=new int[n]; //Definimos el array "numeros" y asignamos el tamaño "n"

    for(int i=0;i<n;i++){ //Utilizamos un For para recorrer los elementos del array
        System.out.println("Ingresa una nota: ");
        numeros[i]=Integer.parseInt(reader.readLine());
        //Convertimos la nota a tipo entero y la almacenamos en "numeros" utilizando como
        //índice a la variable "i" declarada en For, hasta alcanzar el tamaño definido anteriormente.
    }
    for(int i=0;i<n;i++){ //Utilizamos un For para recorrer los elementos del array
        //Comparamos cada valor del vector "numeros" de manera independiente, para imprimir el resultado.
        if(numeros[i]>=11){
            System.out.println(numeros[i]+" - Aprobado");
        }
        else{
            System.out.println(numeros[i]+" - Desaprobado");
        }
    }
}
```

Figure 1: Declaración de un Array en Java.

```
int Vector[]=new int[Tamanio];
```

Figure 2: Declaración de un Array tipo entero en Java.

2.3.2 Pilas

Una pila es una estructura de datos de entradas ordenadas que sólo se pueden introducir y eliminar por un extremo, llamado cima. La pila se puede implementar guardando los elementos en un array, en cuyo caso su dimensión o longitud es fija. También se puede utilizar un Vector para almacenar los elementos. Otra forma de implementación consiste en construir una lista enlazada, de modo que cada elemento de la pila forma un nodo de la listar(Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=294.)

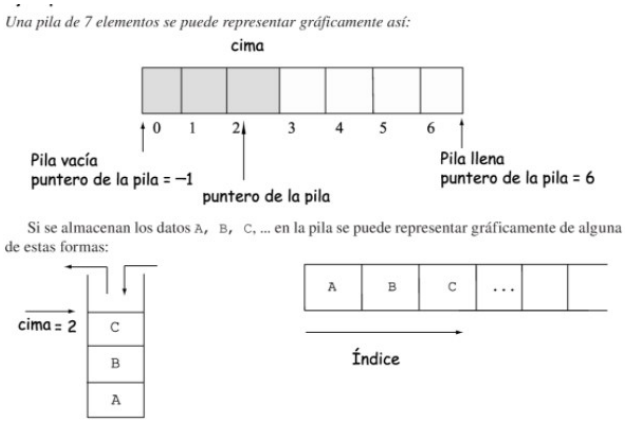

Figure 3: Estructura de una Pila.
Fuente: Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java, pag 296.

```
import TipoPila.PilaLineal;
import java.io.*;

class EjemploPila
{
   public static void main(String [] a)
   {
      PilaLineal pila;
      int x;
      final int CLAVE = -1;
      BufferedReader entrada = new BufferedReader(
                           new InputStreamReader(System.in));

      System.out.println("Teclea los elemento (termina con -1).");
      try {
         pila = new PilaLineal(); // crea pila vacía
         do {
            x = Integer.parseInt(entrada.readLine());
            pila.insertar(x);
         }while (x != CLAVE);

         System.out.println("Elementos de la Pila: ");
         while (!pila.pilaVacia())
         {
            x = pila.quitar();
            System.out.print(x + " ");
         }
      }
      catch (Exception er)
      {
         System.err.println("Excepcion: " + er);
      }
   }
}
```

Figure 4: Desarrollo del ejemplo utilizando una Pila en Java.

2.3.3 Colas

Una cola es una estructura de datos cuyos elementos mantienen un cierto orden, de tal modo que sólo se pueden añadir elementos por un extremo, final de la cola, y eliminar o extraer por el otro extremo, llamado frente. (Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=319.)

- La realización de una cola con un array lineal es notablemente ineficiente. se puede alcanzar la condición de cola llena existiendo elementos del array sin ocupar. Se aconseja no utilizar esta implementación.
- Las bicolas son colas dobles, las operaciones básicas de insertar y retirar elementos se pueden realizar por los dos extremos, A veces se ponen restricciones de entrada o de salida realizar por algún extremo. Una bicola es realmente una extensión de una cola.

```
package TipoCola;

public class ColaLineal
{
   private static fin int MAXTAMQ = 39;
   protected int frente;
   protected int fin;
   protected TipoDeDato [] listaCola;

   public ColaLineal()
   {
      frente =  0;
      fin = -1;
      listaCola = new TipoDeDato [MAXTAMQ];
   }
      // operaciones de modificación de la cola
   public void insertar(TipoDeDato elemento) throws Exception
   {
      if (!colaLlena())
      {
         listaCola[++fin] = elemento;
      }
      else
         throw new Exception("Overflow en la cola");
   }
   public TipoDeDato quitar() throws Exception
   {
      if (!colaVacia())
      {
         return listaCola[frente++];
      }
```

Figure 5: Declaración de la clase Cola - Parte 1

```
      else
         throw new Exception("Cola vacia ");
   }
      //vacía la cola
   public void borrarCola()
   {
      frente =  0;
      fin = -1;
   }
      // acceso a la cola
   public TipoDeDato frenteCola() throws Exception
   {
      if (!colaVacia())
      {
         return listaCola[frente];
      }
      else
         throw new Exception("Cola vacia ");
   }
      // métodos de verificación del estado de la cola
   public boolean colaVacia()
   {
      return frente > fin;
   }
   public boolean colaLlena()
   {
      return fin == MAXTAMQ-1;
   }
}
```

Figure 6: Declaración de la clase Cola - Parte 2

2.3.4 Listas Enlazadas

Una lista enlazada es una colección o secuencia de elementos dispuestos uno detras de otro, en la que cada elemento se conecta al siguiente elemento por un "enlace" o "referencia", La idea básica consiste en construir una lista cuyos elementos, llamados nodos, se componen de dos partes (campos): la primera parte contiene la información y es, por consiguiente, un valor de un tipo generico (denominado Dato, TipoElemento, Info, etc), y la segunda parte es una referencia (denominado enlace o sgte) que apunta (enlaza) al siguiente elemento de la lista. (Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=251.)

Las listas se pueden dividir en cuatro categorias:
-Listas simplemente enlazadas. Cada nodo (elemento) contiene un único enlace que lo conecta al nodo siguiente o nodo sucesort La lista es eficiente en recorridos directos ("adelante").
- Listas doblemente enlazadas. Cada nodo contiene dos enlaces, uno a su nodo predecesor y otro a su nodo sucesort La lista es eficiente tanto en recorrido directo ("adelante") como en recorrido inverso ("atras").
- Lista circular simplemente enlazada. Una lista enlazada simplemente en la que el último elemento (cola) se enlaza al primer elemento (cabeza) de tal modo que la lista puede ser recorrida de modo circular ("en anillo").
- Lista circular doblemente enlazada. Una lista doblemente enlazada en Ia que el último elemento se enlaza al primer elemento y viceversa. Esta lista se puede recorrer de modo circular ("en anillo") tanto en dirección directa ("adelante") como inversa ("atrás").
Ejemplo:

```
public void eliminar(int entrada){
    Nodo actual, anterior;
    Boolean encontrado;
    // inicializamos los apuntadores
    actual = primero;
    anterior = null;
    encontrado = false;
    // búsqueda del nodo  y del nodo anterior
    while ((actual != null) && (! encontrado)){
        encontrado = (actual.dato == entrada);
        // con objetos: actual.dato.equals(entrada)
        if(!encontrado){
            anterior = actual;
            actual = actual.enlace;
        }
    }
    // Enlace del nodo anterior con el siguiente
    if (actual != null){
        // Distingue entre que el nodo sea el cabecera
        // o el resto de la lista
        if( actual == primero){
            primero = actual.enlace;
        }
        else{
            anterior.enlace = actual.enlace;
        }
        actual = null; // No es necesario al ser una variable local
    }
}
```

Figure 7: Ejemplo de Listas Enlazadas

2.3.5 Árboles

Un árbol consta de un conjunto finito de elementos, denominados nodos y de un conjunto finito de líneas dirigidas, denominadas ramas, que conectan los nodos. El número de ramas asociado con un nodo es el grado del nodo. (Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=393.)

Un arbol es un conjunto de uno o más nodos, tales que:

A) Hay un nodo diseñado especialmente llamado raiz.
B) Los nodos restantes se dividen en n¿=0 conjuntos distintos, T1 ... Tn, tal que cada uno de estos conjuntos es un árbol. A T1 ... Tn se les denomina subarboles del raiz. Un árbol binario no puede tener más de dos subárboles. El

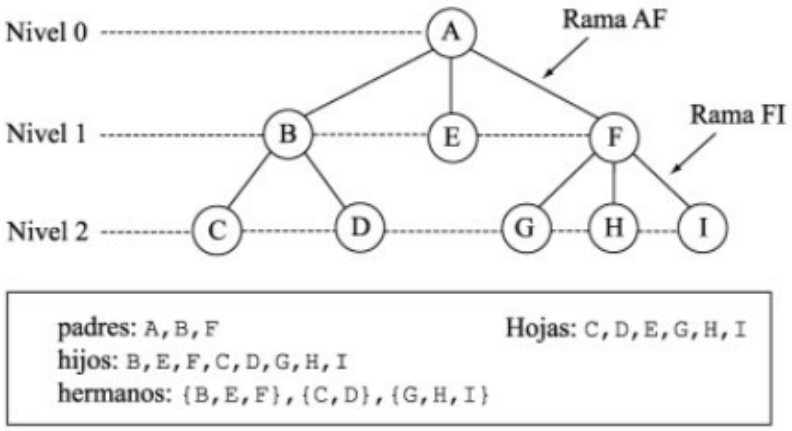

Figure 8: Terminología de árboles

recorrido de un árbol supone visitar cada nodo sólo una vez. En el recorrido preorden, el raíz se procesa antes que los subárboles izquierdo y derecho. Los árboles binarios ordenados, la búsqueda de una clave da lugar a un camino de búsqueda, de tal forma que baja por la rama izquierda si la clave buscada es menor que la clave de la raíz o baja por la rama derecha si la clave es mayor.

```
//Metodos para manipular el Arbol Binario
public void preorden(Nodo p){
   if (p!=null){
        s =  s + " " + p.info;
        preorden(p.izq);
        preorden(p.der);
   }
}
public void inorden(Nodo p){
   if (p!=null){
        inorden(p.izq);
        s =  s + " " + p.info;
        inorden(p.der);
   }
}
public void posorden(Nodo p){
   if (p!=null){
        posorden(p.izq);
        posorden(p.der);
        s =  s + " " + p.info;

   }
}
```

Figure 9: Ejemplo de Árboles

2.3.6 Grafos

Un grato permite modelar relaciones arbitrarias entre objetos. Un grato G = (V, A) es un par formado por un conjunto de vértices o nodos, V, y un conjunto de arcos o aristas, A. Cada arco es el par (u, w) , siendo u, w dos vértices

relacionados. (Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java. Madrid etc, Spain: McGraw-Hill España. Recuperado de https://elibro.net/es/ereader/bibliotecaupt/50117?page=458.)

La longitud de un camino es el número de arcos del camino. En un grato valorado, la longitud del camino con pesos es la suma de los pesos de los arcos en el camino.
La matriz de adyacencia representa los arcos, relaciones entre un par de nodos de un grafo. Es una matriz de unos y ceros, que indican si dos vértices son adyacentes o no. Es una grado valorado, cada elemento representa el peso de la arista y por ello se la denomina matriz de pesos.

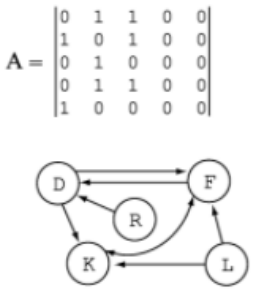

Figure 10: Grafo dirigido con los vértices
Fuente: Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java, pag 461.

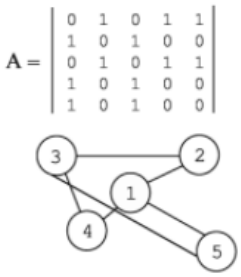

Figure 11: Grafo no dirigido con 5 vértices
Fuente: Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java, pag 461.

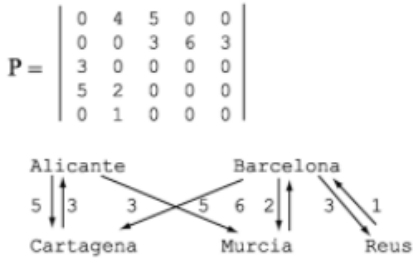

Figure 12: Grafo dirigido con factor de peso
Fuente: Zohonero Martínez, I. y Joyanes Aguilar, L. (2008). Estructuras de datos en Java, pag 461.

3 CONCLUSIONES

Con este Articulo damos con concluido la estructura de Base de Datos Relacionales teniendo en cuenta las Bibliografías dadas de nuestra Asignatura, por ello analizando damos a conocer que existen varios tipos de estructuras de datos para

la BD, de acuerdo a qué tipo de BD deseamos realizar. Por ello aquí damos a entender en cómo se usa cada estructura y en cómo aplicar, a la vez colocamos conceptos concretos que dan a entender claramente nuestros objetivos.

4 RECOMENDACIONES

Las Recomendaciones que podemos dar es tal que el tema ya que es muy extenso y con diversas bibliografías dadas, primero aplicamos en rescatar lo más importante de cada uno, luego hemos unido cada tema en el presente artículo, tuvimos temas complejos pero nada imposible.
Fuera de ello, se recomienda tener una orden previa para poder hacer un artículo y siguiendo la plantilla real de Latex que pide.