# Diseño de la base de datos basada en grafos

Definimos un grafo **G** como el par ordenado **(V,E)** conformado por conjuntos **E** y **V**. Donde **V** es un conjunto de vértices o nodos, donde cada nodo representa alguna estructura de datos o información de interés. El conjunto E es un conjunto de pares ordenados **(v1,v2)**, que llamamos aristas, con **v1,v2** elementos de **V**. **(v1,v2)** representa una relación entre **v1** y **v2**. En esta implementación consideramos **(v1,v2) = (v2,v1)**, para todo **v1,v2** elementos de **V**. Un *camino* entre **v,u** elementos de **V** es el conjunto ordenado *path* **= (v1,v2,…,vn)** de vertices de **V** tales que **(vi, v{i+1})** están en **E**. En este caso, decimos que el *camino* es de longitud **n**.

En la implementación de la base de datos, queremos que cada nodo represente a una persona, o bien alguna característica en específico, como gustos en cuanto a comida, presupuesto etc. Además, modificamos la definición de aristas para permitir incluir un tipo o etiqueta, así no solo relacionamos elementos de nuestro grafo, sino que también podemos llevar un registro del tipo de relación que existen entre los elementos. Por ejemplo, una persona puede estar relacionada con otra persona o con alguna preferencia en cuanto a restaurantes, pero el tipo de relación no es la misma en ambos casos.

En particular, la estructura propuesta es la siguiente: (aunque aún está sujeta a cambios)

**(V) Nodos: [**

*// Representa una persona/usuario que esta usando el sistema de recomendaciones*

**“Persona”: {**

**“id”: { Int }, “Nombre”: { String }, “Sexo”: { Char }, “Edad”: { Int }, “Username”: { String }**

**},**

*// Representa a un tipo de restaurante, i.e. ‘comida china, hamburguesas, comida italiana’, etc.*

**“Tipos\_restaurantes”: {**

**“id”: { Int }, “Descripcion”: { String }**

**} ,**

*// Categorias para el presupuesto estimado a gastar en una salida*

**“Presupuestos”: {**

**“id”:{ Int }, “Valor estimado”: { Decimal }**

**},**

*// Categorias para el tipo de música preferida, i.e. ‘electronica’, ‘rock’, ‘regueton’, ‘baladas’, etc.*

**“Musica”:{**

**“id”: { Int }, “Descripcion”: { String }**

**},**

*// Categorias para el tipo de películas preferidas, i.e. ‘comedia’, ‘drama’, ‘accion’, ‘romanticas’, etc.*

**“Peliculas”:{**

**“id”: { Int }, “Descripcion”: { String }**

**}]**

**Tipos de aristas o relaciones :[**

**“Conocidos”** *// Establece una relacion de amistad entre dos personas*

**“Restaurante”** *// Indica que tipo de restaurante prefiere*

**“Presupuesto”** *// Indica cual es el presupesto propuesto por el usuario*

**“Musica”** *// Indica que ese tipo de música le gusta al usuario*

**“Peliculas”** *// Indica que ese tipo de películas le gusta al usuario*

**“Parecidos”** *// Establece una relación de ‘cercania’ entre dos categorías de un mismo tipo*

**]**

**(E) Aristas: [**

**“Relacion”:{**

**“id”: { Int }, “id\_origen”: { Int }, “id\_destino”: { Int }, “tipo”: { String }**

**}**

**]**

# Pseudocódigo del algoritmo de recomendaciones a utilizar

Consideremos el conjunto de nodos **V** de nuestro grafo **G = (V,E)**. Definimos **P** como el subconjunto de **V** tal que los elementos de **P** son todos los nodos de tipo *Persona* de **V.** Definimos **C = V – P** como el conjunto de todos los nodos de **V** que representan *Características* o *Categorías*.

Dado un vertice **v** de **V**, definimos la *vecindad de* ***v***, **B1(v)**, como el conjunto de vertices **u** de **V** tales que existe **(v,u)** en **E**. Asi mismo, definimos la *n-eaba vecindad* de **v**, **Bn(v),** como la unión de las *(n-1)-vecindades* de los elementos de **B1(v)** (sin tomar en cuenta a **v**, **n>1**). Definimos también **Cn(v,u)** como el número de caminos de longitud **n** que existen entre **v** y **u**.

Para seleccionar la ‘pareja optima’ del usuario **v** elemento de **V** se hace lo siguiente:

* Escogemos un numero entero **k>0** y constantes **a1, a2, … , ak**. Donde **k** determina la profundidad de la búsqueda y **a1, a2, … , ak** son los pesos o importancia que se le da a cada nivel.
* Entonces, a cada elemento **u** de **Bk(v)** le asignamos lo siguiente:
* Ordenamos los elementos los elementos **u** por **S(u)** de mayor a menor
* Mostramos al elemento **u** con **S(u)** mas alto (o los primeros tres), en caso exista.