1.1.1. Crowdsourcing

La generalización de la red redes ha dado paso a nuevas formas de trabajo colaborativo, tanto a nivel altruista como de negocio. Este tipo de actividades suelen incluirse dentro del término conocido como crowdsourcing. Debido a que se trata de un término de reciente acuñación, no existe actualmente un desarrollo bibliográfico amplio sobre el concepto, es por ello que para efectos de esta investigación se han tomado como referencia las definiciones de tres autores citados en el amplio estudio realizado por Díaz (2014) [21] que tienen similitud con la herramienta que se desea desarrollar.

La primera de ellas corresponde a la definición canónica propuesta por Jeff Howe [22], quien acuñó el término crowdsourcing como una contracción y un neologismo de la palabra inglesa crowd (masa de gente o multitud) y outsourcing (externalización), en junio de 2006 en un artículo de la revista Wired "The Rise of Crowdsourcing", traducido al español como “El Ascenso del Crowdsourcing”. En este artículo, Howe definió el crowdsourcing como “el acto, iniciado por una empresa o institución, que tiene como objetivo externalizar una tarea, normalmente realizada por un empleado, a un grupo de individuos grande e indefinido mediante una convocatoria abierta” Howe (2006).

Esta definición, quizás demasiado abstracta, puede ser aplicada de diferentes maneras según la tarea que se vaya a externalizar. Este mismo autor realizó una clasificación inicial diferenciando entre iniciativas de:

Crowdfunding (financiación colectiva)
Crowdcreation o crowdproduction (creación colectiva)
Crowdvoting (votación colectiva)
Crowdwisdom (sabiduría colectiva)

Una definición más amplia es propuesta por Daren Brabham [23], uno de los autores que más ha escrito sobre crowdsourcing, este lo define como “un modelo online de producción y de resolución distribuido de problemas que aprovecha la inteligencia colectiva de las comunidades online para propósitos específicos establecidos por un organizador, bien sea corporación, gobierno o voluntariado”. En resumen, es una

combinación del proceso creativo abierto de estructura bottom-up con los objetivos de la organización de estructura top-down Brabham (2008).

Por otra parte, el estratega y experto en medios sociales Henk van Ess [24], define el crowdsourcing de una manera diferente, haciendo hincapié en la necesidad de una devolución del conocimiento o de los resultados al mundo por razones éticas. Su definición, ampliamente utilizada en la prensa, identifica el crowdsourcing como "la canalización del deseo de los expertos para resolver un problema y luego compartir libremente la respuesta con todo el mundo", que es quizás la definición que más se adapta al objetivo que quiere lograrse con en esta investigación. (Ess, 2010).

Uno de los estudios más amplios encontrados es el de Enrique Estellés [25], quien ha publicado varios artículos en los que estudia unas cuarenta definiciones diferentes sobre el término crowdsourcing, con la intención de elaborar una definición integradora. Como conclusión al trabajo realizado con motivo de su tesis, Estellés da la siguiente definición.

"Actividad participativa online en la que un individuo, institución, organización sin ánimo de lucro o empresa propone a un grupo de individuos de conocimiento, heterogeneidad y número variable, la realización voluntaria de una tarea a través de una convocatoria abierta flexible. La realización de esta tarea, de complejidad y modularidad variable, y en la que la multitud debe participar aportando su trabajo, dinero, conocimiento y/o experiencia, siempre implica un beneficio mutuo. El usuario, o crowdworker, recibirá la satisfacción de una necesidad, sea esta económica, de reconocimiento social, de autoestima o de desarrollo de capacidades personales, mientras que el crowdsourcer obtendrá y utilizará en su beneficio la aportación del usuario, cuya forma dependerá del tipo de actividad realizada". Estellés (2013).

El autor citado anteriormente unifica en su investigación a aquellos términos relacionados al trabajo colaborativo que se pueden considerar dentro del denominado crowdsourcing y que se describen a continuación.

1. Crowdvoting: es el proceso por el que se aprovecha el juicio de la comunidad para organizar, filtrar y clasificar jerárquicamente contenido en concursos de ideas o grandes encuestas.

2. Crowdwisdom: es el proceso de tomar en cuenta la opinión colectiva de un grupo de individuos en lugar de la opinión de un único experto para responder a una pregunta o un problema. Este proceso, aunque no es nuevo en la era de la información, se ha convertido en el principal impulsor de sitios de información social, como Wikipedia o Yahoo! Answers, así como otros recursos web que se basan en la opinión humana (Laemmermann, 2012, citado por Diaz 2013).
3. Crowdcasting: se define como la iniciativa en la que un individuo, empresa u organización plantea a la multitud un problema o tarea, siendo recompensado quien lo resuelva primero o mejor lo realice (Estellés, 2012).
4. Crowdcollaboration: son aquellas iniciativas en las que se produce una comunicación entre los individuos de la multitud, mientras que la empresa iniciadora del proceso queda relativamente al margen. En estas tareas, los individuos aportarán su conocimiento para resolver problemas o plantear ideas de forma colaborativa.
    a. Crowdstorming: se define como tormentas de ideas online, donde se plantean ideas y la multitud participa con sus comentarios y/o votos.
    b. Crowdsupport: se define como aquellas iniciativas en las que los clientes resuelven dudas, problemas o incidencias gracias a la colaboración de la multitud sin necesidad de recurrir a la compañía implicada.
5. Crowdcontent: se define como aquellas tareas en las que la multitud aporta su mano de obra y su conocimiento para crear o encontrar contenido de diversa naturaleza (Doan et al., 2011, citado por Diaz 2013). Se diferencia del crowdcasting en que no es una competición, sino que cada individuo trabaja de manera individual y al final se reúne el resultado de todos.
    a. Crowdproduction: donde la multitud realiza tareas de creación de contenido.
    b. Crowdsearching: donde la multitud se encarga de realizar búsquedas de contenidos disponibles en internet con algún fin.
    c. Crowdanalyzing: donde la multitud trata de llevar a cabo búsquedas de contenidos en documentos multimedia, bien sea en internet o medio físico, como imágenes o vídeos.

6. Crowdfunding: se define como una convocatoria abierta, fundamentalmente a través de Internet, para la provisión de recursos financieros, ya sea en forma de donación o a

cambio de algún tipo de recompensa y/o los derechos de voto con el fin de apoyar las iniciativas para propósitos específicos. Es la forma más conocida y difundida de crowdsourcing (Schwienbacher et al., 2010, citado por Diaz 2013).

7. Crowdopinion: es definido como aquellas Iniciativas que buscan recoger la opinión de los usuarios a cambio de algún tipo de recompensa. Generalmente se utilizan herramientas que permiten recoger la opinión de los usuarios. La elección y entrega de la recompensa asociada depende del criterio, y a veces ingenio, del crowdsourcer.

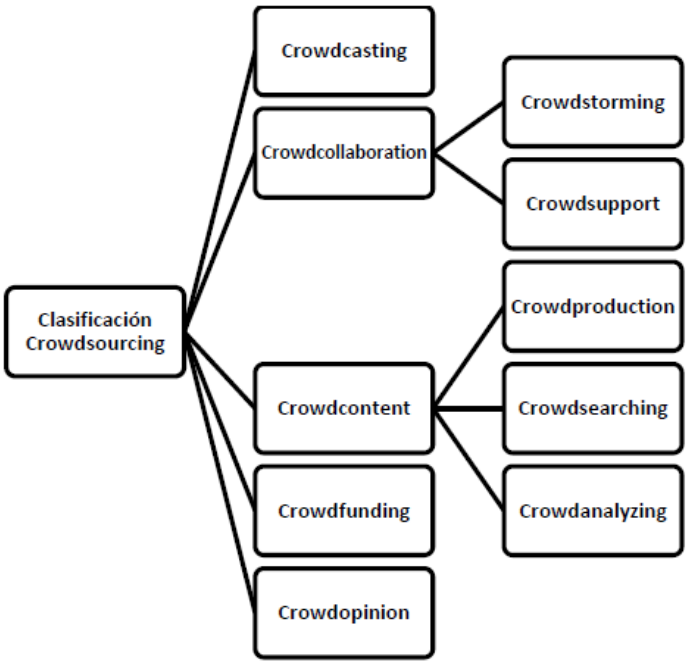


Figura X: Diagrama de Clasificación de las iniciativas crowdsourcing. Fuente: (Estellés, 2012).

1.1.2. Bases de Datos Orientadas a Grafos

1.1.2.1.Definición

Antes de presentar la definición formal de una base de datos orientada a grafos, es necesario introducir el concepto básico de un Grafo.

En Matemáticas y Ciencias de la Computación, un Grafo representa un conjunto de relaciones entre un conjunto de objetos o entidades de interés. En la forma más genérica, un grafo se representa como una tupla G = (V, E), donde V es un conjunto de nodos o vértices y E □□V□V, son un conjunto de aristas o arcos que representan relaciones por pares entre los nodos. Se dice que un grafo es dirigido, si para cualquier par de nodos u,v □□V, (u,v) □□(v,u), es decir, Dada una arista (u,v) u es su *nodo inicial* y v su *nodo final*. Un grafo es no dirigido si la dirección de las aristas no importa. En tales casos, una arista también se representa como un subconjunto de vértices de 2 elementos, en lugar de un par ordenado. [26].

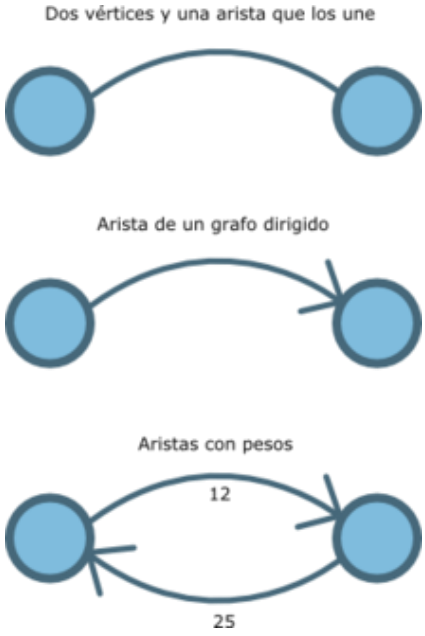


Figura X: Descripción grafica de los vértices y las aristas que los unen. Fuente: propia del autor

La alta demanda requerida al momento de almacenar información en la actualidad y la gran capacidad que se necesita en las aplicaciones informáticas cuando realizan consultas sobre ellas, ha generado que el paradigma NoSQL se posicione como el adecuado para almacenar y gestionar datos. En muchas ocasiones los datos necesitan ser analizados e interpretados para convertirlos en información útil y provechosa, tal situación ha propiciado una creciente necesidad por técnicas/herramientas computacionales que ayuden en estas tareas. Por lo tanto, si se pudieran representar a los datos y las relaciones entre ellos como un solo conjunto, podrían generarse patrones de conocimiento que describan al conjunto de datos y contribuyan a disminuir las

complejas tareas de procesamiento que se ejecutan una vez son obtenidos de las bases de datos relacionales para convertirlos en información.

De esta forma, un sistema de gestión de bases de datos de grafos (en adelante, una base de datos de grafos) es un sistema de gestión de bases de datos NoSQL, que emplea los métodos CRUD. Create, Read, Update y Delete para construir un modelo de datos de almacenamiento basado en grafos. Las bases de datos de grafos generalmente se crean para su uso con sistemas OLTP, siglas en inglés para Procesamiento de Transacciones en Línea. En consecuencia, estas bases de datos normalmente están optimizadas para el rendimiento transaccional, y diseñadas con integridad transaccional y disponibilidad operativa en mente. [27]

Las bases de datos orientadas a grafos están diseñadas para los datos cuyas relaciones son representadas en forma de grafo, es decir, los datos son elementos fuertemente interconectados con un número no determinado de relaciones entre ellos. La información a gestionar por este tipo de almacenamiento pueden ser las relaciones y conexiones sociales, el transporte público, mapas de carreteras o topologías de red, cadenas de suministros, entre otros ejemplos. El almacenamiento está optimizado para hacer recorridos a través del grafo sin utilizar necesariamente un índice para las relaciones entre nodos. Por lo cual, están optimizadas para hacer consultas sobre datos próximos partiendo de uno o más nodos, [28]

Esta definición permite almacenar los datos como nodos y a sus respectivas relaciones subyacentes con otros datos como las aristas del grafo, permitiendo así aplicar conceptos de la teoría de grafos para obtener información recorriendo la base de datos mediante consultas más descriptivas.

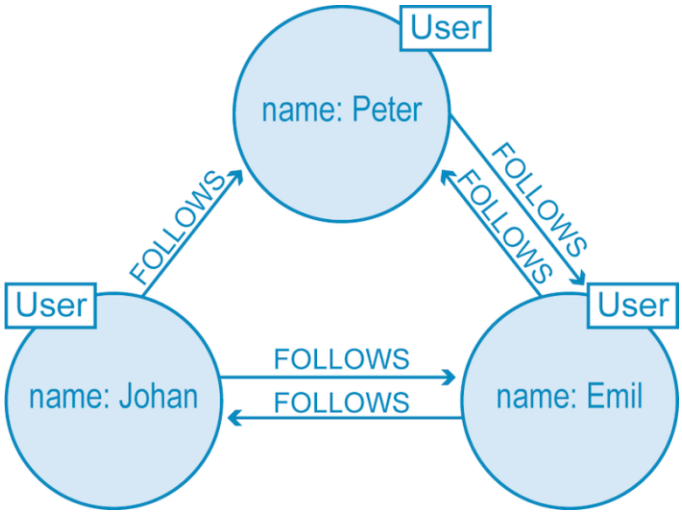


Figura X: Usuarios de Twitter representados en un modelo de grafos basado en propiedades etiquetadas.

Fuente: [27]

La recuperación efectiva de información de un grafo requiere lo que se conoce como un recorrido. Un recorrido en un grafo, también llamado recorrido transversal, implica "caminar" a lo largo de los elementos del mismo de distintas formas. El recorrido es una operación fundamental para la recuperación de datos. Una diferencia importante entre un recorrido y una consulta SQL es que los recorridos son operaciones localizadas. No hay un índice de adyacencia global, en cambio, cada vértice y arista en el grafo almacenan un "mini índice" de los objetos conectados a él. Esto significa que el tamaño del grafo no tiene impacto en el rendimiento en un recorrido y las costosas operaciones de agrupación realizadas en las sentencias SQL JOIN son innecesarias [29].

Los índices internos son necesarios para recuperar rápidamente los vértices en función de sus valores. Proporcionan un punto de partida para comenzar un recorrido. Sin índices que determinen si un elemento particular tiene una propiedad particular requeriría un escaneo lineal de todos los elementos a un costo de O(n), siendo n el número de elementos en la colección. Alternativamente, el costo de una búsqueda en un índice es mucho menor en O(log2n) [29].

1.1.2.2.Modelos

Los modelos de almacenamiento de datos basados en grafos describen las formas que existen de almacenar información en una estructura de grafo. Cabe destacar que aunque a lo largo de la historia han existido varios modelos basados de grafos, para esta investigación solo se han tomado en cuenta los modelos descritos por los autores referidos [30] quienes abarcan el estudio de los modelos implementados por las bases de datos de grafos existentes y más utilizados.

1.1.2.2.1. Modelo de grafo de propiedades etiquetadas

El modelo de un grafo clásico, una tupla G = (V, E), es adecuado para muchos problemas, como el cálculo de centralidades de vértices, sin embargo, no es lo suficientemente rico como para modelar varios problemas del mundo real. Esta es la razón por la cual las bases de datos de grafos a menudo utilizan el modelo de grafo de propiedades etiquetadas Labelled Property Graph - LPG, por sus siglas en inglés, o simplemente llamado grafo de propiedades.

En el modelo LPG, se aumenta el modelo de grafo simple G = (V, E) con etiquetas que definen diferentes subconjuntos (o clases) de vértices y aristas. Además, cada vértice y cada arista pueden tener un número arbitrario de propiedades, a menudo también llamadas atributos. Una propiedad es un par (clave, valor), donde la clave identifica una propiedad y el valor es el valor correspondiente de esta propiedad. Formalmente, un GLP se define como una tupla. Una característica que define a este modelo es que las relaciones siempre tienen un nodo inicial y un nodo final.

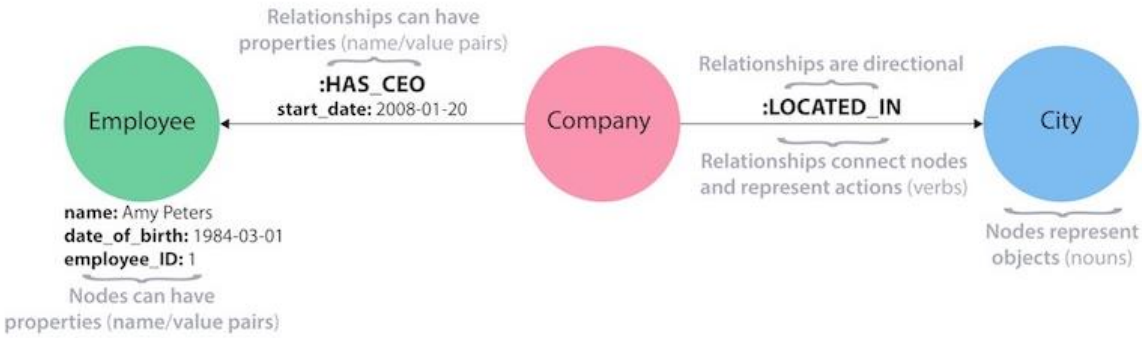


Figura X: Modelo de grafo de propiedades donde se etiquetan tanto a los vértices como a las aristas asignando propiedades a cada uno de ellos según el contexto. Fuente: [27]

1.1.2.2.2. Modelo de Hipergrafo

El modelo de Hipergrafo (dirigido o no dirigido) generaliza a un grafo de propiedades simple, en que cualquiera de sus aristas puede unir a cualquier número de vértices. Formalmente, un hipergrafo también se modela como una tupla (V, E); donde V es un conjunto de vértices y $E \subseteq (P(V) \setminus \emptyset)$ es un conjunto de hiperaristas no vacías de V.

El modelo de hipergrafo en una base de datos puede ser útil cuando los datos incluyen una gran cantidad de relaciones de muchos a muchos, en este caso están destinados a producir modelos de datos precisos y ricos en información. Si bien se considera ampliamente que los grafos de propiedades tienen el mejor equilibrio entre pragmatismo y eficiencia de modelado, los hipergrafos muestran su fuerza particular en la captura de meta-intenciones. Por ejemplo, si se necesita calificar una relación con otra.

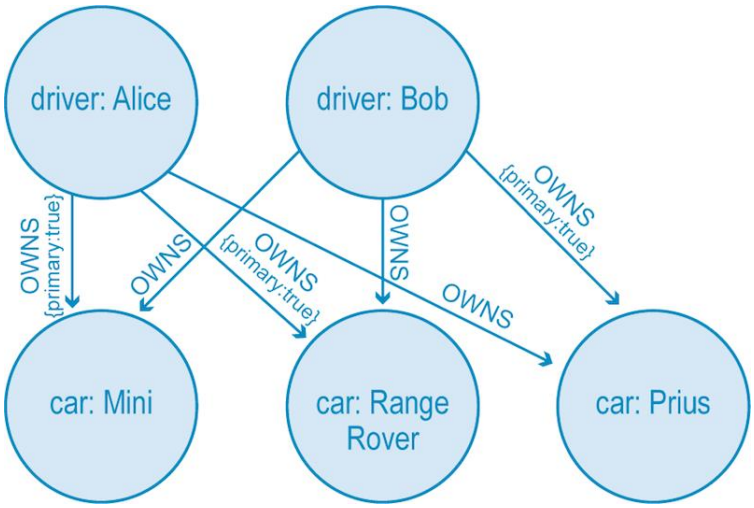


Figura X: Ejemplo de un modelo de hipergrafo donde ambos conductores son dueños de los mismos vehículos. Fuente: [27]

1.1.2.2.3. Marco de descripción de recursos (RDF)

El Marco de Descripción de Recursos (Resource Description Framework - RDF) es una colección de especificaciones para representar información. Fue introducido por el World Wide Web Consortium (W3C) en 1999 y la última versión (1.1) de la especificación RDF se publicó en 2014. Su objetivo es habilitar un formato simple que permita un intercambio de datos fácil entre diferentes formatos de datos. Es especialmente útil como descripción de datos conectados de forma irregular. La parte central del modelo RDF es una colección de triples, cada uno de los cuales consta de un sujeto, un predicado y un objeto, debido a esto a las bases de datos RDF también se le denomina "triple stores". [ref a la misma]

Los sujetos pueden ser identificadores (llamados identificadores uniformes de recursos (URI)) o nodos en blanco (que son identificadores ficticios para uso interno). Los objetos pueden ser URI, nodos en blanco o literales (que son valores simples). Con triples, uno puede conectar identificadores con identificadores o identificadores con literales. Las conexiones se nombran con otro URI (el predicado).

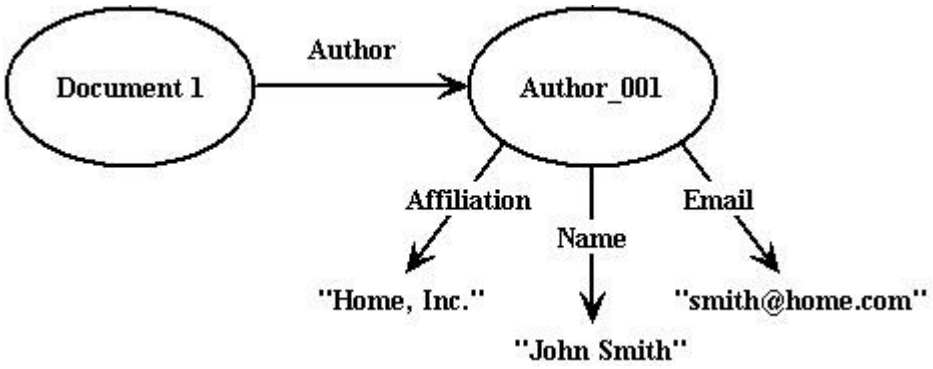


Figura X: modelo RDF para la descripción de un documento
(http://www.dlib.org/dlib/may98/miller/05miller.html)

1.1.2.3.Organización de los datos en memoria

1.1.2.3.1. División de datos en registros

Muchos motores de bases de datos (también llamados basados en registros) dividen los datos en pequeñas unidades llamadas registros. Un cierto número de registros se mantienen juntos en un bloque contiguo en la memoria o en el disco para mejorar la localidad de acceso a datos. Algunos sistemas permiten registros de tamaño variable, otros solo permiten registros de tamaño fijo. A continuación, los almacenes de valores clave suelen mantener un valor único en un solo registro, mientras que en los almacenes de documentos un documento es un registro.

Los sistemas que usan registros para el almacenamiento de grafos a menudo usan uno o más registros por vértice (estos registros a veces se denominan registros de vértice). Las aristas suelen almacenarse en el mismo registro junto con los vértices asociados (origen o destino), de lo contrario, las aristas se almacenan en registros de aristas separadas.

1.1.2.3.2. Almacenamiento de datos en estructuras de índice

Algunas bases de datos (llamadas basadas en índices) almacenan datos en estructuras de índice que se utilizan para mantener ubicaciones de diferentes partes de

los datos para un acceso más rápido a ellos. En tales casos, el índice no apunta a un determinado registro de datos, pero el índice en sí contiene los datos deseados. Algunas de las técnicas utilizadas para mantener los índices van desde los mapas de bits y árboles B + hasta tablas Hash.

1.1.2.3.3. Almacenamiento mediante aristas ligeras.

Algunos sistemas de bases de datos de grafos permiten que las aristas sin etiquetas o propiedades se almacenen como “aristas ligeras”. Estas aristas se almacenan en los registros de los vértices de origen y/o destino correspondientes. Estas aristas ligeras están representadas por el ID de su vértice de destino o por un apuntador a este vértice. Esto puede ahorrar espacio de almacenamiento y acelerar la resolución de diferentes consultas de grafos, como verificar la conectividad de dos vértices.

1.1.2.3.4. Almacenamiento mediante vinculación de registros con apuntadores directos.

En los sistemas basados en registros, los vértices y las aristas se almacenan en registros para permitir una resolución eficiente de las consultas de conectividad (es decir, verificar si dos vértices están conectados), estos registros deben apuntar a otros registros. Una opción es almacenar apuntadores directos (es decir, direcciones de memoria) en los registros conectados respectivos. Por ejemplo, un registro de arista puede almacenar apuntadores directos a registros de vértice con vértices adyacentes.

Otra opción es asignar a cada registro una ID única y usar estas ID en lugar de apuntadores directos para referirse a otros registros. Por un lado, esto requiere una estructura de indexación adicional para encontrar la ubicación física de un registro en función de su ID. Por otro lado, si la ubicación física cambia, generalmente es más fácil actualizar la estructura de indexación en lugar de cambiar todos los punteros directos asociados

Se dice que un motor de bases de datos utiliza adyacencia libre de índice cuando solo usa apuntadores directos y ningún índice para atravesar registros y así hacer recorridos en el grafo.

Aun así todavía se puede usar un índice para encontrar un vértice; el uso de adyacencia sin índice significa que solo la estructura de los datos de adyacencia es libre de índice. El uso de apuntadores directos puede acelerar los recorridos de grafos, ya que

se evitan los recorridos de índice adicionales. Sin embargo, cuando los datos de adyacencia deben actualizarse, generalmente también se necesita actualizar un gran número de apuntadores, generando una sobrecarga adicional.

1.1.2.4.Representación de grafos en sistemas de bases de datos

La idea básica al construir una base de datos de grafos es lograr representar en memoria a un conjunto de elementos como vértices y a las relaciones que existen entre ellos como aristas, a continuación se describen varios enfoques de representación estudiados en [30] que pueden servir de ayuda para comprender como se hace posible almacenar datos utilizando el esquema de un grafo.

1.1.2.4.1. Representación Nativa

Existen tres formas comunes de representar a un grafo de forma nativa; el formato de matriz de adyacencia, el formato de listas de adyacencia y el formato de lista de aristas.

En el formato de Matriz de Adyacencia, una matriz $M \in \{0,1\}^{n,n}$ que determina la conectividad de los vértices utiliza O(n2) de espacio y puede verificar la conectividad de dos vértices en tiempo O(1).

Este tipo de representación es de gran utilidad en el tratamiento algebraico de grafos y para procesos computacionales. La matriz de adyacencias, |V|x|V|, asociada a un grafo no dirigido G=(V,E), es una matriz , $A(G)=[a_{i,j}]$, donde las filas y las columnas representan elementos de V. El elemento $a_{i,j}$ será igual al número de aristas de G, cuyos extremos son los vértices correspondientes a i y j.

Si el grafo es dirigido la matriz de adyacencias $A(G)=[a_{i,j}]$ de G es de orden O(|V|x|V|), cada fila y cada columna corresponden a un elemento de V y $a_{i,j}$ es igual al número de arcos con extremo inicial correspondiente a i y extremo terminal correspondiente a j. Esta matriz no será siempre simétrica

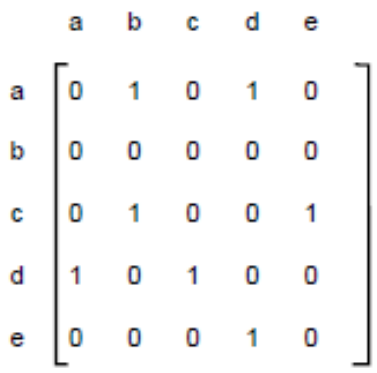


Figura X: Matriz de adyacencia para un grafo no dirigido. Fuente [31].

En la representación de Listas de adyacencia o también, listas enlazadas, cada vértice u tiene una lista de adyacencia asociada. Esta lista de adyacencia mantiene los identificadores de todos los vértices adyacentes a u. Cada lista de adyacencia a menudo se almacena como una matriz contigua de identificadores de vértice. Una lista de adyacencia requiere espacio O(n + m) y puede verificar la conectividad en O(|Au |) ⊆ O, siendo Au la lista de adyacencia para el vértice u, donde u,v ∈ Au ⇔ (u,v) ∈ E.

Las listas enlazadas ofrecen una alternativa para representar grafos y cuya ventaja principal es el manejo dinámico de la memoria, lo cual permite en forma más flexible implementar operaciones sobre el grafo tales como agregar vértices, eliminar lados, además de hacer un mejor uso de la memoria que el observado en la representación estática de los arreglos.

Por otra parte, el formato de lista de aristas es similar al formato de lista de adyacencia en la complejidad asintótica en tiempo y espacio, así como en el diseño general. La principal diferencia es que cada arco se almacena explícitamente, tanto con su vértice de origen como de destino. En las listas de adyacencia y las listas de aristas una posible causa de ineficiencia es escanear todas las aristas para encontrar los vecinos de un vértice dado. Para aliviar esto, se emplean estructuras de índice. Muchas bases de datos de grafos utilizan implementaciones variantes de listas de adyacencia que les permiten recorrer vecindarios y subgrafos de forma eficiente y directa [31].

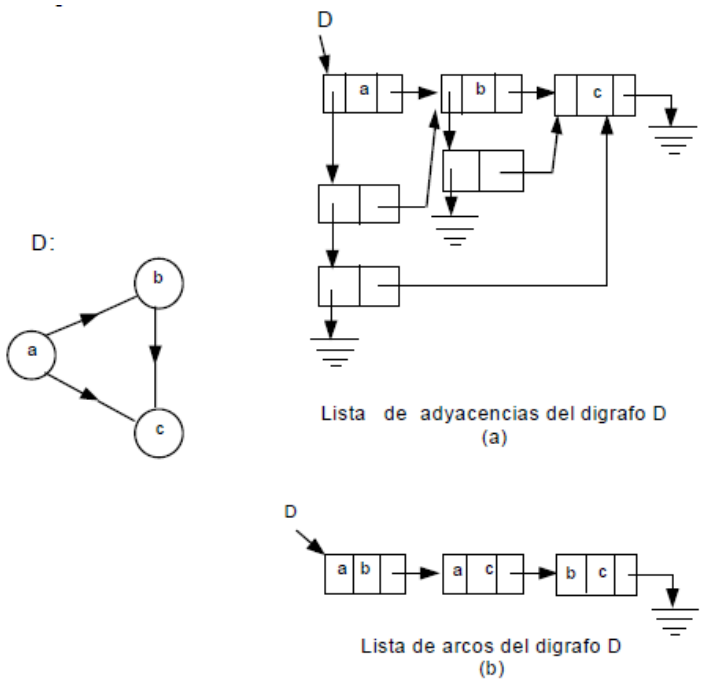


Figura X: Representación con listas de adyacencia de vértices y de aristas para un grafo D. Fuente: [31]

1.1.2.4.2. Representación en pares clave-valor.

La representación en pares de clave-valor es el paradigma NoSQL más simple [54]. En esta representación, los datos se almacenan como una colección de pares clave-valor, con el foco en búsquedas de alto rendimiento y altamente escalables basadas en claves. La forma exacta de las claves y los valores depende de un sistema específico o una aplicación. Las claves pueden ser simples (por ejemplo, un URI o un hash) o estructuradas. Los valores a menudo se codifican como conjuntos de bytes (es decir, la estructura de los valores generalmente no tiene esquema). Sin embargo, un almacén de valores clave también puede imponer un diseño de datos adicional, estructurando los valores sin esquema. Debido a la naturaleza general de los almacenes de valores clave, puede haber muchas formas de representar un grafo como una colección de pares clave-valor. La más común de ellas se ilustra en la figura X, donde las claves representan identificadores únicos para cada par de vértice/arista

Los vértices pueden almacenarse en celdas de memoria, donde un campo dedicado contiene un identificador de vértice o un hash de este identificador. Los vértices adyacentes a un vértice dado v, se almacenan como una lista de identificadores

de los vértices vecinos de v, directamente en la celda de v. Sin embargo, si una arista contiene datos enriquecidos, dicha arista (junto con los datos asociados) también se puede almacenar en una celda dedicada separada.

1.1.2.4.3. Colección de documentos

Un documento es una unidad de almacenamiento fundamental en una clase de bases de datos NoSQL llamadas almacenes de documentos. Estos documentos se almacenan en colecciones y múltiples colecciones de documentos constituyen una base de datos. Los documentos están codificados en algún formato semiestructurado estándar como XML o JSON. El almacenamiento basado en documentos amplía al almacenamiento de pares clave-valor debido a que un documento puede verse como un valor que tiene un cierto esquema flexible.

Este esquema consta de atributos, cada atributo tiene un nombre junto con uno o más valores. Dicha estructura basada en documentos con atributos permite varios tipos de valores, almacenamiento de pares clave-valor y almacenamiento de datos recursivo (los valores de los atributos pueden ser listas o diccionarios de valores clave). En este tipo de bases de datos cada vértice se almacena en un "documento de vértice".

La capacidad de los documentos para almacenar pares clave-valor se utiliza para almacenar etiquetas y propiedades de los vértices dentro de cada documento de vértice correspondiente. Sin embargo, los detalles del almacenamiento de aristas dependen del sistema: las aristas se pueden almacenar en el documento correspondiente al vértice de origen de cada arista, o en los documentos de los vértices de destino. Como los documentos no imponen ninguna restricción sobre qué pares de clave-valor se pueden almacenar, los vértices y las aristas pueden tener diferentes conjuntos de propiedades.

1.1.2.4.4. Representación mediante colecciones de tuplas

Las tuplas forman parte del paradigma NoSQL. La representación en tuplas son una generalización de un modelo RDF: los modelos RDF están restringidos a tripletas de datos (o, en algunos casos, 4 tuplas, también denominadas quads), mientras que los modelos de representación en tuplas pueden contener tuplas de un tamaño arbitrario.

Por lo tanto, el número de elementos en una tupla no es fijo y puede variar, incluso dentro de una sola base de datos. Cada tupla tiene una ID que también puede ser un apuntador directo a una ubicación de memoria directa.

Una colección de tuplas puede modelar un grafo de diferentes formas. Por ejemplo, una tupla de tamaño n puede almacenar apuntadores a otras tuplas que contienen vecindades de vértices. El mapeo exacto entre tales tuplas y datos de grafos es específico para diferentes bases de datos.

1.1.2.4.5. Representación mediante colecciones de tablas

Las tablas son la base de los sistemas de gestión de bases de datos relacionales. Para modelar un grafo como una colección de tablas, se pueden implementar vértices y aristas como filas en dos tablas separadas. Cada vértice tiene una clave primaria única que constituye su identificador. Las aristas pueden relacionarse con sus vértices de origen o destino haciendo referencia a sus claves primarias (como claves foráneas). Las etiquetas y propiedades de un modelo LPG, así como los predicados de un modelo RDF, pueden modelarse con columnas adicionales.

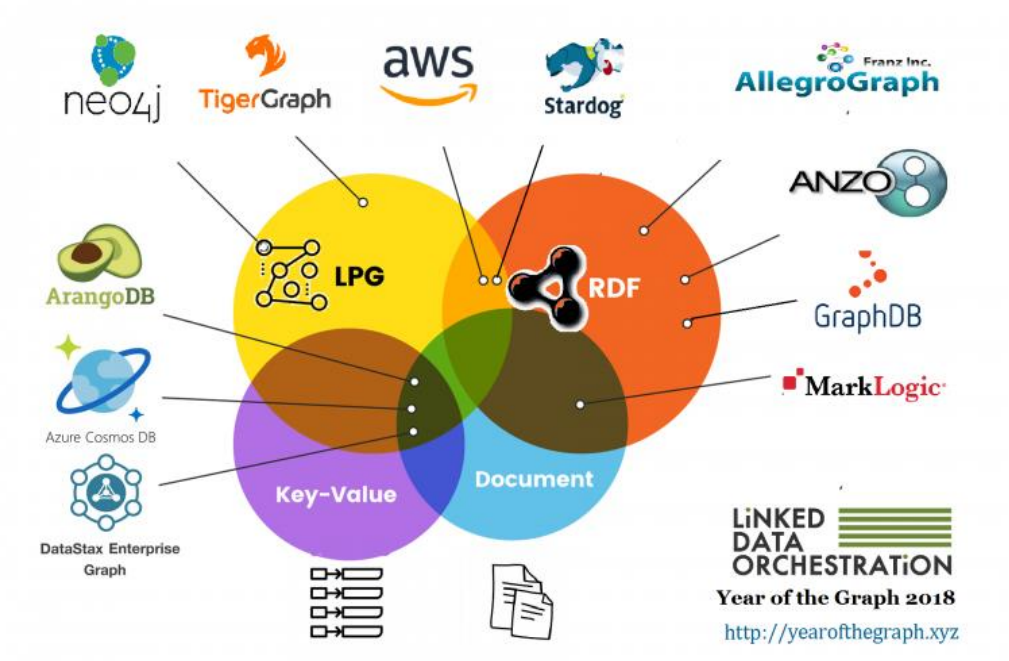


Figura X: Clasificación de los motores de bases de datos de grafos existentes según sus modelos y representación. *Fuente: The linked data Orchestation* [32]

1.1.2.5.Casos de uso

En la actualidad, las bases de datos de grafos han adquirido una gran relevancia en la administración y el manejo datos conectados. En esta sección se estudian 2 de los casos de usos más relevantes presentados por Webber et al. (2017) [33].

1.1.2.5.1. Sistemas de Recomendación

La frase "También te puede gustar" es una frase engañosamente simple que encapsula una nueva era en la gestión de las relaciones entre las personas y los datos que generan. Al ofrecer sugerencias personalizadas, las empresas maximizan el valor que brindan al proporcionar recomendaciones de productos en tiempo real altamente específicas a sus consumidores en línea. Esta capacidad de hacer ofertas atractivas requiere capturar el historial de compras de un cliente y también analizar instantáneamente sus elecciones actuales, antes de compararlas de inmediato con las recomendaciones de productos más apropiadas. Todo este análisis debe hacerse en tiempo real antes de que el cliente se mude al sitio web de un competidor.

Webber J. (2018) [34]. Plantea un escenario de una tienda en línea en donde se pueden almacenar las cestas de compras que los clientes han realizado en la tienda dentro de un grafo como el que se ilustra en la Figura X. Esta figura muestra cómo se utiliza una simple lista vinculada de cestas de compras conectadas por relaciones *NEXT* planteando un modelo que permite crear un historial de compras para el cliente Alice.

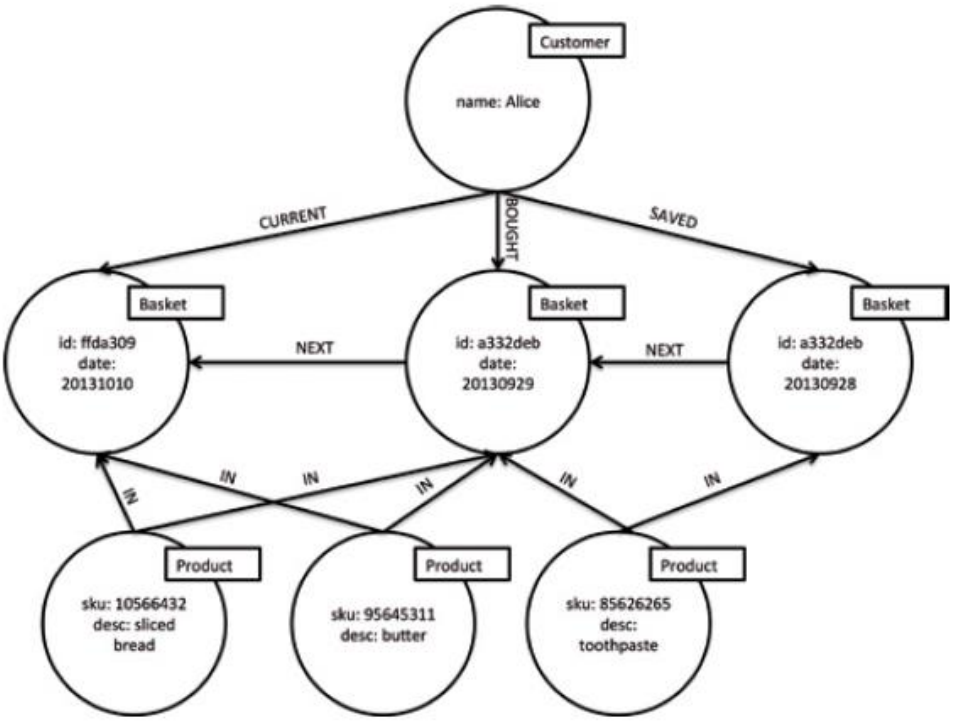


Figura X:

En el grafo puede observarse que Alice visitó tres veces la tienda y guardó una primera cesta con su posible primera compra para más tarde (la relación *SAVED* entre el cliente y los nodos de la cesta) al día siguiente decidió hacer la compra (indicada por la relación *BOUGHT* entre el cliente y el nodo de la cesta) y en ese momento estaba armando la cesta para una nueva compra, (indicada por la relación *CURRENT* que apunta a una cesta activa en la parte superior de la lista vinculada). Es importante comprender que esto no es un esquema o diagrama Entidad-Relación, sino que representa datos reales para un solo cliente.

En forma de grafo, es fácil determinar el comportamiento del cliente: La primera vez que visitó la tienda selecciono pasta de dientes, pero no se comprometió a comprarla de inmediato, visitó la tienda un día después y compró pasta de dientes, pan y mantequilla. En base a esto el sistema pudo recomendarle la pasta de dientes como un producto por el cual Alice podría estar interesada. En una compra posterior, ella decidió comprar pan y mantequilla, ambos productos que había comprado la vez anterior, lo cual puede interpretarse como un patrón repetido en su historial de compras que la

tienda puede utilizar para servirle mejor y recomendarle los productos relacionados al visitar la tienda nuevamente.

1.1.2.5.2. Sistemas de Detección de fraudes

Si bien ninguna medida de prevención de fraude es perfecta, se producen mejoras significativas cuando se mira más allá de los de datos individuales hacia las conexiones que los vinculan. Comprender las conexiones entre los datos y derivar el significado de estos enlaces no significa necesariamente reunir nuevos datos. Se puede extraer información significativa de los datos existentes reformulando el problema de una nueva forma.

Las bases de datos de grafos permiten descubrir patrones que son difíciles de detectar utilizando representaciones tradicionales como las tablas. Uno de los casos de fraude más común es el *First-Party Bank Fraud* (Fraude bancario de primera parte). En este tipo de fraude, un cliente de un banco usa su propio nombre e información para solicitar tarjetas de crédito, préstamos, sobregiros y líneas de crédito bancarias sin garantía y sin intención de devolverlos. La magnitud de estas pérdidas se debe a que los estafadores se comportan de manera muy similar a los clientes legítimos, hasta el momento en que limpian todas sus cuentas y desaparecen rápidamente.[]

Si bien los detalles exactos detrás de cada caso de este tipo de varían de una operación a otra, el patrón a continuación ilustra cómo funcionan comúnmente los anillos de fraude:

1. Un grupo de dos o más personas se organizan en una red de fraude.
2. La red de personas “anillo” por lo general comparten un subconjunto de información de contacto legítima, por ejemplo, números de teléfono y direcciones, combinándolos para crear una serie de identidades falsas.
3. Los miembros del anillo abren cuentas usando estas identidades falsas.
4. Se agregan nuevas cuentas a las cuentas originales: líneas de crédito no garantizadas, tarjetas de crédito, protección contra sobregiros, préstamos personales, etc.
5. Las cuentas se usan normalmente, con compras regulares y pagos puntuales.

6. Los bancos aumentan las líneas de crédito renovables con el tiempo, debido al comportamiento de crédito responsable observado.
7. Un día, el anillo se "revienta" coordinando su actividad, maximizando todas sus líneas de crédito y desapareciendo. A veces los estafadores van un paso más allá y reducen todos sus saldos a cero utilizando cheques falsos inmediatamente antes de desaparecer, duplicando el daño hacia el banco.
8. Se siguen los procesos de cobranza, pero los agentes nunca pueden comunicarse con el estafador.
9. La deuda incobrable se cancela al banco.

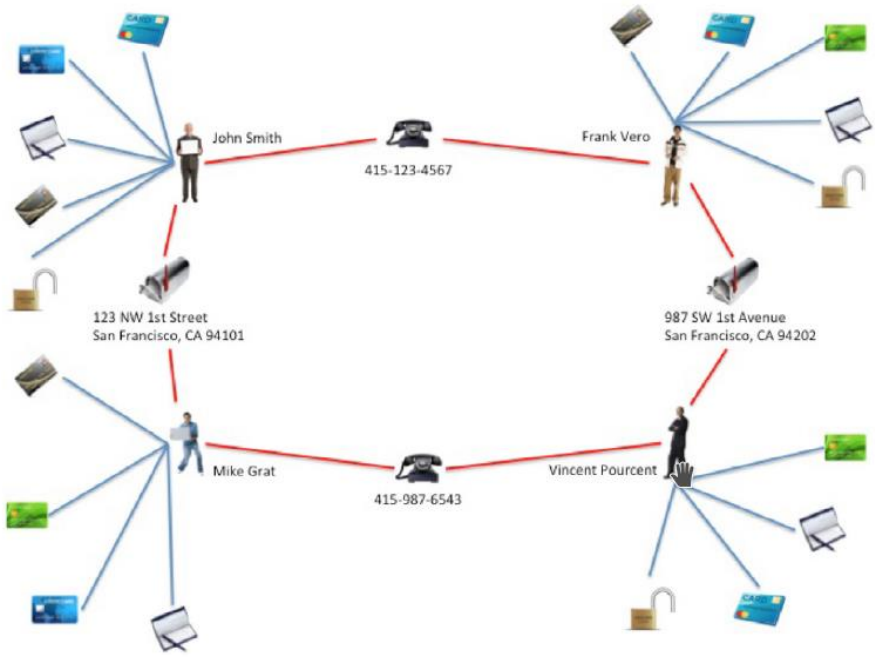


Figura X: Dos personas que comparten dos datos y crean cuatro identidades falsas.

Los números de teléfono se eliminan después de la operación de fraude, y cuando los investigadores llegan a estas direcciones, tanto Tony Bee como Paul Fabre (los estafadores, que realmente viven en esas direcciones) niegan haber conocido a *John Smith, Frank Vero, Mike Grat o Vincent Pourcent*. Las identidades falsas.

Sadowksi et al. (2018)**.** Hace un estudio sobre como las bases de datos de grafos permiten descubrir estos anillos de fraude, que con tecnologías tradicionales de bases de datos relacionales requeriría modelar el problema como un conjunto de tablas y columnas, para luego llevar a cabo una serie de uniones complejas. Dichas consultas son complejas de construir y caras de ejecutar. Escalarlos de una manera que admita el acceso en tiempo real plantea desafíos técnicos importantes, con un rendimiento que empeora exponencialmente no solo a medida que aumenta el tamaño del anillo, sino a medida que crece el conjunto de datos total [27]. El modelo de datos descrito en la Figura X representa el aspecto real de los datos si se consideran como un grafo e ilustra cómo se pueden encontrar anillos, también llamados circuitos, simplemente recorriendo el grafo.

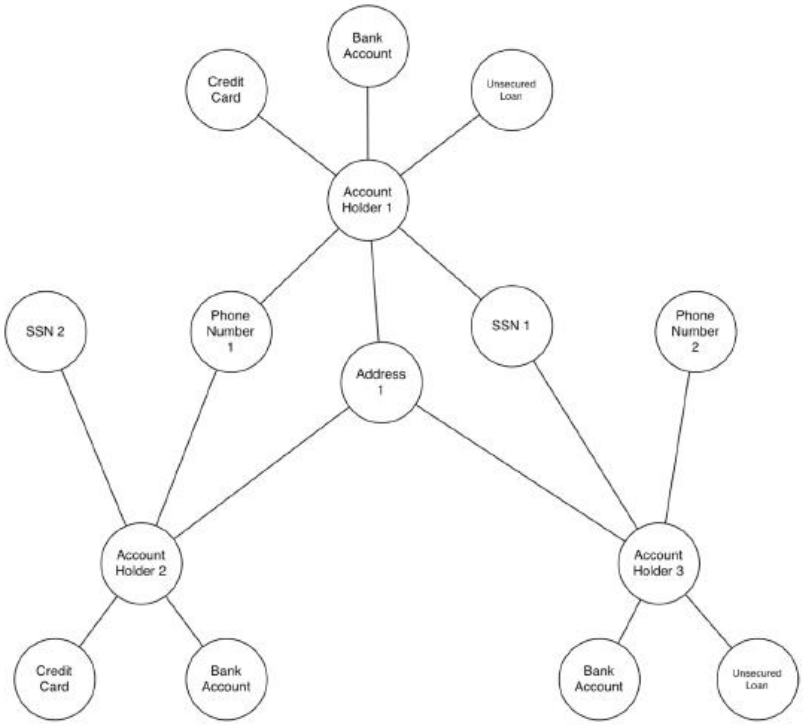


Figura X:

Se puede aumentar la infraestructura de detección de fraude existente en los bancos para admitir la detección de anillos ejecutando consultas de análisis en tiempo real y ejecutando verificaciones durante las etapas clave del ciclo de vida del cliente y la cuenta como:

1. En el momento en que se crea la cuenta,
2. Durante una investigación,
3. Tan pronto como se alcanza un umbral de saldo de crédito.
4. Cuando se “rebota” un cheque.

1.1.3. Web Scraping

Web Scraping es una técnica de minería de datos que consiste en recolectar datos contenidos en páginas web mediante técnicas automatizadas. Lo distintivo del web scraping es que en principio los datos a obtener parecen poco estructurados. Corresonde por tanto al analista de datos identificar cuál es el patrón que siguen los datos, para luego crear y ejecutar un algoritmo de extracción y procesamiento de los mismos. En la práctica lo que se suele hacer es escribir un programa que envíe consultas a un servidor web, el cual aloja los recursos HTML que conforman los sitios web y examine los datos para extraer la información necesaria (Mitchell, 2015)

El web scraping es una solución intermedia entre la recolección manual de datos (Resaltando, copiando y pegando textos de los sitios web) y el acceso automatizado a los mismos mediante alguna herramienta predeterminada como las conocidas Interfaces de Programación de Aplicaciones o (API´s, por sus siglas en ingles).

Esta técnica se aplica cuando tales protocolos no están disponibles y la cantidad de datos que se desea extraer es demasiado grande para que pueda ser realizada de forma manual y suele realizarse en dos etapas; la primera es la etapa de extracción, en la cual se realiza una consulta de datos hacia un sitio y se guardan de manera local y, después, en la segunda etapa, se realiza el análisis de estos datos para obtener la información deseada.[2]

Así mismo, se debe tener en cuenta que las páginas web son documentos escritos en lenguaje de marcado de hipertexto (HTML, por sus siglas en ingles), estos documentos están representados por un árbol estructurado llamado Modelo de Objetos

del Documento (DOM, por sus siglas en inglés) , el cual define la estructura lógica y el modo en que se accede y manipulan los elementos o etiquetas HTML utilizadas para representar la información, estas etiquetas agrupan títulos, párrafos, listas, tablas, imágenes, enlaces, contenedores, etc. como se muestra en la figura 3. [5]

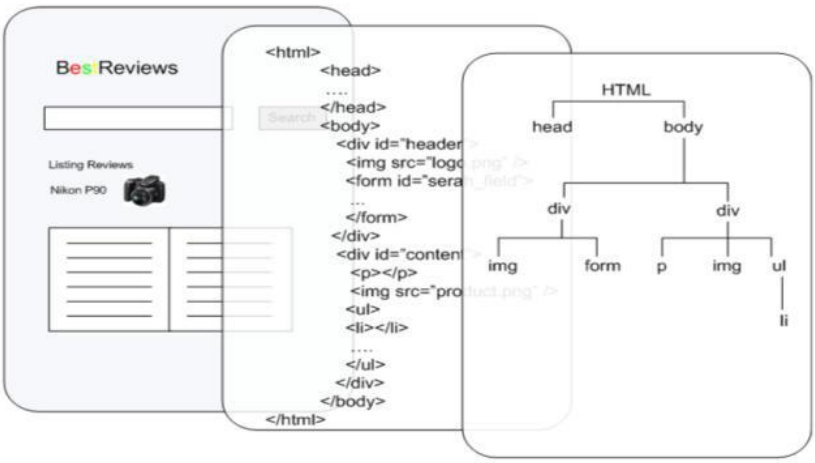
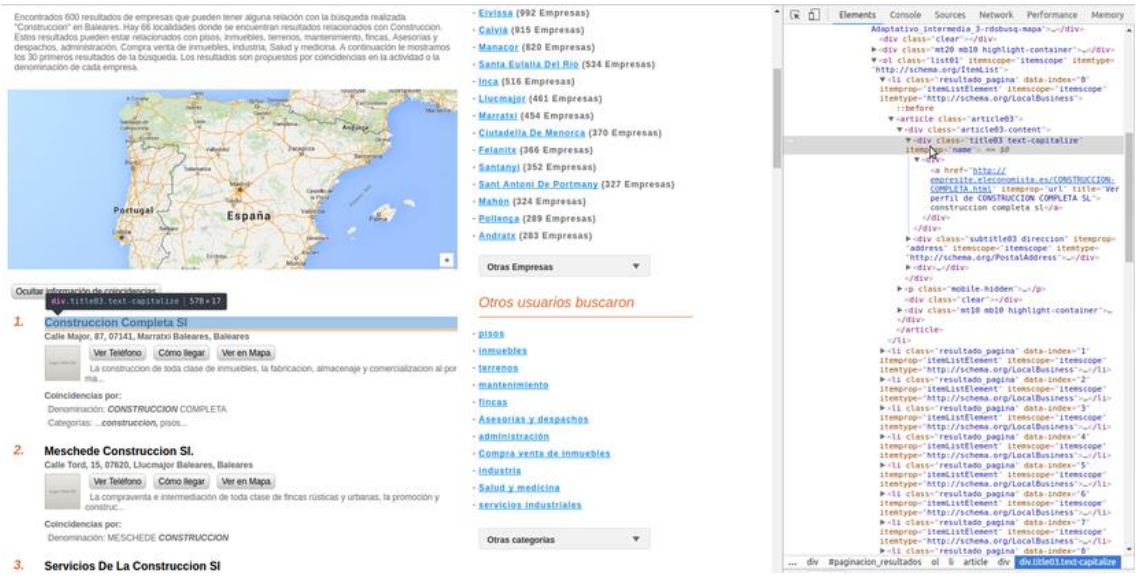


Figura 3: Tres perspectivas diferentes del documento web: el documento en la web, el código HTML y el modelo de objetos de documento [3]

1.1.3.1.Extracción de datos con Web Scraping

El primer paso de la extracción es la inspección por parte del programador de la página web. Normalmente, se realiza a través de la herramienta de inspección proporcionada por el navegador. Una vez se localiza la información deseada, se comprueba si su estructura HTML sigue algún patrón común con el cual poder extraer todos los objetos comunes de la página web [4]. . La Figura X muestra el proceso habitual de inspección del HTML de un elemento deseado con el fin de extraer patrones comunes que aplicar a través de selectores.



Una vez analizada la estructura HTML, el siguiente paso consiste en construir el rastreador definiendo el algoritmo que descargue el HTML del sitio web por medio de una solicitud HTTP GET, lo convierta a una estructura de datos conocida por el lenguaje de programación (generalmente haciendo uso de librerías especializadas) y busque dentro de dicha estructura las rutas definidas especificadas mediante selectores CSS o XPATH, hasta encontrar los elementos HTML deseados y así recuperar la información incluida en ellos, haciendo uso de estructuras cíclicas y condicionales. Por último, se almacenan los datos en el formato requerido por el programador para su posterior análisis.

[20] Toledo R. (2015) *Estudio de gestión de sistemas basados en grafos*. Universidad Carlos III - Madrid, España.

[21] Diaz A. (2014). *CROWDPROJECTS: Caracterización y Clasificación de Proyectos Colaborativos.* Universidad de Oviedo – España.

[22] Howe, J. (2006) *The rise of crowdsourcing*. Wired

[23] Brabham, D. (2013). *Crowdsourcing (MIT Press Essential Knowledge).* Ed. The MIT Press

[24] Van Ess, H. (2010). *Crowdsourcing: How to find a crowd*. ARD ZDF Akademie.

[25] Estellés, E. (2013) *Relación entre el crowdsourcing y la inteligencia colectiva: el caso de los sistemas de etiquetado social*.

[26] Srinivasa S. (2011). *Data, Storage and Index Models for Graph Databases*. International Institute of Information Technology. – India.

[27] Robinson I., Webber J., Eifrem E. (2015). *Graph Databases. New opportunities for connected data*. (2nd ed.). Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[28] Yanes E., Garcia H. (2012). *Bases de datos NoSQL*. Revista Telemática. ISSN 1729-3804. Universidad Tecnológica de La Habana. Recuperado de: http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/74/74

[29] Miller J. (2013). *Graph Database Applications and Concepts with Neo4j*. Georgia Southern University

[30] Besta M. Emanuel P. Gerstenberger. Fischer M. Podstawski M. Barthels C. Alonso G. Hoefler T. (2019). *Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries*. ETH Zurich

[31] Meza H. Ortega M. (2004). *Grafos y Algoritmos.* Universidad Simón Bolívar – Venezuela.

[32] Anadiotis G. (2018). ***What is a graph database? Do you really need one, and if yes, how do you choose?*** Recuperado de: http://linkeddataorchestration.com/the-year-of-the-graph/

[33] Webber J. Robinson I. (2017). "*The Top 5 Use Cases of Graph Databases, Unlocking New Possibilities with Connected Data".* Recuperado de: https://neo4j.com/whitepapers/top-five-use-cases-graph-databases/.

[34] Webber J. (2018). *"Powering Real-Time Recommendations with Graph Database Technology."*. Recuperado de: https://neo4j.com/whitepapers/recommendations-graph-database-business/.

[35] Sadowksi G. Rathle P.(2018)**.** *"Fraud Detection: Discovering Connections with Graph Databases"*. Recuperado de: https://neo4j.com/whitepapers/fraud-detection-graph-databases/.

[36] Román J. (2016). *CRISP-DM: La metodología para poner orden en los proyectos. Recuperado de:* https://www.sngular.com/es/data-science-crisp-dm-metodologia/

[37] Chapman, Pete (NCR); Clinton, Julian (SPSS); Kerber, Randy (NCR); Khabaza, Thomas (SPSS); Reinartz, Thomas (DaimlerChrysler); Shearer, Colin (SPSS); Wirth, Rüdiger (DaimlerChrysler). Step-by-step data mining guide. 2000.