Bienvenidos a la presentación sobre el modelo NoSQL en grafo, también conocido como modelo de grafos o modelo orientado a grafos. En esta presentación hablaremos sobre los casos donde estos modelos son adecuados, daremos información sobre su modelo de datos y sobre cómo acceder a sus datos. También presentaremos brevemente una de las extensiones del modelo en grafo: los almacenes de tripletas. Creemos que esta mención es necesaria debido a su potencial relevancia para almacenar y procesar datos web.

Modelo en grafo

- Introducción
- Ejemplo
- Modelo en grafo
- Uso de modelos en grafo

EIMT.UOC.EDU

Empezaremos introduciendo el modelo en grafo y describiendo algunas características de los grafos que consideramos útiles para entender mejor este modelo. Posteriormente mostraremos, mediante un ejemplo, cómo el modelo en grafo puede representar datos altamente relacionados de forma más simple que el modelo relacional. Después analizaremos las principales características del modelo en grafo y describiremos cómo acceder a sus datos. Finalmente hablaremos de las implicaciones que el uso de grafos impone para el diseño de la base de datos, para la consulta de los datos y de una de sus extensiones: los almacenes de tripletas (en inglés *triplestores*).

Modelo en grafo

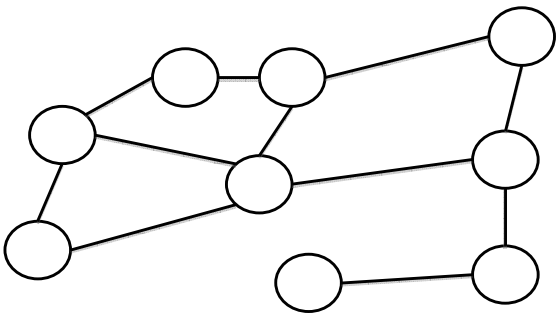
- Introducción
- Ejemplo
- Modelo en grafo
- Uso de modelos en grafo

EIMT.UOC.EDU

Vamos a empezar introduciendo los conceptos básicos de grafos sobre los que se sustenta este modelo.

Modelos en grafo y grafos

- El modelo en grafo utiliza estructuras de grafo para representar y almacenar los datos.
- ¿Grafos?



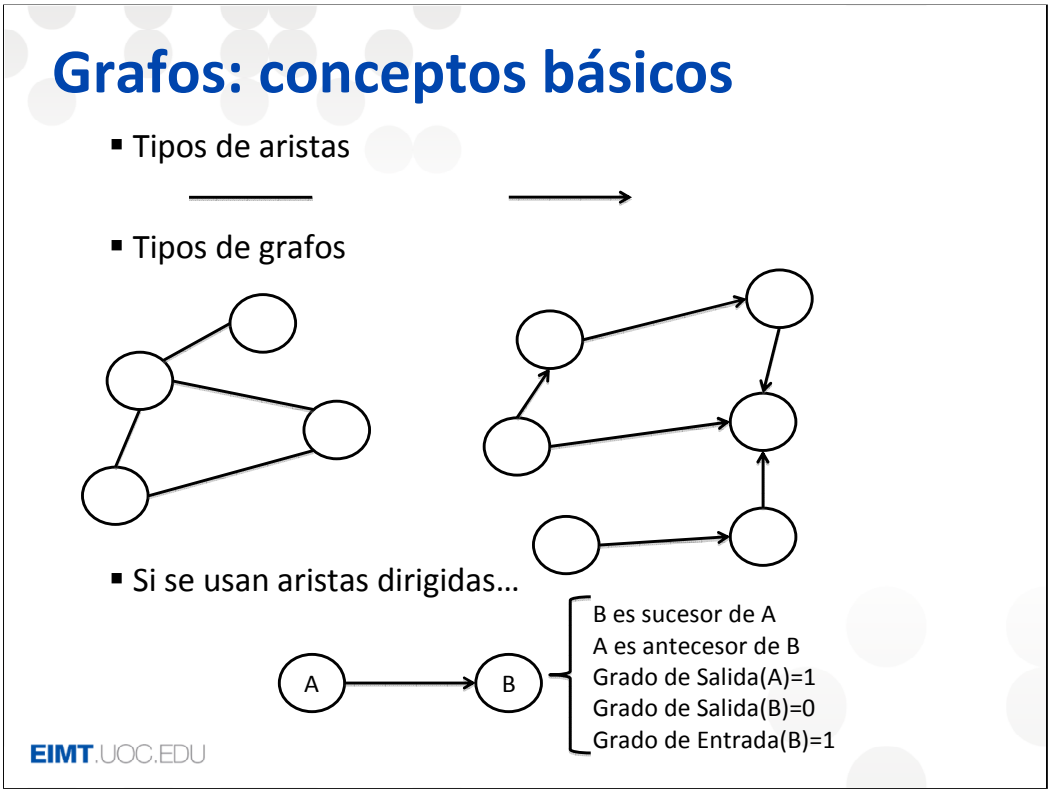
EIMT.UOC.EDU

El modelo en grafo difiere totalmente de los modelos de agregación explicados en las presentaciones previas. En este modelo los datos no se almacenan mediante agregados, sino mediante grafos.

Esto nos impone una pregunta: ¿Qué son los grafos?

Para entender el modelo en grafo de NoSQL no es necesario un gran conocimiento de los mismos. No obstante, antes de continuar, vamos a introducir algunos aspectos básicos sobre grafos. Ésta no pretende ser una explicación exhaustiva ni completa respecto teoría de grafos, sino la mínima necesaria para contextualizar los modelos en grafo en relación a la teoría de grafos.

Un grafo es una representación abstracta de un conjunto de objetos. Los objetos de los grafos se representan mediante vértices (también llamados nodos) y aristas. Los tipos, propiedades y algoritmos sobre grafos se estudian en una rama de las matemáticas denominada matemática discreta.



El tipo de aristas en un grafo pueden ser básicamente dos: dirigidas y no dirigidas. Las aristas no dirigidas se interpretan como aristas bidireccionales y se representan mediante una línea sin ninguna punta de flecha en los extremos. En contraposición, las aristas dirigidas son unidireccionales y se representan mediante una flecha que indica la dirección de la relación. Las aristas dirigidas también se conocen como arcos.

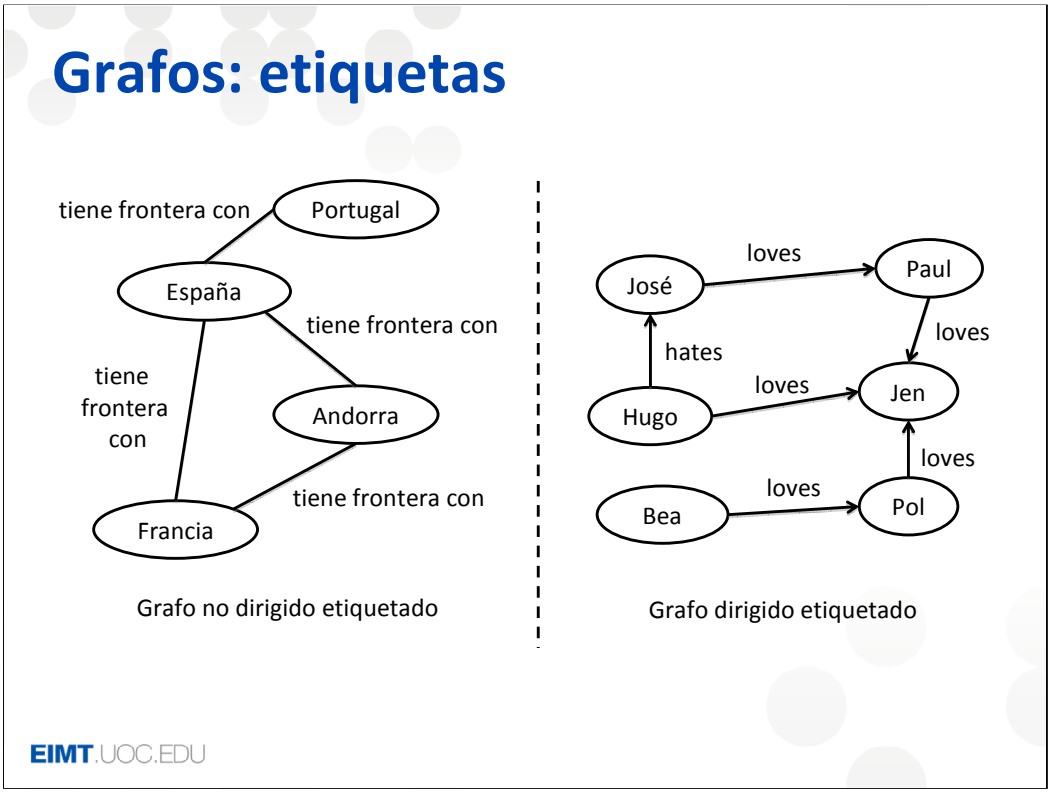
El tipo de aristas que contiene un grafo nos permite clasificar el grafo en distintos tipos. Dos de los tipos más conocidos son los grafos no dirigidos, cuyas aristas son no dirigidas, y los grafos dirigidos (o digrafos) que sólo contienen aristas dirigidas.

Cuando un grafo usa arcos, se utiliza la siguiente terminología para describir las propiedades de los distintos nodos:

• Se dice que un nodo B es sucesor de otro nodo A, cuando hay un arco que relaciona A con B, siendo A el origen y B el destino. De forma simétrica, se define la propiedad de antecesor, que indica que un nodo A es origen de una relación que lleva a B. En el ejemplo inferior de la transparencia podemos decir que el nodo B es sucesor del nodo A y que el nodo A es antecesor del nodo B.

• Por otro lado, se define el grado de salida de un nodo como el número de arcos que parten de dicho nodo y el grado de entrada como el número de arcos que llegan al nodo. En el ejemplo, el grado de salida de los nodos A y B son 1 y 0, respectivamente, y el grado de entrada de los nodos A y B son 0 y 1.

El objetivo de una base de datos es representar un conjunto de datos con una semántica concreta. Estas estructuras parecen interesantes, pero por ahora son difíciles de interpretar y entender a simple vista. Es decir, a partir de los dos grafos de la figura, es difícil saber qué información representan. Para resolver ese problema se pueden utilizar etiquetas.



En un grafo los vértices y las aristas pueden ser etiquetados o no etiquetados. Para el caso que nos ocupa utilizaremos grafos etiquetados (grafos con vértices y aristas etiquetados), ya que las etiquetas nos facilitan la identificación y diferenciación de los elementos del grafo y nos permiten entender (o interpretar) mejor lo que representan.

A continuación podemos ver un par de ejemplos de grafos etiquetados. Hemos intentado huir de los típicos ejemplos de grafos de redes sociales para mostrar que este tipo de estructuras son muy versátiles y adaptables a múltiples dominios de aplicación.

Los grafos etiquetados que vemos aquí son los mismos que hemos estado viendo hasta ahora. En este caso, gracias a las etiquetas, es fácil ver lo que representan.

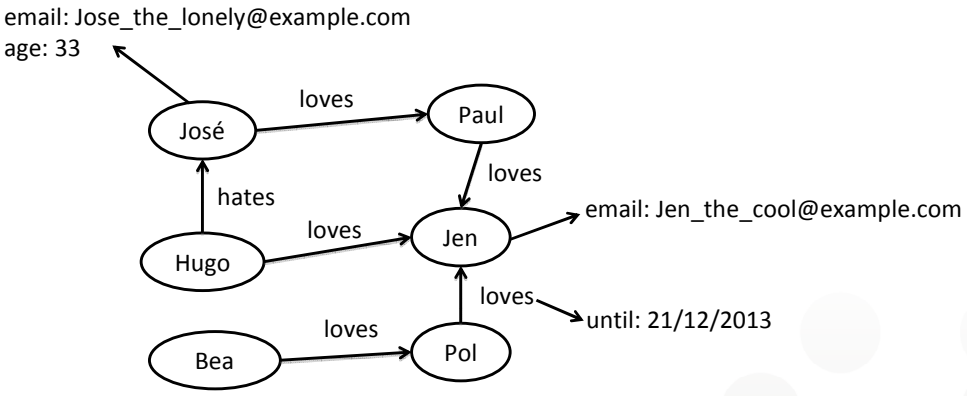
El grafo de la izquierda representa la situación geopolítica de distintos países, donde los nodos representan países y las aristas representan las fronteras que existen entre países. Notad que el hecho que haya una frontera entre dos países es una relación bidireccional por lo tanto, en este caso, se trata de un grafo no dirigido. Este grafo podría utilizarse para responder múltiples preguntas, como por ejemplo, saber el número mínimo de fronteras que debemos atravesar si queremos viajar de un país a otro.

El grafo de la derecha representa un conjunto de personas y la relaciones de amor/odio entre ellas. En este caso se ha usado un grafo dirigido porque las relaciones de amor/odio no son bidireccionales. Sino George Clooney, por ejemplo, amaría a millones de personas. Un grafo como éste tiene multitud de aplicaciones, como por ejemplo, detectar personas populares (en este caso es Jen, que tiene un grado de entrada de 3) o personas poco populares (por ejemplo José, que no es querido por nadie y es odiado por una persona).

En este punto, nos podemos preguntar, “Sólo se almacena una etiqueta por nodo y arista… ¿Qué pasa si queremos almacenar más información sobre nodos o aristas?” Bien, habría diversas maneras de hacerlo: añadiendo nuevos nodos y aristas en el grafo, añadiendo propiedades, etc. Nosotros nos centraremos en el uso de propiedades ya que es la utilización más extendida en NoSQL.

Grafos: propiedades

- Pueden asignarse tanto a nodos como a aristas.
- Están formadas por un par <clave, valor>.



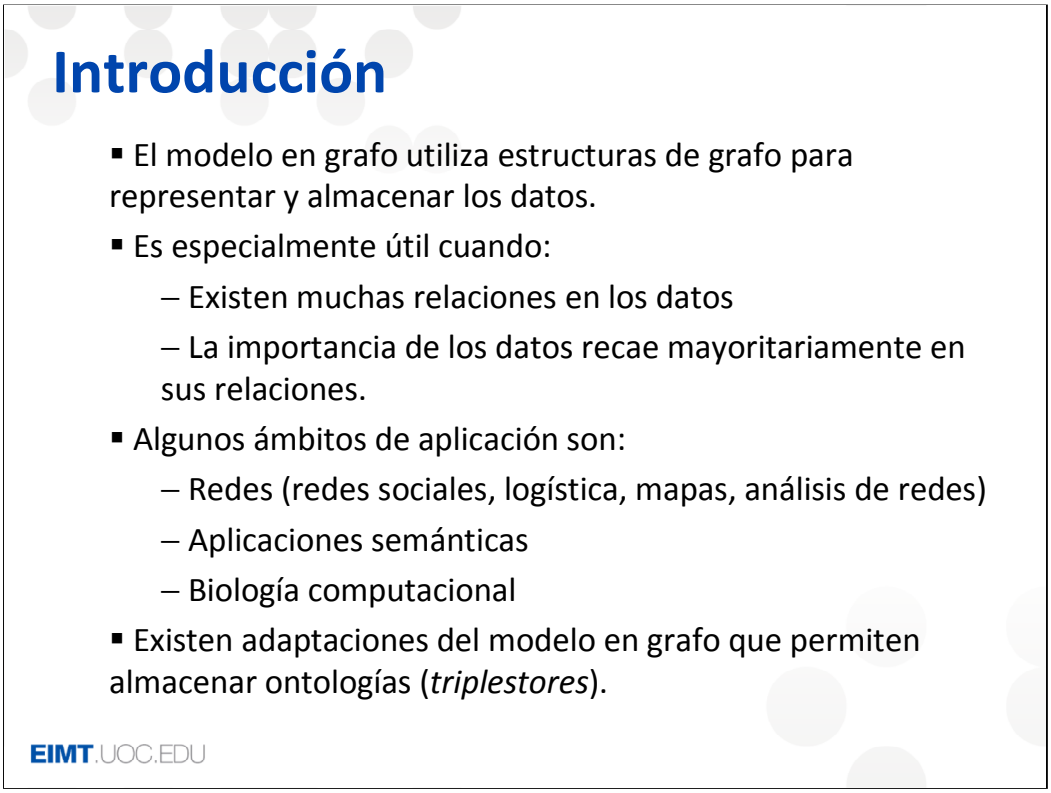
EIMT.UOC.EDU

Los grafos de propiedades son grafos dirigidos compuestos por tres elementos: nodos, arcos y propiedades.

Las propiedades son pares *<clave, valor>* y pueden asignarse tanto a nodos como a aristas.

En el grafo de ejemplo, podemos ver que se han añadido propiedades a los nodos de Jen y de José. En el caso de Jen para indicar su *email* y en el caso de José para indicar su *email* y su edad. Por otro lado, también podemos ver que se ha creado una propiedad *until* sobre la arista *loves*, para indicar que a Pol le ha gustado Jen hasta el 21 de diciembre del 2013.

Este tipo de grafo, el grafo de propiedades etiquetado, es el usado mayoritariamente por las bases de datos en grafo NoSQL. Aunque pueda haber sistemas gestores de bases de datos que usen otros tipos de grafo.



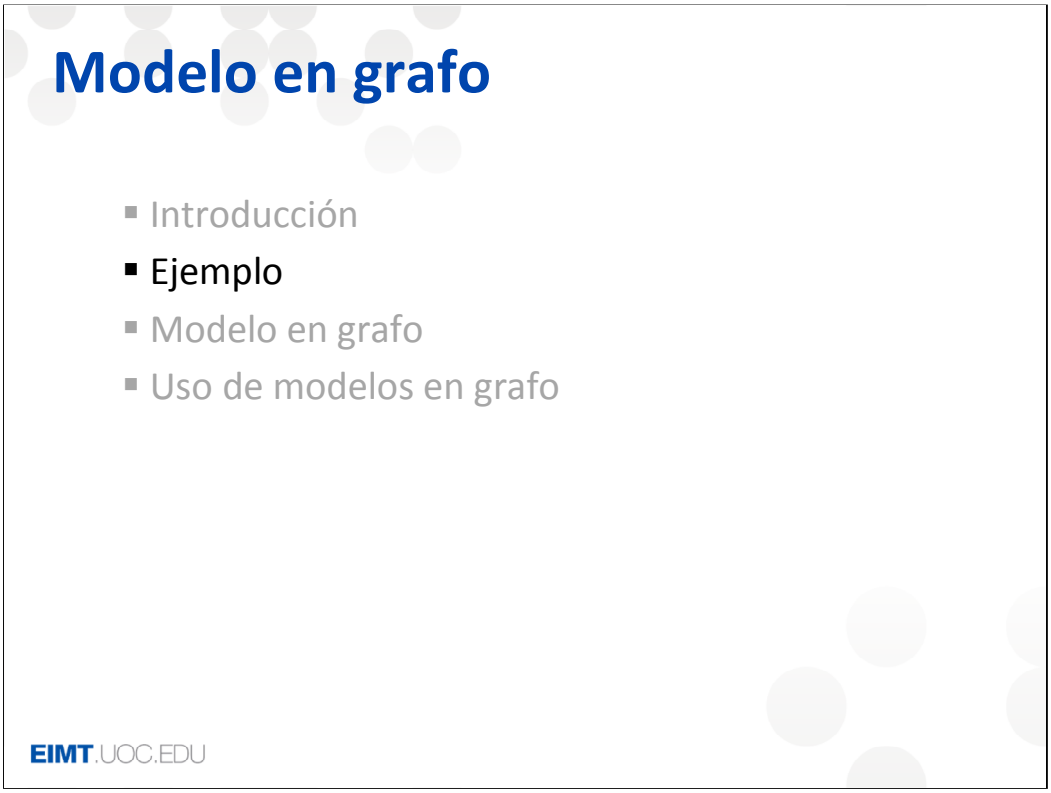
Bien, una vez claro qué es un grafo, qué puede representar y cómo, vamos a continuar con la introducción.

A partir de lo que hemos visto, es fácil intuir que el modelo en grafo es útil cuando los datos a almacenar tienen multitud de relaciones y cuando la importancia recae más en las relaciones entre los datos que en los datos en sí. En consecuencia, este tipo de bases de datos tiende a almacenar pocos datos de los objetos del mundo real pero muchos datos sobre sus interrelaciones, a diferencia de lo que acostumbra a suceder en bases de datos relacionales, donde hay mucha información de los objetos (dicha información sobre los objetos se representa en diferentes relaciones) y pocas interrelaciones entre objetos (representadas por claves foráneas).

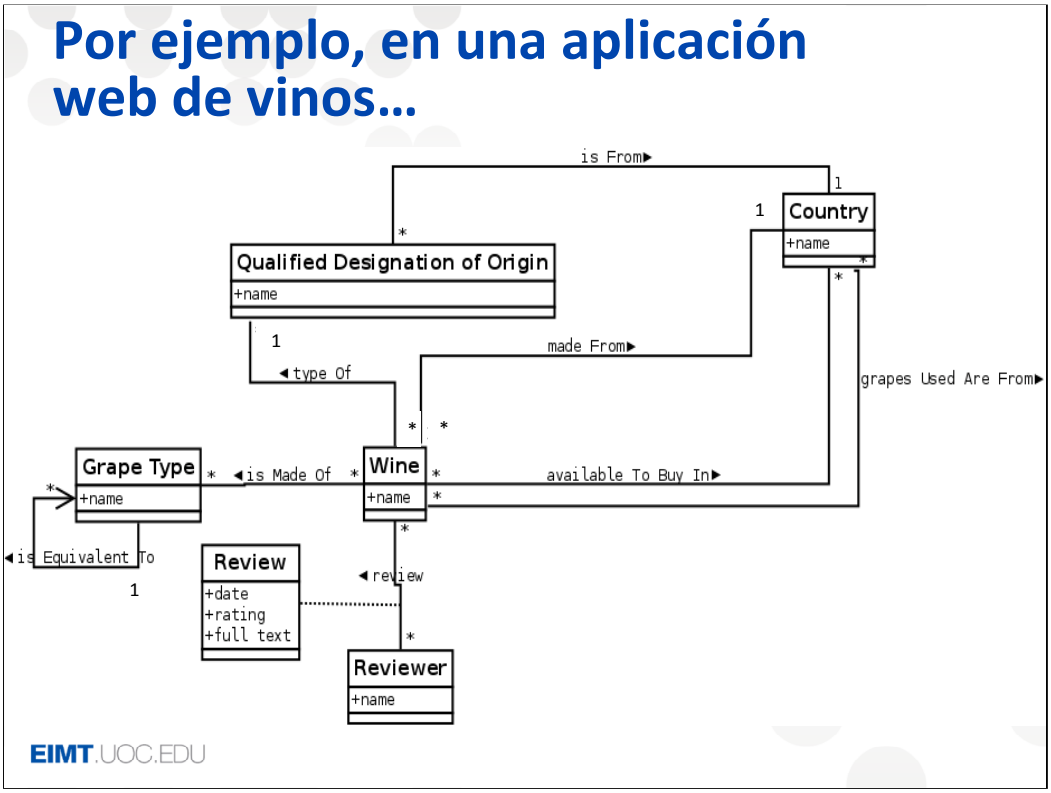
Las bases de datos en grafo se utilizan en multitud de ámbitos y para diferentes tipos de problemas. Algunos de los más comunes son:

- Dominios donde la información puede verse como una red de elementos altamente interconectados. Algunos ejemplos son las redes sociales, como Facebook o Twitter, aplicaciones de logística y enrutamiento (donde se puede representar la información geográfica –los mapas– mediante grafos) y análisis de redes.
- Las bases de datos en grafo también son adecuadas para almacenar los datos utilizados en aplicaciones semánticas, como pueden ser buscadores, asistentes virtuales o recomendadores.
- Otros ámbitos, como por ejemplo la biología computacional.

Hoy en día el modelo en grafo tiene diversas extensiones que permiten tratar algún tipo de datos de forma más eficiente. Una de sus extensiones más relevantes son los *triplestores*, que permiten almacenar ontologías en formato RDF. No entraremos en detalle en el tema. Simplemente comentar que las ontologías son el mecanismo que se utiliza en la Web semántica para dar significado a los datos de la Web, facilitando crear servicios web que se comporten de forma más inteligente. Por lo tanto, si en algún momento es necesario procesar datos relativos a la Web semántica, será interesante estudiar esta opción.



Hasta aquí hemos visto una ligera introducción a los modelos en grafo y a la teoría de grafos. A continuación vamos a ver, con un ejemplo, cómo un modelo orientado a grafos puede representar de forma más natural datos altamente relacionados.

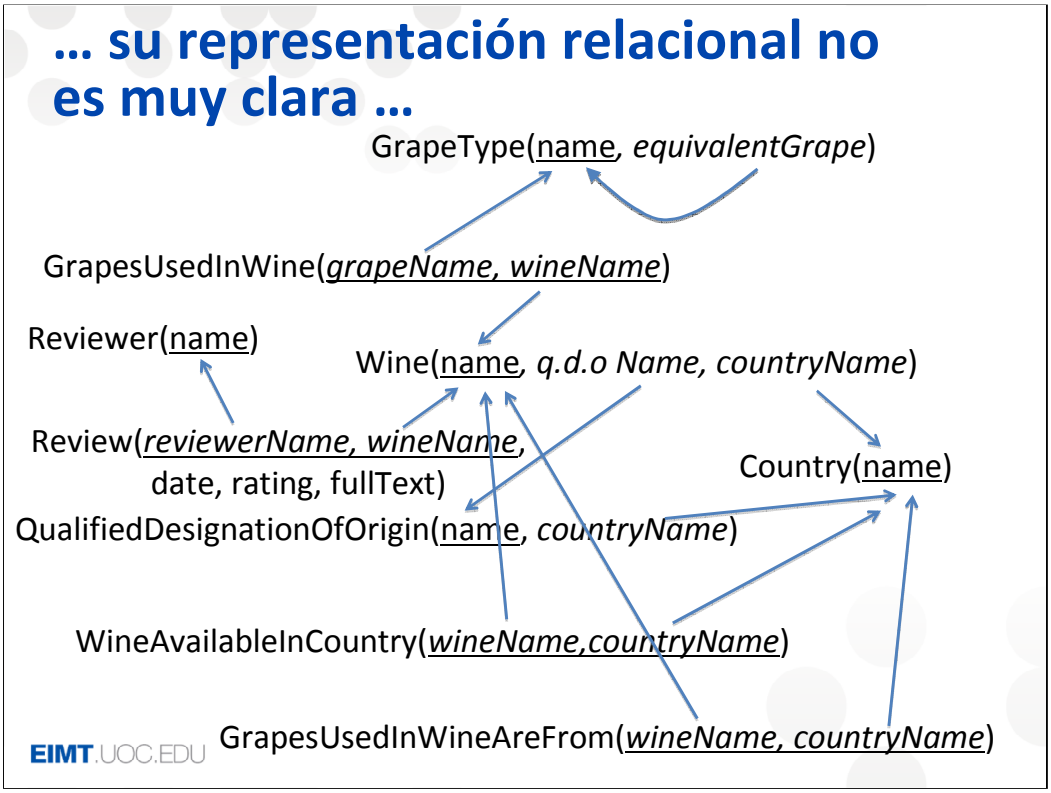


Supongamos que queremos crear una base de datos para almacenar la información de una página web de vinos. Un posible modelo conceptual simplificado del dominio de interés podría ser el que se muestra en este diagrama de clases de UML.

El elemento principal es el concepto vino, no entendido como una botella concreta sino como los tipos de vino que se comercializan: el Cavernet Souvignon Reserva de Mon Frepant o el Marqués de Pisto Crianza de Rioja. Cada vino usa un conjunto de tipos de uva que los caracteriza, como por ejemplo, Riesling, Moscatel, Pinot, Macabeo, etc. Hay tipos de uvas que son equivalentes entre sí y pueden utilizarse indistintamente para conseguir resultados similares.

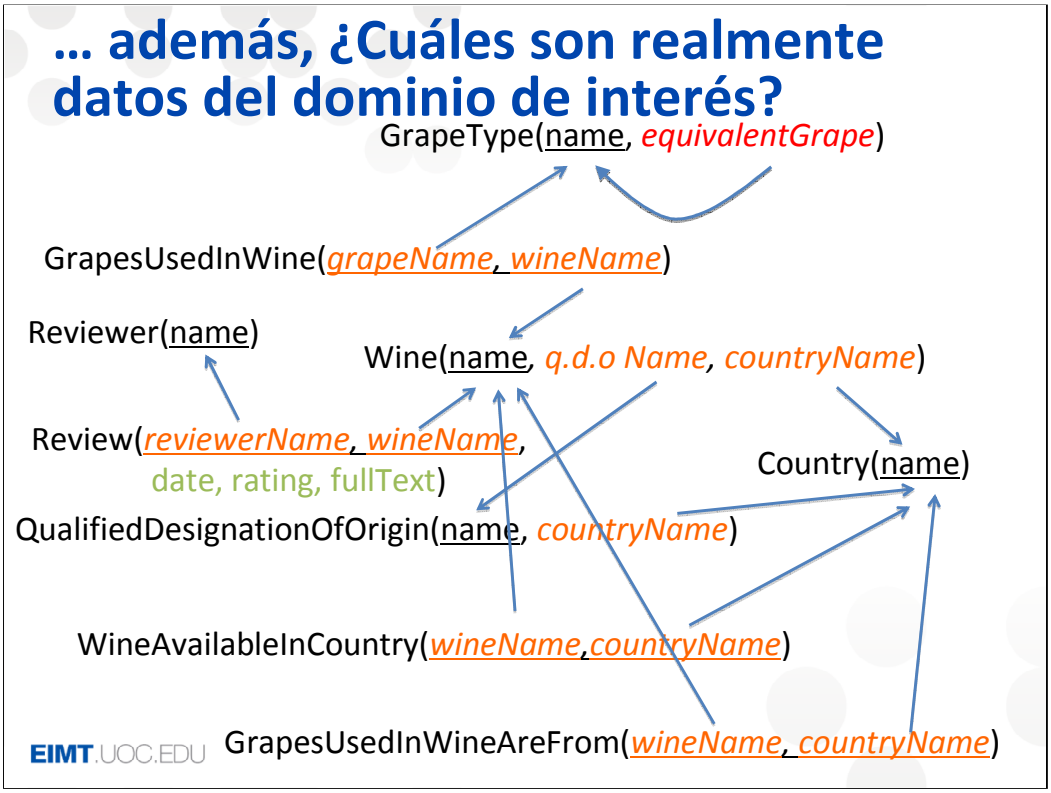
El sitio web sólo trabaja con vinos que tienen denominación de origen. Las denominaciones de origen responden a vinos realizados en zonas características, con amplia cultura vinícola y de reconocido prestigio. Éstas son locales y pertenecen a un país. Otros datos relacionados con información geográfica que conviene tener en cuenta es en qué país se hicieron los vinos (esta relación podría derivarse automáticamente a partir de la denominación de origen), en qué países se pueden adquirir y de qué país (o países) son las uvas utilizadas en su elaboración.

La web también ofrece críticas realizadas por enólogos reconocidos. Las críticas contienen la fecha de la crítica, la puntuación otorgada y el texto de la crítica.



A continuación podemos ver el modelo lógico relacional asociado al modelo conceptual mostrado. Las claves primarias están subrayadas y las claves foráneas aparecen en letra cursiva.

Como se puede ver, un modelo conceptual tan simple como el propuesto tiene una representación relacional no muy natural y que puede ser difícil de entender debido al alto grado de interrelaciones que contiene.



Sí estudiamos a fondo el modelo lógico resultante nos encontramos con un hecho interesante. Hay pocos datos que representen información sobre el dominio (atributos en verde). Simplemente los atributos fecha, puntuación y texto de las críticas y quizá también los nombres (por ejemplo, el nombre del vino, el nombre del país etc.). El resto de atributos básicamente se utilizan para identificar (las claves primarias) y relacionar las filas de la base de datos (las claves foráneas).

En particular, de los 20 atributos de nuestro modelo de bases de datos relacional, 12 de ellos son claves foráneas.

Eso significa que más de la mitad de los datos de la base de datos se utilizan para representar relaciones entre los datos. Por lo tanto, se utilizan más construcciones para representar relaciones que para representar datos.

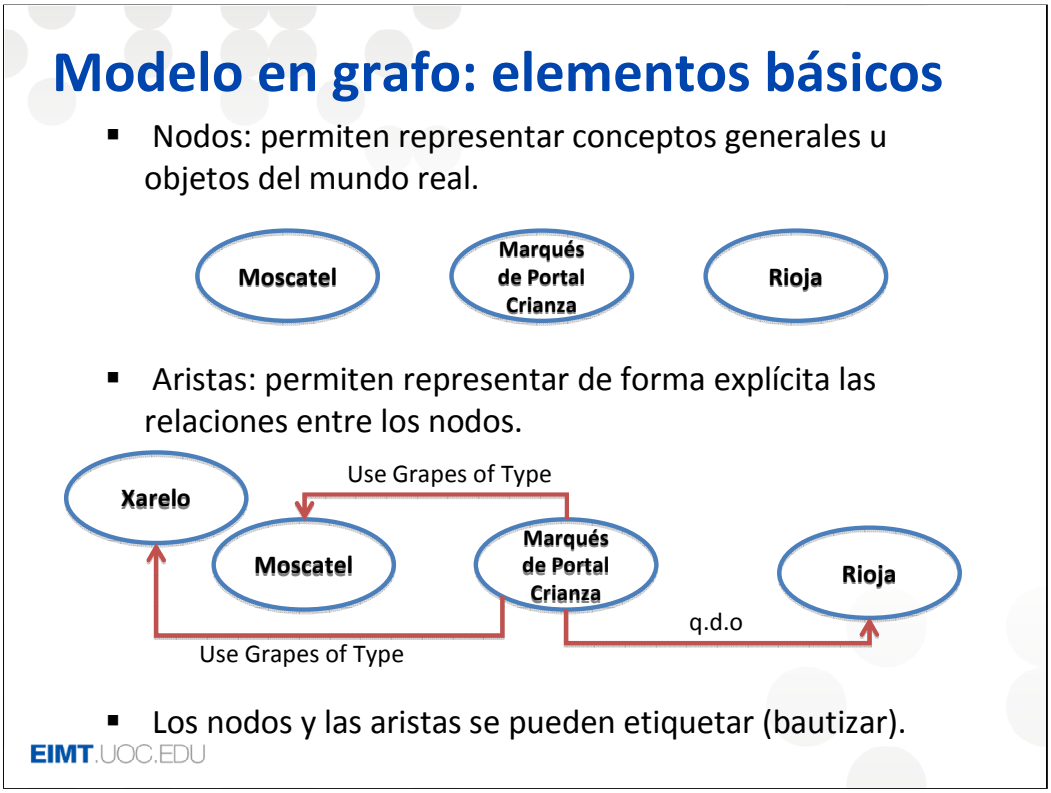
En casos como éste es necesario plantearse si el modelo relacional es el más adecuado para almacenar la base de datos. Aquí podría ser más útil utilizar un modelo de agregación en el caso de que el acceso a los datos se hiciera siempre de la misma forma. En contrapartida, el modelo en grafo sería más adecuado en el caso de que se deba garantizar una alta independencia entre las consultas y el almacenamiento de los datos. Es decir, que se quiera total libertad a la hora de realizar consultas sobre la base de datos o que las funcionalidades del sistema a crear aún no estén completamente definidas.

Modelo en grafo

- Introducción
- Ejemplo
- Modelo en grafo
- Uso de modelos en grafo

EIMT.UOC.EDU

Una vez vista la motivación de los modelos en grafo, vamos a ver qué elementos los componen y sus características principales.



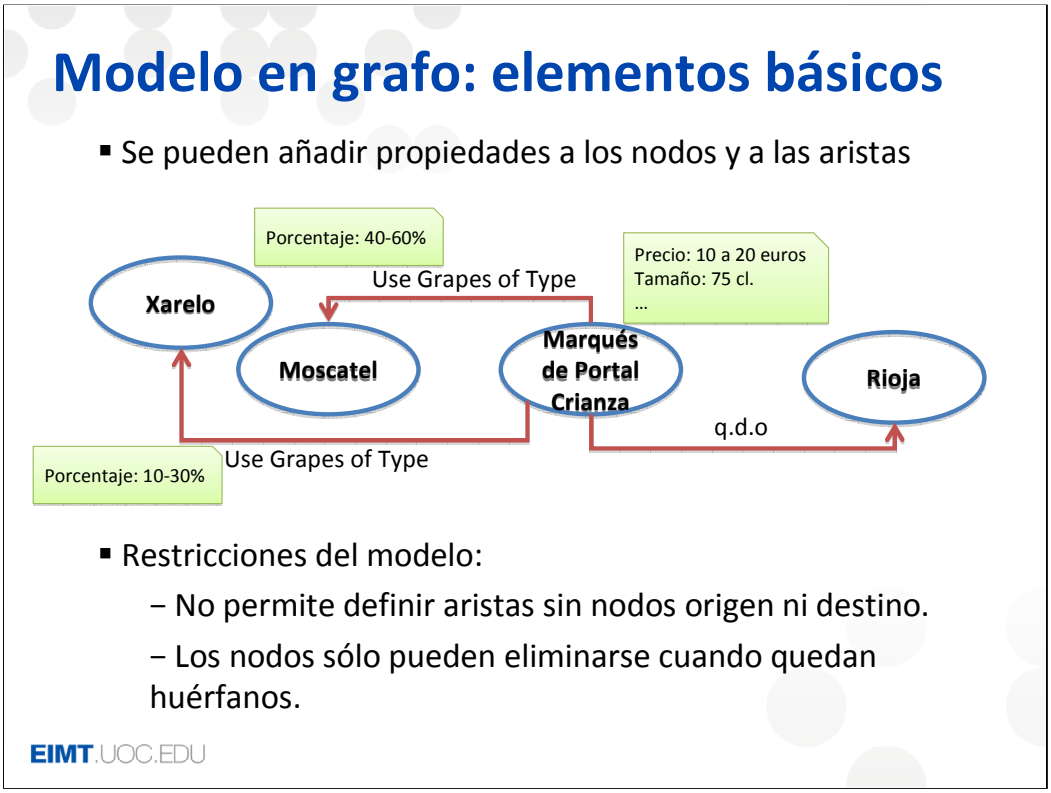
Como en el resto de modelos de datos NoSQL, no hay un modelo en grafo estándar. En nuestro caso vamos a suponer que el modelo en grafo responde a un grafo de propiedades etiquetado. Bajo esta asunción, los modelos en grafo están compuestos de nodos, aristas, etiquetas y propiedades.

Los nodos o vértices son elementos que permiten representar conceptos generales u objetos del mundo real. Vendrían a ser el equivalente a las relaciones en el modelo relacional. No obstante, hay una diferencia importante: en el modelo relacional las relaciones permiten representar conceptos (vino) y sus filas permiten representar instancias de los mismos, es decir objetos del mundo real (Marqués de Portal Crianza). Por lo tanto, los conceptos y los objetos del mundo real se representan mediante construcciones distintas. No obstante, en el modelo en grafo, los conceptos y los objetos del mundo real se representan de la misma forma: mediante nodos. No hay una diferenciación semántica de los mismos a nivel de modelo. A pesar de ello, algunos sistemas gestores de bases de datos NoSQL en grafo, como por ejemplo Neo4J, añaden construcciones extra para permitir identificar los conceptos a los que pertenecen los objetos del dominio. Ello permite mayor riqueza semántica al consultar los datos de la base de datos.

En el modelo en grafo los nodos y las aristas se pueden etiquetar, eso quiere decir que se puede asignar una cadena de texto a los mismos para facilitar su lectura y proveer mayor semántica. Ejemplos de nodos serían el tipo de uva Moscatel, el vino Marqués de Portal Crianza y la denominación de origen Rioja.

Las aristas o arcos son relaciones dirigidas que nos permiten relacionar nodos. Las aristas representan relaciones entre objetos del mundo real y serían el equivalente a las claves foráneas en el modelo relacional o a las asociaciones en los diagramas de clases de UML.

Es común que distintas relaciones compartan la misma etiqueta, ya que la etiqueta en las aristas se utiliza para identificar el tipo de relación al que pertenecen. Por ese motivo existen dos relaciones con la etiqueta “Use Grapes of Type”, entre el nodo “Marqués de Portal Crianza” y los nodos “Xarelo” y “Moscatel”. Que estas dos aristas tengan la misma etiqueta significa que ambas tienen el mismo significado: el tipo de uva utilizado para crear el vino “Marqués de Portal”. Otra relación del modelo es la que permite indicar que dicho vino tiene denominación de origen Rioja.



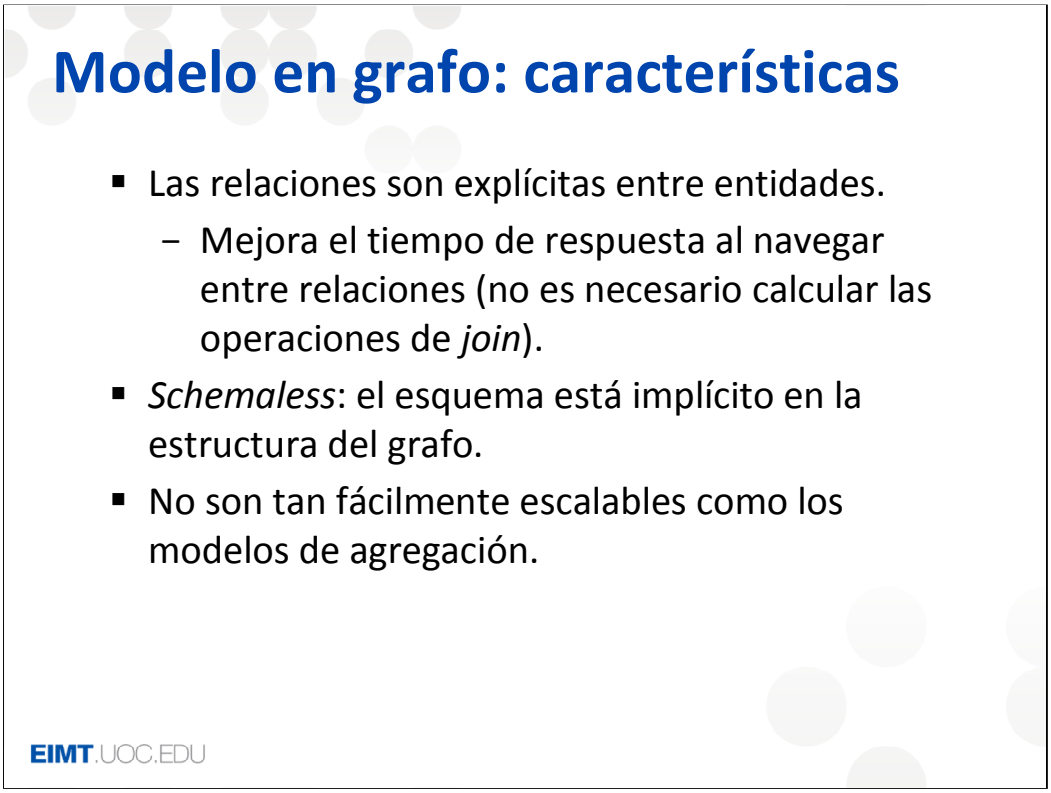
Tal y cómo hemos comentado es posible definir propiedades, tanto a nivel de nodo como a nivel de arista.

Las propiedades son parejas *<clave, valor>* que se asignan a un elemento del modelo. La clave es una cadena de caracteres, mientras que el valor puede responder a un conjunto de tipos de datos predefinidos.

Las propiedades serían el equivalente a los atributos en el modelo relacional. Esto implica una diferencia importante respecto al modelo en grafo y el relacional, que es la mayor riqueza semántica de las relaciones entre datos en el modelo en grafo. En el modelo relacional las relaciones entre datos se representan mediante claves foráneas y por lo tanto, cuando se quiere añadir información (atributos) a una relación es necesario crear una nueva relación para representar la relación y sus atributos, lo que se llama reificar la relación. En cambio, el modelo en grafo permite que las relaciones entre datos contengan propiedades, lo que permite una representación más natural, clara y eficiente de las mismas.

En la transparencia podemos ver algunos ejemplos de propiedades. En particular el nodo “Marqués de Portal Crianza” tiene dos propiedades para representar el rango de precios del vino y el volumen contenido en sus botellas. Las relaciones entre este vino y sus tipos de uva también han sido enriquecidos con un par de propiedades para indicar que el vino posee entre un 40 y un 50 por ciento de uva moscatel y entre un 10 y un 30 por ciento de uva Xarelo. Semánticamente, ésta es una propiedad de la relación entre un vino y el tipo de uva utilizado, ya que se refiere a la cantidad de un tipo de uva utilizada para hacer un vino y recordemos que la relación “Use Grapes of Type” significa “que uvas hay que utilizar para hacer un vino”. El uso de un modelo de grafos permite definir estas propiedades directamente en las aristas, sin tener que construir estructuras complementarias.

El modelo en grafo impone pocas restricciones de integridad. Básicamente dos, siendo la segunda consecuencia de la primera. No es posible definir aristas (arcos) sin nodos origen y/o destino, y los nodos sólo pueden eliminarse cuando quedan huérfanos. Es decir, cuando su grado de entrada y salida es cero.



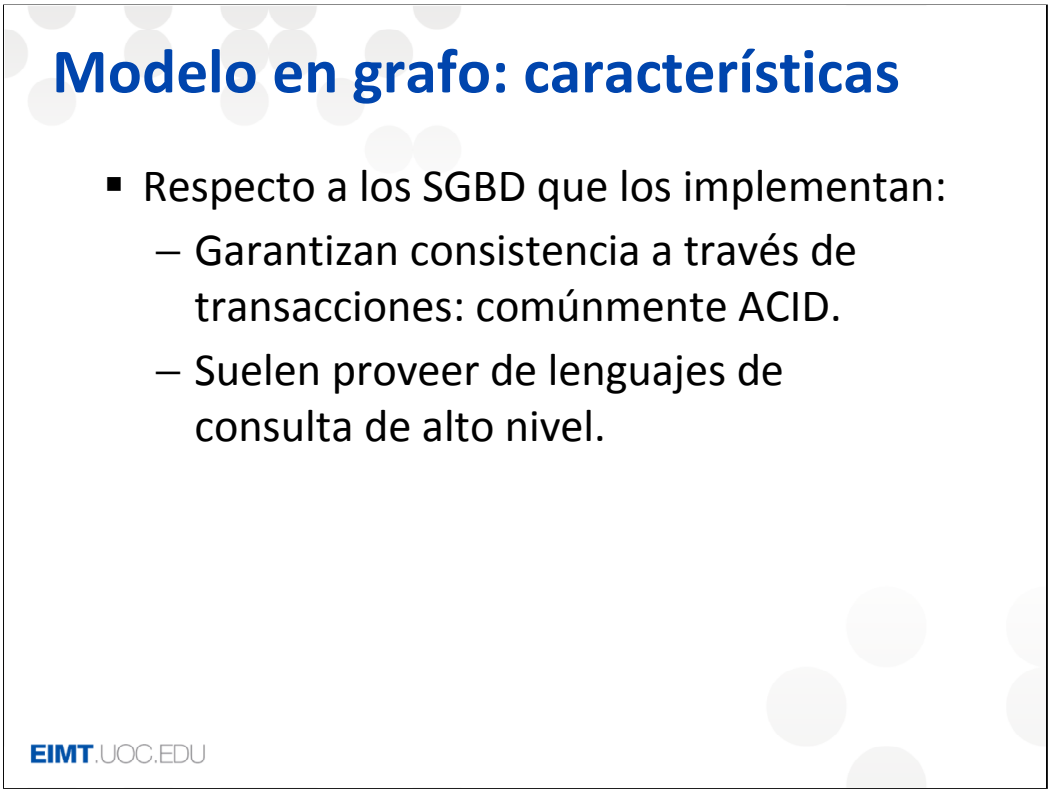
Una vez vistos los elementos que componen el modelo de datos en grafo, vamos a ver sus características más relevantes.

La característica principal es que las relaciones están explícitamente representadas en la base de datos. Eso simplifica la recuperación de relacionados de forma muy eficiente. Esta eficiencia es mayor que en modelos de datos relacionales, donde las relaciones entre datos están representadas de forma implícita (claves foráneas) y requieren ejecutar operaciones de combinación (en inglés *join*) para calcularlas.

Al igual que los otros modelos de datos NoSQL vistos, este modelo es *schemaless*. Por lo tanto, no es necesario definir un esquema antes de empezar a trabajar con una base de datos en grafo. No obstante, en este caso, hay parte del esquema implícitamente definido en los grafos. Por ejemplo, los arcos que representen el mismo tipo de relación acostumbran a utilizar una etiqueta común para indicar su tipo y semántica.

Como consecuencia de la característica anterior, este modelo permite añadir nuevos tipos de nodos y aristas fácilmente, y sin realizar cambios en el esquema. Aunque es conveniente hacerlo de forma ordenada y planificada, para evitar añadir tipos de relaciones o nodos redundantes o mal etiquetados.

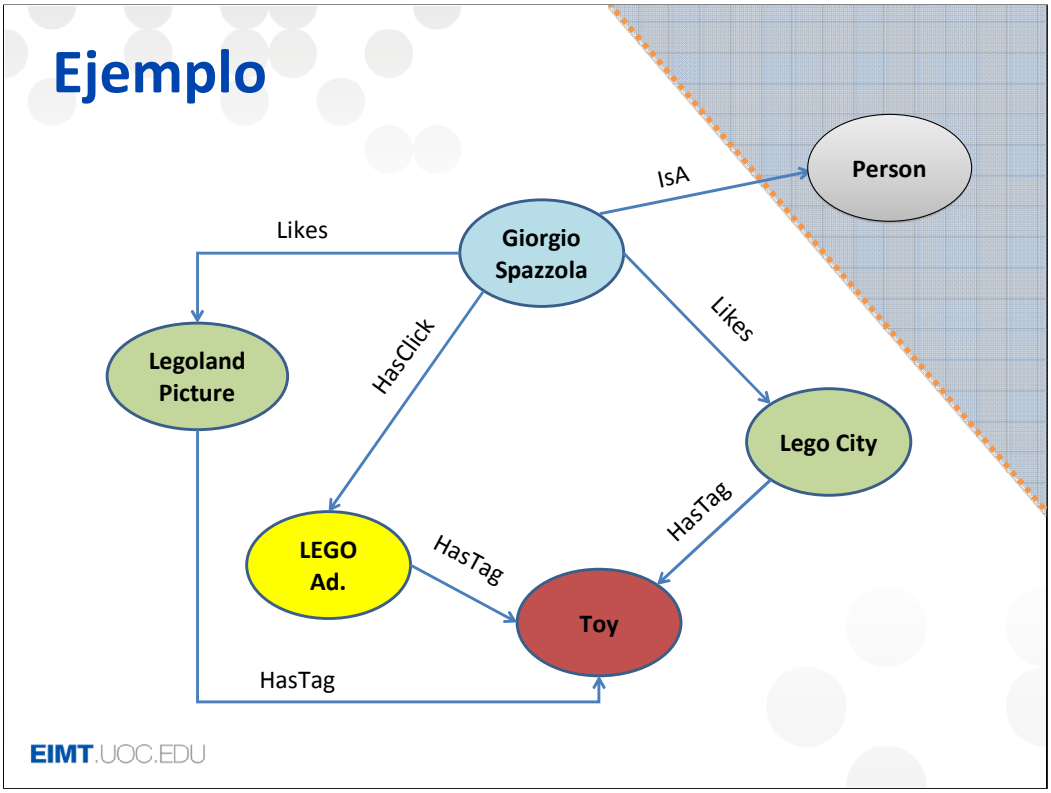
Los modelos en grafo no son tan fácilmente escalables como los modelos de agregación. Los datos están altamente relacionados, eso implica que distribuir los datos en diferentes ordenadores debe hacerse con mucho cuidado para no "romper" relaciones entre los datos. Por este hecho, la distribución de datos en estos modelos es compleja y requiere de información del dominio para realizarse de forma correcta. En el ejemplo que nos ocupa, una manera de identificar los datos susceptibles a distribuir sería mediante la relación "Vino vendido en país" que indica en qué países se venden los vinos. Según dicha información, podríamos distribuir geográficamente la información en función de dónde se venden los vinos. Es razonable pensar que la gente de España estará más interesada en la información de vinos que se pueden encontrar en España, y lo mismo para la gente de otros países. No obstante, debe tenerse en cuenta siempre que esta información depende del dominio y que por lo tanto deberá ser usada después de haber comprobado su veracidad mediante distintas evidencias (análisis del sitio web, estudios de mercado, etc.).



Otras características, no tanto del modelo en grafo en sí, sino de los sistemas gestores de bases de datos que los implementan, es cómo se garantiza su consistencia y cómo se accede a sus datos.

La gestión de transacciones en los modelos en grafo está pensada con una filosofía más cercana a las bases de datos relacionales que a los modelos de datos NoSQL de agregación. Los modelos en grafo normalmente utilizan un sistema de transacciones tipo ACID para garantizar la consistencia, hecho que puede dificultar la escalabilidad horizontal. Normalmente, una escalabilidad vertical puede ser más conveniente en estos sistemas.

Por otro lado, los sistemas de gestión de bases de datos que implementan un modelo en grafo acostumbran a proporcionar lenguajes de más alto nivel que los basados en modelos de agregación. Por ejemplo, podemos encontrar el uso de Cypher en Neo4J, que es un lenguaje declarativo parecido a SQL, y Gremlin (ver https://github.com/tinkerpop/gremlin/wiki para más información), que es un lenguaje de dominio de gestión de grafos y que está soportado por varios sistemas gestores de bases de datos en grafo.



A continuación podemos ver otro ejemplo de modelo en grafo. En este caso se representa información sobre una red social. Hemos pintado en distintos colores los nodos para mostrar gráficamente los conceptos a los que pertenecen: azul para las personas, verde para los recursos web, amarillo para los anuncios y rojo para las etiquetas. Este tipo de clasificación puede realizarse en algunos sistemas gestores de base de datos en grafo, como por ejemplo Neo4J, pero no es lo más común.

En el diagrama podemos ver que hay una persona llamada Giorgio Spazzola a la que le gustan dos recursos web: una foto de Legoland y una ciudad de Lego. Además, esta persona ha hecho *click* en un anuncio de Lego. También podemos ver que estos tres elementos (los dos recursos web más el anuncio de Lego) están etiquetados con la etiqueta *Toy*. A partir de estos datos, sería fácil inferir que Giorgio Spazzola es alguien interesado en juguetes y ofrecerle recursos web personalizados.

El fragmento comentado hasta ahora responde a la representación de objetos del mundo real, pero no a conceptos abstractos. Tal y como hemos dicho anteriormente, en el modelo en grafo podemos mezclar conceptos con objetos. Un ejemplo de ello sería añadir un nodo para describir el concepto persona (*Person*) y una relación que indique que Giorgio Spazzola es una persona. Tener información sobre conceptos en la base de datos y sus instancias nos permitiría realizar búsquedas más potentes, como por ejemplo, “Para cada persona, devolver las 5 etiquetas con que más se han etiquetado los recursos web relacionados con ella”. Sin la información de quién es una persona, la resolución de esta consulta sería muy difícil, sino imposible.

Modelo en grafo

- Introducción
- Ejemplo
- Modelo en grafo
- Uso de modelos en grafo

EIMT.UOC.EDU

En el siguiente apartado introduciremos cómo se accede a los datos de los grafos, qué consideraciones hay que tener en cuenta al diseñar una base de datos de grafos y posibles extensiones de este modelo.

Acceso a los datos

- El acceso a los datos en un modelo en grafo se realiza mediante navegación, donde se define:
  - Origen de la búsqueda
  - Patrón de navegación:
    - Qué relaciones (arcos) utilizar y cuándo
    - Condiciones que deben satisfacer los datos
    - Tipo de navegación (profundidad vs anchura)
    - Nivel de profundidad
  - Otros:
    - Nodos, aristas y propiedades a devolver
    - Orden
    - Etc.

EIMT.UOC.EDU

En las bases de datos en grafo la consulta de datos se realiza mediante navegación. Es decir, se elige un nodo (o un conjunto de nodos) como nodo origen y se navega a partir de ese nodo siguiendo los patrones de navegación que se indican en la consulta.

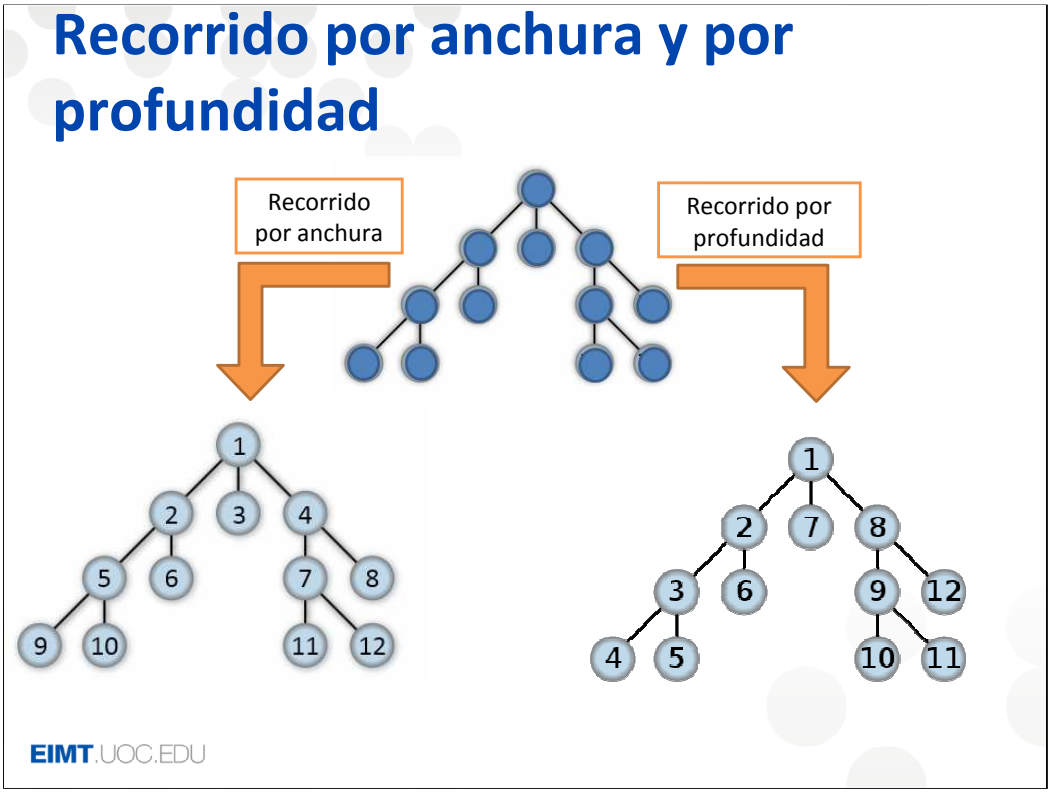
Los patrones de navegación se pueden referir a:

1) Qué relaciones se deben utilizar para la navegación y bajo qué condiciones (normalmente referidas a las relaciones)

2) Filtros que indiquen qué condiciones deben satisfacer los datos y las relaciones en la consulta. Estos filtros se refieren normalmente a propiedades, pero también podrían referirse a nodos y/o aristas.

3) Cómo debe realizarse la navegación. Se puede decidir si se realiza una búsqueda por profundidad o por anchura y el nivel de profundidad de la búsqueda (cuántas relaciones navegar antes de retornar el resultado final)

Otros aspectos indicados en la consulta son los datos (nodos, aristas y propiedades) que se deben devolver, número máximo de elementos a devolver, en qué orden, etcétera.



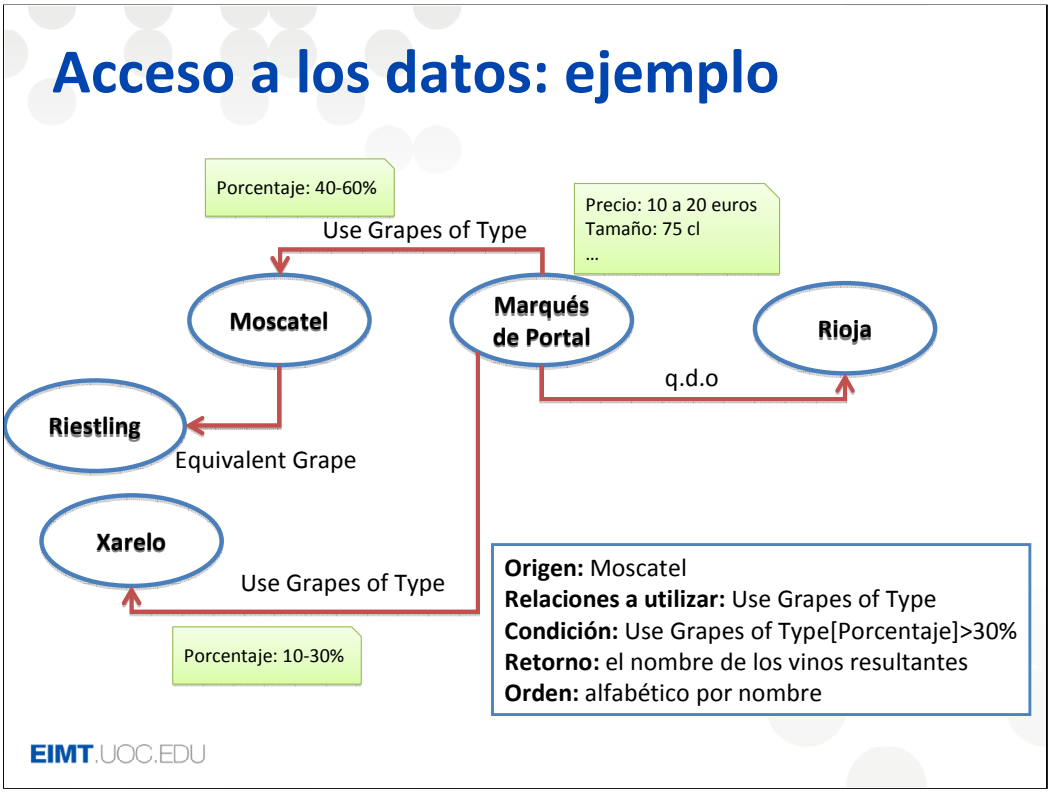
Antes de mostrar un ejemplo de consulta, vamos a explicar en más detalle en qué consiste el recorrido por anchura y por profundidad.

En el recorrido por anchura el grafo se recorre por niveles, donde cada nivel viene determinado por el número de aristas necesarias para llegar, desde el nodo origen (o principal), a los nodos del nivel. En este tipo de recorrido se parte del nodo principal y después se recorren todos los nodos directamente relacionados con él, es decir, aquéllos a los que se puede navegar desde el nodo principal usando sólo una arista. Posteriormente, se recorren todos los nodos directamente relacionados con los nodos vistos en el paso anterior (a los que se puede acceder desde el nodo principal usando sólo dos aristas), y así sucesivamente.

En la transparencia podemos ver un grafo de ejemplo. En el lado izquierdo de la transparencia podemos ver el orden en que se recorrerían sus nodos en un recorrido por anchura. Vemos que primero se consultaría el nodo inicial o principal (etiquetado como 1), después los nodos directamente conectados a éste (etiquetados como 2, 3 y 4). Posteriormente, se recorrerían los nodos del siguiente nivel: los conectados a los nodos 2 (el 5 y el 6), el 3 (ninguno) y el 4 (el 7 y el 8). Finalmente, se recorrerían los nodos del último nivel: los nodos 9 y 10 (que se acceden a través del nodo 5) y los nodos 11 y 12 (que se acceden a través del nodo 7).

En el recorrido por profundidad se escoge un camino y se sigue hasta que finalice. Una vez finalizado el camino, se vuelve atrás (al punto donde se realizó la última elección), se elige otro camino y se continúa hasta el final del mismo. Una vez llegado al final se vuelve atrás y se escoge otro, y así sucesivamente hasta que no queden más caminos que recorrer. En la parte derecha de la figura podemos ver un ejemplo de recorrido por profundidad. En este caso se empieza por el nodo origen o principal (nodo 1) y se escoge el camino de la izquierda (nodo 2). En el nodo 2, se vuelve a escoger el camino de la izquierda, recorriendo el nodo 3. En el nodo 3 se vuelve a escoger el camino de la izquierda, devolviendo el nodo 4. Una vez llegado al nodo 4 no podemos seguir navegando, por tanto volvemos al último nodo donde hicimos una elección (nodo 3) y escogemos el camino alternativo, en este caso el nodo 5. Visitado el 5 debemos volver atrás. Como el nodo 3 no tiene ningún camino por recorrer, debemos volver atrás. Por tanto, volvemos al nodo 2 y escogemos el camino alternativo, llegando al nodo 6. En el nodo 6 no podemos continuar y debemos retroceder hasta que encontremos un nodo con caminos no recorridos (nodo 1). Visitamos el nodo 7 y volvemos al nodo 1 para continuar el camino de la derecha (nodo 8). Se continua siguiendo el mismo algoritmo que hemos ido comentando, visitando los nodos 9 y 10, posteriormente el 11 y finalmente el 12.

Es importante tener en cuenta qué tipo de recorrido queremos realizar cuando el grafo contiene ciclos (combinaciones de aristas que provoquen caminos cerrados) o cuando el número de elementos a devolver esté limitado.



Supongamos que tenemos el esquema de la figura y queremos realizar una consulta que retorne el nombre de los vinos que tienen más de un 30% de uva de tipo moscatel ordenados alfabéticamente.

En dicho caso deberíamos establecer el origen de la consulta en Moscatel, es decir, indicar que debemos usar la arista "Use Grapes of Type" para encontrar los vinos que estamos buscando, que sólo debemos navegar por aquellas relaciones donde el porcentaje es superior al 30%, que queremos retornar el nombre de los vinos seleccionados y que el orden en que se reportan los datos debe ser alfabético.

Otra posible pregunta podría ser "Quiero saber las uvas que son equivalentes a la uva Moscatel". Esta pregunta podría requerir definir el tipo de navegación a utilizar, ya que se utilizaría una relación reflexiva (y transitiva) como es "Equivalent Grape" que podría dar lugar a un gran número de resultados (e incluso a ciclos) que podría ser interesante limitar a priori. Una manera de hacerlo sería pedir que se realizara una búsqueda con sólo 2 niveles de profundidad. Así se tendrían en cuenta sólo las uvas equivalentes a Moscatel y a su vez las equivalentes de éstas. Otra, posibilidad sería limitar el número de resultados a devolver en la consulta. Al trabajar con grafos la existencia de ciclos debe tenerse en cuenta ya que puede ser problemática.

Más adelante en el curso, cuando hablemos de Neo4J, entraremos en más detalle en cómo realizar consultas sobre grafos.

Consideraciones de diseño

- No requiere la creación explícita de un modelo de datos, pero es importante hacerlo a priori para:
  - Identificar las relaciones que existen entre los datos
  - Evitar utilizar nombres distintos para la misma relación semántica
  - Añadir al grafo conceptos cuando puedan ser útiles en la explotación de datos
- Se debe tener en cuenta que, aunque las relaciones sean dirigidas, pueden recorrerse en ambos sentidos.
  - Dirección implica semántica, no navegabilidad.

EIMT.UOC.EDU

Una vez introducido cómo consultar los datos en un modelo en grafo, es relevante hablar brevemente algunos aspectos relativos al diseño de bases de datos en grafo. El objetivo de esta transparencia es comentar brevemente sus principales características.

Al no tener un esquema explicito, el diseño de la base de datos es mucho más sencillo. No es necesario definir un modelo conceptual del dominio de aplicación, sino identificar los conceptos más importantes del mismo. Una vez hecho, es importante identificar las distintas relaciones que pueden existir entre los datos. Una vez identificadas, se debe establecer qué nombre se utilizará para cada relación. Es importante no utilizar distintos nombres para una misma relación ya que eso dificultaría enormemente la consulta de los datos. Recordad que el nombre asociado a una relación indica su tipo.

Por otro lado, tal y como ya se ha comentado, los grafos contienen objetos pero no tienen por qué contener conceptos. No obstante, es conveniente añadir conceptos a los grafos cuando éstos puedan ser útiles en la explotación de los datos. Por ejemplo, en el caso de que debamos consultar muy frecuentemente las personas que satisfacen unas determinadas condiciones, sería conveniente crear un nodo Persona y asociar dicho nodo con todas las personas del grafo mediante una relación. Eso facilitaría enormemente la búsqueda, ya que ésta podría empezar seleccionando todos los nodos relacionados con el nodo Persona.

Aunque no sea necesario, realizar un esquema conceptual (por ejemplo, un diagrama de clases UML) antes de crear la base de datos puede ayudar enormemente a la realización del diseño más adecuado para la base de datos en grafo.

Otro aspecto a tener en cuenta, es el hecho de que las relaciones suelen poder navegarse en ambas direcciones. Eso hace que no sea necesario definir relaciones inversas sólo por temas de navegabilidad. Simplemente es necesario definir una relación inversa cuando ésta sea semánticamente relevante para la base de datos a crear.

Y hasta aquí la presentación de modelos en grafo. No obstante, antes de terminar, hemos creído importante hablar de una de sus extensiones: los *triplestores*.

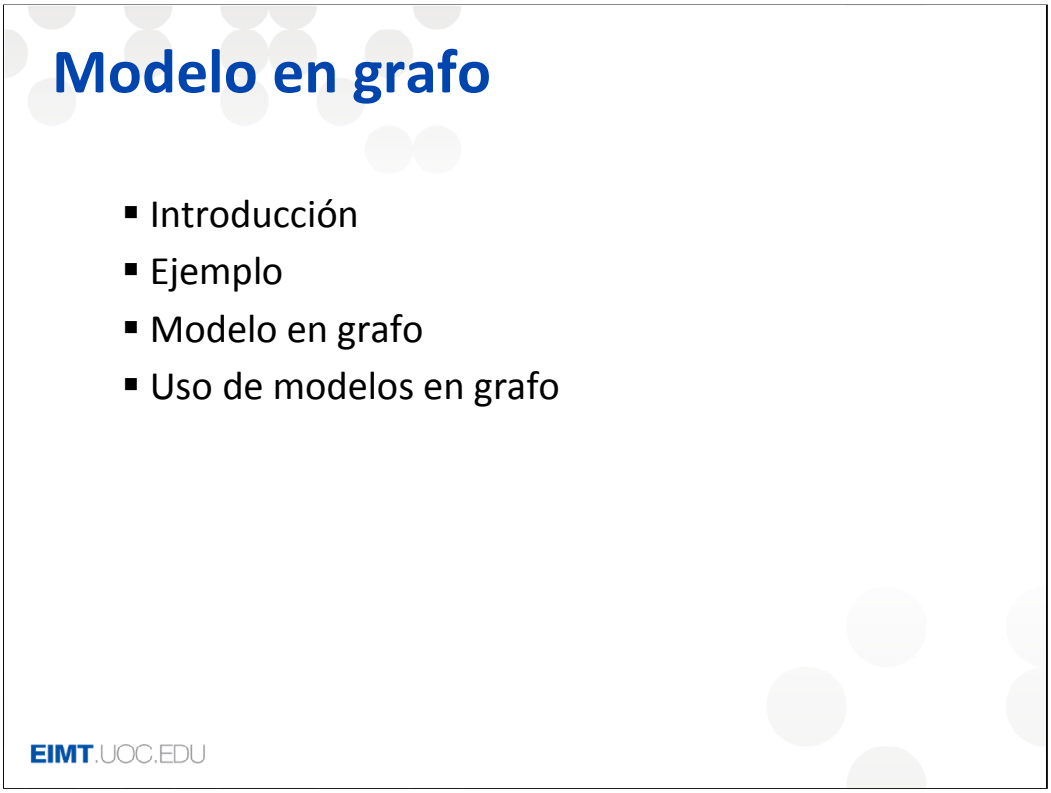
El modelo en grafo tiene extensiones que permiten tratar otros tipos de datos que pueden representarse mediante grafos. Quizá la extensión más relevante son los almacenes RDF o *triplestores*. Los *triplestores* siguen un modelo similar al presentado para tratar de forma más eficiente tripletas RDF. Las tripletas RDF son uno de los mecanismos más utilizados para representar la semántica de los datos que existen en la Web y responden al modelo de datos utilizado por el Resource Description Framework (o RDF) del W3C.

Las tripletas son tuplas de tres elementos: un recurso de origen, un recurso de destino y un predicado que indica cómo están relacionados estos recursos. La manera de identificar estos tres elementos es mediante URIs (*Uniform Resource Identifier* en inglés). Las URIs son identificadores de recursos web. Por ejemplo, una tripleta RDF podría indicar que esta asignatura (URI de la asignatura) tiene una carga docente (predicado) de 4.5 créditos (valor escalar de destino).

Las tripletas RDF pueden concatenarse, formando un grafo dirigido etiquetado.

Los *triplestores* proporcionan muchas más funcionalidades y son más eficientes que las bases de datos en grafo al tratar con ficheros RDF. Eso es debido a que el modelo de datos RDF es simple y está consensuado por el WC3. Además, existe un lenguaje de consulta potente y estándar para trabajar con datos RDF: SPARQL y existen distintos formatos estándar de intercambio de datos que facilitan la compartición de datos.

Por lo tanto, cuando sea necesario tratar datos RDF o generar algún servicio relacionado con la Web semántica, los almacenes de datos RDF deberán ser una opción a considerar.



Aquí finaliza la presentación de los modelos en grafo.

En esta presentación hemos introducido los modelos en grafo utilizando un ejemplo motivador para ver en qué situaciones es conveniente usarlos. Posteriormente hemos mostrado los elementos que componen este modelo, sus principales características y sus ámbitos de aplicación. A continuación, se ha introducido cómo consultar los datos en este modelo y qué hay que tener en cuenta al diseñar bases de datos en grafo. Finalmente, se han introducido los *triplestores*: una extensión de los modelos en grafo para tratar datos RDF.

Referencias

I. Robinson, J. Webber & E. Eifren (2013). *Graph Databases,* O'Reilly.(http://graphdatabases.com/)

M. Gyssens& J. Paredaens (1990). "A Graph Oriented Object Database Model", *ACM SIGMOD International Conference*. (http://bit.ly/JL2QWc).

P.J. Sadalage & M. Fowler. (2013). *NoSQL Distilled. A brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education. (http://bit.ly/1koKhBZ).

Joe Celko's (2013). *Complete Guide to NoSQL*. Elsevier. (http://www.sciencedirect.com/science/book/9780124071926)

E. Redmond, J Wilson (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*, The Pragmatic Bookshelf. (http://pragprog.com/book/rwdata/seven-databases-in-seven-weeks)

EIMT.UOC.EDU

Esperamos que hayáis disfrutado y aprendido con este vídeo. A continuación encontraréis algunas referencias que os permitirán profundizar más en los temas tratados.

Que tengáis un buen día.