ISAAC:

¿Qué se os viene a la mente cuando al ver esta imagen? Yo lo tengo muy claro, me recorre el tesón existencial de tener que estar viendo siempre lo mismo, el profundo aburrimiento que acosa a cada día a más ciudadanos de a pié, incapaces de encontrar nuevas e hilarantes formas de culturizarse, de sentir la mayor excitación que cualquier hombre ansía desde que se levanta por la mañana, la del saber, por supuesto.

Por eso, tenemos la desfachatez de presentaros: ConoceMadrid, un grafo de conocimientos diseñado única y exclusivamente con su bienestar en mente, que recoge toda la información relevante para visitar los museos, monumentos y otras entidades culturales de Madrid, al que mediante un sencillo lenguaje de consultas como es SPARQL podemos efectivamente consultar.

No os pasa que, si ya os da pereza ir a museos de por sí, acabáis yendo a los mismos 3? No os pasa que vienen vuestros amigos de fuera a Madrid a hacer un poco de turismo y no sabéis qué lugares recomendarles? No os pasa que queréis hacer un plan cultural con la familia y os cuesta mucho decidir?

Pues para tanto los madrileños poco enterados como para los turistas confusos, hemos generado un grafo que recoge toda la información relevante para visitar los museos, monumentos y otras entidades culturales de Madrid, al que mediante un sencillo lenguaje de consultas como es SPARQL podemos efectivamente consultar.

Para ello, hemos seleccionado una base de datos muy completa y con la licencia apropiada para explotar y distribuir. Esta licencia es la siguiente-

Licencia Nuestra: CC BY-NC (Creative Commons Reconocimiento-NoComercial), Es posible alterar o difundir la obra original siempre y cuando se haga referencia al autor de la misma, careciendo de fines comerciales. La obra derivada no está obligada a mantener la misma licencia que la obra original.

RODRIGO:

Lo primero que hemos hecho ha sido un esquema general de cómo queríamos que quedara nuestro grafo. En él, se pueden observar las diferentes clases (y subclases) en azul, y sus propiedades tanto en las cajitas que tienen debajo como en las flechas que los unen. Las propiedades de las cajas serían de tipo 'datatype properties', (ya que su rango es un dato o literal), mientras que las propiedades de las flechas son de tipo 'object properties' (ya que por el contrario su rango es otra clase).

Además, aparecen otro tipo de flechas en el esquema y estas son las que indican las subclases.

Vamos a comentar el esquema. Este parte de un objeto ‘entidad’ que se refiere al museo, monumento, etc como entidad cultural, como organización. Este objeto tiene un nombre, horario, nivel de accesibilidad (esto es, en una escala del 0 al 3, lo sencillo que resulta acceder al edificio a personas con limitaciones físicas. Refleja el hecho de que tenga rampas, ascensores, etc), descripción, página web, número de telefono, email o ubicación. Esta última propiedad es de tipo 'object property', cuyo rango sería el lugar físico en el que está ubicada dicha entidad.

De esta manera, el ‘lugar físico’ es una nueva clase con dirección, barrio, distrito y coordenadas…

ALBA:

En base al esquema anterior y a determinados criterios que comentaremos en las siguientes diapositivas, hemos realizado la siguiente ontología.

En primer lugar comentar los dominios o namespaces de donde hemos escogido los términos ontológicos utilizados.

Como podéis ver, hemos dado prioridad a dominios más usados y conocidos (como pueden ser dbpedia-owl, schema o org), ya que al ser los más usados por los usuarios a la hora de hacer consultas en la web de linked data, fomentamos la utilidad de nuestro grafo y un mayor número de personas pueden disfrutar de sus beneficios.

Aun así, no todos los términos que deseábamos incluir existían en estos dominios, o estaban definidos de manera apropiada en los mismos. Por lo tanto, también hemos incluido dominios menos conocidos (como pueden ser a-loc o arp) que sí que satisfacían nuestras necesidades, y hemos creado un dominio propio al que se le refiere como ‘ns’ para representar algunos términos más específicos de nuestra ontología que no encajaban con ninguna ontologías ya existente.

Además, hemos usado los tipos de datos apropiados mostrados en pantalla.

Al sustituir los términos ontológicos seleccionados en nuestro esquema, nos quedó la siguiente ontología. Como ya ha dicho rodrigo, las diferentes clases (y subclases) en azul, y sus ‘datatype properties’ en las cajitas que tienen debajo y las 'object properties' en las flechas que las unen.

No tiene sentido que os comente otra vez sus términos, pero si resaltar varias cosas:

- La primera, que hemos cuidadosamente seleccionado org:Site para representar a las entidades culturales como entidades y dbpedia-owl:Place para representar su localización física, ya que como ha dicho rodrigo, existe una diferencia muy importante entre ambas.

- La segunda, el rango de 'datatype properties' es xsd:float en el caso de números (decimales) con los que tiene sentido operar (como pueden ser las coordenadas x e y) pero es xsd:string para otros números con los que no tiene sentido operar (como el número de teléfono o el nivel de accesibilidad), además de para cadenas de literales. Además, hemos usado el tipo de datos xsd:anyURI como rango para la propiedad schema:url.

- Por último, hemos usado la herencia de propiedades en nuestra ontología, a partir de la cual las propiedades definidas para la clase padre son accesibles desde las clases hijas. Esto ocurre en el caso de la clase ns:Stop y subclases metro, bus, cercanias, etc; que heredan la 'datatype property' 'dbpedia-owl:information' de su clase padre y sus instancias la pueden aplicar sin necesidad de hacer referencia a la clase padre en sí.

Esta ontología la hemos transformado a ttl como se ha pedido.

ESTEFI:

Para el resource naming strategy:

- Hemos usado URIs de tipo hash, ya que contamos con un conjunto de datos relativamente pequeño.

- Hemos creado y usado el dominio http://museosymonumentos.es/, ya que tiene sentido ubicar el modelo e instancias derivadas de nuestro conjunto de datos en el mismo.

- Además, hemos nombrado la ontología "ConoceMadrid". Este nombre se va a incluir en todas las ontologías creadas a modo de subcarpeta, para que quede claro que, dentro de todos los museos y monumentos, nuestro conjunto de datos se refiere a aquellos ubicados en Madrid.

Basados en estos principios, así quedarían las uris:

Estas se dividen en 3 tipos: aquellas que definen clases (y subclases), propiedades e instancias.

Las primeras se construyen de la siguiente manera: http://museosymonumentos.es con 'ontology', ya que una clase forma parte del modelo (de la ontología); seguido del nombre de la ontología que es 'ConoceMadrid' que actúa a modo de subcarpeta como se ha mencionado

finalmente; y por último #className. En nuestro modelo únicamente hemos definido 2 clases con nuestro namespace, que son Stop y BicimadStop, como se muestra en los ejemplos. Es importante destacar que estas clases, como todas las demás, deben escribirse con la primera letra en mayúscula.

Las segundas se construyen de manera parecida: http://museosymonumentos.es con 'ontology', ya que nuevamente una propiedad forma parte del modelo (de la ontología); seguido del nombre de la ontología que es 'ConoceMadrid'; y por último #propertyName. En nuestro modelo hemos definido las propiedades 'accessibility' y 'hasNearbyStop' con nuestro namespace, como se muestra en los ejemplos. A diferencia de las clases, las propiedades se definen en todo minúsculas.

Finalmente tenemos las instancias, que se construyen a partir del dominio con 'resource', ya que en este caso, representan datos, no al modelo en sí; seguido del nombre de la clase de la instancia que se desee crear y de #identifier, que es el identificador de dicha instancia en su clase padre. Nosotros hemos decidido que dicho identificador sea el llamado 'PK', que es un entero positivo único para cada entidad y que la representa en la web municipal de la Comunidad de Madrid así como en nuestra base de datos. De esta manera, las instancias por ejemplo serían: (leer ejemplos)

GUILLE:

Después de definir nuestra ontología nos pusimos con la limpieza y preparación de los datos, para que pudieran ser explotados utilmente. Este proceso consistió en un continuo análisis de posibles problemas que podía tener nuestro dataset para adaptarse a la ontologia y su solución. Los problemas más destacables han sido:

- La existencia de columnas inútiles, ya sea porque contenían todo valores vacíos, valores de tipos muy dispares y difíciles de tratar o valores simplemente innecesarios para el objetivo de nuestra aplicación. Estas columnas han sido eliminadas.

- La localización de la info de todas las subclases de paradas en la misma columna y de manera desordenada dificultaba la explotación de las paradas y la clasificación de las mismas, para realizar búsquedas más concretas. Para ello, separamos la info de cada subclase de para en una columna separado.

- Al contrario, la separación de la info de los diferentes elementos que componen una dirección postal (calle, número, código postal, …) en varias columnas resultaba innecesario y suponía un mayor coste tanto computacional como de almacenamiento, por lo que las unimos todas en la misma.

- La existencia de comas en las celdas del csv, que se confundían con separadores de celdas al pasar el materializer y las cuales cambiamos por espacios en blanco, puntos o punto y comas según nos pareció más apropiado.

- La existencia de celdas en blanco, que de nuevo generaban error con el materializer, por lo que las rellenamos con la cadena 'NULL' para evitarlo. Estos valores serán filtrados en las queris, para no generar resultados innecesarios.

- El encoding

Cuando conseguimos que la preparación y limpieza de los datos fuera lo suficiente claros, pasamos a hacer la reconciliación con wikidata (en ciertas columnas por ejemplo la de distrito o barrio de la entidad cultural)

Por último quiero destacar la importancia de este proceso el cual finaliza la base del proyecto que compone junto a la ontologia.

ALBA:

Y ya por último hemos realizado el mapping. Este está compuesto por 10 tripplesmaps que os comentaremos rápidamente.

El primero es el que tiene como sujeto org:Site y como predicados todas estas propiedades, que son las mismas que estaban definidas en la ontología.

- Las 'datatype properties' tienen una columna del csv asociada, de la que extraen sus valores (como dbpedia-owl:name que extrae sus valores de la columna NOMBRE del csv o owl:sameAs que los extrae de la columna same-as-name del csv).
-
- Por otra parte, las 'object properties' tienen un join Condition con otro tripplesmap (como schema:place que tiene un joinCondition con el tripplesmap de schema:Place, y ns:hasNearbyStop que se instancia varias veces y cada una de ellas tiene un joinCondition con el tripplesmap de las subclases de ns:Stop)

GUILLE:

Después tenemos el tripplesmap de sujeto schema:Place y como predicados todas estas propiedades.

- Nuevamente, las ‘datatype properties’ tienen una columna del csv asociada.
- Mientras que las ‘object properties’ tienen un join Condition con otro tripplesmap. Estas son dbpedia-owl:neighbouthood, dbpedia-owl:district y schema:geo. Las dos primeras inicialmente estaban incluidas como ‘datatype properties’ en nuestra ontology, pero como hemos reconcilliado sus valores con los de Wikidata, ha sido necesario separarlas en nuevos tripplesmaps para asociarles el predicado owl:sameAs y correspondientes links (como se ve a la derecha).

Los objetos neighbourhood y district a los que se conecta no existían en la ontología original, pero al reconciliarlos con la nube de linked data ha sido necesario incluirlos en tripplesmap independientes.

RODRIGO:

La tercera es schema:geo y este es su tripplesmap, construido como el resto y del cual no es necesario decir mucho más.

Por último, los tripplesmap de las subclases de ns:Stop, cada uno de ellos con su ‘datatype property’ schema:description heredada de su clase padre. Además, como se puede observar, hemos hecho link entre wikidata y las estaciones de cercanías.

Y LAS QUERIES