Sistemas de recuperación de Información Web, 2019-2

Trabajo 1 (25%). Integrantes: Máximo 4

Objetivo: Desarrollar un sistema de información web a través del cual un usuario puede realizar consultas de un dominio específico en diferentes bases de datos.

Cada equipo deberá seleccionar un dominio de un campo de conocimiento sobre el cuál trabajará, y el cual NO puede ser el mismo que el de algún otro equipo (Registrar en el archivo compartido de Drive los nombres de los integrantes y el tema que trabajará). Para efectos de explicación del trabajo se supondrá que el tema seleccionado fue Películas.

Se recomienda leer toda la especificación del trabajo antes de iniciar a desarrollarlo.

1. Bases de conocimiento

A. **Modelo (5%)**: Seleccionar mínimo 4 entidades con 2 a 4 propiedades cada una, mínimo 6 relaciones entre las propiedades y 1 relación de herencia entre las entidades. Crear un pequeño diagrama Entidad Relación o Clases UML que indique el modelo general de los datos.

Para el caso de ejemplo se seleccionan las siguientes entidades y atributos:

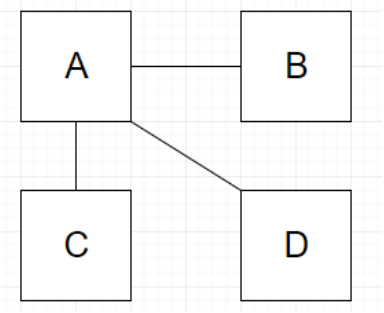
Película (Nombre, Año, sinopsis), Persona (Nombre, Fecha Nacimiento), Director (Nombre, Fecha Nacimiento), Actor (Nombre, Fecha Nacimiento), Distribuidora (Nombre, País).

Las relaciones son:

Película es dirigida por Director, Director dirige Película, Actor actúa en Película, Película tiene actor Actor, Película es distribuida por Distribuidora, Distribuidora distribuye Película

La relación de herencia está dada entre las entidades Actor y Director con Persona

Se debe cumplir mínimo que exista una entidad A que tenga relación directa con todas las demás entidades, como se muestra en la siguiente imagen (Para el ejemplo, A = Película)



Además, se debe cumplir que al menos un atributo sea el mismo para varias entidades (Para el ejemplo, Película, Actor y Distribuidora tienen Nombre)

B. **Base de conocimiento RDF (10%)**: Crear en formato XML/RDF un conjunto de datos que contenga mínimo 5 instancias de cada entidad con todas sus relaciones y atributos.
Nota: La base de datos debe ser en formato RDF por lo que no se pueden utilizar elementos ni sintaxis OWL (¡Cuidado al crearla en Protege en este caso!)

C. **Base de conocimiento OWL (10%)**: Crear en formato OWL un conjunto de datos que contenga mínimo 7 instancias de cada entidad con todas sus relaciones y atributos. La ontología se debe publicar en un servidor Virtuoso
Nota:
   - Puede utilizar Protege en este caso.
   - Debe existir al menos tres relaciones *sameAs* para tipo individuals, en la cual se creen instancias diferentes que realmente son las mismas.

D. **Base de conocimiento Dbpedia (5%)**: Explorar dbpedia e indicar los nombres de las entidades y relaciones creadas en el modelo (Numeral A) con los recursos de dbpedia. Para el caso de ejemplo, dbo:director, dbo:abstract, dbo:distributor, dbr:Company, dbp:name, dbo:birthDate, etc.). Es posible que no encuentre todas las relaciones en Dbpedia, pero deben existir la mayoría que permitan hacer las consultas expresadas el trabajo

E. **Base de conocimiento relacional (15%)**: Crear una base de datos en MySQL, PostgreSQL, Oracle o SQL Server que represente el modelo y contenga al menos 5 instancias de cada entidad con todas sus relaciones y atributos. Utilizar un servidor D2R para mapear la base de datos relacional a una base semántica y crear un endopoint para su consulta. Consultar documentación en http://d2rq.org/

Nota: Se debe cumplir que:

- Cada entidad, relación y atributo se llame igual en mínimo dos y máximo tres de las cuatro bases del conocimiento, y diferente en las demás (Para el ejemplo, la relación es director de se puede llamar *pel:director_de* en la base RDF y relacional, *dbo:director* en Dbpedia y *movies:director_de* en OWL). Tener presente el namespace
- La base de conocimiento en Virtuoso y la base en D2R deben estar publicadas en servidores diferentes y accesibles desde Internet (Puede ser en Heroku, Google Cloud, AWS, etc.). Se debe poder acceder a cada EndPoint de manera individual para realizar consultas en SPARQL a través de una URL. La base RDF debe ser accesible a través de una URL.

2. Sistema de consulta

A. **Ontología de integración (10%)**: Debido a que en cada una de las bases del conocimiento se tienen nombres y espacios de nombre diferentes, es necesario crear una ontología de integración en OWL que importe las demás ontologías y relacione los recursos con las propiedades *equivalentClass, equivalentProperty, sameAs, inverseOf*, etc. Para el caso de ejemplo, *pel:director_de owl:equivalentProperty movies:director_de*

B. **Sistema de consulta**: Realizar una aplicación en Java con interfaz de usuario GUI (Desktop o Web) que tenga las siguientes opciones en el menú principal:

- Consulta de instancias (10%): Permita al usuario seleccionar una entidad (Para el ejemplo Película, Actor, Director, Persona, etc.) y consultar todas las instancias asociadas a la misma (Avatar, James Cameron, Zac Efron, Fox, etc.). Se debe poder visualizar todos los atributos de cada instancia. Debe existir la opción de filtrar por mínimo un atributo para cada entidad: el usuario ingresa el valor a filtrar (Filtrar las películas que salieron antes de X año, los actores con nombre que contenga la palabra "Ang", etc.).
  Cuando se seleccione una instancia se debe poder visualizar todas las relaciones que tiene con otras instancias, así como las instancias equivalentes.
  Debe existir una opción "indirecto" que al seleccionarla el sistema trae las instancias de entidades "hijas" de la seleccionada. Para el ejemplo, si se selecciona la entidad Persona y la opción "indirecto", el sistema arrojaría a James Cameron (Director) y a Zac Efron (Actor).

- Instancia con igual valor de atributo (10%): Seleccionar una entidad, luego seleccionar una instancia de dicha entidad y luego un atributo. El sistema debe arrojar la lista de las instancias de la misma clase que tienen el mismo valor en su atributo que la instancia seleccionada. Por ejemplo, si se selecciona *Película, Avatar, Año* traerá todas las películas que se hicieron en 2009

- Estadísticas por atributo (12,5%): Seleccionar una entidad y luego un atributo de dicha entidad. El sistema deberá arrojar el conteo de instancias que tienen valores en el atributo y en el caso de ser un atributo numérico el máximo, mínimo y promedio (Al menos un atributo entre todas las entidades debe ser numérico). Si es un texto deberá mostrar la longitud promedio de la cadena.
  También debe permitir ingresar una opción de filtrado, en el caso que el atributo sea numérico permitirá seleccionar la opción "mayor que" y "menor que", y en el caso que sea de texto la opción "contiene". Cuando el usuario ingresa un valor en el campo de filtrado el sistema arroja las estadísticas únicamente para las instancias que cumplan dicha condición. Si no ingresa un valor en este campo, el sistema calcula las estadísticas para todas las instancias de la entidad. Por ejemplo, si selecciona *Película* y *Año*, e ingresa 2005 y selecciona "mayor que", el sistema arrojará el conteo, máximo, mínimo y promedio del año de las películas que fueron lanzadas después del 2005

- Relación entre instancias (12,5%): Seleccionar una entidad y una instancia de dicha entidad, luego seleccionar otra entidad y una instancia de dicha entidad. El sistema debe mostrar "hay relación" o "no hay relación" si existe o no una relación directa entre ambas entidades según corresponda. Por ejemplo, si selecciona *Película-Avatar* y *Director-James_Cameron* arrojará "hay relación" porque encuentra la relación *es_dirigida* entre *Avatar* y *James_Cameron*. Debe tener en cuenta que, si una entidad no tiene una relación directa, se puede hacer la deducción utilizando la inversa de la propiedad (Debe existir al menos dos relaciones *inverseOf* en la ontología de integración). Por ejemplo, si Avatar no tiene la relación *es_dirigida* por *James_Cameron* pero *James_Cameron* tiene la relación *es_director* de *Avatar*, se debe mostrar "hay relación".
  También debe existir una opción "indirecta", que al seleccionarla hará la consulta teniendo en cuenta que las dos instancias no necesariamente tienen una relación directa, pero puede existir una entidad en el medio que relacione las dos de manera indirecta (En la figura las entidades C y B pueden estar relacionadas de manera indirecta por una entidad A, al igual que C con D y B con D). Para el caso de ejemplo, si selecciona *Director-James_Cameron* y *Distribuidora-20th_Century_Fox* el sistema arrojará "hay relación" porque James Cameron es director de Avatar y Avatar es distribuida por 20th Century Fox.

Nota:

- Todas las consultas deben realizarse en SPARQL. El sistema debe consultar en todas las bases de conocimiento haciendo uso de la ontología de integración. La base RDF se debe cargar en la aplicación a través de Jena, y la base OWL, Dbpedia y D2R ser consultadas con Jena a través de los Endpoints de cada una.
- NO se debe "quemar" en código los nombres de las relaciones, entidades o atributos. Estos deben ser cargados desde la ontología de integración. Por ejemplo, para mostrar al usuario la lista de entidades se debe consultar las clases en la ontología; para mostrar los atributos de una entidad se debe consultar las data properties asociadas a dicha entidad, etc.
- La ontología de integración debe tener el tipo de dato de cada atributo, de modo que sea posible deducir que estadísticas mostrar.
- Puede suceder que algunos atributos o relaciones no estén en Dbpedia, por lo tanto el sistema no arrojará siempre resultados desde dicha base de conocimiento, sin embargo se debe garantizar que al menos cada entidad tenga un atributo en Dbpedia, así como cinco instancia. También deben existir al menos tres de las seis relaciones.
- La aplicación debe utilizar sistema de control de versiones Git y publicar el repositorio.

Entregable: Un documento que contenga los siguientes ítems:

- Integrantes
- Nombre del tema o dominio

- Modelo del dominio (ER o clases)
- Ontología RDF
- Ontología OWL
- Relación de recursos de Dbpedia
- Datos de acceso a la base de datos relacional (Servidor donde se encuentre)
- Datos de acceso a los servidores Virtuoso y D2R
- URL del repositorio de Git con código fuente
- URL de los Endpoints Virtuoso y D2R
- Ontología de integración en OWL
- Aplicación ejecutable (Si se desarrolló desktop) o enlace (Si se desarrolló web)

Calificación final: 80% aplicación, 20% sustentación