**Instrucciones para la ejecución de los Scripts**

1. DESCARGAR FUENTE DE DATOS

Accedemos a la web <https://dblp.uni-trier.de/xml/> y descargamos el fichero **dblp.xml.gz** en formato xml.

Ubicamos el fichero en una ruta accesible para los scripts que se ejecutarán a continuación.

1. CAMBIAR CODIFICACIÓN DEL FICHERO ORIGEN:

Teniendo en cuenta los problemas de codificación del fichero, ejecutamos un proceso de cambio de codificación, ejecutando el script **convert.py**.

Para la ejecución de este fichero pasaremos como entrada el fichero original **dblp.xml.gz** y generamos un nuevo fichero en formato utf-8 llamado **dblp\_utf8.xml.gz**.

Descomprimimos el nuevo fichero **dblp\_utf8.xml.gz**, para obtener el fichero **dblp\_utf8.xml,** con codificación utf-8 sobre el que ejecutaremos los scripts de tratamiento y carga de datos a las diferentes bases de datos.

1. TRATAMIENTO Y CARGA DE DATOS EN MONGODB

Para este paso levantamos el servidor MongoDB y a continuación ejecutamos el script **XML\_to\_MongoDB.py**, que leerá el fichero **dblp\_utf8.xml** y ejecuta todo el proceso de parseo y conversión del fichero a estructura JSON que se ajuste al diseño descrito en la memoria para la base de datos MongoDB.

Este script crea la conexión con la base de datos, crea la nueva colección e inserta directamente los documentos en la colección a medida que se leen, parsean y transforman.

La ejecución de este proceso toma aproximadamente 30 minutos (depende de las características de la máquina) y como resultado crea automáticamente en MongoDB una nueva base de datos con una única colección de autores, que dentro contiene todos los documentos leídos del xml.

Abrimos Robo3T, damos doble click sobre la colección y a partir de este paso puede lanzarse la ejecución de las consultas planteadas en la práctica sobre la colección de documentos.

1. TRATAMIENTO Y CARGA DE DATOS EN NEO4J

Para ejecutar esta sección de la práctica, utilizamos el script **XML\_to\_Neo4jDesign.py**, que lee del fichero **dblp\_utf8.xml** y teniendo en cuenta el diseño descrito en la memoria para esta base de datos parsea el xml, transforma los campos y calcula nuevos campos (p.e. los ids. Como resultado de esta ejecución, generamos tres ficheros csv así:

* authors.csv: Nodos autores
* titles.csv: Nodos Obras
* relations.csv: Relaciones entre autores y obras

Una vez finalizada la ejecución de **XML\_to\_Neo4jDesign.py** y generados los ficheros csv, realizamos la carga de datos a la base de datos Neo4j así:

* Se copian los ficheros csv en la carpeta bin del home de Neo4j.
* Se ejecuta el siguiente comando para realizar el import:
  + LINUX:

~HOME/neo4j-community-3.3.4/bin$ ./neo4j-admin import --nodes "authors.csv" --nodes "titles.csv" --relationships "relations.csv" --ignore-missing-nodes --ignore-duplicate-nodes

* + WINDOWS: Por CMD desde la ruta C:\neo4j-community-3.5.3\bin ejecutamos el siguiente comando:

neo4j-admin import --mode csv --database=prueba1.db --nodes authors.csv --nodes titles.csv --relationships relations.csv --ignore-missing-nodes --ignore-duplicate-nodes

La ejecución carga los ficheros en un tiempo aproximado de 10 minutos. Una vez completada la ejecución lanzamos el servicor de Neo4j y abrimos el navegador.

* En localhost:7474 tendremos ya cargada la base de datos según las relaciones y el diseño descrito en la memoria de la práctica y es posible realizar las consultas planteadas.