**Fundamentos teóricos / practicos (II)**

**1. Acceso a la información: Conectores**

**Introducción. ¿Para qué sirven? ¿ Como los usaremos?**

**Comparativa**

**Cassandra**

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

**Mongodb**

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

**Neo4j**

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

**2. Representaciones de la información**

**Vistas**

**Bibliografía**

Conectores de Cassandra. Toda esta información esta reflejada en el documento oficial que podemos encontrar bajo <https://datastax.github.io/python-driver/api/cassandra/cluster.html>.

Mongodb. Más información de la libreria: <https://api.mongodb.com/python/current/index.html>

1. Acceso a la información: conectores

<para que sirven los conectores de base de datos, y para que los usaremos>

**Comparativa**

<explicacion de la rubrica de ventajas>

|  |  |  |
| --- | --- | --- |
|  | Ventajas | Desventajas |
| Cassandra | Permite balanceo de carga entre los nodos disponibles  Pool de threads para conexiones  Permite añadir una capa de seguridad | <a rellenar> |
| Mongodb | <a rellenar> | <a rellenar> |
| Neo4j | <a rellenar> | <a rellenar> |

Requerimientos comunes

- Python 3.5

- Librerias numpy, pandas y algunas librerías para representación gráfica: matplotlib

%config IPCompleter.greedy=**True**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

A continuación disponemos de una pequeña explicación de los conectores usados y las ventajas por las capacidades que ofrecen o por la naturaleza de la base de datos.

**Cassandra**

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

*cassandra.cluster*. Contiene una clase principal denominada *Cluster* que se conecta a un cluster de Cassandra estableciendo una conexión encapsulada en el objeto *Session*. Se pueden añadir ciertas configuraciones en acorde a la arquictura definida o a la forma de conexión:

**Instalación y configuración**

Configuración e instalación del conector / compatibilidad con Django

pip install cassandra-driver

pip install django-cassandra-engine

Macos

brew install libev

**Librerias usadas**

Hemos necesitado de las librerias panda, numpy

**Configuración y creación de la sesión**

Para permitir realizar conexiones y consultas de forma genérica hemos implementado un conjunto de funciones auxiliares.

* La función get\_session encapsula la conexión sobre nuestro "keyspace" generando el objeto Session característico del conector de cassandra.
* La función q es una abreviatura de query y permite acondicionar la realización de consultas y parametrizar valores sobre ellas. El resultado es un objeto tipo data frame, por lo cual estaría totalmente adaptado al entorno y sin necesidad de ser característico para el tipo de conector o base de datos.

session = get\_session("espaciodatos")

q(session, "select \* from incidents where incidentId = {id}", id = 10)

**def** get\_session(keyspace):

*"""Obtiene el conector con la sesión actual al keyspace indicado."""*

**return** (Cluster(['127.0.0.1']).connect(keyspace))

**def** q(session, query, \*\*kwargs):

*"""Función auxiliar para encapsular las queries producidas por cassandra en formato dataframe."""*

**return** pd.DataFrame([row **for** row **in** session.execute(query.format(\*\*kwargs))])

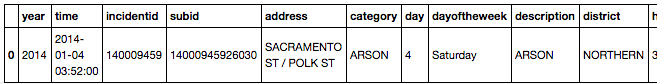
conn = get\_session("incidents")

**Caso de uso de sesión: ejemplo de consulta**

(El ejemplo de uso) De esta manera podemos obtener una sessión para la realización de las consultas:

**def** getIncidents(session, limit = 100):

**return** q(session, "select \* from incidents.overall limit **{limit}**",limit=limit)

****

**Mongodb**

El paquete *pymongo* contiene herramientas para trabajar desde Python con MongoDb. La estructura de los datos en documentos tipo JSON con un esquema dinámico llamado BSON, lo que implica que no existe un esquema predefinido. Los elementos de los datos se denominan documentos y se guardan en colecciones.

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

**Neo4j**

El conector \_neo4j-driver\_está basado en el driver oficial de Neo4J para Python. Esto ha implicado una transformación de las funciones por defecto para poder obtener resultados que sean útiles más tarde para nuestra aplicación.

Introducción

Instalación y configuracion

Librerias usadas

Configuración y creación de la sesión

Caso de uso de sesión: ejemplo de consulta

2. Representacion de la información

Vistas

Cassandra

Mongodb

Neo4j

Gráficos

Bibliografía

De todos los articulos que se consulten añadir aquí la referencia url y una breve descripción