JDBC River - Cargando datos a través de plugins

domingo, 16 de noviembre de 2014 12:46

https://github.com/jprante/elasticsearch-river-jdbc

¿Qué es un river?

Un river de ElasticSearch es código que se ejecuta en el entorno de memoria de ElasticSearch.
Se ejecutará únicamente en uno de los nodos del cluster. Si el nodo cae, el river se recoloca en cualquier otro nodo de cluster.
Los rivers, generalmente, están orientados a recopilar datos e indexarlos.

Tipos de rivers

Son muchos los rivers que se han desarrollado para ElasticSearch.
Cada river está generalmente especializado en conectarse a diferentes fuentes de datos para mover los datos a ElasticSearch.
El river es capaz de dialogar con la fuente de datos y transformar la información que hay en ella en documentos válidos para ElasticSearch tras una configuración previa.

Rivers desarrollados por el equipo de ElasticSearch

- CouchDB River,
- Wikipedia River,
- Twitter River,
- RabbitMQ River

Rivers desarrollados por la comunidad open source:

- JDBC River
- MongoDB River

Ejercicio - importación de datos desde MySQL

Para trabajar sobre los rivers de ElasticSearch, vamos a utilizar el river JDBC para importar datos desde un MySQL server. El proceso es muy sencillo:

1. Instalaremos el river en nuestra instancia de ES
2. Configuraremos el river mediante el RESTApi de ES
3. Ejecutaremos una importación de datos desde una tabla del servidor de MySQL

Enunciado del problema

Haciendo uso de la siguiente base de datos de ejemplo cargada en un servidor MySQL:

https://dev.mysql.com/doc/employee/en/employees-installation.html
https://launchpad.net/test-db/

```
$ cd ~/ciff
$ wget https://launchpad.net/test-db/employees-db-1/1.0.6/+download/employees_db-full-1.0.6.tar.bz2
$ tar -xjf employees_db-full-1.0.6.tar.bz2
$ cd employees_db/
$ mysql -t -u root < employees.sql
```

Configurar el river de JDBC en ElasticSearch para consumir la información de la tabla Employees con un refresco de 4 segundos.
Crear el mapping adecuado para el tipo de datos suministrado, sabiendo que:

1. El mapping de columnas debe seguir la siguiente aproximación:

emp_no	employeeId
birth_date	birthday
last_name	surname
gender	gender
hire_date	hiredOn
first_name	name

2. El usuario desea ejecutar búsquedas autocomplete sobre el campo apellido del empleado.
3. Únicamente deberán aparecer empleados cuyo salario sea superior a 60k

Solución

Creación del Índice y del Mapping adecuado

1. Aseguramos que el índice no existe en ES

   DELETE /employeesidx

1. Creamos en analizador en el índice donde vamos a crear el nuevo tipo

```
"settings": {
  "number_of_shards": 1,
```

```
    "analysis": {
      "filter": {
        "filtro_para_autocompletar": {
          "type": "edge_ngram",
          "min_gram": 1,
          "max_gram": 20
        }
      },
      "analyzer": {
        "autocompletar": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "filtro_para_autocompletar"
          ]
        }
      }
    }
  }
```

2. Sacamos el esquema de la tabla employees de la base de datos para tener referencia de los tipos:

```
CREATE TABLE `employees` (
  `emp_no` int(11) NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3. Haciendo uso de este esquema, generamos el mapping adecuado para nuestro índice

```
{ "mappings": {
        "employeesmapping": {
            "properties": {
                "employeeId": {"type":"long", "store":"yes","precision_step":"0" },
                "name": {"type":"string", "store":"yes", "index":"analyzed" },
                "surname": {"type":"string", "store":"yes", "analyzer":"autocompletar"},
                "gender": {"type":"string", "store":"yes", "index":"analyzed" },
                "birthday": {"type":"date", "store":"yes" },
                "hiredOn": {"type":"date", "store":"yes" }
            }
        }
    }
}
```

1. Y agregamos el nuevo índice en ES

```
PUT /employeesidx
{
    "settings": {
        "number_of_shards": 1,
        "number_of_replicas": 1
    },
    "mappings": {
        "employeesmapping": {
            "properties": {
                "employeeId": {"type":"long", "store":"yes" },
                "name": {"type":"string", "store":"yes", "index":"analyzed" },
                "surname": {"type":"string", "store":"yes","index":"analyzed" },
                "gender": {"type":"string", "store":"yes", "index":"analyzed" },
                "birthday": {"type":"date", "store":"yes" },
                "hiredOn": {"type":"date", "store":"yes" }
            }
        }
    }
}
```

Instalación del river de JDBC

1. Instalamos el plugin de JDBC en el servidor de ES mediante el siguiente comando:

```
$ ./bin/plugin --install jdbc --url
```

http://xbib.org/repository/org/xbib/elasticsearch/plugin/elasticsearch-river-jdbc/1.5.0.5/elasticsearch-river-jdbc-1.5.0.5-plugin.zip

2. Una vez instalado el plug-in, tendremos que descargar el driver para MySQL:

```
$ curl -o mysql-connector-java-5.1.33.zip -L 'http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.33.zip/from/http://cdn.mysql.com/'
```

3. Hacemos un unzip del fichero que nos hemos descargado

```
$ unzip mysql-connector-java-5.1.33.zip
```

4. Agregamos el driver que hemos descargado en el directorio de ElasticSearch

```
$ cp mysql-connector-java-5.1.33-bin.jar $ES_HOME/plugins/jdbc/
```

5. Y le damos permisos al fichero .jar para que pueda ser ejecutado (como mínimo un 644)

```
$ chmod 644 $ES_HOME/plugins/jdbc/*
```

6. Una vez tenemos el plug-in configurado, reiniciamos el servidor de ES mediante supervisor

```
$ sudo service elasticsearch restart
```

7. ES proporciona información sobre su estado en los ficheros de log. Si vamos a esos ficheros y echamos un vistazo a su contenido, deberíamos encontrar información sobre el arranque del nuevo plug-in como parte del sistema:

Configuración del river de JDBC

Approach 1 - Ejecutamos el JOIN directamente en la configuración del river - Posible impacto en performance

```
PUT /_river/my_jdbc_river/_meta
{
    "type" : "jdbc",
    "jdbc" : {
        "url" : "jdbc:mysql://localhost:3306/employees",
        "user" : "root",
        "password" : "Accenture01",
        "sql" : "SELECT E.emp_no AS employeeId,birth_date AS birthday,last_name AS
surname,gender AS gender,hire_date AS hiredOn, salary FROM employees E, employees.salaries S
WHERE E.emp_no = S.emp_no AND salary > 60000",
        "index":"employeesidx",
        "mapping":"employeesmapping"
    }
}
```

Para verificar que el river está dado de alta correctamente hacemos un
GET /_river/_search
----> Resultado

```
{
     "_index": "_river",
     "_type": "my_jdbc_river",
     "_id": "_meta",
     "_score": 1,
     "_source": {
          "type": "jdbc",
          "jdbc": {
               "url": "jdbc:mysql://localhost:3306/employees",
               "user": "root",
               "password": "",
               "sql": "SELECT E.emp_no AS employeeId,birth_date AS birthday,last_name AS
               surname,gender AS gender,hire_date AS hiredOn, salary FROM employees E,
               employees.salaries S WHERE E.emp_no = S.emp_no AND salary > 60000",
               "index": "employeesidx",
               "mapping": "employeesmapping"
          }
     }
},
{
     "_index": "_river",
     "_type": "my_jdbc_river",
     "_id": "_status",
     "_score": 1,
     "_source": {
          "node": {
               "id": "u47EzIXFQf-sUkoURR1WIg",
```

```
            "name": "Fafnir",
            "transport_address": "inet[/192.168.1.37:9300]"
        }
    }
}
```

Si queremos saber cuál es el estado de ejecución del river, hacemos un

```
GET /_river/jdbc/*/_state?pretty
---> Resultado
{
   "state": [
      {
         "name": "my_jdbc_river",
         "type": "jdbc",
         "started": "2015-05-16T20:22:02.669Z",
         "last_active_begin": "2015-05-16T20:22:02.715Z",
         "current_active_begin": "2015-05-16T20:22:02.715Z",
         "last_active_end": "2015-05-16T20:23:32.751Z",
         "map": {
            "lastStartDate": 1431807722719,
            "aborted": false,
            "lastExecutionEndDate": 1431807812751,
            "lastEndDate": 1431807812794,
            "counter": 1,
            "suspended": false,
            "lastExecutionStartDate": 1431807722730
         }
      }
   ]
}
```

Si todo ha ido correctamente, deberíamos tener contenido en el índice que hemos creado:

```
GET employeesidx/_count
---> Resultado
{
    "count": 307795,
    "_shards": {
        "total": 1,
        "successful": 1,
        "failed": 0
    }
}
```

Approach 2 - Creamos una vista y lanzamos la query sobre la vista

```
Mysql>
USE `employees`;
CREATE  OR REPLACE VIEW `employee_ES` AS
    SELECT E.emp_no AS employeeId,birth_date AS birthday,last_name AS surname,gender AS
    gender,hire_date AS hiredOn, salary FROM employees E, employees.salaries S WHERE E.emp_no =
    S.emp_no AND salary > 60000;

Elasticsearch>
# Borramos el contenido del indexador anterior JDBC que habíamos creado
DELETE /_river/my_jdbc_river

# Agregamos una nueva configuración para el river que hace uso de la vista
PUT /_river/my_jdbc_river/_meta
{
    "type" : "jdbc",
    "jdbc" : {
        "url" : "jdbc:mysql://localhost:3306/employees",
        "user" : "root",
        "password" : "Accenture01",
        "sql" : "SELECT * from employee_ES",
        "index":"employeesidx",
        "type":"employeesmapping"
    }
}
```

Approach 3 - Just in one hit

```
Elasticsearch>
    PUT /_river/my_jdbc_river/_meta
```

```
{
    "type": "jdbc",
    "jdbc": {
        "url": "jdbc:mysql://localhost:3306/employees",
        "user": "root",
        "password": "Accenture01",
        "sql": "SELECT * from employee_ES",
        "index": "employeesidx",
        "type": "employeesmapping",
        "index_settings": {
            "index": {
                "number_of_shards": 1,
                "number_of_replicas": 1
            },
            "type_mapping": {
                "mappings": {
                    "employeesmapping": {
                        "properties": {
                            "employeeId": {
                                "type": "long",
                                "store": "yes"
                            },
                            "name": {
                                "type": "string",
                                "store": "yes",
                                "index": "analyzed"
                            },
                            "surname": {
                                "type": "string",
                                "store": "yes"
                            },
                            "gender": {
                                "type": "string",
                                "store": "yes",
                                "index": "analyzed"
                            },
                            "birthday": {
                                "type": "date",
                                "store": "yes"
                            },
                            "hiredOn": {
                                "type": "date",
                                "store": "yes"
                            }
                        }
                    }
                }
            }
        }
    }
}
```

2. Abrimos de nuevo el log de ES para verificar que el river ha sido configurado correctamente y que muestra actividad. Debemos observar actividad en el log que se actualizará cada minuto.
3. El river habrá comenzado a indexar información de la base de datos y esta estará disponible en el siguiente índice por defecto:

```
GET employeesidx/_search
```

Posibles Errores

Mensaje:

```
[river.jdbc.SimpleRiverSource]
    while opening read connection: jdbc:mysql://localhost:3306/employees
    No suitable driver found for jdbc:mysql://localhost:3306/employees
```

Root cause:
No tenemos correctamente instalado el JDBC para MySQL en la máquina o el .jar no es visibile para ES.

Solución:
Bajamos el jar desde la página de MySQL y lo agregamos al folder lib de ES