



**UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS**

Facultad de Ingeniería

Departamento de Ingeniería en Sistemas



Asignatura: Base de datos II

Proyecto: Base de datos NoSQL

Catedrático: Ing. Rene Velasquez

Integrantes:

Kevin Edgardo Arambu Betancourth 20131006833

Noldin Doney Zambrano Zelaya 20121005631

Karen Melissa Nuñez Pastrana 20141011456

Elvin Moisés Sánchez Medina 20131008178

Lugar y Fecha: Tegucigalpa M.D.C. Lunes 29 de Abril 2019.

Índice

Portada	1
Índice	2
Introducción	3
Conexión con la base de datos Cassandra	6
Pantallas de aplicativo	9
Conclusiones	12

Introducción

En informática, NoSQL (a veces llamado "noSQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBD (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, su funcionalidad y el rendimiento a escala. Usan una variedad de modelos de datos, que incluyen documentos, gráficos, clave-valor, en-memoria y búsqueda.

Tipos de bases de datos NoSQL

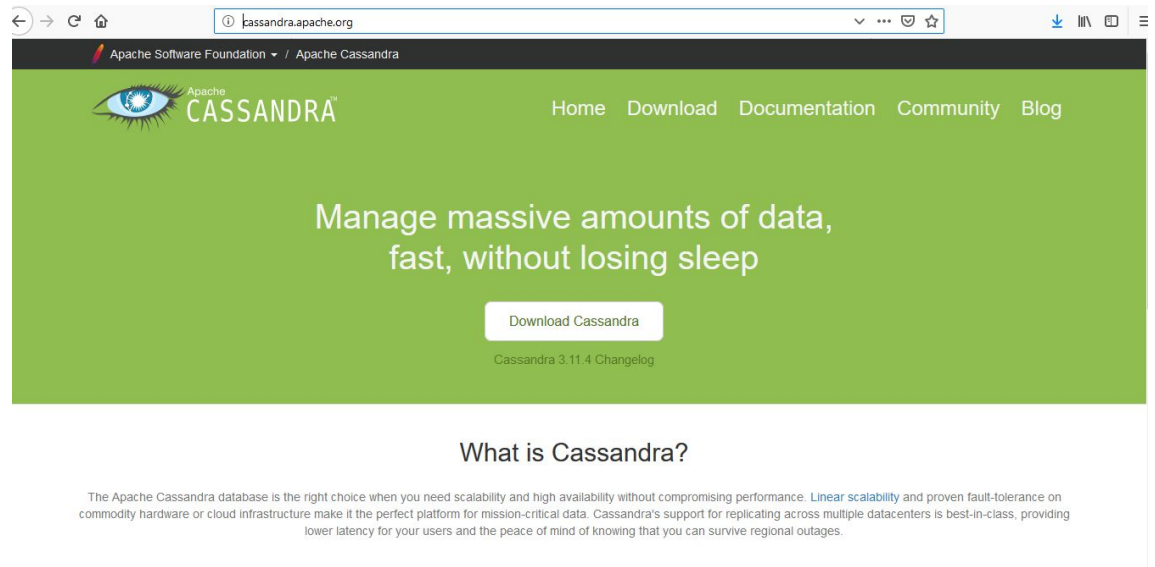
- Clave-valor
- Documentos
- Gráficos
- En memoria
- Buscar

En nuestro aplicativo hemos usado Apache Cassandra que es una base de datos NoSQL distribuida y basada en un modelo de almacenamiento de «clave-valor», de código abierto que está escrita en Java. Permite grandes volúmenes de datos en forma distribuida. Por ejemplo, lo usa Twitter para su plataforma. Su objetivo principal es la escalabilidad lineal y la disponibilidad. La arquitectura distribuida de Cassandra está basada en una serie de nodos iguales que se comunican con un protocolo P2P con lo que la redundancia es máxima. Está desarrollada por Apache Software Foundation.

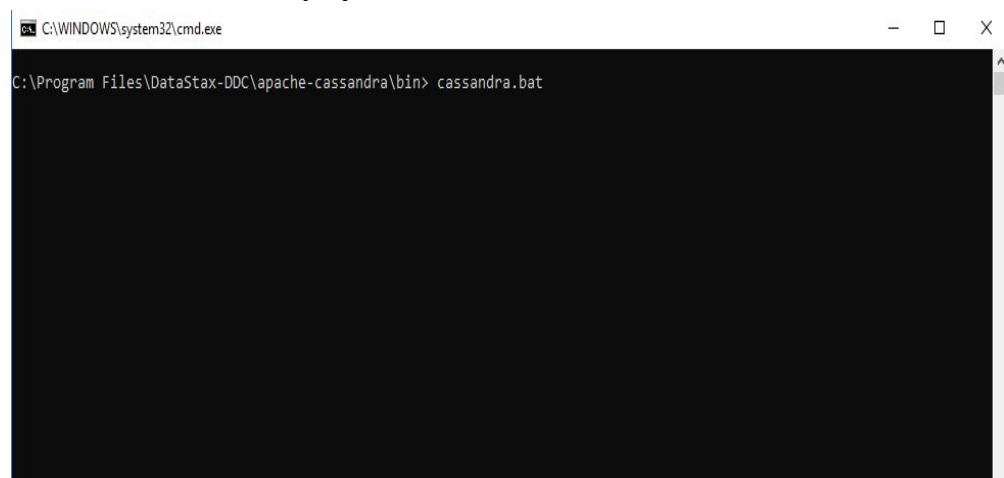
Requerimientos de Software

Instructivo de requerimientos

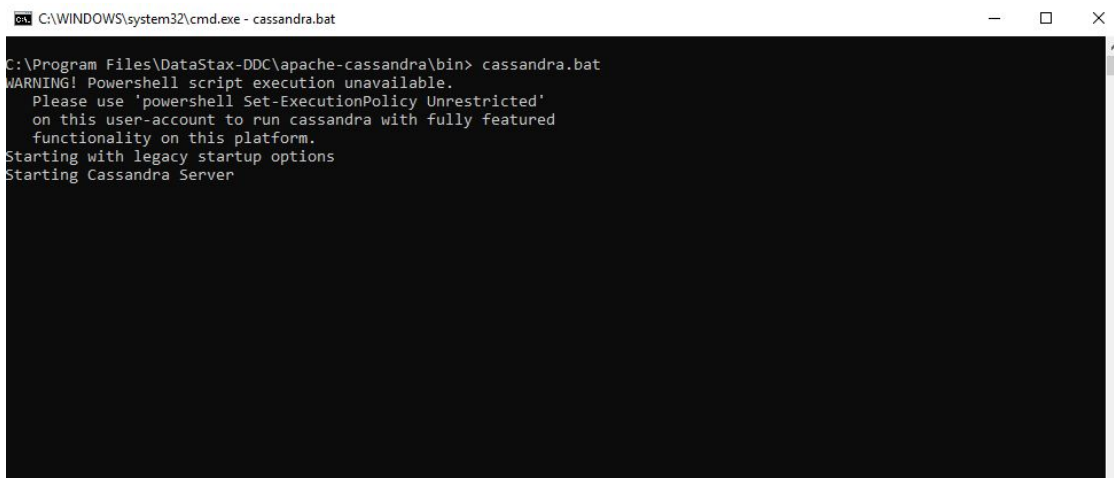
1. Descarga de cassandra, descargar en la pagina oficial
<http://cassandra.apache.org/>



2. Instalación de Cassandra, se omite pasos básicamente porque es dar siguiente a todo.
3. Arrancar el servidor de cassandra : Dirigirse a la ruta donde está instalado cassandra y ejecutar el comando cassandra.bat



Servidor de cassandra Iniciando



```
C:\WINDOWS\system32\cmd.exe - cassandra.bat

C:\Program Files\DataStax-DDC\apache-cassandra\bin> cassandra.bat
WARNING! Powershell script execution unavailable.
Please use 'powershell Set-ExecutionPolicy Unrestricted'
on this user-account to run cassandra with fully featured
functionality on this platform.
Starting with legacy startup options
Starting Cassandra Server
```

4. Descargar drivers para ejecutar cassandra desde NodeJS

npm install express

npm install cassandra-driver

npm install morgan

npm install debug

npm install jade

Conexión con la base de datos Cassandra

```
JS farmacia.js x
1
2 var express = require('express');
3 var router = express.Router();
4 var cassandra = require('cassandra-driver');
5
6 var client = new cassandra.Client({contactPoints: ['127.0.0.1'],localDataCenter:'datacenter1'});
7 client.connect(function(err, result){
8   console.log('farmacia: cassandra conectado');
9 });
10
```

Ejemplo de Ejecución de consultas

Insertar: Inserción de datos a la base de datos.

```
/* POST Agrega Farmacia */
router.post('/', function(req, res){
  idFarmacia = cassandra.types.uuid();

  var insertFarmacia = 'INSERT INTO farmacianosql.Farmacias(idfarmacia, nombrefarmacia, direccionfarmacia, propietario, farmaceuticoencargado) VALUES(?,?,?,?,?)';

  client.execute(insertFarmacia, [idFarmacia, req.body.nombreFarmacia, req.body.direccionFarmacia, req.body.Propietario, req.body.farmaceuticoEncargado], function(err, result){
    if(err){
      res.status(404).send({msg: err});
    } else {
      console.log('Farmacia Agregada');
      res.redirect('/');
    }
  });
});

module.exports = router;
```

Modificar:

```
var obtenerFarmaciaporID = 'SELECT * FROM farmacianosql.Farmacias WHERE idfarmacia = ?';

/* Obtener farmacia por id. */
router.get('/:idfarmacia', function(req, res) {
  client.execute(obtenerFarmaciaporID, [req.params.idfarmacia], function(err, result){
    if(err){
      res.status(404).send({msg: err});
    } else {
      res.render('editarFarmacia', {
        idfarmacia: result.rows[0].idfarmacia,
        nombreFarmacia: result.rows[0].nombreFarmacia,
        direccionFarmacia: result.rows[0].direccionFarmacia,
        Propietario: result.rows[0].Propietario,
        farmaceuticoEncargado: result.rows[0].farmaceuticoEncargado,
      })
    }
  });
});
```

```
/* POST Editar Farmacia */
router.post('/', function(req, res){
  var insertFarmacia = 'INSERT INTO farmacianosql.Farmacias(idFarmacia, nombreFarmacia, direccionFarmacia,
  Propietario,farmaceuticoEncargado) VALUES(?,?,?,?,?)';

  client.execute(insertFarmacia, [idfarmacia, req.body.nombrefarmacia, req.body.direccionfarmacia,
  req.body.propietario, req.body.farmaceuticoencargado],
  function(err, result){
    if(err){
      res.status(404).send({msg: err});
    } else {
      console.log('Farmacia Agregada');
      res.redirect('/');
    }
  });
});
```

Eliminar:

```
var borrarFarmacia = "DELETE FROM farmacianosql.Farmacias WHERE idFarmacia = ?";

router.delete('/:id', function(req, res){
  client.execute(borrarFarmacia,[req.params.id], function(err, result){
    if(err){
      res.status(404).send({msg: err});
    } else {
      res.json(result);
    }
  });
});

module.exports = router;
```

Listar:

```
var obtenerTodasFarmacias = 'SELECT * FROM farmacianosql.Farmacias';

router.get('/', function(req, res){
  client.execute(obtenerTodasFarmacias,[], function(err, result){
    if(err){
      res.status(404).send({msg: err});
    } else {
      res.render('index', {
        farmacia: result.rows
      })
    }
  });
});

module.exports = router;
```


Pantallas de aplicativo

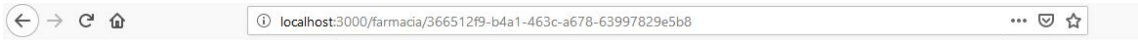
Listado de todas las farmacias



Opción de agregar farmacias



Opción de Editar Farmacia



Red de Farmacias NoSQL Cassandra Base de Datos II

Detalle de Farmacias

[Regresar](#)

- Nombre de Farmacia:
- Direccion:
- Propietario:
- Farmaceutico Encargado:
- ID: 366512f9-b4a1-463c-a678-63997829e5b8

[Editar](#) | [Eliminar](#)



Red de Farmacias NoSQL Cassandra Base de Datos II

Editar Farmacia

Opción de Eliminar Farmacia



Red de Farmacias NoSQL Cassandra Base de Datos II

Detalle de Farmacias

[Regresar](#)

- Nombre de Farmacia:
- Direccion:
- Propietario:
- Farmaceutico Encargado:
- ID: 366512f9-b4a1-463c-a678-63997829e5b8

[Editar](#) | [Eliminar](#)

Red de Farmacias NoSQL Cassandra Base de Datos II

Detalle de Farmacia

[Regresar](#)

- Nombre de Farmacia:
- Direccion:
- Propietario:
- Farmaceutico Encargado:
- ID: 366512f9-b4a1-463c-a678-63997829e5b8

[Editar](#) | [Eliminar](#)

Estas seguro?

OK

Cancel

Conclusiones

Podemos concluir que Cassandra es una solución para muchos casos. Lo ideal es tener claro desde el principio el caso de uso y el tipo de consultas que haremos para diseñar la base de datos coherentemente, de esta manera podremos manejar grandes volúmenes de datos y aprovecharnos de las ventajas de esta potente base de datos distribuida.

Podemos concluir que la curva de aprendizaje de Casandra es muy rápida debido que la sintaxis es muy parecida a las bases de datos relacionales, el lenguaje CQL nos proporciona funciones bastante parecidas a lo que estamos acostumbrados en SQL, haciendo más fácil acostumbrarse a utilizar las diversas funcionalidades de esta tecnología.