Ajuntament de Barcelona

Cibernàrium

Introducció a NoSQL

Capacitació Tecnològica per a Professionals i Empreses

Data

Barcelonactiva

Barcelona Activa: Qui som?

Barcelona Activa, integrada en l'àrea d'Economia, Empresa i Ocupació, és l'organització executora de les polítiques de promoció econòmica de l'Ajuntament de Barcelona.

Des de fa 25 anys impulsa el creixement econòmic de Barcelona i el seu àmbit d'influència donant suport a les empreses, la iniciativa emprenedora i l'ocupació, alhora que promociona la ciutat internacionalment i els seus sectors estratègics; en clau de proximitat al territori.

Barcelona Activa va ser guanyadora del Gran Premi del Jurat 2011, atorgat per la DG d'Empresa i Indústria de la Comissió Europea en el marc dels *European Enterprise Awards*, per la iniciativa empresarial més creativa i inspiradora d'Europa.

Àrees d'activitat de Barcelona Activa

Barcelona Activa s'estructura en tres grans blocs de serveis a les **Empreses**, a l'**Emprenedoria** i a la **Ocupació**. La **Formació** és un instrument transversal present en els tres blocs, així com també tot el relacionat amb l'economia social.

Una xarxa d'Equipaments Especialitzats

Xarxa de Proximitat

13 antenes Cibernàrium a biblioteques
10 punts d'atenció en Ocupació

Index

1. Introducció a NoSQL

1.1 Introducció

NoSQL (Not only SQL) és una nova nomenclatura que engloba moltes tecnologíes de diferent procedència, i fins i tot, amb diferents objectius, que comparteixen un denominador comú: oferir certes capacitats que manquen a SQL.

Per tant, podem dir que son repositòris de dades, que per la seva estructura, ens ofereixen possibilitats que l'esquema relacional no podia oferir.

1.1 Introducció

Històricament, podem situar el seu naixement en la Web 2.0, on les interaccions deixen de ser unidireccionals, i passem a tenir un ecosistema interactiu que augmenta la generació de dades exponencialment.

Aquest increment de dades fa que els antics sistemes, es degradin rapidament, fent necessàri per els grans actors, disposar de nous sistemes més flexibles.

1.1 Introducció

A l'inici d'aquestes necessitats, es va arribar a l'aproximació de millorar la maquinària

Rapidament s'en van adonar que la tecnología física tenia les seves limitacions

Per tant, només quedava generar sistemes ad-hoc per cada problemàtica particular.

Aquests sistemes van anar evolucionant en paral·lel, per cada problema, fins a resultar en solucions software robustes i confiables, orientades a diferents problemes concrets.

I així neix el moviment NoSQL

1.2 Motivacions

Podem resumir les causes de l'apareixement d'aquestes solucions en els següents punts

- Impossibilitat de seguir creixent vía maquinària (excés de costos) al nivell que requerien els volums de dades

- Necessitat d'evolucionar l'esquema de dades de forma iterativa

- Els anteriors punts van generar-se en entorns amb molts recursos (Google, Amazon, Facebook), pel qual es va facilitar la generació de noves propostes robustes amb una bon finançament.

1.2 Motivacions

Entre totes les motivacions que han portat a generar aquests sistemes, podem destacar-ne les més interessants:

- *Oferir transparència sobre l'escalabilitat horitzontal*
- *Facilitar el sharding*
- *Reduir costos en infraestructures concretes*
- *Flexibilització d'esquemes*
- *Focalitzar el problema i oferir una solució que s'hi adapti*

(perquè adaptar els problemes a les solucions existents?)

1.3 Característiques

Adicionalment, podem destacar les característiques comunes dels Data stores pertanyents a la categoría NoSQL:

- *Facilitar l'ús dels clústers de load balancers*
- *Persistència de dades*
- *Escalabilitat en memòria disponible*
- *Esquema dinàmic, permetent una migració d'esquema sense penalitzacions en temps*
- *Sistemes de consulta pròpis de cada motor (no estandaritzats)*

1.4 Tendència o necessitat?

És un moviment de tendència, o realment una necessitat?

Actualment, podem trobar desplegaments basats en aquestes tecnologíes en molts dels serveis que utilitzem a diari

- Amazon
- Yahoo
- Google
- Facebook
- Twitter

1.5 Utilitat

Però, és útil per qualsevol desenvolupament?

La idea sobre SQL és donar una solució ubiqua on, basant-se en unes regles, poden replicar-se la major part dels problemes.

NoSQL no es basa en aquesta idea

Cada problema requereix un estudi per veure quina eina ens pot servir millor

1.5 Utilitat

Algunes de les causes que ens poden portar a integrar una solució NoSQL son:

- L'escalabilitat de la solució relacional és inviable (normalment, a nivell de costos)
- Les dades disponibles son superiors a l'ordre del Terabyte
- L'esquema de base de dades no és homogèni
- O vist des d'un altre punt, el cost de l'homogenització ens porta a un esquema desproporcionat
- El model de negòci genera quantitats ingents de dades
- L'esquema consisteix simplement en definicions simples de columnes del tipus Large Object (CLOB o BLOB)
- Les consultes a realitzar son complexes i requereixen un model ad-hoc.

1.6 Tipologies

Revisem ara les tipologies de solucions NoSQL, segons la seva taxonomia:

- ***Key-value store****: son les solucions més senzilles, amb els esquemes més simples. Es tracta de sistemes on es desen combinacions de claus i valors. Realment, es pot pensar com a repositòris sense esquema.*
- ***Document store****: tenen com a noció principal el document. Aquest, com a tal, es conforma amb un esquema obert, i sol estar codificat en un llenguatge de processat establert, com JSON, XML o YAML. L'esquema, per tant, es basa en una col·lecció d'objectes que contenen contingut arbitrari, com ara altres objectes.*

1.6 Tipologies

- ***Graph database***: *es tracta de bases de dades especialitzades per conjunts de dades, els quals conformen grafs en les seves relacions. Alguns exemples conceptuals poden ser esquemes de relacions socials, enllaços d'un sistema de transport...*

- ***Wide column stores***: *son plataformes a mig camí entre un key-value store i una base de dades relacional. Mantenen el concepte de taula i fila, però la unitat real d'emmagatzemament és la columna. Això permet que una mateixa fila pugui tenir diferent nombre de columnes.*

- ***Altres***: *existeixen altres tipologíes, com RDF databases o XML databases, però el seu ús no és actualment difós, i no entren per tant en l'àmbit d'aquest curs.*

1.7 Propostes de mercat

Una breu revisió sobre les principals propostes del mercat

MongoDB: C++

Molt útil en el cas de necessitar consultes dinàmiques, com una BD relacional, però sense un esquema tancat.

Riak: Earlang i C

Bona escalabilitat, estructura tipus Dynamo, amb menys complexitat. Orientat principalment a la tolerància a errors

CouchDB: Erlang

Útil per dades acumulatives, com sistemes CRM o CMS.

Redis: C i C++

Pot utilitzar-se en entorns de canvi rapid de dades, incloent l'esquema. Pot ser usat com a reemplaçament dels entorns amb Memcache.

1.7 Propostes de mercat

Hbase: Java

En tots els entorns on es necessiti Hadoop, com entorns d'anàlisi de dades.

Cassandra: Java

Útil en entorns on es generen moltes més escriptures que lectures. Un bon exemple n'és la Banca.

Hypertable: C++

Una implementació basada en Hbase. Útil en tots els casos en els que es pot utilitzar Hbase

Accumulo: Java/C++

Una altre alternativa a Hbase

Neo4j: Java

En cas de necessitar un esquema ric o complex de dades, interconectat a l'estil d'un graf. Exemples en son topologíes de networking, pathfinding, rutes en relacions socials...

1.7 Propostes de mercat

ElasticSearch: Java

Útil quan es tenen objectes amb camps flexibles. Un exemple d'aplicació pot ser un servei de cites amb informació de geolocalització, gustos, peferències, diferències...

CouchBase: Erlang/C

Qualsevol aplicació on la baixa latència d'accés i l'alta concurrència al servei son una necessitat, com el joc online.

VoltDB: Java

Útil en casos d'actuació sobre quantitats massives de dades d'entrada, com els sistemes de control de fabricació.

1.8 Replicació

Redundancia vs Escalabilitat → Replicació

Les solucions NoSQL han sigut ideades en diferents entorns, destinades per diferents finalitats.

Tanmateix, trobem un punt comú a la majoría d'elles: La replicació.
En SQL, la replicació sol ser complicada de gestionar, sobretot a nivell de sharding.

1.8 Replicació

Moltes solucions NoSQL son gairebé transparents en la seva replicació: ofereixen horitzontailtat en

***Redundancia**: Els nous nodes es destinen a donar suport (normalment de lectura), oferint el mateix contingut de dades que altres nodes als que repliquen.*

***Escalabilitat**: L'escalabilitat horitzontal proporcionada, en la majoría de sistemes, és proporcionada de manera senzilla i transparent, només necessitant afegir noves màquines, alhora que s'afegeix la seva configuració al clúster.*

1.9 Big Table

Es tracta d'una idea propietària de Google

Es basa en un sistema

- *Amb compressió*
- *D'alt rendiment*
- *Construit sobre Google File System i altres technologies*

La idea és un mapping de dos índexs més un timestamp, definint un valor (que anomenarem byte array).

Així aconseguim indexació en 3 dimensions (files, columns i temporal)

Es la base de moltes implementacions de solucions NoSQL

1.10 Amazon Dynamo

Es tracta d'un sistema propietàri d'Amazon

La infraestructura es descomposa en els següents aspects:

- *Alta disponibilitat*
- *Estructura key-value*
- *Distribuit*

Estructurat en un multi-master (el client és l'encarregat de resoldre conflictes de versions)

S'ha derivat DynamoDB a Amazon d'aquesta idea

1.10 Amazon Dynamo

Les implementacions oficials son:

- *Apache Cassandra*
- *Voldemort*
- *Riak*

2. Cassandra

2.1 Introducció

Cassandra és una solució NoSQL, amb origen a Facebook

L'ha assumit la fundació Apache

La descripció simple es la d'una implementació de BigTable en una infraestructura del tipus d'Amazon Dynamo.

2.1 Introducció

Algunes de les característiques que comparteix amb aquestes idees son:

- *Modelat a l'estil de BigTable*
- *Eventual consistency (donat un període suficientment llarg sense canvis, tots els updates s'hauràn propagat)*
- *Gossip (protocol de comunicació màquina a màquina)*
- *Estructura master-master de Dynamo*

2.2 Taula comparativa

Engine	MongoDB	Cassandra	Redis	HBase
Imatge				
Desenvolupador	10gen	Apache Software Fundation	Salvatore Sanfilippo	Apache Software Fundation
Data de llançament	2009	2008	2009	2010
Usuaris destacats	Foursquare, Craigslist	Twitter, Digg	StackOverflow	Facebook
Tipología	Document	Columna	Key-value	Columna

2.2 Taula comparativa

Llicència	AGPL	Apache	BSD	Apache
Llenguatge	C++	Java	C/C++	Java
Exemple d'ús	CMS, sistemes de vots, desat de comentàris	Banca, logging	Stock prices, Analítics, dades en temps real	Missatgeria en temps real
Millor utilitzar en...	Consultes dinàmiques, escriptures freqüents, poques consultes estadístiques	Grans volums d'escriptures en front a les lectures	Canvis freqüents de dades, escriptura freqüent, poques consultes estadístiques	Lectures aleatòries en grans bases de dades

2.2 Taula comparativa

Punts clau	Manté algunes propietats d'SQL, com els índexs i les consultes	Alta disponibilitat: es tracta d'un encreuament entre les idees de BigTable i Dynamo	Molt ràpid	Modelat sota la direcció de BigTable. Dissenyat per desar ingents quantitats de dades.
Control de concurrència	Locks	Multiversion concurrency control (MVCC)	Locks	Locks
URL	mongodb.org	cassandra.apache.org	redis.io	hbase.apache.org
Procés de replicació	Asíncron	Asíncron	Asíncron	Asíncron

2.2 Taula comparativa

Sistemes operatius	Linux, Mac, Windows	Linux, Mac, Windows	Linux, Mac, Windows	Linux, Mac, Windows
Desat de dades	Disc	Disc	RAM	Hadoop
Característiques	Consistència, tolerància a particions, persistència	Alta disponibilitat, tolerància a particions, persistència	Consistència, tolerància a particions, persistència	Consistència, tolerància a particions, persistència

2.3 Característiques

Les principals característiques pròpies de la plataforma, es poden agrupar en:

- Descentralització dels nodes (no single point of failure)
- Replicació suportada
- Escalabilitat garantida per arquitectura
- Tolerant a fallades
- Suport per MapReduce integrat, a través de l'èina Hadoop
- Suport per Pig i Hive
- Query language pròpi (molt similar a SQL) anomenat CQL
- Suport per expiració de dades (columnes)
- No suporta operadors d'agregació en el node (SUM, MIN...). Les ha d'implementar el client.
- La estructura és un sistema key-value store, on les claus poden mapejar diferents valors.

2.3 Característiques

En una nomenclatura més tècnica:

- *Consistent hashing*
- *Columnar*
- *SSTable storage*
- *Gossip protocol*
- *Append-only*
- *Hinted handoff*
- *Memtable*
- *Read repair*
- *Compaction*

2.4 Quins projectes l'utilitzen?

Utilitzat per molts projectes importants:

- *Reddit*
- *Digg*
- *Facebook (ara migrat a HBase)*
- *Twitter*
- *Netflix*
- *CERN ATLAS*

2.5 Eines

Conté diverse seines per el seu ús

- ***Cassandra-cli****: un command line interface, molt útil per interactuar directament amb el motor.*
- ***Drivers*** *per diferents llenguatges / plataformes (entre ells, java, python , .Net, ruby...)*

2.6 Conceptes pròpis de la plataforma i ús

Podem identificar conceptes pròpis de la plataforma (basics):

- ***Column***: *Partició minima de dades. Conté una tripleta de nom, valor i timestamp.*
- ***Row***: *s'identifica per una row key, determinant quin node el conté.*
- ***Column Family***: *És l'homònim a una taula en el model relacional. De fet, a partir de CQL 3 es pasa a anomenar-les tables.*
- ***Table***: *colleció de columns ordenades.*
- ***Keyspace***: *Agrupació de column families comunes. Es pot comparer a un esquema de dades d'una BD relacional.*

2.6 Conceptes pròpis de la plataforma i ús

La clau del disseny sobre Cassandra es basa en que l'arquitectura ha de ser orientada a les consultes que es volen executar, i no a modelar entitats i relacions

Concepts clau:

- ***Cluster***: *grup de nodes per emmagatzemar dades. Un cluster pot constar d'un sol node*
- ***Replicació***: *és el process de còpia de dades en multiples nodes. La finalitat és assegurar la tolerància a errors i la confiabilitat.*
- ***Particionament***: *el particionador distribueix les dades entre els nodes. L'objectiu és aconseguir la funcionalitat de load balancing.*
- ***Data center***: *grup de nodes configurats en comú, per motius de replicació.*

2.6 Conceptes pròpis de la plataforma i ús

Idees d'ús sobre la plataforma:

- És una infraestructura bona si tenim més escriptures que lectures (per exemple, en el cas de logging)

- Si tots els components de software son Java (Cassandra està implementat en Java, així que la integració és directe)

- Per exemple, banca o indústria financera

2.7 Beneficis

Beneficis esperats:

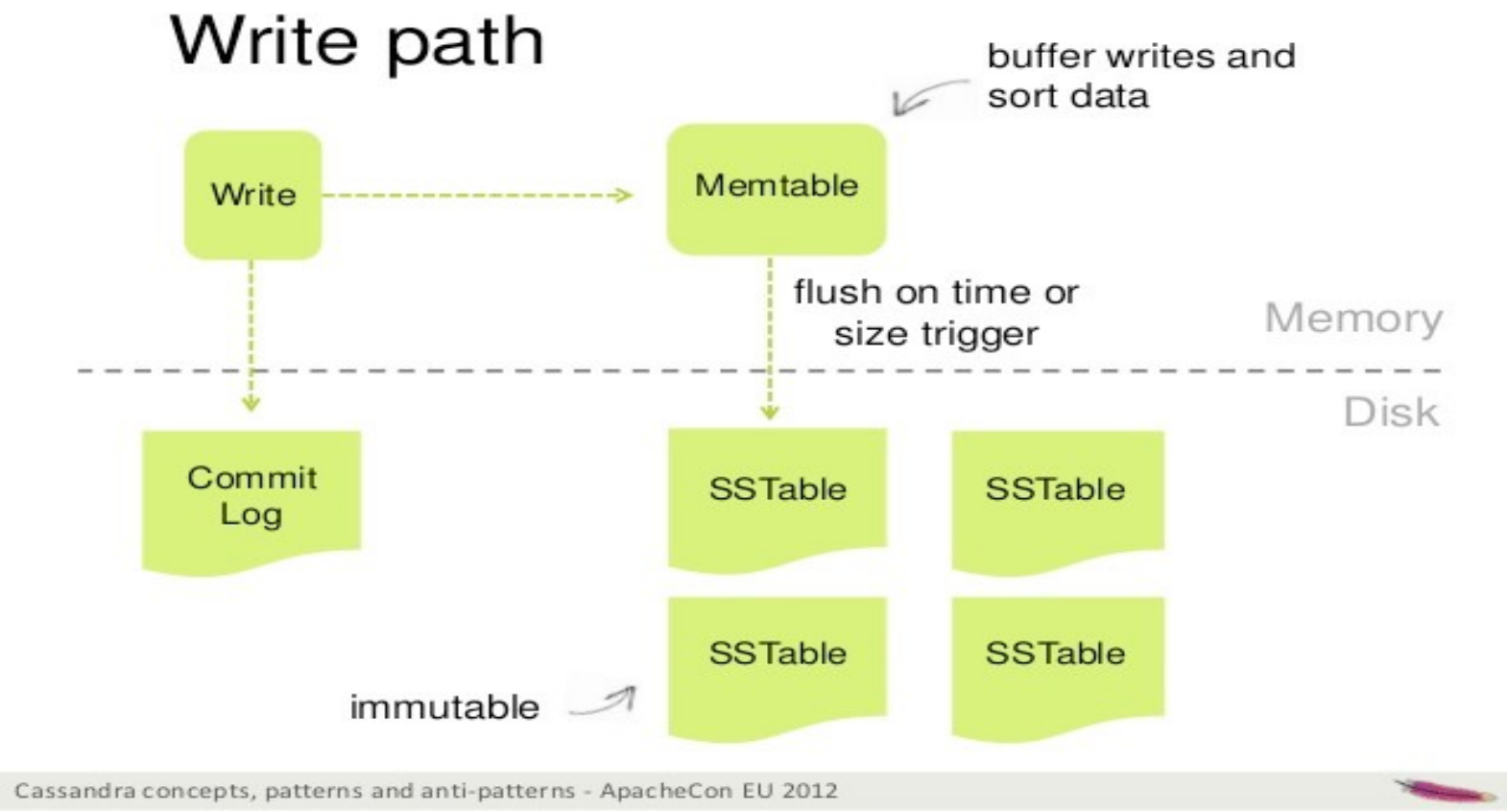
- *Escalabilitat horitzontal*
- *Alta disponibilitat (fault tolerance)*
- *Baixa latencia (sobretot en escriptures)*
- *Cluster homogeni: no hi ha un node “master”*
- *Model de dades ric (no és un simple key-value)*

2.8 Com funciona?

Com funciona? El workflow bàsic

- Podem tenir fins a 2 billions de columns
- Les columns son triplets de nom, valor i timestamp
- Agrupem un conjunct de columns sota una row key per indexar-les. Aquest conjunct l'anomenarem row.
- Al conjunct de rows, l'anomenarem Column Family

2.8 Com funciona?



2.9 CQL i tipus de dades

CQL: Query language especific de Cassandra. Totes les instruccions acaben amb un punt i coma.

A continuació, donem una referència ràpida. En ella trobareu una relació de les principals opcions, a més de les opcions requerides i un exemple d'ús. Podeu trobar la resta d'informació a:

http://cassandra.apache.org/doc/cql3/CQL.html

2.9 CQL i tipus de dades

Referència

Definició

CREATE KEYSPACE: crea un nou keyspace. A més, es defineix la estratègia de replicació. Els caràcters vàlids son alfanumèrics, i està limitat a 32 caràcters.

Opcions

Replication: la estratègia de replicació i les opcions del Keyspace.

Exemple

CREATE KEYSPACE MyKeyspace WITH replication {'class': 'SimpleStrategy', 'replication_factor': 3};

2.9 CQL i tipus de dades

USE: utilitza un keyspace specific

Exemple

```
USE MyKeyspace;
```

ALTER KEYSPACE: modifica un keyspace existent.

Opcions

les mateixes que en CREATE KEYSPACE

Exemple:

```
ALTER KEYSPACE MyKeyspace WITH replication {‘class’:
‘SimpleStrategy’, ‘replication_factor’: 4};
```

DROP KEYSPACE: elimina un keyspace.

Exemple

```
DROP KEYSPACE MyKeyspace;
```

2.9 CQL i tipus de dades

CREATE TABLE: crea una nova taula o column family (legacy). Es defineixen totes les propietats

Opcions

PRIMARY KEY: pot ser d'una sola columna, o multicolumna. Si és multicolumna, s'ha d'especificar al final de la definició de la taula (una taula no pot tenir només columnes pertanyents a la PK: com a minim hi ha d'haver una columna que no hi pertanyi)

Exemple:

```
CREATE TABLE myTable{
Attr1 text PRIMARY KEY,
Attr2 varint ,
Attr3 uuid
) WITH compaction = { 'class': 'LeveledCompactionStrategy' };
```

2.9 CQL i tipus de dades

ALTER TABLE: cambia l'esquema de columns de la taula. Aquesta és una operació de temps constant, degut a que les columnes buides no ocupen lloc al disc (es defineix un esquema que només s'evalua en el cas que hi hagi dades)

Opcions

afecten les mateixes opcions que a CREATE TABLE

Exemple:

```
ALTER TABLE myTable
    ALTER myattr1 TYPE uuid;
```

DROP TABLE: elimina una taula del keyspace

Exemple:

```
DROP TABLE myTable;
```

2.9 CQL i tipus de dades

TRUNCATE: elimina totes les dades d'una taula

Exemple

```
TRUNCATE myTable;
```

CREATE INDEX: crea un nou index per una columna determinada.

Exemple:

```
CREATE INDEX myIndex on myTable (myAttr);
```

DROP INDEX: elimina un index secundàri

Exemple

```
DROP INDEX myIndex;
```

2.9 CQL i tipus de dades

Manipulació

INSERT: genera nous elements, o actualitza un d'existent. La operació és atòmica. Té com a opcions TTL (segons) i TIMESTAMP(microsegons), parametritzats per defecte.

Exemple

INSERT INTO myTable (attr1, attr2) VALUES ("str1", 1) USING TTL 600;

UPDATE: actualitza una o més columnes amb noves dades. La operació és atòmica.

Exemple

UPDATE myTable USING TTL 230 SET attr1 = "val1", attr2 = 78 WHERE attr3 = "val3"

2.9 CQL i tipus de dades

DELETE: elimina files i columns. Si les columns son proporcionades després de DELETE, només s'eliminaran aquestes columns de les files seleccionades. Sino, tota la fila sera eliminada. La operació és atòmica.

Exemple

```
DELETE FROM myTable WHERE attr1 IN (123, 456);
```

BATCH: genera una única operació atòmica a partir d'un conjunct d'operacions. Les operacions suportades son INSERT, UPDATE i DELETE. Es pot utilitzar TIMESTAMP, però només a nivell de BATCH, no de les operacions interiors.

Exemple:

```
BEGIN BATCH
INSERT INTO myTable1 (attr1, attr2) VALUES ('v1', 'v2');
UPDATE myTable2 SET attr2 = 45 WHERE attr1 = "u2";
DELETE attr5 FROM myTable3 WHERE attr2 = 12;
UPDATE myTable4 SET attr4 = 89;
APPLY BATCH;
```

2.9 CQL i tipus de dades

Consultes

SELECT: genera una consulta sobre una taula concreta mitjançant uns filtres determinats per l'usuari. Les consultes s'han d'aplicar sobre la clau primaria (sigui simple o composta), o sobre una columna que estigui indexada.

Exemple

SELECT attr1, attr2 FROM myTable WHERE attr1 IN (1, 2, 3) AND attr2 > 34;

2.9 CQL i tipus de dades

Finalment, cal parlar dels Data Types disponibles:

- Ascii: string ascii
- Bigint: enter de 64 bits
- Blob: conjunct de bytes arbitraris
- Boolean: lògic
- Counter: comptador
- Decimal: punt flotant de precisió variable
- Double: punt flotant de 64 bits
- Float: punt flotant de 32 bits
- Inet: Adreça IP
- Int: enter de 32 bits
- Text: string codificada en UTF-8
- Timestamp: marca de temps

2.9 CQL i tipus de dades

Timeuuid: marca de temps id
Uuid: enter id
Varchar: string codificada en UTF-8
Varint: enter de precisió variable
Col·leccions: tipus de data compostos
Map: col·lecció de parelles clau-valor. Similar a un diccionari (o hash)
Set: col·lecció de valors únics
List: col·lecció de valors

Cal destacar que tenim disponibles diverses funcions que ens ajuden en l'ús de CQL. Com a exemple, tenim now().

2.10 Instal·lació

// Dependencies
sudo aptitude install openjdk-7-jdk openjdk-7-jre
// Install
wget http://ftp.cixug.es/apache/cassandra/1.2.5/apache-cassandra-1.2.5-bin.tar.gz
tar xzvf apache-cassandra-1.2.5-bin.tar.gz
sudo bin/cassandra -f

// Config
vim conf/
log:
- file: log4j-server.properties
– property: log4j.appender.R.File
- file: cassandra-env.sh
– property:
--- MAX_HEAP_SIZE
--- HEAP_NEWSIZE

// CLI
./bin/clqsh

2.11 Driver

Tenim diverses opcions per utilitzar aplicacions en conjunt amb Cassandra. Per homogeneitat, en aquest curs comptarem amb drivers (o interfícies) per Python.

pycassa
http://pycassa.github.io/pycassa/tutorial.html

* Podeu trobar implementacions de drivers per Cassandra per molts llenguatges actuals

2.12 Exemple

./cqlsh

```
DESCRIBE KEYSPACES;
CREATE KEYSPACE myKeySpace WITH REPLICATION = {'class':
'SimpleStrategy', 'replication_factor': 3};
USE myKeySpace;
CREATE TABLE people (
person_id int PRIMARY KEY,
first_name text,
last_name text
);
DESCRIBE TABLES;
INSERT INTO people (person_id, first_name, last_name) VALUES  (12, 'Juan',
'Par');
INSERT INTO people (person_id, first_name) VALUES (13, 'Pepe');
CREATE INDEX ON people (first_name);
SELECT * FROM people WHERE first_name = “Pepe”;
DROP TABLE people;
```

3. Exercicis sobre Cassandra

3.1 Implementació via shell
3.2 Implementació per aplicació
3.3 Benchmarking

3.1 Implementació via shell

En aquesta primera part pràctica, generarem una serie de projectes a partir de la consola del sistema que estem tractant.

- ***Twitter***: *generarem un clon del software conegut com Twitter. Com a definició, prendrem l'esquema d'usuari, tweet i seguidor.*
- ***Registre d'hotel***: *necessitem un sistema d'informació per el registre d'un hotel. S'ha de contindre informació sobre totes les habitacions, així com un historial tant d'usuaris com d'ús.*
- ***Log enriquit***: *volem tenir una peça de software que mantingui informació de log, concretament el còdi, una marca de temps, i un objecte, que contindrà informació arbitrària (fins i tot, altres objectes)*

3.1 Implementació via shell

- ***Item vault***: *un banc d'informació sobre objectes varis dins diverses col·leccions. L'objectiu és donar la versatilitat a un negòci d'objectes extranys per poder tenir un sistema d'informació amb totes les característiques d'aquests.*

- ***Enfermatim***: *base de dades d'informació sobre malalties. Aquest ha de contenir informació sobre geografía, causes, públic afectat, i estadístiques.*

- ***Workouts***: *tenim una gran cadena de gyms, i volem mantenir una base de coneixement sobre les màquines que tenim, així com dels clients i el seu creixement personal (muscular, dietes, salut, dies d'entrenament...)*

- ***Job box***: *es tracta d'un directòri d'ofertes de treball sobre tot el món. Volem informació sobre el treball, així com localització, empresa, perfils necessitats, i tota la informació de valor sota aquest concepte.*

3.2 Implementació per aplicació

En la segona part, generarem els mateixos projectes anteriors, a partir del llenguatge de programació que nosaltres escollim i els seus bindings sobre el sistema que tractem.

3.3 Benchmarking

Finalment, en aquesta tercera part, generarem scripts que omplirán les nostres bases de dades, i farem benchmarking sobre els processos d'escriptura i de lectura.

Barcelonactiva

Ajuntament de Barcelona

bcn.cat/barcelonactiva
bcn.cat/cibernarium