ceu-economics-and-business.github.io

NoSQL. – ECBS 5146 SQL and Different Shapes of Data

8–11 minutes

Overview

Teaching: 180 min

Questions

- What is polyglot persistence and how solves the data storge problems of the present days?
- What are the common treats of NoSQL solutions?
- How can a data analyst choose the appropriate data solution tailored to her/his task?

Objectives

- Understanding the polyglot persistence
- Understanding eventual consistency
- Understanding SQL VS NoSQL
- Understanding NoSQL families
- Introducing Key-Value Stores
- Introducing Apache Zeppelin

- Practice with Redis
- Introducing Time Series DBMS
- Introducing Document Stores
- Practice of MongoDB with Airbnb data
- Introducing Column Stores
- Introducing Search Engines
- Practice of Solr with Kaggle Flight data
- Introducing Graph DBMS
- Practice of Neo4J with Paradise Papers

Keywords

#BASE

#NOSQL FAMILIES

#APACHE ZEPPELIN

#REDIS

#MONGO

#SOLR

#NETWORK SCIENCE

#NEO4J

#PARADISE PAPERS

Table of Content

> Redis
>
> MongoDB
>
> Solr
>
> Neo4j

Redis

Links to help you

https://redis-py.readthedocs.io/en/stable/index.html

https://redis.io/commands#

Exercise interface: http://34.243.146.179 **(Jupyter Notebook)**

Username/Password will be distributed during class. Ask on Teams if you haven't got one.

Connect to Redis with Python

```
import redis
r = redis.Redis(host='something', port=something)
```

Set a value with a key

Get value by key

Overwrite with expiration

Is it there?

Set a number

```
r.set("something", 10)
r.get('something')
```

Increase number value

```
r.incr("something")
r.get('something')
```

Store multiple key-value

```
r.mset({'one': 1, 'two': 2, 'three': 3})
```

Display all keys stored in DB

Retrieve multiple values by key

REDIS Exercise

USING THE DOCUMENTATION, FIND HOW TO DELETE A VALUE BY KEY AND HOW TO CHECK THE EXISTENCE OF A KEY.

MongoDB

Links to help you

https://docs.mongodb.com/manual/

https://www.w3schools.com/python/python_mongodb_getstarted.asp

AirBnb dataset

Connect to MongoDB with Python

```
import pymongo
import pprint
mongo = pymongo.MongoClient("mongodb://
something:something")
```

Select a database and a collection

If database or collection does not exist, will be created with the first data write (eg. insert line)

```
db = mongo["mydatabase"]
customers = db["customers"]
```

List collections stored in the database

```
db.list_collection_names()
```

Insert a document

```
id = customers.insert_one({ "name": "John",
"address": "Boston Highway 37" }).inserted_id
```

Find

Find the customer inserted by id. Pretty print the result.

```
pprint.pprint(customers.find_one({"_id": id}))
```

Find multiple customers by “name” field. Inverse sort by “address”.

Limit to 5. In order to print the result we iterating over the result set and pretty print each resulting JSON.

```
for customer in customers.find({"name":
"John"}).sort("address",-1).limit(5):
    pprint.pprint(customer)
```

Count

```
customers.count_documents({"name": "John"})
```

Distinct

Find the customers called “John” which address starts with “Boston” and print out distinct addresses.

```
for customer in
customers.find({"name":"John","address": {"$regex":
"^Boston"}}).distinct("address"):
    pprint.pprint(customer)
```

Airbnb Sample database

```
airbnb = db["airbnb"]
pprint.pprint(airbnb.find_one())
```

Advance filtering

Filter by a deeper JSON field. Print only part of JSON.

```
for listing in airbnb.find({ "address.country":
"Spain" }).limit(10):
    pprint.pprint(listing['address']
```

```
['government_area'])
```

MONGO Exercise

COUNT HOW MANY AIRBNB LISTINGS WE HAVE IN THE SAMPLE DATABASE HAVING “COUNTRY_CODE” “US” OR “ADDRESS.MARKET” STARTWITH “M” (USE MONGODB DOCUMENTATION)

Solution

Solr

Links to help you

https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser

http://yonik.com/solr/query-syntax/

Exercise interface: http://34.243.146.179:8081/solr/#/flightdelays/query

A simple query

SOLR has different connectors to programming languages. For simple query testing, we don’t need to program because SOLR is offering so called HTTP Rest interface. These are basically url calls from a browser.

The simplest query (the result is limited by default to 10):

```
http://34.243.146.179:8081/solr/flightdelays/select?
```

```
q=*:*
```

In SQL, this would be something like:

```
SELECT * FROM flightdelays LIMIT 10;
```

Ranges

List records from the last 10 years where tail number is N520JB:

```
http://34.243.146.179:8081/solr/flightdelays/select?
fl=DISTANCE,ORIG_CITY,DEST_CITY&q=TAIL_NUMBER:N838UA
AND DATE:[NOW-10YEARS TO *]&sort=DISTANCE desc&rows=5
```

In SQL, this would be something like:

```
SELECT distance,orig_city,dest_city FROM flightdelays
  WHERE tail_number='N520JB' AND date >=
DATE_SUB(NOW(),INTERVAL 10 YEAR)
  ORDER BY distance DESC
  LIMIT 5;
```

String search / Fuzzy search

List records where tail numbers starting with any character, followed by “2”, followed by 2 any character, followed by “jb”. Display only tail number in the result set:

```
http://34.243.146.179:8081/solr/flightdelays/select?
fl=TAIL_NUMBER&q=TAIL_NUMBER:?8??a
```

Fuzzy searches is based on the Damerau-Levenshtein Distance or Edit Distance algorithm. Fuzzy searches discover terms that are

similar to a specified term without necessarily being an exact match. To perform a fuzzy search, use the tilde ~ symbol at the end of a single-word term

In the next example we list records with destination city close to “columbas” by distance of 2. The distance referred to here is the number of term movements needed to match the specified phrase.

```
http://34.243.146.179:8081/solr/flightdelays/select?
fl=DEST_CITY&q=DEST_CITY:kolumbas~2
```

Facets

Same as before, but this time return back distinct destination cities as well:

```
http://34.243.146.179:8081/solr/flightdelays/select?
q=DEST_CITY:Boise~3&facet.field=DEST_CITY_str&facet=on
```

This previous result sounds like a combined result of the following SQLs:

```
SELECT * FROM flightdelays WHERE DEST_CITY LIKE
'columbas%' LIMIT 10;

SELECT dest_city, COUNT(*) FROM flightdelays
  WHERE DEST_CITY LIKE 'columbas%'
  GROUP BY dest_city;
```

Geo spacial search

Return back records within a circle defined by center point of

39.85,-104.66 [lat,lon] and diameter of 2 kilometer. Display only ORIG_CITY and ORIG_LOCATION_p in the result set and facests for ORIG_CITY_str.

```
http://34.243.146.179:8081/solr/flightdelays/select?
d=2&facet.field=ORIG_CITY_str&facet=on&fl=ORIG_CITY,OR
geofilt}
&pt=39.85,-104.66&q=*:*&sfield=ORIG_LOCATION_p
```

SOLR Exercise

HOW MANY FLIGHTS ARRIVED IN SAN FRANCISCO WITH NO DELAY ALTHOUGH THEY DEPARTED AT LEAST 50 MINS BEHIND THE SCHEDULE?

Neo4j

Links to help you

(https://neo4j.com/developer/cypher-query-language/)

(http://neo4j.com/docs/developer-manual/current/cypher/)

(https://cloudfront-files-1.publicintegrity.org/offshoreleaks/neo4j/guide/index.html)

Exercise interface: https://sandbox.neo4j.com/

Register and start a sandbox with “Paradise Papers by ICIJ”

Simple queries

In Neo4J the SELECT is called MATCH. One of the simplest query

is selecting 25 Officer nodes :

```
MATCH (n:Entity)
RETURN n.name LIMIT 25
```

In SQL, this would be something like:

```
SELECT n.name FROM Entity AS n LIMIT 25;
```

We can use WHERE clause to filter our result:

```
MATCH (o:Officer)
WHERE o.countries CONTAINS 'Hungary'
RETURN o
```

In SQL, this would be something like:

```
SELECT o.countries FROM officer AS o WHERE
o.countries LIKE '%Hungary%';
```

NEO4J Exercise 1

RETURN THE FIRST 10 ADDRESS NODES

NEO4J Exercise 2

HOW MANY PROPERTIES AN ADDRESS NODE HAS?

NEO4J Exercise 3

RETURN THE FIRST 10 COUNTRIES OF THE ADDRESS NODE. WHAT IS THE LAST COUNTRY IN THE LIST?

NEO4J Exercise4

HOW MANY ADDRESS NODES HAS ‘Mexico’ AND ‘Monaco’ IN THEIR ADDRESS PROPERTY?

Joins

Find the Officers and the Entities linked to them (double MATCH)

```
MATCH (o:Officer)
MATCH p=(o)-[r]-(c:Entity)
RETURN p
LIMIT 50
```

In SQL, this would be something like:

```
SELECT *
FROM officer as o
INNER JOIN Entity as c
USING (relationship)
```

Find the nodes associated “the duchy of lancaster”:

```
MATCH (o:Officer)
WHERE toLower(o.name) CONTAINS "the duchy of
lancaster"
MATCH p=(o)-[r]-(c)
RETURN p
```

Same, but this time return the nodes 2 hops away:

```
MATCH (o:Officer)
WHERE toLower(o.name) CONTAINS "the duchy of
```

```
lancaster"
MATCH p=(o)-[*..2]-(c)
RETURN p
```

Count

Which country has the most addresses

```
MATCH (n:Address)
RETURN n.countries, count(*)
ORDER BY count(*) DESC
LIMIT 10
```

NEO4J Exercise 5

LIST THE NAME AND NUMBER CONNECTIONS OF THE TOP 10 MOST CONNECTED OFFICERS FROM BULGARIA.WHO IS THE NO1?