Referat Integrarea Sistemelor Informatice

Baze de date NoSQL – avantaje si proiectare

Student: Andrei Bumbu

Master Anul 2

1. Introducere

In present utilizarea unei baze de date in organizatii este un lucru **obligatoriu**. Persistarea datelor si **analizarea** acestora pentru a trage concluzii referitoare la activitatea companiei si de a **previziona** performata pe viitor sunt necesare pentru *supravietuira pe termen lung*.

In prima parte se vor aduce in discutie **bazele de date traditionale**, tehnologie care a dus la dezvoltarea masiva a organizatiilor si de a extinde posibilitatile realizaii de aplicatii. Acestea sunt inca folosite in majoritatea aplicatiilor ce se afla in productie, dar incep sa se loveasca de anumite **limitari** deoarece nivelul de date utilizate s-a marit exponential.

Pentru a oferi un context aproximativ, 'The big four', cei 4 giganti *(Google, Amazon, Microsoft si Facebook)* stocheaza aproximativ **1200 de petabytes**, tradusi in 1.2 milioane de terabytes.

In aceasta situatie, al cresterii exponentiale de date, SQL-ul nu reuseste sa faca fata deoarece acesta are nevoie de realizarea normalizarilor ce rezulta in ierarhii complexe, devenind foarte incet. In aceasta situatie in care pentru o foarte lunga perioada **MySQL domina piata**, au inceptu sa apara solutii ce rezolva aceasta complexitate si ofera posibilitatea de a stoca date intr-un mod mult mai simplist, iar aceasta solutie sunt bazele de date **NoSQL**.

Cateva tipuri de baze de date sunt:

a) Baze de date relationale (RDBMS)

Bazele de date relationale reprezinta un system de management al datelor bazat pe modelul relational de date. Asa cum am specificat, majoritatea bazelor de date in present folosesc acest model.

Exemple: Oracle, MySQL, PostoreSQL

b) Baze de date de procesare si analizare online (OLAP)

OLAP *(On-Line Analytical Processing)* este o tehnologie utilizata pentru **organizarea** bazelor de date foarte mari ale firmelor si de a oferi suport decizional pentru afaceri. Bazele de date OLAP sunt impartite in mai multe **cuburi**, iar fiecare cub este proiectat de un administrator de cub pentru a se potrivi cu modul in care **regasiti** si **analizați** date.

Exemple: Vertica, Hadoop

c) Baze de date non-relationale (NoSQL)

Bazele de date NoSQL ofera un mechanism de **stocare** si **interogare** de date care este modelat altfel decat abordarea clasica, utilizata de bazele de date relationale. Aceasta au aparut de abia in **secolul 21**, sustinute de companiile Web 2.0. Aceste baze de date se utilizeaza foarte mult in **'big data'** si in aplicatii **'real-time'**, in timp real. De asemenea acestea sunt denumite si *'Not Only SQL'* pentru a sugera faptul ca este suportat in continuare si **SQL-ul,** ca si limbaj de interogare.

Exemple: MongoDB, HBase, InfiniteGraph, Apache Ignite, ArangoDB, Scylla

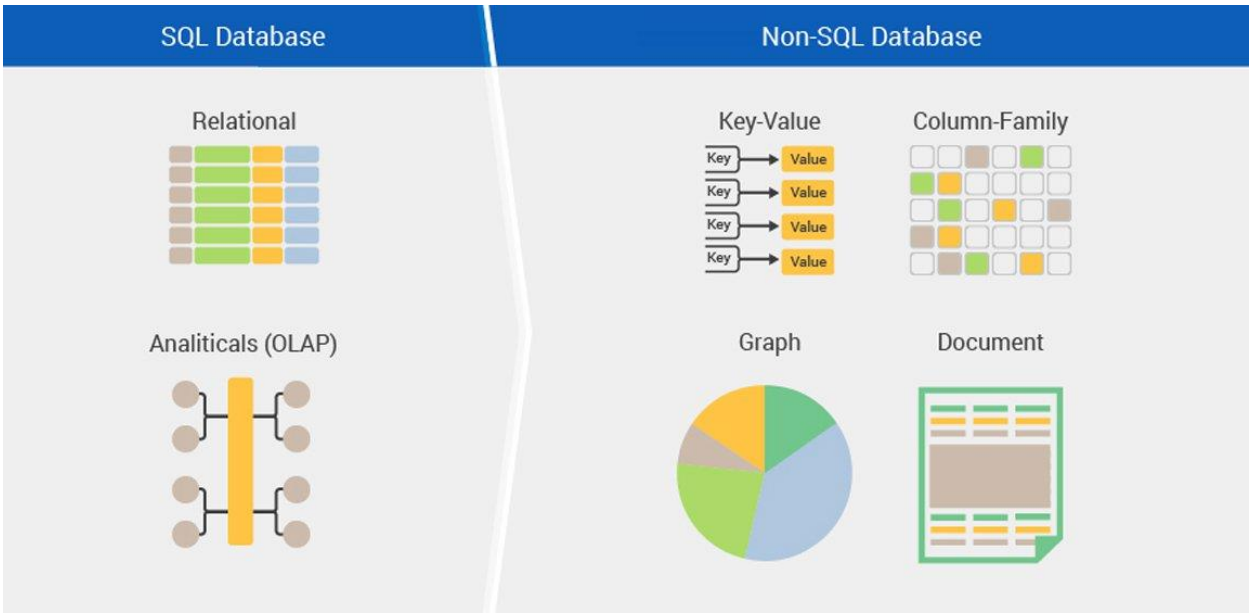

Figura 1 – Comparatie baze de date SQL si baze de date No-SQL

1.1 GDPR

Anul 2019 a adus cu el definirea unei politi international de stocare a datelor ce a dus la refacerea politicilor din toate organizatiile. Cateva detalii despre aceasta se regasesc in continuarea acestui subpunct.

In mod evident, datele cu caracter personal stocate pentru perioade mai lungi decat cele necesare prelucrarii pentru scopul identificat vor deveni in mod automat excesive. Totodata, ele ar putea deveni nerelevate si chiar inexacte.

Regulamentul nu stabileste perioada standard de stocare a datelor cu caracter personal si nici reguli detaliate care sa ajute operatorii ori persoanele imputernicite sa stabileasca aceasta perioada.
Revine asadar operatorilor sarcina sa stabileasca perioadele de retinere a datelor cu caracter personal prelucrate.

Stabilirea perioadei de stocare a datelor trebuie sa asigure un just echilibru intre nevoia operatorului de a retine datele cu caracter personal pe de o parte si drepturile si interesele legitime ale persoanelor vizate pe de alta parte.
Stergerea datelor prea devreme, in contextul in care operatorul ar putea avea (inca) nevoie sa le prelucreze, l-ar putea pune pe acesta intr-o situatie dificila. Totodata, stocarea datelor personale pentru mai mult timp decat este necesar risca sa incalce principiile prelucrarii datelor cu caracter personal, astfel cum acestea sunt prevazute in Regulament.

De asemenea, in cazul in care datele cu caracter personal sunt stocate mai mult decat este nevoie, va creste inutil volumul de date pentru care vor trebui asigure securitatea datelor si posibilitatea exercitarii drepturilor de catre persoanele vizate. In consecinta, adoptarea unei politici de retentie a datelor nu doar ca asigura respectarea Regulamentului, ci usureaza sarcina operatorului in ce priveste managementul datelor.

In contextul prelucrarii datelor cu caracter personal, pentru a se conforma regulilor privind retentia datelor, operatorii vor implementa doua tipuri de reguli interne:
a) politici de arhivare, in baza carora datele cu caracter personal care nu sunt prelucrate in activitatea curenta, dar pentru retinerea carora exista o justificare, sa fie arhivate cu respectarea garantiilor privind securitatea datelor
b) politici de stergere, in baza carora se vor revizui datele cu caracter personal prelucrate si se vor sterge, sau, dupa caz, se vor anonimiza acele date cu caracter personal de care nu mai este nevoie.

2. Definire

O definitie a bazelor de date NoSQL este oferita de site-ul **nosql-database.org**. Acesta descrie bazele de date NoSQL ca noua generatie de baze de date ce au urmatoarele specificatii: *nu sunt relationale, sunt distribuite, open-source si se caracterizeaza prin scalabilitate orizontala*. Alte caracteristici ce trebuiesc mentionate sunt lipsa unei scheme pentru a modela baza de date, prezinta suport pentru replicare, API, nu respecta in intregime ACID, stocheaza o cantitate mare de date.

O baza de date NoSQL ignora principiile RDBMS şi nu stocheaza date folosind tabele ci folosind chei de identificare. Datele pot fi regasite in functie de cheile atribuite. De asemenea acest tip de baze de date imbunatatesc si reactia la schimbari de-a lungul timpului. Intr-un sistem relational nu exista flexibilitatea necesara pentru a asimila schimbari asupra modelului de date. Faptul ca bazele de date NoSQL nu au o schema de date fixa face aceste baze de date sa fie mult mai flexibile şi adaptabile la schimbări de model in cursul anilor.

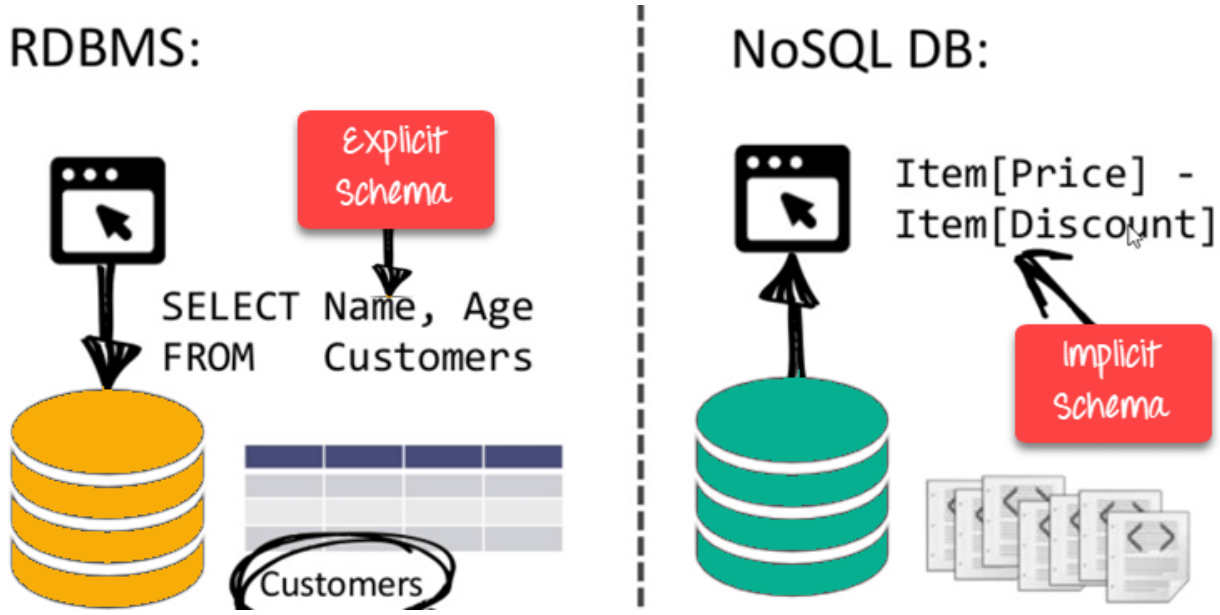

Figura 2 – Comparatie intre schema explicita (RDBMS) si schema implicita (NoSQL)

Conform definitiei date de Rick Cattell, bazele de date NoSQL prezinta 6 caracteristici de baza:

1. Abilitatea de a **scala** orizontal pe mai multe servere;

2. Abilitatea de a **replica** si **distribui** datele pe mai multe servere;

3. **CLI** *(call level interface)* caracterizat prin simplitate;

4. Un model concurential **mai slab** decat modelul relaţional;

5. Utilizarea eficienta a **indexarii** distribuite si a memoriei **RAM** pentru o stocare eficienta;

6. Abilitatea de a **adauga dinamic** noi atribute la inregistrarile existente.

3. Tipuri

a) Perechi cheie-valoare

Bazele de date 'key-value', **cheie valoare**, sunt conceptual disctionare distribuite si nu au schema predefinita. Cheia poate fi sintetica sau generate automati, iar valoare poate fi orice tip de date, inclusive BLOB-uri.

Un alt concept specific este bucket-ul, acesta reprezentat o grupare logica pentru mai multe chei-valori, realizand acest lucru fara sa grupeze fizic datele in acelasi loc pe disc. Pot exista chei identice, dar in bucket-uri diferite.

Pentru a interoga o valoare este necesara cunoasterea cheii si a bucket-ului, deoarece adevarata cheie sub care se pastreaza valoarea este o combinatie dintre cele doua.

Ca si analiza a teoremei CAP, bazele de date cheie- valoare exceleaza la A si P, dar sacrifice din C intr-o masura acceptabila si garanteaza faptul ca o sa ofere foarte rapid date, dar nu mereu actulizate.

Exemple: Dynamo, InfinityDB, Arango DB

b) Stocare documente

MondoDB este cea mai populara baza de date ce se bazeaza pe documente. Acestea sunt flexibile la tipul de date, deoarece nu au o schema predefinita. Conceptual lucreaza cu documente de diferite forme: XLS, PDF, JSON, XML etc.

In esenta nu sunt decat o specializare a bazelor de date cheie-valoare. Un document se scrie/citeste folosind o cheie. Pe langa functionalitatea de cheie-valoare, bazele de date de tip document adauga functionalitati de gasire a documentelor bazat pe continutul acestora. Raportat din nou la teorema CAP, bazele de date orientate document exceleaza la C si P.

Exemple: Apache CouchDB, BaseX, CosmosDB, MongoDB.

c) Graf

Acest tip de baza de date este proiectat pentru datele care pot fi reprezentate ca un graf, constand in elemente interconectate cu un number finit de relatii intre ele. Aceste tipuri de date pot fi de exemplu: relatii sociale, puncta de transport public, harta, retele tehnologice etc.

Exemple: IBM DB2, Alegro Graph, ArangoDB

d) Orientate obiect

In aceste baze de date informatiile sunt reprezentate sub forma de obiecte, asemanator cu obiectele utilizate in programare. Acestea reprezinta o combinatie intre bazele de date relationale si cele non-relatiionale.

Exemple: ODABA, Perst, JADE

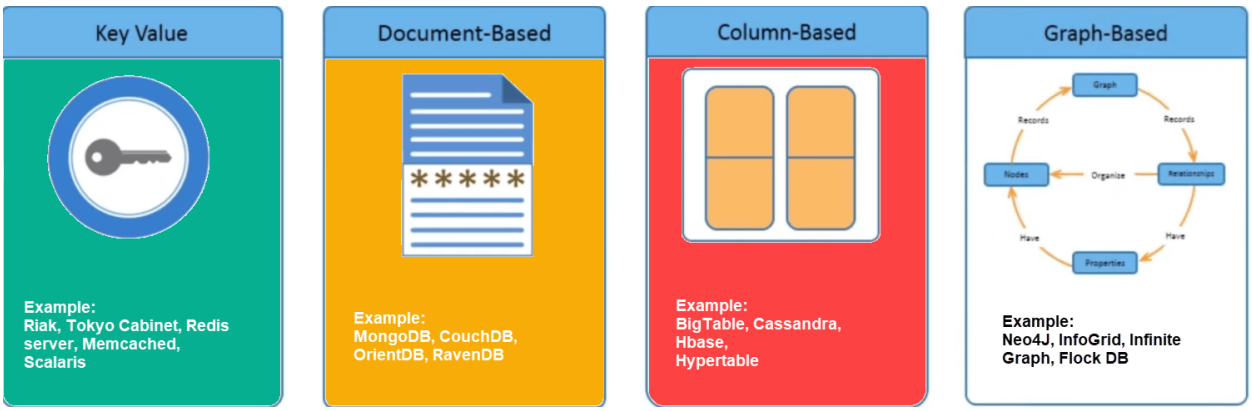

Figura 3 – Tipuri de baze de date NoSQL

4. Avantaje

Bazele de date NoSQL prezinta o serie de avantaje ce le face extrem de folositoare in contextual din prezent, dar nu reprezinta rezolvarea pentru orice tip de aplicatie. Exista si cazuri in care solutiile relationale se potrivesc mai bine.

a) Portabilitate

Bazele de date NoSQL utilizeaza servere ce respecta modelul DHT (Distributed Hash Table), iar acest lucru se traduce in faptul ca utilizatorul are nevoie de o cheie primara pentru a interoga si adauga date.

b) Modele simple

Procesarea datelor se realizeaza online, lucru care duce la o performanta ridicata si la posibilitatea de a scala pe orizontala. In cazul datelor de sesiune ale utilizatorului exista posibilitatea de a retine copii ale acestora in sisteme NoSQL.

c) Capacitate de transfer ridicata

Aceste sisteme permit un volum ridicat de transfer, comparative la sistemele RDBMS, iar acesta limita poate ajunge chiar la zeci de petabytes pe zi.

d) Performanta

Arhitectura acestor baze de date, impreuna cu toate avantajele cumulate duc la o performanta superioara bazelor de date traditionale, in contextual unui volum ridicat de date.

e) 'Schemaless' – fara schema

Acestea nu au nevoie de o definirea unei scheme initiale, lucru care simplica foarte mult dezvoltarea unui sistem ce foloseste acest tip de baze de date. Acest aspect duce si la beneficiul de a putea adauga dinamic date in plus, la informatii existente.

f) Tipuri de date

Asa cum a fost specificat si anterior, aceste sisteme permit utilizarea multor tipuri de date. Acestea ofera o solutie pentru majoritatea situatiilor ce pot aparea, in materie de tipuri de date.

Fiecare caz poate fi tratat cu un tip diferit de baza de date NoSQL. Cateva exemple sunt: utilizarea stocarii cheie valoare pentru valori simple, precum liste sau mape si utilizarea stocarii de documente in cazul datelor complexe de tip ierarhic.

g) Open source

Aceasta industrie ca crescut exponential, lucru ce a dus la aparintia multor concurenti in aceasta zona, atat folosind un model de business monetizat, cat si open source.

Variantele open-soruce permit companiilor sa experimentele in contextual lor utilizarea bazelor de date NoSQL fara sa investeasca o suma ridicata de bani.

Toate aceste avantaje contribuie la cresterea constanta a popularitatii acestor tipuri de baze de date si la extinderea posibilitation solutiilor informatice in materie de viteza si procesare de date.

5. Proiectare

Bazele de date NoSQL urmeaza acelasi sistem, acelasi set de pasi la fel ca bazele de date traditionale. Acestea trebuie sa realizeze modelul datelor, sa le transforme in date logice (alegerea tipului de baza de date) si sa implementeze acest sistem.

a) Fazele proiectarii

- Proiectare conceptuala

In acest pas logica de business se imbina cu ingineria dezvoltarii unui sistem de baze de date si reiese conceptul datelor, modelul pe care il va respecta aplicatia.

- Proiectare logica

Proiectarea logica consta in dezvoltarea unei scheme logice, incepand de la schema conceptuala realizata anterior. In acest pas nu tinem cont de solutia software ce urmeaza a fi folosita, organizarea trebuie sa fie independenta de aceasta.

- Proiectare fizica

Proiectarea fizica este momentul in care se dezvolta schema fizica a bazei de date. Aceasta descrie structura datelor logice realizata anterior, in contextual unui produs software.

b) Exemplu proiectare – Firebase (Firestore)

In acest exemplu mi-am propus realizarea unei aplicatii de comenzi de mancare, ce utilizeaza pentru persistenta, solutia de baze de date NoSQL oferita de Firebase, Firestore. Aceasta baza de date este de tip document.

- Interfata grafica

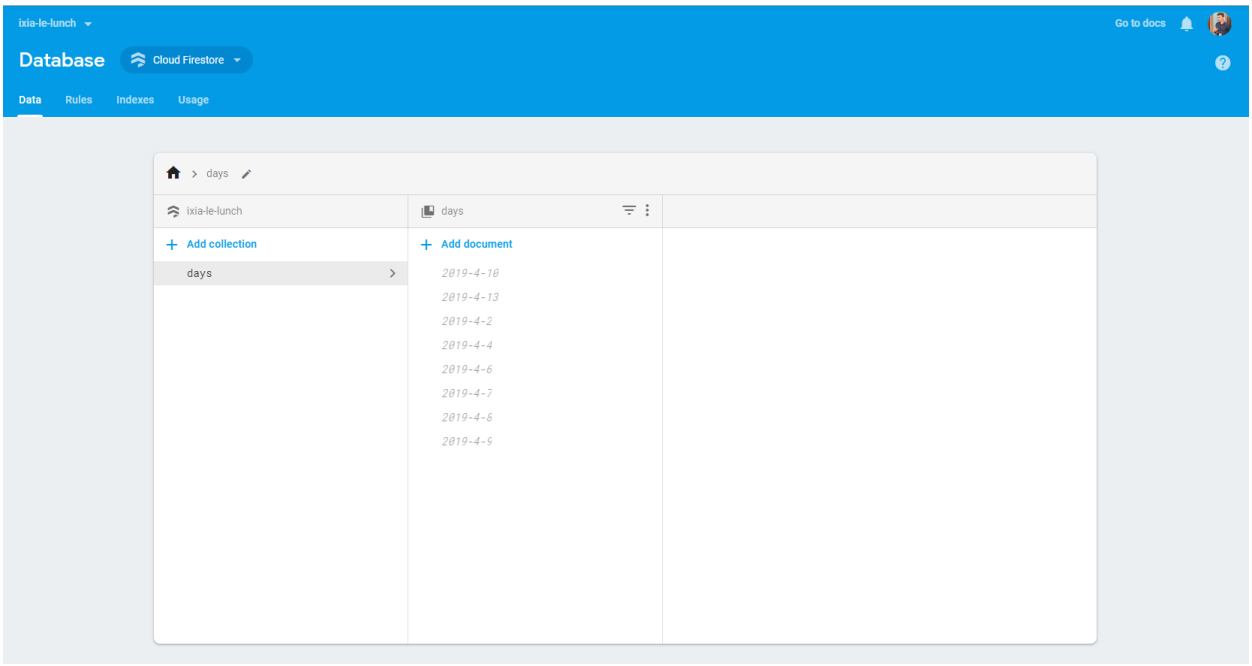

Figura 4 – Interfata grafica Firebase

- Structura datelor

Ca si model de date am folosit urmatoarele:

Colectia days in care se vor stoca in fiecare zi comenzile. Aceasta contine mai multe **documente** cu formatul “YYYY-MM-DD”, pentru fiecare zi in parte. Fiecare document contine un vector denumit **orders**, care la randul lui contine o serie de **documente** semnificand fiecare comanda realizata, cu campurile **name** si **order**, de tip string, si **sum**, de tip number.

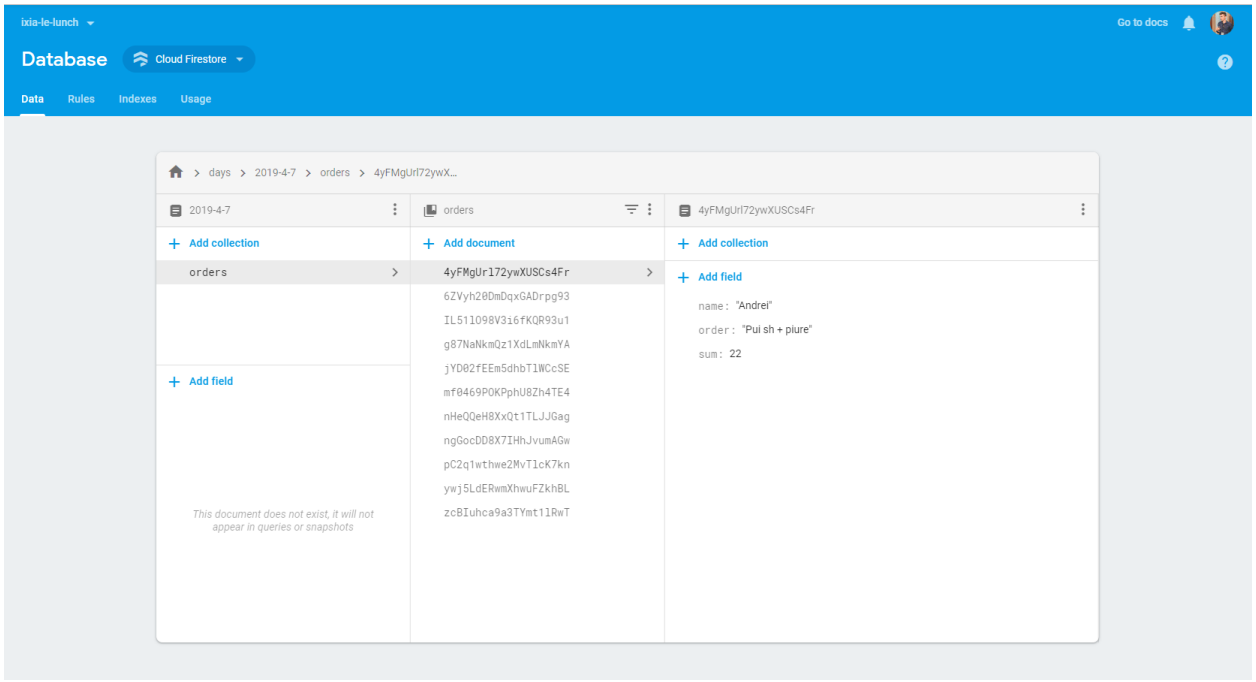

Figura 5 – Structura datelor

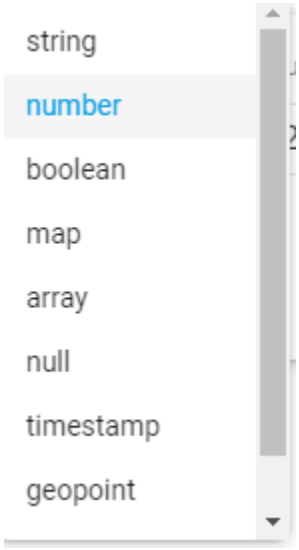

Figura 6 – Tipuri de date acceptate

- API utilizare

Acesta baza de date ofera un API prin care se pot realiza interoragari si adaugari de date. Prin query-ul **“db.collection”** putem interoga o colectie, iar prin **“.doc”** putem accesa un document.

Prin metoda **“.get”** putem obtine datele din baza de date.

De asenemea, prin metoda **“.set”** putem modifica datele din interiorul unui document.

```
app.get("/orders", (req, res) => {
    const ordersRef = db.collection("days").doc(getDate()).collection("orders");

    ordersRef.get()
    .then((snapshot) => {
        if(snapshot.empty) {
            res.status(200).send([]);
        } else {
            let result = [];

            snapshot.forEach(document => {
                result.push(document.data());
            })

            res.status(200).send(result);
        }
    })
    .catch(error => {
        console.error(error);

        res.status(500).send("You have work to do!");
    })
})

app.post("/order", (req, res) => {
    const newDocument = db.collection("days").doc(getDate()).collection("orders").doc();

    newDocument.set(req.body);

    res.status(200).send("Added!");
})
```

Figura 7 – Exemplu API Firebase

- Exemple rezultate

Urmatoarele doua figure reprezinta request-uri prin care se pot interoga si adauga date. Prin request-ul de POST se adauga o noua comanda, iar prin cel de order se obtin toate comenzile din ziua curenta.

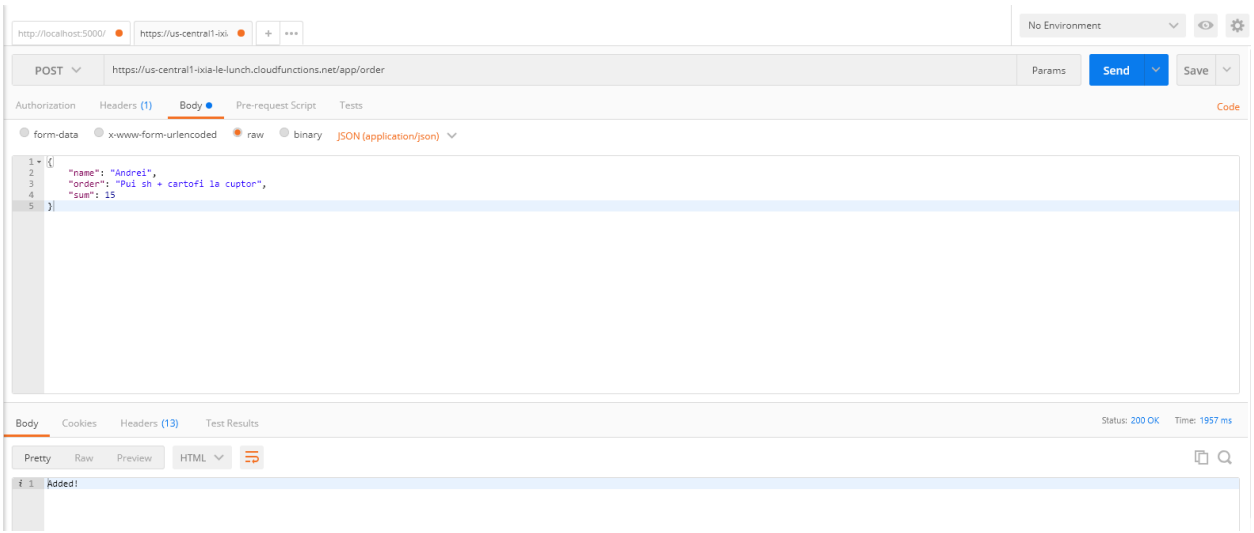

Figura 8 – Request POST, adaugare date

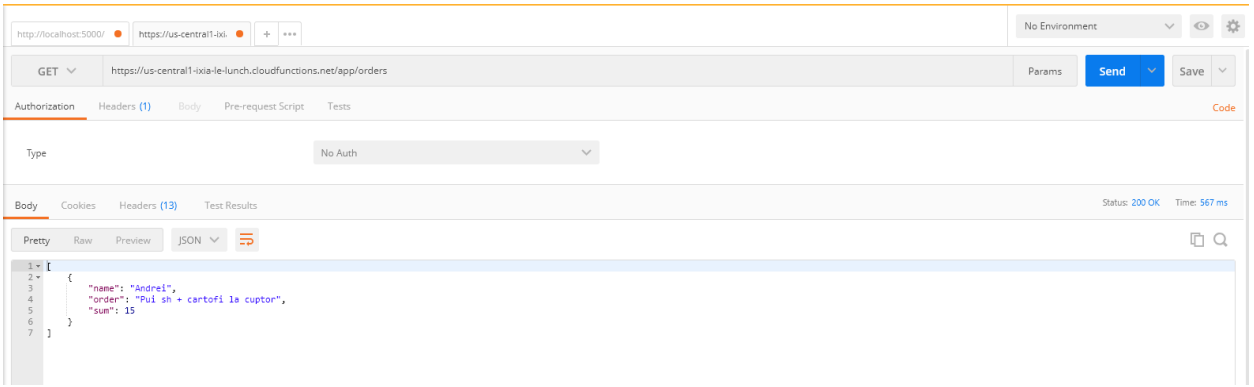

Figura 9 – Request GET, interogare date

6. Concluzii

Asadar, dupa analizarea informatiilor prezentate in capitolele precedente, putem trage o prima concluzie. Aceasta consta in faptul ca aplicatiile web noi ce isi propun sa intre pe piata se orienteaza direct catre un sistem de baze de date NoSQL.

Totusi, aceasta alegere trebuie realizata in situatia in care ne propunem de la inceput sa lucram cu volume foarte mari de date. In caz contrar restrictiile unei baze de date SQL ar putea totusi sa aduca mai multe beneficia pe termen lung aplicatiei.

In opinia mea, aceste sistem, atat cele traditionale cat si cele moderne, ofera solutii pentru probleme diferite. Tine de personalele responsabile proiectarii acestora sa faca alegerea in functie de nevoia pe termen mediu si lung.

Ca o directive de viitor consider ca bazele de date NoSQL ar trebui sa evolueze intr-o directive in care sa impuna mai multe restrictii, pentru a putea organiza mai bine datele in organizatii foarte mari, in care sunt implicate mai multe echipe, din tari si orase diverse.

Bibliografie

https://support.office.com/ro-ro/article/prezentare-general%C4%83-a-olap-online-analytical-processing-15d2cdde-f70b-4277-b009-ed732b75fdd6

http://nosql-database.org/

https://en.wikipedia.org/wiki/NoSQL

https://en.wikipedia.org/wiki/Relational_database_management_system

https://www.mongodb.com/scale/advantages-of-nosql

https://firebase.google.com/

https://firebase.google.com/docs/firestore/