Proiect la baze de date

-Cerința 19-

Justificarea utilității migrării la o bază de date de tip NoSql

Justificarea necesității/utilității migrării la o bază de date de tip NoSql. Identificarea scenariilor în care utilizarea unei baze de date NoSQL este mai avantajoasă decât a unei baze de date relaționale.

a)Prezentarea structurii baze de date de tip NoSQL.

O bază de date NoSQL este o bază de date nerelațională ce stochează datele diferit. NoSQL funcționează cu modele flexibile de date, în comparație cu SQL, unde datele sunt păstrate în tabele predefinite. Spre exemplu, MongoDB stochează datele sub forma unui document JSON.

Pentru a prezenta motivele migrării la o astfel de bază de date, este esențial să înțelegem întâi diferențele dintre aceste două tipuri de baze de date. În primul rand, structural, SQL asigură o structură sigură a datelor la prețul flexibilitătii. Este important ca atunci cand pornim o baza de date SQL sa planificăm cu atenție modul în care vom împărți informațiile. Pe de altă parte, NoSQL este un tip mai flexibil, ce necesită mai puțin "setup".

Astfel e mai ușor sa dezvoltăm o bază de date NoSQL, însă este nevoie sa aplicăm mai multe principii ulterior pentru a asigura consistență între date și legăturile dintre acestea.

În al doilea rand, SQL se bazează în principal pe "vertical scaling" (se îmbunătățește hardware ul serverului), pe cand NoSQL funcționează pe conceptul de "horizontal scaling" (creșterea numărului de servere, folosind mai multe noduri pentru a împărți resursele hardware). Acest principiu vine cu dezavantajul lipsei de consistență între noduri și riscul ca unul să cadă, lăsând "goluri".

O altă diferență se găsește în modalitatea de interogare: pentru SQL se folosește un limbaj prestabilit, standardizat, iar pentru NoSQL nu este unul exact, ci se folosește de multiple API-uri sau interfețe de interogare, în funcție de modelul ales.

În ultimul rând, aceste baze de date sunt folosite în scopuri diferite: SQL este folosit cand cunoaștem structura foarte bine, în structuri de planificare ce au nevoie de interogari complexe. Bazele de date NoSQL sunt folosite în cazuri precum aplicații mobile și interfete folosite deseori de alți utilizatori. În linii mari, SQL strălucește în cazul în care este nevoie de o structura bine definită, în timp ce NoSQL este potrivit pentru un volum mare de date ce necesita flexibilitate.

Așadar, pentru baza de date a stocului unui magazin de muzica ce reține datele clienților și are nevoie constanta de actualizări referitoare la comenzile efectuate de aceștia, ar fi mai o idee potrivită să experimentăm și cu un tip nerelațional, având în vedere faptul că putem adăuga comenzile într-o manieră dinamică.

Am ales MongoDB ca sistem de baze de date NoSQL, iar în următoarele puncte voi prezenta procesul de migrare. Acesta va fi prezentat în mod conceptual.

b)Prezentarea comenzilor pentru crearea bazei de date.

Înainte de a crea baza de date, am ales să denormalizez structura pentru a o potrivi mai bine cerințelor unei baze de date nerelaționale. După cum am prezentat anterior, este mai ușor să structurăm așa-zisele interogări în MongoDB atunci când nu am tabele multiple. Astfel prefer să unesc câteva tabele între ele, pastrându-le separate doar pe cele relevante:

- Pastrez tabelul “Clienti”: acesta va retine atat datele despre clienti cat si comenzile lor. (efect al denormalizarii)
- Pastrez tabelul “Produse” desi este optional, avand produsele comandate in profilul clienților, reținem si produsele ce nu au fost comandate pana la momentul respectiv.
- Pastrez tabelul “Note”, considerandu-l relevant pentru diverse interogari si statistici

Pentru aceste acțiuni, voi folosi extensia “MongoDB for VSCode”.

- În primul rând, voi migra tabelele pe care le-am folosit în baza de date relațională, in urma modificarilor precizate mai sus, folosind comenzile:

```
use proiectbd
db.createCollection("clienti")
db.createCollection(“produse”)
db.createCollection(“note”)
```

- Al doilea pas în acest proces este similar cu cel al bazei de date relaționale, și anume popularea tabelelor cu informațiile transferate din baza de date inițială.

```
db.clienti.insertMany([
 {
  _id: ObjectId("1"),
  nume: "Camil Petrescu",
  nr_telefon: "0719601988",
  email: "patullui.procust@gmail.com",
  zi_de_nastere: ISODate("1974-05-12"),
  comenzi: [
   {
    id_comanda: ObjectId("1"),
    data: ISODate("2025-06-01"),
    nume_curier: "Mickey Mouse",
    pret: 598,
    mod_de_plata: "card",
    produse: [
```

```
      {
        id_produs: ObjectId("10"),
        nume: "This is all Yours",
        cantitate: 2,
        pret: 200,
        tip: "album",
        artist: "Alt-J",
        record_label: "Universal"
      },
      {
        id_produs: ObjectId("12"),
        nume: "Sweetener",
        cantitate: 1,
        pret: 198,
        tip: "vinyl",
        artist: "Ariana Grande",
        record_label: "Sony Music"
      }
    ]
  }
 ]
},
{
 _id: ObjectId("2"),
 nume: "Marius Andrei",
 nr_telefon: "0711235813",
 email: "marius.andrei@yahoo.com",
 zi_de_nastere: null,
 comenzi: [
  {
   id_comanda: ObjectId("2"),
   data: ISODate("2025-06-02"),
   nume_curier: "Curierul Fan",
   pret: 45,
   mod_de_plata: "ramburs",
   produse: [
    {
     id_produs: ObjectId("3"),
     nume: "Arhitectul din Babel",
     cantitate: 1,
     pret: 45,
     tip: "album",
     artist: "Alternosfera",
     record_label: "Sony Music"
    }
   ]
```

```
    }
   ]
  }
])
```

```
db.produse.insertMany([
  {
    _id: ObjectId(),
    nume: "This is all Yours",
    pret: 200,
    tip: "album",
    artist: "Alt-J",
    record_label: "Universal"
  },
  {
    _id: ObjectId(),
    nume: "Sweetener",
    cantitate: 1,
    pret: 198,
    tip: "vinyl",
    artist: "Ariana Grande",
    record_label: "Sony Music"
  },
  {
    _id: ObjectId(),
    nume: "Arhitectul din Babel",
    cantitate: 1,
    pret: 45,
    tip: "album",
    artist: "Alternosfera",
    record_label: "Sony Music"
  }
])
```

```
db.note.insertMany([
  {
    id_client: ObjectId("665afc901234abcd5678e001"),
    id_produs: ObjectId("665afc901234abcd5678e003"),
    nota: 4.5
  },
  {
    id_client: ObjectId("665afc901234abcd5678e005"),
    id_produs: ObjectId("665afc901234abcd5678e007"),
```

```
    nota: 5.0
  }
])
```

c)Prezentarea comenzilor pentru inserarea, modificarea și ștergerea documentelor sau înregistrărilor într-o bază de date NoSQL.

Deși am prezentat comenzile de inserare în subpunctul anterior, voi demonstra câteva operații de modificare și ștergere din baza de date non-relațională.

- Update-uri: Prima interogare schimbă emailul unui client, modificându-l după keyword-ul de nume, iar cea de-a doua adaugă o nouă comandă unui client.

```
db.clienti.updateOne(
  { nume: "Camil Petrescu" },
  { $set: { email: "unddindr@hotmail.com" } }
)

db.clienti.updateOne(
  { nume: "Marius Andrei" },
  {
    $push: {
      comenzi: {
        id_comanda: ObjectId(),
        data: ISODate("2025-06-18"),
        nume_curier: "Curierul Fan",
        pret: 198,
        mod_de_plata: "card",
        produse: [
           {
           id_produs: ObjectId(),
           nume: "Sweetener",
           cantitate: 1,
           pret: 198,
           tip: "vinyl",
           artist: "Ariana Grande",
           record_label: "Sony Music"
         }
        ]
      }
    }
  }
)
```

- Ștergeri: Această comandă șterge din baza de date clientul cu numele respectiv.

```
db.clienti.deleteOne({ nume: "Camil Petrescu" })
```

d)Exemplificarea comenzilor pentru interogarea datelor, incluzând operațiuni de filtrare și sortare.

- Interogarea găsește și afișează clienții ce au avut o comandă livrată de “Curierul Fan”.

```
db.clienti.find({
  "comenzi.nume_curier": "Curierul Fan"
})
```

- Interogarea afișează clientul cu numele selectat, în acest caz “Camil Petrescu”.

```
db.clienti.find({ nume: "Camil Petrescu" })
```

- Sortează descrescător clienții în funcție de cât de recente au fost comenzile efectuate de aceștia.

```
db.clienti.find().sort({ "comenzi.data": -1 })
```

Concluzii, observații:

În urma studierii modelului de baze de date non-relațional, am înțeles și în practică diferența dintre cele două tipuri, avantajele și dezavantajele fiecăruia. Sunt de părere că pentru cerințele mele este mai folositor un model relațional, precum cel dezvoltat până acum, deoarece un magazin de muzică desi are intrări dinamice (ex. comanda fiecărui client), este mai importantă reținerea multor informații într-un tabel structurat cu mai multe restricții, datorită stocului mare de produse și numărului mare de clienți. În cazul în care baza de date non-relațională ar fi fost folosită într-un context real, ar fi existat mult prea multe redundanțe, iar stocul produselor și informații despre acestea ar fi accesate mult mai dificil decât în cazul inițial.

Motivul pentru care nu am folosit interogări complexe este faptul că există posibilitatea ca unele informații să se repete, rezultând în overlapping-ul datelor. Astfel, revin la concluzia explicată anterior, cum că modelul relațional este mai avantajos în contextul proiectului meu.