Aceste exercitii de mai jos sunt in faza de draft final. Va rog sa-mi spuneti daca aveti recomandari de imbunatatire si le voi implementa.

18. Cerinta : Exemplificarea *isolation levels* prin exemple de tranzacţii care se execută în paralel în condiţii de concurenţă, evidenţiind efectele diferitelor niveluri de izolare asupra concurenţei şi integrităţii datelor.

Raspuns:

19. Cerinta : Justificarea necesităţi/utilităţii 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 relaţionale.
    a) Prezentarea structurii baze de date de tip NoSql.
    b) Prezentarea comenzilor pentru crearea bazei de date (spre exemplu a colecţiilor într-o bază de date de tip document)
    c) Prezentarea comenzilor pentru inserarea, modificarea şi ştergerea documentelor sau înregistrărilor într-o bază de date NoSQL.
    d) Exemplificarea comenzilor pentru interogarea datelor, incluzând operaţiuni de filtrare şi sortare

Raspuns:

Avantajele bazelor de date NoSQL Bazele de date NoSQL oferă mai multe avantaje care le fac potrivite pentru anumite scenarii:

- Flexibilitate în schema de date - nu necesită definirea unei scheme fixe în avans, permiţându-ţi să stochezi date cu structuri diferite în aceeaşi bază de date. Astfel poti să te mişti rapid şi să stabileşti modelul de date din mers.
- Scalabilitate orizontală - folosesc un proces numit "partiţionare" pentru a scala orizontal, adăugând mai multe computere pentru gestionarea datelor pe mai multe servere.
- Performanţă ridicată: Pentru anumite tipuri de operaţiuni, în special pentru citiri şi scrieri simple, bazele de date NoSQL pot oferi o performanţă superioară.
- Disponibilitate ridicată: Multe baze de date NoSQL sunt proiectate pentru a asigura o disponibilitate ridicată a datelor, chiar şi în cazul unor defecţiuni ale sistemului.

Scenarii avantajoase pentru utilizarea bazelor de date NoSQL

- Bazele de date NoSQL sunt ideale pentru aplicaţii care necesită operaţiuni rapide de citire şi scriere, cum ar fi aplicaţiile de chat sau platformele IoT.
- Pentru prelucrarea şi analizarea volumelor mari de date nestructurate sau semi-structurate, bazele de date NoSQL oferă performanţă şi flexibilitate superioare.
- Platformele care stochează cantităţi mari de date nestructurate, cum ar fi postările şi interacţiunile utilizatorilor, beneficiază de flexibilitatea bazelor de date NoSQL.
- Când datele sunt distribuite pe mai multe servere şi trebuie să fie recuperate rapid, bazele de date NoSQL sunt o alegere excelentă.
- Proiectele care evoluează rapid şi necesită modificări frecvente ale structurii datelor pot beneficia de schema flexibilă a bazelor de date NoSQL.

a) Prezentarea structurii baze de date de tip NoSql.
Bazele de date NoSQL pot fi clasificate în mai multe tipuri, fiecare cu propria structură şi caz de utilizare, dupa cum urmeaza:

- Baze de date orientate pe documente - stochează datele în format document (de obicei JSON sau BSON). MongoDB este un exemplu popular de bază de date orientată pe documente. Baza de date conţine colecţii (echivalentul tabelelor din SQL), Colecţiile conţin documente (echivalentul rândurilor), Documentele conţin câmpuri (echivalentul coloanelor), Documentele din aceeaşi colecţie pot avea structuri diferite.
- Baze de date cheie-valoare - stochează datele ca perechi de chei şi valori. Exemple includ Redis şi Riak.
- Baze de date orientate pe coloane - stochează datele în coloane în loc de rânduri, ceea ce le face eficiente pentru anumite tipuri de interogări analitice. Exemple includ HBase şi Cassandra.
- Baze de date de tip graf – au entitati si relatiile intre ele. Neo4J este un exemplu popular.

b) Comenzi pentru crearea bazei de date NoSQL - MongoDB

use nume_baza de date - Această comandă creează o bază de date nouă sau comută la o bază de date existentă.

db.createCollection(“nume_colectie”) – creaza explicit o colecţie, desi colectiile pot fi create

implicit cand inserezi primul document.

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

Inserarea documentelor:

```
db.nume_colectie.insert({
  "nume": "Test",
  "prenume": "Utilizator",
  "varsta": 30,
  "gen": "M",
  "tara": "Romania"
})
```

Pentru a insera mai multe documente odată:

```
db.nume_colectie.insertMany([
  { "nume": "Ion", "varsta": 25 },
  { "nume": "Maria", "varsta": 30 }
])
```

Modificarea documentelor:

```
db.nume_colectie.update(
  { "gen": "F" },
  { $set: { "tara": "Anglia" } },
  { multi: true }
)
```

Această comandă actualizează toate documentele care au câmpul "gen" cu valoarea "F", setând câmpul "tara" la valoarea "Anglia".

Ștergerea documentelor care au campul “gen” cu valoarea “M”:

```
db.nume_colectie.remove({ "gen": "M" })
```

d)Exemplificarea comenzilor pentru interogarea datelor, incluzând operaţiuni de filtrare şi sortare

Interogări pentru a gasi toate documentele dintr-o colectie:

```
db.nume_colectie.find()
```

Filtrare documente care au gampul “gen” cu valoarea “F”

```
db.nume_colectie.find({ "gen": "F" })
```

Filtrare doar pe anumite câmpuri:

```
db.nume_colectie.find(
  { "gen": "F" },
  { "nume": 1, "varsta": 1 }
)
```

Sortare rezultate - găseşte documentele cu genul "F", selectează câmpurile "nume" şi "varsta", şi le sortează în ordine crescătoare după vârstă.

```
db.nume_colectie.find(
  { "gen": "F" },
  { "nume": 1, "varsta": 1 }
).sort({ "varsta": 1 })
```