## *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.*

1. Cerinta : Exemplificarea *isolation levels* prin exemple de tranzacții care se execută în paralel în condiții de concurență, evidențiind efectele diferitelor niveluri de izolare asupra concurenței și integrității datelor.

**Raspuns:**

1. Cerinta : Justificarea necesități/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.
   1. Prezentarea structurii baze de date de tip NoSql.
   2. Prezentarea comenzilor pentru crearea bazei de date (spre exemplu a colecțiilor într-o bază de date de tip document)
   3. Prezentarea comenzilor pentru inserarea, modificarea și ștergerea documentelor sau înregistrărilor într-o bază de date NoSQL.
   4. Exemplificarea comenzilor pentru interogarea datelor, incluzând operațiuni de filtrare și 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, permițându-ți să stochezi date cu structuri diferite în aceeași bază de date. Astfel poti să te miști rapid și să stabilești modelul de date din mers.
* Scalabilitate orizontală - folosesc un proces numit "partiționare" pentru a scala orizontal, adăugând mai multe computere pentru gestionarea datelor pe mai multe servere.
* Performanță ridicată: Pentru anumite tipuri de operațiuni, în special pentru citiri și scrieri simple, bazele de date NoSQL pot oferi o performanță superioară.
* Disponibilitate ridicată: Multe baze de date NoSQL sunt proiectate pentru a asigura o disponibilitate ridicată a datelor, chiar și în cazul unor defecțiuni ale sistemului.

Scenarii avantajoase pentru utilizarea bazelor de date NoSQL

* Bazele de date NoSQL sunt ideale pentru aplicații care necesită operațiuni rapide de citire și scriere, cum ar fi aplicațiile de chat sau platformele IoT.
* Pentru prelucrarea și analizarea volumelor mari de date nestructurate sau semi-structurate, bazele de date NoSQL oferă performanță și flexibilitate superioare.
* Platformele care stochează cantități mari de date nestructurate, cum ar fi postările și interacțiunile utilizatorilor, beneficiază de flexibilitatea bazelor de date NoSQL.
* Când datele sunt distribuite pe mai multe servere și trebuie să fie recuperate rapid, bazele de date NoSQL sunt o alegere excelentă.
* Proiectele care evoluează rapid și 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 și 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 conține colecții (echivalentul tabelelor din SQL), Colecțiile conțin documente (echivalentul rândurilor), Documentele conțin câmpuri (echivalentul coloanelor), Documentele din aceeași colecție pot avea structuri diferite.
* Baze de date cheie-valoare - stochează datele ca perechi de chei și valori. Exemple includ Redis și 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 și 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 colecție, 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 operațiuni de filtrare și 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** - găsește documentele cu genul "F", selectează câmpurile "nume" și "varsta", și le sortează în ordine crescătoare după vârstă.

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