**Aplicatie web care converteste unele comenzi din format SQL in format NoSQL (MongoDB)**

**Proiect Big Data**

**Servicii Software Avansate, anul I**

Studenti

Cristina Buciu

Sergiu Isopescu

Claudiu Nedelcu

**Introducere**

Bazele de date reprezinta colectii de date. In literature de specialitate o baza de date se refera la intregul sistem de gestiune a bazei de date, insa la nivelul acestui proiect acest termen va referi strict colectia de date. Bazele de date au fost create din dorinta de a stoca si gasi cu usurinta date intr-un mod simplu si eficient. De-a lungul timpului, s-au dezvoltat 2 tipuri de arhitecturi de baze de date, SQL si NoSQL. Diferenta dintre acestea este modul de stocare a informatiei si modul de interogare a bazei de date.

In cazul bazelor de date SQL, informatia este organizata in tabele, fiecare intrare dintr-o tabela are aceeasi forma, are valori pentru aceleasi atribute, iar performanta este scazuta. Daca se doreste adaugarea unui atribut nou pentru o tabela, trebuie refacuta intreaga arhitecura a tabelei, adaugat din nou tot continutul existent, iar interogarea tabelelor este slab cotata ca timp.

In cazul bazelor de date NoSQL, informatia este organizata in colectii, fiecare colectie este alcatuita din documente in format JSON cu continut variat. Astfel, daca se doreste stocarea unui atribut nou, este suficient sa-l adaugam in structura documentelor dorite. Bazele de date NoSQL ofera flexibilitate, scalabilitate si perfomanta in privinta interogarii.

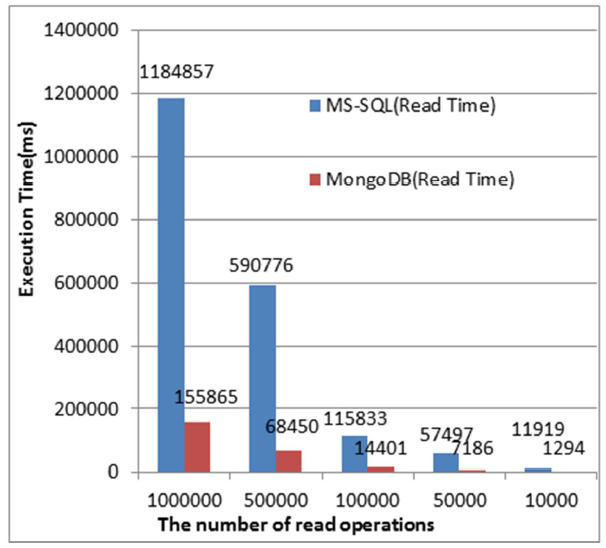
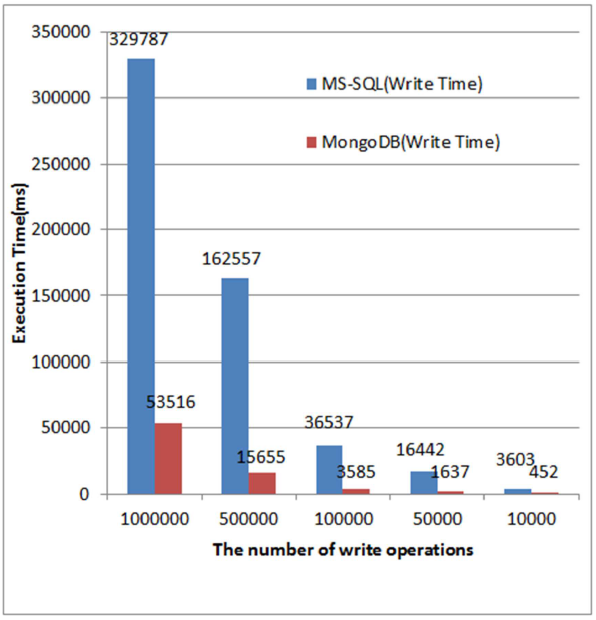


Figura 1: Comparatie timpi de executie pentru scriere/citire

**MongoDB**

MongoDB este o baza de date NoSQL, orientata pe documente si folosita pentru stocarea volumelor mari de date. Aceasta asigura agilitate si scalabilitate in businesses. Mai multe companii(Facebook, Google, Amazon) au decis sa foloseasca MongoDB datorita perfomantelor foarte bune pe care le ofera.

Fiecare baza de date MongoDB contine colectii alcatuite din documente. Fiecare document este in format JSON si poate fi alcatuit dintr-un numar variat de campuri (date). Documentele bazei de date MongoDB nu au nevoie sa respecte o schema a bazei de date, intrucat aceasta este creata dinamic. MongoDB permite stocarea de liste, dictionare si alte structuri de date complexe.

**Motivatie**

Conform articolului [1] privind comparatia dintre cele doua tipuri de baze de date si cu sustinerea Figurii 1, se urmareste migrarea bazelor de date SQL in format NoSQL din motive de performanta. Din punct de vedere al timpului de executie, atat pentru operatiile de citire, cat si pentru operatiile de scriere, se remarca eficienta bazei de date MongoDB (NoSQL).

De asemenea, conform articolului [2], performanta privind timpii de executie se conserva inclusiv in cazul operatiilor multi-threading.

**Abordari existente**

In urma analizei solutiilor de conversie existente deja pe internet, am remarcat doua aplicatii web [3] [4] care realizeaza doar conversia comenzilor de interogare SQL. Aplicatia propusa urmareste extinderea paletei de comenzi SQL convertibile in format NoSQL.

De asemenea o alta aplicatie care realizeaza o conversie intre SQL si NoSQL este aplicatia Studio 3T care poate fi descarcata si folosita contra cost de pe site-ul acesteia [5]. Migratia SQL la MongoDB permite importarea mai multor tabele SQL intr-o singura colectie MongoDB inclusiv relationarea dintre tabele(one-to-one, one-to-many)

**Conversie comenzi SQL-NoSQL**

|  |  |  |
| --- | --- | --- |
| **Comanda** | **SQL** | **NoSQL** |
| Creare tabela / colectie | CREATE TABLE table\_name (     column1 datatype,     column2 datatype,     column3 datatype,    .... ); | db.createCollection(  ‘collectionName’  ); |
| Stergere tabela / colectie | DROP TABLE table\_name; | db.collectionName.drop(); |
| Stergere linie / document | DELETE FROM table\_name  WHERE condition; | db.collectionName.drop(  {conditions}  ); |
| Inserare linie / intrare | INSERT INTO table\_name  (column1, column2, ...) VALUES  (value1, value2, ...); | db. collectionName.insert({  ' column1': ' value1',  ' column2': ' value2'  ...  }); |
| Interogare cu conditii | SELECT col1, col2, ... FROM table\_name WHERE conditions  ORDER BY *col1, col2, ...* ASC | DESC; | db.collectionName.find({  col1: 1  col2: 1  ...  }, { conditions }).sort({  col1: 1 | -1  col2: 1 | -1  ...  }); |
| Interogare cu conditii si grupare | SELECT col1/function1, col2/function2, ... FROM table\_name WHERE conditions  GROUP BY *col1, col2, ...*  ORDER BY *col1, col2, ...* ASC | DESC; | db.collectionName.group({key: {  col1: 1  col2: 1  ...  }, initials : {function1val=0 ...}  , reduce: function(prev, obj){// apply function}  , finalize: function(prev) {}  “cond”: { conditions }).sort({  col1: 1 | -1  col2: 1 | -1  ...  }); |
| Interogare pe mai multe tabele (inner join) | SELECT col1, col2, ... FROM table\_name1  INNER JOIN table\_name2  ON condition\_join WHERE conditions  ORDER BY *col1, col2, ...* ASC | DESC; |  |

Termenii folositi in MongoDB sunt prezentati in tabelul Q, unde se observa ca pentru fiecare element din SQL putem gasi o solutie si in NoSQL.

|  |  |
| --- | --- |
| **Termeni si concepte SQL** | **Termeni si concepte MongoDB** |
| Table | Collections |
| Rows  Columns | Key-Value  Pairs, Documents |
| Index | Index |
| Table joins | Embedded Documents and Linking |
| Fixed schema | Schema less |
| Primary Key (explicit) | Primary key (implict) |

Tabel Q: termeni si concepte SQL/NoSQL

**Schema de conversie**

Fiecare baza de date din MongoDB este alcatuita din colectii. Fiecare colectie pastreaza datele sub forma de documente (asemanatoare cu un JSON) care este echivalent cu stocarea sub forma de tabele din SQL. In timp ce o linie contine date in setul de coloane, un document NoSQL are o structura asemanatoare cu JSON care se numeste BSON in limbajul MongoDB. Un exemplu de stocare este dat in Figura E care este echivalent cu o linie dintr-un tabel SQL.



Figura E: Stocare MongoDB [8]

Pentru a intelege mai bine maparile, putem lua un exemplu de o tabela SQL “users” si structura corespunzatoare in MongoDB. Asa cum se vede si in Figura W, fiecare linie din tabelul SQL se transforma intr-un document (Json) si fiecare coloana intr-un camp din MongoDB.



Figura W: Conversie SQL-NoSQL

Relatiile in SQL se realizeaza folosind cheile primare si straine si interogarea acestora cu ajutorul directivei JOIN. Nu exista o astfel de mapare in MongoDB, dar relatiile din NoSQL sunt create folosind documente si linkare lor. Astfel nu pot exista dependente intre comenzi deoarece acest lucru poate duce la comenzi NoSQL care nu vor tine cont de aceste dependente asadar vor aparea erori. Acest lucru afecteaza interogarile SQL care contin mai multe tabele cu directiva “JOIN” astfel ca comenzile SQL trebuie sa contina doar “INNER JOIN” cu directiva “ON” sau “USING” pentru a specifica clar care este campul comun dintre cele doua tabele.

**Solutia propusa**

Pentru rezolvarea problemei de conversie a cererilor SQL in cereri NoSQL s-a folosit Python 3.8.0 impreuna cu Flask 1.1.1 pentru Backend si HTML 5, CSS si Javascript pentru partea de Frontend.

**Frontend**

Proiectul consta intr-o aplicatie web cu o interfata simpla si usor de folosit in care sunt scrise cereri SQL pentru a fi convertite ulterior, in cereri NoSQL. Pagina contine doua ‘textbox-uri’, unul pentru cereri SQL, respectiv unul pentru cele NoSQL. Se pot redacta comenzi doar pe cel destinat limbajului SQL, urmand ca in textbox-ul pentru NoSQL sa fie afisate acelasi comenzi, dar in formatul folosit de MongoDB. De asemenea aplicatia permite si incarcarea unui fisier cu comenzi SQL, pentru a fi convertit urmand ca rezultatul sa fie afisat in pagina web in textbox-ul pentru NoSQL.

Pagina web are in componenta in partea de jos un chenar in care este specificat timpul de conversie a cererilor dorite. Ca atare au fost realizate teste de performanta pe mai multe fisiere de dump SQL.

**Backend**

**Rezultate obtinute**

|  |  |  |
| --- | --- | --- |
| Numar de comenzi din fisierul SQL | Cereri folosite | Timp (secunde) |
| 10 | SELECT, INSERT, CREATE TABLE | 0.0119 |
| 1000 | SELECT, INSERT, CREATE TABLE | 0.0898 |
| 8000 | SELECT, INSERT, CREATE TABLE | 0.4012 |
| 30000 | INSERT | 2.0345 |
| 40000 | SELECT, INSERT, CREATE TABLE | 2.5804 |

Tabel X: Rezultate obtinute pe diferite fisiere de dump de SQL

Conform tabelului X se observa ca aplicatia reuseste sa obtina cererile convertite intr-un timp rezonabil.

**Bibliografie:**

[1] Kumar, Lokesh, Shalini Rajawat, and Krati Joshi. "Comparative analysis of nosql (mongodb) with mysql database." *International Journal of Modern Trends in Engineering and Research* 2.5 (2015): 120-127.

[2] Wu, Chieh Ming, Yin Fu Huang, and John Lee. "Comparisons between mongodb and ms-sql databases on the twc website." *American Journal of Software Engineering and Applications* 4.2 (2015): 35-41.

[3] <http://www.querymongo.com>

[4] <https://klaus.dk/sqltomongodb/>

[5] Documentatia oficiala a bazei de date MongoDB: <https://docs.mongodb.com>

[6] <https://www.w3schools.com/sql.asp>

[7] <https://studio3t.com>

[8] https://code.tutsplus.com/articles/mapping-relational-databases-and-sql-to-mongodb--net-35650

<https://severalnines.com/database-blog/linking-creating-mongodb-joins-using-sql-part-2>

https://www.red-gate.com/simple-talk/blogs/enjoying-joins-in-mongodb/