**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 literatura 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. Principalele diferente dintre acestea sunt 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.

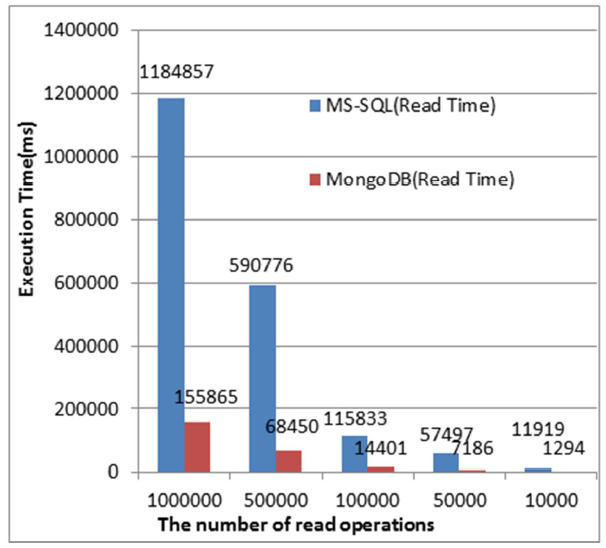
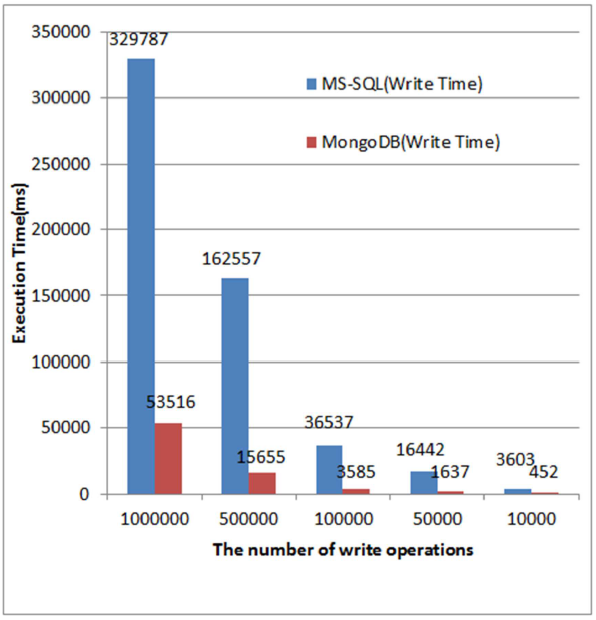


Figura 1: Comparatie timpi executie pentru scriere/citire

**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 are 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 sau [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; | db.getCollection("collection Name").aggregate([{              "$project" : {                  "\_id" : NumberInt(0),                  " condition\_join\_left " : 1.0,                  " condition\_join\_right" : 1.0              }          },          {              "$lookup" : {                  "localField" : " on\_field ",                  "from" : "from table",                  "foreignField" : "on\_field",                  "as" : "alias"              }          },          {              "$unwind" : {                  "path" : "alias",                  "preserveNullAndEmptyArrays" : false              }          }]).sort({‘coloana’:’valoare’}) |

Tabel 1: Instructiunile in limbaj SQL si NoSQL

Termenii folositi in MongoDB sunt prezentati in tabelul 2, 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 2: 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 2 care este echivalent cu o linie dintr-un tabel SQL.



Figura 2: Stocare MongoDB [8]

Pentru a intelege mai bine maparile, putem lua un exemplu de o tabela SQL “users” si structura corespunzatoare in MongoDB. Conform Figurii 3, fiecare linie din tabelul SQL se transforma intr-un document (JSON), iar fiecare coloana devine un camp.



Figura 3: Conversie SQL-NoSQL

Relatiile in SQL se realizeaza folosind cheile primare si straine, iar interogarea acestora cu ajutorul directivei JOIN. Nu exista o astfel de mapare in MongoDB, dar relatiile din NoSQL sunt create folosind documente si linkarea lor. Astfel, nu pot exista dependente intre comenzi deoarece acest lucru poate duce la comenzi NoSQL care nu vor tine cont de aceste dependente, deci vor aparea erori. Acest lucru afecteaza interogarile SQL care contin mai multe tabele cu directiva “JOIN”, astfel incat 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 [13] impreuna cu Flask 1.1.1 [14] pentru Backend si HTML 5, CSS, JQuerry [15] 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 aceleasi 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**

Fiecare comanda primita va fi separata in cuvinte in functie de spatii folosind functia “split” din Python. Primul cuvant din lista de cuvinte va reprezenta comanda in sine. Comenzile acceptate de catre aplicatie sunt: “Select”, “Drop”, ‘’Create”, “Delete”, “Insert”, “Show” si “Use”. Daca este introdusa o comanda care nu se afla prin cele enumerate mai sus programul va intoarce un mesaj de eroare in care se specifica faptul ca acea comanda este gresita (daca nu se afla in lista, se presupune automat gresita). Sintaxa instructiunilor este asemanatoare cu cea din [12].

In functie de comanda primita se va crea un obiect specific acelei comenzi pentru a putea fi convertita in format NoSQL pentru MongoDB. Fiecare comanda este diferita si are nevoie de o abordare diferita. In continuare se vor prezenta metodele de conversie pentru fiecare comanda acceptata.

**Drop**

Comanda DROP se foloseste insotita de cuvantul cheie “table”. Instrucțiunea “DROP TABLE” este utilizata pentru a renunța la un tabel existent într-o bază de date. Echivalentul NoSQL al acestei comenzi este “db.table\_name.drop();”.

**Insert**

Instrucțiunea INSERT INTO este utilizată pentru a insera înregistrări într-o tabela. In NoSQL este asemanator ca in SQL deoarece pentru fiecare coloana dorita din tabel se pune valoarea aleasa. Spre deosebire de SQL, in NoSQL trebuie specificate si coloanele care se afla in tabel pentru a se putea realiza convertirea fara erori a instructiunii de inserare.

**Delete**

Instrucțiunea DELETE este utilizată pentru a șterge înregistrările existente dintr-un tabel. La aceasta se pot adauga conditii cu intructiunea WHERE. Pentru convertirea conditiilor in NoSQL, acestea au fost transformate in forma postfixata pentru a putea fi respectate parantezele. Fiecare conditie este considerata operand (va fi inlocuita cu o litera mica), iar cuvintele cheie AND si OR sunt operatori. Astfel pentru a converti o conditie simpla precum “WHERE id > 5 AND (varsta > 10 OR nume LIKE (“Ana”, “Maria”))” va arata ca in expresia (1) si se transforma in forma postfixata la fel ca in expresia (2).

Dupa ce conditiile sunt transformate in forma postfixata, se convertesc in format NoSQL pentru MongoDB. In tabelul 3 sunt prezentate conversiile semnelor din cadrul fiecarei condtiii.

|  |  |
| --- | --- |
| **SQL** | **NoSQL (MongoDB)** |
| > / >= | $gt / $gte |
| < / <= | $lt / $lte |
| = / != | - / $ne |
| in / is / not | $in / - / $ne |

Tabel 3: Echivalentul operatorilor de relationare in NoSQL

**Select**

Instrucțiunea SELECT este utilizată pentru a selecta date dintr-o bază de date. Datele returnate sunt stocate într-un tabel de rezultate, numit set de rezultate. In NoSQL trebuie stabilit daca instructiunea SELECT contine GROUP BY sau JOIN. In cazul in care aceasta nu contine nicuna din cele doua instructiuni de mai sus, se foloseste db.table\_name.find() in care se afla o lista. In aceasta lista se adauga coloanele dorite intr-un JSON si conditiile din sectiunea WHERE in alt JSON.

In cazul in care exista instructiunea GROUP BY, este necesara folosirea comenzii db.table\_name.group() care contine un JSON, asa cum se specifica in documentatie [5]. Aici trebuie specificate, la cheia ‘key’, coloanele pe care se face gruparea, iar condiitile se trec la cheia ‘cond’, la fel ca in cazul instructiunii fara GROUP BY. De asemenea, daca exista functii (sum, avg, count), este necesara adaugarea cheii ‘initials’ pentru a initializa variabile (deobicei cu 0) pentru stocarea rezultatului functiei. Functiile vor fi transformate in limbaj NoSQL pentru MongoDB si vor fi adaugate la cheia “reduce” in JSON asa cum este prezentat si in [11].

In cazul in care instructiunea SELECT contine JOIN, este necesara folosirea comenzii NoSQL “db.getCollection(table\_name).aggregate()”, asa cum se specifica in [9]. Spre deosebire de SQL, in NoSQL trebuie specificat campul pentru conditia de JOIN, astfel fiind necesar ca in cererea SQL sa fie specificat prin instructiunea ON ca si in [10]. In interiorul comenzii ‘aggregate’ trebuie adaugate instructiunile ‘lookup’ si ‘unwind’. Prima trebuie sa contina campul din primul table, ‘localField’, tabelul propriu-zis ‘from’, campul din al doilea tabel (cel pe care se face JOIN), ‘foreignField’ si aliasul ‘as’. In instructiunea ‘unwind’ se va specifica calea catre al doilea tabel ‘path’ si un boolean setat pe fals, 'preserveNullAndEmptyArrays', astfel incat dacă calea este nulă, lipsește sau tabelul este gol, ‘unwind’ nu emite un document.

De asemenea, orice cerere SQL de SELECT poate contine instructiunea ORDER BY, aceasta se adauga la sfarsitul cereri NoSQL. Astfel ca indiferent ce fel de cerere avem (cu sau fara Join sau grupare), aceasta se adauga astfel db.table\_name.metoda(...).sort({camp: 1/-1}). Campul reprezinta coloana pe care se doreste sortarea, iar valorile 1 sau -1 reprezinta metoda crescatoare, respectiv descrecatoare de sortare. Tot la sfarsitul instructiunii NoSQL se poate specifica si cate valori sa fie returnate prin instructiunea .limit(<valoare>). Valoarea respectiva poate fi cuprinsa in intervalul [0, inf), iar daca este trecuta pe 0 nu va fi luata in considerare.

Comenzile **SHOW** si **USE** se folosesc identic atat in SQL, cat si in NoSQL (MongoDB), fara a fi necesara convertirea lor.

**Rezultate obtinute**

|  |  |  |
| --- | --- | --- |
| Numar comenzi 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 4: Rezultate obtinute pe diferite fisiere de dump de SQL

Conform tabelului 4, se observa ca aplicatia reuseste sa obtina cererile convertite intr-un timp rezonabil.

**Concluzii**

Sistemul relațional de gestionare a bazelor de date nu va fi suficient pentru aplicatiile mari de baze de date, deoarece au un dezavantaj pentru acestea, cum ar fi viteza de timp si de executie si scalabilitatea interogarilor. Dar necesarul de stocare pentru noua generatie de aplicatii, este foarte diferit de aplicațiile anterioare. Putem alege NoSQL (MongoDB) in loc de MySQL datorita a doi factori: ușurinta de utilizare si performanta de sincronizare. Astfel, prin intermediul acestei aplicatii web, orice utilizator poate converti un numar mare de comenzi din format SQL in NoSQL (MongoDB).

**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

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

[10] <https://www.red-gate.com/simple-talk/blogs/enjoying-joins-in-mongodb/>

[11] <https://kb.objectrocket.com/mongo-db/mongodb-group-by-multiple-fields-using-aggregation-function-464>

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

[13] Python Software Foundation. Python Language Reference, version 3.8. Available at [http://www.python.org](http://www.python.org/)

[14] Miguel Grinberg. 2014. Flask Web Development: Developing Web Applications with Python (1st. ed.). O’Reilly Media, Inc.

[15] https://web.archive.org/web/20110708015734/http://alan.blog-city.com/jquerylinedtextarea.htm