

Tema2

Lost in NoSQL

| | |
|-------------------|--|
| Responsabili Tema | Marilena Panaite, Andreea Marica, Daniel Nicolescu |
| Data Publicării | 14 noiembrie 2018 |
| Deadline | 8 decembrie 2018 (ora 22:00) După deadline se acceptă teme până la data de 15 decembrie cu depunere de 1 punct/zi |

1) Introducere

Scopul acestei teme este de a simula implementarea unei baze de date NoSQL [1], distribuită pe mai multe noduri numita **LNoSQL**, ce suportă un limbaj simplu de interogare. Mai multe detalii despre cum anume este implementată această bază de date și tipurile de comenzi pe care le suportă sunt prezentate în secțiunea de cerințe.

Bazele de date relaționale sunt folosite în implementarea multor aplicații web, bancare și reprezintă baza multor implementări ale modului de stocare a datelor. Putem alege între două implementări generice în funcție de modelarea datelor: baze de date relaționale ce folosesc limbajul MySQL [2] sau baze de date non-relaționale ce pot avea diverse implementări (document, key sau column).

2) Cerințe

a) Descriere Lost in NoSQL

Baza de date non-relațională numită “*Lost in NoSQL*” este folosită pentru a stoca eficient date sub formă de entități ce vor fi repartizate pe mai multe noduri **N**. Vor putea fi stocate mai multe tipuri de entități caracterizate de atribute, dintre care unul va fi considerat *cheie primară* (după care va fi identificată unic o valoare a entității stocate în tabel).

Baza de date LNoSQL trebuie să asigure și redundanța datelor, ceea ce înseamnă că datele vor putea fi disponibile indiferent de câte noduri sunt up la un moment dat. În acest sens vor fi implementate următoarele funcționalități:

- Fiecare *entitate* va avea o caracteristică ce poate fi configurată la crearea numită *factor de replicare* (RF), ce va fi mai mică sau egală cu numărul de noduri **N** ($RF \leq N$). Acest factor reprezintă numărul de copii ale instanțelor entităților pe fiecare dintre nodurile bazei de date (de exemplu, dacă avem o bază de date cu $N = 5$ noduri și entitatea Produs are $RF = 3$ atunci orice intrare a entității Produs va trebui scrisă în 3 din cele 5 noduri ale bazei de date).
- Un nod al bazei de date are capacitate limitată (*MaxCapacity*) și dacă se atinge această limită scrierea se va face în următorul nod care este liber (nu și-a atins capacitatea maximă).

- Un nod al bazei de date poate sa conțină entități de toate tipurile declarate și nu este constrâns în a stoca un singur tip de entitate.
- Fiecare entitate din baza de date va fi caracterizată de mai multe atribute ce pot fi de tipul *String*, *Integer* sau *Float*.
- Ordinea de scriere în nodurile bazei de date se va face începând cu cel mai ocupat nod. Dacă două noduri au același grad de ocupare se va alege cel cu id-ul mai mic.
- Fiecare entitate are o cheie primară după care se va identifica unic fiecare instanță și va fi întotdeauna primul atribut al entității la creare.
- Pentru fiecare instanță a unei entități din baza de date se va ține cont de data la care s-a făcut ultimul update (se va folosi pentru aceasta timestamp-ul local al mașinii).

b) Limbajul de Interogare – LQL

Ca orice baza de date, fie SQL sau NoSQL, va trebui sa suporte un set de comenzi pentru crearea entităților, inserarea de instanțe ale entităților, actualizare și ștergere. Pentru baza de date LNoSQL acestea vor fi:

i) Creare Baza de date:

- Comanda prin care se creează baza de date cu constrângerile menționate de număr de noduri și capacitate maximă a unui nod. În exemplele și testele din tema va exista o singură baza de date.
- Comanda:

CREATEDB <Db_Name> <No_Nodes> <Max_Capacity>

ii) Creare Entitate:

- Comanda prin care se creează o entitate cu un factor de replicare și atributele sale care pot fi de tipul: *String*, *Integer* sau *Float*. Ordinea atributelor din comanda de creare va fi reflectată și în celelalte comenzi.
- Primul atribut va fi întotdeauna considerat **cheie primară** – după care o instanță a unei entități va fi identificată unic.
- Comanda:

CREATE <Entity> <RF> <No_Attributes>
Attribute_1 Tip_Atribut1
Attribute_2 Tip_Atribut2
....
Attribute_n Tip_Atributn

iii) Inserare instanță:

- Comanda prin care se inserează o instanță a unei entități. Operația înseamnă replicarea instanței pe nodurile bazei de date în funcție de RF specific entității.
- Inserarea într-un nod se va face ordonat după cea mai recentă intrare.
- Comanda:

INSERT <Entity> <Val_Attr1> <Val_Attr2> ... <Val_Attrn>

iv) Ștergere instanță:

- Vor fi șterse toate copiile instanței respective de pe toate nodurile în care a fost inserată
- Val_attr1 va reprezenta valoarea cheii primare pentru acea instanță
- În cazul în care nu exista va fi printat un mesaj de eroare: „NO INSTANCE TO DELETE”
- Comanda:

DELETE <Entity> <Primary_Key>

v) Actualizare instanță:

- Actualizează toate copiile instanței respective de pe toate nodurile în care a fost inserată
- Comanda:

UPDATE <Entity> <Primary_Key> <Attr1> <New_Val_Attr1> ... <Attrn> <New_Val_Attrn>

vi) Regăsire instanță

- Returnează o instanță cu toate valorile și nodurile în care se află.
- Dacă nu exista acea instanță se va returna un mesaj de eroare „NO INSTANCE FOUND”
- Formatul pentru printarea valorilor de tip float se va fi folosi DecimalFormat cu formatul #.##
- Comanda:

GET <Entity> <Primary_Key>

- Rezultat:
<Node_1> <Node_2>...<Node_n> <Entity> <Attr1>:<Val_Attr1> ... <Attrn>:<Val_Attrn>

vii) Print baza de date

- Printează întreaga baza de date împreună cu toate nodurile și datele ce se regăsesc la acel moment în fiecare nod.
- În cazul în care baza de date nu conține nicio instanță a unei entități se va afișa mesajul de eroare „EMPTY DB”
- Comanda:

SNAPSHOTDB

- Rezultat:

*<Nod_1>
<ENTITY1> <Attr1>:<Val_Attr1> ... <Attrn>:<Val_Attrn>
<ENTITY2> <Attr1>:<Val_Attr1> ... <Attrn>:<Val_Attrn>
...
<Nod_2>
<ENTITY1> <Attr1>:<Val_Attr1> ... <Attrn>:<Val_Attrn>
...*

Bonus:

i) Clean-up

- Comanda are ca efect ștergerea instanțelor mai vechi de un anumit timestamp (nanosecunde) dat ca parametru.
- De asemenea, trebuie păstrată în continuare ordinea de scriere în noduri după cele mai recente intrări.
- Comanda:

CLEANUP <DB_NAME> <TIMESTAMP(ns)>

ii) Full Database

- Implementarea unui mecanism prin care se va scala baza de date în cazul în care este umplută la capacitate maximă (nu mai este loc de inserare în niciun nod)

- In acest caz, va fi creat un nou nod de fiecare data cu aceeași capacitate <Max_Capacitate> ca a celorlalte noduri.

2) Format date de intrare si de ieșire

Datele de intrare se vor citi tot timpul dintr-un fișier ce va fi primit ca parametru la rulare <input>.

Datele de ieșire se vor scrie in fișierul <input>_out si vor conține pe fiecare linie rezultatul comenzilor citite de la input inclusiv mesajele de eroare. In cazul de fata vor avea rezultate comenzile:

- *SNAPSHOTDB*
- *GET <Entity> <Primary_Key>*

| Date de intrare – input | Date de iesire – output |
|---|--|
| CREATEDB Tema2 5 3 | Nod1 |
| CREATE Produs 3 3 Id Integer Nume String Pret Float | Produs Id:2 Nume:Paine Pret:1.5 Produs Id:1 Nume:Apa Pret:5.2 |
| INSERT Produs 1 Apa 5.2 | Nod2 |
| INSERT Produs 2 Paine 1.5 | Produs Id:2 Nume:Paine Pret:1.5 |
| SNAPSHOTDB | Produs Id:1 Nume:Apa Pret:5.2 |
| UPDATE Produs 2 Pret 2.4 | Nod3 |
| GET Produs 2 | Produs Id:2 Nume:Paine Pret:1.5 |
| GET Produs 1 | Produs Id:1 Nume:Apa Pret:5.2 |
| DELETE Produs 1 | Nod1 Nod2 Nod3 Produs Id:2 Nume:Paine Pret:2.4 |
| GET Produs 1 | Nod1 Nod2 Nod3 Produs Id:1 Nume:Apa Pret:5.2 |
| SNAPSHOTDB | NO INSTANCE FOUND |
| DELETE Produs 3 | Nod1 |
| | Produs Id:2 Nume:Paine Pret:2.4 |
| | Nod2 |
| | Produs Id:2 Nume:Paine Pret:2.4 |
| | Nod3 |
| | Produs Id:2 Nume:Paine Pret:2.4 |
| | NO INSTANCE TO DELETE |

3) Constrângeri de implementare

Pentru aceasta tema se vor folosi doar implementările puse la dispoziție de clasa *Collections* [3] din Java si nu alte structuri de date. De asemenea, algoritmii de sortare folosiți in tema vor fi cei deja implementați in JDK si nu se vor accepta implementări proprii ale sortării. Mai multe detalii despre implementările din Collections gasiti la [4].

4) Testare

Tema va fi testata pe platforma *vmchecker* folosind testele puse la dispoziție împreună cu enunțul temei (pentru checker va trebui sa existe clasa *Tema2* cu metoda *main*). Testele vor fi grupate astfel încât se va putea obține punctaj parțial doar pe implementarea unor comenzi.

5) Punctaj

Punctajul pe tema va fi obtinut dupa cum urmeaza:

- 90% teste automate
- 10% codestyle, JavaDoc si README
- Code inspection (-100%)

- [1] <https://en.wikipedia.org/wiki/NoSQL>
- [2] <https://en.wikipedia.org/wiki/SQL>
- [3] <https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>
- [4] <https://docs.oracle.com/javase/tutorial/collections/implementations/summary.html>
- [5] <https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html>