Project Report on

Sql2Neo: Interconversion of SQL, NoSQL and Neo4j formats

Submitted by

Aayush Jain (16IT101)

Aditi Rao (16IT103)

Under the Guidance of

Shruti J R

Department of Information Technology, NITK Surathkal

Date of Submission: 11 June 2020

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Insitute of Technology Karnataka, Surathkal

June 2020

Abstract

Insert abstract here.

Contents

1 Introduction

Intro here

2 Literature Survey

2.1 Related Work

Literature survey

2.2 Outcome of Literature Survey

Base paper is [1].

2.3 Problem Statement

Problem Statement

2.4 Objectives

- O1
- O2

3 Requirement Analysis

Sql2Neo is intended to be a command line tool that enables interconversion between SQL, NoSQL and Neo4j formats.

3.1 Functional Requirements

F1. SQL databases must be fully converted to a Neo4j database. This includes indexes, constraints, records and relationships.

F2. NoSQL databases must be fully converted to a Neo4j database. This includes constraints (inferred) and records.

F3. SQL queries must be translated to a Neo4j query to provide interoperability between databases.

3.2 Non-Functional Requirements

NF1. Convert NoSQL database to intermediate SQL format to provide compatibility.

NF2. Manage data loading from databases in order to prevent memory overuse and overflow errors.

NF3. Provide Sql2Neo as an isolated tool (with packaged dependencies) to improve usability.

Most software and hardware requirements intersect with the implementation tools, as discussed in section 6.

4 System Design/Architecture

Architecture here

5 Methodology

5.1 Converting SQL-databases to Neo4j format

The first step to convert an SQL database is to extract attribute details of each table. This data is available in the `information_schema` of the database. The data for each attribute answers the following questions:

- should this attribute be *indexed*?
- does this attribute have a *uniqueness constraint*?
- is this attribute a *foreign key reference* to another table's attribute?

Algorithm 1: Extract attribute details of SQL database

Input: R, a relational database

Output: AS, an attribute set containing details of each attribute

```
AS ← φ;                                          /* empty map */
foreach table in information_schema(R) do
    AS[table] ← φ;                               /* empty map */
    foreach attr in table do
        AS[table][attr] ← {index, unique, fk};
    end
end
return AS;
```

The index and uniqueness constraint data enable the Neo4j setup to be as closely modelled to the relational one. Furthermore, indexing on attributes is maintained across systems and rigid constraints are also satisfied while inserting in the Neo4j database.

Since Neo4j stores data as JSON documents, it does not define a rigid and formal schema. Owing to this property, indices and constraints can be created before the data is actually inserted. In Neo4j, index creation is not idempotent, meaning that creating the index twice reults in an error. Additionally, constraints implicitly create an index on the specified attribute (much like relational databases), thus constraints are applied only on those attributes that are not indexed.

Algorithm 2: Create indices and constraints in Neo4j

```
Input: AS, an attribute set containing details of each attribute
Result: G, a graph database with applied indices and constraints
foreach table in AS do
    foreach attr in table do
        if attr must be indexed then
            CreateIndex(G, table, attr);
        else if attr has constraint but not indexed then
            /* if attr is indexed, then it meets uniqueness constraint
            */
            CreateUniquenessConstraint(G, table, attr);
    end
end
```

In terms of Cypher Query Language (CQL), `CreateIndex(G, table, attr)` is equivalent to

```
CREATE INDEX index_name ON:table(attr);
```

`CreateUniquenessConstraint(G, table, attr)` is equivalent to

```
CREATE CONSTRAINT constraint_name ON (t:table) ASSERT t.attr IS
    ↪ UNIQUE;
```

Conversion of a table's records to Neo4j nodes is a fairly straightforward task. A naive approach is followed where each tables records are converted to a node. This implies that certain tables that behave purely as relationships are also converted to nodes instead of being retained into Neo4j.

`CreateNewNode(G, table, record)` is equivalent to

```
CREATE (t:table $record);
```

provided that `record` is a map of key-value pairs (assumed as parameter).

Finally, relationship conversion is performed. This step takes the foreign key relations from each table and maps them to a relation in Neo4j. This also means that the semantics of the relation are lost since the edge (in Neo4j) does not provide actual data. Rather it must be inferred from the database.

Algorithm 3: Populate the graph database with records

```
Input: R, a relational database
Result: G, a populated graph database
foreach table in R do
    foreach record in table do
        CreateNewNode(G, table, record);
    end
end
```

Algorithm 4: Create relationships between nodes in the graph data

```
Input: AS, an attribute set containing details of each attribute
Result: G, a graph database with relationships
foreach table in AS do
    foreach attr in table do
        if attr is foreign key then
            fk_table, fk_attr ← GetFKReference(attr);
            CreateRelationship(G, table, attr, fk_table, fk_attr);
        end
    end
end
```

`CreateRelationship(G, table, attr, fk_table, fk_attr)` is equivalent to

```
MATCH (a:table), (b:fk_table) WHERE a.attr = b.fk_attr CREATE (a)
  ↪ -[r:relationship_name]-> (b);
```

6 Implementation

The proposed approah was implemented on a system running Ubuntu 20.04 LTS. The following additional software is used:

- Python 3.8.2
  - mysql-connector-python 8.0.20
  - py2neo 4.3.0
  - pymongo 3.10.1
  - python-dotenv 0.13.0
  - pandas 1.0.4
- MySQL v8.0.20 for testing SQL databases
- MongoDB v3.6.8 for testing NoSQL databases
- Neo4j 4.0.5

7 Results and Analysis

Results

8 Conclusion and Future Works

Conclusion

References

[1] M. Singh and K. Kaur, “Sql2neo: Moving health-care data from relational to graph databases,” in *2015 IEEE International Advance Computing Conference (IACC)*, pp. 721–725, 07 2015.