Inner Workings Behind Babel

[Conversion Specifications]

Frank Roetker
Rose-Hulman Institute of Technology
5500 Wabash Ave.
Terre Haute, IN, USA
roetkefj@rose-hulman.edu

Jordon Phillips
Rose-Hulman Institute of Technology
5500 Wabash Ave.
Terre Haute, IN, USA
phillijk@rose-hulman.edu

Ricky Shomer
Rose-Hulman Institute of Technology
5500 Wabash Ave.
Terre Haute, IN, USA
shomerrt@rose-hulman.edu

ABSTRACT

In the world of database management systems, relational databases are dominate. This isn't too surprising since a relational database can be used to solve many problems. However, using a relational database as a solution is not always the most efficient answer.

Babel seeks to alleviate the problem of a company being locked-in to using a RDBMS. A company isn't going to throw away their data stored in a RDBMS to restart with a graph database even if it might speed up their query times. However, if given a tool that was to fully export their data to a Neo4j system, they might.

Keywords

Neo4j, Cypher

1. INTRODUCTION

In the world of database management systems, relational databases are dominate. This isn't too surprising since a relational database can be used to solve many problems. However, using a relational database as a solution is not always the most efficient answer.

NoSQL has gained a footing in the database domain, solving problems in a different light than Relational databases. Neo4j, a NoSQL graph database takes a much different stance when viewing the world. "Reality is a graph"[1] is the mantra that Neo Technology takes when looking at data. This may be a greater fit for solving problems that involve deeply connected data which in a RDBMS would be distributed across many tables and relying on many foreign keys.

Babel seeks to alleviate the problem of a company being locked-in to using a RDBMS. A company isn't going to throw away their data stored in a RDBMS to restart with a graph database even if it might speed up their query times. However, if given a tool that was to fully export their data to a Neo4j system, they might.

2. BABEL

2.1 MS SQL data

The first step in the process of converting a MS SQL database to another database is defining the SQL queries to find all correct data from the MS SQL database. This is actually not as much of an issue; MS SQL Server keeps much of this data for you.

In a relational database, foreign key tables are commonly used to join multiple tables together. This is the equivalent to relationships in our graph database. So our query needs to include foreign keys, their tables, and the tables/columns that they are constrained to:

```
SELECT INFO.TABLE_NAME, INFO.COLUMN_NAME,
       FK.FKTABLE_NAME, FK.FKCOLUMN_NAME
FROM

(SELECT TABLE_NAME, COLUMN_NAME
        FROM INFORMATION_SCHEMA.COLUMNS) AS
             INFO

LEFT OUTER JOIN

(
 SELECT C.TABLE_NAME [TABLE_NAME],
        KCU.COLUMN_NAME [COLUMN_NAME],
        C2.TABLE_NAME [FKTABLE_NAME],
        KCU2.COLUMN_NAME [FKCOLUMN_NAME]
  FROM  INFORMATION_SCHEMA.
      TABLE_CONSTRAINTS C
        INNER JOIN INFORMATION_SCHEMA.
            KEY_COLUMN_USAGE KCU
          ON C.CONSTRAINT_SCHEMA = KCU.
              CONSTRAINT_SCHEMA
            AND C.CONSTRAINT_NAME = KCU.
                CONSTRAINT_NAME
        INNER JOIN INFORMATION_SCHEMA.
            REFERENTIAL_CONSTRAINTS RC
          ON C.CONSTRAINT_SCHEMA = RC.
              CONSTRAINT_SCHEMA
            AND C.CONSTRAINT_NAME = RC.
                CONSTRAINT_NAME
```

```
            INNER JOIN INFORMATION_SCHEMA.
                TABLE_CONSTRAINTS C2
              ON RC.UNIQUE_CONSTRAINT_SCHEMA =
                  C2.CONSTRAINT_SCHEMA
                AND RC.UNIQUE_CONSTRAINT_NAME =
                    C2.CONSTRAINT_NAME
            INNER JOIN INFORMATION_SCHEMA.
                KEY_COLUMN_USAGE KCU2
              ON C2.CONSTRAINT_SCHEMA = KCU2.
                  CONSTRAINT_SCHEMA
                AND C2.CONSTRAINT_NAME = KCU2.
                    CONSTRAINT_NAME
                AND KCU.ORDINAL_POSITION = KCU2
                    .ORDINAL_POSITION
 WHERE  C.CONSTRAINT_TYPE = 'FOREIGN_KEY'
) AS FK

ON FK.TABLE_NAME = INFO.TABLE_NAME
    AND FK.COLUMN_NAME = INFO.COLUMN_NAME
ORDER BY INFO.TABLE_NAME, INFO.COLUMN_NAME
```

This consolidation of data is what we are going to need to consider when designing the algorithm for distinguishing potential nodes from relationships.

2.2 Conversions

This section will explain how Tables are going to be converted for Neo4j. These are ordered in order of complexity, not number of Foreign keys.

In the Relational database world, data is organized in Tables (or Relations). These tables are made up of Columns (*Attributes*) and Rows (or Tuples). There is also the notion of Foreign keys. These allow data from one table to be consistent with data from another - allowing the data in tables to be joined.

Graph databases on the other hand join data in a much more organic way. The idea of Tuples is thrown out for Nodes. These nodes have Properties; properties are similar to columns. As for the graph relative to Foreign keys, we have Relationships. However, Relationships are much more than just a Foreign key: they physically connect Nodes to each other. Like Nodes, Relationships also can also have properties.

3. CONCLUSIONS

4. ACKNOWLEDGMENTS

We would like to thank Dr. Sriram Mohan for his help during the start of this project. Without his guidance, we would not have achieved our final product.

5. REFERENCES

[1] N. Technology. The neo4j manual v2.0.0-m02. May 2013.