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[1].

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.

In this guide, I will be using an Urban Exploration database as an example. You don't need to have an understanding of what the database will represent or what the data means to understand how the conversions are going to be handled.

After running our SQL query on the database, we will be presented with the results that are in Figure 2. After viewing these results, it may become more clear how the data is going to be divided up.

2.2.1 0 Foreign Keys

In the event that the table in question has 0 foreign keys, we would consider it a "Base Table". This table does not rely on any other information stored in any other table and thus is an entity in and of itself.

[1]These are ordered in order of complexity, not number of Foreign keys

We treat tuples from this type of table as a Node in regards to Neo4j. This is the simplest conversion between MS SQL and Neo4j, because it doesn't rely on any other data already existing in the database. This conversion will make a new node with Properties for each Attribute owned by the Tuple.

2.2.2 2 Foreign Key

2.2.3 1 Foreign Key

2.2.4 3 Or More Foreign Key

Figure 2: Table Info from SQLQuery1

TABLE_NAME	COLUMN_NAME	FKTABLE_NAME	FKCOLUMN_NAME
Comment	Comment	NULL	NULL
Comment	POID	Point of Interest	Id
Comment	PUsername	Person	username
Comment	TimeStamp	NULL	NULL
Friend	FriendName	Person	username
Friend	PUsername	Person	username
Guild	Description	NULL	NULL
Guild	Name	NULL	NULL
Guild	PUsername	NULL	NULL
Membership	GName	Guild	PUsername
Membership	PUsername	Person	username
Owns	POID	Point of Interest	Id
Owns	PUsername	Person	username
Person	Name	NULL	NULL
Person	Password	NULL	NULL
Person	TimeStamp	NULL	NULL
Person	username	NULL	NULL
Point of Interest	Code	NULL	NULL
Point of Interest	Description	NULL	NULL
Point of Interest	Id	NULL	NULL
Point of Interest	Latitude	NULL	NULL
Point of Interest	Longitude	NULL	NULL
Point of Interest	Name	NULL	NULL
Point of Interest	TimeStamp	NULL	NULL
Rate	POID	Point of Interest	Id
Rate	PUsername	Person	username
Rate	Rating	NULL	NULL
Visit	POID	Point of Interest	Id
Visit	PUsername	Person	username

3. CONCLUSIONS

3.1 Future Additions

Babel has a number of additions planned for the future. Babel, being written for Neo4j v2.0.0-M02, makes use of the new features in store for the next release of Neo4j. These features are not all fully developed, and thus Babel is limited in many ways.

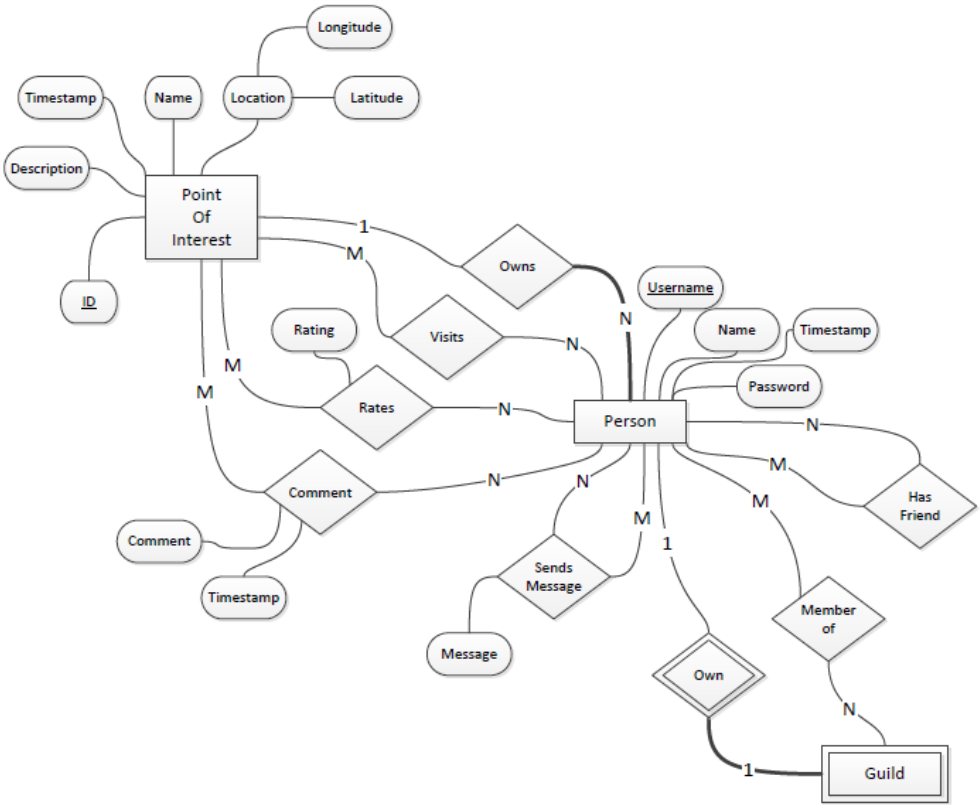

Figure 1: Urban Exploration Entity Relationship Diagram

First, the Rest API that is provided for Neo4j has the option for nodes to be inserted in a bulk transaction. This greatly increases the speed of inserts, because one transaction is much faster than having to wait for as many transactions as there are tuples. Babel does not currently use this option, due to the fact that nodes do not yet have the ability to gain a label in the Rest API for bulk inserts.

Relationships are expensive actions. Babel currently queries the nodes added to the database for their address before creating a relationship between them. This would mean that relationships take at most 3 transactions before they are created. The most obvious option to move forward from where we are right now would be to cache the location of the nodes after inserting them into Neo4j. This would reduce relationship creation to one transaction. From there, we could even use a bulk insert for relationships, reducing their transaction count to even less.

Another plan that is in store for Babel is continuous updates from the MS SQL Server. Babel currently only supports running once per Neo4j database; if you want to update a Neo4j database with new information that was added to MS SQL Server, you will need to purge the database and then rerun Babel. Forgoing this issue and allowing users to run Babel as many times as they want without having to purge their databases is of great importance.

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.