Conversion of a Legacy Proprietary Backend to an Open-Source Architecture Database Model

GEOG 574, Final Project Report

Bradley Andrick

May 4th, 2017

Abstract

This report compares the results of open-source database technologies vs. proprietary databases to determine if there is any significant difference in speed of queried result sets. The databases are specifically designed to handle spatial data and benchmarks are tested using both spatial and non-spatial queries, with an emphasis on the former. The comparison utilizes PostGRES and Microsoft(MS) SQL Server as the database engines while keeping other variables constant as to provide the best indication of Database Management System (DBMS) performance. Design of the spatial database used for testing mirrors the actual structure of a database utilized by a typical vegetation management company with an emphasis on recreating as close to real world design practices as possible.

Introduction and Background

Vegetation management can be used throughout the development of a forest stand and provides one of the best opportunities to ensure that stands of the desired species composition and structure develop within an economically feasible period of time[1]. Traditionally a niche industry, vegetation management has been slow to adopt technological advances. The technology that is used usually deals with the process of the physical management itself, leaving little time devoted to enhancing the planning process that must occur before utility right-of-way's can be managed. As the industry migrates from paper planning to digital archiving, the need arises for optimization. Today, there are several options to choose from when selecting a new data collection platform, however, most of these are rooted in proprietary and expensive technologies. This project aims to look at the alternatives that exist in the open source sector and complete a performance comparison between a selected open source utility and a commonly used proprietary option. The data collection result that is needed from the backends requires spatial awareness, therefore database engines with a spatial component will be utilized.

In today's market many databases with spatial components exist, including: PostGRES/PostGIS, MS SQL Server, Oracle Spatial, MySQL Spatial, SQLITE/SpatiaLite, CouchDB/Geocouch. For the purposes of this comparison, two of the more mainstream databases were selected: PostGRES and MS SQL Server. Since POSTGRES/POSTGIS was a major focus in course (Geog 574) it was a good choice since the knowledgebase and resources were readily available. MS SQL Server is currently being utilized in at least one known vegetation management solution and therefore gave a good model for what needs the industry might have.

A major factor in IT management decisions when it comes to database management is the possibility of commonly used products that focus on different industries that can be utilized together. ArcGIS, the geospatial systems sold by Esri, holds a large market share and therefore should be taken into consideration when deciding upon a spatial DBMS. ArcGIS currently supports: Oracle, Microsoft SQL Server, IBM DB2, IBM Informix and PostgreSQL.[2]. Since both PostgreSQL and Microsoft SQL Server are among the possible backends, it makes this comparative analysis valuable to anyone needing to make the decision between sticking with one of the proprietary options or choosing the only open source DBMS supported by Esri's enterprise geodatabase technology.

Methods

Data Background:

The test data for this project came from a demo user database sample created by a vegetation management company. Before use, any information that was non-generic was removed to preserve data structure while maintaining testing capabilities. The fields and tables that remain are robust enough to allow for adequate comparison testing. While many tables were removed, the number of fields in the LOCATION and UNITS tables were left fairly large – which is very different from what was used as demo data during course labs. The challenges associated with become particularly apparent in the model design phases that follow.

Model Design:

a) Conceptual Model: ER Diagram

The model design phase involved reviewing the available demo dataset and designing a model that kept some of the complexities of the sample structure while being simplified enough to fit the scope of this project. The ER diagram was then built using ERDPlus[2]. Figure 1 shows the final ER diagram detailing the entities, relationships, attributes, and cardinality for the database. Crow's Foot Notation was used to show cardinality and the standard ER representations of entities as rectangles, relationships as diamonds, and attributes as ovals was used. Note that the number of attributes made it difficult to visualize the LOCATION and UNIT tables in this manner.

b) Logical schema diagram mapped by the conceptual model

Figure 2 and 2a show the logical schema diagram for the database model. The traditional format was used however, the LOCATION and UNITS tables contained too many attributes to fit within the formatting and had to be appended in the second figure (figure 2a).

Database Implementation:

The database implementation took place in two parts. The first was in the MS SQL Server environment and the second was within a PostGRES/PostGIS environment. Before either of the databases were built as modeled, a backup file was used to restore the sample database into a new MS SQL Database. This served as a point of reference on which to base the model within MS SQL. Data from the restored database was then exported into a new database that would serve as

(continued on page 6)

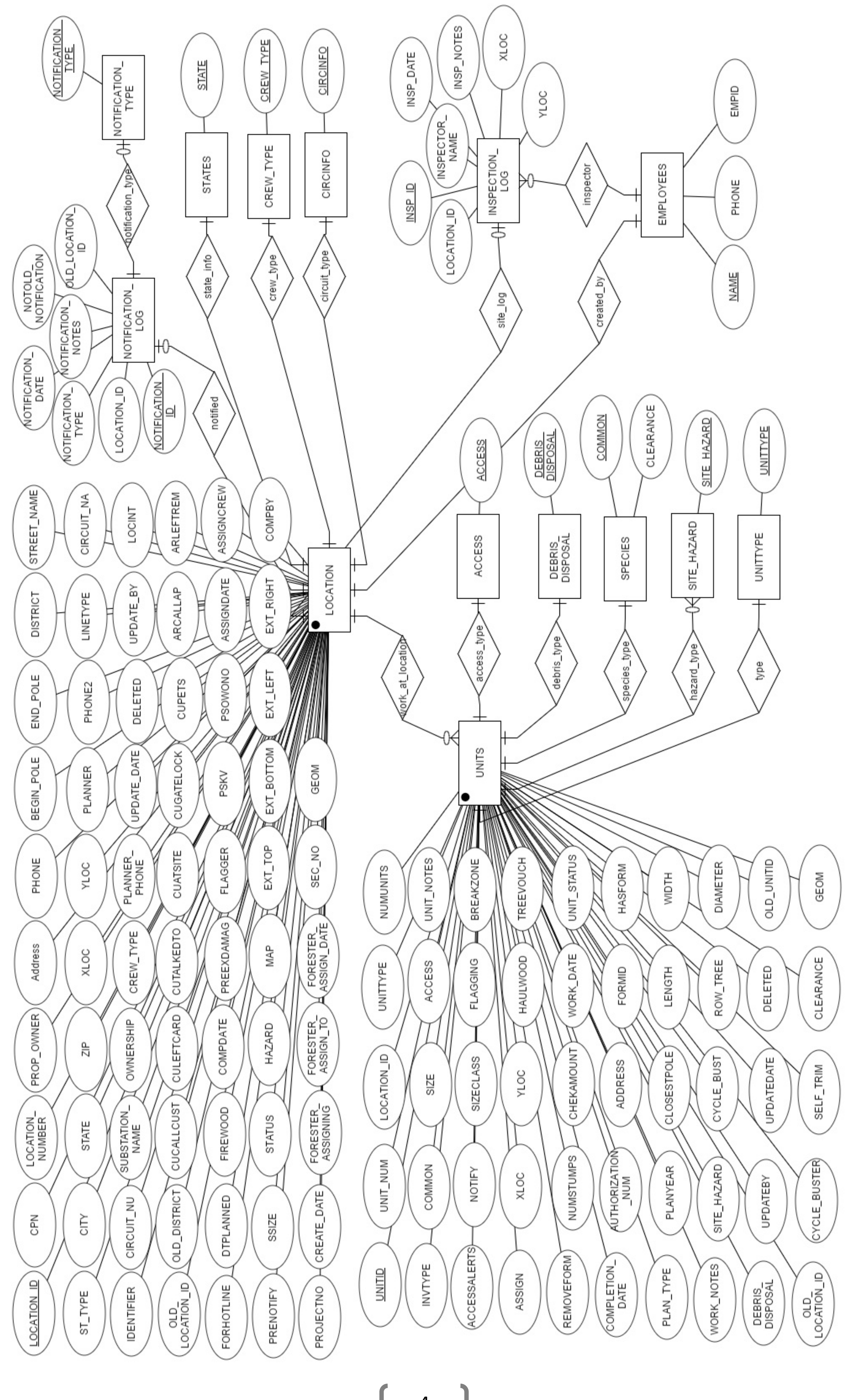


Figure 1: ER Diagram showing structure of sample vegetation management database

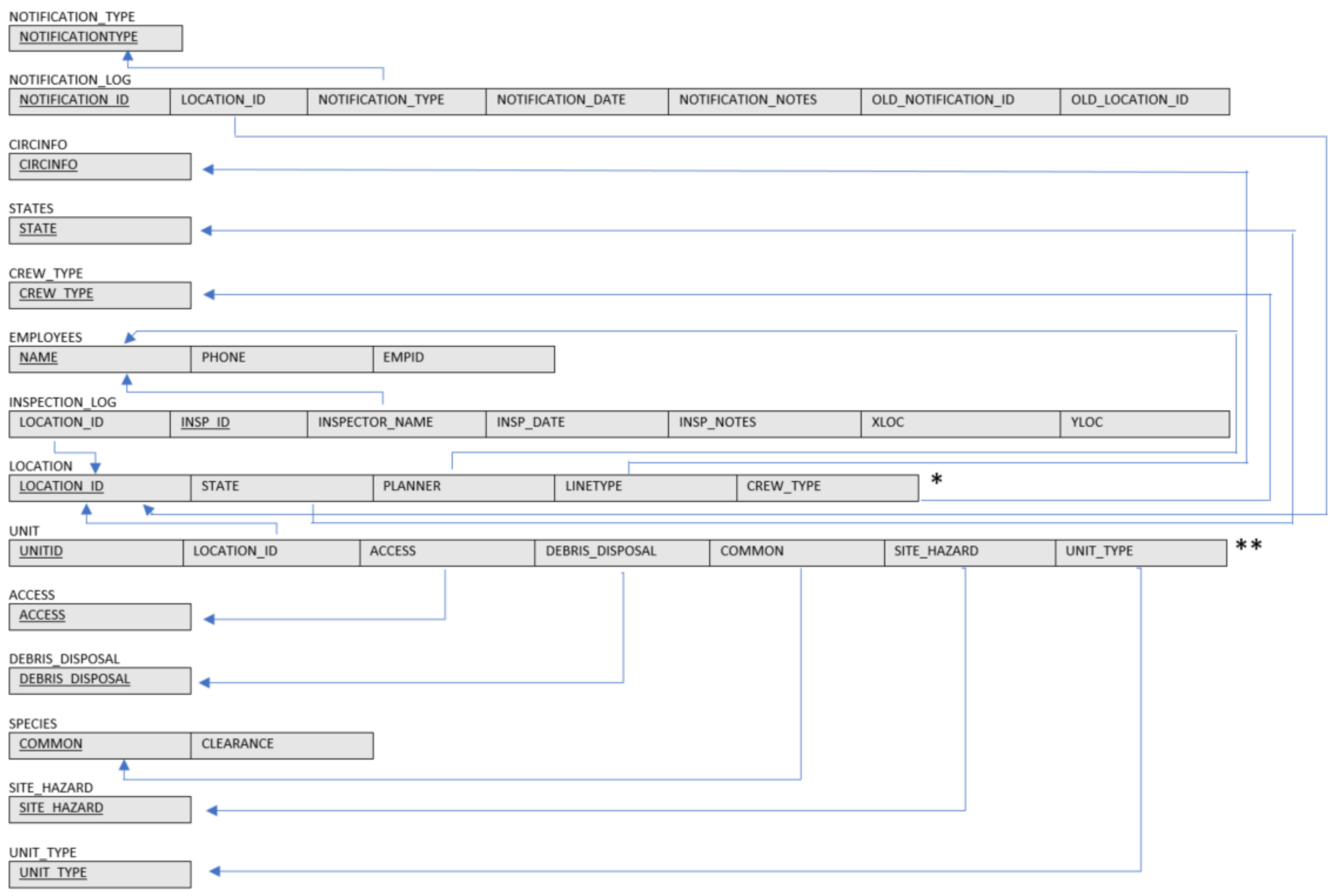


Figure 2: Logical Schema diagram of database structure

*NOTE: * & ** Below are the remaining fields for the LOCATION and UNIT tables respectively*

*

SEC_NO	CPN	LOCATION_NUMBER	PROP_OWNER	ADDRESS	PHONE	BEGIN_POLE	END_POLE	DISTRICT	STREET_NAME
ST_TYPE	CITY	ZIP	XLOC	YLOC	PHONE2	CIRCUIT_NA	IDENTIFIER	CIRCUIT_NU	SUBSTATION_NAME
OWNERSHIP	PLANNER_PHONE	UPDATE_DATE	DELETED	UPDATE_BY	LOCINT	OLD_LOCATION_ID	OLD_DISTRICT	CUCALLCUST	CULEFTCARD
CUTALKEDTO	CUATSITE	CUGATELOCK	CUPETS	ARCALLAP	ARLEFTREM	FORHOTLINE	DTPLANNED	FIREWOOD	COMPDATE
PREEXDAMAG	FLAGGER	PSKV	PSOWONO	ASSIGNDATE	ASSIGNCREW	PRENOTIFY	SSIZE	STATUS	HAZARD
MAP	EXT_TOP	EXT_BOTTOM	EXT_LEFT	EXT_RIGHT	COMPBY	PROJECTNO	CREATE_DATE	FORESTER_ASSIGNING	FORESTER_ASSIGN_TO
FORESTER_ASSIGN_DATE	geom								

**

UNITID	UNIT_NUM	LOCATION_ID	UNITTYPE	NUMUNITS	INVTYPE	COMMON	SIZE	SIZECLASS	ACCESS
ACCESSALERTS	NOTIFY	FLAGGING	BREAKZONE	UNIT_NOTES	ASSIGN	XLOC	YLOC	HAULWOOD	TREEVOUCH
REMOVEFORM	NUMSTUMPS	CHEKAMOUNT	WORK_DATE	UNIT_STATUS	COMPLETION_DATE	AUTHORIZATION_NUM	ADDRESS	FORMID	HASFORM
PLAN_TYPE	PLANYEAR	CLOSESTPOLE	LENGTH	WIDTH	WORK_NOTES	SITE_HAZARD	CYCLE_BUST	ROW_TREE	DIAMETER
DEBRIS_DISPOSAL	UPDATEBY	UPDATEDATE	DELETED	OLD_UNITID	OLD_LOCATION_ID	CYCLE_BUSTER	SELF_TRIM	CLEARANCE	geom

Figure 2a: Logical Schema extended fields

the basis for the revised MS SQL Server sample database. The PostGRES database was built to mirror this MS SQL structure because the MS SQL environment was less familiar.

The two spatial columns from the sample data included a duplicate geometry column that was unnecessary and therefore removed. The remaining spatial columns were then used to create an x,y Event Layer in ArcMap and then exported to preserve the new geometry point field. This new LOCATION shapefile was then imported back into MS SQL Server using the following GDAL/OGR command:

```
C:\Program Files (x86)\GDAL>ogr2ogr -f MSSQLSpatial "MSSQL:server=DESKTOP-
NA0406L\SQLEXPRESS;database=DEMO_FINAL;trusted_connection=yes" "C:\Users\andri\Documents\GIS\UW-
Madison\Geog 574\Final Project\Data\LOCATION.shp"
```

The same process was repeated for the UNIT table using the following command:

```
C:\Program Files (x86)\GDAL>ogr2ogr -f MSSQLSpatial "MSSQL:server=DESKTOP-
NA0406L\SQLEXPRESS;database=DEMO_FINAL;trusted_connection=yes" "C:\Users\andri\Documents\GIS\UW-
Madison\Geog 574\Final Project\Data\UNIT.shp"
```

After the data was fully loaded into the MS SQL Server database, the corresponding constrains and relationships were set.

Next, the PostGRES/PostGIS database was built using pgAdmin and the PostGIS Shapefile and DBF loader 2.1 plugin. After the tables were loaded in PostGRES, the relationships and constraints were set using pgAdmin's object browser and table properties. It is important to note that this could have been scripted for the table creation and relationship/constrain properties however because of the differences in the data types and nomenclature between PostGRES and MS SQL Server, the process was completed manually in order to preserve database integrity and mirror uniformity between the two database management systems.

Database Manipulation:

To compare the conversion potential of the research question, three queries were created at varying levels of complexity. The purpose of this was to understand the capabilities of each database engine to query data, track query speeds, and compare query syntax. Each set of queries are list below in figures 3, 4 and 5.

MS SQL Server:

```
USE DEMO_FINAL
GO
SET STATISTICS TIME ON;
GO
SELECT * FROM LOCATION;
GO
SET STATISTICS TIME OFF;
GO
```

PostGRES:

```
SELECT * FROM "LOCATION";
```

Figure 3: Query #1 – Basic Non-spatial query

MS SQL Server:

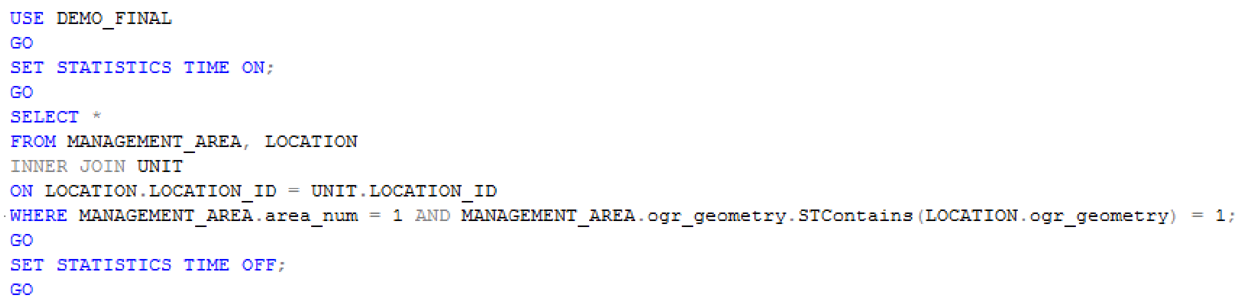
```
USE DEMO_FINAL
GO
SET STATISTICS TIME ON;
GO
DECLARE @t geometry;
DECLARE @h geometry;
SET @t = (SELECT ogr_geometry FROM LOCATION WHERE LOCATION_ID = 1543732);
SET @h = (SELECT ogr_geometry FROM LOCATION WHERE LOCATION_ID = 1543733);
SELECT @t.STDistance(@h)as DISTANCE;
GO
SET STATISTICS TIME OFF;
GO
```

PostGRES:

```
SELECT ST_Distance(a.geom, b.geom) as DISTANCE
FROM "LOCATION" a, "LOCATION" b
WHERE a."LOCATION_ID"=1543732 AND b."LOCATION_ID"=1543733;
```

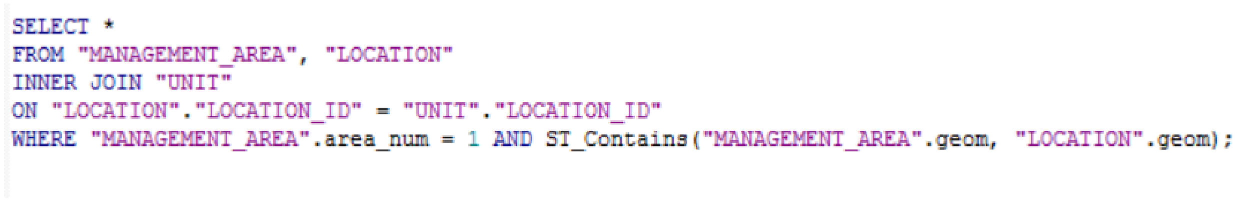
Figure 4: Query #2 – Basic Spatial Query

MS SQL Server:



```
USE DEMO_FINAL
GO
SET STATISTICS TIME ON;
GO
SELECT *
FROM MANAGEMENT_AREA, LOCATION
INNER JOIN UNIT
ON LOCATION.LOCATION_ID = UNIT.LOCATION_ID
WHERE MANAGEMENT_AREA.area_num = 1 AND MANAGEMENT_AREA.ogr_geometry.STContains(LOCATION.ogr_geometry) = 1;
GO
SET STATISTICS TIME OFF;
GO
```

PostGRES:



```
SELECT *
FROM "MANAGEMENT_AREA", "LOCATION"
INNER JOIN "UNIT"
ON "LOCATION"."LOCATION_ID" = "UNIT"."LOCATION_ID"
WHERE "MANAGEMENT_AREA".area_num = 1 AND ST_Contains("MANAGEMENT_AREA".geom, "LOCATION".geom);
```

Figure 5: Query #3 – Advanced Spatial Query

Note: *Query #3 used an additional polygon layer in order to utilize the ST_Contains command. This layer was added after the rest of the database was modeled and built and did not have any new relationships to the other tables.*

Results

Quantitative Results:

Query Completion Time (milliseconds)			
Time Trial #1	Query #1	Query #2	Query #3
PostGRES	577	31	1138
MS SQL Server	177	15	192
Time Trial #2			
PostGRES	587	19	1115
MS SQL Server	126	22	206
Time Trial #3			
PostGRES	556	32	1119
MS SQL Server	142	31	187
Average			
PostGRES Average	566.66	27.33	1124
MS SQL Average	148.33	22.66	195

Table 1: Query speed test results

Qualitative Results:

In my project proposal, I theorized that PostGRES/PostGIS would have faster query times because PostGRES has a more dedicated spatial component and the open source community has many dedicated contributors to the OpenGeo initiative, however as Table 1 shows, my time trails resulted in MS SQL Server having faster query completion times. This conclusion shows that it does not make sense to migrate existing MS SQL databases to PostGRES if you are looking for faster speed alone. However, there are still other reasons you may want or need to make a conversion; familiarity, cost, or query complexity. Regardless of reason, my methods did show that the conversion is possible.

Additional considerations include full spatial query functions to see if one database is more robust in it's over all capabilities. Also, spatial indexes were not considered in the scope of this project and might provide different results if the datasets were fully optimized.

References

1. Robert G. Wagner, Keith M. Little, Brian Richardson, Ken Mcnabb; The role of vegetation management for enhancing productivity of the world's forests. Forestry (Lond) 2006; 79 (1): 57-79. doi: 10.1093/forestry/cpi057
2. Esri. "Types of Geodatabases." An Overview of the Geodatabase—ArcGIS Help | ArcGIS Desktop. Esri, 2016. Web. 01 May 2017. http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/types-of-geodatabases.htm
3. ERDPlus. (2015). ERDPlus. Retrieved May 03, 2017, from http://erdplus.com/