To SQL or NoSQL That is the Question

Data can come in many forms. Additionally, one could also classify this data in various ways dependent on content. To compound this issue further, a multitude of systems are present to store, retrieve, and interact with this data. To Clarify, the data itself and the database management system(DBMS) are two separate entities and the DBMS determines how the data is stored. The two major classifications of these DBMS are SQL and NoSQL. SQL is commonly used in smaller data management techniques and was the first of its kind. SQL uses a set structure to represent its data as tables to store its data with rows as individual entries and columns as properties of these entries, known as relationships. NoSQL has no such constraints. Each entry in a NoSQL database can have a separate structure, also called schema. There are some other types of DBMS that will be discussed later in the reading.

The end users of these DBMS are traditionally larger enterprises. However, in recent years there has been a rise in the lower level industry taking advantage of these resources. The same can be said for the users of the DBMS. In general, the software industry in the modern era is quickly taking over as the demand for larger data sets is ever increasing. Is this the correct assumption? The answer to that is not as simple. The best choice can only be made when considering two things, the content of the data and the typical interactions one expects to encounter. One might ask why that is important.

The content of and interactions between the database will have a profound impact on how the data should be treated. If the data is uniform with minimal optional fields and maximum connections, then it would be recommended to use SQL. If the data being used is unknown or non-uniform with very little connection between them, it may be better to use a non-relational DBMS. This of course leaves a good window open for interpretation. In this case, there are other factors one may accommodate for. The next one to consider is the interaction expected. For most basic queries, SQL is just as good if not better than NoSQL, this is directly correlated to the algorithms used for analyzing the datasets and the structure of the stored data. Some of the better use cases for NoSQL include problems related to graphing algorithms, such as shortest paths and minimum spanning trees.

SQL data is stored as tables, mentioned earlier, which means that the total data consists of n number of entries and p number of properties, this includes empty values. NoSQL data is stored as key value pairs. This means that there will be n number of entries each with their own separate p number of properties. One can conclude that this takes up less space and makes some traversal easier than tables. This leads us to another advantage of NoSQL over SQL, scalability. Because of the storage type, SQL is known to be vertically scalable. This means that to increase the data size while maintaining performance, the CPU, memory and disk speed must be increased. NoSQL is said to be horizontally scalable. Performance will be maintained through the addition of computing machines attached to the server.

An additional note to mention is the language of interaction, known as the query language. In all relational databases, such as SQL(Oracle), MySQL, SQLite, and Postgres, the query language is generally the same with only minor differences. This is not the case for non-relational databases such as MongoDB, BigTable, Redis, and Neo4j. in non-relational databases each DBMS has its own query language or API that is used to interact. Some non-relational DBMS have implemented SQL style queries and some use mapping/reducing functions to query.

Let us now begin a review of this authors experience using both relational and non-relational databases using two of each as examples. These four will be oracle’s SQL, MySQL MongoDB, and Neo4j. When first learning to use SQL, the author was given a sample data set and a command to build a database two separate ways. This was the same for both the non-relational databases learned to make this comparison.

With the SQL variants, the commands were simple to comprehend and expand upon. The entire interaction with both SQL variants, Oracle and MySQL was generally the same. First we create a table, then we give it values, and last we implement constraints and relationships. The most difficult part about this system for the author to grasp was making complex queries. One may find themselves having difficulty with this if there are many tables and the data is not easily found or calculated. Both DBMS had a graphical interface to interact with and would show the tables of data inside the programs they utilized. With the NoSQL, the two DBMS used were very different from each other and had separate query languages with separate syntax.

The first of the NoSQL DBMS attempted was MongoDB. MongoDB operates all inside a windows shell and was a little complex to setup. However, with this system, we had preexisting data in a CSV file. A CSV file, or comma separated value, is a commonly used standard for exporting a database to a file. With a simple command, and having the CSV file in the correct directory, the file was loaded and was ready to use in fractions of a second. The query language was a little complex at first but with a couple queries following a simple tutorial, this author could create complex queries easier than with SQL queries. The disadvantage to using MongoDB is that there is not an easy way of viewing the data for visual confirmation of correctness. Because there is no schema to the data, any of the entries may be mixed in with other subschemas for differing entries, also referred to as Nodes in NoSQL. With a SQL database, each differing entry type should belong to its own respective table, making an unfamiliar user able to inspect the data for easier querying. This is not the case with MongoDB. Entries with nested nodes as properties can become confusing to new users.

This leads us to our last example. Neo4j is what is known as a graph database. These DBMS are considered NoSQL databases but they operate a little differently than MongoDB does. The first thing that is separates MongoDB from Neo4j is the query language. Neo4j uses a language called Cypher and was developed for use with Neo4j but has risen in popularity recently and in October 2015 the OpenCypher project started which will try to standardize this to a universal Query language for all NoSQL DBMS. The keywords of this language are very similar to SQL queries. The actual syntax is a different. Because this is a non-relational DB, Neo4j does not infer relationships we have to specify them. That is where things start to get confusing. In this model, the graph model, data is stored in nodes with properties of those nodes inside and relationships between them as edges in the graph. The image below demonstrates this from the sample data given as actors and movies they have played in.

.