Relational & Non Relational Databases

Eric Grant

Northwestern Michigan College

CIT 178

Keith Weber

Structured Query Language, commonly abbreviated to SQL, is effectively a computer language used to speak to databases. A relational database is a method of database management that stores data within tables. SQL is the only way to manipulate data within relational databases. Though, there are many types of relational databases and each type has it’s own proprietary extensions that modify SQL to the point of making it feel like a new language (Serra).

Microsoft SQL Server (MSSQL) and Oracle database are the two leading relational database management systems when it comes to large scale projects. Coming in a close third is MySQL. While MySQL may not be as popular in general it has more appeal when it comes to storing data for web applications (Serra).

MSSQL uses a modified version of SQL called Transact SQL (T-SQL). This is a simpler and easier to use form of SQL. By default in this language every command you run is executed and committed to the database immediately. This means in the case that you need to change two things simultaneously you will have to use specific syntax to imply that you want them done at the same time. Another quirk of this SQL variant is the way data is organized. All objects are grouped by database names. Users are then given accounts that are granted access to specific database names (Stansfield).

Oracle database uses their own variant of SQL known as Procedural Language SQL (PL/SQL). This SQL variant is more complex than T-SQL. The primary differences are in the way it commits changes and the way database objects are organized. The default in T-SQL is to only commit changes when told too. Actions will change the local memory, then all of the actions will be committed at the same time when commanded to commit. When it comes to organizing data, T-SQL groups objects by schemas, putting related objects together as opposed to grouping objects under a database name. Each schema is then shared with all users unless the user is given restrictions (Stansfield).

MySQL is the odd one out. It works better for web apps thanks to built in compatibility with PHP but unlike MSSQL it requires third party tools to work with .NET. MySQL also allows for different variants of SQL to be selected, making it easier to get specific smaller jobs done by selecting more appropriate tools. It also has the major advantage of being free and open source. Two things that small companies love to hear (Serra).

While SQL and relational databases are the majority, they are not the only ways to access and manage data. Non relational, or NO-SQL, databases store data without requiring tables or SQL. There are four primary types of non relational databases: Key-Value Store, Column Store, Graph Database, Document Store. In a Key-Value Store data is simply assigned to a key. With Column Store arranges data into columns. Graph Databases store data in key-values then connect them logically (Serra).

Document Store databases work by assigning data to a key then storing it in documents. This data can be entire pages of relevant information all saved on one page. The most used Document Store database is MongoDB. The reason for it’s popularity is that it’s written in JavaScript, a simple and light language that many people are familiar with. In MongoDB you select a document, then select the keys you want to look at or change. This is all done without SQL (Homan).

MongoDB isn’t the only choice when it comes to non relational databases. Cassandra is a top competitor with MongoDB. They’re both document databases which means at their core they are very similar. The main difference is in the way data is organized and your ability to edit that data. In MongoDB data is organized in an object oriented fashion, where each document can be nested under another creating effectively a folder full of other folders and documents. On the other hand Cassandra stores data in tables similar to relational databases. MongoDB also allows you to set multiple indexes to a single piece of data allowing you to search for data using different terms. The main advantage to Cassandra is the way it hosts the data and allows you to edit it. In MongoDB only one server lets you edit and if that server goes down it has to transfer that ability to a server under it, which can take time. In Cassandra any server allows editing and multiple servers can be hosts, so when one goes down the other is still up (Rangegowda).

The major advantage to non relational databases are that they’re simple, scalable, and fast. Typing out a line of code to select the data you want then typing out what you want to do with it is faster than writing out a sheet of SQL that joins together tables of data. It also makes it easier to store massive amounts of data. With all the data contained inside a document it can be moved to a new server no problem. With relational databases all the data is split into tables, if you move half of the tables to a new server you're going to have problems when you try to join them. In terms of processing speed, it’s quicker to pull from one document instead of many different tables (Serra).

That being said, Non Relational Databases also have some downsides that prevent them from being better in every situation. The big problem is that when everything is inside a document, you can’t instantly join it with another and get data from two different places. Instead you have to pull the data separately then manually put it together in your code. This can get messy and hard to manage. The other problem, it’s different. People are used to SQL and the way it works, trying to move everything over to a completely new system and learn a new way of doing things is not always an option (Serra).

All and all, both relational and non relational databases have their place in the world of data management. As big companies continue to expand and more data needs to be stored i’m sure we will see a rise in the use of non relational databases. At the same time, if it ain't broke, don’t fix it. Relational databases work perfectly fine in many situations giving no real reason to stop using them.

References

Homan, J. (2014, April 6). Relational vs. non-relational databases: Which one is right for you?

Retrieved from Pluralsight website: https://www.pluralsight.com/blog/software-development/

relational-non-relational-databases

Rangegowda, D. (2016, August 9). Cassandra vs. MongoDB. Retrieved from ScaleGrid website:

https://scalegrid.io/blog/cassandra-vs-mongodb/

Serra, J. (2015, August 27). Relational databases vs Non-relational databases [Blog post]. Retrieved

from James Serra's Blog: http://www.jamesserra.com/archive/2015/08/

relational-databases-vs-non-relational-databases/

Stansfield, J. (2014, March 13). Microsoft SQL Server vs. Oracle: The Same, But Different? Retrieved

from Seguetech website: https://www.seguetech.com/

microsoft-sql-server-vs-oracle-same-different/