Sara Huang
Problem Set 2
11/09/2018

There are many variable factors in today's life that contribute to the ever-growing, ever-expanding advancement of science and technology. In particular, one of those factors is the Internet. The Internet has quite literally, taken over our lives, so to speak, and it is quite remarkable to see how ingrained and natural it fell into place as one of the most irreplaceable aspects of our daily lives. As with the many benefits the Internet brings, such as quick access, there also arises many complications as well. In order for the Internet to become so great, there of course exists an underlying structure that is capable of handling such a vast amount of data, which is where databases come in. Databases are, simply put, ways of storing data for easy access. Clearly, different ways are used so that different tasks become easier in one way versus another, alluding to different needs that databases serve for people. For the Internet specifically, as updates happen spuriously and sporadically, accumulation of those modifications adds up very fast so there needs to be a way to meet that kind of mass update.

The Internet makes our lives easier, yes, but also makes our lives complicated too. What we take for granted every day is the simplicity and easy use of it and fail to realize the underlying complexity of the whole system. On the surface, the Internet is a wonderful thing, but underneath, lies a more complicated structure, especially with regards to maintaining that data. As previously mentioned, changes to the Internet happen rapidly and there needs to be a system in place that can meet those just as quickly as they are being committed [1]. This fluid data backbone structure causes entities to be moving, and have dynamic qualities.

Because the Internet is so large, it becomes unwieldy and hard to manage big data. (Ferguson et al.) Furthermore, since it is so vast, there is a very wide scope to cover that cannot possibly be all mentioned. Therefore, it is in our best interest to hone in on the main problems with the WWW: big data and data integration. To combat these issues, two popular forms of database management systems are proposed: SQL and NoSQL. The two database designs are alike and different in many ways, but in order to see why NoSQL is preferred over SQL, understanding and analysis of the two are required.

SQL (Structured Query Language) is best at handling data where there are relationships between different variables of data. It was the first database type to access many records with just one command and eradicated the need for an index to obtain a record. SQL made its debut in 1974 and was first released in 1986. Traditionally, the SQL database is pretty standard. After all, it gets the job done. They work for relational databases and are primarily table-based. There needs to be a predefined schema to

use SQL on querying and manipulating the data. This method is safe and guaranteed, so long as the data is robust and reliable. The SQL language has a particular format and style to its use, namely clauses, expressions, queries, and statements. These make up any interactions with the database via SQL. The clauses are parts of the statements and queries, which may be optional depending on the query you give to SQL. The expressions are what one is looking for in the query and the predicates are the specific conditions that are applied to the statement. The queries are the handlers in obtaining the data based on clauses, expressions, and predicates and the statements are the commands given to SQL as outlined by the clauses, expressions, and predicates. Multiple views can be created in SQL to accommodate different users. As for speed, it is pretty fast in dealing with large relational databases. That is the primary function and goal for designing SQL, which is what makes it so strong and lasting. Moreover, SQL can be extended to object-oriented programmabilities, such as SQLAlchemy with Python, JavaScript, or C++. SQL's versatility in being allowed to manifest itself across different programming languages is what makes it appealing and attractive to its users and developers. SQL is also portable in terms of data. It supports PCs, laptops, tablets, and more. In general, SQL databases are known for following ACID properties since it follows the traditional relational schema-based database system [2].

However, even with all these advantages and solutions that SQL aims to bring to the table, there are many disadvantages too that SQL is unable to resolve. For instance, sometimes there is high cost associated with operating and accessing versions of SQL. The interface is also difficult to access in some cases, making SQL unwieldy and hard to use. To help alleviate some of these issues, we turn to NoSQL. NoSQL databases provide ways to store and retrieve data that is modeled other than the traditional relational databases that SQL was based off of. Pressured by Web 2.0 companies, NoSQL databases are widely used in big data and real-time web applications to make up for what SQL cannot do in those cases. NoSQL (nonSQL or "Not Only SQL") supports not only SQL as in their name implies (that it may support SQL-like query languages) but also nonrelational databases. Since the premise is designed off of a relationless database, it follows that some operations are faster in NoSQL than in SQL. Some disadvantages that NoSQL has in its structure are that transactions don't really support ACID, consistency is compromised in order to support speed, joins across tables are not as easy as in SQL, and there isn't really a standardized interface, making it hard for different users to get accustomed to. To compensate for those disadvantages though, NoSQL has a property called "eventual consistency" in which any changes to a database can be updated everywhere it needs to be. This makes adding new entries or updating entries in NoSQL much more efficient than in SQL since the concept of dynamic, unstructured data comes into play in NoSQL that allows for flexibility in data storage. This may make the results not up-to-date or inaccurate in NoSQL, as there are

constant updates that take milliseconds to process ("stale reads"). Some NoSQL systems can show signs of lost writes or data loss from this type of management. To mitigate this problem. NoSQL takes on a write-ahead logging approach to avoid data loss. For transactions across multiple databases, both SQL and NoSQL suffer difficulty in maintaining integrity across all databases that are affected. NoSQL incorporates SQL in its language, albeit in a different form so NoSQL bests SQL. NoSQL has SQL-like capabilities in it, just not exactly the same as SQL; however, NoSQL is more general in the sense that they don't require a strictly defined database schema as in SQL so it is adaptable to whatever the situation calls for at hand.

For solving web problems specifically, the data is best represented in graphical form, as shown in Florescu et al. They propose a directed graph to model the WWW. NoSQL has high flexibility and complexity in tackling these databases alongside with performance and scalability which can vary depending on the scale and range of the data being handled. Since most NoSQL databases cannot do joins in queries given to NoSQL, there needs to be an alternative to designing the database schema. The three main ways of handling relational data in a NoSQL database are: using multiple queries, caching and replicating data, and nesting data. In SQL, to merge tables, it would generally on the same command or same query line. In NoSQL, it can take the form of multiple queries to get the job done, but since NoSQL is faster than SQL usually, the cost of having to do more queries balances out the time complexity that it takes to do the same job. In SQL, foreign keys are often stored to illustrate how one table is related to another. In NoSQL, foreign keys are manifested into the model's data so data lookup is really easy to access. If reads are more common than writes, caching and replication of data is preferable. With databases like MongoDB, sometimes it is standard to put more data into smaller units. Nesting data in this case will give users all that they need for a particular task or command at hand. MongoDB supports ACID and join properties so the degree in which it has the capability of SQL is favorable.

With the elastic scalability of NoSQL, it makes use of ever-changing data. In today's world especially with rapidly changing entities, NoSQL is a must designed to feed off that. They are made in such a way that transparent expansion is possible on a large scale and are a better fit for society. To satiate the dominance of transactions, NoSQL is able to easily deal with large amounts of databases and transactions. There is a top-down approach in NoSQL in that management is low-level, with simplified data models, fewer tuning and enhancements, and less administration requirements [2]. This makes NoSQL economical and conservative in resources, making it cheaper and less wasteful in the long run to implement. Economy in storage systems and servers needed to use NoSQL are also much cheaper since relational databases often need expensive storage systems and servers to run. This means that even with large amounts of data, processing and storage of that data is affordable in NoSQL as compared to SQL.

NoSQL is able to overcome many of the limitations that SQL was founded upon and can build off of that. To go about trying to solve the initial problem of big data and data integration of the Internet, the graph model application of NoSQL is used. The types of queries used in NoSQL would be multiple queries and the indices would be keywords or phrases that people are generally looking for.

There have been numerous endeavors to try and integrate databases and the WWW together. Initially, using SQL, as that was the accepted standard, an Oracle gateway was made to handle SQL SELECT statements which would allow the users to enter in a SQL query and that would in turn become search text to the Web server [3]. The results of the query are then sent back to the user. This Oracle gateway has been improved as it allows for HTML connections between rows and tables and updates of data in predefined tables. Today, it is the Oracle Web Application Server built to support transactional services over the Web. It is able to merge HTML-based Web tools with business transactions through multithreading so that the Web servers the businesses are based upon are secure and reliable [4]. Since then, the evolution into NoSQL database modifications has been phenomenal. Another attempt (SQL-based) at trying to make the relationship between the Web and databases possible is through WDB, a software tool that was written in Perl for the conversion of data from the database into hypertext links on the Web. A database entity could be accessed directly from a Web URL, making the entire database a big hypertext data system [5].

For NoSQL-based attempts at trying to resolve the aforementioned problems with the database and WWW, MongoDB is the most popular method. With MongoDB, it leverages the work of Oracle and NoSQL to provide modern applications on websites [6]. Designed for handling unstructured, massive, nonrelational data, MongoDB brings to the forefront its ability into metamorph inherently in its nexus architecture. In MongoDB, data is stored as documents in binary form so that for blogging websites, for example, are modeled in two categories: users and articles. In each blog document that they have, each contains comments, tags, and related entries [6]. This is useful and a step up from the SQL attempts at trying to resolve non-relational schemas. Another attempt at trying to solve the problem of big data is Hadoop, a subsidiary of Apache. Hadoop was designed with the goal of being able to store and handle large databases reliably and does so through using MapReduce [7]. The founders of Hadoop implemented it on Yahoo processing clusters of data and noticed that it was performing really well, so well that it ultimately became the backbone of Yahoo today. Through allocating storage and computation amongst numerous servers, large clusters of data can be accessed while maintaining durability of the data and still being affordable.

The Internet is a prevalent and domineering force in today's world as it has become so integrated into our daily lives that living even a single day without it becomes unbearable. Because of the increasing reliance on the WWW (World Wide Web) as a

means for gathering information (data), there is a rising need to organize and store that data for access by all users. In fact, Google's mission statement is "to organize the world's information and make it universally accessible and useful." Alternatively, it is “Don’t be evil” (now “Do the right thing.”). With great power comes great responsibility and the tools we have at our disposal are quite phenomenal. Google is pretty much indispensable, surfing the web for what you are looking for, and has become synonymous with the go-to for looking up information. Surely if they were to use a database, NoSQL is their best bet. Obviously, Google is web-based and with the changing Internet, the only way to match that kind of change would be to use NoSQL for modifying data. Amazon also too has a similar concept in their stock inventory, except it is handled differently in that buyers are unbeknownst to the quantity available until immediately preceding checkout. It is not that their database is slow or incompetent in any way; it is so that they attract more customers to its website to buy and do not wish to deter customers by providing the most recent status up to date on their online shopping cart so as not to disappoint them right away. This is strategic and perhaps can be attributed to the success of Amazon today. NoSQL also works best in this case because Amazon has hundreds (maybe even thousands) of transactions per second, so they of course need a trustworthy database system to update very fast, in a matter of seconds. Yahoo in turn also uses NoSQL through their use of Hadoop. This shows that NoSQL is visionary, empowering the technologies most used today and onward.

**Bibliography:**

[1] D. Florescu, A. Levy, and A. Mendelzon, “Database Techniques for the World-Wide Web: A Survey,” [Online]. Available: https://cse.buffalo.edu/~mpetropo/CSE705-FA08/pubs/webdb.pdf. [Accessed November 14, 2018].

[2] B. Sethi, S. Mishra, and P. Patnaik, “A Study of NoSQL Database,” *International Journal of Engineering Research & Technology*, vol. 3, issue 4, April, 2014. [Online serial]. Available: https://www.ijert.org/phocadownload/V3I4/IJERTV3IS041265.pdf. [Accessed November 14, 2018].

[3] H. Lu and L. Feng, “Integrating Database and World-Wide Web Technologies,” [Online]. Available: https://www.cs.purdue.edu/homes/ake/ec/activities/rev-form.pdf. [Accessed: November 14, 2018].

[4] Oracle Web Application Server, Oracle Corporation, [Online]. Available: http://www.olab.com/products/websystem/web-serve, 1997. [Accessed November 14, 2018].

[5] B. F. Rasmussen. “*WDB - a Web interface to SQL databases*,” [Online]. Available: http://arch-http.hq.eso.org/bfrasmus/wdb/wdb.html, 1995. [Accessed November 14, 2018].

[6] MongoDB Architecture Guide. [Online]. Available: https://jira.mongodb.org/secure/attachment/112939/MongoDB_Architecture_Guide.pdf, 2015. [Accessed November 14, 2018].

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” *International Conference on Massive Storage Systems and Technology*. [Online]. Available: http://storageconference.us/2010/Papers/MSST/Shvachko.pdf. [Accessed: November 15, 2018].