"NoSQL" And Service Science

Donna M. Schaeffer, Ph.D., Marymount University, USA
Patrick C. Olson, Ph.D., National University, USA

ABSTRACT

Discussions of "NoSQL" naturally tend to contrast a new approach with a tired old approach. An example is the discussion in Linux Journal (Batholomew, 2010) which focused on differentiating between these new "non-relational" products and "traditional" (relational) systems. While enthusiasm for the new (and the unjustified raised expectations) has often been the hallmark of information systems and computing, in this instance the enthusiasm may be providing a poor and misleading explanation. A more careful explanation is needed, particularly as these approaches and products may be needed to provide the realization of many promising ideas such as those associated as with Web Services.

Keywords: NoSQL; Service Science; Relational Database

INTRODUCTION

Data is at the very heart of information, computing, and service science. It provides the means and the matter of interactions. Data has been in every story about the history of computing. In fact, data has been a critical part of almost every transition in computing hardware from Babbage, to Hollerith, to Hopper, to Gates and Jobs, to Berners-Lee and now NoSQL.

The transitions in how data has been handled are important turning points for information systems and provide an indication of how important NoSQL may become. For example, data is part of the hardware on the Eniac and inadvertently illustrates the understanding of information systems at that time. The result is a machine that must be rewired for new data. The idea of separating data and processes to divide and conquer problems is still not available for early program languages. The result is the program must be rewritten for new data. This is refined to listing data separately at the end of programs, and finally to storing data and programs separately. This separation provides the circumstances needed to begin to discern the differences in the attributes of data and programs. Over time it appears that data structures are relatively static, programs are more dynamic than data structures, and data is the most dynamic aspect of these systems. This revelation gives a good indication of how to build support systems for data and yields components like data definition language (DDL) and data manipulation language (DML). These concepts evolve into database management systems (DBMS) and in a move to give a firm philosophical foundation for the data oriented approach E. F. Codd develops the relational model in the early 1970's.

This history involves many participants, and has many stories about new versus old approaches. However, the relational model may be something more analogous to the generative transformational grammar that made programming languages workable two decades earlier rather than a fashion to be discarded with the onset of a new and different fashion. Discovering this first involves establishing what is NoSQL.

WHAT IS NOSQL?

The *Linux Journal* cover is an advertisement for "Database Debate: SQL vs. NoSQL" and is certainly intended to attract purchasers to the magazine. While such a "debate" could go unnoticed in many fields, an attribute of computing and information systems is that these fields have a high rate of change, therefore the members of the computing and information systems fields are likely to be concerned. An additional challenge is that the topic is database, which is sometimes considered more settled than other parts of computing and information systems.

The author of "SQL vs. NoSQL" (Batholomew, 2010) works very hard to be an objective journalist and this is evident in his effort to attribute comparisons of NoSQL and Relational Database Management Systems (RDBMS) to those working with NoSQL. This effort is partly the result of the author working for a company that is involved with a product that is a "community-enhanced branch of MySQL", an RDBMS product. Thus, the following statement is likely to be a real belief from those involved in the "NoSQL" community;

Most classic RDBMSes initially were designed to run on a single large server. That is how it was done in the late 1970s and the early 1980s, and the idea exists in the design of many RDBMSes to this day. (Batholomew, 2010)

The value of this statement is that it reveals an extremely important attribute of NoSQL, even though the characterization of RDBMSes misses the mark. The statement clearly indicates that NoSQL advocates are using the NoSQL products/approaches because RDBMSes do not "work" for their needs. Certainly the sales departments for the big commercial RDBMSes would dispute this, but even a cursory understanding of the size and distribution of a data support system needed for many applications on Web should provide some sympathy for the idea that the NoSQL advocates are putting forward. So, from this example it is clear that NoSQL is about the data problems posed by trying to make Web based systems.

An unfortunate aspect of this statement is that it is likely that the mainframe and minicomputer world of that era that spawned RDBMSes is mysterious to many involved in computing today. One wonders how many people that worked with mainframe or minicomputers would agree with the idea that it was a "single large server" environment. This lack of knowledge about the heritage for our field is troubling. It is particularly troubling as the goals of those developing RDBMSes are so similar to the NoSQL advocates as illustrated in the following statement from a 1985 MIS Quarterly article;

But the advent and increasing use of relational DBMSs, fourth generation languages, personal computers, and information centers affect all aspects of data processing including data administration. If there is a common thread through those new technologies, it is that they all tend to move the power of the computer to the end user. (Gillenson, 1985)

The early 1980s were about hierarchical and network DBMSs and RDBMSs were only gaining traction. The goal of empowering the end user is clearly stated in this excerpt and is clear the intention of these involved with NoSQL.

The deduction noted above may have revealed the most important attribute of NoSQL, however, a broader look at this is certainly required. One way to achieve this is to visit the nosql-database.org (Edlich, 2012) site. On this site 150 NoSQL databases are discussed. They are put in 12 categories. These categories are generally related to handling aspects of data that are needed for Web based systems. Could the needs implied by Web based systems be supported by commercial RDBMS vendors? The answer does not seem clear to the authors. Certainly, we know that the commercial products can perform some of these activities; however, issues of performance and cost are clearly involved. The authors are mindful that Google and Amazon are very large customers of the NoSQL approach. Finally, since the Web is involved the issue must be viewed beyond commercial solutions. One important aspect of Web based systems is the process of starting small and growing. This is probably where NoSQL got its name. Web based systems started small and used the famous free alternative to the commercial RDBMSes, and it didn't scale as the Web based system grew. Thus, the title became NoSQL.

WHAT IS THE RELATIONAL MODEL?

It is fair to state that the relational model comes from E. F. Codd. In fact his paper entitled "A Relational Model of Data for Large Shared Databanks" in the June 1970 *Communications of the ACM* (Codd, A Relational Model of Data for Large Shared Databanks, 1970) is the beginning of the relational model. However, that understates some important aspects of this paper.

One of these is elevating the idea of data independence. This paper begins with this requirement and makes an art of reinforcing the idea repeatedly. In the current context of Web based systems it would require that users not need to know anything about the infrastructure that is providing the data. The idea expressed in the paper is the "internal organization of the data" and the modern Web equivalent of this would be much broader than a single machine.

Another important aspect of this paper is the fact that the competing data models are carefully compared to the relational model and shown to be deficient. These models were the hierarchical model and the network model. The arguments presented are still quite relevant. One wonders about "connection traps" and indexing problems in Web based systems.

The word relation refers to the use of the word in mathematic logic and in fact a book on mathematical logic is included the paper's references. Additionally, it is made clear that the relation refers to the columns (fields). While the rows (tuples) have the same data structure, they are independent instances. In a sense being relational is about how the columns are affiliated with each other. Another important aspect of this is that the entry in a column is considered a set. Thus, unforeseen data types (documents or graphic elements) are accounted for from the beginning of the relational model.

IS SQL RELATIONAL?

The term NoSQL is probably related to developers discarding a popular free RDBMS in favor of a different piece of software that more closely met their needs. Additionally, SQL is the usual means of providing a DDL, DML and Data Query Language (DQL) for most commercial and open source products. Thus the association between SQL and RDMS is so strong that it is easy to understand how someone might get the impression that rejecting the limitations of an SQL implementation is also a rejection of the relational model. However, this is not correct for two reasons. One reason is that most NoSQL products are still dealing with sets and relations.

The other reason is that SQL is not relational. That does not mean that it cannot be used to build a relational system, but it is still not relational. This argument was made by E. F. Codd in Datamation (Codd, Fatal flaws in SQL, 1988). In this article he discusses three "fatal" problems with SQL. One notable problem is that SQL allows duplicate rows. This ruins data consistency.

WHAT IS ACID?

In many discussions about the benefits of NoSQL "ACID" joins performance and cost among the primary reasons for using NoSQL. ACID is an acronym that stand for atomicity, consistency, isolation and durability. ACID is often discussed as a relational requirement that impedes the establishment of successful Web based systems.

The ACID attributes are certainly required for an accounting system. They are the means of providing an audit trail. However, these are not attributes discussed by E. F. Codd.

These attributes do appear in a paper on transaction support in 1981 (Gray, 1981) and are explicitly stated in another paper on transaction support in 1983 (Härder & Reuter, 1983). These are important contributions and are necessary to build the transaction processing systems on which everyone in the modern world depends. However, this is not an alternative data model. It is more like an application built, in part, with the support of the relational model.

It might not be possible for contemporary members of information systems to envision with world that IS professionals of the late 1970s and early 1980s worked. However, there is one aspect of that world in very important to this discussion. That is, information systems generally reported to the Chief Financial Officer. In fact, the first computer supported application in many organizations was often Payroll. ACID is a very good set of attributes for data in a Payroll system. However, the close association between ACID and the Relational Model does make it reasonable to assume that ACID is required.

ACID and NoSQL

If an organization were to adopt a NoSQL approach, how are accounting systems supported? From the discussion above, two points inform this. First, it is possible to build a relational system on top of a tool that is not relational. It appears this has been done with SQL for decades. Second, ACID seems to have been established on top of relational systems. So following this same process, it seems likely that an ACID system could be developed within a NoSQL approach. To make this idea more specific, how would one build a General Ledger system if the enterprise approach to data uses a NoSQL approach?

Discussions on the Web related to specific NoSQL solutions sometimes make the direct statement that a General Ledger system should not be built with the particular NoSQL solution. However, this begs the question how do the famous big users of NoSQL accomplish this? When one considers that the major need of some big users of NoSQL is about supporting Web based transactions and that ultimately you would want this integrated with the accounting systems, it is hard to believe that a commercial RDBMS is operated solely for accounting with some sort of elaborate data conversion effort to move data from the NoSQL system into the accounting system. In fact, it seems like that approach would force ACID like activities into the NoSQL systems in order to satisfy the needs of external auditors. The more likely arrangement is segregating accounting within the enterprise wide NoSQL.

IS NOSQL MOVING TOWARDS CODD'S IDEA?

Clearly some of the many categories of NoSQL provided at nosql-database.org deal with data that is not relational. For example, a system aimed at document management that tries to include data from inside the documents as well as about the document would not be relational. That is, this kind of system muddles the necessary distinction between data about the document and data in that document. However, other NoSQL efforts appear to be striving to achieve Codd's ideas.

One of the common themes in discussions about the need for NoSQL is the ability to handle data on the Web. In 1985 Codd discussed several aspects of what is required to really be "relational" in *ComputerWorld* (Codd, Is Your DBMS Really Relational?, 1985). One of the requirements, which the article notes was not getting much attention at the time, was distribution independence. For many NoSQL approaches this is the main goal.

NoSQL in Web Services

Information systems and computing professions have envisioned amazing systems placed on the Web. Many of these are possible because of the Web services concept. Databases that support distribution independence make many of these possible.

CONCLUSION

No matter what the characterization, the NoSQL collection of approaches to data will progress. The authors hope that those involved have been properly recognized, and that they are proud of their contributions. For our part as College Professors we will endeavor to give these advances their due. The authors will also remember E.F. Codd and the many others that have worked over the decades to bring our field forward. We will also try to make the point that the reason "relational" does not disappear is that it is a philosophy for an approach to data. The competing suggestions have been the network and the hierarchical models – and these sometimes have important shortcomings. The many suggested alternatives to "relational" have often been about programming methods or new data types and simply do not address a philosophical approach to data. This is not a problem as these improvements have helped everyone, however, it is important for professionals to know the foundations of their field.

AUTHOR INFORMATION

Donna Schaeffer, Ph.D., has taught technology, leadership, and ethics courses at universities in the United States, Germany, and Korea. She has won awards for outstanding teaching three times in her academic career. She is a Professor in the School of Business Administration at Marymount University in Arlington, VA. She earned her

Ph.D. in the Management of Information Systems at Claremont Graduate School. Dr. Donna is a participant on the Internet Advisory Caucus for the U.S. Congress and the Business Advisory Forum of the United Nations. She has published over 50 articles and book chapters. Donna M. Schaeffer, Ph.D., Associate Professor, Marymount University, School of Business, 2807 N. Glebe Road, Arlington, VA 22207 USA. Telephone: 703.284.5718. E-mail: donna.schaeffer@marymount.edu (Corresponding author)

Patrick C. Olson, Ph.D., is currently an Associate Professor of Computer Science and Information Systems at National University. He earned his PhD in Management of Information Systems at Claremont Graduate University in 1999. He has an MS from USC in Systems Management. His undergraduate degree is from the University of Montana. He has been CIO at Menlo College where he developed, directed, and implemented enterprise-wide IP Telephony in 2000. He has held faculty positions in MIS at the University of Nevada and Cal Poly, Pomona. He started his career in the data center at Hughes Aircraft Company. Patrick C. Olson, Ph.D., Associate Professor, National University, School of Engineering and Tech., 3031 Tisch Way, 100 Plaza East, San Jose, CA 95128 USA. Telephone: 408.236.1152. E-mail: polson@nu.edu

REFERENCES

1. Batholomew, D. (2010, September 1). SQL vs. NoSQL. *Linux Journal*.
2. Codd, E. F. (1970, June). A Relational Model of Data for Large Shared Databanks. *Communications of the ACM*.
3. Codd, E. F. (1985, October 14). Is Your DBMS Really Relational? *ComputerWorld*.
4. Codd, E. F. (1988). Fatal flaws in SQL. *Datamation, 34*(16), pp. 45-48.
5. Edlich, S. (2012, November 8). *NoSQL Your Ulimate Guide to the Non-Relational Universie*. Retrieved November 8, 2012, from No-SQL Databases: nosql-database.org
6. Gillenson, M. (1985). Trends in Data Administration: 1981-1985. *MIS Quarterly, 9*(4), 317-325.
7. Gray, J. (1981). The Transaction Concept: Virtues and Limitations. Proceedings of the 7th International Conference on Very Large Databases (pp. 144–154). Cupertino CA: Tandem Computers.
8. Härder, T., & Reuter, A. (1983). Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys*, 287–317.

NOTES