Proceedings of Informing Science & IT Education Conference (InSITE) 2015

Cite as: Mason, R. T. (2015). NoSQL databases and data modeling techniques for a document-oriented NoSQL database. *Proceedings of Informing Science & IT Education Conference (InSITE) 2015,* 259-268. Retrieved from http://Proceedings.InformingScience.org/InSITE2015/InSITE15p259-268Mason1569.pdf

NoSQL Databases and Data Modeling Techniques for a Document-oriented NoSQL Database

Robert T. Mason
College of Computer & Information Sciences,
Regis University, Denver, CO, USA

RMASON@REGIS.EDU

Abstract

NoSQL databases are an important component of Big Data for storing and retrieving large volumes of data. Traditional Relational Database Management Systems (RDBMS) use the ACID theorem for data consistency, whereas NoSQL Databases use a non-transactional approach called BASE. RDBMS scale vertically and NoSQL Databases can scale both horizontally (sharding) and vertically. Four types of NoSQL databases are Document-oriented, Key-Value Pairs, Column-oriented and Graph. Data modeling for Document-oriented databases is similar to data modeling for traditional RDBMS during the conceptual and logical modeling phases. However, for a physical data model, entities can be combined (denormalized) by using embedding. What was once called a foreign key in a traditional RDBMS is now called a reference in a Document-oriented NoSQL database.

Keywords: NoSQL Databases, NoSQL Data Modeling, Database Technologies.

Introduction

The increase of data volume (Big Data) during the last decade is attributed to a variety of data sources, such as social media, GPS data, sensor data, surveillance data, text documents, e-mails, etc. For example, the Internet of Things adds urgency for companies to be able to handle vast amounts of data (Devlin, 2014). Data that was once considered too expensive to store, can now be captured, stored and processed. A decade ago, large data stores that were measured in Terabytes are now being measured in Petabytes (1,000 Terabytes). According to the media "hoopla", we are living in a brave new (improved) world that has been created by the ingenuity of Web 2.0 companies, such as Yahoo, Google, Amazon and Facebook (Mohan, 2013). The term NoSQL has come to mean databases that are alternatives to the conventional RDBMS (e.g. Oracle, MS SQL Server and IBM DB2). However, scholars that have watched the evolution of the database technology over the last 30 years are cautious and skeptical (Mohan, 2013). Many of the lessons learned during the evolution of RDBMS are being ignored or discounted by the venture capitalist companies at the center of the NoSQL database movement. Many of the technical problems that have been resolved and automated by RDBMS database vendors are now the responsibility of NoSQL database administrators and application developers. Mohan (2013) cautioned that this type

Editor: Eli Cohen

of adhoc development approach for NoSQL databases can lead to long-term disastrous results for end users.

NoSQL Databases

Although traditional Relational Database Management Systems (RDBMS) have existed for decades and are constantly being improved by the database vendors, RDBMS struggle to handle the large volumes of data (Mohan, 2013). However, a new category of database technology called NoSQL databases is able to support larger volumes of data by providing faster data access and cost savings. Basically, the cost savings and improved performance of NoSQL databases results from a physical architecture that includes the use of inexpensive commodity servers that leverage distributed processing. For example, according to Cloudera (2014), traditional RDBMS SAN storage costs average $30,000+ per terabyte, whereas storage for NoSQL databases average $1000 per terabyte. This dramatic reduction in storage cost has made it possible to store data that was previously considered too expensive.

In addition to distributed processing and inexpensive hardware, NoSQL databases differ significantly on their approach to maintaining data integrity and consistency (Roe, 2012). A more relaxed approach to data consistency helps NoSQL databases improve the performance of data storage. Because RDBMS highly value data integrity, they use the ACID theorem for data consistency which was presented in the early 1980's by Jim Grey. ACID is an acronym for Atomicity, Consistency, Isolation and Durability and supports the concept of a transaction.

Atomicity – ensures that all tasks with a transaction are performed completely (all or nothing).

Consistency – ensures that a transaction must leave the database in a consistent state at the end of the transaction.

Isolation – ensures that transactions are isolated and do not interfere with other transactions.

Durability – ensures that once the transaction is complete, the data will persist permanently, even in the event of a system failure.

In contrast, NoSQL databases use the CAP theorem (Consistency, Availability and Partition Tolerance) for data consistency which was presented in 2000 by Eric Brewer at an ACM symposium (Roe, 2012). The CAP Theorem states that of the three possible combinations of CAP, only two are available at a given point in time. In another words, NoSQL databases can have partition tolerance (in a distributed environment) and consistency or partition tolerance and availability, but not all three factors at the same time according to Abramova, Bernardino, & Furtado (2014). Cap theorem has evolved into what is now called BASE (Basically Available, Soft state and Eventual consistency).

Basically Available – means that data will be available, however the response from the database can be a failure to retrieve the data or the data retrieved may be in an inconsistent state.

Soft state – means that the data can change over time as the database seeks consistency.

Eventual consistency – means that sooner or later, the database will eventually become consistent.

Mohan (2013) argues that although NoSQL database do not guarantee the concept of ACID transactions, they must support some form of a smaller transaction to promote data consistency within the database. Early RDBMS supported the concept of uncommitted reads and different levels of locking. Therefore, the concept of BASE is not a new idea in the area of database technologies and distributed processing. ACID was applied to RDBMS as customers demanded better data integrity and reliability from the RDBMS vendors. Leavitt (2010) cautions users that the

lack of ACID requires additional programming to guarantee data consistency and therefore makes NoSQL databases less reliable. Since NoSQL databases don't support SQL, complex query programming can be time-consuming and challenging.

NoSQL databases also differ from traditional RDBMS in the areas of structure and scaling. RDBMS require that the database structure (schemas) must be defined in advance of loading and then accessing the database. This predefinition requirement is often called Schema Write by industry experts and can be a burden when data is in an unstructured format (Moniruzzaman & Hossain, 2013). In contrast, NoSQL column-family databases, such HBase and Cassandra, provide a flexible schema structure that facilitates changes to the schema definition when new types of data are encountered. This type of a flexible schema structure is called Schema Read. New data sources can be easily incorporated into a read schema within minutes, thus altering the structure and data content dynamically. However, Sadalage and Fowler (2013) caution that although a NoSQL database schema can be dynamically altered, there is still the consideration that existing database applications that use the database will have to be altered to use the new data structures. Thus, there is an expense of maintaining existing code to use new data structures, which should be considered when making structural changes to existing NoSQL databases.

Scaling for RDBMS is done vertically and is usually accomplished by adding additional memory and/or CPU to one or more servers. However, NoSQL databases can scale both horizontally and vertically (Abramova, Bernardino, & Furtado, 2014). Most commonly, additional nodes (commodity servers) are added to a NoSQL cluster (group of servers) to scale horizontally (called sharding). Because distributed processing is used for a NoSQL database, after new nodes are added to the cluster, the existing data is automatically distributed evenly across the cluster by the NoSQL database management system. To avoid data loss, complicated data replication methods are applied across commodity servers in preparation for an eventual server failure. The failed server can be easily replaced within minutes and the replicated data is used to populate the new server with the original data.

Types of NoSQL Databases

There are four types of NoSQL databases: Document-oriented, Key-Value Pairs, Wide Column (or Column Family) and Graph (Abramova, Bernardino, & Furtado, 2014). IT organizations will use one or more of the NoSQL database types based upon the characteristics of the data that must be processed.

Document-oriented – as the name implies, stores related information in the form of a document. In a third normal form data model, data is normalized (separated) into different entities with relationships to reduce redundancy and to avoid update anomalies. Within a document-oriented NoSQL database document, the data is denormalized, semi-structured and stored hierarchically (Moniruzzaman & Hossain, 2013). For example, each book in a library of books can be stored in a collection of documents called BOOK. Not only will the book title be stored in a BOOK document, but details about the book such as a list of one or more authors, publication date, edition, publisher, publisher location and ISBN numbers will be also embedded in the same document.

In addition to embedding information within a particular document, it is possible to provide a reference to another collection of documents (Moniruzzaman & Hossain, 2013). A reference (link) is a similar to the concept of a foreign key that is used by RDBMS. A Document-oriented NoSQL Database has a similar structure to a XML document which is hierarchical. MongoDB is an example of a NoSQL document-oriented database. An example of a particular BOOK document is shown below in Figure 1:

Book Title: Business Intelligence and Analytics: Systems for Decision Support

By Ramesh Sharda, Dursun Delen, Efraim Turban

Publication Date: 2015

Edition: 10th

Publisher: Pearson

Publisher Location: Upper Saddle River, NJ:.

ISBN-13: 978-0-13-305090-5

Figure 1. An example of a NoSQL document for a particular book.

Key-Value Pairs – stores information in form of matched pairs with only two columns permitted - the key (hashed key) and the value (Moniruzzaman & Hossain, 2013). The values can be simple text or complex data types such as sets of data. Data must be retrieved via an exact match on the key. The advantage of this type of NoSQL database is that new types of data about a book can easily be added to the database as new key value pairs. Examples of NoSQL databases that use Key-Value Pairs are Project Voldemort, Cache and Dynamo. In the prior book example, the book information from Figure 1 would be stored as shown in Table 1.

Table 1. An example of how key value pairs are stored in a NoSQL database.

Key	**Value**
Book Title	Business Intelligence and Analytics: Systems for Decision Support
Author (set)	Ramesh Sharda
	Dursun Delen
	Efraim Turban
Publication Date	2015
Edition	10th
Publisher	Pearson
…	…

Wide Column (Column Family) – has a format of data storage that is very similar to RDBMS (Abramova, Bernardino, & Furtado, 2014). Although RDBMS tend to have simple data types and a predefined schema (structure), Column-oriented NoSQL databases provide much more flexibility. They can support complex data types, unstructured text and graphics (e.g. jpeg, gif, bmp, etc.). For example, in the example shown in Table 2, author, publication date, edition and publisher can all be included in a complex data type called book details. Cassandra is an example of a Column Family NoSQL database.

Table 2. An example of how data is stored in a column-oriented NoSQL database.

Business Intelligence and Analytics: Systems for Decision Support	
Book Details (includes authors, year, edition, publisher, etc.)	
Ramesh Sharda	
Dursun Delen	
Efraim Turban	
2015	
10th	
Pearson	

Graph – supports data that has an undefined number of network connections (Abramova, Bernardino, & Furtado, 2014). This type of data supports map data, bus transportation links and relationships found in social media. For example, traversing a graph to find the shortest distance between cities is a daunting task using a conventional RDBMS. However, a Graph NoSQL database can facilitate this type of processing. Allegro, Neo4j and Virtuoso are examples of Graph Databases. Figure 2 is an example of a graph database for the distance between major cities in Colorado.

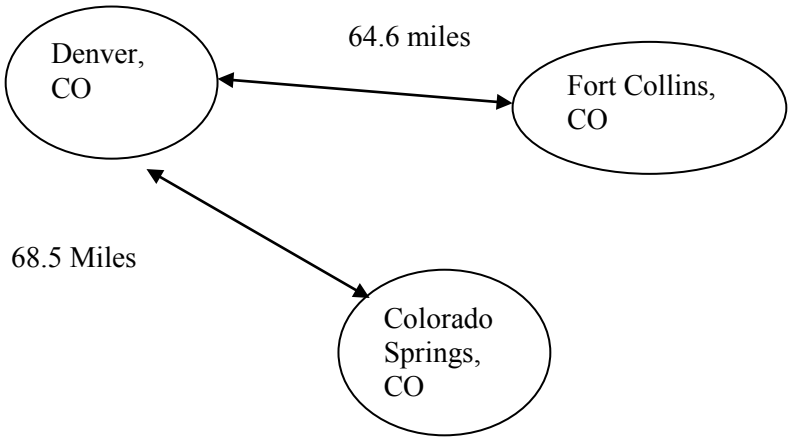

Figure 2. An example of a Graph NoSQL Database for the distance between cities.

Data Modeling Design Techniques for a MongoDB NoSQL Database

MongoDB is an open source Document-oriented NoSQL database that was initially developed in 2007 by a company called 10gen (Medina, 2014). Data modeling on the conceptual level (CDM) and the logical data model (LDM) is very similar to what is done for a RDBMS (Hoberman, 2014). Either normalized data models (3rd normal form) or dimensional models (Star Schemas) are acceptable modeling approaches. The major changes to the data model occur when the data model is transformed from the LDM to a physical data model (PDM). For example, shown in Figure 3, is a very simple example of a 3rd normal form data model (a.k.a. Entity Relationship Diagram - ERD) for a fictitious Rental Car company that includes entities and the associated relationships.

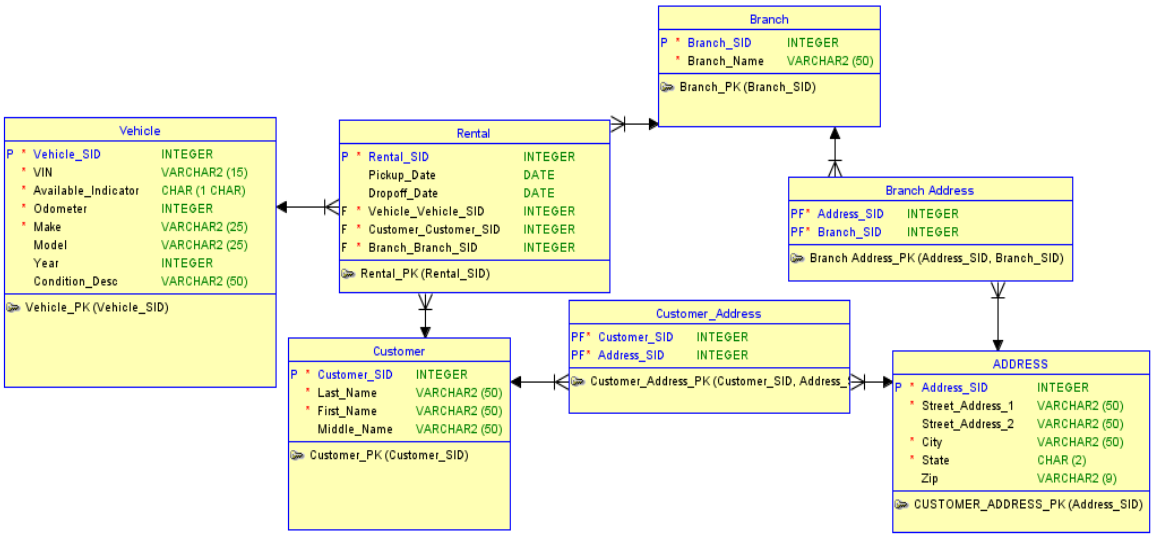

Figure 3. An example of a Data Model for a fictitious Rental Car company.

In this example, an occurrence of a rental is created when a vehicle is rented by a customer from a particular branch. Customers must have a physical address and a Branch must have a physical address.

Hoberman (2014) shows how to conduct a grouping process to convert a LDM into a PDM which can be later used to create the Document-oriented NoSQL Database. There are two options within MongoDB for modeling relationships, embedding data or referencing documents. Embedding is the process of merging together two or more entities into one entity using a hierarchy. Referencing is similar to creating a foreign key in a RDBMS that serves as a pointer form one entity (collection) to another entity (collection).

There are five heuristics that Hoberman (2014) suggests for deciding whether to embed or reference:

1. Data that is that is frequently queried from multiple entities at the same time can be embedded into one document.
2. Entities that are considered dependent entities can be embedded into one entity.
3. If there is a one to one relationship between two entities, we embed one of the entities into the other entity.
4. Entities that experience similar volatility (inserts, updates and deletes) at the same rate can be embedded together.
5. Entities that are not key entities, but have relationships with key entities, can be referenced and not embedded.

Using the LDM provided for the Rental Car Company and the heuristics listed above, entities are grouped together as shown in Figure 4.

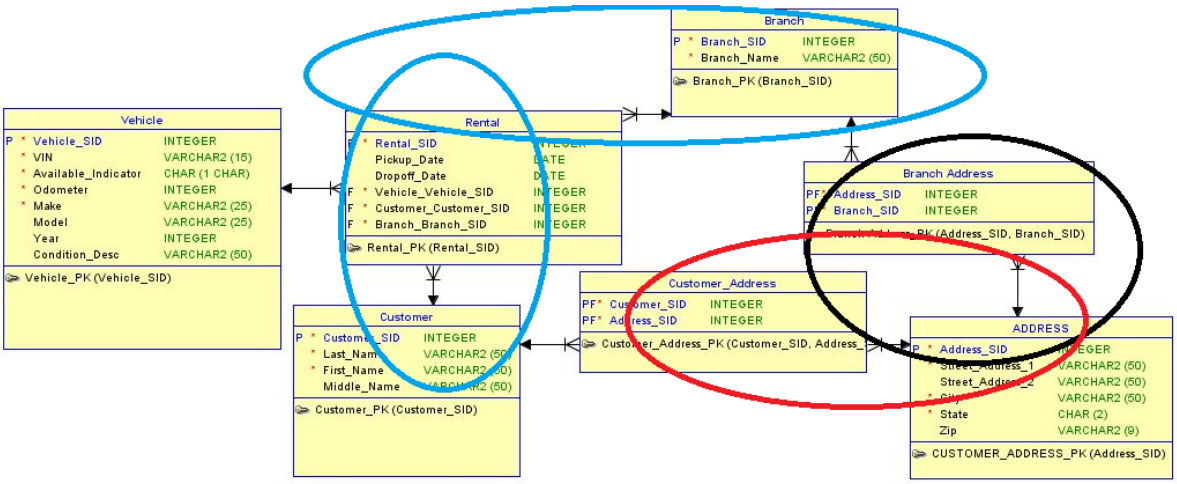

Figure 4. An example of groupings for a Rental Car LDM.

The logic behind these entity groupings for the PDM is as follows:

- Branch Address (black circle) and Customer Address (red circle) are both dependent entities of the Address entity. Therefore, two embedded entities for Customer Address and Branch Address will be created.
- Rental, Customer and Branch (blue circles) will be frequently queried together and will have the same volatility. Although Rental is not a dependent entity since it has a Rental_SID, the volatility makes it a good candidate to be embedded in a new entity called CustomerRental. This new entity will include the embedded attributes of Branch SID and Name and will have a reference to Vehicle which is an independent entity. Figure 5 shows the PDM after the grouping process has been completed. Notice that Customer Rental has references to Branch Address and Customer Address.

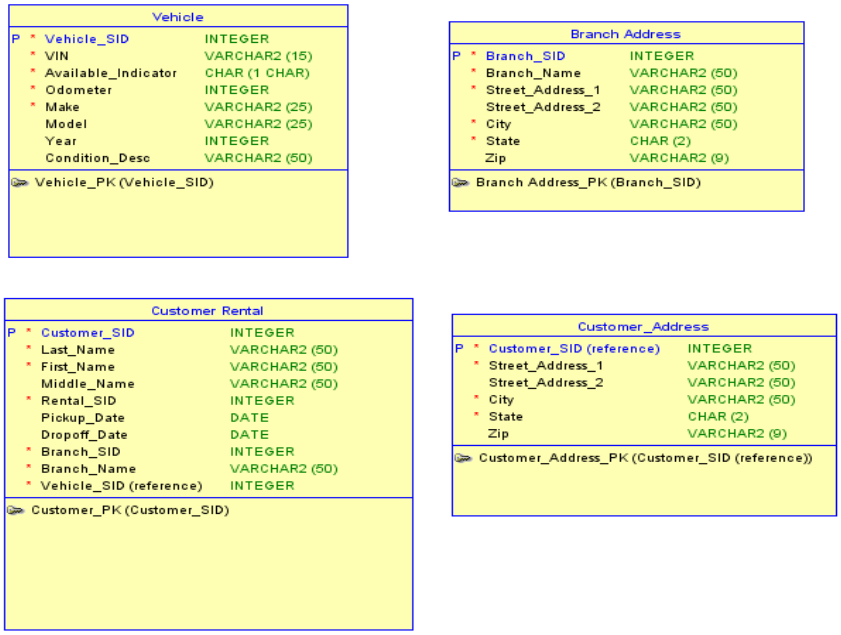

Figure 5. Shows the PDM after the grouping process has been completed.

The PDM can then be used as a template for a collection of documents. Below are samples of MongoDB documents for the Vehicle entity and the CustomerRental entity. Partitioning and additional secondary indexes can be added to improve performance during physical implementation (MongoDB, 2014).

Vehicle:

{ Vehicle_SID: "2134567",

VIN: "1234ZC33456XYZ2",

Available_Indicator : "Y",

Odometer : "3507",

Make : "Toyota",

Model : "Camry",

Year : "2014",

Condition_Desc : "Scratches to paint on the left back door and scratches on the rear bumper"

}

In the example shown below, John R. Smith has rented two cars over the last year. The first car is the Toyota Camry that was picked up at the Aurora location and returned to the Broadway location. Notice the reference to the Toyota Camry via the Vehicle_SID. The second car rental included a different car that was rented at the end of August and returned in September. Notice how the two rentals are embedded in the document underneath CustomerRentals.

Customer Rental:

{CustomerSID: "987765",

LastName: "Smith",

FirstName: "John",

MiddleName: "Ricardo",

Rentals: [

{ Rental_SID: "2300453",

Pickup_Date: ISODate("2014-07-13"),

Branch_SID: "3374",

Branch_Name: "Aurora"

Dropoff_Date: ISODate("2014-07-19"),

Branch_SID: "3370",

Branch_Name: "Broadway",

Vehicle_SID: "2134567"},

{ Rental_SID: “2307111”,

Pickup_Date: ISODate(“2014-08-25”),

Branch_SID: “3374”,

Branch_Name: “Aurora”

Dropoff_Date: ISODate(“2014-09-02”),

Branch_SID: “3374”,

Branch_Name: “Aurora”,

Vehicle_SID: “2134977”},

] }

Abramova, Bernardino, & Furtado (2014) conducted performance testing using five different NoSQL databases that included the testing of MongoDB. They found that MongoDB had the largest increase in run time (slow performance) when the number of updates increased. MongoDB uses a data locking mechanism that has the direct effect of slowing update performance. However, in regards to read performance, MongoDB performed very well and is considered a database that is optimized for reads. The authors did not provide details about how the test data was modeled (e.g. embedded or referenced).

Conclusion

NoSQL databases are an important component of Big Data for storing and retrieving large amounts of data. RDBMS use the ACID theorem for data consistency, whereas NoSQL Databases use BASE. RDBMS scale vertically and NoSQL Databases can scale both horizontally (sharding) and vertically. Although NoSQL databases provide performance gains, some researchers are cautious and skeptical about data consistency. This paper describes the four types of NoSQL databases: Document-oriented, Key-Value Pairs, Column-oriented and Graph. Data modeling for Document-oriented databases is similar to data modeling for traditional RDBMS during the conceptual and logical modeling phases. However, as was demonstrated in this paper, physical data modeling for a document-oriented database is different. Separate entities can be merged (denormalized) into one document by using embedding and the concept of a foreign key is supported by a reference.

References

Abramova, V., Bernardino, J., & Furtado, P. (2014). Which NoSQL database? A performance overview. *Open Journal of Databases (OJDB), 1*(2), 17-24.

Cloudera. (2014). *Cloudera Big Data and Hadoop Sessions*, May 21, 2014. Location: Hyatt Hotel, Denver, CO.

Devlin, B. (2014). *Business un-intelligence: The marriage of BI and Big Data*. ACM Webinar on June 17, 2014.

Gartner. (2014). *Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business*. Retrieved from http://www.gartner.com/technology/home.js

Hoberman, S. (2014). *Data modeling for MongoDB: Building well-designed and supportable MongoDB databases* (1st ed.) Basking Ridge, NJ: Technics Publications. ISBN: 978-1-935504-70-2.

Leavitt, N. (2010). Will NoSQL databases live up to their promise? *Computer*. Published by the IEEE Computer Society. p. 12-14.

Medina, J. (2014). *NoSQL 133 success secrets – 133 most asked questions on NoSQL – What you need to know* (1st ed.). Rock Hill, SC: Emereo Publishing. ASIN: B00QE3WDDO

Mohan, C. (2013). History repeats itself: Sensible and NonsenSQL aspects of the NoSQL hoopla. *EDBT/ICDT 2013 Joint Conference, March 18–22, 2013, Genoa, Italy.* ISBN: 978-1-4503-1597-5.

Moniruzzaman, A. B., & Hossain, S. A. (2013). NoSQL database: New era of databases for big data analytics - Classification, characteristics and comparison. *International Journal of Database Theory and Application, 6*(4), 1-14.

MongoDB. (2014). *The MongoDB 2.6 manual*. Retrieved from http://docs.mongodb.org/manual/core/introduction/.

Ohlhorst, F. (2013). *Big data analytics: Turning big data into big money*. Hoboken, NJ. John Wiley and Sons. ISBN: 978-1-118-14759-7.

Roe, C. (2012). *ACID vs. BASE: The shifting pH of database transaction processing*. Retrieved from http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/

Sadalage, P., & Fowler, M. (2013). *NoSQL distilled: A brief guide to the emerging world of polyglot persistence*. Upper Saddle River, NJ: Pearson. ISBN: 978-0-321-82662-6.

Sharda, R., Delen, D., & Turban, E. (2015). *Business intelligence and analytics: systems for decision support* (10th ed.). Upper Saddle River, NJ: Pearson. ISBN-13: 978-0-13-305090-5.

Biography

Bob Mason joined Regis University as a full-time faculty member in January of 2011 after completing his Ph.D. at Nova Southeastern University located in Davie, FL. Bob is the program coordinator for two programs: MS in Database Technologies and MS in Software Engineering and Database Technologies. Prior to accepting this position with Regis, Bob was employed by various Fortune 500 companies for 25 years as a Data Architect, DBA and Software Engineer. He also was an affiliate faculty member at Regis in the area of Database Technologies for 10 years. Bob has just completed a new course for the Regis University CC&IS MS in Database Technologies degree program called Introduction to NoSQL Databases that covers the four types of NoSQL databases with hands-on lab exercises. Courses on specific NoSQL databases, such as Cassandra, MongoDB and Neo4J will be available soon within CC&IS.