APPROVAL SHEET

Title of Dissertation: Convergence-Directed, Semantic Model for Integrating Large-Scale, Dynamic, and Heterogeneous Databases

Name of Candidate: John W Hebeler
Doctor of Philosophy, 2017

Dissertation and Abstract Approved: per Approval Form
Dr. Zhou
Professor
Department of Information Systems

Date Approved: 7/27/2017

ABSTRACT

Title of Document: Convergence-Directed, Semantic Model for Integrating Large-Scale, Dynamic, and Heterogeneous Databases

John W. Hebeler

Directed by: Dr. Lina Zhou and Dr. Victoria Yoon

Data abounds today. The ability to properly employ large amounts of data for business decisions and opportunities is critical to business success. Rarely does a single data source or database prove sufficient for a dynamic, unfolding business action. Business decision-making requires integrating data quickly and efficiently. However, the inherent differences between databases make it both time-consuming and costly to achieve a useful integration. More importantly, the storage of much of today's data has migrated away from the traditional, relational database technology and switched to NoSQL database technology. The latter provides a new set of challenges for data integration.

To address the above challenges, the semantic integration model offers a path to simplify integration across different NoSQL databases. It achieves this through an iterative, incremental integration that directly involves the non-technical business user – the key to exploiting the business opportunity. The model contrasts current integration methods and is evaluated against a prototype that implements and tests the model with appropriate data participants. The model demonstrates an easier method to quickly review potential data integration candidates, integrate selected candidates, and maintain the alignment of the data integration with the evolving NoSQL technologies and the business opportunity itself.

Convergence-Directed, Semantic Model for Integrating Large-Scale, Dynamic, and Heterogeneous Databases

By John W. Hebeler

Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, Baltimore County, in partial fulfillment of the requirements for the degree of Doctor of Philosophy
2017

Table of Contents

1 Chapter 1: Introduction

1.1 Problem Context

Business opportunities abound. Today's assembly of massive data offers to reveal and aid the advancement of a business opportunity. "Data are to this century what oil was to the last one: a driver of growth and change" (Economist, 2017). This data often resides in NoSQL databases – databases especially designed to ingest, store, and retrieve massive quantities of data. NoSQL represents a new category of database technology that focus on efficiently managing large data sets at the expense of some capabilities offered by relational databases. The NoSQL Market is expected to garner $4.2 billion by 2020 (Allied Market Research, 2015). However, the massive data stored with NoSQL technologies unintentionally creates a maze that challenges a business user to properly integrate the data for a given opportunity within the necessary timeframe and cost. The business user must turn to rare, technically skilled NoSQL resources that, themselves, struggle to keep up with the rapid and proprietary advancements within each NoSQL database including a non-standard method to dynamically extend the structure throughout the database. NoSQL data integration requires many translations between the business requirements and the technical implementations, which may further impede the use of data. These integration challenges lead to less use of the available data and thus deprive business decisions of a more, data-enriched view. A model and method that can close the gap between the opportunity and the relevant data for business users via minimizing the required technical skill and time would lead to more informed and timely business decisions.

Figure 1 outlines the data integration challenge. A business user familiar with a given business domain and associated potential opportunities seeks information to refine their business actions. This information typically resides in various databases. With extensive technical assistance, the business user must gather, integrate, query, and determine completeness of the data to aid their actions. Each step requires a translation between the business domain and the technical requirements. Additionally, information builds on each other. This requires multiple passes through the integration process. Due to the technical demands, the business user has to await the final integration outcome without the ability to provide any input during the actual integration. This all leads to a protracted, expensive integration process that can easily miss the integration of valuable information.

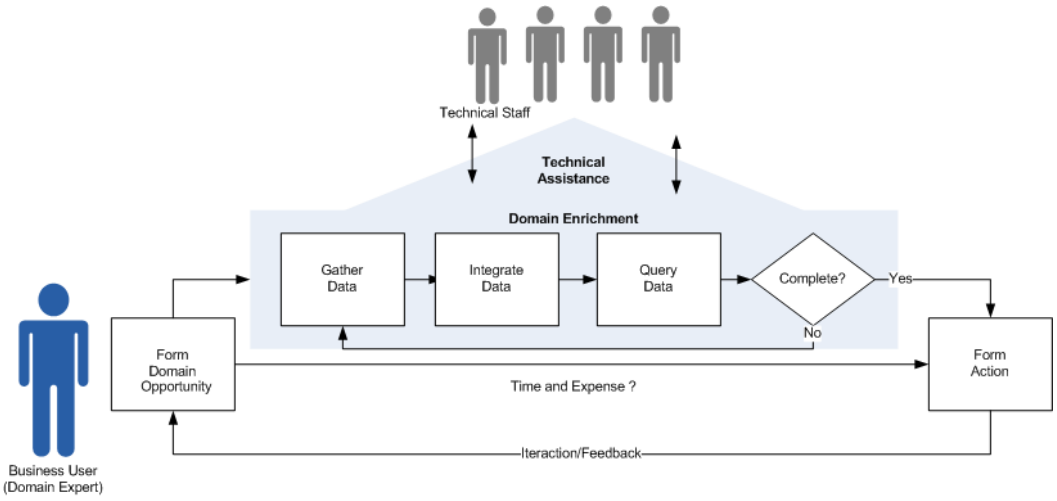

Figure 1: Data Rich Decision Convergence

Given the above business context, what methods help accelerate the process of turning a data-rich opportunity into action, especially in the era of big data?

Data integration is generally defined as the process of combining data residing in different locations and technologies into a unified, useful form able to increase the value of the data previously in isolation (Lenzerini M. , 2002). This process includes data access, resolution of data structural differences, and data conflict resolution (Ziegler & Dittrich, 2004). The ability to combine large data sets covering several domains can increase the value of data owing directly to the integration. Effective integration enables a business user to consider all of its customers, suppliers, expenses etc. to provide a clear picture of the overall business or a specific area, such as how customers in one location differ from those in another location. The integration of existing data is becoming ever more indispensable in order to meet the business and customer needs (Ziegler & Dittrich, 2004).

Data integration galvanizes disparate data into more useful business artifacts - an artifact that increases the value of the data, and in turn the corporate value. Corporations invest heavily in data integration. Investment runs approximately 40% of a typical Information Technology budget. Additionally, the market for tools that support integration runs in excess of $6.44 billion in 2017 and is estimated to grow to $12.24 Billion by 2022 (Markets, 2017). This level of effort and investment provides strong evidence for the need of the modern enterprise to effectively and efficiently integrate data. Rationales for integration fall into three main categories: revenue increase, increasing customer satisfaction, and costs savings due to process improvement (Gold-Bernstein & Ruth, 2004).

Data continues to amass, which has brought about the big data era. Big data enables expansive reaches into virtually all domains. However, its value often depends on the ability to evaluate and integrate related data across multiple databases to form a high resolution, comprehensive picture of a given domain of interest (Brodie, 2010). Data integration always faces tough mismatches and heterogeneity problems across different interfaces, technologies, structures, formats and semantics. Big data not only faces challenges similar to challenges those found in traditional relational database integration, but also extends them with new challenges including sheer scale, dynamic structures, and a growing number of accessible NoSQL databases - many of which are outside of the user's control. This requires new ways to quickly integrate potential database candidates. Without such advances, many NoSQL data sources may remain unused and isolated, losing the potential value of integrating them into a larger knowledge base.

There are numerous real-world examples of data integration to create business value. For instance, a company can combine its marketing database, which contains details regarding multiple marketing campaigns with a customer-ordering database to determine the effectiveness and efficiencies of a specific marketing campaign. Medical companies could use integration to combine drug research findings with electronic patient records to determine the efficacy of a drug treatment program. Some values of integration may not be realized until the integration takes place and valuable information is discovered.

Effective data integration creates value and leverages the growing richness of accessible data or information. However, traditional integration methods remain labor intensive,

time consuming, costly, highly specialized, and error prone. This underlies many failed integration efforts which are often caused by inherent complexity, delays, and costs.

Even after successful data integration, businesses struggle to focus the integrated information on the specific domains of interest. The sheer magnitude of the unified data can actually obfuscate the integrated value. Extraction of information from big data requires another level of complexity. Additionally, it is difficult to deal with ad hoc or serendipitous data discoveries – most go overlooked and are buried under the data mass. The traditional integration approach focuses attention on the integration effort itself, while missing many potential discoveries after the integration is completed. Data scale, semantics, and specific data needs quickly overwhelm traditional integration methods, which is typified as *data overload.*

The advent of big data with its associated NoSQL stores provides a new foundation for useful, powerful data integration. NoSQL stores bring not only new business opportunities but also new technical challenges to effectively integrate data across different NoSQL stores. NoSQL stores allow unprecedented data storage and retrieval. However, these technologies are mired in proprietary interfaces, decentralized and dynamic data structures, and evolving rapidly – all of which make data integration more difficult.

1.2 Research Issues

The increase of data sources, technologies, and business dynamics opens up new possibilities for data integration. Research that advances data integration increases the

value of existing data and the corresponding business value. Data integration begins with a data review by answering the following questions: What is in the data? Is it worth integrating the data into a larger data picture? In order to review a data source for integration, one must first understand what it contains and the potential role it plays in a business opportunity. Proper review of a data source helps collect the requirements of an actual integration and minimizes the time and resources needed. Similarly, after data is integrated, does the integrated database meet the data needs of an opportunity? What makes data integration so difficult? In order to better understand the key issues regarding data integration, an ideal integration model outlines the essential data integration requirements.

1.2.1 Ideal Integration Model

Figure 2 outlines the ideal integration model that covers the critical facets of data integration (Ziegler & Dittrich, 2004). All data integration efforts attend to each of the areas to some extent. The diagram decomposes the integration into five major areas: transformation, representation, unification, alignment, and adaptability. Transformation extracts the native database data and structure into a normalized common form. Transformation coverts the data across four levels – foundation that represents the underlying operating environment (e.g Mac OS), the technology that represents the actual database technology (e.g. MongoDB), the format that represents the arrangement or structure of the data (e.g. JSON), and finally, semantics that represent the meaning or purpose of the data. Representation forms the data into a common, understandable form, independent of specific database technologies nomenclature. Therefore, representation

properly represents all the information and its context across the various database technologies. Unification normalizes the common formatted data. This requires identification, correlation, cleansing, and summarization. Unification creates a useful combined data source. Alignment produces purposeful information from the integrated store based on user and/or system needs. Alignment focuses the information on the intended domain of interest requested by a system or user.

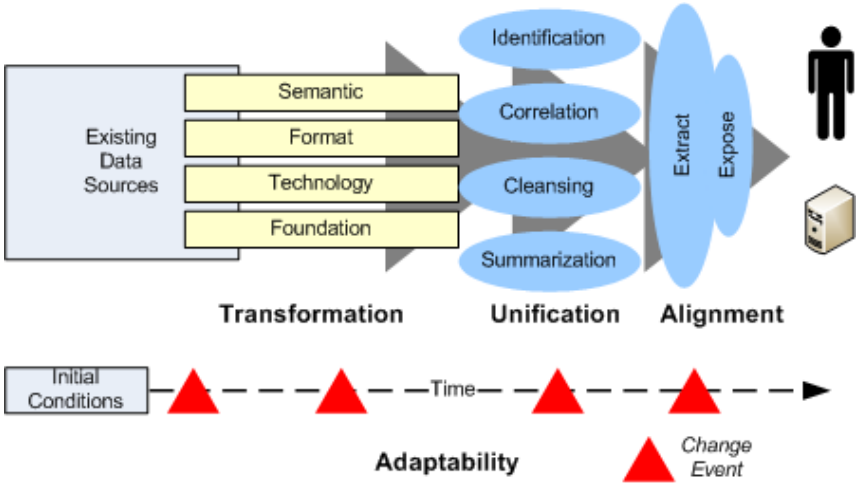

Figure 2: Ideal Functional Data Integration Model

A final consideration of the ideal model is the time and cost to produce the integration including changes that may occur to various data sources and business opportunity domains. Integration that is more costly than the perceived integration value and/or takes longer than the opportunity permits is simply not pursued. Therefore, it is critical to consider the time and cost factors. Additionally, the integration faces likely change events such as modified data structures, upgraded technologies, and changes in the business opportunity domain. Each area can introduce dynamics or changes to the integration solution. Therefore, the ideal solution considers the time and cost not only for

the initial integration solution but also for maintaining that solution over time. The combination of the initial and dynamic factors into a final ideal model introduces *adaptability*. The solution satisfies the initial conditions and then continuously aligns to changes and/or dynamics in the evolving business opportunity.

Thus, the ideal model summarizes data integration into five integration themes: transformation, representations, unification, alignment, and adaptability. This ideal model assembled from various research efforts allows comparisons among various integration methods to highlight the capabilities, priorities, and limitations of each approach. The model helps to reveal the strengths and weaknesses of various data integration approaches.

Finally, it important to note the role of the business user - the non-technical user, driving the integration. They maintain the critical insights as to the value of the integrated data. Their involvement at the various phases often determines the ultimate value of the integrated data. If their involvement is limited, there is an increased likelihood that the integrated data will not meet their needs. This leads to additional integration efforts that delay and increase the costs of the proposed business opportunity.

1.2.2 Traditional Data Integration Model

The traditional data integration model emphasizes the *technical* transformation of data from a source system to the integrated system in a controlled sequence. Traditional integration considers low level details and offers either a point-to-point integration or a form of data centralization such as data warehousing. Integration occurring at this

technical level focuses on syntax, formats, and technologies without regard to semantics. Integration typically takes two routes – various Extract, Transform, and Load (ETL) approaches that transform and move the data to a central store (or warehouse) and a federated query approach that first decomposes the query to integrate the requested data source subsets and then unifies the results from multiple queries. Both focus on the technical levels, bypassing much of the inherent knowledge and information behind the data that often remains left behind in the source application. Data integration systems actively combine the data primarily from two perspectives: Global as View (GAV) and Local as View (LAV). While GAV places the integration complexity on the mediation between source and unified schema or domain, LAV places the integration complexity on the mediated queries. GAV works best with changing domains whereas LAV works best for changing integrated data sources (Bennet & Bayrak, 2011). The new application incorporating the integrated data takes on much of the knowledge responsibilities within its processing steps outside of the integrated data. This further diffuses the knowledge value of the data.

Traditional data integration methods address data integration with an a priori understanding of the data and its purpose (Cure, Lamolle, & Le Duc, 2013). Due to inherent technology hurdles, technologists perform the integration tasks somewhat isolated from the business needs. This can result in data marts and data warehouses that do not align with the business requirements. This problem is not uncovered until the integration process is completed. Big data/NoSQL integration needs to go beyond these methods due to its dynamics, size, and lack of control. Big data often forms a massive and dynamic picture of the data. Scale and time demands prohibit copying the data to

another source, thus preventing the construction of data warehouses with their inherent latency. Dynamics enable fundamental structural additions at any time and location within big data making it better reflect the current domain while keeping the user of the data aligned with these changes. These big data integration challenges quickly overwhelm traditional integration approaches leaving many integration possibilities out of reach (Dong & Srivastava, 2013).

Figure 3 captures the essence of the traditional approaches into an integration model that highlights the five integration themes from the ideal model.

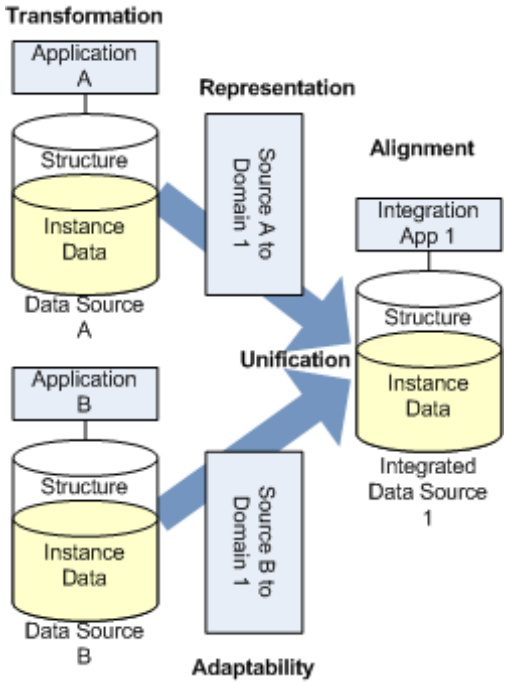

Figure 3: Traditional Integration Model

The traditional model transforms and unifies the data based on the technologies below the semantics. The transformation, representation, and unification approaches are typically

highly coupled. The fundamental differences between the various traditional integration approaches is when the extract, transform, and load operations occur. In a centralized data warehouse, they occur prior to any requests or queries. In a federated query, the ETL occurs in response to a query. These differences impact performance and resource needs. Finally, the WAV and LAV approaches design various mapping strategies to map the data from original source to the integrated data source. Note in all cases, the data interacts directly with the database bypassing the application. This forces the model to deal with low-level details contained in the database and the integration model loses the data context contained in the source application. Applications often initiate complex efforts to enhance and relate the data contained in the data source. This creates the necessity to build another application to provide such context for the integrated domain data. This reflects the reality of most data systems being application-centric rather than data-centric. The latter would consider uses of the data beyond the application and could actually assist in the integration.

Data integration for NoSQL stores remains in its infancy. The scale often prohibits the ETL transformations allowed in smaller stores due to the inability to perform a timely data transfer and creates inherent latencies. Additionally, the NoSQL stores handle data structure in a non-centralized, dynamic fashion. NoSQL stores do not offer a standard, description of the data structure because the structure can be defined anywhere within the store and is dynamic. Consequently, many traditional integration approaches would fail due to their sheer scale and/or structural dynamics.

1.2.3 Traditional Integration Model Limitations

The traditional model based on an application-centric view with limited semantics results in significant limitations for integration efforts. The following details the key limitations organized in the five themes presented in the ideal integration model.

- Transformation Limitations
    - Limited Data Expressiveness: The model lacks an expressive data format that fully captures the richness of the data. This requires the diffusion of this expressiveness throughout the integration application and often pushes low-level integration issues onto the user. (Date, 2005) (W3C S. , 2008)
    - Application-centric data resources: The current model must consume data resources that are application-centric rather than data-centric. This buries the data in a complex application context that makes it very difficult to extract the data minus the encumbrance of the application context. In general, data systems are not designed for integration (Ziegler & Dittrich, 2004).
    - Syntax, Format, and Technology Integration: The traditional model integrates at a level that incurs a high degree of complexity and must deal with the major differences across the many existent syntax, formats, and technologies that store and manage data. (Ramler & Wolfmaier, 2008). The lack of semantics forces technicians to perform the integration rather than the actual users.

- Big Data Scale: The sheer scale of big data often disallows transforming the data. The NoSQL database, Cassandra, holds data for Apple Computers and Netflix. The Apple database contains over 75,000 nodes storing 10 Petabytes. The Netflix database contains 2,500 nodes, 420 TB and over 1 trillion read requests per day (Cassandra, 2016).
- Big Data Structural Dynamics: NoSQL databases do not offer a standard method to extract the structure of stored data. The NoSQL model does not require a data structure upfront. Data personnel do not have to design the structure in advance of the data ingest. (Wayner, 2012).

- Representation Limitations
  - Single Domain View: The unified information need to fit into one domain view. This causes integration difficulties when there is a lack of agreement across various meanings of the labels and terms due to different perspectives found in different databases. Merging these different perspectives encumbers the integration process and limits domain specialization in retrieving information from the integrated database, which obfuscates the benefits of integrated information (Lukovic & Mogin, 1996).
- Unification Limitations
  - Limited ability to deal with dirty data: Dirty data includes missing information, conflicting information, and incorrect information present when bringing multiple data sources together. It presents a huge burden to corporations with over $600 billion spent on data quality issues (Lucas,

2010). Traditional methods deal with them *a priori* to the actual integration. This creates high risk because it is difficult to identify and fix all the dirty data problems prior to the integration. This often leads to high cost due to last minute efforts that can also induce additional errors. This limitation significantly inhibits large-scale integration since dirty data is constantly present and it is virtually impossible to remove all of it (Fan & Gui, 2007) (Kim, Choi, Hong, Kim, & Lee, 2003). Additionally, NoSQL stores assume that dirty data always exists and their sheer size prohibits efficient cleanup efforts.

  - Complex Relationships Among the Data: This requires a complex technical understanding of the data such as foreign keys, triggers, field constraints and so on, creating data models that require a strong expertise to understand, evaluate, and integrate. Names of various artifacts to accommodate these complex structures come at the expense of clear, easy-to-understand semantic labels.

- Alignment Limitations
  - Failure to Evaluate each Data Candidate Early: The business user does not see a database's integration contribution until the integration is complete. This can lead to incorporating unnecessary data and delays the addition of useful data (Dayal, Castellanos, Simitsis, & Wilkinson, 2009).
  - Limited Support for Domain Extraction: The ultimate goal of data integration is to extract useful information. The actual integration effort merely puts all the data into a single unified accessible form. It does little

to establish specific domains of interest to help identify and extract desired results. The specific domains must compromise themselves into the one unified view and handle unique data considerations outside the main integration store (Lukovic & Mogin, 1996).

- Limited Integration Standards: Traditional integration lacks well-accepted integration standards, making it difficult to extract key information and share integration approaches. The inability of integration solutions to build on one another results in increased costs, protracted delays, and higher risks.
- Limited Reasoning: Traditional methods lack logical reasoning that can assist the integration in handling error correction, removing duplicates, replacing missing information, and resolving conflicts. Reasoning can also supply independent alignments of the data allowing focused independent domains.

- Adaptability Limitations
  - Closed World Model: Traditional efforts close the data model making future changes difficult. For example, a closed model assumes that it has complete information. If a customer's name is not present in the database than they are not a customer. This closed assumption makes integration with another customer database fraught with possible conflicts. Unexpected data changes create havoc with the integration causing the project major maintenance costs and/or the inability to adapt to the change.

This includes inabilities to properly deal with discovered poor data quality and future data integration candidates.

- Highly labor intensive and Error Prone: The many manual steps in integrating the various technical, format, and syntax complexities invite errors and are time consuming.
- Isolated Integration Efforts: Due to the lack of framework that can help simplify integration, each integration effort is unique and isolated. Different efforts are unable to share data structures and terms. Thus, every effort starts from scratch and is unable to build on previous integration efforts.
- Point-to-Point Integration of Databases: Integration is often forced to uniquely connect one data source with another one in a point-to-point fashion or create an accepted standard database that integrates the data. The former creates the classic N^2 dilemma and its associated inefficiencies. The latter creates data warehouses that demand a common, accepted data model. Either approach can work on small, fully controlled data sources but quickly fails with many data sources and/or multiple, complex data perspectives that make common models infeasible.
- Brittle, Non-Adaptive Integration Solutions: The complex integration wiring and transformation working at the low levels forms a very brittle solution that inhibits change. Brittle solutions break or fail with even minor changes. This often results in paralyzing the various data sources and associated applications involved in the integration. They in turn lack

the ability to adapt to changing business demands. As a result, new demands may require entirely new data sources (new data silos) or large maintenance efforts on the existing integration solution (Bennet & Bayrak, 2011).

- Extensive Duplication: Traditional efforts often rely on a centralized data approach that requires the movement and duplication of the integrated data. This incurs the expense of moving and duplicating the data and also creates data latency. The latency causes the actual data and the copied data to possibly be out of sync with the original data source, which can produce data inaccuracies and anomalies. This limitation grows more serious as the demand for real-time, dynamic data increases.

These limitations result in negative business impacts for data integration that include high cost, protracted time lines, and potential errors. It also requires high technical expertise and incurs a large delay with iterations throughout the integration process due to learning and/or changes. Hence some of the value of the data and the business opportunity is lost.

1.2.4 Integration Trends

Current technology and data trends drive additional demand for integration while placing new requirements on integration solutions. The trends move towards easily accessible and dynamic data.

- Plunging Cost of Infrastructure: Data requires movement, processing, and storage. In accordance with Moore's law and others, costs continue a precipitous plunge with increasing data availability.
- New Data Structure/Storage Options: Many new forms of data storage (e.g. NoSQL) and their associated diverse structures have emerged in cloud computing, Internet storage, and the like. Consider HBase, Cassandra, Neo4J, along with the movement to make large data sources available to the public (Government U. S., 2011) to name but a few. Many are rapidly evolving.
- Growing Participation in Producing Data: More parties including individuals, corporations, and governments contribute to producing the data. Many are quite new to these contributions. Worldwide Internet participation has grown over 500% in the last ten years. The United States has over an 84% penetration rate with many countries over 90% (Internet World Stats, 2017).
- New Data Technologies: New technologies from cheap cloud data storage to simple-to-use blogs, wikis, and short message technologies encourage the creation and accumulation of data from many parties including technology novices.
- Development of the Semantic Web: The Semantic Web, now over a decade old, has matured to include a wide array of tools, technologies, and semantic data sources. As of 2010, Linked Data contains a web of over 26 billion statements with over 400 million cross-data set links (Bizer & al, State of the LOD Cloud, 2011) (Yamaguchi & al, 2011). The Semantic Web continues to evolve and grow to drive greater use of data and easier integration. Additionally, the Semantic

Web contains strongly supported standards for knowledge representation including logic and reasoning.

- Real Time, Dynamic Data and Associated Structures: This changing data often demands dynamic changes in its storage structure, thus evolving the schema that stores the data. Twitter serves as an example of new types of real time data – data that forms powerful insights into what is happening in real time (Yamaguchi & al, 2011).
- Increasing Dirtiness of Data: With data emerging from multiple sources many of which lack coordination, certification, and/or the same perspective causes a growing list of data conflicts, errors, and missing data for an integration effort.
- Growing Base of Structured Data: Data is quickly moving from unstructured, human-consumable forms to structured forms capable of improved system interrogation and integration. This is seen in several efforts to provide access to machine-readable information such as Linked Data (Heath, 2011).
- Increasing Distribution of Data: Data with its large resource needs and various localities continues to be more fragmented and distributed across the various geographies and topic domains.
- Growing diversity of the data type: Data is no longer merely strings and numbers. Critical data also exists as videos, pictures, maps, and audio.
- Increasing Value of Just-in-Time integration: The need for rapid integration of data to reveal value, if any, will be a significant factor (Zhu, 2008). There is no way to know the value before-hand, in many cases. This makes it impossible to

make a traditional business case regarding integration since the value cannot always be predetermined.

- Growth of Proprietary Integration Tools: Many new integration tools are emerging to serve the many integration opportunities. However, they lack standards, require translation into a proprietary format, and are often very expensive (SAS, 2011). This approach can lead to isolated, expensive integration solutions.
- Growth of data outside of a business's and user's direct control: This requires careful inspection and review for possible integration.

These trends offer new challenges and benefits for integrating the data.

1.2.5 Specific NoSQL Issues

Big data finds storage in non-traditional technologies known as NoSQL storage. NoSQL is designed around the benefits of big data. First and foremost, the NoSQL technologies scale data storage across many physical devices allowing massive data storage. NoSQL technologies also offer a dynamic structure as opposed to the rigid structure imposed by traditional relational databases. This allows the database to remain relevant as it adapts to new structural requirements without the need to discard or update the massive amount of existing data already present in the NoSQL database. These two forces create unique features for data integration that are critical for integration success.

1. Scale: The sheer size of big databases prevents the easy movement and creation of separate databases as employed in data warehouses and data marts. The time and resources required often eliminate this as a possibility.
2. Structure: The sheer volume of data often simplifies the structures to allow for efficient introspection. They rarely have deep, complex relationships within the data elements, as this creates large inefficiencies with reading and writing operations. Additionally, the simplicity of the structure lends itself to be semantically rich with meaningful, useful named data labels throughout while avoiding complex interrelationships between the data and the associated complex names.
3. Dynamic Structure: Structure definitions can exist anywhere within the store. Frequently, no single method exists to extract the structure. This allows updating the structure and creating new structural elements at any time and at any place within the store. Conversely, it makes the actual structure or structures difficult to obtain. Structure is the key to understanding the contents of the NoSQL store.
4. Evolving NoSQL Technology: Technology continues to evolve to meet the growing understanding of big data needs, creating a challenge for data integration methods that attempt to integrate while technical changes occur.
5. Proprietary Technology: Each NoSQL technology is proprietary and there are no current efforts to standardize the technologies to allow easier integration.
6. Big Data Ownership: Due to the size and scope of big data, integration efforts may require integration with an external data source that is outside of the effort's control. Therefore it could change significantly without any warning.

7. Outdated Documentation: Due to the scale, dynamics, and exigencies of NoSQL databases, documentation often lags behind the NoSQL store or doesn't exist at all.
8. Scarce expertise: NoSQL demands new technical skills that are very different from traditional database skills. The need for these skills far outstrips the demand, placing further burdens and costs on integration efforts.

1.2.6 Issue Summary

The inefficiencies and ineffectiveness of the traditional integration model result from its low level, technical-oriented approach, extensive manual intervention, and the high level of integration customization required for each effort. The traditional integration model may fail to reap much of the value from existing data sources and is poorly positioned to serve the rapidly evolving integration opportunities with NoSQL databases.

The traditional integration model restricts integration to its technical components and pushes the semantics or meaning of the data to outside application and/or the users. This delays not only the integration but also, more importantly, the review of the data's role in a business opportunity. So, while users require useful semantics, the integration effort is moribund in technical challenges and jargon. The challenges extends as big data plays a larger role. NoSQL stores are simply too large to move, do not easily reveal the contained data's structure, and continue to evolve in non-standard ways. Yet the large stores of data can offer benefits to many business opportunities.

The trends clearly lead to more data, in more diverse formats, at more locations with more participants. More and more of this data will be machine consumable in ways that potentially ease the complexities and delays surrounding integration. Integrating this growing amount of data potentially leads to improvements in the quality and timeliness of opportunities that benefit from the data.

1.3 Research Questions

Big data stored in NoSQL requires integration approaches that evolve to better capture the value of the opportunities for large-scale, dynamic data distributed in many formats and technologies across many domains. This research examines methods to improve integration and better position data to unleash its potential. *The fundamental challenge is how to enable a business user to quickly integrate multiple big databases to aid a business decision.*

The research develops a data integration method based on early, continuous, and direct interaction of the non-technical business user with the databases by revealing the meaning, or semantics of the data from a business perspective rather than requiring a technical translation. Semantics references the meaning of the data. This allows integration to occur at a business level to potentially reduce cost, increase timelines, and effectively deal with the multitude of dynamic, diverse data sources. Semantics hit at the heart of the value of the integrated data – its meaning. Early semantic transformation enables the user to quickly review the data prior to the technical integration. This review includes specific tables and fields in the data, critical in forming integration with large

databases. As the integrated database grows, the user can then continually review it to see if it meets their needs. This allows an iterative process that continuously enriches the data integration until the user is satisfied. It also allows the user to add data later to the integrated database as the business opportunity arises.

The overall research question is this: How can data semantics be used for integrating large scale, diverse, and dynamic NoSQL data sources? To answer, the following specific questions must be addressed:

1. How can *semantic transformation/representation* provide a useful view of the underlying database for integration?
2. How can *semantic unification* integrate various NoSQL databases?
3. How can *semantic alignment* focus the unified semantics on a specific domain of interest?
4. How can *semantic adaptability* using *convergence-directed integration* produce an efficient approach to NoSQL data integration?

1.4 Research Objectives

1.4.1 Overall Objective

The research focuses on integration solutions based on semantics. The research will advance the current integration model in two key areas: incorporating semantics to advance integrated data and integrating the data on the rapidly emerging big data stores found in NoSQL databases. Five major steps are required to advance NoSQL integration

– *transformation*, *representation*, *unification*, and *alignment* toward a given domain of interest and *adaptability*. Finally, the research employs multiple methods to validate the generalization of the semantic approach to ensure proven integration advancement.

Integration using semantic technologies is not a new topic. Existing research has touched on this and identified it as the key issue in data integration. Semantic integration falls into three categories: programmatic semantics, schema-based integration, and declarative approaches (Zhou, 2010).

Recent advances in the Semantic Web enable advances in semantic integration. The Semantic Web offers a supported standard for data representation, logic, rules, and remote/local access (Organization, Semantic Web Standards, 2001). These standards have produced two valuable results – a multitude of tools and a vast array of available Semantic data. The data includes sharable structures and easily integrated data, allowing integration efforts to work in collaboration with a wide array of tools *and* information.

The Semantic Web advances combined with the current trends open an opportunity to advance and implement a proposed semantic integration model through the integration and development of various Semantic Web tools and available Semantic Web information to produce useful business information from large-scale, dynamic data integration.

The research will produce a working model based on existing Semantic Web tools (and by developing additional tools and sources where necessary) that validate an adaptive,

cost-effective model and method to integrate large-scale, dynamic information sources based on semantics.

1.4.2 Transformation and Semantic Representation

Semantics integration begins with a transformation of NoSQL databases into a representation based on a semantic view of the contained information. This process takes the basic data and enriches it with discovered semantics. It takes three steps – determination of the specific NoSQL technology, for each NoSQL technology maintains different interface protocols; the extraction of the information necessary to describe the NoSQL database; and the transformation into a common semantic form.

The semantic integration method contains the protocol information for each supported NoSQL technology. The method then probes the candidate database for a match to the protocol. Once a match occurs, the method interrogates the candidate database for descriptive information such as the structure, size, and technical parameters. NoSQL data sources hold structural information in each record. Each record could potentially contain a different structure. NoSQL technologies do not offer a standard method to extract the structural information, as do relational databases. Therefore, existing research methods employ an exhaustive read of every record. This approach does not scale to the potential size of a NoSQL database, which could reach many petabytes. The semantic integration method employs a statistical sampling of the database records rather than an exhaustive read of the records. This avoids the time and processing delays that result when a candidate data source scales to its potential, which could reach many petabytes.

Finally, once the structural information is uncovered, it is combined with data examples for each structure so as to reinforce and expand the meaning behind the structural terms. Additionally, the technical integration information such as the login, network location, query parameters, and NoSQL technology type is recorded. All of this information is transformed into a semantic ontology. The ontology describes the NoSQL database and contains all the information necessary to obtain information for the integration of a candidate database.

This semantic richness of the ontology varies depending on the level of semantics available in the underlying technology through the naming of the various data artifacts tables and fields. Secondly, the semantics can be enriched and expanded through a variety of semantic analytics and manual assistance, many of which are available and becoming standardized (Euzenat J. , 2008), such as using reasoners and/or adding enrichment ontologies. Lastly, the ontology holds meta-information regarding the NoSQL database to enable the location and extraction of information using its native methods such as the technical location, login information, and query parameters. This allows the actual data to reside in the native store with the semantic meta-layer controlling the meaning and access methods, thus eliminates the need to copy and synchronize the data to a central integration database. The ontology serves as a proxy to each of the integrated native NoSQL databases.

1.4.3 Semantic Unification

Once an enriched semantic ontology is established from a database, the ontologies combine at the semantic level to form unified data integration. There is no need to combine the actual data until requested. Ontologies easily combine and avoid conflicts found in traditional relational databases. Once combined, semantic analytics perform conflict resolution, duplication removal, correlation, and the like. Unification normalizes the integrated data. These steps can continually reach out to ontological resources to refine, focus, and maintain relevancy over changing data sources and domains of interest.

1.4.4 Semantic Alignment

Semantic alignment focuses the integration effort for a given domain of interest. Semantic alignment occurs in two phases: during unification and when providing results in a domain of interest. Given the focused domain of interest expressed via semantics, the domain offers powerful techniques to extract desired information and patterns. Semantics offers a triad of extraction techniques that can be used together or in isolation depending on the information needs. This includes search, navigation, and query.

Search enables basic Boolean search based on keywords without semantics providing a simple way to find focus data areas for an information quest. Navigation follows Semantic links that enables semantic results to follow a related path to additional information. Semantic filters can clear away superfluous information and highlight paths related to the request. Finally, when a path becomes one of interest or a path is known *a priori*, the request can be captured in a formal query via a query language.

The interactions can actually change the domain of interest, which would work with the other two areas above to continually adjust the domain and its associated data sources to align with the requestor's needs.

The four steps outlined above - transformation, representation, unification and alignment - all take advantage of the flexibility inherent in the semantic integration model. This includes organic assembly of an initial integration, automatic enrichment, and flexible formation of semantic data extraction, allowing the semantic integration model to form quickly and maintain alignment with both the data sources and the domain of interest.

1.4.5 Adaptability

A modern data integration system must be adaptive. Adaptability enables a quick start, maintains alignment, and protects the investment in an integration effort.

1. Adaptability enables a quick jump-start for the integration effort, allowing a rapid review to determine the value of the integration effort. Many integration efforts may not see the end value until the integration occurs. Thus, without a quick easy start, many integration efforts will not go forward.
2. Adaptability aligns the integration effort with the data and domain dynamics. This is key in any large-scale effort since the likelihood of change increases as the size and number of integration databases increase. This is also compounded when data sources are out of the control of the integration effort.

3. Adaptability preserves the investment in the data and integration by maintaining an up-to-date view of the data and allowing the data requirements to change without major efforts.

Adaptability employs convergence-based integration that occurs throughout the integration development process. Traditional solutions that lack adaptability often fail to meet the business needs and fail to incorporate business learning along the integration steps.

Adaptability throughout the process overcomes these limitations. Semantics reveal underlying data meanings to the user allowing a degree of convergence to occur at early stages including the initial review of the data. The user can direct the integration effort as the semantic revelation uncovers both pertinent and non-pertinent data. This allows some degree of requirements, development, and extraction to occur simultaneously by the user who is interested in the actual meaning or semantics of the results. Additionally, the technical underpinnings of the Semantic Web can also help the convergence through extensive reasoning and logic. The convergence is thus able to incorporate learning directly into the integration development.

Figure 4 highlights the key differences between a traditional integration method and the semantic integration method.

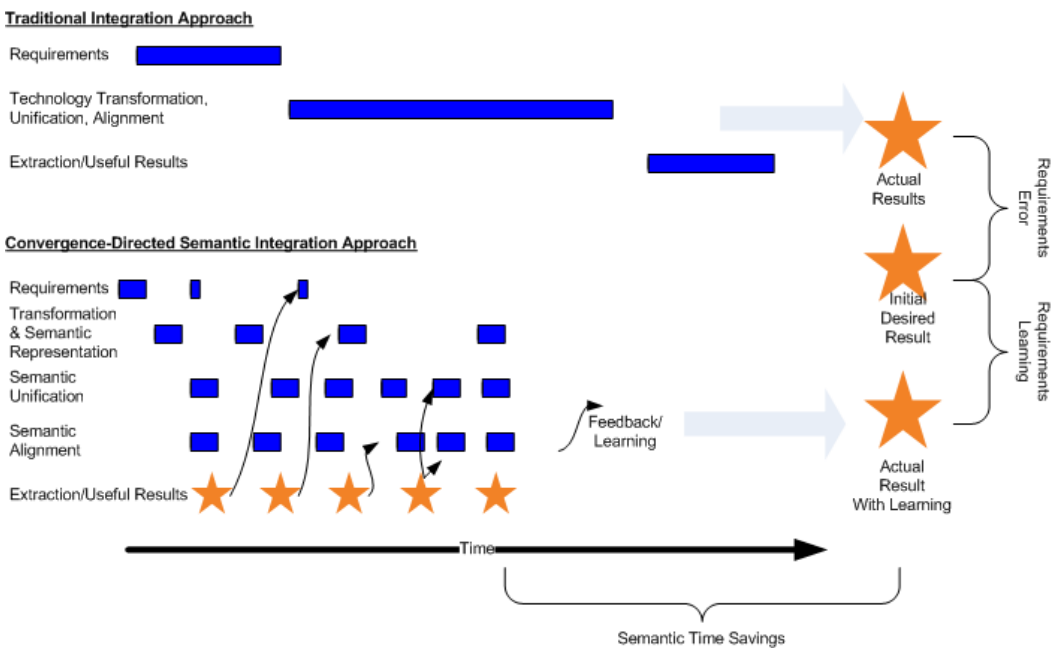

Figure 4: Integration Method Comparison

The traditional model takes longer to produce useful results and there is no opportunity for incremental results to influence the end state. The convergence-directed Semantic Integration approach produces limited results quickly. These results can then help direct the remaining development efforts, allowing the solution to not only produce useful results quickly but also *adapt,* as the integration and business opportunity is better understood. Whereas the traditional approach mimics a waterfall development method, the semantic approach follows an agile development approach that uses incremental and iterative methods. The convergence model is also more likely to efficiently and effectively produce useful information since both relevant and irrelevant databases are identified early during the integration process.

1.4.6 Method – Design Science Research Method

The design science methodology focuses on the creation of new and innovative artifacts - artifacts that address a significant problem. The artifact proves itself through a robust review that clearly demonstrates its significance in addressing the problem (Hevner & al, 2004). Additionally, the artifacts undergo refinement through an empirical review period that further advances its benefits.

The earlier section detailed the relevance of advancing the methods of data integration especially in light of the multiple trends that increase the relevancy.

The research will produce several pertinent and reinforcing artifacts.

1. A Semantic integration model that advances the overall integration of NoSQL databases.
2. Semantic analytic methods that transform, represent, unify, align, and adapt the data.
3. A Semantic implementation that illustrates and demonstrates the key constructs of the Semantic Integration model.

These artifacts will go through an evaluation to demonstrate the advancement using various means to triangulate the research results.

The method to address each research objective centers on the creation and implementation of a working semantic integration model as an advancement to the current integration mode. The working implementation of the model employs the

Semantic Web as a specific implementation technology using a set of exemplar data sources, semantic integration techniques, and semantic extraction techniques.

The working model exposes the semantic integration model to an empirical examination through a user trial task using the semantic integration model in comparison to the traditional integration model. This provides both quantitative and qualitative views on the model.

Due to the large scope of the effort several limitations are recognized. The model will operate in a read-only fashion and thus not deal with the complexities of transactions and updates. The implementation and study will assume relatively clean data and therefore not focus efforts on data quality or enrichment issues.

1.5 Research Significance

Businesses, governments, and individuals make enormous investments in evaluating and integrating data. However, most efforts are expensive and limited. Current trends point to more integration opportunities. Advancement in data integration offers a lower cost and faster turn-around enabling new integration opportunities and producing new value from existing data.

The semantic integration model enables more information to truly produce a higher fidelity view of the domain of interest. This should lead to better, more informed decisions that incorporate more pertinent and timely information. Additionally, the

semantic integration model promises to lower the cost of integration. This will open up many new integration opportunities.

Advancements in data integration produce value from existing assets – accessible information. These advancements would allow insights to new medical treatments, new financial vehicles, and improved customer service, among others. Simply put, better integration provides a higher fidelity decision framework leading to improved decisions.

1.5.1 Key Contributions

1. Improved integration of useful data extraction methods based on semantics for NoSQL databases. This includes uncovering the structural information contained in a dynamic NoSQL store in a timely fashion.
2. Applicability of Semantic Web standards, tools, and Semantic Web sources to aid semantic data integration.
3. A Semantics-based integration structure and methods to advance integration efforts.
4. Applicability of convergence methods to incrementally and iteratively advance data integration.
5. Advanced evaluation methods through the comparing and contrasting of various traditional and semantic integration methods.

1.6 Organization of Dissertation

Chapter 1 provides an overview of the research. It introduces the fundamental background that frames the research, the research questions, and an overview of the research methods.

Chapter 2 provides the background information that lays the foundation of this investigation. This includes studies on data integration methods. It highlights the research by using semantics to advance integration in various domains including the specific use of the Semantic Web.

Chapter 3 describes the research model, methods, and implementation in detail. This includes a detailed set of methods and key coding constructs of the implementation.

Chapter 4 describes the model evaluation methods that include executive interviews, survey instruments, and a data integration task. It reports and summarizes the evaluation results. The task is based on artifacts produced by the implementation of the semantic integration model and associated methods.

Chapter 5 outlines the contributions, limitations, and future research directions.

2 Chapter 2: Related Work

2.1 Research Areas Overview

The difficulties and challenges in data integration represent large costs and lost opportunities for business, governments, and even individuals. The challenge doesn't abate but worsens, even years after the inception of data integration. The ability to make coordinated, organization-wide responses to today's business problems is thwarted by the lack of data integration (Goodhue, Wybo, & Kirsch, 1992). It continues into this century: "Data integration becomes mission critical" was noted in the 10 marketing trends to watch in 2011 (Blunt, 2011).

Database technology was introduced in the late 1960s to support various business applications. As the computing platform grew to include multiple applications and data sources, the need for integrating data became apparent (Ziegler & Dittrich, 2004). The initial solutions depended on a small global database schema. This produced challenges in dealing with the various ways of expressing the data both technically and semantically. Zoom forward to today, and the challenge continues but at a scale, in both amounts and timelines, unthinkable just a few years ago.

Traditional integration solutions and their corresponding models depended on factors absent in many of today's information sources. These include control and comprehension. Formally, data sources and associated data integration efforts were kept behind corporate walls. Here the corporation controlled the data source entirely and fully understood it (for the corporation typically created and maintained the data source). This

made integration achievable as well as maintainable. Today's information and data environment finds almost the opposite. Although many opportunities still exist behind corporate walls, they have been swept up by the same wave of data creation and new technologies – creating a microcosm of the larger data environment. The larger data environment, fed by the Internet and its associated innovative applications, has produced massive amounts of information in a multitude of forms and locations. The sheer ease of information creation has produced a new timelines and associated urgency to the data. For example, Twitter produces 250 million short messages a day (Nakano, 2011). Users can search these messages on keyword, topic, creator, and more. On the other end, big data or cloud computing will approach 100 - 160 billion dollars of investment in 2014 (Hickey, 2010) (Williams, 2009). Small companies and even individuals are involved with major cloud technologies through several major product offerings from Apple (Apple, 2012), Amazon, and Google.

The lack of control and comprehension coupled with the dynamics and massive amount of data seriously challenge traditional integration methods (Mohan, 2013). Fortunately, new technologies offer capabilities to create an adaptive, and efficient environment to leverage the growing mass of information scattered amongst the many technologies and formats. Additionally, the growing mass of information itself provides insight and assistance with data integration. These integration efforts, themselves, can be shared and built upon. Finally, the current data landscape shows no sign of stabilization. New technologies, formats, and data-producing applications will continue to emerge and advance. Data integration advancements must adapt to this continuously changing data landscape to produce additional value from this large data investment.

The key to success is moving from a technological perspective to a semantic one. This idea is not new. Semantic integration has been explored for almost a decade. Semantic technologies are usually considered a key factor in dealing with the huge amount of data available today (Verter, 2010).

The Semantic Web, outlined by Tim Berners Lee in early 2001 (Berners-Lee, Hendler, & Lisssila, 2001) has continued to grow through a collection of standard knowledge representation technologies, compatible tools, and a growing foundation of semantically structured information. This combines with several new information visualizations based on semantically-rich contextual information, automated reasoning, and manual semantic guidance.

Two dimensions help determine the value of the data and its integration with related data: business metrics and data metrics. Business metrics examine the improved benefits for the bottom line on investment time and cost. Data metrics uncover the data quality and associated data advantages that contribute to improving the business bottom line (Martin, Poulovassilis, & Wang, 2014).

2.2 Data Integration Model

Data integration requires distinct interfaces to each data source and corresponding processing workflows. In order to best compare and contrast various data integration methods, a data integration model is formed.

Data integration can occur at different levels of architecture (Ziegler & Dittrich, 2004) from the low-level data storage to a user interface. Integration establishes the technical interface that points to extracting, transforming, unifying, and aligning data. Moving up the stack, the integration reveals different levels of semantics or meanings along with different technologies. The following provides a more abstract view of the conceptual decomposition of the architecture levels.

- Foundation: It represents the integration of the underlying computing infrastructure such as the operating system. Regardless of a higher-level integration steps, the various operating environments should interact with one another. For the most part, higher software levels such as the integration application handle this level.
- Technology: This represents the multitude of technologies that store and/or interact with data. These include database technologies, file storage, technology formats such as XML and RDF, and so on.
- Format: It represents the actual format of the data. For example, dates and locations often exist in a multitude of formats.
- Semantic: This represents the meaning of the data. Data may have multiple meanings and the actual interpretation may depend on its context.

Additionally, data integration requires the following processes and staging steps (Giordano, 2010) (Fan & Gui, 2007).

1. Data extraction

2. Data cleansing
3. Transformation into common form
4. Combining into integrated data form

These steps form a workflow that moves data from one data application and purpose to another data application and purpose.

This can be further conceptualized by examination of the various stages: transformation, representation, unification, and alignment. Transformation handles the extraction and isolated cleansing, representation forms the data into a standardized form, unification combines data from multiple sites and performs the necessary processing, and alignment advances the data to form a useful purpose.

Two additional steps complete the workflow. The integrated data requires some form of access/extraction to its subscribing clients and the data must expose itself in a useful visualization. Many visualizations of integrated data exist. (van den Heuvel & Rayward, 2011) (Tufte, 2001)

Two other considerations complete a full description of data integration: integration breadth and integration timeliness. Data integration breadth represents its scope or reach to disparate data and associated technologies along with the amount of data. Two dimensions can describe the timeliness: the time it takes to integrate the data including integration development time and the time when actual data transfer takes place. Timeliness must also consider its adaptation to changes in the integration effort from new

data structures to new uses of the integrated data. Adaptation becomes critical when technologies and business opportunities evolve quickly.

Putting together the various factors of integration architectural levels, data integration workflow, breadth, and timeliness forms a data integration model, as shown in Figure 5.

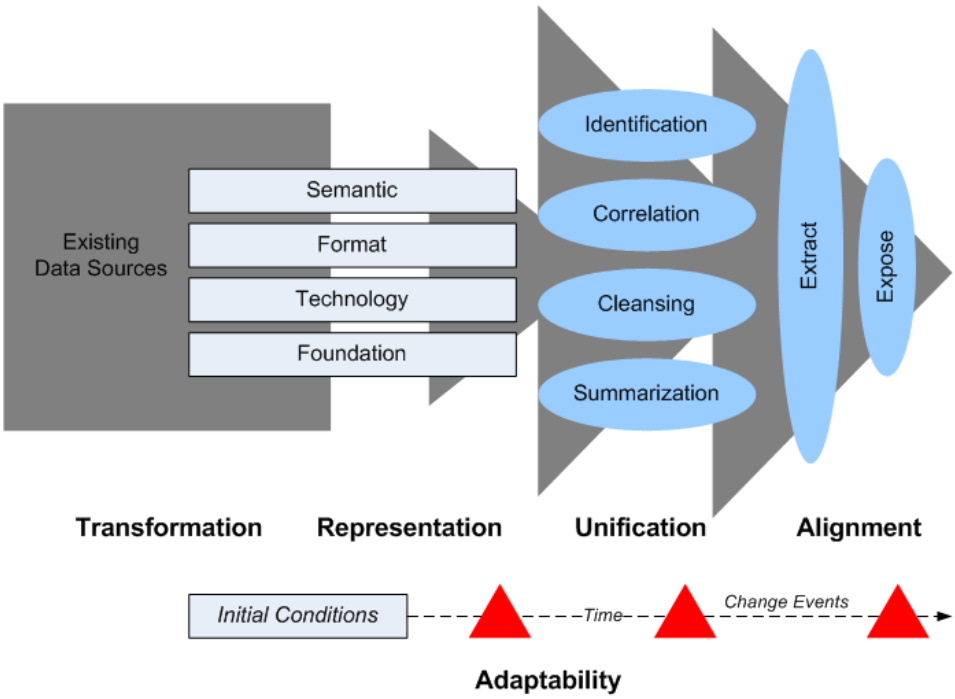

Figure 5: Data Integration Model

The data integration model highlights the movement from isolated, existing data sources through transformation at the various architectural levels, unification to bind multiple data sources, and finally through to extraction and exposure to the integration domain. Additionally, the model handles timeliness through adaptability to represent the initial and subsequent efforts due to the underlying data and/or data use dynamics.

2.3 Traditional Data Integration

Traditional data integration focuses on the technical framework formed by relational databases. Relational databases typically exist to serve specific applications and are tightly bound to the associated application. In general, information systems are simply not designed for integration (Ziegler & Dittrich, 2004).

Traditional data integration must deal with the limited semantics within the data structures and often its narrow scope within the originating data application.

Traditional data integration focuses on relational databases, which prior to the Internet contain the majority of structured data. This took two different courses – Local as View (LAV) and Global as View (GAV) (Lenzerini, 2002).

The LAV approach presents the content of each data source in terms of a view over a global schema. This requires an enterprise data model. Each data source must provide a mapping between the local schema and the global schema. Adding a new source simply requires an additional map to the existing global schema. LAV variations exist that highlight the similarities between the mappings and the global schema (Lenzerini, 2002).

The GAV approach presents each global element as a characterization of the data source element. The approach depends on a stable source systems and a federated query. The global view forms a lens that brings each source into focus.

GAV models the global schema as a set of views over the source systems whereas LAV models the source database as a set of views over the global schema (Wikipedia, Data Integration, 2011).

The global schema can be virtual or physical. The former creates an on-demand federated query. The latter results in a data warehouse approach that avoids multiple, coordinated queries but requires larger data movements and the inherent latency evolved in transforming and copying the data to the data warehouse (Bennett & Bayrak, 2011).

Figure 6 summarizes traditional data integration. Transformation occurs dynamically when responding to a federated query or in a batch mode when transferring to a data warehouse. The key difference is demonstrated when the data movement and transformation take place, either while responding to a query or in a batch operation.

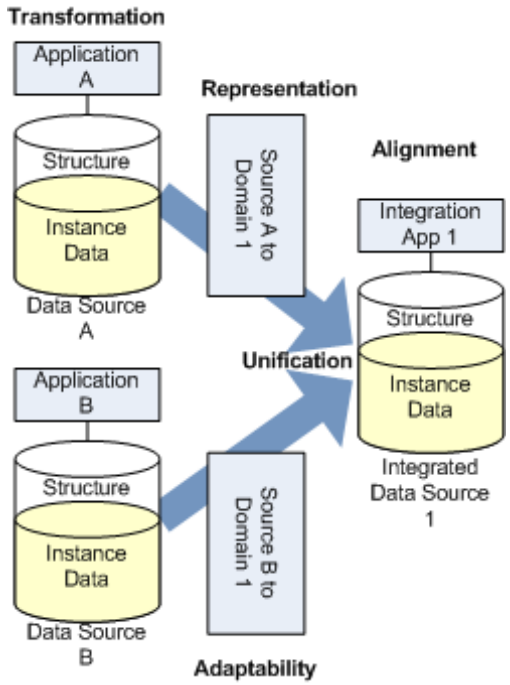

Figure 6: Traditional Data Integration

Figure 6 highlights the middleware that transforms and unifies the data. This is traditionally referred to as Extract, Transform, and Load (ETL) and is typically a custom solution that is not shareable. Alignment occurs in the virtual (federated queries) or physical data warehouse through query methods that combine with the integration application. One limitation of a query language is that it typically requires application cooperation to extract and display useful information from the integration.

Traditional integration suffers from significant limitations. The many manual and custom tasks required for data integration exert a high toll in time and cost. This directly leads to poor scalability and poor adaptability. Additionally, the level of customization precludes sharing of the integration approach. Most data integration efforts start from scratch. The scale in terms of the number of data sources and source heterogeneity further increase the

difficultly (Goodhue, Wybo, & Kirsch, 1992). This indicates that challenges for traditional integration will significantly rise due to the sheer increase in both number of sources and their diversity.

The physical integration of data misses the semantics. This results in developers hard coding much of the meaning that represents the data. This approach produces brittle systems with little flexibility that are expensive to maintain. Most commercial integration systems are limited in this way (Uschold & Gruninger, 2004). Relational databases require tables to provide the relationship information. In addition, these tables and field names define the relationships but not the semantic meaning behind the relationships. As a result, it is difficult, if not impossible, to retrieve semantics directly from the names.

Although these methods support controlled relational database integration, they exhibit serious limitations when challenged with more advanced Internet technologies such as big data storage in NoSQL databases. These databases are highly dynamic and massive in scale. The large scale and data dynamics do not easily permit the extensive data movement required in traditional database integration.

2.4 Semantic Integration

Semantic heterogeneity has long been recognized as a key challenge for data integration (Buccella, Cechich, & Brisaboa, 2005) (Doan & Halevy, 2005). Information exists at many levels from meaningless data bits to actionable knowledge. Semantics refers to an explanation of the data in terms of the real world – the meaning (Zhou, 2010). Semantic

solutions are typically based on an ontology that contains classes and relationships between the classes (Noy, 2004). The ontology provides a formal description of a given domain. Ontologies may also offer the benefit of reasoning. Semantic technologies are a key factor for dealing with the huge amounts of data available today. Semantics map expressions in a given technical format with things or facts in a given world (Vetere, 2010).

Without citing particular technologies, there exist three main ways to achieve semantic integration. The distinguishing factor is where the integration takes place: in a programming language; schema mapping using syntax; or through declarative techniques. Additionally, the semantic integration can incorporate into a global semantic schema, individual pairing, or some hybrid form. Although global schemas offer a direct clear method for enterprise data integration, it falters with scale. The P2P method becomes adaptive to larger scale data sources despite its initial startup costs (Zhou, 2010). Effective methods employ a combination of the two.

Fundamentally, semantic integration techniques are similar to traditional techniques. The key difference is where the integration takes place. Traditional methods integrate at the technical level but not the semantic level.

Semantic integration typically integrates at the ontology level and involves ontology mapping where the semantics are aligned between the data sources. Several methods exist to align ontologies; a shared (or global) ontology where each source maps to the shared ontology, heuristics, and machine learning. Shared ontologies map individual data sources to an agreed upon ontology. There are several published ontologies helpful to

shared ontologies such as DOLCE and SUMO (Noy, 2004). Heuristics take advantage of various semantic algorithms using lexical and structural components to form an alignment between two ontologies. Machine learning techniques cover various procedures that associate terms with various methods that can include heuristics and/or training. Noy further notes that declaration-based semantic integration improves the description, explanation, conjunction, integration, and reasoning contained in the information (Noy, 2004).

Formal ontology provides clear and definite semantic descriptions and offers a good basis for enterprise information integration and semantic interoperability *(Noy, Semantic Integration: A Survey of Ontology-Based Approaches, 2004).*

Multiple research efforts are focused on ontology mapping methods (Buccella, Cechich, & Brisaboa, 2005), (Kalfoglou & Schorlemmer, 2003), (Kaza & Chen, 2008). These techniques discussed automated methods, which have made great strides but still fall short of comprehensive, understandable mappings. In contrast, *manually* supported techniques are effective in offering visual interactions with a user. (Granitzer M. , Sabol, Onn, Lukose, & Tochtemann, 2010). Other techniques emphasize adaptively employing multiple strategies (Idrissi & Vachon, 2009). Additionally, there have been various methods employed to evaluate the mapping effectiveness (OAEI, 2011), (Euzenat, Meilicke, Stuckenschmidt, Shvaiko, & Trojahn, 2011).

Several types of mapping challenges exist between ontologies (Klein, 2001); language level mismatches, ontology level mismatches, explication mismatches, and encoding mismatches. Language level mismatches can be addressed somewhat through language

translations. Ontology level mismatches include using the same concept term with a different meaning - conceptualization mismatches. Explication mismatches result from paradigm differences such as using different top-level ontology. Encoding mismatches are caused by using different formats for measurements, time, and the like. Formally, these mapping challenges are typically addressed by manual mapping (Noy & Musen, 2000).

Recent advancements in algorithms move the mapping from the individual names and labels to the rich relationships found in the data context level. Context mapping takes advantage of the relationships in addition to the term or symbol. Several methods exist to provide some level of automated mapping (YongTao, FengJuan, & HuiJuan, 2010) (Comito, Patarin, & Talia, 2006) (Doan & Halevy, 2005). Some methods focus on the terms and use related information resources such as WordNet (W3C, Wordnet in RDF, 2011). Other forms leverage the context or relationships that surround the term. Both types of solutions depend on a wide range of available semantic data to allow for comparisons and enrichments.

The results of these mapping algorithms, despite being promising, do not fully merge the ontologies successfully (Buccella, Cechich, & Brisaboa, 2005). Semantics is ultimately a human invention requiring some level of human interaction.

Semantic integration can exist as a top down integration approach using a top-level ontology or direct integration using a P2P model. The former depends on reaching an agreement on the top-level ontology. Although possible in controlled groups, this is not possible in large-scale integration effort that employ a large number of data sources with

different owners. P2P technologies and architectures provide more scalability and flexibility in addressing integration (Moujane, Chiadmi, Benhlima, & Wadjinny, 2009).

Extensive groundwork has been established for semantic integration in the last two decades. For much of that time, there was a lack of a focused technology platform to allow collaboration in creating, managing, and using semantic data. This has resulted in isolated semantic solutions that contain potentially useful data, tools, and integration approaches.

2.4.1 Semantic Integration Platform – The Semantic Web

The emergence and evolution of the Semantic Web provides a common platform to address the semantic issues. Previously, the diversity of plausible semantic solutions impeded progress (Kalfoglou & Schorlemmer, 2003). The Semantic Web provides a common, standard expression of semantics (Herman, 2011). The various Semantic Web standards result in a platform that contains ontology and data languages, reasoning constructs, various tools, and a growing body of available data and associated ontologies. Contributions come from all major sectors including government (Government U. , 2010), open source community (Bizer & al, State of the LOD Cloud, 2011), commercial (MacManus, 2010) (Sindice, 2012), and many individual efforts (Foaf, 2010). Additionally, mapping sites exist that expose mapping algorithms (Euzenat J. , 2008), transformation methods, (Bizer, 2010) and semantic equivalence (sameAs, 2012) along with many forums and conferences that share experience and skills.

This extensive combination of standards, tools, semantic data, and shared experience found in the Semantic Web offer a path to fulfill the promise of semantic integration (Langegger, Wolfram, & Martin, 2008).

Semantic Web standards provide a common, accepted way to create, manage, and query Semantic Web data. The key standards include:

Knowledge and Data Representation: Resource Description Framework (RDF) provides a data structure, RDF Schema provides basic structure including classes and properties, and OWL Web Ontology Language provides additional logical expressions such as cardinally, equivalence, and class restrictions. These representations operate under the open world assumption. This is conducive to data integration due to its lack of non-asserted assumptions and openness to new assertions including structural assertions.

Semantic Web Query Language: SPARQL offers a flexible and powerful method to provide query semantic data (W3C, Prud'hommeaux, & Seaborne, 2008). Additionally SPARQL offers a method (e.g. CONSTRUCT query) to add to the data given underlying conditions in the semantic data. This can be used to form subsets or add data in the form of a rule. SPARQL offers a web service standard – SPARQL protocol for RDF (Clark, Feigenbaum, & Torres, 2008) to allow a web service SPARQL interface. This combines with SPARQL 1.1 (Harris & Seaborne, 2012) to allow federated queries.

Rule Languages and Standards: Rules assist underlying logical ontology constructs. They also allow fine grain control. Rules provide custom expansion of the expressivity of the data representation. There are several rule standards, such as Semantic Web Rule Language (SWRL) (Horrocks, Patel-Schneider, & Boley, 2004), Jena Rules (Jena, 2010), and SPARQL Inference Notation (SPIN) (Knublauch, Idehen, & Hendler, 2011). Rule Interchange Format (RIF) provides an overarching standard to provide a common rule exchange. (Kifer & Boley, 2010)

The standards allow for interoperable tools. The key tools include:

Reasoners: Reasoners provide inference to the ontology constructs. The RIF standard categorizes the different capabilities of the available reasoners. Currently, there exists a multitude of open source and commercial reasoners compatible with the Semantic Web (Wikipedia, Semantic Reasoner, 2011).

Triple Stores: Triple stores provide storage for the semantic data. They also offer various services such as a SPARQL query interface. A multitude of open source and commercial triple stores are available (Wikipedia, Triplestore, 2012).

Programming Frameworks: Programming frameworks allow programmatic interaction with Semantic data. The preferred framework is Jena (Apache, 2011). Jena provides extensive methods to create, manage, and query Semantic Web data and constructs.

Ontology Integrated Development Environments (IDE): IDEs construct and validate ontologies. There are several open source (e.g. (protege, 2011)) and commercial versions of IDEs.

As previously mentioned, there are extensive amounts of Semantic Web data available including rich domain ontologies and data. Correspondingly, there is a multitude of transformation and alignment technologies and standards.

Large scale, high quality ontologies depend on effective and useable methodologies that produce ontologies (De Nicola, Missikoff, & Navigli, 2009). Ontology engineering provides formal methods to create and validate an ontology – "To provide a basis of building models of all things in which computer science is interested in" (Mizoguchi, 1998). Creating an effective Semantic Web ontology benefits from ontology engineering practices of formally defining the domain and associated semantics through logic expressions captured in the ontology. Semantic tools such as Protégé allow these expressions to be visualized and verified (Pouchard, Ivezic, & Schlenoff, 2000).

Ontology engineering outlines formal methods to design an ontology (Mizoguchi, 1998). These include requirements to ensure the ontology is intelligible to both end users and computers, allow combinations of ontological elements to form larger concepts, contain both a conceptual layer and symbolic elements, and ensure the ontology can interact with an object oriented computing language to fully exploit its potential.

The existing tools and the design of the semantic integration model adheres to these requirements. The Semantic Web using OWL/RDF can hold complex concepts that can

exist alone or as combinations of multiple concepts. Additionally, tools can extract the concepts for user consumption contained in the ontology above the symbolic format expressed in OWL and RDF. Thus, the semantic web can be both understandable by a user while also being processed by software in a computer. The Semantic Web enables adherence to the practices promoted in ontology engineering.

2.5 Emerging Data Sources and Semantic Integration

NoSQL databases contain structured and semi-structured data but lack semantics and few standards. They exist as isolated databases that contain highly useful and timely information. Thus, although the NoSQL database contains the richness of massive data, they lack the standards and semantics to easily integrate this rapidly growing set of data sources.

There is not a single definition of cloud computing and their associated databases (Grossman & Gu, 2009). One clear division of cloud databases is between SQL and NoSQL databases. The former contain traditional relational databases hosted in a virtual environment as virtual appliances. This simply places the same technologies into a new, more flexible virtual environment. The latter, (NoSQL databases) take advantage of cloud dynamics for data and employ a simpler schema. The NoSQL approaches quickly scale beyond relational databases. There is a clear need to investigate how different NoSQL sources interoperate (or integrate). (Grossman & Gu, 2009)

Big Data databases can be segmented along several key data dimensions: consistency, availability, and partition tolerance as detailed in Figure 7.

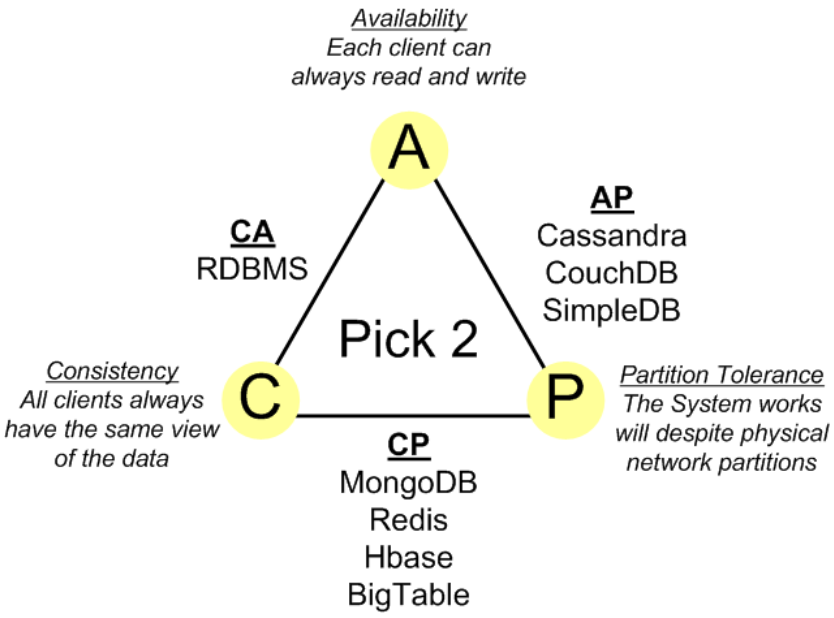

Figure 7: Key Dimensions of Cloud Databases (Hurst, 2010)

Consistency ensures the same view of the data (Brewer, 2012). Thus the data source doesn't return different answers for the same question asked by different clients. Availability ensures that the data source returns an answer. Partition tolerance ensures that the data source can expand and distribute across multiple physical storage devices. The three categories help to distinguish between the various data source offerings in the cloud. This also points out the limitation since achieving high levels in one area may prove detrimental to another. For example, a cloud database that maintains high availability may suffer in consistency since the various nodes in a cloud may not always be in sync. Correspondingly, a database that wants to provide high consistency may lower availability to ensure all nodes are in sync. There is a multitude of types of NoSQL databases and they continue to evolve. This is likely to continue as data requirements evolve and grow.

A multitude of data exists that is publically accessible via various NoSQL databases. Amazon offers dozens of large data sets accessible via their cloud technology (Amazon, Public Data Sets on AWS, 2012).

NoSQL databases focus on performance due to the scale demands. Three primary forms of NoSQL databases exist: column, document, and graph NoSQL databases. These databases are very different from traditional relational databases and focus on reading/writing quickly, supporting mass storage, allowing for ease of expansion, and low cost (Han, Haihong, & Du, 2011). Additionally, the databases incorporate a flexibility not found in SQL databases because structural information can be added at any time and may reside at any location within the database.

All three types of NoSQL stores operate on the basic principle of a *key* that ties together common data to form a data record. For example, a column based store contains a row that holds a unique key in one column followed by a column name such as "Department Name" followed by a third column with the actual department name. The next row (or a row anywhere else) holds the same key but the column name is different such as "Department City" with the third column holding the city name. A document-based store also maintains a key but the key binds to a document. Here a document does not refer to a traditional MSWord document but rather a structure format in XML or JSON. Finally, a graph NoSQL store holds a key that binds to many relationships such as Department Name. The NoSQL graph database connects relationships at only one level (Neo4J, 2107). In other words, they are not full graphs that implement the concept of inheritance (such as "human" inherits relationships associated with "living thing"). This constraint

allows the same relationship processing as the other two types: column and document in the semantic integration model.

All NoSQL databases embed the structure information within each record. This allows each record to share the same structure, use a slightly altered structure such as adding a new field like the email address, or use a completely different structure. There is no enforcement on common structural elements. This flexibility provides a highly dynamic store that quickly adapts to new, updated storage needs while preserving the existing data. A structural change only impacts the newly added data. The existing data stays connected to its original structure. Thus, structural information is spread throughout the database. As the database grows, so do the challenges of finding the structural information and keeping documentation of the database up-to-date.

2.5.1 Semantic Cloud Research

Devising a semantic integration model for existing NoSQL data distributed throughout a cloud is a key focus area. Cloud data integration incurs in two types of heterogeneities – vertical and horizontal. Vertical represents heterogeneities within a single cloud. This can be addressed with some level of standardization. Horizontal heterogeneity refers to differences across multiple clouds. These heterogeneities prompt the use of semantic integration models for integration (Sheth & Ranabahu, 2010). Related semantic integration models that are based upon similarity link network (SLN) and association link network (ALN) support cloud data integration (Liu, Lou, & Liang, 2009).

Research in Semantic computing, which is a field associated with the Semantic Web, combines various elements of semantics, natural language processing and data mining (Wikipedia, Semantic Computing, 2011). Semantic computing attempts to extend the Semantic Web's breadth and depth through integration of the various cloud enabled components from user interfaces to pervasive computing. (Sheu, Wang, Wang, & Paul, 2009). This forms a semantic computing architecture. Figure 8 illustrates the architecture.

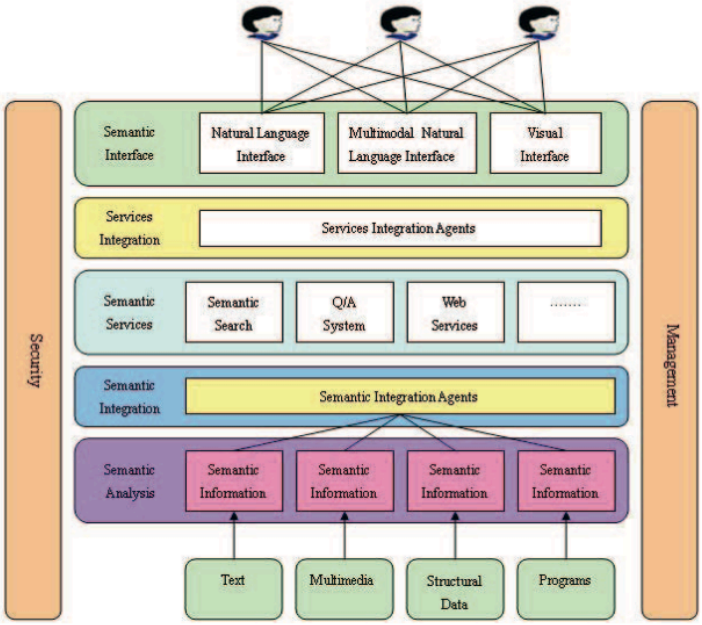

Figure 8: Architecture of Semantic Computing (Sheu, Wang, Wang, & Paul, 2009)

The Semantic integration agents in the figure transform and unify the underlying data sources. In order to move this model into cloud resources, it has to overcome several hurdles. Sheu notes that cloud providers have no standards, open protocols, or discovered mechanisms. There is no global index that searches across clouds. They

propose a Semantic Search Engine and note the compatibility of semantic computing and cloud computing. A semantic search engine is analogous to a typical search engine except that it considers the semantics behind the search such as context information. A semantic search engine could employ custom algorithms, natural language extensions and/or portions of the Semantic Web. It stores semantic information about Web resources to allow for complex web searches through a semantic match rather than a syntax match or other (Kassim & Rahmany, 2009). Semantic search engines offer a flexible way to integrate various cloud stores.

Several research efforts aim to establish an ontology to describe cloud data resources (Youseff, Butrico, & Da Silva, 2009). This could contribute to the simplification of the overall data integration through an understanding of the various cloud components and their relationships.

Several research efforts have attempted to put semantic data directly into the NoSQL stores, employing the scalability of the NoSQL and enabling its ability to process large amounts of data to run semantic reasoning (Zeyliger, 2010). This also allows SPARQL query processing to integrate cloud stores through a query complier (Husain, McGlothlin, Khan, & Thuraisingham). These solutions depend on the cloud data already having the Semantic Web data, which is not typically the case.

NoSQL data management creates several challenges for integration of any kind. NoSQL data is elastic only up to the point that it can be decomposed and parallelized. No data standards exist within any NoSQL technology. NoSQL databases continue to evolve and could change in incompatible ways. The data is often replicated across great distances

incurring hidden latency. The databases were not initially designed as a complete end-to-end analysis system (Abadi, 2009).

2.6 NoSQL Data Integration

Several methods have been proposed to integrate NoSQL databases using semantics (Livenson & Laure, 2011), (Gagnon, 2007), (Cure, Kerdjoudj, Faye, Le Duc, & Lamollo, 2012), (Cure, Lamolle, & Le Duc, 2013). They each require an exhaustive analysis to uncover structural information due to the NoSQL ability to hold structural information throughout the entire store. Their solutions read every record in the NoSQL database to determine the complete structural information. This is slow at best and intractable at worst. Additionally, they do not take convergence on a topic domain into consideration. Both limitations increase costs and time requirements, thus limiting the ability to explore additional databases.

These methods take a traditional view of data integration and assume an exhaustive approach will work, even at the large scale of a typical NoSQL store. Additionally, they fail to note that the structures and relationships between the data are much simpler and easier to use directly than the structure found in a relational database. The empirical results presented in the papers cited above employed a small scale NoSQL database, and only investigated small databases. Further, they did not consider the creation of an end-to-end integration using ontologies as a proxy for the native data store, but only transformed the database structure to an ontology.

2.7 Summary

Data integration should consider an end-to-end approach that includes transformation, representation, unification, and alignment performed in methods that maintain the alignment with the data sources and user objectives. Such a collection of various steps and techniques resulted in an ideal integration model. This serves to compare and contrast the various integration solutions.

Traditional integration methods suffer from extensive customization that results in brittle, time consuming solutions that fail to maintain alignment with dynamic data sources and user information goals. They also neglect to address the many data failures increasingly evident in large data integration efforts such as conflicts, duplication, missing information, and errors. Traditional integration remains constrained to its initial scope – a few well-controlled data sources of moderate size and typically in relational database formats and technologies. Traditional methods bind syntax and formats without regard to semantics. The semantics are contained in the enveloping application and/or the actual user.

Semantics integration offers a credible method to enable integration across a broad set of diverse data contained within multiple technologies because semantics pushes the integration point above the technology complexities. Formally, most of the useful data is not in a semantic form and contains little semantics. NoSQL stores, given their inherent simplicity, offer at least basic semantics - semantics useful enough to expose to a business user. Even if the semantics exist in a standard ontology, ontology alignment

techniques used for integration have failed to completely unify the various ontological structures through automation. This forces manual mapping to overcome impeded, large-scale integration. Several attempts have combined visual techniques with automated techniques to advance integration. However, many semantic attempts have been held back by the lack of standards and tools. This resulted in isolated solutions.

Semantics have now evolved into the Semantic Web that maintains widely supported standards in RDF, OWL, and SPARQL to name a few. This has spurred the development of an effective platform to build, store, reason, and query semantic data. Such a platform results in a multitude of semantic data and associated ontologies. This provides an opportunity to use this semantic platform and its associated data to further advance data integration. Additionally, reasoning and ontology enrichment can further advance the inherent basic semantics found in the NoSQL data source. Visualizations have also been developed to reveal the underlying semantic patterns allowing the non-technical user to assist and guide an integration effort rather than just a group of technical experts. This allows the data integration to evolve to the business needs more precisely and more rapidly.

Finally, the current methods are further challenged by the increasing business dependence on NoSQL databases. Their sheer scale precludes many integration approaches due to time and size constraints. But they also fail to deal with the dynamic structures and evolving proprietary NoSQL technologies or benefit from the simpler, more direct artifact names found in NoSQL databases. Fortunately, NoSQL databases offer basic semantics and minimize or eliminate complex relationships.

3 Chapter 3: Method

3.1 Semantic Integration Model

The semantic integration model advances the current integration approach by moving the integration focus from the technological underpinnings of data to the semantics of data. The shift improves the creation and adaptability of integration solutions for the rapidly emerging NoSQL databases. This semantic integration perspective becomes achievable as a result of the advances in the Semantic Web and associated user interfaces, which offer standards, tools, and knowledge sources to realistically enable semantic integration along with the complementary profile of NoSQL technologies.

The model recognizes the strategic role of the user, especially the non-technical user, through creating an integration framework that allows continual user guidance to focus and direct the integration. This allows disparate data source extractions to dynamically adapt and unify to meet the specific user's needs. The user plays an active role throughout the process of collecting useful results. This is a key success differentiator because semantics are ultimately a human artifact. This exposure of semantics rather than technologies allows the user to contribute to the integration itself. User self-direction allows the effort to *converge* on the user-intended solution rather than *depend* on a removed set of technologists and the time delay associated with such an effort.

The model also recognizes the strategic (and rather obvious) role of data sources and the dynamics of changing contents and structure. Data sources come and go. Data structures change. The model offers adaptability to recognize and incorporate these changes

without catastrophic hurdles throughout the integration process. The semantic integration model creates a decoupling between the semantics and the various data sources allowing data sources to join and separate seamlessly.

These two key aspects - user role and data source adaptability of the Semantic Integration model - gracefully permit both user and data source dynamics, a key in large-scale integration across diverse interests and technologies.

The model is explored in three levels: conceptual, logical, and application. The conceptual model ties the various components to the ideal model for transformation, representation, unification, alignment, and adaptability. The logical model ties the logical concepts to specific technical components, standards, and data sources through defined methods. The application model ties the model and methods to an actual implementation suitable for a rigorous evaluation.

The evaluation of the model uses two approaches to provide appropriate triangulation. The first interviews executives with business responsibilities involving data. The second uses an actual implementation of the model to enable users to interact with its visual artifacts to accomplish a defined task. Finally, participants from both approaches provide feedback on the method in terms of usability and ease of use when compared to existing alternatives.

Overall, this approach maps to the design science method that demands artifacts, problem relevance, rigorous evaluation, and the incorporation of existing aids.

3.2 Structural Analysis of NoSQL databases

Extracting useful structural artifacts from a NoSQL store forms a major assumption of the method. Relational models offer standard structure extraction with complex tools such as the Erwin modeler to dissect and picture the complex structure. However, relational databases have very complex models with many indirections of the data based on the various relationships. NoSQL models lack the benefit of easily identifying the structures contained in the database. An important deduction in support of the method is the validation of this assumption.

Relational databases are constructed of tables and relationships among the tables. Each relational table has one defined structure and all records in the table strictly adhere to that defined structure (Date C. , 2005). NoSQL databases have the notion of tables but allow for multiple structures within the table. The fields contained within each structure may vary significantly. For NoSQL databases, a table is merely a collection or grouping of records where each record within the table defines its structure. The same structure might be repeated throughout the table, or many different structures may exist within the same table. The structural flexibility of NoSQL databases allows the table to evolve and grow rather than have to be reconstructed when any change in structure occurs. To this point, the NoSQL databases have different names for this structure. Cassandra (Cassandra, 2016), calls it a column family while MongoDB (MongoDB, 2017) calls this structure a collection, and Neo4J (Neo4J, 2107) calls it a node. Each adheres to the concept of a collection of records heretofore referred to as a table. Focusing on the performance and scale rather than data integrity, transactions, and complex relationships,

this table delineation in NoSQL tables achieves some logical or performance-related partitioning. However, the structural flexibility disallows a simple extraction of the structural information contained in a NoSQL table**, r**equiring advanced technical skills to retrieve the native data. The semantic integration method aims to identify the structures of each table to assist business users, with limited technical skills, in retrieving and integrating data from NoSQL databases.

After reviewing the various documents of several relational and NoSQL databases (Oracle, 2017) (MongoDB, 2017) (Accumulo, 2017), Table 1 outlines the key differences and their impacts for NoSQL databases. It can be concluded that NoSQL technologies support the creation of a simpler, non-normalized structure without complex relationships to other tables within the database. However, the structural information is stored throughout the database and supported by a harder-to-find technical skill base. A detailed discussion follows the table.

Table 1: NoSQL databases compared with Relational Databases

Capability	Relational DB	NoSQL DB	NoSQL Impact
Atomicity	Transaction	Row, Column, Document	Discourages cross-table relationships
Consistency	Transaction	Eventual	May reflect different answers to the same query.
Isolation	Transaction	Row, Document	Discourages cross-table relationships

Durability	Full Audit Trail	Not Assured	Redundancy protects the data
Data Size	Limited	Practically Unlimited	Limits data movement
Distribution	Limited	Practically Unlimited	Rapid expansion, scale as needed
Structure	Relational	Column, graph, document	No standard structure
Structure Information	Centralized retrieved with call	Decentralized across the entire store, part of every record	Flexible structure stored throughout the database
Structure Semantics	Often attempt to reflect complex relationships through the structure	Simple straightforward structure due to external use and lack of representing complex data elements	Simple, non-normalized structure
Structure Complexity	Relational complex with many cross	Minimal cross references – strongly	More straightforward structures, limited cross references

	references	discouraged	
Normalization	Encouraged and supported	Not encouraged	Repeated information throughout, avoids need for cross references
Skill Base	Established and plentiful including certifications	Rare with spotty, proprietary certifications	Difficult to acquire the technical staff necessary to extract value
Maturity	Mature	Rapidly Evolving	Difficult to form any standard, constant expansion of capabilities
Emphasis	Consistent Information	Massive information ingest and retrieval	Speed/Scale over data protection

The first four rows deal with the ACID qualities – atomicity, consistency, isolation, and durability. NoSQL databases focus on flexibility and lack of coordinated transactions. This limits reading and writing atomically related information in multiple locations with assured consistency. Consistency beyond basic atomicity enforced at single record levels provided by NoSQL databases require complex external controls – this strongly discourages rich relationships outside of the limited atomicity. Since it is not possible to maintain full consistency, NoSQL stores minimize or lack complex relationships that cross many tables. Hence structural analysis need not focus on extracting rich

relationships. Additionally the durability feature is swept away by the sheer scale limiting useful extractions used in traditional data warehouses and data marts.

The next section in the table focuses on scale. It is not surprising that NoSQL databases focus on scale – initial scale and the ability to scale as demand increases, whenever that occurs. This requires spreading out the data to multiple servers, fostering inconsistency across the data since there is no physical way that an update can propagate to all the NoSQL servers simultaneously. The only way to accomplish consistency would be to lock all the relevant areas until all areas are updated – this would simply take too much time and is usually avoided. In contrast, the NoSQL stores employ non-normalized data and freely repeat information throughout the store. Therefore, NoSQL databases cannot guarantee consistency but are only designed for *eventual* consistency. This also extends to structural information and no simple access method can retrieve the entire structural information. Additionally, the size requires dynamic structural information because it is unfeasible to discard the database and start over due to an updated structure. NoSQL structure, unlike the strict control found in a relational database, allows changes in structure at any time, stored at any location within the store. Fortunately, the structure is tightly bound to the contents, which repeat the structural association. Therefore, the most employed structures are repeated the most, and vice versa. The NoSQL structures tend to be simpler with fewer relationships than those found in relational databases and are distributed throughout a larger store. Popular structures repeat throughout and thus can be found more easily.

Finally, while relational databases are well established with standards, NoSQL are rapidly evolving. The latter currently offers no standards. NoSQL database evolution continues to offer new features, including features that create incompatibilities with previous releases. This evolving target pushes standards even further out and standardization is unlikely to appear in the near future. The evolution, lack of standards, and newness of NoSQL databases creates a skill shortage and requires custom work to extract data from each NoSQL database. Thus, technical skills are a constraint on the value of NoSQL data due to their scarcity and the common requirement to access the data via custom technical work.

In conclusions NoSQL data models reflect a simpler data model that often contains straightforward semantic information. Relationships are replaced with non-normalized repeated data and structure. The structure is stored throughout the database and is tightly bound to individual data records. NoSQL methods that read and write data are still evolving and currently offer no standard methods. This directly benefits the model in multiple ways: complete structural information can be found throughout the store; the model is straightforward with few, if any, relationships to other models; and the technology continues to evolve creating challenges for the technical staff. The semantic integration model is beneficial in extracting key information from NoSQL databases without technical skills and the extracted information offers useful descriptions of the data. This allows a non-technical user to quickly review multiple NoSQL sources and provide direct, useful information for NoSQL integration as well as reviewing the integrated database for completeness.

Relational databases hold many types of relationships such as one-to-many, many-to -one, and many-to-many. These relationships form constraints as well aid in data normalization. Tables are required to maintain many of these relationships and the table and column names reflect these relationships such as 'Person2Organization' and 'Postition2Role'. These names obfuscate the semantic meaning and present difficulties in extracting useful semantics directly from the labels and names used to describe the tables and fields within a database. Although relational databases could fit into the overall method, this complexity excludes them from consideration. Additionally, many tools exist to examine and interact with the relational data model but they still require extensive technical expertise to understand the complex relationships.

3.3 Conceptual Architecture

The conceptual architecture contains the key integration steps in producing useful integration results as shown in Figure 9. This aligns with Figure 2: Ideal Functional Data Integration Model. Transformation and representation convert data sources to their semantic form. Unification establishes a unified data model. Alignment creates and maintains the focus of the integration on the desired results and integrated data sources. These steps collectively move data from its various technologies, formats, and individual semantics to a form useful for the user. Adaptability allows the user to update integration due to new data sources, updated data sources, and/or new business opportunities.

3.3.1 Overview.

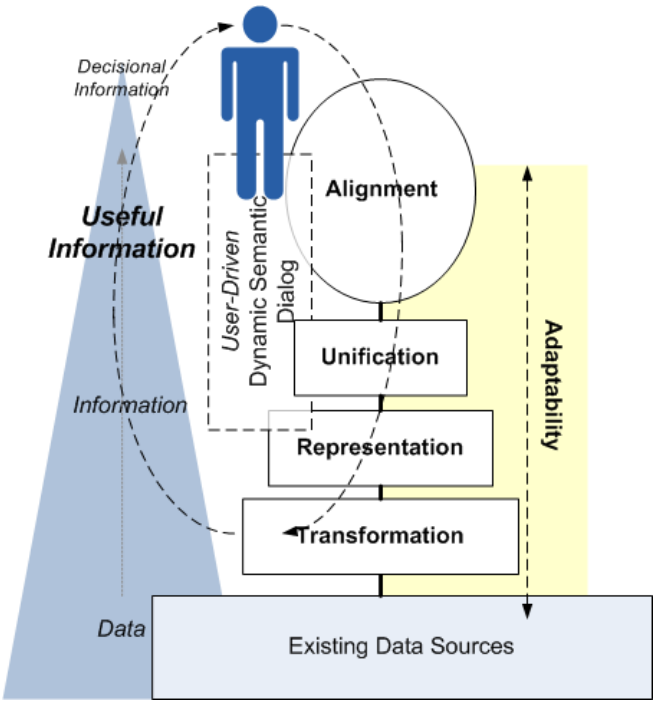

Figure 9: Conceptual Architecture

Rather than a static and batch approach typically employed by traditional methods, the semantic integration model continuously adapts through a progression of user-initiated and system automated methods. This iterative, adaptive approach allows on-demand integration rather than a priori integration. This allows the integration to converge on the complete integration solution and form it as one learns about the data and the business opportunity throughout the process. Semantic integration properly incorporates changes in the data sources and/or in the user focus. This approach enables rapid composition in the early stages of the integration effort and conveys early, up-front value to the integration while helping guide further investment in the integration effort.

The sections below provide additional details on the various conceptual components, their interaction, and relevance to producing useful information for the non-technical business user.

3.3.1 User-Driven Dynamic Semantic Dialog to enable Integration Convergence

Data integration ultimately serves a data user. Integration provides a useful meaning to the underlying databases that serve that user. The interpretation of meaning is subjective. Therefore, semantics starts with the person who can best provide the meaning behind the integration pursuit – the business user.

The semantic integration model absorbs this user meaning in an iterative manner through a semantic dialog. The semantic dialog provides the initial semantic seeds to start the integration and the semantic guidance to fine tune and refine the semantic integration. The dialog forms a partnership with the user in creating the integration solution – the user provides their semantics while the integration offers continually refined semantic data. The conversation continues until the user converges on the required information derived from the on-going Semantic integration. The conversation starts with examining data candidates for possible inclusion and proceeds until the business user is satisfied with the integrated data from the various data candidates.

The Semantic Web enriches this dialog through the many existing semantic data sources and methods. The user and the Semantic Web partner in developing and refining a path to the desired integrated information from multiple NoSQL databases.

3.3.2 Transformation

Transformation identifies the structure and contents of the database. Transformation handles the impedance mismatch between the underlying data technologies and the semantics useful to the Semantic Web. It is an iterative approach. Each newly uncovered structure is included. Transformation eventually converges on a complete set of structural information, which is key to identifying the contents. In addition to the key structural elements and associated data, transformation also identifies technology specifics. The latter provides the technical information necessary for the actual integration.

3.3.3 Representation

The identified structural elements in the common format received from Transformation are then converted to the Semantic Web using OWL and RDF conversion. This creates an ontology with associated instance data.

Standard Semantic Web tools such as reasoners and visualizers can use the resulting ontology. The richness of semantics web technologies enables a more complete capture of the underlying data concepts and their associated relationships, for the expressiveness of the Semantic Web far exceeds that of NoSQL databases.

The representation maintains information to obtain underlying data from its native source and format when requested. Thus, a Semantic Web query via SPARQL is converted to other various native data commands, as required by the underlying technology. For

example, the SPARQL call is converted to a proprietary NoSQL query. This has the added benefit of not requiring data duplication and its associated data movement latency.

Finally, many tools offer a powerful, interactive visualization of the ontology. This allows business users to inspect the structure and associated data for possible inclusion into an integrated knowledgebase as well as reviewing the integrated data for completeness for the given task. This is key to the iterative, convergence-based integration driven by a non-technical business user.

3.3.4 Unification

Semantic Unification combines and unifies the various semantic transformations. This further enriches and reduces the integrated data.

Unification consists of two distinct areas that work together: methods to recognize semantic similarities and data conflicts combined with semantic enrichment via external Semantic Web data sources.

The Semantic Web offers several advantages to unification. The Semantic Web standards require the same format and uniqueness for all Semantic Web data resources. Thus, Semantic Web data exposed via semantic transformation can easily be combined due to the uniqueness of every data element. The uniqueness is made possible due to the Unique Resource Identifier (URI) contained in every Semantic Web data element. Additionally the Semantic Web provides constructs that allow merges between the data elements without destroying the original data. Logic statements assert the equivalence of

both data classes (e.g. Person equals Human) and data instances (e.g. Joe H equals Joe Hanover). Upon the removal of these statements, the data reverts to the original differences. This allows for corrections and updates.

Additionally, the many available semantic analytic tools and reasoners can process the combined semantic integration. The analytic tools correlate information and repair errors. This can be done iteratively (and even be removed on discovering an incorrect unification step). The unification structures can be built upon and analyzed for future efforts.

External semantic sources can provide additional semantic context to determine similarities/disjointedness and deal with data errors such as missing data, conflicting data, and errors. The external sources complement the primary data sources.

3.3.5 Alignment

Alignment is the process that allies the complete integrated data with the business user's needs for data. Alignment is a continual process for the user may need to include additional data, the underlying integrated data may change, and the business opportunity itself may change. The semantic integration methods stay aligned with these changes. Several areas within the semantic integration method assist alignment: initial review, updated review, integrated data review, and selective query. These alignment review areas require the rapid analysis provided by the semantic integration method. The user can quickly review a NoSQL database candidate, review an include NoSQL database for any change, review the complete integrated NoSQL database, and selectively provide a query that provides detailed integration results.

The Semantic Web offers key assistance in alignment for the semantic integration method. The Semantic Web standards provide guidance and a powerful query language. The relationships between data are themselves semantic, enabling a meaningful traversal through the data. Additionally, many Semantic Web sources exist to help define and focus a particular area of interest. Finally, various alignments can co-exist with the same integrated data. This allows multiple perspectives and does not force a common view. These perspectives can conflict with one another, build on one another, or simply be completely independent.

3.3.6 Adaptability

Similar to alignment, adaptability exists throughout the various components. Adaptability continuously adjusts to changes and refinements driven by the Semantic dialog with the user, as well as changes to the data sources. Data source can changes vary from the addition of entirely new data sources to data structural changes to data changes.

The Semantic Web and its inherent semantics offer several ways of dealing with changes. The first level is the open world principle adhered to by the Semantic Web. This allows the addition of new data and structures without conflicts. Thus, the semantic structure of the integrated data can form to a structure that adapts and forms over time. There is no need to "get it right" the first time. The structure can take shape over time. Additionally, the ability to provide logical constructs allows a reasoner to adjust to changes by providing the logic to recognize and reorganize when change occurs.

Adaptability is the key that drives the overall integration effort. The user starts with a basic integration request that builds over time based on what the system incorporates and what the user learns. This allows the incorporation of learning from the user as the user explores the available data. The user need not know the final form of information required but rather just a starting point. This adaptability not only maintains alignment but also builds the initial alignment. The integration self-forms over time given the available data and the guidance of the user.

Adaptability employs convergence to incrementally and iteratively form the integrated data. Convergence maintains a *know nothing* approach to integration that learns over time. This is completely opposite from traditional integration which maintains an *a priori* view towards the data model and associated integration. Thus, the data is assembled as needed without any prior knowledge. Each data source is added until the business user is satisfied that it would address the business opportunity.

3.3.7 A Conceptual Example

An example helps to illustrate how different components from the conceptual architecture work together to build a useful integration from multiple data sources.

The user, having a specific business use case, requires relevant data. The user identifies a set of potential candidate NoSQL databases. The method allows the user to quickly review the contents of each database candidate. The review allows the user to select the entire data candidate or just certain tables and fields. Once selected, the meta-data regarding the NoSQL technology and associated structure is stored in an integrated

ontology. This process continues through each candidate. As each candidate is evaluated, the user studies the integrated ontology for suitability. If the user finds the integrated ontology complete, the user can then request the integrated data. The semantic integration method then uses the meta-data stored in the ontology to retrieve the data from the native NoSQL sources and present the results. Even at this point, the user is free to add additional NoSQL sources if the results are not sufficient to aid the opportunity or if the opportunity has changed.

3.4 Technical Architecture and Process Flow

The technical architecture follows the constructs of the conceptual architecture. Figure 10 provides an illustration of the technical architecture and associated process flow. The method provides an iterative, user-driven approach to evaluating and integrating databases focusing on NoSQL databases. The method consists of six phases, as outlined in Figure 10. There are two main branches represented by the two columns in the figure. The left column determines the inclusion of a specific database candidate of one of the three forms of NoSQL: document, column, or graph. Its goal is to provide useful structural information of the candidate store for review. The right column incorporates accepted integration candidates to form an integrated data domain of interest to a specific user pursuit. The two columns work together to iteratively and incrementally form a useful domain of data. In fact, the two columns are fundamentally the same but employ different levels of databases. The left column integrates an actual NoSQL database, whereas the right column integrates the selected databases to form the integrated database domain useful for the non-technical business user's opportunity.

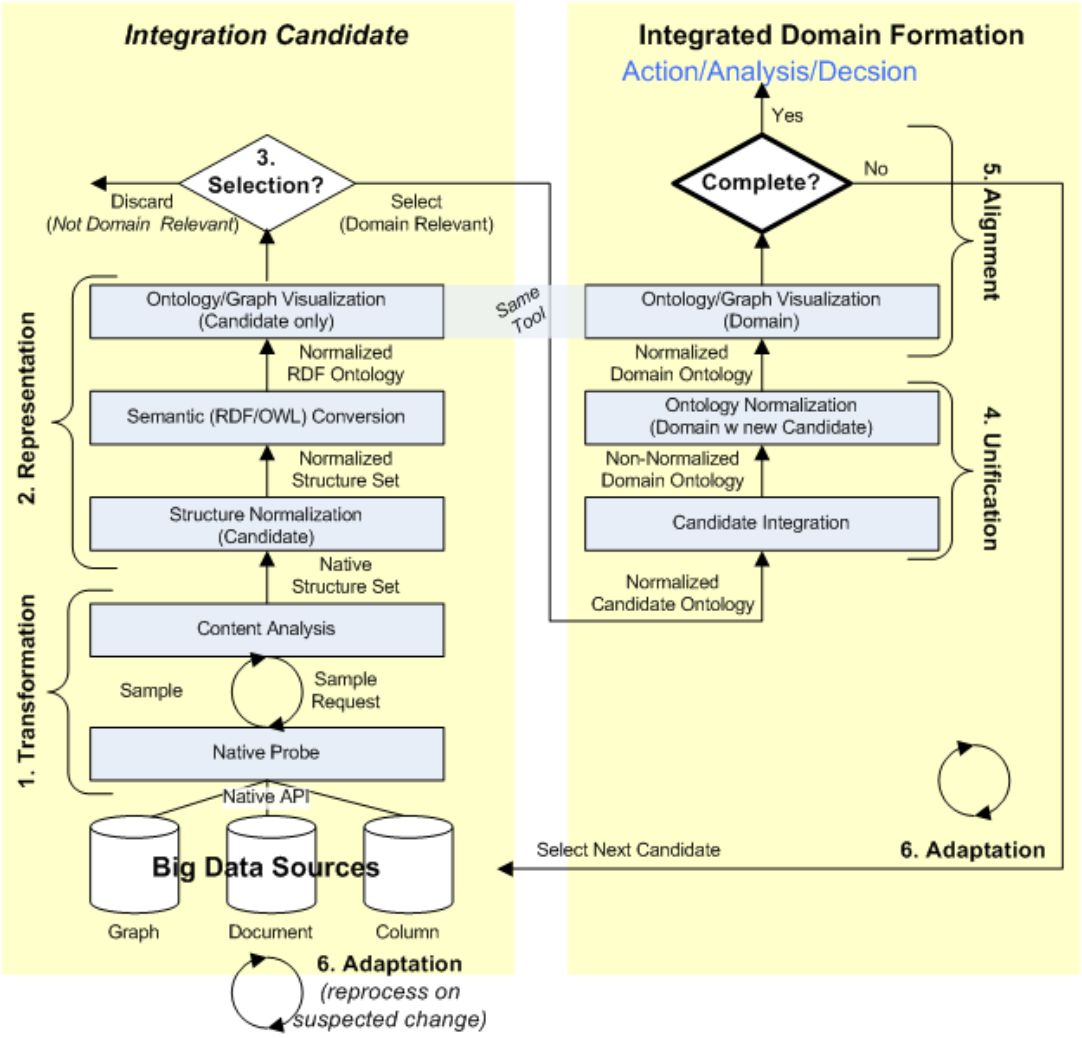

Figure 10: Technical Architecture

3.4.1 Data Sources

Integration depends on the data from databases. Semantic integration enables the inclusion of NoSQL data. The technical architecture selects several different examples of NoSQL technologies to best illustrate its abilities to handle and integrate data from various data technologies and sources. NoSQL stores exist in three major forms: graph

(e.g. Neo4J), document (e.g. MongoDB), and column (e.g. Cassandra). The method allows re-inspection of a given data source if changes occur or are suspected.

3.4.2 Transformation

The first step in processing information from a data source is transformation. This translates the various platform, technology, and formats found in a common data structure that includes the domain of the data and the technology factors required to integrate and query the data.

Semantic transformation consists of two steps: Native Probe and Content Analysis. The Native Probe uncovers the type of NoSQL technology by attempting to interact with the various communication ports and NoSQL protocols. This assumes standard ports, which is the typical case, while also allowing these values to be overridden. Figure 11 outlines the major steps.

Each NoSQL technology uses standard, defined ports and protocols. Thus, the native probe assembles a NoSQL technology characteristics list that includes the standard ports protocols, and programming interface for each supported NoSQL database. The probe selects the first set in the list and uses that information to attempt a connection to the identified candidate. If the selection is successful, the transformation continues to content analysis. If not, the probe moves to the next entry in the list. This continues until a connection succeeds or the probe exhausts the list. The latter case indicates that the probe was unsuccessful in identifying the candidate.

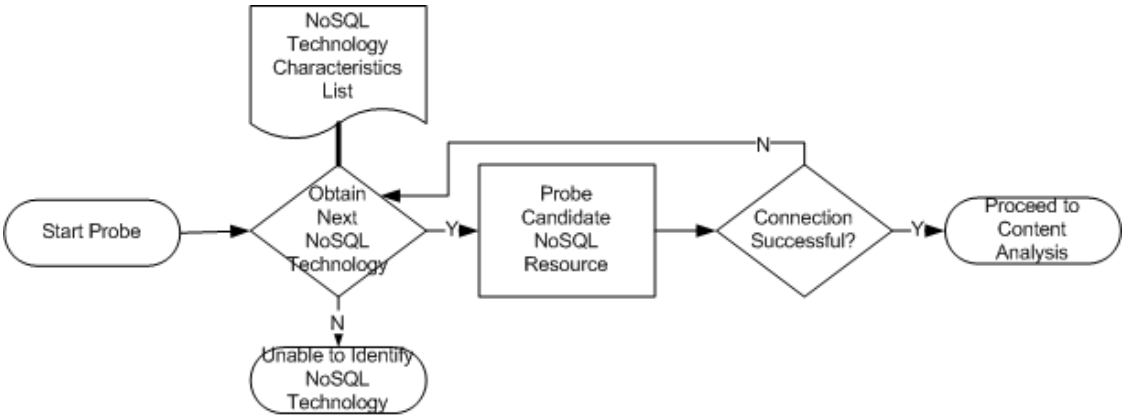

Figure 11: Native Probe Process Flow

Figure 12 outlines the Content Analysis process flow. The first step calculates the length and random location of the record request based on the type of store, table size, and resolution factor. The method then retrieves a random sample of NoSQL records.

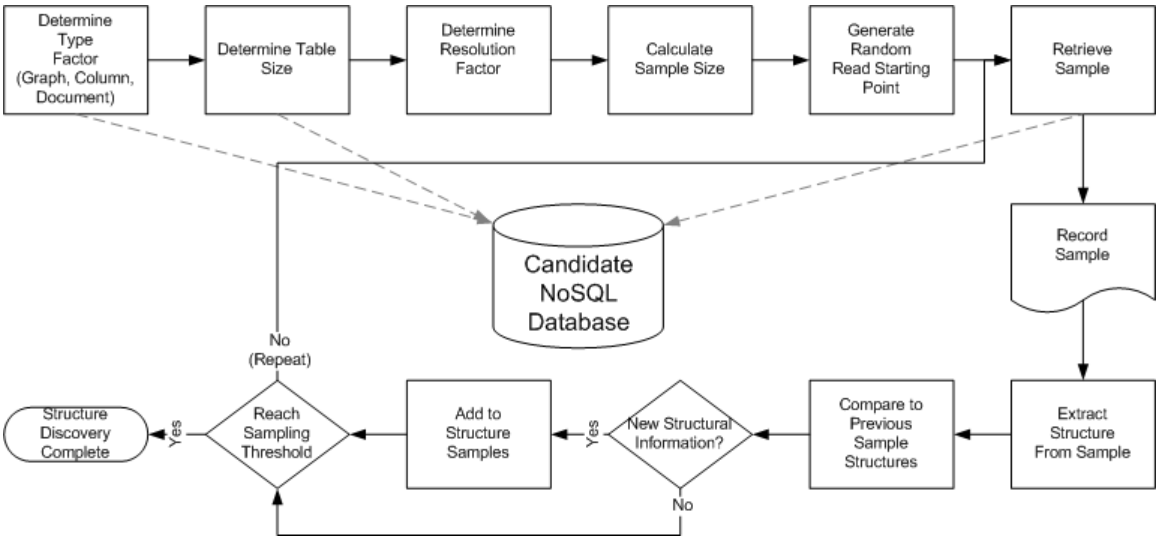

Figure 12: Content Analysis Process Flow

The method employs the sequential sampling method. This requires a sampling plan that outlines the sample size of records within a given table and the number of attempts to

ensure an effective review of possible structures within the table (Wald, 1945). These parameters can be adjusted as needed. Initially, the method uses the table size to determine the sample size and number of attempts. Basic field trials determine a useful sample size and convergence attempts without uncovering any new structural information. This sampling method could indeed miss some structural information. Given that the structural information is repeated for each record, dominant structures would almost assuredly be identified with raw structures possibility being missed. The user is free to tune the parameters according to their needs to identify all structures (takes more time) or just the main, popular structures (takes less time).

The received records are analyzed for structural information. For each new sample, the Content Analysis determines if the new records contain new structural information. The structures are compared using string matching for each structural member. The method would recognize similar but different structures as different, leading to some unnecessary duplication for the business user. However, these subtle differences are required for the actual integration with another database.

New structural information is added to the source description. The method continues requesting additional random records until the method reaches a steady state (hence, no new structural information is added after a defined number of trials). This results in a rapid review of the structural information without the need of exhaustive examination. Of course, using the sampling method may lead to missing a key structure (and hence data element) but structures in NoSQL repeat with each record. This provides a statistical view that may miss only a small amount of structural information, if any at all.

The factors can be adjusted to accommodate different needs as to the capture of the structural information. The speed of this approach allows fast rapid updates. In a sense, this approach integrates a comprehensive structural view across the single store. This is actually a form of integration since structural information in a NoSQL store can be quite diverse.

The Content Analysis could use simple or sophisticated comparisons to determine new structural elements. Given the relative simplicity of NoSQL stores, simple string comparisons may be sufficient. However, extensions using semantic logic and word similarities can extend this capability, if required. This could be accomplished iteratively if, as one uncovers the structure, additional semantic methods could help better reveal it.

Content Analysis completes with a set of structures extracted from the sampling process. The structures contain the databases, tables, and column elements with an example value for each structural element to aid comprehension.

3.4.3 Representation

Representation contains three steps: normalization, semantic conversion, and visualization.

Representation converts the structural information received from Content Analysis to a Semantic Web Ontology in OWL and RDF. This establishes classes for the database, database table, and class attributes for each column name, along with example values for

each data element. Relationships are also created to represent the containership of the database and tables.

Representation takes advantage of the various advances in ontology mapping and ontology alignment. The alignment API (Euzenat J. , 2008) allows a flexible way of mapping and sequencing multiple alignment analytics to unify the analytics results. This includes alignment confidence where the user can set the level of precision and recall, based on acceptable levels of false positives and false negatives.

Representation can leverage the many existing ontologies found across the Internet (Pride, 2010). Semantic Unification can use this service and others to find and integrate a compatible ontology. The ontology contains logic and context to enrich the integrated data while also correcting it.

Figure 13 details the transformation.

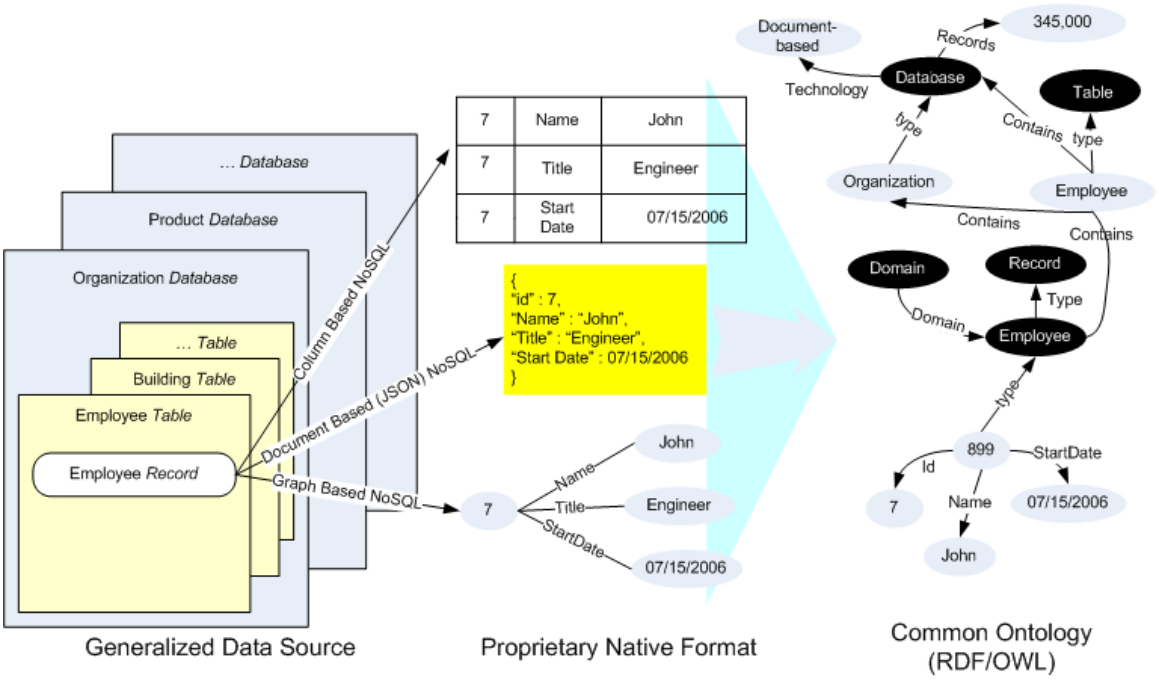

Figure 13: Semantic Conversion: From NoSQL to an Ontology

The left most boxes reflect the generalized nature of a NoSQL database containing databases, tables, and records. Each technology may employ different expressions, but they essentially are the generalized form of a NoSQL database. The database and table names typically provide semantic information as the above example demonstrates. NoSQL structures are simple and thus often rely on common, everyday words. They lack the need to construct complex relationships found in relational databases. The simplicity of NoSQL allows a simple capture of this structural information directly. Thus, the database and table names often provide useful structural semantic information. The middle column of the figure illustrates the three types of NoSQL stores. Starting from the top, they include column, document, and graph. These usually also contain useful

semantic information. This example shows the same record information in the three different formats.

The conversion takes structural information from both key areas: database and table name, as well as the record information, to form the ontology. This enables two dimensions within the ontology – a structural approach based on the technical structure and the domain structure. The technical structure allows navigation through the structure based on databases and tables names (including the size and technology type), which is essential to actual data integration. The domain structure allows the non-technical user to view the structure example data and determine its usefulness for a given business area divorced from the technical underpinnings. The record information also contains an example of the actual instance such as the *Name* being *John*. This allows an additional level of introspection on the structure if the record name is unclear. For example, if the column name was simply *ss* but the instance data was 123-45-8976. The user could correctly conclude that the *ss* column represents the *social security number*. The ontology also includes both the technical descriptions necessary to perform an actual technical integration such as the technology, database name, table name, and query string.

Given multiple structures within a table in NoSQL databases, a new table instance for each structure is created in the ontology as a new structure is uncovered. Therefore, a given table would contain several instances each containing different fields and different query information. The user is free to select any number of structures from the table. Additionally, a count of records mapped to that structure is included in the table, reflecting a magnitude of that particular structure.

Representation results in an RDF/OWL ontology that captures the structural information, the technical information required for integration, and actual parameters to extract the data if the candidate is selected. Figure 14 outlines the ontology structure used to describe the NoSQL database.

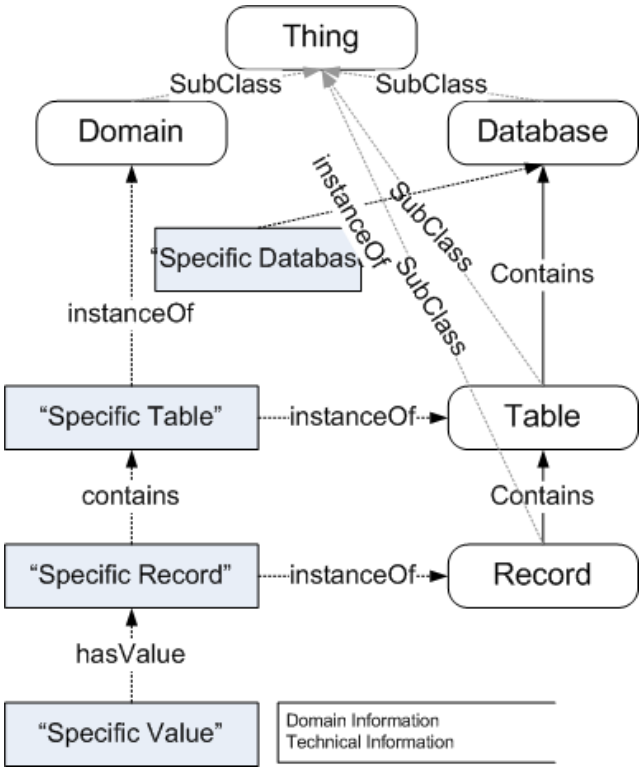

Figure 14: Foundation Ontology

The unshaded rounded rectangles represent classes. The shaded rectangles represent actual instances found in the database. The ontology enables two distinct paths through the structure. The technical path identifies the database and tables. The domain path just goes right to the tables. A table typically contains semantics regarding the contents and thus starts the domain. The table contains records or fields with each having a representative actual value. This allows the user to examine not only the table and field

names but also the values contained within the field. The values fall into two categories: domain or technical. The former contain the actual business information. For example, if the record name is “stock price” the value would be the actual price, such as $50. The latter contains technical integration information. A record count is included to indicate the amount of the information contained in the table.

This ontology provides a rich semantic context that includes the database name, the table name, the field name, and a sample field value. Together they provide a semantic understanding of the contents. Any one value by itself, may not be sufficient to determine the contents and certainly not enough to set up an actual technical integration. For example, just knowing the title of the database would typically not be enough to determine its value for a given business opportunity.

The table instance also contains the technical information to allow for integration such as the URL, port number, login information, and query parameters. The query parameters allow the integration to retrieve the table elements when needed directly from the native NoSQL source. Thus actual data movement is not required until requested by an alignment request. The data stays in place in the native NoSQL database. The integrated data is up-to-date.

The ontology is constructed by recursively exploring the structure data returned from the probe phase. The exploration starts with each database contained in the NoSQL technology. For each database, the tables are identified. For each table, the structures are recorded with example values for each field. Each record is connected to a table and each table connected to a database. Each table contains all of the technical information

required to retrieve data when needed. The various NoSQL technologies do not always use the same terms as databases and tables but the concept remains true across the wide range examined for this research.

Representation can also employ semantic reasoning and logic to further compress and enrich the ontology. For example, using WorkNet's ontology (University, 2013), the structure could compress similar words. A simple enrichment with a geographical ontology/database could expand MD to Maryland. Semantic reasoners could recognize similar patterns and add relationships. Additionally, machine learning or similar analytics could eliminate superfluous characters to clarify the actual meaning of the semantic word.

Ontology visualization displays the structure to a business user for their consideration. Ontologies offer many standard tools to visualize the structure and instance data. This allows a non-technical user to examine the structure, in semantic terms and relationships, for consideration in building a large integrated database. The ontology visualization can be driven by the SPARQL query language that focuses on the domain, the technical structure, or a given area through standard SPARQL calls, thus reducing the complexity of a large structure. Figure 15 provides an example of an ontology visualization.

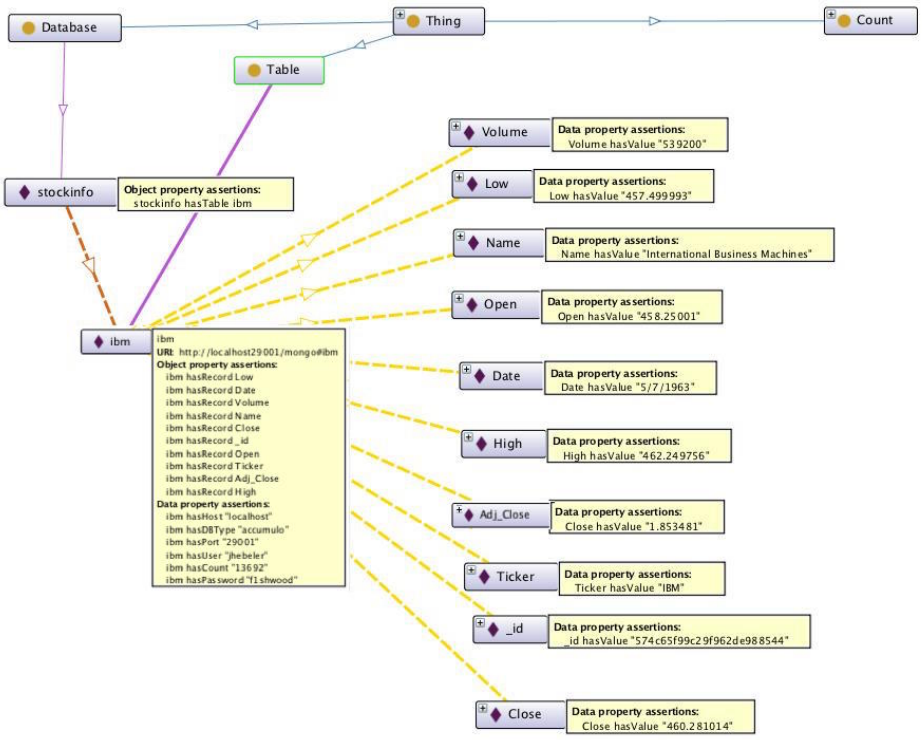

Figure 15: Structural Visualization Example

The ontology visualization example includes both the technical structure and the domain contents. The latter is key to evaluating its business use for integration of the data. The former is key to actually perform the technical data integration. In Figure 15, the database is called *stockinfo*, which contains a table called *ibm*. The table holds a record containing multiple fields such as *High*, *Date*, *Open*, and *Ticker*. One record is displayed with values to provide additional context to each field name. So 'hasValue' can be interpreted as the stock price. Additionally, the table provides technical information such as the host name, connection port, and technology type – all essential in performing an actual integration. The visualizations employed the open source ontology tool Protégé, which also validates the ontology constructs.

3.4.1 Unification

Unification consists of two stages: Candidate Integration and Normalization. The latter stage employs many of the same methods used in Representation since both data sets are in the standard ontology form of OLW/RDF.

Candidate Integration with the Semantic Web is a simple aggregation. Since each data source employed a unique descriptor (URI) in the ontology, the combined data sets do not contain any conflicts. All the statements simply combine.

Although there are no conflicts due to the uniqueness of the URI, the newly aggregated database could contain duplications at two levels – concept duplication (e.g., each integrated database contains a table statement that is the same concept in all the integrated stores), and instance duplication because both could contain the IBM Corporation. Despite the same information and/or the same concept, each contains different URIs reflecting their different databases origins. This not only prevents any conflicts but also obfuscates the same data artifact as two distinct items.

Unification identifies duplicated concepts through the same identification process used in the candidate normalization but limited to the last field in the URI. On recognition of duplication, unification uses OWL:equivalentClass for merging concepts and OWL:sameAs for merging instances. This treats two or more different records as logically equivalent. Thus, the OWL:sameAs applied to these two table URIs referred to as URI1 - http://nosql1/stocktransaction and URI 2- http://nosql2/stocktransactions merges them as completely equivalent. A query that used the first or second URI would

return both. If indeed they contain duplicated information, adding the unique keyword to the query would eliminate the duplication. If the duplication strains storage, a semantic rule could be applied to delete the redundant information. The flexibility extends to corrections. One only eliminates the OWL:sameAs statement to separate the concepts back into the two original different concepts.

Figure 16 illustrates an ontology normalization example. The original knowledge base contains a table, *ibm,* that contains stock information. The illustration presents two candidates to the right. Both could be integrated using OWL:sameAs. The upper right augments the table entry of *ibm* with office information. The lower right augments with additional stock information. The user can examine specific values to see if this database merely repeats the information or adds information from different dates.

If the user requested a query that referenced either the *ibm* table in candidate 1 or the *ibm* table in candidate 2, the aggregated results would return from the query. Thus, the query would return not only the stock price information from candidate 2 but also the office information from candidate 1.

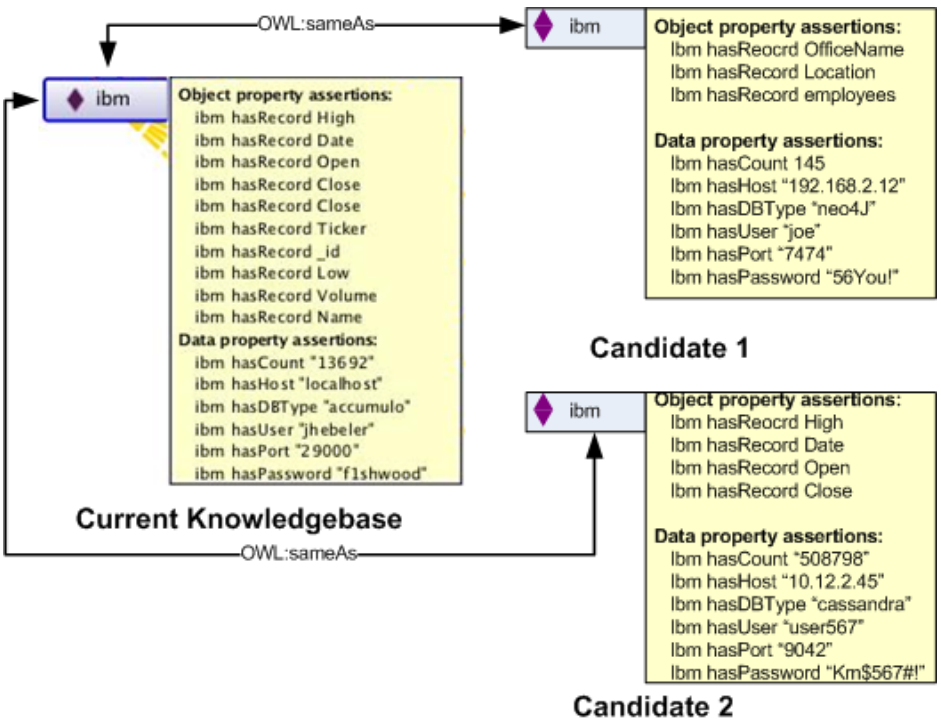

Figure 16: OWL sameAs Semantic Equivalence

A sample query is as follows:

```
Select * where {
<Candidate1 URL>/ibm ?relationship ?value.
}
```

Code Constructs 1: Integration SPARQL Query

The SPARQL query in Code Constructs 1 lists Candidate 1. The processing would expand the list to Candidate 2. This would then retrieve both tables from the different NoSQL databases. Each table contains the necessary information including the NoSQL database type and interfacing information, such as URL location, login information, and the native query command. In turn, each native table is queried with the results converted into an ontology. Unlike before, where only the structural information was

converted, the integration converts the entire results of the query representing all the pertinent data. This ontology is then integrated into the knowledge base simply by adding all the statements in it. The knowledge base already contains the equivalence statements and thus drives the normalization. Once complete, the entire query is rerun with the contained information now in the integrated database and provided to the user.

The integrated knowledge base remains an ontology that contains the logic fields with the Semantic Web equivalence statements used to normalize the ontology.

The actual data integration employs the technical fields contained in the ontology to extract the data requested for the integration. This includes the address or URL, the port configuration, login information, and query parameters. This can be accomplished in two ways – forward or backward integration. Forward integration pulls in the data at the time of the integration. Backward integration pulls the data when the user requests the data from a query. Forward integration allows fast retrieval of the data by the user, as the data is already present but may require more space to fill all the data – much of which might never be requested. It also quickly gets out of sync with the native store. Backward integration maintains the latest data since it is coming directly from the source. Backward integration is the preferred method since it doesn't duplicate information and always has the most current information. With proper architecture, the retrieval of information from multiple stores can offer high performance since the operations can happen in parallel.

3.4.2 Semantic Alignment

Semantic alignment uses the visualization of the integrated database to determine if the database is complete. Semantic alignment uses the same standard ontology viewing tools as the Representation stage does. Nevertheless, the visualization contains the integrated database not just the candidate.

The ontology visualization tools allow selection and filtering of the integrated store. Additionally, a query using SPARQL can further focus the visualization on the area of concern.

Semantic Alignment maintains the focus of the integrated data on the user's domain of interest. The alignment can offer multiple steps to allow the user to converge and focus the domain data on the intended integrated business data results. Additionally, SPARQL and corresponding ontologies provide a rich framework to capture the intended area of interest. The results from SPARQL queries allow the visualization to adapt to the user needs. For instance, the user might just want to see the top-level structures or drill down to one specific lower-level structure and examine its members. This allows the user to navigate through complex structures.

The actual integration of the native data happens in direct response to a user query. The query is based on the tables and fields exposed in the ontology. However, the ontology contains only the metadata describing the integrated structure and associated technical information for each native NoSQL database. The ontology is useful in determining the completeness of the integrated data. When the user executes a query to retrieve the

integrated information, the ontology information is used to query the native NoSQL database. Figure 17 details the process to retrieve actual integrated data from the native NoSQL database.

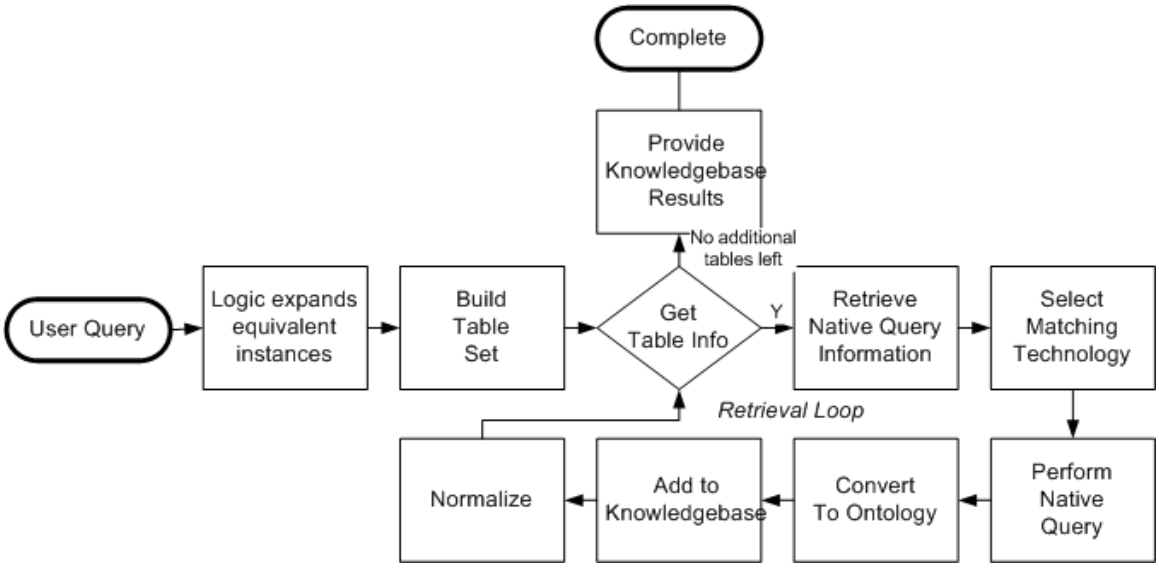

Figure 17: Integrated Data Processing

The user initiates a query against the ontology containing the structural and technical information of the various integrated databases. The reasoner extends the query to include equivalent instances listed in the query reacting to the OWL:sameAS statements. The query then retrieves the requested tables. Each table in the ontology, as outlined earlier, contains the technical information to interact with the native NoSQL database. This includes the database type. This type is matched up against the set of database characteristics that provide the actual access commands. After matching the correct one, the native commands execute the query contained in the table to retrieve its actual contents. This data in the native format is then converted as the previous step into an

ontology. This time, all of the data is converted, not just the structural information. This ontology is then added to the integrated database. In a nutshell, the integration is adding ontology statements to the integrated database. Normalization occurs due to any existing equivalent statements. After each table is queried, the query is re-run against the newly populated knowledge base and the integrated results are returned to the user.

In summary, Table 2 outlines the stage, input, output, and verification of semantic data integration. This provides the inputs and outputs of each step in the method along with the verification method. Additionally, it notes whether the step is automated or not. The non-automated steps form the convergent dialog with the user. Each step contains a validation process to ensure that the data complies with syntax rules. Once the information is converted into an ontology in the later steps, verification of the soundness of the ontology can leverage existing ontological reasoning tools.

Table 2: Integration Method Details

Stage	Activity	Automated?	Input	Output	Validation
Transformation	Native Probe	Yes	NoSQL database	Record Samples	Valid NoSQL Response
	Content Analysis	Yes	Record Samples	Record Structures	Structure Validation
Representation	Structure Normalization	Yes	Record Structures	Unique Structure Set	Duplication Validation
	Semantic Conversion	Yes	Unique Structure Set	RDF/OWL Ontology	Ontology Verification
	Candidate Ontology Visualization	No	RDF/OWL Ontology	Visualization of Ontology	User Acceptance or Rejection
Unification	Candidate Integration	Yes	RDF/OWL Candidate Ontology	RDF/OWL Aggregated Ontology	Ontology Verification
	Ontology Normalization	Yes	RDF/OWL Aggregated Ontology	Normalized RDF/OWL Aggregated Ontology	Ontology Verification
	Integrated Ontology Visualization	No	RDF/OWL Ontology	Visualization of Ontology	User Acceptance or Rejection
Alignment	Query	No	Integrated Ontology	Requested Data	User Acceptance or Rejection
Adaptation	Change in opportunity	No	New Database	Requested Data	User Acceptance or Rejection

3.4.3 Model Implementation Application

Several key technologies support the implementation of the various components as noted above.

- OWL2: W3C standard Ontology and Instance Language. It supports description logic and many standard technical representations including an XML and JSON format.

- SPARQL1.1: W3C standard Semantic Web query language. SPARQL endpoints represent a complementary W3C standard that provides a RESTful service that allows remote, federated SPARQL queries.
- Reasoners: Various implementations carry out inference across assertions within a Semantic Web ontology. The reasoners allow various logic coverage options to allow performance tuning.
- Rule Languages: Rule languages form extensions to the logic expressed in the ontology. They allow operations outside of description logic. They include Jena rules, SWRL, and SPIN.
- Triple Stores: Multiple implementations provide storage and access to the triple store contained in a Semantic Web data resource. Triple stores also provide transformation tools and may perform reasoning.
- Jena: Semantic Web Programming Framework that allows connections to storage, SPARQL interpretation, ontology and instance formation, and reasoning coupling.
- Visualization: Tools include Protégé among others.

3.5 Semantic Integration Model Implementation

The implementation requires a set of appropriate data candidates that contribute to a useful domain while also exercising various NoSQL technologies. This forms a useful integration implementation that simultaneously validates the approach using different NoSQL technologies. The candidates are then integrated into the Semantic Web application that allows a convergence, working with the user, to the desired integrated results.

The identified data source candidates are developed into a semantic integration implementation that properly represents and integrates the unified information. The development effort adheres to the technical architecture (see Figure 10: Technical Architecture). This includes transformation, representation, unification, alignment, and adaptability. Each development step contrasts with traditional integration methods as outlined in Figure 2.

3.5.1 Use Case – Building a diversified financial portfolio

The implementation employs a use case that requires data integration of several different NoSQL databases. A business user, who has no technical skills, needs to identify stocks that outperformed several major financial market indexes over the last five years for assembly into a financial portfolio. This use case requires data from multiple financial data sources holding investment data including specific stock transaction histories and multiple stock exchange histories.

The desired stock information is publicly available in multiple databases, all of which are out of the individual's control. The user depends on the up-to-date documentation, if it exists, to determine its contents. The semantic integration model allows the non-technical user to examine and review the various databases, without documentation, and be assured that the information is current. The model produces an ontology representing the database and visualizes the ontology to allow the user to inspect the database. The user reviews the visualized ontology, which contains the data structures. If the user determines that the data would be useful, the ontology is aggregated into the integrated

ontology. The integrated ontology contains all the data structures and technical data required for integrating the user-selected data. The user can continue to review additional sources and, if warranted, integrate them. With each integration, the user examines the integrated ontology to see if the integrated data would be sufficient to address the requirements for building the portfolio. This allows an incremental and iterative approach to integration with the non-technical user driving the integration effort. It also allows learning through better understanding of the data possibilities. Once completed, the user can request a query of the desired data. The query uses the technical data in the ontology to retrieve the data directly from the native source. If the returned data is deemed insufficient, the user can continue the process by integrating additional NoSQL databases.

Figure 18 outlines the workflow for employing the method.

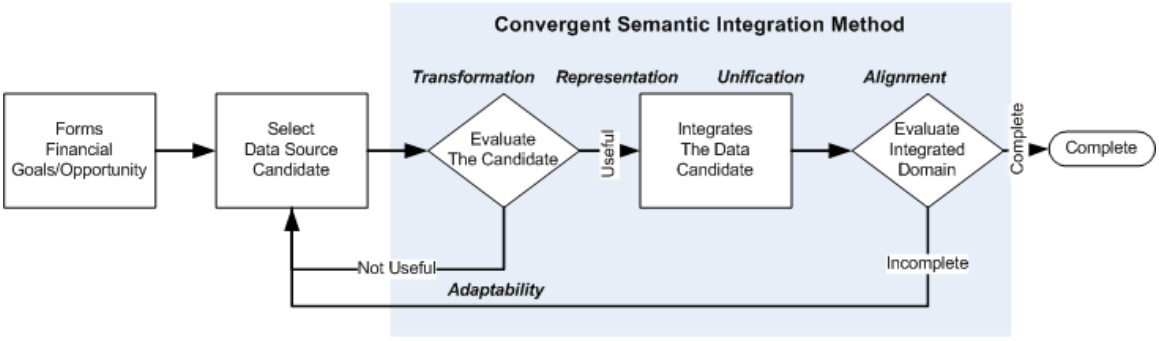

Figure 18: Financial Use Case for building a stock portfolio

3.5.2 Semantic Integration Model Implementation

The semantic integration model implementation realizes this use case. Several NoSQL data sources are selected with each using different NoSQL technologies. The semantic integration method starts with the transformation phase, which is initiated with the native probe to attempt to connect to the data source. The native probe attempts to connect using the list of parameters for each of supported NoSQL databases. Code Constructs 2 covers the specific interface required for the probe. The probe contains a method structure for the key operations to interface with each NoSQL database. Each supported NoSQL database creates a Class that instantiates each of these methods. Using this generalized approach allows for more conceptual operations across different NoSQL databases and minimizes NoSQL specific software. This provides polymorphism that isolates the specific details for each of supported NoSQL databases – this allows fast incorporation and updating of evolving NoSQL technologies.

```
public interface Probe {
        int MAXSAMPLES = 40;
        int BATCHSIZE = 100;
        public boolean connect(String host, int port, String user, String pw);
        public int extractStructures();
        public Model convert2RDF();
        public boolean sameAs (Object first, Object second);
        public boolean populate();
}
```

Code Constructs 2: NoSQL Probe Interface Methods

The probe code maintains a set of Classes that each implement the probe interface outlined in Code Constructs 2 for the different NoSQL technologies. The probe steps through each entry starting with the connect method. If it succeeds, the connect method returns a true value and the probe continues to extract the structural information noted in Code Constructs 3 for each of the traversed tables. Note that this includes the technical information as well as the elements contained in the array. Code Constructs 3 notes the technical information listing the structural elements uncovered.

```
public class DBStructure {
        public enum DatabaseType  {mongo, accumulo, neo4j, cassandra};
        private String dbName = null;
        private DatabaseType databaseType = DatabaseType.mongo;
        private String tableName = null;
        private String query = null;
        private long rowCount = 0;
        private String hostName = "localhost";  // default host
        private int portNumber = 27017;  // default port
        private String userName = null;
        private String password = null;
        private ArrayList<Object> structures = new ArrayList<Object>();
```

Code Constructs 3: Structure Extract Information

Once the connection is established, queries are performed in accordance with the random sample method detailed earlier. Each record is examined for structural information. In the content analysis portion of transformation, the structure information is compared to the structural information collected so far. If the structural information is new, it adds to the collection until no further new information is found.

The content analysis starts with a listing of databases for NoSQL database, which can actually contain multiple databases. The probe recursively searches each database, then each contained table, and finally a specific sample record from each table. Code Constructs 4 outlines the methods used to extract table and technical information.

```
// Next see if table is new
                        if(!Objects.equals(currentTable, table)){
                                // Create table object
                                tableInstance = m.createResource(URIdb
+  currentTable);
                                m.add(tableInstance, isType, tableClass);
                                table = currentTable;
                                // Associate with database
                                m.add(databaseInstance, hasTable, tableInstance);
                                Resource counterInstance = m.createResource(URIdb +
currentTable+ "counter");
                                m.add(counterInstance, isType, counter);
                                m.add(counterInstance, hasValue,
Long.toString(dbs.getRowCount()));
                                m.add(tableInstance, hasHost, dbs.getHostName());
                                m.add(tableInstance, hasPort,
Integer.toString(dbs.getPortNumber()));
                                m.add(tableInstance, hasUser, dbs.getUserName());
                                m.add(tableInstance, hasPassword,
dbs.getPassword());
                                m.add(tableInstance, hasDBType,
dbs.getDatabaseType().toString());
                                m.add(tableInstance, hasQuery, dbs.getQuery());
                                m.add(tableInstance, hasCount,
Long.toString(dbs.getRowCount()));

                                // Also set the table up as a class with instances
for each row
                                domainInstance = m.createResource(URIdomain +
```

```
currentTable);
                                m.add(domainInstance, isType, domainClass);

                        }
```

Code Constructs 4: Probe Table and Technical Information Retrieval

For each table traversed, all technical information is recorded including the query parameters. This provides the technical information required for data integration of this native database when requested. The ontology acts only as a proxy so as to maintain the most up-to-date structural and technical data and forgo the need for additional storage. This technical information provides a path back to the native NoSQL database.

Lastly, the probe collects record information, as shown in Code Constructs 5.

```
while(saveObjIt.hasNext()){
                                        // Allocate one record structure for each
element                         databaseObj = m.createResource(URI+  tableName);
                                        recordInstance =
m.createResource(URIdb+  saveObjIt.next().toString());
                                        //Resource databaseClass =
m.getResource(URIdb);
                                        m.add(recordInstance, isType, recordClass);
                                        //m.add(recordInstance, isType,
domainInstance);
                                        // Connect each record to the specific table
                                        m.add(tableInstance,hasRecord,
recordInstance );
                                        // Connect each record to the specific
domain
                                        m.add(recordInstance, isType, domainClass);
```

```
                                        m.add(domainInstance,              hasMember,
recordInstance);
```

Code Constructs 5: Probe Record Retrieval

Representation is the next processing step. The first part is normalization to eliminate duplication. This is done by determining whether or not to add the structural information via a comparison method. This is outlined in Code Constructs 6. It steps through existing structural elements and compares them to the new structure. A structure could contain many elements so each element needs to be compared. Since a slightly updated structure does not match any of the existing structures, it is added to the structure list. This approach allows the user to compare the usefulness of the two similar structures. The user might see a need for both, one, or neither.

```
public boolean sameAs(Object saveObj, Object obj) {
                    DBObject one = (DBObject) saveObj;
                    DBObject two = (DBObject) obj;
                        boolean different = true;

                        Map saveObjMap = one.toMap();
                        Set saveObjSet = saveObjMap.keySet();
                        Iterator saveObjIt = saveObjSet.iterator();

                        Map objMap = two.toMap();
                        Set objSet = objMap.keySet();
                        if( objSet.isEmpty() ){
                                return true;
                        }
                        Iterator objIt = objSet.iterator();

                        while(saveObjIt.hasNext()){
                                String compareKey = saveObjIt.next().toString();
```

```
                        if(!two.containsField(compareKey)){
                                different = false;
                                System.out.println("KEY NOT FOUND");
                                break;
                        }
                }

                return different;
        }
```

Code Constructs 6: Normalization

Representation continues after the normalization to convert to a Semantic Web ontology in OWL and RDF. This requires the creation of OWL Classes for the main database artifacts such as table and instances of those classes. Additionally, the relationships with the database are maintained. Tables contain records and records contain fields. Fields contain the actual data. The technical information is also associated with each table. The reference model implementation contains the following methods in Code Constructs 7 to form the conversion to OWL/RDF for the table information and Code Constructs 8 handles the record information, which includes actual instances to aid the semantics of the field name and table name.

```
for(DBStructure dbs:structures){

                currentDatabase = dbs.getDbName();
                currentTable = dbs.getTableName();
                System.out.println("CURRENT TABLE: " + currentTable);


                // Set database object if new
                if(!Objects.equals(currentDatabase, database)){
                        // Create database object
```

```
                                databaseInstance =
m.createResource(URIdb+  currentDatabase);
                                m.add(databaseInstance, isType, databaseClass);
                                // Set this up so we don't repeat above
                                database = currentDatabase;
                        }

                        // Next see if table is new
                        if(!Objects.equals(currentTable, table)){
                                // Create table object
                                tableInstance = m.createResource(URIdb
+  currentTable);
                                m.add(tableInstance, isType, tableClass);
                                table = currentTable;
                                // Associate with database
                                m.add(databaseInstance, hasTable, tableInstance);
                                Resource counterInstance = m.createResource(URIdb +
currentTable+ "counter");
                                m.add(counterInstance, isType, counter);
                                m.add(counterInstance, hasValue,
Long.toString(dbs.getRowCount()));
                                m.add(tableInstance, hasHost, dbs.getHostName());
                                m.add(tableInstance, hasPort,
Integer.toString(dbs.getPortNumber()));
                                m.add(tableInstance, hasUser, dbs.getUserName());
                                m.add(tableInstance, hasPassword,
dbs.getPassword());
                                m.add(tableInstance, hasDBType,
dbs.getDatabaseType().toString());
                                m.add(tableInstance, hasQuery, dbs.getQuery());
                                m.add(tableInstance, hasCount,
Long.toString(dbs.getRowCount()));

                                // Also set the table up as a class with instances
for each row
                                domainInstance = m.createResource(URIdomain +
```

```
currentTable);
                                m.add(domainInstance, isType, domainClass);

                            }
```

Code Constructs 7: Ontology Creation from Table Information

Code Constructs 8 shows the code for creating the record and field ontology.

```
for(Object d: dbs.getStructures()){
                        DBObject dbo = (DBObject) d;
                            Map saveObjMap = dbo.toMap();
                            Set saveObjSet = saveObjMap.keySet();
                            Iterator saveObjIt = saveObjSet.iterator();
                            Collection instances = saveObjMap.values();
                            Iterator instanceInterator = instances.iterator();
                            while(saveObjIt.hasNext()){
                                    // Allocate one record structure for each element
                    databaseObj = m.createResource(URI+  tableName);
                                    recordInstance =
m.createResource(URIdb+  saveObjIt.next().toString());
                                    //Resource databaseClass = m.getResource(URIdb);
                                    m.add(recordInstance, isType, recordClass);
                                    //m.add(recordInstance, isType, domainInstance);
                                    // Connect each record to the specific table
                                    m.add(tableInstance,hasRecord, recordInstance );
                                    // Connect each record to the specific domain
                                    m.add(recordInstance, isType, domainClass);
                                    m.add(domainInstance, hasMember, recordInstance);
```

Code Constructs 8: Ontology Creation from Record/Field Information

Once the ontology is created, the model implementation uses the ontology tool Protégé to validate and visualize the ontology for the user review. Figure 19 shows the visualization of the stock portfolio ontology for review.

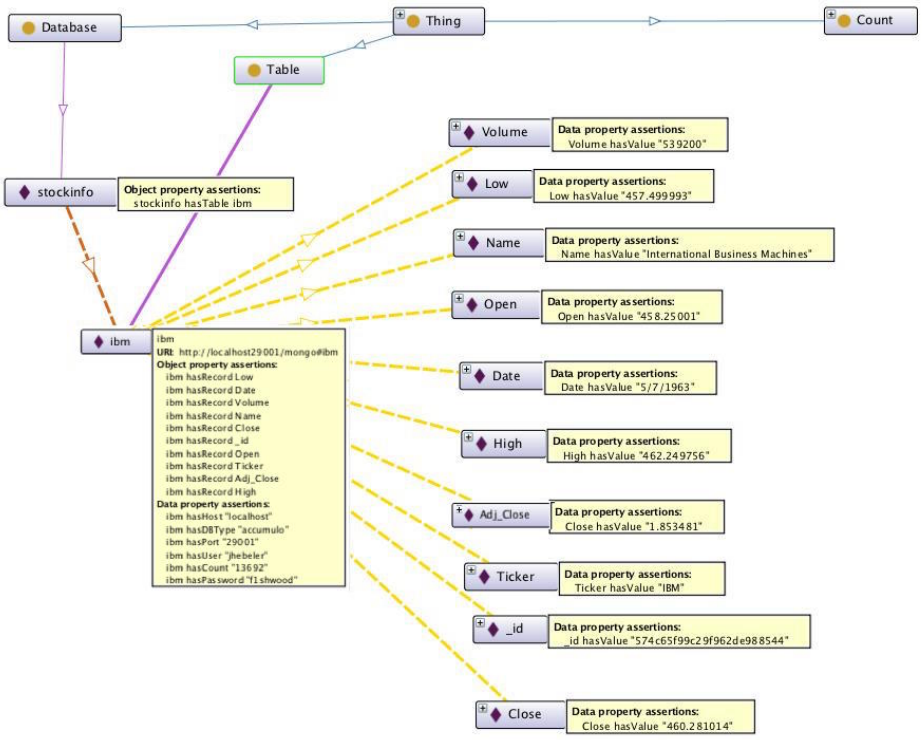

Figure 19: Financial Visual of Candidate Database

If the visualization of the ontology developed for the candidate database is not considered useful and relevant, the candidate ontology is discarded.

If the user selects the database, they can further narrow the scope of the integration by selecting specific databases, tables, and even fields. The user need not integrate the entire NoSQL database. The selected database ontology artifacts are added to the integrated ontology store. The first selected database forms the initial knowledge base.

Next the ontology is normalized to remove redundancies. This takes advantage of various Semantic Web logical constructs. Notably, OWL:sameAs and OWL:equivilentClass. The former logically merges instances such as *International Business Machines* with *IBM*. The latter merges classes such as URI1:table with URI2:table This allows two or more different ontology elements to be considered the same from a logic perspective. The logic can use both the field name and its value to determine if the two are actually the same. Additionally, logic can make similar words equivalent. The ontology still maintains the inherent differences by not changing the URI contained in each statement. This enables the native origin to remain intact and available because each one represents data from a different database and must integrate data from that source. For example, the OWL:sameAs statement connects two instances logically but the original statements with their different URIs are untouched. Thus, if the related OWL:sameAs statement is removed, the ontology reverts to the two being different. This flexibility allows for corrections. Thus a virtual database is created from the integrated database, yet it still retains the underlying differences that are found in the URI. The addition of these statements requires an ontology reasoner, of which there are many available, or the creation of an ontology rule. Code Constructs 9 illustrates merging the *ClosingPrice* instance from one ontology to the *Close* instance in another ontology as well as merging two classes with the same name, *Table*, but in different ontologies with different URIs. This requires a reasoner to properly extend the merge to the impacted statements. The reasoning can occur at the query request (backward chaining) or at integration time (forward chaining). The semantic integration method employs backward chaining to maintain the up-to-date data and therefore does not need additional storage.

```
<rdf:Description rdf:about="http://uri1/ClosingPrice">
  <owl:sameAs rdf:resource="http://uri2/Close"/>
</rdf:Description>

<owl:Class rdf:about="http://uri1/Table">
  <equivalentClass rdf:resource="#http://uri2/Table"/>
</owl:Class>
```

Code Constructs 9: Semantic Equivalence

The visualization contains the integrated ontology and equivalent semantic statements after activating a reasoner. The user evaluates the integration knowledge base for completeness for their specific task. If complete, the user initiates a query against the integrated database. The integrated database contains the information necessary to retrieve data from the various original databases because each ontology table concept holds the connection information, login information, and query parameters. If incomplete, the user needs to identify additional candidate NoSQL stores and repeat the above process.

The visualization allows the user to determine alignment towards their particular goals.

Adaptability can be driven by two sources of changes: change in user needs or change in the candidate NoSQL databases. The user is thus free to drop portions of ontology that are no longer needed or investigate a new or possibly updated NoSQL store for inclusion. The efficiency of this step enables flexibility. The user can examine the current state of the NoSQL database through the semantic integration method and, if necessary, replace the original ontology to reflect that data source. Thus, the user can quickly adapt to changes in their business direction or changes in the underlying NoSQL databases.

3.6 Design Science Research Method

Fundamentally, the research method for the semantic integration model is based on design science and its emphasis on proven artifacts (Hevner & al, 2004). A semantic integration model represents the key artifact that is proven through actual implementation and evaluation.

The design science method resulted in the following:

1. A semantic integration model that advances the overall integration of NoSQL databases.
2. Semantic integration methods that transform, represent, unify, align, and adapt the data.
3. Semantic integration model implementation that illustrates and demonstrates the key constructs of the semantic integration model.

Design science represents seven key guidelines (Hevner & al, 2004). Table 3 outlines the guidelines and research outcomes for the semantic integration model.

Table 3: Design Science Guidelines and Research Outcomes

7 Guidelines	Outcome
Design as Artifact	Semantic Integration Model, Methods, & Implementation
Problem Relevance	Data Integration with Large Data Sets (NoSQL)
Design Evaluation	Quantitative and Qualitative Evaluation with Tasking
Research Contribution	Improved Method of Large Scale Data Integration
Research Rigor	Ontology Rigor and Tasking (Analytic Methods)
Design as a Search Process	Built upon existing tools and methods
Communication of Research	Technical and Business Oriented Results

4 Chapter 4: Evaluation

The goal of the semantic integration method is to assist a non-technical user who is addressing an opportunity that benefits from data integrated from multiple NoSQL databases. The method allows the review and integration of multiple NoSQL databases with a minimum of technical assistance. This brings the opportunity closer to the data than with traditional methods that require translations to technical personnel. The improved efficiency allows the inspection, review, and integration of more databases within the users' timeframe leading to a more informed outcome.

The evaluation centers on exercising the semantic method implementation through a defined task. This includes a survey that enables the comparison of other integration actions and methods along with the ease-of-use and effectiveness of the semantic integration method.

The semantic integration method contains many steps that are transparent to its actual user and represent the internal processing of the method. These *hidden* steps include the identification of the NoSQL database technology, the extraction of the structural information, the conversion to an ontology, the integration of the various ontologies from the native NoSQL stores. The method permits two key elements for inspection by the user, allowing the semantic dialog of data integration convergence: The first is the ontology that represents a candidate database and second is the newly created ontology that represents the current integrated data. These two areas are highlighted in Figure 20. An ontology visualization of the candidate database, as shown in the top-left in Figure 20,

requires traversing all the proceeding steps from the native probe through the normalization. A similar ontology visualization representing the integrated database, as shown in the top-right of Figure 20, requires the candidate integration through candidate normalization with the existing database.

The evaluation of the method exposes these ontology visualizations to the user within a given context to determine the usefulness of the overall method. The ontology visualizations provide a testable artifact that represents the underlying operations of the method.

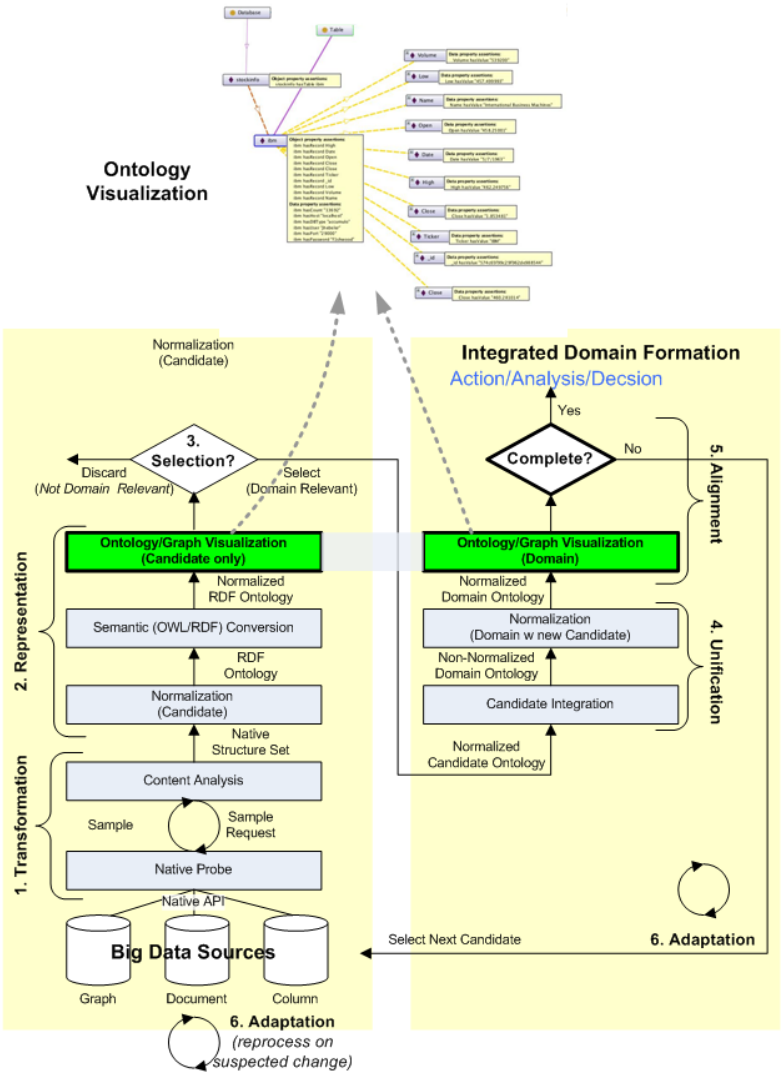

Figure 20: Two Ontology Visualizations

Rather than producing a conceptual version of the visualization, a model implementation was developed to produce ontologies from actual NoSQL databases. The representation results in ontologies that are then visualized for the user. The model implementation used several actual real-world databases with different NoSQL technologies and produced corresponding ontology visualizations without modifications to the underlying semantics contained in the various database artifacts, such as table names and field names.

This produced a ‘worst case scenario’ since no semantic enhancements were made to the database structure names, such as tables and fields. This worst case scenario can be improved using ontology reasoners and related ontologies. The user sees the same names used by the database technical creators - the names used in the database were the same names communicated via the ontology. If necessary, the ontology semantics could be enriched with synonyms from the WordNet (Miller, 1995) (Fellbaum, 1998). The review also tests whether this enrichment is necessary or the inherent terms used in the NoSQL database are sufficiently semantically rich to convey a meaning to the business user. The model implementation contains code constructs written in Java containing the Jena semantic library for ontology construction and the various drivers that establish interactive connections with the NoSQL databases. The model implementation code is available at the github repository (https://github.com/jhebelerDS/BigDataIntegration). The major code constructs were detailed in the previous chapter.

The model implementation probes for various NoSQL technologies. Once a connection is successful, it generates an RDF/OWL ontology that describes the given NoSQL database, including its hierarchical structure, instance examples, and technical parameters, such as the NoSQL technology, connected ports, and login information. The method then uses *Protégé* to validate and visualize the ontology. These ontology visualizations then become the artifacts produced to represent the model. It should also be noted that the model implementation was tested against *Accumulo*, a column based NoSQL database and *MongoDB*, a document based NoSQL database. The testing included known databases that contained multiple structures within a given table throughout the database store to test the underlying method’s suitability in extracting the entire structure of

various NoSQL databases. The known databases were constructed for testing purposes and deposited structures in various places throughout the database including at the beginning of the storage area, the end of the storage area, and randomly mixed throughout the storage area with other structures. In all cases, it successfully extracted the structure. However, this is by no means exhaustive and further testing is required to fully deploy the method.

The model implementation integrates the selected candidate NoSQL databases into a unified ontology. There it performs ontology normalization by uncovering duplication and using OWL:sameAs and OWL:equivalentClass statements to logically eliminate the duplications. The original ontology technical information remains intact, allowing queries back to the original source when required for the actual use of the data. This provides the most up-to-date data without requiring any additional storage.

4.1 Evaluation Methods

The evaluation of the semantic integration method follows Hevner et al. (2004). The evaluation answers the following questions: *Does the proposed semantic integration framework enable a non-technical user to quickly review the value of a candidate and the integrated NoSQL database without any technical assistance and in minimum time? Can the review list specific data elements to be integrated?*

The main evaluation approaches are classified into qualitative and quantitative methods. The qualitative approach is exemplified by case study and interview, and the quantitative approach examines the artifact qualities by performing an assigned task. Both types of

evaluation methods are employed to provide triangulation for validating the impacts of the semantic integration method, as introduced in Section 3.4.3. In both approaches, the participants are asked to contrast current integration methods with the semantic integration method. The participants are not technical users of NoSQL databases, rather they are users who recognize the value of data but are not familiar with the underlying technologies and database concepts. The model implementation produces and visualizes ontologies from several existing NoSQL databases. As discussed above, the visualized ontology becomes the main evaluation instrument to assess the utility of the proposed semantic integration model.

4.2 Qualitative Approach: Interview

The goal of the qualitative evaluation approach is to obtain an executive view on the value of the proposed approach for integrating NoSQL databases and insights as to how their organization currently evaluates and integrates NoSQL databases. Interviews are a key technique for IS case study research (Benbasat et al. (1987)). Providing rich data, interviews are appropriate to evaluate the potential utility of design artifacts in a business setting (Adomavicius et al. 2008).

Interview Process

The interview follows an interview script that was pretested through multiple reviews with potential candidates to ensure the proper interpretation and understanding of each question. The interviews cover questions on a participant's background, views on NoSQL/Big Data, their ability to perform the ontology tasks used in the survey, and asks

them to contrast the semantic integration method with their current methods for NoSQL integration.

Participants

The interview participants consisted of three executives from two major companies (>100,000 employees with a global presence), all of whom are involved with NoSQL databases. In addition, they represent several different business domains covering information systems. They each have managed information systems projects, customers, and employees for many years. They are familiar with the business value of NoSQL data sources and associated big data challenges. However, they do not have technical hands-on skills and are not directly involved in technical work. The companies are well known for their technical acumen and accomplishments.

Findings

The interview was conducted in the office of each executive using a computer. Detailed notes were taken because the companies forbade direct recording. Each interview took approximately 1 hour.

The interview findings are summarized as the following:

1. What is your current role/position and number of years?

   Three senior managers were interviewed and their years of experience were 7,25, and 34 years.

Each executive manages and leads a technical group of software developers, system engineers, and project managers. They design, develop, and support advanced information systems, many of which are highly data dependent. Their work often employs NoSQL databases and relies on massive big data architectures. All three handle a variety of business domains including financial, health, and government agencies. Their current role is not technical and they do not directly interact with the technologies.

2. How familiar are you with Data Management Technology? (Relational, Object, NoSQL) Please describe your experience.

   Each was familiar with the technologies from a business perspective but had no hands-on experience. Their organizations have been involved with big data technologies for over five years – "practically since their inception".

   "Primary experience has been with relational databases"

   "Familiar with the concepts but do not use it directly"

   "Been with the technology practically from its inception"

3. What is your role with respect to selecting and implementing data management technologies used in your organization? (Relational, Object, NoSQL)?

   None of them played in a role in directly selecting a data management technology. This was the responsibility of a technical person on their staff.

"I have direct responsibility and authority for approving data management technologies"

4. How many years of experience do you have with data management technologies? (Relational, NoSQL technologies) How do you currently use NoSQL databases in your organization? Is use increasing or decreasing? What is the most significant value of NoSQL technology in your organization?

   Each executive maintained many projects and programs that included data integration. NoSQL has emerged has a key technology along with relational databases. NoSQL databases along with their associated processing are now in many of the projects and are a key technology for their organization's success.

   "I was surprised how relational databases still offer benefits".

   "Data management technologies are a core offering of my business unit. In particular, databases such as MongoDB are foundational to many of our contracts, especially in support of efforts related to aggregation, e.g., MapReduce."

5. What are the most significant challenges in leveraging NoSQL databases (review, use, and integration, and analysis)?

   The number one problem for all three participants is finding skilled employees to leverage the NoSQL databases and the contained data. This demonstrates the

rarity of such skills and highlights the leverage of data contained in the NoSQL databases for the value is worthless without technical assistance.

> *"There is a high-demand/low-available candidate pool for staff with [NoSQL] database skills/training."*
>
> *"Very difficult to get skilled folks. We have to come up with very creative approaches"*
>
> *"Finding the right folks"*

6. What is the future of big data – direction, growth, and relevance to your role, organization, and company?

   All three indicated the huge growth and demand for big data and the associated NoSQL databases. They each tied their organizational success to their success with these technologies. Each company has major programs, customers, and internal operations aimed at leveraging big data and its storage in NoSQL databases.

> *"NoSQL is critical to my future."*
>
> *"Big data is critically-important to my business and the future of our Corporation. We have been increasing our investments in this area, including formal skills development and recruiting, academic and commercial partnering, and development/integration into core products and capabilities."*

7. How do you see the availability and growth of big data skills? How critical is this to your success?

   Again, finding staff and proper skills was highlighted as extremely difficult. Each company is currently looking into methods to grow and attract these skills.

   > *"There is a critical shortage and "war for talent" in this high-demand market. This demand will only continue to increase."*

8. Describe your current methods for evaluating and integrating the content of a NoSQL database

   All three executives would be unable, by themselves, to produce any insights into the content and use of an unknown NoSQL database. They would all need to immediately involve a skilled NoSQL database technician. Given that there are many different NoSQL technologies, this could involve a search for technical staff. This leads to further time delays and may even lead to dismissing the database altogether.

   > *"I would contact someone who could help me".*

   > *"Hand it over to technical folks"*

9. Would faster integration of NoSQL databases aid your critical decisions? If so, how?

All agreed that this would be helpful for their various projects and programs, especially given the evolution in NoSQL technology and it proliferation in industry.

> *"Absolutely! Many of our current staff are primarily experienced with relational databases. Accelerating their adoption/transition to NoSQL would offer significant business benefits in this critical growth area for us."*

The findings of the interviews not only confirm the importance and growth of NoSQL technology but also demonstrate that it is held captive by technical employees. Employee skills are the critical component. The executives did not mention the need for more data or more computing resources – only the need to acquire skilled employees. The findings show that a business executive who makes critical business decisions relies on technical folks to communicate the value of a given NoSQL database. This creates two concerns: reliance on an effective business-technical translation with regard to integration of NoSQL databases and scarcity in technical skills to achieve that integration. These two concerns would act to limit access to this powerful technology in response to business opportunities. The limitation requires additional time and effort. Therefore, many databases may go unexamined due to time and skill constraints. This limits the information that is available for making a business decision or taking action.

Additionally, all three executives participated in the quantitative evaluation process described in the next section. Each was provided with an overview of the ontology visualization and was left to complete the task. All three completed the task with 100%

accuracy. Each executive correctly selected the databases, tables, and fields necessary to complete the task.

These interviews helped confirm the need and value of better business tools to exploit the contents of NoSQL databases.

4.3 Quantitative Approach

The quantitative evaluation consists of four major parts: a pre-test, an overview of the proposed method, semantic integration method task, and a post-test. The pre-test asks participants' about their demographics, their experience with NoSQL database technologies, and their knowledge in evaluating and integrating NoSQL databases through experience questions. It also asks what actions participants would take given an unknown NoSQL database. The method overview section provides a brief description of the integration method and presents a sample visualization to illustrate the key mnemonics. The description includes definitions of the concepts of a database, table, and records as well as technical connection information, such as technology type, ports, and login information. These concepts are required to perform an actual integration. The NoSQL review task is performed by viewing three different visualized ontologies that represent a NoSQL database. From the visualization, the participant determines the database's effectiveness towards a business goal and lists the specific data artifacts necessary to contribute to the goal. Finally, participants are asked to complete a post-test. The following two sub-sections introduce the last two parts of the evaluation in detail.

4.3.1. NoSQL Selection Task

Participants are asked to evaluate the usefulness of a given NoSQL database for the creation of a portfolio of stocks that beat the five-year average for S&P 500, NASDAQ, and the DOW. Figure 21 illustrates the process of the NoSQL Selection Task.

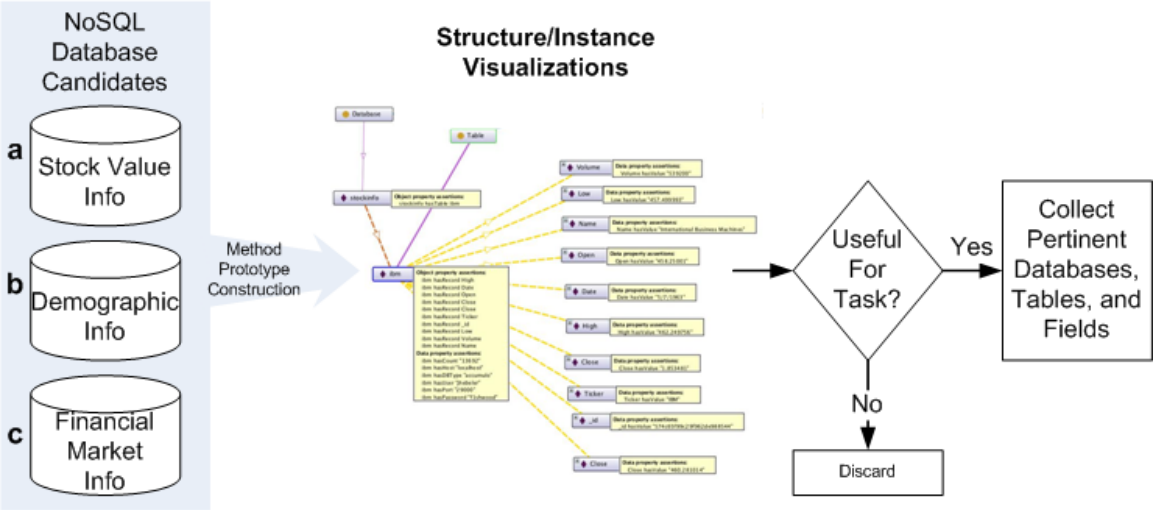

Figure 21: Selection Process of a Quantitative Approach

The implemented method generates an ontology from an actual NoSQL databases (shown on the left as candidate a, b, and c) and then presents an ontology visualization to the participants (shown in the middle). Participants are asked to assess the usefulness of each NoSQL database by reviewing its visualized ontology in performing the assigned task – building the stock portfolio. If the ontology is considered useful, the participants must list the specific database(s), table(s), and fields that are useful to the task. The options include the three visualized ontologies generated by the proposed method from three actual NoSQL databases. (See Figure 23, Figure 24, and Figure 25). Two of the three

NoSQL databases would be considered directly useful to the task, whereas one NoSQL database is not useful. For each of the three visuals, participants are asked to answer five questions, as shown in Table 4.

The user was provided a brief overview of the mnemonics and definitions, shown in Figure 22.

NoSQL integration offers to vastly enrich a decision or action due to combining multiple huge data sets. **Understanding the data structure contained in a given NoSQL technology is key to evaluating its value and enabling integration.** Yet, the flexibility of these technologies makes uncovering the structure very difficult in time and technical complexity.

This research method generates a visual into NoSQL data to reveal its structure quickly and without the need of technical skills. You will help evaluate the usefulness of this method and its visual approach.

A couple definitions will aid your task. NoSQL data resembles traditional structured data in presenting a hierarchical organization of the data. The data is organized in three levels. The highest level is a *database* itself similar to a named file cabinet. A database is a collection of named *tables* that act similar to a file folder within the file cabinet. Each table holds *records* similar to a document filed away in a file folder. Records, like documents, hold the actual data. This hierarchy allows grouping of the data so that it is easy to store and find. For example, NoSQL data could have an *employee* database that contains a *person* table that contains *person* records. Each *person* record holds the person's name, position, and so on. NoSQL data can have multiple databases containing multiple tables each holding multiple records containing the actual data. The blue boxes on the right highlight this storage hierarchy.

Below is a review of the type of visual you will examine with the method flow in the lower left corner. The orange highlights key areas.

NoSQL Data Visualization

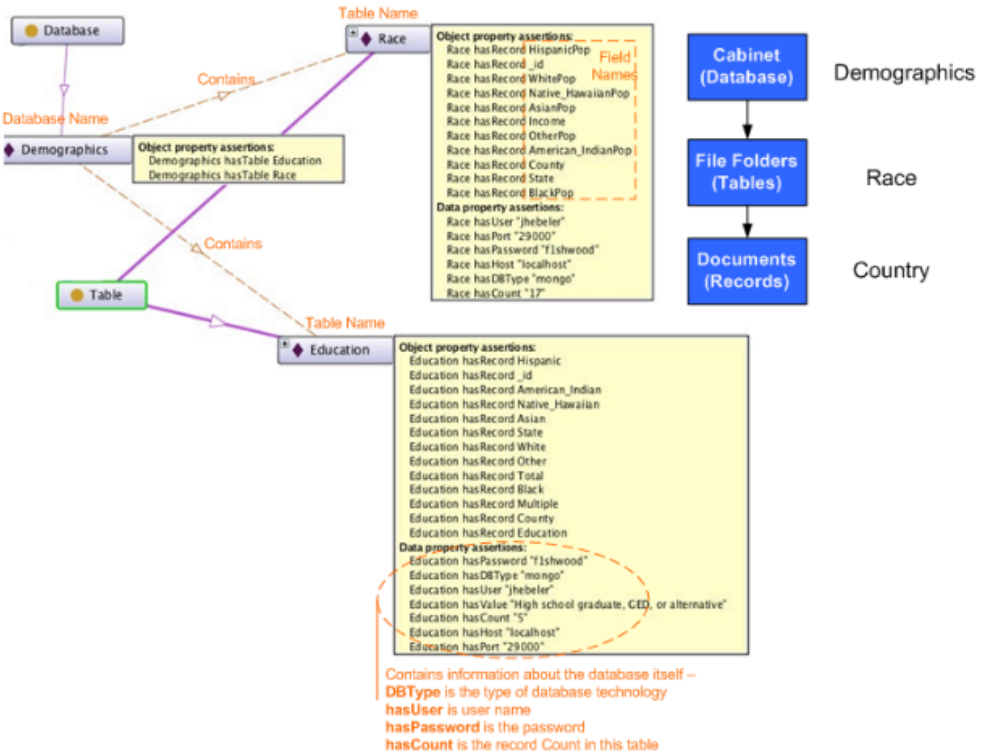

Figure 22: Introduction for Survey Participants

The following is a sample task description provided for the survey participants:

> *Determine, from the visual, if this NoSQL data is useful in aiding in the creation of a portfolio of stocks that beat the five-year average for the S&P 500, NASDAQ, and DOW.*

Table 4: Five Questions for NoSQL Selection Tasks

1. Would this data be useful toward the task?
2. If useful, list the names of the databases useful toward the task.
3. If useful, list the names of tables useful toward the task.
4. If useful, list the names of the fields useful toward the task.
5. List the NoSQL technology.

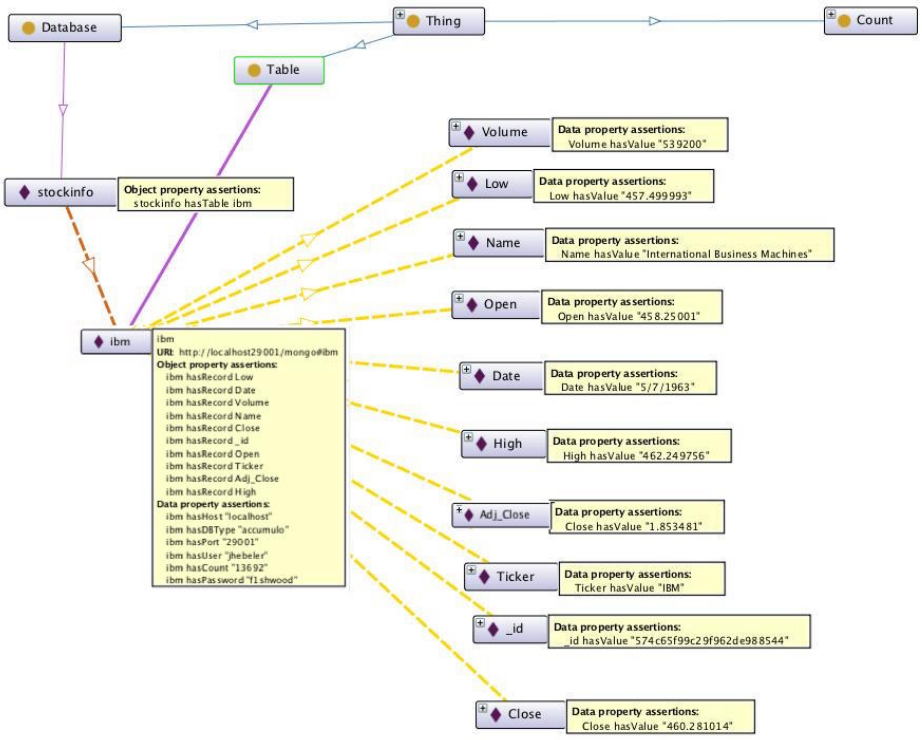

Figure 23: Visualized Ontology 1

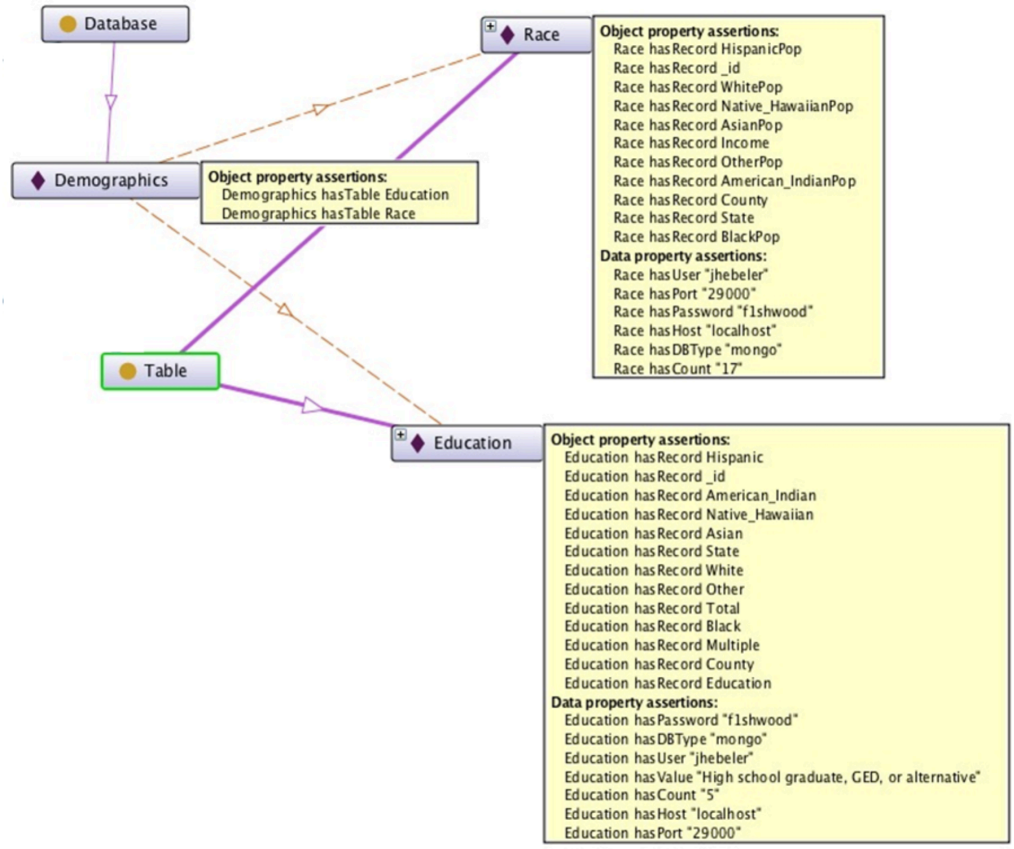

Figure 24: Visualized Ontology 2

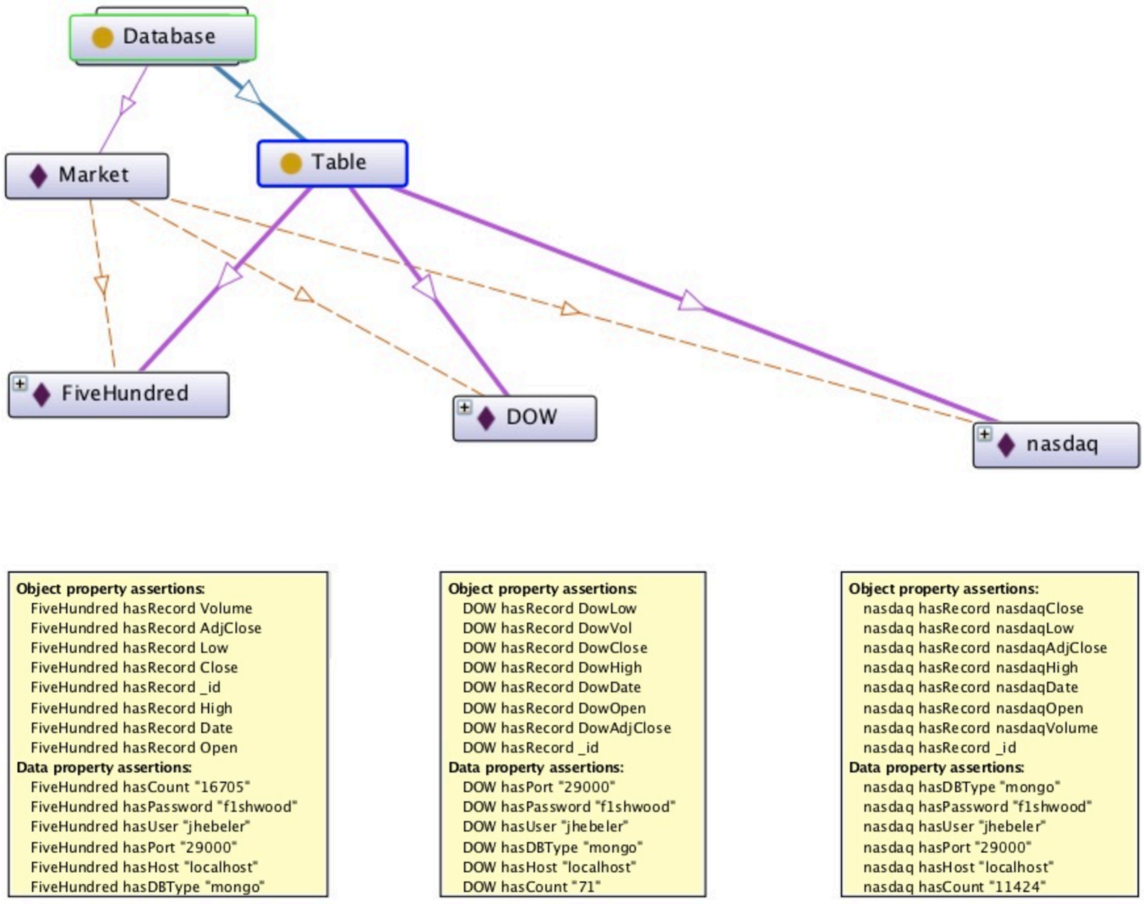

Figure 25: Visualized Ontology 3

4.3.2 Post-test

The post-test questions focused on the perceived usefulness, its perceived ease of use, the participant's intention to use the proposed method, and any suggestions for improvement. This study employed the proven research instruments from the technology adoption model (Davis, 1989), which has been adopted and validated from many studies. This dissertation measures each questionnaire item using a 5-point Likert scale. The questions compare the participant's current data integration approach with the semantic integration method for data integration from NoSQL databases:

Please provide your agreement with the following statements with regard to using the visuals for the tasks of evaluating/integrating data from NoSQL compared to the current approach you would use.

1. Accomplish the task faster
2. Improve the task performance
3. Improve the productivity of the task
4. Enhance effectiveness of the task
5. More useful in the task

Please provide your agreement with the following statements with regard to the ease of use of the visuals for NoSQL data evaluation/integration.

1. Easier to learn
2. Easier to manipulate
3. Clear/Better understandable interaction
4. More flexible to interact with
5. Easier to become skillful
6. Easier to use

4.3.3 Data Collection

Fifty-five people participated in this study. Participants were graduate students in Business Schools as well as the three interviewed executives. Table 5 shows the demographic information of participants. Age and gender offer a reasonable cross-

section with a skew towards younger participants. Additionally, the individuals are familiar with the value of data and databases in general but not technically skilled in NoSQL.

Table 5: Demographics of Participants

Gender	Male	33 (60%)
	Female	22 (40%)
Age	21-34	36 (65%)
	35-50	13 (24%)
	51+	6 (12%)
Years of Business Experience	0 Year	17 (32%)
	1-5 Years	19 (36%)
	6-10 Years	7 (13%)
	11-20 Years	8 (15%)
	21+ Years	2 (4%)

Table 6 shows the participants' experience with NoSQL technologies and their outlook on NoSQL technology. Less than 10% of the participants had experience from either a business or technical perspective with regard to NoSQL technologies. This is an important prerequisite to ensure limited exposure to the technical aspects of NoSQL technologies. Interestingly, the majority found it a critical technology where skills are scarce but available information is plentiful. This is typical of an emerging technology that evolves quickly and generates a strong demand for learning resources ranging from

on-line forums to published books. In addition to the executives, the participants were Master's degree candidates familiar with financial terminology but who have limited technical knowledge. Similar to the executive interviewees, they did not have hands-on experience with NoSQL databases and relevant NoSQL data integration.

Table 6: Participants Experience with NoSQL

Question	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Average
I have extensive business experience with NoSQL technologies	22 (43%)	19 (37%)	6 (12%)	3 (6%)	1 (2%)	1.86
I have extensive technical experience with NoSQL Technologies	23 (45%)	19 (37%)	4 (8%)	4 (8%)	1 (2%)	1.84
NoSQL data are critical to business success	2 (4%)	2 (4%)	17 (33%)	24 (47%)	6 (12%)	3.59
There is a scarcity of NoSQL Technical Skill	0 (0%)	1 (2%)	25 (49%)	17 (33%)	8 (16%)	3.63
There is a growing body of knowledge on NoSQL technologies such as books, articles, conferences, and training that offer useful information	0 (0%)	1(2%)	20 (40%)	25 (50%)	4 (8%)	3.64

Participants were asked an open-ended question about the methods they currently use to evaluate a NoSQL database. The answers varied but none of the participants were able to use a NoSQL database directly without receiving assistance from a technical source.

Some avoided it all together, reflecting the risks associated with using any Internet link. Below are some of their responses.

> *"Seek advice from someone who had a good understanding of NoSQL and research the topic myself"*
>
> *"I would have to first spend extensive time Googling and reviewing documentation on how to retrieve, view and manipulate the data before going any further."*
>
> *"Check with my IT area."*
>
> *"I'm not sure. I have no experience with NoSQL*."

In the method overview section, the participants were asked about their level of understanding after reading the overview. Among the participants, 50% indicated that they completely understood the overview, and 50% indicated that they somewhat understood. No participant indicated that they did not understand the proposed method or its visualized ontology.

4.3.1 Results

4.3.1.1 Results of NoSQL Selection Task

Table 7 shows the results of a NoSQL selection task. As demonstrated in the table, the vast majority correctly determined the usefulness of the database – a key value of the method. This alone could accelerate the inclusion of useful data in an integration effort because the business user would *not* depend on a (rare) technical person to make that initial selection. Additionally, clear majorities were able to determine key database

artifacts and associated technologies, which are required to actually perform a technical integration of the databases. Finally, there seems to be an improvement in task performance as the participants gain experience with the visualizations.

Table 7: A Summary of NoSQL Selection Results

	NoSQL Database 1	NoSQL Database 2	NoSQL Database 3
Correctly determined if the NoSQL database is useful for the task	36 (71%)	42 (82%)	46 (89%)
Correctly listed the database	29 (58%)	N/A	32 (64%)
Correctly listed the tables	25 (45%)	N/A	38 (69%)*
Correctly listed the fields	34 (62%)	N/A	35 (64%)
Correctly listed the NoSQL Technology	31 (56%)	31 (56%)	32 (64%)

Table 8 summarizes the number of correct answers each participant provided for one or more of the three databases. Virtually all participants evaluated at least one database correctly. Over 90% evaluated at least two correctly, and over half identified all three correctly.

Table 8: Percentage of Correct Selection by Participant

Participant who provided an answer	#(%)
Correctly answered at least one out of three databases	52 (96%)
Correctly answered at least two out of three databases	50 (91%)
Correctly answered all three databases	30 (55%)

Finally, the time to complete the survey indicates how quickly the participant could evaluate the database and extract useful database artifacts to answer the questions. No time limit was imposed on the participant. After eliminating incomplete surveys (3) and clear outliers where completion time exceeded 24 hours (2), the average time was 31 minutes. This demonstrates the efficiency for a non-technical user to evaluate a NoSQL database.

4.3.1.2 Results of Post-test

Table 9 shows the participant's overall assessment of the method's usefulness. The data shows that the overall results on perceived usefulness are positive. However, as noted earlier, most of the participants did not have the necessary knowledge or skill to evaluate or integrate NoSQL data.

Table 9: Participants' Feedback on Usefulness of the Semantic Integration Method

	Strongly Disagree	Disagree	Neither Agree Nor or Disagree	Agree	Strongly Agree	Total	Average
Accomplish the task faster	0 (0%)	2 (4%)	20 (38%)	20 (38%)	11 (20%)	53	3.75
Improve task performance	0 (0%)	2 (4%)	15 (29%)	28 (54%)	7 (13%)	52	3.77
Improve task productivity	0 (0%)	1 (2%)	18 (35%)	25 (48%)	8 (15%)	52	3.77
Enhance task effectiveness	0 (0%)	4 (8%)	17 (33%)	23 (44%)	8 (15%)	52	3.67
More useful in task	0 (0%)	3 (6%)	19 (37%)	24 (46%)	6 (12%)	52	3.63

Table 10 shows the distribution percentage of the participants' overall assessment of the method's ease-of-use. The results show that the vast majority found the visualized ontology to be easy to use for the task.

Table 10: The Ease of Use of the Visualized Ontology

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree	Total	Average
Easy to learn	1 (2%)	3 (6%)	13 (25%)	26 (49%)	10 (19%)	53	3.75
Clear/Better Understandable interaction	1 (2%)	2 (4%)	16 (31%)	24 (46%)	9 (17%)	52	3.73
More flexible to interact with	1 (2%)	1 (2%)	22 (42%)	22 (42%)	6 (12%)	52	3.60
Easy to become skillful	1 (2%)	1 (2%)	20 (38%)	25 (48%)	5 (10%)	52	3.62
Easy to use	1 (2%)	0 (0%)	17 (33%)	27 (52%)	7 (13%)	52	3.75

Table 11 shows the distribution percentage of the intention to use the method. The results show strong intentions of the participants using this proposed tool/method in this study.

Table 11: Intention to use the method

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Total	Average
I would use these visuals for uncovering the structure and contents of an unknown NoSQL data.	1 (2%)	0 (0%)	15 (28%)	24 (45%)	13 (25%)	53	3.91
I would use these visuals for aiding the integration of an unknown NoSQL data	0 (0%)	1 (2%)	13 (25%)	24 (47%)	13 (25%)	51	3.96

The final section contains several open-ended questions regarding the usefulness, potential, and suggestions for improvement. Some of the participants' responses are quoted below.

Open-ended Question 1: Were the visualized ontology easy to use?

> *"The visuals were good for helping me to understand the makeup of a nosql database and how records are stored."*
>
> *"Hierarchy and labeling were easy to follow."*
>
> *"The use of color and the legend for table versus database if my eye gets lost."*
>
> *"Clear representation of columns and tables, databases"*

Overall, the visualization was sufficient to communicate the key elements of the database because a significant majority completed the tasks correctly.

Open-ended Question 2: What overall potential do you foresee for the visuals in evaluating and integrating NoSQL data?

> *"More visual access can determine more understanding of the database. People new to NoSQL would be more comfortable using the visuals."*
>
> *"easier to grasp the concepts"*
>
> *"Aiding non technical users to come up to speed in understanding the system and enable self service"*
>
> *"This has great benefits"*

Clearly, the visuals work for the intended audience of non-technical, business users. This typical simplicity of NoSQL structures lends itself to a basic visual.

Open-ended Question 3: Would you use these visuals to determine the value of a NoSQL database and to integrate its data?

Out of the participants, 81% responded positively.

> *"yes, I believe the visuals used here would be effective for helping someone understand an actual nosql database that they are working with and help them to integrate the data because it shows where to insert each record in the fields section."*
>
> *"I would. It seems simpler than the traditional style of writing NoSQL."*
>
> *"Definitely but with a caution- One has to be careful to update the visuals and also make sure that the visual correctly represent the system as an error will have serious impacts"* One note on this comment The visualization is produced

directly from the database. Therefore, it can easily maintain the correct representation.

Open-ended Question 4: What improvements would you recommend for the visuals?

Below are some of the suggested improvements. Most participants recommended clearer text and better use of colors and shapes. Advancements to the visuals would include many of these suggestions, especially the idea of making it interactive. The interactivity would allow the user to explore the key areas for them and it would effectively manage larger scale database structures.

> *"Database and tables can be notified with a different symbol. Currently both are shaped diamond"*
>
> *"Instead of having separate boxes for database and table, perhaps it would be better to color code the boxes with the database/table names - for example, on the previous page, yellow for market and green for the FiveHundred box and the other 2 tables' boxes."*
>
> *"Use more color and make them interactive."*

4.4 Discussion

NoSQL databases contribute to the foundation of Big Data opportunities, as they have the ability to scale and hold massive data. Databases become useful when properly placed in a business context aimed at a business opportunity. Unfortunately, the nature of NoSQL databases makes this challenging for the non-technical business user and requires a skilled technical person to assist in integration of NoSQL databases. As noted in both the

interviews and surveys, a non-technical person must continually seek out a skilled technician to uncover the contents of a NoSQL database, and given the scarcity of technically skilled persons this is a critical business constraint. This can limit the use of NoSQL databases and its data to address a business opportunity due to the scarcity of skilled labor, shrinking time frames of an unfolding business opportunity, and the translation necessary between a business user and a technician.

The semantic integration method demonstrates through this evaluation that it can aid the process of evaluating and integrating NoSQL databases. Non-technical business users can quickly evaluate a NoSQL candidate database and even select the key database artifacts, such as table names, to aid the integration effort. Inappropriate databases are quickly discarded allowing more time to identify useful databases.

The task of evaluating NoSQL databases itself, beyond the qualitative questions, demonstrated its usefulness since most participants successfully completed the task. More than half of the participants successfully evaluated all three databases and over 90% evaluated two out of three correctly. This came about with minimum training and without the ability to ask questions or for clarification. Adding the opportunity for additional dialogue would likely improve the success rate. Without the semantic integration method, the users, as they indicated, must reach out to additional resources. Simply put, they would not, by themselves, be able to complete the task without the aid of the semantic integration method. Otherwise, only a skilled technical professional could have completed these tasks. Many of the participants were not familiar with

NoSQL or even databases themselves, yet were still able to garner the substance and key elements of a NoSQL database. The evaluations reach the following conclusions.

1. The semantic integration method enables non-technical users to integrate a NoSQL database for a given business purpose through an examination of the produced ontology visualizations.
2. The semantic integration method enables non-technical users to select specific key database artifacts, such as tables and fields required for integration. Over half correctly identified all of the necessary elements. This allows the method to directly integrate the identified data. This iterative method allows a non-technical user to build an integrated NoSQL store.
3. The semantic integration method enables rapid integration of NoSQL sources despite the lack of documentation and knowledge of changes allowing the ability to evaluate more NoSQL databases. The average time to fulfill the task of evaluating three NoSQL databases and answering survey questions would, in many cases, allow for ample time to include the data in an unfolding business opportunity, as well as affording the time to examine additional databases to improve overall understanding of the business opportunity.

The results assume a relative, semantically rich structure with few relationships in the NoSQL databases. However, these assumptions are reasonable given the empirical study and the principals behind most NoSQL databases. The method does allow for dynamic and heterogeneous structures scattered throughout the database, decreasing the need for documentation maintenance.

5 Chapter 5: Discussion and Conclusion

Data integration remains a tough challenge to aid in the formation of a business decision or action. The rapid increase in data production combined with associated new technologies to capture and query this data further challenges data integration. Without advancements in data integration methods, much data potential is lost.

NoSQL technologies offer the ability to collect and interact with enormous amounts of data. New integration methods offer to expand the value of this data. By doing so, it not only overcomes data overload but makes that data truly capable of aiding business decision makers.

Traditional data integration depends on the translation of a business opportunity into protracted, complex technical steps. This incurs the many limitations formally outlined and discussed above, which increase both the cost and time required for the data integration. A new model, the semantic integration model, joins the business user who directly interacts with the data and automation tools that integrate the data. This method offers the capabilities to better leverage massive data across many large databases in a cost-effective manner.

The semantic integration model addresses this massive data integration opportunity through three key features:

1. The incorporation of artifact naming into NoSQL databases to create useful semantics.

2. The incorporation of sequential statistical methods to capture and normalize distributed structural information not readily available.
3. The incorporation of the Semantic Web ontologies such as RDF and OWL, to capture the structural information for ontology visualization for the business user and the technical information required for automating the actual data integration.

The speed and coverage of these features allows the business user to review and integrate many databases, as well as to iterate through those databases based on various dynamics and learning that may occur during the integration process. This results in a more informed, comprehensive view of a business opportunity leading to better business decisions and actions.

5.1 Contributions

The main contribution addressed by the semantic integration model is leveraging and integrating big data contained in the various NoSQL databases across various NoSQL technologies. Big data is being continuously absorbed, stored, and indexed in the rapidly evolving NoSQL databases. These new technologies serve scale first, allowing massive data storage, rapid ingest of data, and rapid query replies while the data is protected from change with dynamic structures. The rapid evolution of these technologies thwarts standardization for each NoSQL technology that offers only proprietary interface methods. Specialized technology paths are required to access each type of NoSQL technology requiring different technical skills. A method that meets these challenges

expands the business potential contained in the various databases and their possible integration.

The fundamental challenge is to allow a business user to leverage multiple big NoSQL databases to quickly reach a business decision or take a business action.

This dissertation addresses the above challenge by answering the following research questions, which led to the development of the semantic integration model and related research contributions.

1. How can *semantic transformation* reveal the semantics of technically complex and evolving data storage components found in various NoSQL databases and associated technologies?

   The proposed semantic transformation uses a statistical approach to quickly converge on the structural contents regardless of the structural information placement in the large big data NoSQL database. NoSQL databases repeat structural information throughout the store and the statistical approach ensures the capture of the structural information that would otherwise require an exhaustive search through all records. As each structure is uncovered, it is compared to the current set of extracted structures. Duplicated structures are not incorporated into the set. This helps normalize the structural information, and allows for rapid analysis of the structure regardless of size.

2. How can *semantic representation* adequately provide a useful capture of the underlying NoSQL database for technical integration?

The Semantic Web offers a common format that is expressive enough to represent the structure and technical details of NoSQL databases. The expressiveness of the Semantic Web, as implemented in this study, goes beyond the basic expressiveness of NoSQL offerings by adding the ability of logic inference to further enrich the structural and technical information. The semantic representation also supports formal validation.

3. How can *semantic unification* integrate various data sources revealed through semantic transformation?

The semantic integration model takes advantage of the logic with the Semantic Web technologies and thus provides simple integration via aggregation. Due to the uniqueness of the semantic data artifacts, there are no technical conflicts. Conceptual normalization is achieved through logical equivalence statements that do not affect the actual data. This allows for corrections to and recovery of the original data. Furthermore, by using logic, similar terms and concepts can also be merged.

4. How can *semantic alignment* focus the unified semantics on a specific domain of interest?

The visualization of the ontology and its relationships provides a view into the data that enables the user to determine its alignment towards the user's requirements. This includes both visualizations: the candidate database review and the integrated database. The former determines the useful addition of a potential database candidate.

The latter determines whether the integration effort is complete for a given business opportunity.

5. How can *semantic adaptability* using *convergence-directed integration* produce a more flexible and automated approach to integration?

   The efficiency of the method allows new databases or updated ones to be quickly revaluated for a given business purpose. Accordingly, it supports rapid adaptation to changing business directions. This also allows the business user to learn from each database candidate's integration and then redirect the effort within the integration.

Simply put, the method allows a non-technical, business user to evaluate and integrate NoSQL databases. Specifically, this research makes the following key contributions.

1. Improved integration of relevant data based on the semantics and relationships contained within the NoSQL database.
2. Incorporation of Semantic Web standards and tools to aid semantic data integration. This allows the leverage of tools, ontologies, and data available via the Semantic Web.
3. Improved integration of new, emerging databases that are characterized by scale and structural dynamics, including uncovering the structural information from a dynamic NoSQL store.
4. Applicability of convergence methods to incrementally and iteratively evolve data integration to the user's requirements.

Since the data is evaluated and integrated faster via user interactions rather than waiting for the full technical integration – the speed of data, similar to the speed of money, increases its overall value. The data can be used to address more business situations and therefore is more valuable.

The contributions can be further decomposed based upon individual steps of the integration model, such as the following.

- Transformation
    - Improves Data Access: The ability to quickly determine the NoSQL technology type and the contained structures aids the ability to actually use the NoSQL database. This doesn't depend on documentation and is completely up-to-date which is especially important since structural additions and modifications can be added at any time.
    - Adapts to new NoSQL technologies and their rapid evolution: The method separates and isolates the proprietary code thus allowing a minimal change when a new NoSQL technology or update occurs. The method quickly adapts to NoSQL changes.
    - Handles Big Data Scale: The method uses statistical sampling to extract the structural information since most NoSQL stores do not offer a direct way of obtaining the structural information. This allows the method to quickly converge on the structure without the need to exhaustively examine every record in the NoSQL database, which could be prohibitive in extremely large NoSQL stores.

- Representation
    - The method represents key information regarding the NoSQL store via a Semantic Web Ontology. This is a standard method of expressing an ontology and offers many extensions and tools. A Semantic Web ontology captures the underlying structure, structure relationships, and technical metadata to enable automatic data integration.
    - The representation eases conflicts and duplications through the use of the OWL constructs that can perform logic to filter errors, populate missing data, eliminate duplications, and merge equivalent database elements.
- Unification
    - Multiple Views: The flexibility of an ontology allows for multiple perspectives to examine the data. This is demonstrated by allowing the selection of various perspectives such as domain relevant structures to technical structures that include databases and tables.
    - Enhanced ability to deal with Dirty Data: Integrating data into an ontology is as simple as adding in the statements from the candidate ontology. Once integrated in the complete ontology, logic and rules can be applied to filter out the data. This filtering can be done permanently (physical removal) or virtually where the logic excludes or merges certain statements. This allows conflicts to be studied from one perspective while allowing another perspective to not contain the suspect data.
    - Complex Relationships Across the Data: The Semantic Web via OWL enables complex relationships. The initial implementation took advantage

of only a few, such as the equivalence statements. Even if the NoSQL structures contain richer, more complex relationships, the OWL ontology would still be able to correctly represent them.

- Alignment
    - Ability to Evaluate each Data Candidate within the integration process: The semantic integration method enables early review so as to avoid integrating unnecessary information, while also quickly advancing a useful candidate, and learning what is available. The latter may help evolve the business opportunity itself.
    - Enhanced Support for Domain Extraction: Due to the ability of queries and the flexibility of the underlying ontology, multiple domains of interest either technical or business can be correctly reflected within a given integrated database. This not only takes into account multiple perspectives but also allows the user to change their perspective.
    - Extended Integration Methods: The semantic integration method makes integration straightforward. All of the ontology statements aggregate together then logic separates the domains and unifies equivalences resulting in a non-destructive normalization for the ontology that still holds the required information to perform a query on the original NoSQL database.
    - Improved Reasoning: The ontology offers a rich set of reasoning and rules to enable filtering, error correction, equivalence, and so on.

- Adaptability:
    - Open World Model: The method employs the ontology to reflect an open world model allowing rapid integration without conflicts at any point since no assumptions are built into a given integration. This is especially important to NoSQL stores that maintain dynamic structures with asynchronous updates.
    - Simplified Integration: The method allows a common integration process that avoids N^2 connections between databases, simplifying the integration.
    - Adaptive Integration Solutions: The method supports iterative, incremental integration of data sources from multiple technologies. This feature improves adaptability and ease of use of the integration method.
    - Minimum Duplication: The method relies on retrieving the actual data from the native source when requested. There is no need to copy the data and accordingly, no need to maintain data synchronization.

These contributions ease data integration across valuable, yet evolving NoSQL databases. The method enables fast integration with minimum technical assistance thus allowing more data to support a given business opportunity.

5.2 Method Alignment with Trends

The Semantic Integration Method aligns with trends in business and technology.

- Plunging cost of Infrastructure: The method scales to handle the larger storage capabilities.

- New Data Storage Options: The method isolates the direct interactions with a native NoSQL storage and thus, minimizes the efforts required to evaluate and integrate from a new or updated NoSQL technology.
- Growing participation in Producing Data: Again, the method scales to handle new sources and with its quick review, allows a user to peruse many more data sources than currently possible. Thus, the method takes advantage of growing participation.
- Development of the Semantic Web: The advances in the Semantic Web via new reasoners, display options, and the like would quickly advance this method since its foundation is in the Semantic Web.
- Real-Time, Dynamic Data and Associated Structures: Since the method does not copy data but merely provides a path to the actual data, a business user gets the latest information. Additionally, as streaming data becomes more useful, the data source integration could enable a similar interface like that used by the NoSQL stores, thereby allowing streaming data to be another data candidate. They maintain similar challenges in that the structure is dynamic and contained within each record.
- Increasing Dirtiness of Data: As data sources grow and a business taps into external databases outside of their control, the dirtiness of the data will likely increase. The ontology's foundation allows many ways to correct this with minimal impact to the actual operations. It also follows that the dirtiness is subjective to the given domain and query, allowing multiple perspectives on the dirtiness determination.

The semantic integration method is well aligned with future trends and is able to quickly take advantage of these advancements.

5.3 Limitations

The limitations of the method lie in seven areas: disruption to integration workflows, complex structures, non-semantic structures, maintaining up-to-date interfaces to each NoSQL technology, awareness of new NoSQL technologies, overall data cleanup and preprocessing, and obtaining and evolving necessary technical skills.

The introduction of a new workflow clearly disrupts the current procedure. This change incurs cost and risks. This is especially true with business critical actions and decisions. A major change in the ways the business uses data is critical. It requires an incremental and iterative approach to integrating the method into current business practices. The method can be introduced through the selection of a minor business opportunity and its associated data sources. Incorporating the method in this way proves its value and allows the various users and supporters time to acclimate themselves. As confidence and use practices grow, the method can expand to other opportunities. Current operations can continue in parallel and may continue indefinitely for certain situations. The method is not mutually exclusive and allows other methods to operate simultaneously.

Complex structures consist of entities that maintain relationships with many other fields in *one to many*, *many to one*, and *many to many* relationships. The method would indeed be challenged by such structures. However, antidotal reviews have found that NoSQL databases do not maintain such relationships and additionally the technology and its

priorities do not encourage such relationships. This is a limitation and future research is required to see if those conclusions are indeed substantiated across expanded use of NoSQL databases.

Non-semantic structure names consist of human-meaningless database names, table names, and field names. The NoSQL technologies allow this but again, this goes against the typical use of these database artifacts. The method also allows semantics to be derived by providing field values to allow the user or enrichment ontologies to aid in building the semantics. For example, a table name of simply '*ss*' offers little semantics. However, if the user is also provided an instance example such as '134-45-6789', they may infer that *ss* represents social security numbers. Of course this is not possible in all cases. There is little reason for building a data model with meaningless semantic artifacts unless the intention is to purposely disguise the data. This is unlikely since the data is intended to be used and shared.

Maintaining the method requires updating the interfaces as they evolve. This requires technical skills and alertness across the NoSQL technologies. Additionally, depending on the changes, the method may need to obtain multiple interfaces to the same technology for each major version. All of this is doable but requires investment and resources. This extends to recognizing and incorporating new NoSQL technologies.

Data cleanup and preprocessing tasks can grow large and time consuming. The method can incorporate additional rules, logic from the ontologies, and additional enrichment ontologies. This, again, requires skills and investment. Using ontologies to clean up and

enrich data should improve efficiency so the cleanup would work across all the NoSQL technologies. There is no need to do a separate cleanup for each NoSQL technology.

Finally, the method does require technical skills to maintain and evolve. These skills are currently hard to obtain. Fortunately, the method minimizes those needs over current practices.

5.4 Future Research Extensions

The method provides research paths to several useful extensions including field study and recommendations; improved, interactive visualizations; enrichment of ontologies and associated logic; data preprocessing logic; adding alerting on databases changes; and extensions to structured data outside of NoSQL.

An empirical field study requires access to real world business situations and measuring the performance of given use cases, varying the size of the databases, the structures, the technologies, and so on. The findings of the study would provide recommendations for the actual implementation architecture. This requires hardening of the prototype to make it easy to deploy and monitor, as well as extensive testing. Field experiments will highlight areas for improvement in visualization, capture of the various NoSQL structures, and data integration performance. Methodological limitations of such an approach must also be considered.

The interactive ontology visualizations can be extended in several ways. For instance, it can support more complex structures. Additionally, alternative visual representation

beyond a graph, tree, or other structure may be explored for certain structure types. Some of the NoSQL technologies offer their own visualizations of the data. These visualizations may suggest effective techniques that could be incorporated into the ontology visualization used in the semantic integration method. However, these visualizations would not serve as a substitute for the ontology visualization because each is unique to its particular NoSQL technology. These different visualizations would likely be confusing to the user and would unnecessarily make them aware of the particular NoSQL technology. Additionally, this is orthogonal to the goal of one common view of the data regardless of the underlying NoSQL technology. One common view must be generated to present the integrated database visualization across multiple NoSQL technologies.

Enrichment of ontologies and associated logic would allow for expressing rich semantics. The logic could look for word and phrase similarities across the database terms and, when appropriate, make them equivalent. Additionally, the ontologies could actually enrich the integrated data itself by adding supporting information. LinkedData (LinkedData, 2017) offers many useful ontologies that describe a myriad of terms and relationships. These could be reviewed and integrated along with the NoSQL databases. Additionally, reviewing ontology engineering with regard to various domains would align the ontology with larger research efforts aimed at improving ontology development. The model's development of the ontology would be further enriched by following the state-of-the-art in ontology advancements.

Relational databases also hold rich data. However, the complex relationships that bind one table to another bring challenges of uncovering their semantics or meaning and properly exposing this to a non-technical user. Overcoming these challenges would aid business evaluation of integrated relational databases since many of the commercially available database visualizations do not simplify the extensive relationships. Future efforts in the advancement of the semantic integration model should explore the inclusion of relational databases and work through methods to simplify the complex relationships and extract the business semantics behind those relationships. This would allow the integration of large amounts of existing, valuable data.

Currently, the semantic integration model is a *pull* system where the user requests a structure analysis for possible integration. The method could advance to a *push* system where the structural analysis runs automatically and compares results. It could then provide an alert when a change to the structure is made. This would employ the existing structural comparator in the normalization of a candidate. Identification of a new structure would also initiate an alert. The automatic updates could provide data to understand changes in the structure over time, providing views to indicate the volatility of the data and its timeline.

Finally, the method could extend to integrate structured data beyond NoSQL databases such as streaming data, expressed in formats like XML or JSON. As with NoSQL data sources, the method can sample the stream to determine the structure, then convert to an ontology that maintains the interface information, and provide a visualization to the user.

6 Bibliography

Abadi, D. (2009). Data Management in the Cloud: Limiations and Opportunties. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* .

Adomavicius, G., Bockstedt, Jesse C. ,Gupta, A., Kauffman, R.J., Making Sense Of Technology Trends In The Information Technology Landscape:A Design Science Approach, MIS Quarterly, Vol. 32 No. 4, December 2008, pp. 779-809.

Amazon. (2011, 11 8). *Amazon Public Data Sets*. Retrieved 11 8, 2011, from http://aws.amazon.com/datasets/Encyclopedic?browse=1

Amazon. (2012, 1). *Public Data Sets on AWS*. Retrieved 1 22, 2012, from Amazon: http://aws.amazon.com/publicdatasets/

Apache. (2011). *Apache Jena*. Retrieved 1 21, 2012, from apache: http://incubator.apache.org/jena/

Apple. (2012). *iCloud*. Retrieved 1 7, 2012, from apple.com: http://www.apple.com/icloud/

Benbasat, I., Goldstein, D. K., and Mead, M. 1987. "The Case Research Strategy in Studies of Information Systems," MIS Quarterly (11:3), September, pp. 369-386.

Bennet, T., & Bayrak, C. (2011). Bridging the Data Integration Gap: From Theory to Implemenation. *ACM SIGSOFT Engineering Notes , 36* (3).

Bennett, T., & Bayrak, C. (2011). Bridging The Data Integration Gap: From Theory to Implementation. *ACM SIGSOFT Software Engineering Notes , 36* (3), 8.

Berners-Lee, T., Hendler, J., & Lisssila, O. (2001, 5). The Semantic Web. *Scientific America* , 6.

Bizer, C. (2010, 11 29). *The D2RQ Platform - Treating Non-RDF Databases as Virtual RDF Graphs*. Retrieved 11 16, 2011, from wiwiss: http://www4.wiwiss.fu-berlin.de/bizer/d2rq

Bizer, C., & al, e. (2011, 9 19). *State of the LOD Cloud*. Retrieved 10 16, 2011, from wiwiss: http://www4.wiwiss.fu-berlin.de/lodcloud/state/

Blunt, K. (2011, 1). 10 Marketing Trends to Watch in 2011. *Expert Opinion* .

Buccella, A., Cechich, A., & Brisaboa, N. (2005). Ontology-Based Data Integration Methods: A Framework for Comparison. *Revista Colombiana de Computación.*

Clark, K., Feigenbaum, L., & Torres, E. (2008, 1 15). *SPARQL Protocol for RDF*. Retrieved 1 21, 2012, from w3: http://www.w3.org/TR/rdf-sparql-protocol/

Comito, C., Patarin, S., & Talia, D. (2006). A Semantic Overlay Network for P2P Schema-Based Data Integration. *Proceedings for the 11th IEEE Symposium on Computers and Communications (ISCC'06)*. IEEE.

Date, C. J. (2005). *Database in Depth.* Oreilly.

Davis, F. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly , 13* (3), 319-340.

DBpedia. (2011, 10 23). *DBpedia*. Retrieved 10 23, 2011, from DBPedia: http://dbpedia.neofonie.de/browse/rdf-type:Person/rdf-type:Athlete/

Doan, A., & Halevy, A. (2005). Semantic Integration Research in the Database Community. *AI Magazine , 26* (1).

Euzenat, J. (2008, 2 29). *INRIA & LIG, February 29, 2008 Alignment API.* Retrieved 10 16, 2011, from alignapi: http://alignapi.gforge.inria.fr/

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & Trojahn, C. (2011). Ontology Alignment Evaluation Initiative: Six Years of Experience. *Journal of Data Semantics , 6720*, 158-192.

Fan, H., & Gui, H. (2007). Study on Heterogeneous Data Integration Issues. *IEEE* .

Fellbaum, C. (2011, 6 21). *WordNet - A lexial database for English*. (P. University, Producer) Retrieved 1 4, 2012, from WordNet: http://wordnet.princeton.edu/

Foaf. (2010). *foaf project*. Retrieved 1 21, 2012, from The Friend of the a Friend Project: http://www.foaf-project.org/

Giordano, A. D. (2010). *Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture.* IBM Press.

Gold-Bernstein, B., & Ruth, W. (2004). The Business Imperative for Enterprise Integration. In W. R. Beth Gold-Bernstein, *Enterprise Integration: The Essential Guide to Integration Solutions* (p. 432). Boston: Addison-Wesley Professional.

Gold-Bernstein, P., & Hass, L. (2008). Information Integration in the Enterprise. *Communication ACM* , 72-79.

Goodhue, D., Wybo, M., & Kirsch, L. (1992, 9). The Impact of Data Integration on the Costs and Benefits of Information Systems. *MIS Quarterly* , 293-311.

Government, U. (2010, 11 11). *Data.gov Semantic*. Retrieved 1 21, 2011, from data.gov: http://www.data.gov/semantic/

Government, U. S. (2011, 11 4). *Data.gov*. Retrieved 11 4, 2011, from Data.gov: http://www.data.gov/metric

Granitzer, M., Sabol, V., Onn, K., Lukose, D., & Tochtemann, K. (2010). Ontology Alignment—A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Future Internet , 2* (3), 238-258.

Granitzer, M., Sabol, V., Weng Onn, K., Lukose, D., & Tochtermann, K. (2010). Ontology Alignment - A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Future Internet , 2*, 238-258.

Grossman, R., & Gu, Y. (2009). On the Varieties of Clouds for Data Intensive Computing. *Bulletin of the IEEE Computer Society Technical Committe on Data Engineering* , 7.

Harris, S., & Seaborne, A. (2012, 1 5). *SPARQL 1.1 Query Language*. Retrieved 1 21, 2012, from w3: http://www.w3.org/TR/sparql11-query/

Heath, T. (n.d.). *Linked Data - Connect Distibuted Data Across the Web* . Retrieved 11 16, 2011, from linkeddata: http://linkeddata.org

Hebeler, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2009). *Semantic Web Programming.* Wiley Publishing.

Herman, I. (2011). *Publications of the W3C Semantic Web Activity*. Retrieved 1 21, 2012, from w3: http://www.w3.org/2001/sw/Specs

Hevner, A., & al, e. (2004, 3). Design Science in Information System Research. *MIS Quarterly* .

Hickey, A. R. (2010, 8 19). *SMB Cloud Spending To Approach $100 Billion By 2014*. Retrieved 1 7, 2012, from crn.in: http://www.crn.in/Software-019Aug010-SMB-Cloud-Spending-To-Approach-100-Billion-By-2014.aspx

Horrocks, I., Patel-Schneider, P., & Boley, H. (2004, 5 21). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Retrieved 1 21, 2012, from w3: http://www.w3.org/Submission/SWRL/

Hurst, N. (2010, 3 15). *Visual Guide to NoSQL Systems*. Retrieved 1 22, 2012, from nahurst: http://blog.nahurst.com/visual-guide-to-nosql-systems

Husain, M., McGlothlin, J., Khan, L., & Thuraisingham, B. Scalable Complex Query Processing Over Large Semantic Web Data Using Cloud. *2011 IEEE 4th International Conference on Cloud Computing* (pp. 187-194). IEEE.

Idrissi, Y., & Vachon, J. (2009). An adaptive multi-strategy approach for semantic mapping. *C3S2E '09 Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering* (pp. 7-15). New York: ACM.

Internet Stats. (2011, 9 20). Retrieved 9 20, 2011, from Internet Stats: http://www.internetworldstats.com/stats.htm

Jena. (2010, 3 28). *Jena 2 Inference Support*. Retrieved 1 21, 2012, from jena.sourceforge: http://jena.sourceforge.net/inference/

Kalfoglou, Y., & Schorlemmer, M. (2003, 1). Ontology Mapping: the State of the Art. *Knowledge Engineering Review* , 1-31.

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology Mapping: The State of the Art. *The Knowledge Engineering Review , 18* (1), 43.

Kalin, M. (2009). *Java Web Services: Up and Running.* O'Reilly Media Inc.

Kassim, J., & Rahmany, M. (2009). Introduction to Semantic Search Engine. *2009 International Conference on Electrical Engineering and Informatics* (pp. 380-386). Selangor: IEEE.

Kaza, S., & Chen, H. (2008). Evaluating ontology mapping techniques: An experiment in public safety information sharing. *Decision Support Systems , 45* (4), 714-728.

Kifer, M., & Boley, H. (2010). *Overview of Rule Interchange Format*. Retrieved 1 21, 2012, from W3C: http://www.w3.org/2005/rules/wiki/Overview

Kim, W., Choi, B.-J., Hong, E.-K., Kim, S.-K., & Lee, D. (2003). A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery , 7*, 81-99.

Kirilov, K. (2011, 4 26). *Cloud Computing Market Will Top $241 Billion in 2020.* Retrieved 10 16, 2011, from http://www.cloudtweaks.com/2011/04/cloud-computing-market-will-top-241-billion-in-2020/

Kitchenham, B., Linkman, S., & Law, D. (2002). DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal , 8* (3), 120-126.

Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. *The Knowledge Engineering Review , 18* (1), 53-62.

Knublauch, H., Idehen, K., & Hendler, J. (2011, 2 22). *SPARQL Inferencing Notation (SPIN)*. Retrieved 1 21, 2012, from w3: http://www.w3.org/Submission/2011/02/

Laboratory for Applied Ontology. (n.d.). Retrieved 1 4, 2012, from DOLCE: http://www.loa.istc.cnr.it/DOLCE.html

Langegger, A., Wolfram, W., & Martin, B. (2008). A Semantic Web middleware for virtual data integration on the web. *ESWC'08 Proceedings of the 5th European Semantic Web conference on The Semantic Web: research and applications* (p. 15). Berlin: Springer-Verlag.

Lenzerini, M. (2002). Data Integration: A Theoritical Perspective. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 233-246). New York: ACM.

Liu, F., Lou, X., & Liang, G. (2009). Semantic Cloud based on SLN and ALN. *2009 Fifth International Conference on Semantics, Knowledge, and Grid* (pp. 314-317). IEEE.

Lucas, A. (2010). Corporate Data Quality Management. *2010 5th Information Systems and Technologies (CISTI)*. Iberia.

MacManus, R. (2010, 7 2). Facebook & the Semantic Web. *ReadWriteWeb* .

Maedche, A., & Staab, S. (2002). Measuring Similarity between Ontologies. *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW*.

Moujane, A., Chiadmi, D., Benhlima, L., & Wadjinny, F. (2009). A study in the P2P data integration process. *Computer Systems and Applications, 2009. AICCSA 2009* (pp. 57-58). IEEE/ACS .

Nakano, C. (2011, 10 18). *New Twitter Statistics Reveal 100M Monthly Active Users & 250M Daily Tweets #w2s*. Retrieved 1 7, 2012, from cmswire.com:

http://www.cmswire.com/cms/social-business/new-twitter-statistics-reveal-100m-monthly-active-users-250m-daily-tweets-w2s-013103.php

Noy, N. (2004). Semantic Integration: A Survey of Ontology-Based Approaches. *SIGMOD , 33* (4), 65-70.

Noy, N., & Musen, M. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. *Proceedings of the Seventeenth National Conference on Artificial Intelligence.* Austin: AAAI/MIT Press.

OAEI. (2011). *Ontology Alignment Evaluation Initiative*. Retrieved 1 21, 2012, from OAEI website: http://oaei.ontologymatching.org/2011/

Organization, W. S. (2008, 11 26). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Retrieved 9 20, 2011, from W3C: http://www.w3.org/TR/2008/REC-xml-20081126

Organization, W. S. (2001). *Semantic Web Standards*. Retrieved 9 20, 2011, from W3C: http://www.w3.org/2001/sw

Pride. (2010, 9 29). *Ontology Lookup Service*. Retrieved 1 23, 2012, from ols: http://www.ebi.ac.uk/ontology-lookup/

Programmableweb. (2012, 1 22). *API Dashboard*. Retrieved 1 22, 2012, from programmable web: http://www.programmableweb.com/apis

protege. (2011). *protege*. Retrieved 1 21, 2012, from standford: http://protege.stanford.edu/

Ramler, R., & Wolfmaier, K. (2008, October 9). Issues and Effort in Integrating Data from Heterogeneous Software Repositories and Corporate Databases. *ESEM* .

sameAs. (2012, 1). *sameAs interlinking the Web of Data*. Retrieved 1 21, 2012, from samas: http://sameas.org/

SAS. (2011, 12 3). *SAP Business Objects*. Retrieved 12 3, 2011, from http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/index.epx

Sauser, B., & al, e. (2010). Integration Maturity Metrics: Development of an Integration Readiness Level. *Information Knowledge System Management , 9*.

Sheth, A., & Ranabahu, A. (2010, 5). Semantic integration modeling for Cloud Computing, Part 1. *Internet Computing* , 81-84.

Sheu, P., Wang, S., Wang, Q., & Paul, R. (2009). Semantic Computing, Cloud Computing, and Semantic Search Engine. *2009 IEEE International Conference on Semantic Computing.* IEEE.

sindice. (2012). *Indice The Semantic Web Index*. (Deri, Producer) Retrieved 1 21, 2012, from Sindice: http://www.sindice.com/

Tufte, E. (2001). *The Visual Display of Quantitative Information.* Graphics Pr.

Uschold, M., & Gruninger, M. (2004). Ontologies and Semantics for Seamless Connectivity. *SIGMOD Record , 33* (4), 58-64.

van den Heuvel, C., & Rayward, W. (2011). Facing Interfaces: Paul Otlet's Visualization of Data Integration. *Journal of the American Society for Information Science and Technology , 62* (12), 2313-26.

Verter, G. (2010). Semantics in the Age of the Data Deluge. In A. D'Atri (Ed.), *Information Systems: People, Organization, Institutions, and Technologies* (pp. 415-421). Physica-Verlag HD.

Vetere, G. (2010). Semantics in the Age of the Data Deluge. In D. Sacca, *Information Systems: People, Organizations, Insttitutions, and Technologies* (pp. 415-421). Physica-Verlag HD.

W3C. (2011, 11 8). *Wordnet in RDF*. Retrieved 11 8, 2011, from http://www.w3.org/2006/03/wn/wn20/

W3C, Prud'hommeaux, E., & Seaborne, A. (2008, 1). *SPARQL Query Language for RDF*. Retrieved 1 21, 2012, from W3C: http://www.w3.org/TR/rdf-sparql-query/

Wikipedia. (2012, 1 11). *Cloud database*. Retrieved 1 21, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Cloud_database

Wikipedia. (2011, September 20). *Data Integration*. Retrieved September 20, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Data_Integration

Wikipedia. (2011, 5 11). *Semantic Computing*. Retrieved 1 21, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Semantic_computing

Wikipedia. (2011, 12 23). *Semantic Reasoner*. Retrieved 1 21, 2012, from wikipedia: http://en.wikipedia.org/wiki/Semantic_reasoner

Wikipedia. (2012, 1 19). *Triplestore*. Retrieved 1 21, 2012, from wikipedia: http://en.wikipedia.org/wiki/Triple_stores

Williams, A. (2009, 11 25). *Merrill Lynch: Cloud Computing Market Will Reach $160 Billion...Really*. Retrieved 1 7, 2012, from readwriteweb.com: http://www.readwriteweb.com/enterprise/2009/11/merrill-lynch-cloud-computing.php

Yamaguchi, Y., & al, e. (2011). Tag-Based User Discovery using Twitter Lists. *2011 International Conferences on Advances in Social Networks Analysis and Mining.*

YongTao, J., FengJuan, Q., & HuiJuan, W. (2010). Ontology-Based Research on Heterogeneous Database Semantic Integration Strategies. *2010 Second International Workshop on Education Technology and Computer Science. 10*, pp. 477-479. IEEE.

Youseff, L., Butrico, M., & Da Silva, D. (2009). Toward a Unified Ontology of Cloud Computing. *Grid Computing Environments Workshop, 2008. GCE '08* , (pp. 1-10). Austin.

Zeyliger, P. (2010, 3 22). *How Raytheon BBN Technologies Researchers are Using Hadoop to Build a Scalable, Distributed Triple Store*. Retrieved 8 19, 2010, from

Developer Center: http://www.cloudera.com/blog/2010/03/how-raytheon-researchers-are-using-hadoop-to-build-a-scalable-distributed-triple-store/

Zhou, J. e. (2010). A Survey of Semantic Enterprise Information Integration. *Information Sciences and Interaction Sciences (ICIS).*

Zhu, Y. e. (2008). Data Updating and Query in Real-Time Data Warehouse System. *2008 International Conference on Computer Science and Software Engineering.*

Ziegler, P., & Dittrich, K. (2004). Three Decades of Data Integration - All Problems Solved. *18th IFIP World Computer Congress - Building the Information Society , 12*.