INSTITUTE FOR DEFENSE ANALYSES

Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 3, Risks, Mitigation Approach, and Roadmap

Francisco L. Loaiza-Lemos, *Project Leader*
Russell J. Smith

December 29, 2017

IDA Document
D-8852

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a non-profit corporation that operates three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges.

About This Publication

This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Task BC-5-4277, "Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure Portal Implementation," for Army CIO/G-6 (SAIS-AOD). The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
David A. Wheeler

For more information:

Francisco L. Loaiza-Lemos, Project Leader
floaiza@ida.org, 703-845-687

Margaret E. Myers, Director, Information Technology and Systems Division
mmyers@ida.org, 703-578-2782

INSTITUTE FOR DEFENSE ANALYSES

IDA Document D-8852

Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 3, Risks, Mitigation Approach, and Roadmap

Francisco L. Loaiza-Lemos, *Project Leader*

Russell J. Smith

Executive Summary

This document was prepared by the Institute for Defense Analyses (IDA) in support of the FY 16 Army Study "Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation."

This document constitutes the third deliverable under the project description and addresses the study's objective of assessing the maturity and applicability of graph database technology as a practicable materiel solution that reflects legacy system realities and that can effectively and efficiently deliver the needed *at-rest* and *in-motion* force structure products for the planned DFS portal.

Specifically, the third deliverable identifies technical risks together with suitable mitigation approaches and describes a roadmap and its milestones for a possible technology insertion to support the planned Army DFS Portal. The IDA team applied rapid prototyping techniques as part of the continuing evaluation of alternatives to stress the implementations of the graph databases chosen for the study. The team used data collected during those activities to continue maturing the decision process needed to determine the best-of-breed options. The assessments leverage the metrics elaborated in the initial phase of the study, which were documented in the first deliverable.[1]

Background

This phase of the study is aligned with the goals and objectives of the Department of Defense (DoD), as expressed in its Global Force Management Data Initiative (GFM DI), whereby DoD is seeking the standardization of all authorized force structure data so that it can be understandable to, and usable by, both warfighting and business systems across the DoD Enterprise.[2] As noted in the two previous [3] deliverables under this project, the challenge in all of the related activities is the harmonization of data that currently resides

---

[1] IDA Document D-8345, *Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 1, Preliminary Characterization of Data Sources, Representation Options, Test Scenarios and Objective Metrics*, F. Loaiza, D. Visser, February 24, 2017.

[2] http://www.prim.osd.mil/init/init_osdmanpower.html

[3] The second deliverable was IDA Document D-8516, *Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 2, Technical Feasibility, Affordability, and Architecture Integration Options*, F. Loaiza, D. Visser, June1, 2017.

in a large number of relational legacy systems so that it can be readily used in the generation of *at-rest* and *in-motion* force structure products.

The main motivation for exploring graph database technology has been its potential for cost reduction along with the procedural simplicity of an approach that directly recasts the legacy source data in the form of Resource Description Framework (RDF) triples,[4] collects them in a graph data store, and then uses the triples to generate the force structure products. However, the adoption of any new technology, with the changes that it brings to established workflows and procedures, carries technical risks. This phase of the analysis, therefore, further explores the nature and severity of those risks, and suitable ways to mitigate them. This document also presents a preliminary road map aligned with the overarching DoD and Army strategy to make data visible, accessible, understandable, trusted, and interoperable and discusses milestones for a possible technology insertion to support the planned Army DFS Portal.

Document Structure

This document is organized as follows:

1. Section 1 presents a catalog of technical risks likely to show up when replacing relational data stores with graph databases as the main technology supporting the storage and retrieval of data needed to operate the Army DFS Portal, and how best to mitigate them.
2. Section 2 documents a possible road map for incorporating graph databases into the mix of technologies supporting the Army DFS Portal, and how the adoption of this technology is consistent with the DoD data strategy.
3. Section 3 provides the current set of conclusions and recommendations for this phase of the study.
4. Appendix A revisits the notion of using a "semantic layer" (e.g., the explicit addition of classes and semantic declarations, such as class equivalences, to the knowledge base) to enable the harmonization and manipulation of data from disparate sources using an alternate representation that lends itself to programmatic manipulation. Specifically, the discussion highlights the ability to use different representations to improve the efficiency of the data processing and manipulation. With regard to the latter point, the appendix discusses the use of well-established programming languages, such as Prolog,[5] for this purpose.

---

[4] https://www.w3.org/RDF/

[5] Prolog was chosen during this phase of the project because there are already graph database implementations that support the use of Prolog for data extraction, as an alternative to SPARQL queries. In addition, since Prolog is a declarative programming language, the queries look quite similar to the SPARQL counterparts.

5. Appendix B contains a number of Python scripts that were used to generate the test data used in testing the ideas presented in Appendix A. The code is licensed for free reuse, and it is intended to help other groups in their evaluations.

Scope

As in the two previous deliverables, the results described in this document do not address any of the complexities inherent in the policies and procedures embedded in the "as-is" systems that currently support the population of the Army Organization Server under the GFM DI initiative, which would come into play for scenarios in which the source data to be converted into RDF triples is in the form of XML instance documents that conform to the GFM DI specifications. It is, therefore, assumed that those XML instance documents can both be generated and would be accessible as inputs for subsequent manipulations required by the graph database approach.

Although other popular programming languages now offer libraries for creating, querying, and modifying graphs, this phase of the analysis did not attempt to compare and contrast them in relation to the language used as an example of programmatic manipulations, namely, Prolog. Depending on time and resources available, the IDA team may be able to revisit this aspect of the graph database technology use and document it in the final report.

Finally, as noted in the two previous deliverables, the performance of off-the-shelf, standard computer equipment has proven inadequate for handling in near real time, i.e., in a second or less, large graphs, i.e., graphs containing tens or hundreds of billions of triples. The near-real time search and retrieval of data in those scenarios most likely will need special-purpose hardware and software. Therefore, the data sets used in this phase of the study serve only to demonstrate the underlying principles and are not to be interpreted as reference performance benchmarks for actual implementation.

Analytical Approach

The work performed for this phase of the study concentrated on answering the following questions:

- What are the main technical risks associated with the use of graph databases as part of the technology mix supporting the Army DFS Portal?
- What mitigation approaches can be brought to bear so that the potential benefits associated with the use of graph databases will not be negated by the associated technical risks?
- What implementation roadmap would be most appropriate in light of all the risks and alternatives?

- What key steps should be taken first to facilitate the adoption of graph databases as part of the overall solution architecture supporting the Army DFS Portal?
- What are the enterprise-wide implications for the Army of adopting a graph database approach?
- How can other data representations of the RDF triples content be leveraged for the purpose of implementing a semantic layer that aids in the harmonization of data from multiple disparate sources?
- What are the lessons learned and how can they help inform the decision process needed to determine best-of-breed options?

Conclusions and Recommendations

Based on the analytical work performed during this phase, the IDA team concluded the following:

- As briefly noted in the previous deliverables, the main risk associated with the adoption of graph databases when compared to relational data stores in the context of massive graphs is their inferior performance with respect to data retrieval and complex query execution. For interactive applications, any data storage and retrieval technology that requires more than one or two seconds to deliver the answer is unlikely to be a strong contender in the solution architecture that supports those use cases.
- However, some proprietary graph database solutions for "big data" are reaching a sufficient level of maturity to be competitive with relational data stores in terms of performance. Specifically, the combination of graph databases and frameworks for distributed storage and processing, such as Apache Hadoop and Apache Spark, make it possible to efficiently partition very large datasets to compensate for any slowdowns caused by the size of the graphs.
- The idea of a "semantic layer" for organizing the resources in an RDF triple store can be readily implemented using alternative data representations that are not only closely related to the graph formalism – and, therefore, can be readily converted back and forth – but that also can be directly processed using a programming language (e.g., Prolog).
- The key rationale for using graph databases is mainly to enable the cost-effective handling of legacy data, bypassing the laborious and expensive extraction, transformation, and loading (ETL) associated with traditional approaches, and said rationale is supported by all the findings obtained so far.

For this stage of the study, the preliminary recommendations are as follows:

- Continue the evaluation of available graph database implementations, both proprietary and open source, and expand the scope to include other promising NoSQL choices.
- Conduct additional comparisons regarding the use of other programming languages and data representations for the purpose of implementing a "semantic layer" as part of the graph database solution.
- Explore applicable emerging "big data" solutions with regard to their applicability in a future implementation of the Army DFS Portal.

Contents

Figures and Tables

1. Technical Risks and Mitigation Approach

This chapter discusses a number of the typical risks associated with the use of graph databases and ways in which those risks can be mitigated. The catalog is not exhaustive.

A. Performance Risk

Table 1-1. Summary of Performance Risks and suggested Mitigation Approach

Performance Degradation	
Risk	**Mitigation**
• Queries become slow as the size of the graph grows	• Implement the graph database with optimized hardware. This may include among other things housing the application in one or more powerful, dedicated servers with multiple multi-core processors; provisioning the servers with large RAM capacity (128 GB or more); and using high-speed solid-state hard drives for data persistence. • Partition large graphs into separate subcomponents hosted in high-performance machines that can be managed as a single instance using frameworks for distributed storage and processing such as Hadoop.[6] • Use, where appropriate, the equivalent of "materialized views" for queries that take substantial time to complete (e.g., tens or even hundreds of minutes). This means that the slow queries are executed off-line against the complete graph and the results are then stored as a secondary graph to be used to respond to subsequent requests. • Develop special-purpose code to supplement the capabilities of the graph database engine being used. This may include using hybrid solutions where the data may be stored in multiple representations (e.g., RDF triples and compiled binary files), some of which can then be processed with utilities written in high-performance languages (e.g., C/C++). • Host the entire graph database in a highly scalable cloud solution, such as Amazon Web Services, that can handle not only the data volume demands but also the processing requirements.

Preliminary tests conducted with some representative graph database applications – already documented in the two previous deliverables[7,8]– show that the size of the graphs can have a severe impact on performance.

---

[6] http://hadoop.apache.org/

[7] IDA Document D-8345, *Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 1, Preliminary Characterization of Data Sources, Representation Options, Test Scenarios and Objective Metrics*, F. Loaiza, D. Visser, February 24, 2017.

[8] IDA Document D-8516, *Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 2, Technical Feasibility, Affordability, and Architecture Integration Options*, F. Loaiza, D. Visser, June 1, 2017.

For example, when using a non-optimized hardware configuration (e.g., a laptop with one Intel i7 processor having eight cores and 16 GB of RAM), queries that involve as little as 8 million subgraph traversals, each involving from two to six edges, will sometimes require more than 10 seconds. This performance is clearly inadequate if one intends to power an interactive, web-based solution, where users expect queries to execute in under one second.

Table 1-1 above summarizes the performance risks that can be expected when adopting graph database technologies and the ways in which they can be mitigated. We highlight here that the elasticity, the use of a pay-as-you-go consumption model, and the speed of deployment makes cloud solutions very appealing, especially if classification issues and protection against cyber-attacks can be properly mitigated.

B. Data Quality Degradation Risk

The success of an information service, such as the planned Army DFS portal, depends not only on having adequate response times, but perhaps even more so, on the quality of the data. When users feel that the data is either obsolete or unreliable, they will stop using the information service. A well-understood and applicable method for ensuring good data quality is to adopt a comprehensive data governance for all the resources that are incorporated and maintained in the Army DFS.[9]

In addition, when leveraging the capability of graph databases to store data from any number of sources to create a de facto *data lake*, it is imperative that one maintain complete oversight of the resources that have been committed to the repository so that they can continue to be accessible and processable by the users. Failure to figure out which data and metadata are essential to power the information services to be offered by the Army DFS portal may turn the underlying graph database from a data lake into a *data swamp*.[10]

Table 1-2 summarizes the data quality risks that can be expected when adopting graph database technologies and the ways in which they can be mitigated.

---

[9] A fairly detailed analysis of the issues related to data quality can be found in IDA Document D-4275, *Development of a Data Quality Framework for Creating and Maintaining Army Authoritative Data Sources*, F. Loaiza, C. Roby, E. Simaitis, S. Wartik, March 2011.

[10] https://www.cio.com/article/3199994/big-data/3-keys-to-keep-your-data-lake-from-becoming-a-data-swamp.html

Table 1-2. Summary of Data Quality Risks and suggested Mitigation Approach

Data Quality Degradation	
Risk	**Mitigation**
• Underlying data lake turns into a data swamp	• Analyze data sources and specify maximum data volumes to be transferred from each source during the initial phase of the creation of the data lake stored in the graph database. Keep in mind that ease of data collection does not necessarily equate to ease of data use. • Analyze the data sources and build a conceptual information model to define the metadata needed to characterize it. Associate the appropriate metadata to each piece of data collected in the data lake. • Assess the applicability of techniques such as unsupervised machine learning to help mature and enrich the conceptual information model with the required metadata. • Implement a data governance process to ensure data quality during the life cycle of the graph database implementation and use. Define quantitative metrics to create verifiable data quality targets (e.g., **metric**: percentage of RDF triples without associated metadata; **data quality target**: less than 5%). • Analyze the data sources and determine when their data is likely to become obsolete. Define a process to prune obsolete data from the data lake to ensure good performance and high data quality. • Conduct on a regular basis (annual, biennial) a comprehensive review of the evolving goals and objectives of the Army DFS portal to ensure that the implemented solution is adequately aligned. • Conduct on a regular basis (annual, biennial) a review of emerging technologies applicable to data quality maintenance and improvement (e.g., advances in artificial intelligence, natural language processing, etc.). • Establish a lifecycle management strategy for the preservation and protection of all the digital assets comprising the Army DFS portal.

C. Cybersecurity Risk

Graph databases offer an elegant way to collect data into single repositories that can then satisfy the needs of multiple components of the enterprise. But in the case of the planned Army DFS, this same capability also represents a risk, since aggregating large volumes of force structure data can reveal sensitive aspects of the processes employed by the Army to maintain and regenerate its forces. Expressed in a different way, a fully populated and operational Army DFS portal constitutes a very attractive target for cyber-attacks intended to either damage it or exfiltrate its contents.

As stated in DoD Directive 8000.1, it is DoD policy to treat information as a strategic asset and to protect it to the maximum extent possible.[11] Extensive and applicable cybersecurity guidance for the Army DFS portal can be found in (1) a series of National Institute of Standards and Technology (NIST) Special Publications (SP), (2) in guidance

[11] DoDD 8000.1, *Management of the Department of Defense Information Enterprise (DoD IE)*, March 17, 2016 (available at https://fas.org/irp/doddir/dod/d8000_01.pdf).

from the Committee on National Security Systems (CNSS), and (3) in the Federal Information Processing Standards (FIPS). NIST partnered with the DoD, the Office of the Director of National Intelligence (ODNI), and the CNSS to develop a common information security framework for the federal government and its contractors. We highlight in this section some of the applicable guidance documents (see Table 1-3).

Table 1-3. Summary of Applicable Federal Information Systems Cybersecurity Publications

Federal Information Systems Cybersecurity Publications	
Publication Number	**Title**
• NIST SP 800-37 rev 1	• *Guide for Applying the Risk Management Framework to Federal Information Systems - A Security Life Cycle Approach*
• NIST SP 800-53 rev 4 [12]	• *Security and Privacy Controls for Federal Information Systems and Organizations*
• NIST SP 800-30 rev 1	• *Guide for Conducting Risk Assessments*
• FIPS 199	• *Standards for Security Categorization of Federal Information and Information Systems*
• FIPS 200	• *Minimum Security Requirements for Federal Information and Information Systems*
• CNSSI 1253	• *Security Categorization and Control Selection for National Security Systems*

Determining the security controls required to ensure adequate protection of the data that would be stored in a graph database powering the Army DFS portal starts with a set of steps listed in NIST SP 800-37 rev 1. That publication describes a Risk Management Framework (RMF) process comprising the following six steps:

- Step 1 – Categorize the Information System,
- Step 2 – Select Security Controls,
- Step 3 – Implement Security Controls,
- Step 4 – Assess Security Controls,
- Step 5 – Authorize Information System,
- Step 6 – Monitor Security Controls.

Incorporating cybersecurity measures consistent with the RMF in the Army DFS portal development cycle – from the start rather than as an afterthought – can help determine key aspects of the implementation, such as the appropriate hosting location, the appropriate classification of the data loaded in the graph database, and the subsequent selection of the necessary security controls. The other documents listed in Table 1-3 can be used to build the necessary justification to obtain the required Authority to Operate (ATO). The DFS portal will receive its ATO – as most Army systems do – from the Army CIO/G6. The NIST SP 800-53 rev 4 control catalog contains security controls that the

[12] Revision 5 is expected to be released at the end of December 2017.

developers of the Army DFS portal ought to pay special attention to. Table 1-4 lists a subset of controls in the Access Control (AC) family intended to protect the confidentiality and integrity of the graph database.

Table 1-4. Sample of Access Control (AC) Family Controls for the Army DFS

Access Control (AC) Family Cybersecurity Controls	
Control Number	**Control Description**
• AC-20	• "Use of External Information Systems" – Provides guidance applicable to exchanges of information with systems outside of the DFS portal.
• AC-21	• "Information Sharing" – Establishes criteria for exchanging information based on privileges of the authorized users.
• AC-22	• "Publically Accessible Content" – Establishes criteria for determining whether or not to place DFS portal content in a publically accessible system.
• AC-23	• "Data Mining Protection" – Provides guidance regarding data mining prevention and detection techniques including, for example: (i) limiting the types of responses provided to database queries; (ii) limiting the number/frequency of database queries to increase the work factor needed to determine the contents of such databases; and (iii) notifying organizational personnel when atypical database queries or accesses occur.
• AC-24	• "Access Control Decisions" – Ensures that access control procedures are established within the graph database.

The Army DFS portal will have to be hosted at the correct level of network classification, and given the variety of data repositories across multiple domains that will provide either legacy data or updated current data, accommodations will have to be made to ensure that portal queries and data from potentially different classifications can be exchanged adequately. Cross-domain solutions (CDS) can provide essential segmentation and isolation of the data needed to handle data correctly in accordance with its classification level while enabling queries and data to cross the classification boundaries. Table 1-5 summarizes the data aggregation risks and mitigation approaches for protecting across security domains.

Table 1-5. Summary of Data Aggregation Risks across Security Domains and suggested Mitigation Approaches

Data Aggregation Risks	
Risk	**Mitigation**
• Inadequate protection when handling data resident in systems exhibiting multiple classification levels may prevent optimal DFS operations	• Analyze aggregated data for proper classification • Store aggregated data in a data lake at the highest classification level • Use CDS to exchange data from lower data repositories to the higher classification network hosting the graph data lake • Leverage CDS to pass through graph database queries from higher to lower classification networks • Implement CDS based on guidance from the Cross Domain Enterprise Service (CDES) offered through the Defense Information Systems Agency (DISA)

In addition to the above measures aimed at protecting data needed by the future Army DFS portal, the IDA team highly recommends that the implementation leverage the lessons learned and best practices from both government and the commercial world. Adopting what industry considers the best approaches in Identity and Access Management Service (IAMS) can minimize the risks associated with unauthorized access. Liberal use of encryption of data, both at rest and in transit, can be extremely helpful in reducing the potential damage associated with data exfiltrated during a cyberattack.

D. Additional Regulatory Frameworks Non-Compliance Risk

Although most of the activities that will involve the future Army DFS portal may take place within the national boundaries, its implementation should consider the impact of using it to host non-U.S. force structure data during international and coalition missions. Review of the various status of forces agreements (SoFAs) may be necessary to avoid conflict with regulatory data protection schemes that are being adopted in the near future, such as the General Data Protection Regulation (GDPR) within the European Union (EU), which will come into effect in May 2018.[13]

Any American company doing business or supporting users/customers in the EU will have to comply with the GDPR. Because of the severity of the penalties, many information technology (IT) contractors in the United States may begin to build into their solutions components needed to ensure compliance with the GDPR. Both cost and performance risks are potentially associated with this development (see Table 1-6).

[13] https://www.eugdpr.org/

Table 1-6. Risks Associated with Additional Regulatory Frameworks

Non-U.S. Regulatory Frameworks Compliance Risks	
Risk	**Mitigation**
• Heightened protection of personal data required by non-U.S. regulatory frameworks (e.g., GDPR) may negatively affect coalition and multinational missions.	• Review existing SoFAs and seek to update them to prevent non-compliance issues when conducting coalition and multinational missions. • Identify when personal data can be excluded or pseudonymized when shared to reduce the risk of exposure. • Review the operational impact of using IT solutions that already satisfy requirements imposed by the non-U.S. regulatory frameworks and assess whether it is mainly a cost issue or whether it affects performance and policy. For example, ability to retain shared non-U.S. force structure data containing personally identifiable information (PII), need to ensure adequate encryption for PII, right to request erasure of PII data after some specified period of time. • Review the operational impact caused by a requirement to report breaches and data leaks in systems located outside of the United States but used to operate/interface with the DFS portal and carve exceptions where the burden of compliance is deemed to be unacceptable from an operational point of view.

E. Work Flow Risk

As noted earlier, the adoption of graph databases will also trigger changes in the current workflows. Some of the risks associated with these changes are highlighted in Table 1-7.

Table 1-7. Risks Associated with Changes in the Workflows

Non-U.S. Regulatory Frameworks Compliance Risks	
Risk	**Mitigation**
• New workflows do not support requirements for metadata collection and integration.	• Perform a thorough review of metadata requirements and ensure that the workflows can associate each data item collected with the necessary metadata. • Explore the applicability of automation techniques to reduce human error and ensure high quality of results. • Develop automated test cases and submit all outputs from workflows to careful testing prior to ingestion – in particular, to ensure that all data remains visible and accessible. • Maintain careful versioning of the graph database content to ensure that errors in ingested data can be efficiently corrected. • Conduct on a regular basis (annual, biennial) a review of emerging technologies applicable to data quality monitoring and data governance enforcement.

2. Roadmap for Using Graph Databases in the Army DFS portal

A. Background

The commercial world continues to realize and exploit the benefits that can be obtained from applying data analytics and business intelligence processes to large data collections that leverage both graph database and cloud implementations of data lakes.[14] Practitioners from the commercial world highlight the following best practices and lessons learned:[15]

- Cost and schedule estimates should not be overly optimistic – it "is going to cost more and take longer than you planned."[16]
- Setting up the solution architecture may require as long as it will take to fine-tune it – a one-year horizon may not be unusual.[17]
- For heavy and/or repetitive data migration jobs, it may be advisable to build tools that automate, at least in part, this aspect of the data migration.
- Cost saving strategies offered by cloud providers (e.g., bidding on spare compute nodes for temporary processing uses) can bring substantial savings but should be used sparingly within the rate-determining steps of critical workflows because the processing nodes can be snatched away by users willing to pay more for them.
- The maturity of both proprietary and open source implementations selected as part of the solution architecture need to be carefully vetted to prevent common but costly issues, such as file corruption and data losses, caused by undetected bugs in the software releases.

[14] http://bigdata.teradata.com/US/Articles-News/7-Questions-About-Data-Lakes-and-Hadoop/

[15] http://searchdatamanagement.techtarget.com/feature/Cloud-big-data-clusters-test-users-on-migration-management?utm_medium=EM&asrc=EM_NLN_83870256&utm_campaign=20171012_Migration%20dragons,%20zombie%20clusters%20and%20other%20dangers%20in%20big%20data%20clouds;%20Oracle's%20machine%20learning%20push&utm_source=NLN&track=NL-1816&ad=917048&src=917048

[16] Statement by Chris Mills, who leads the big data team at The Meet Group Inc. (see Footnote 10 above).

[17] See the example from the Meet Group in Footnote 10 above.

- In a similar vein, careful consideration should be given to the mix of private and commercial clouds – not every process is suited for migration to a commercial cloud, and some may need to remain within controlled non-cloud environments.

B. Graph Data Base Technology Insertion – Timeline and Milestones

After selecting the specific graph database implementation that will be used in the planned Army DFS portal, a series of key steps should be taken to ensure that the overall architecture solution is adequate and can deliver the expected functionality.

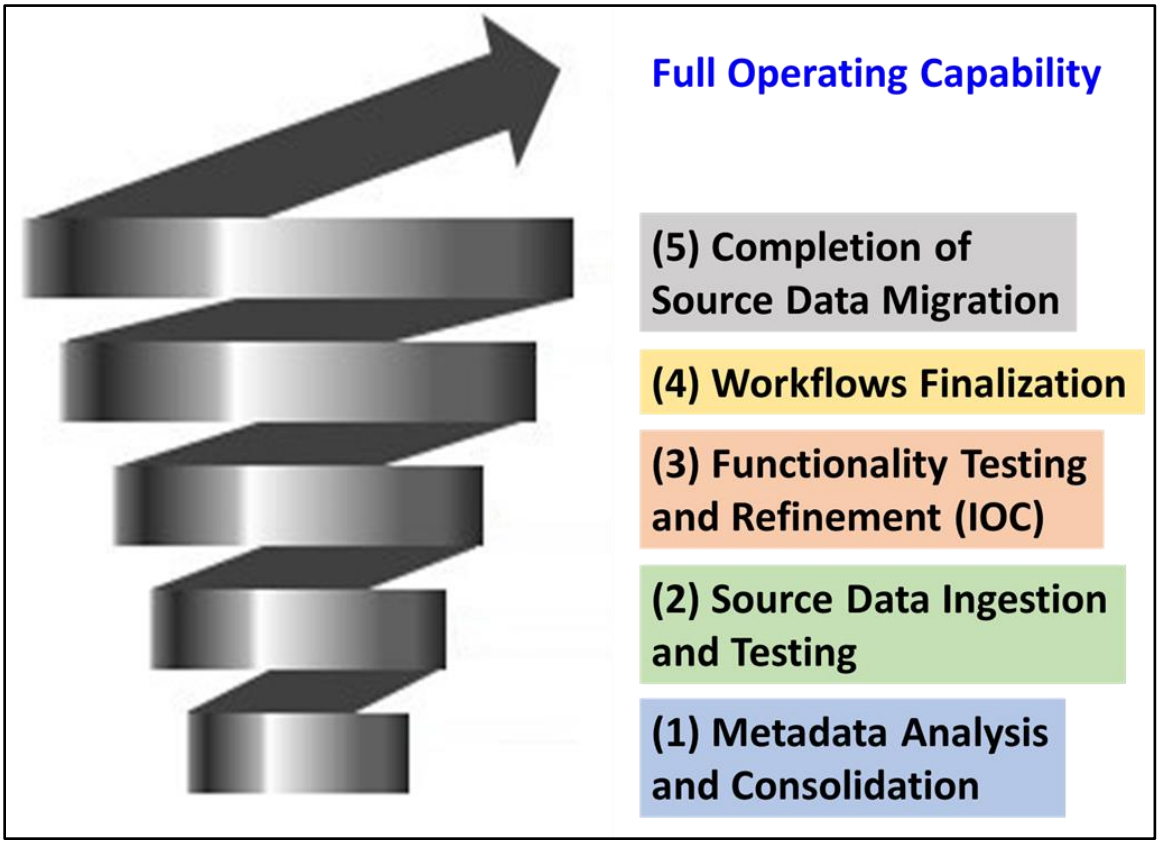

Figure 2-1. Notional Spiral Development for Integrating Graph Database Capabilities in the DFS Portal

Figure 2-1 highlights, in the form of notional spiral development, five of the most important steps that should be carried out over a period of six to 12 months, which would address a number of the potential risks associated with the use of graph databases, as discussed in the preceding chapter.

Step 1 comprises analysis of the required metadata tags that need to be added to the RDF triples generated from the source relational data stores and harmonization and consolidation of the metadata tags that will ensure that the SPARQL queries can always reach all the data within the graph database. Step 2 comprises data ingestion of the properly metadata-tagged RDF triples, testing of the functionality of the queries, and performance

of the entire solution. This step also ensures that the response times are within the acceptable range of values needed for the selected use cases. In Step 3, if the basic functionality tests are satisfactorily concluded, the data retrieval and manipulation capabilities of the implemented solution can be extended and refined to cover the full set of functional requirements. Step 4 allows the implementers to review and finalize the workflows. These ought to cover metadata extraction, harmonization, and consolidation – which will be required every time RDF triples from a new legacy data source are added to the graph database – as well as tagging and testing of the SPARQL queries against updated data. Workflows for maintenance functions such as performing backups, as well as less common ones intended to perform periodic revision and update of the workflows themselves to accommodate new requirements for the Army DFS Portal over its life cycle should also be considered. Step 5, the final one in the notional spiral development shown in Figure 2-1, covers the completion of the data migration for the sources identified for use in the fully operational release version of the graph database.

As noted in the preceding paragraph, the expectation is that the workflows will contain all the steps necessary to operate the graph database and support the capabilities of the planned Army DFS Portal over its entire life cycle.

C. Strategic Plan Implementation Management

The overarching DoD, as well as the Army, data strategy is to make data visible (V), accessible (A), understandable (U), trusted (T), and interoperable (I) – [VAUTI].[18] Strategic plans generally promulgate both goals and objectives aimed at realizing the strategy. The challenge for the enterprise is to ensure that these goals and objectives are reached so that the overall strategy is successfully carried out. A methodology that specifically focuses on the management of a strategic plan is the *balanced scorecard methodology*.[19] Its approach consists of identifying the metrics and measures needed to assess the progress being made in each of the goals and objectives of a strategic plan, which in turn guides the implementation activities toward the desired end state. The balanced scorecard methodology uses four perspectives: financial, internal processes, organizational, and end user.

The intent is to carefully select the metrics and measures for each of those perspectives so that they are in optimal alignment with the chosen strategy, thereby ensuring that the enterprise will achieve its goals and strategic objectives. Because the implementation of the strategy stretches over time, it is beneficial to consider metrics and

---

[18] Army Data Strategy, February 2016, available at http://ciog6.army.mil/Portals/1/Home/Tabs/Strategy/20160303_Army_Data_Strategy_2016.pdf

[19] *Balanced Scorecard Step-by-Step for Government and Nonprofit Agencies*, Second Edition, Paul R. Niven, John Wiley and Sons, Inc., ISBN 978-0-470-18002-0, 2008.

measures that apply before the systems that compose a given solution architecture are in place, those that must be applied once the system achieves its initial operational capability, and those that apply when the system enters its full operational capability and for the rest of its expected life cycle.

D. Perspectives Description and Purpose

As noted above, the balanced scorecard methodology partitions the management of a strategic plan implementation along four perspectives. Their brief overviews and purposes are presented below.

1. Financial Perspective

The financial metrics and measures are intended to ensure that the organizations charged with the implementation of their respective data strategy plans can achieve their objectives in an effective and efficient manner, which minimizes risks associated with the typical and unavoidable DoD budget fluctuations. In our case, these metrics and measures call for the identification of the cost models and activities needed to ensure that the planned Army DFS Portal is adequately funded.

2. Internal Process Perspective

The internal process metrics and measures are applied to processes that are essential to achieving the Army VAUTI data strategy. This covers the efficient operation, from the perspective of the end users, of the processes that fulfill both the mission of information systems, such as the planned Army DFS portal, and the value proposition of the organization. The metrics and measures are intended to help the organizations charged with implementing and running the Army DFS Portal to remain as close as possible to the Army data strategy. End user satisfaction may require periodic revision of the internal processes rather than just focusing on the incremental improvement of existing activities. Examples of areas that merit review and update are Service development and delivery, partnering with the community, and reporting.

3. Organizational Perspective

The organizational metrics and measures are essentially enablers for the other three perspectives, and their importance should not be underestimated since they are foundational to the achievement of the goals and objectives of the Army's data strategy. These metrics and measures ameliorate the gaps that may exist between the organizational infrastructure of employee skills, the information systems, and the organizational climate (e.g., culture) on the one hand and the skill levels in those areas necessary to achieve the results that Army organizations participating in the implementation and operation of the Army DFS Portal have identified.

4. End User Perspective

The end user metrics and measures aim at identifying the needs and expectations of the intended end users of the planned Army DFS Portal, i.e., Who are they? What do they expect from the Army DFS Portal? In addition, these metrics and measures also can guide the organizations involved in the implementation and operation of the Army DFS Portal to pinpoint the value added for the end users, which can then be used to justify funding and the continuation of the Army DFS Portal operations.

E. Correlation of Graph Database Use with Goals and Objectives within the Army VAUTI Data Strategy

Table 2-1 presents an assessment of how the use of graph database technology can benefit the Army VAUTI data strategy ("H" denotes a high benefit; "M" denotes a medium benefit). As noted in previous deliverables, use of a common, and relatively simple format, such as the one employed for RDF triples, is a very strong facilitator for exposing data in a manner that is easy to post and retrieve. This in turn supports the goal of making data visible.

Table 2-1. Assessment of Benefit from Using Graph Database Technology with Regard to the Army VAUTI Data Strategy Goals and Enabling Objectives

Army Data Strategy Goals and Enabling Objectives		Graph DB Benefit
GOALS	**OBJECTIVES**	
Make Data Visible (V)	Post Data to Shared Spaces	H
	Register Metadata Related to Structure and Definition	—
Make Data Accessible (A)	Create Shared Spaces and Data Services (Also Information and IT Services)	H
	Associate Security-Related Metadata	—
Make Data Understandable (U)	Create Data Models	H
	Establish Data Integration	H
	Identify Information Requirements Traceability	M
Make Data Trusted (T)	Identify Authoritative Data Sources	—
	Create Secured Availability (Data Security and Data Access Security)	—
Make Data Interoperable (I)	Comply with Information Exchange Specifications	H
	Establish Master Data Management/Unique Identifiers	M
	Establish Community-Based Information Sharing	H
	Establish Translation and Mediation	H

Similarly, using a technology that lowers the barrier to posting and retrieving data supports the efficient construction of shared spaces and data services that use that type of data store. This in turn supports the goal of making data accessible.

As noted in the preceding sections, a key step in the use of the graph database technology to support the operations of the planned Army DFS Portal is the identification of the metadata needed to ensure that the data can be retrieved using appropriate queries. The harmonization of the metadata is also an enabler for better data integration. This in turn supports making the data understandable.

Perhaps the goal and objectives best supported by the use of graph database technology is the one related to data interoperability. Not only would be easier to exchange data among graph databases that store their records in a representation such as RDF, but the federation of data resources would be much simpler and could leverage currently available semantic tools.

F. Recommended Metrics and Measures for the Army DFS Portal

1. Financial Perspective

Table 2-2 shows a set of metrics and measures that could be applied to the implementation of the planned Army DFS Portal under the financial perspective of the balanced scorecard methodology.

Table 2-2. Metrics and Measures for the Financial Perspective

Fiscal Metrics and Measures		
Metric	**Time line**	**Measure**
• Develop a cost model for all materiel required for the Army DFS Portal	LoB*	• 100% coverage of essential components for the Army DFS Portal
• Develop a life-cycle cost model for the Army DFS Portal equipment	LoB	• 100% coverage of maintenance costs for the selected solution architecture of the Army DFS Portal
• Develop cost model for the Army DFS Portal personnel costs	LoB	• 100% coverage of costs associated with staffing and training of personnel required for full operational capability of the Army DFS Portal
• Obtain funding for the initial implementation phase of the Army DFS Portal	LoB	• 100% coverage of costs for first spiral of the Army DFS Portal implementation plan
• Obtain funding for completion of the Army DFS Portal implementation plan	RoB*	• 100% coverage of costs for successive spirals comprising the Army DFS Portal implementation plan
• Obtain funding for the Army DFS Portal life-cycle	RoB	• 100% of yearly coverage of operational costs during life-cycle of implemented the Army DFS Portal solution architecture – to include training of new personnel unfamiliar with the various technologies

LoB = Left of Boom; RoB = Right of Boom

As alluded earlier, these metrics and measures can be separated into those that apply before the Army DFS Portal is implemented (left of boom) and those that would ensure its

operation after it reaches its full operating capability status (right of boom). The utilization of graph database technology to power the portal in turn supports the Army VAUTI data strategy, as discussed in the previous section (see Table 2-1 above).

2. Internal Process Perspective

Table 2-3 shows a set of metrics and measures that could be applied to the implementation of the planned Army DFS Portal under the internal process perspective of the balanced scorecard methodology.

Table 2-3. Metrics and Measures for the Internal Process Perspective

Internal Process Metrics and Measures		
Metric	**Timeline**	**Measure**
• Develop a timeline and schedule for upgrades applicable to the entire life-cycle of the Army DFS Portal	LoB*	• 100% coverage of envisioned system upgrades during system life-cycle
• Develop a framework for improving services and delivery of functionality within the Army DFS Portal	LoB	• Increase by 30% yearly the quality of service based on end user feedback • Reduce by 30% yearly the response time within the Army DFS Portal
• Identify processes that benefit from enhanced information visualization capabilities and establish procedures for achieving optimal information visualization within said processes	LoB	• 100% identification of processes that can benefit from enhanced information visualization • 25% reduction in processing time in workflows that adopt enhanced information visualization capabilities
• Monitor the timeline and schedule for planned upgrades	RoB*	• 10% or less deviation from the vetted timeline and schedule for any given period covered by the schedule
• Monitor end user satisfaction to ensure services are either adequate or improving	RoB	• Decrease by 30% yearly the number of negative evaluations provided by the end users of the Army DFS Portal
• Monitor work flows that leverage enhanced information visualization capabilities	RoB	• Decrease by 25% or more the processing times in workflows through the use of information visualization capabilities

LoB = Left of Boom; RoB = Right of Boom

The utilization of graph database technology to power the portal in turn supports the Army VAUTI data strategy, as discussed in the previous section (see Table 2-1 above).

3. Organizational Perspective

Table 2-4 shows a set of metrics and measures that could be applied to the implementation of the planned Army DFS Portal under the organizational perspective of the balanced scorecard methodology.

Table 2-4. Metrics and Measures for the Organizational Perspective

Organizational Metrics and Measures		
Metric	**Timeline**	**Measure**
• Develop a timeline and schedule for personnel training applicable to the entire life-cycle of the Army DFS Portal	LoB*	• 100% coverage of all personnel training required for the adequate operation through the entire life cycle of the Army DFS Portal
• Develop a timeline and schedule for gap analysis of system component effectiveness applicable to the entire life-cycle of the Army DFS Portal	LoB	• 100% coverage of information subsystems comprising the Army DFS Portal infrastructure
• Develop a timeline and schedule for reviewing the organizational climate (e.g., potential cultural barriers) that encompass the entire life-cycle of the Army DFS Portal	LoB	• 100% coverage of workflows in the Army DFS Portal where personnel may be prone to resist changes
• Monitor compliance with vetted timeline and schedule for personnel training encompassing the entire life-cycle of the Army DFS Portal	RoB*	• 10% or less deviation from the vetted timeline and schedule for any given period covered by the schedule
• Monitor compliance with vetted timeline and schedule for identifying gaps in the system component effectiveness encompassing the entire life-cycle of the Army DFS Portal	RoB	• 10% or less deviation from the vetted timeline and schedule for any given period covered by the schedule
• Monitor compliance with vetted timeline and schedule for identifying cultural barriers and organizational climate within workflows that encompass the entire life-cycle of the Army DFS Portal	RoB	• 10% or less deviation from the vetted timeline and schedule for any given period covered by the schedule

LoB = Left of Boom; RoB = Right of Boom

The utilization of graph database technology to power the portal in turn supports the Army VAUTI data strategy, as discussed in the previous section (see Table 2-1 above).

4. End User Perspective

Table 2-5 shows a set of metrics and measures that could be applied to the implementation of the planned Army DFS Portal under the end user perspective of the balanced scorecard methodology.

Table 2-5. Metrics and Measures for the End User Perspective

End User Metrics and Measures		
Metric	**Timeline**	**Measure**
• Conduct a risk analysis and identify countermeasures to ensure infrastructure integrity with respect to cyber-attacks	LoB*	• 100% coverage of attack surface offered by the selected solution architecture and the countermeasures needed to protect the network and systems infrastructure
• Develop a process to capture end user feedback	LoB	• 100% capture and review of end user feedback with respect to response times, overall quality of service, ease of use, services offered, etc.
• Develop a process to capture operator performance	LoB	• 100% capture and review of how operators interact with the Army DFS Portal during the conduct of their work
• Monitor adequacy of the selected countermeasures regarding their ability to protect the infrastructure integrity and review nature and severity of emerging threats	RoB*	• 10% or less downtime for the Army DFS Portal due to infrastructure compromise caused by cyber attacks • Zero exfiltration incidents of encrypted and appropriately pseudonymized data
• Monitor end user satisfaction	RoB	• Decrease by 30% yearly the number of negative evaluations provided by the end users of the Army DFS Portal
• Monitor the Army DFS Portal operator proficiency	RoB	• Decrease by 30% yearly the number instances in which inadequate service is provided by operators due to subpar performance

LoB = Left of Boom; RoB = Right of Boom

The utilization of graph database technology to power the portal in turn supports the Army VAUTI data strategy, as discussed in the previous section (see Table 2-1 above).

3. Conclusions and Recommendations

A. Conclusions

Based on the analytical work performed during this phase, the IDA team concluded the following:

- As briefly noted in the previous deliverables, the main risk associated with the adoption of graph databases when compared to relational data stores in the context of massive graphs, is their inferior time performance associated with data retrieval and complex query execution. For interactive applications, any data storage and retrieval technology that requires more than one or two seconds to deliver the answer is unlikely to be a strong contender in the solution architecture that supports those use cases.
- However, some proprietary graph database solutions for "big data" are reaching a sufficient level of maturity to be competitive with relational data stores in terms of performance. Specifically, the combination of graph databases and frameworks for distributed storage and processing, such as Apache Hadoop and Apache Spark, make it possible to efficiently partition very large datasets to compensate for any slowdowns caused by the size of the graphs.
- The use of a "semantic layer" (i.e., metadata that characterizes a record expressed in the form of an RDF triple) can be readily implemented using both RDF statements and alternative data representations that are not only closely related to the graph formalism – and, therefore, can be readily converted back and forth – but can also be directly processed using a programming language (e.g., Prolog).
- The key rationale for using graph databases is mainly to enable the cost-effective handling of legacy data, bypassing the laborious and expensive extraction, transformation and loading (ETL) associated with traditional approaches, and said rationale is supported by all the findings obtained so far.

B. Recommendations

For this stage of the study, the preliminary recommendations are as follows:

- Continue the evaluation of available graph database implementations, both proprietary and open source, and expand the scope to include other promising NoSQL choices.

- Conduct additional comparisons regarding the use of other programming languages and data representations for the purpose of implementing a “semantic layer” as part of the graph database solution.
- Explore applicable emerging “big data” solutions with regard to their applicability in a future implementation of the Army DFS Portal.

Appendix A
Alternate Representations of Graphs and Programmatic Manipulation via Prolog

1. Introduction

As noted in the main body of this document, an essential component of the graph database implementation is the definition and use of all the metadata necessary to ensure that the information collected is understandable and easily retrievable. Another key facet of the implementation is the ability to efficiently perform the necessary operations to update and maintain the data. Although RDF triples are excellent for re-expressing records that reside in relational data stores, other representations are ideally suited for specific solutions (e.g., JSON LD serializations for web-based applications) or representations that lend themselves to programmatic manipulation (e.g., Prolog knowledge bases).

In this appendix we briefly discuss the ability to convert between RDF and JSON LD serializations and Prolog knowledge bases. It should be noted that commercial solutions such as Allegro Graph already support the use of Prolog for data manipulation and retrieval as an alternative to SPARQL queries.

2. Converting RDF to JSON LD and Back

Figure A-1 shows the first 10 records of a notional **Person** table. As shown therein, each record contains a unique key, as well as fields for the first name, the last name, the date of birth, and the key of another instance of **Person** related via the "**knows**" predicate.

PERSON

perID	fname	lname	dob	knows
PER1010000005	BRADLEY	TSYE259721232	2156-2-24	PER1090000005
PER1010000010	REBA	SZJP793046932	2126-1-18	PER1090000010
PER1010000015	LOREEN	VYUQ154107420	2118-3-21	PER1090000015
PER1010000020	TOMAS	VXZB737172133	2143-7-5	PER1090000020
PER1010000025	RICKIE	PQRE343720763	2139-8-24	PER1090000025
PER1010000030	TYLER	LZTD516422263	2179-8-4	PER1090000020
PER1010000035	SHELDON	BFDX464616044	2144-9-9	PER1090000035
PER1010000040	KANDACE	JILX582426703	2116-10-22	PER1090000040
PER1010000045	JENNINE	HCCI710228113	2151-1-30	PER1090000045
PER1010000050	TANJA	KFJA676788231	2129-10-25	PER1090000050

Figure A-1. Snippet of the Notional Person Table

Each of those records in the **Person** table can be readily converted into RDF triples. Figure A-2 shows the RDF equivalent for the first two records (see Figure A-1 above), using the Turtle serialization.

```
@prefix ns1: <http://usa/graphportal/resources/personnel#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ns1:PER1010000005 a ns1:BradleyPerson ;
   ns1:dob "2156-2-24"^^xsd:string ;
   ns1:fname "BRADLEY"^^xsd:string ;
   ns1:knows ns1:PER1090000005 ;
   ns1:lname "TSYE259721232"^^xsd:string .

ns1:PER1010000010 a ns1:RebaPerson ;
   ns1:dob "2126-1-18"^^xsd:string ;
   ns1:fname "REBA"^^xsd:string ;
   ns1:knows ns1:PER1090000010 ;
   ns1:lname "SZJP793046932"^^xsd:string .
```

Figure A-2. Representation of the Person Records in RDF Using Turtle Serialization

```
[ {
  "@id" : "http://usa/graphportal/resources/personnel#PER1010000005",
  "@type" : [ "http://usa/graphportal/resources/personnel#BradleyPerson" ],
  "http://usa/graphportal/resources/personnel#dob" : [ {
    "@value" : "2156-2-24"
  } ],
  "http://usa/graphportal/resources/personnel#fname" : [ {
    "@value" : "BRADLEY"
  } ],
  "http://usa/graphportal/resources/personnel#knows" : [ {
    "@id" : "http://usa/graphportal/resources/personnel#PER1090000005"
  } ],
  "http://usa/graphportal/resources/personnel#lname" : [ {
    "@value" : "TSYE259721232"
  } ]
}, {
  "@id" : "http://usa/graphportal/resources/personnel#PER1010000010",
  "@type" : [ "http://usa/graphportal/resources/personnel#RebaPerson" ],
  "http://usa/graphportal/resources/personnel#dob" : [ {
    "@value" : "2137-7-21"
  } ],
  "http://usa/graphportal/resources/personnel#fname" : [ {
    "@value" : "REBA"
  } ],
  "http://usa/graphportal/resources/personnel#knows" : [ {
    "@id" : "http://usa/graphportal/resources/personnel#PER1090000010"
  } ],
  "http://usa/graphportal/resources/personnel#lname" : [ {
    "@value" : "SZJP793046932"
  } ]
```

Figure A-3. Example of JSON LD Serialization of RDF Triples

Figure A-3 shows the corresponding JSON LD serialization of the RDF triples expressed in Turtle (see Figure A-2), obtained using the open source Java tool **rdfconvert**.[20] The serialization not only captures the key-value pairs for the first name, last name, date of birth, and the known instance of **Person**, but it also reflects the namespaces used in the RDF representation. The **rdfconvert** tool also supports the conversion of the JSON LD serializations back into RDF (not shown here).

The **rdfconvert** tool can take as input a file serialized in any of the following formats: RDF/XML, N-Quads, N-Triples, Turtle, TriG, TriX, RDF/JSON, JSON-LD, or BinaryRDF. The output of the **rdfconvert** tool can be in any of the following serializations: RDF/XML (the default value), N-Quads, N-Triples, N3, Turtle, TriG, TriX, RDF/JSON, JSON-LD, and BinaryRDF.

As schematically depicted in Figure A-4, the above demonstration means that records contained in an RDF triple store can be easily converted and passed to a web application in the form of a data stream that uses the JSON-LD serialization, and conversely, that data captured by a web application and transferred as a JSON-LD data stream can be efficiently converted into a Turtle file before ingestion into the RDF triple store, allowing a solution architecture to leverage the optimized capabilities of each of the components with relatively minimal effort.

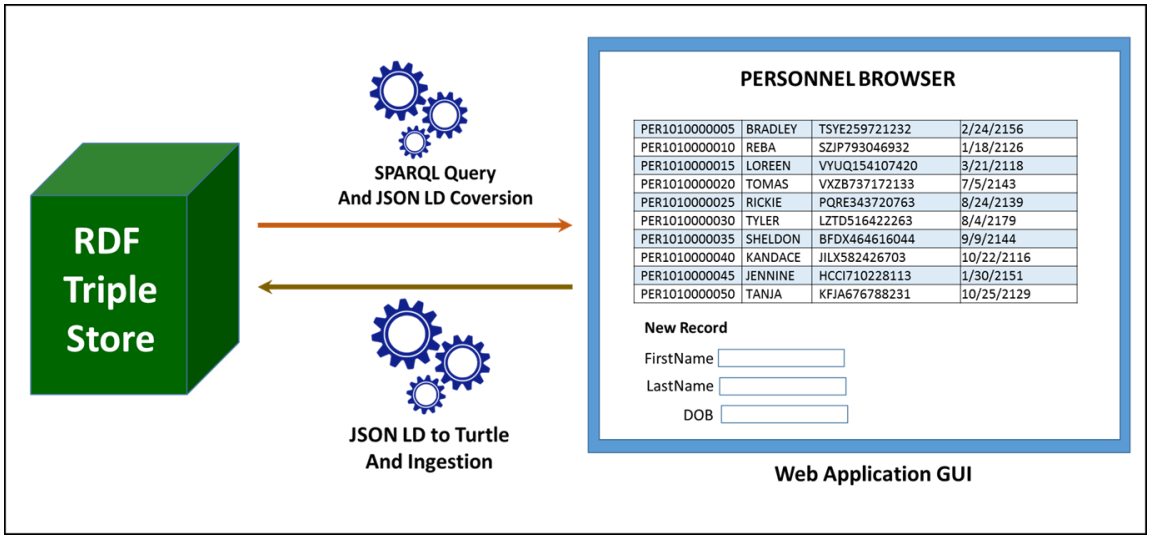

Figure A-4. Depiction of a Notional Round Trip Using JSON LD and Turtle Serializations

3. Representation of RDF Data Using Prolog Knowledge Bases

The representation of records in the form of RDF triples is very similar to the way in which Prolog knowledge bases (PKBs) are written. Figure A-5 shows how the RDF triples

[20] https://sourceforge.net/projects/rdfconvert/

corresponding to the *first name* field in the **Person** table (see Figure A-1 above) can be re-expressed as a PKB.

```
fname(p1010000005,'BRADLEY') .
fname(p1010000010,'REBA') .
fname(p1010000015,'LOREEN') .
fname(p1010000020,'TOMAS') .
fname(p1010000025,'RICKIE') .
fname(p1010000030,'TYLER') .
fname(p1010000035,'SHELDON') .
fname(p1010000040,'KANDACE') .
fname(p1010000045,'JENNINE') .
fname(p1010000050,'TANJA') .
```

Figure A-5. A Portion of the Prolog Knowledge Base Corresponding to the RDF Triples Containing the "fname" Predicate

Similarly the RDF triples that express the association of one instance of **Person** to another instance of **Person** under the predicate "**knows**" can be re-expressed as a PKB in the manner shown in Figure A-6.

```
knows(p1010000005,p1090000005) .
knows(p1010000010,p1090000010) .
knows(p1010000015,p1090000015) .
knows(p1010000020,p1090000020) .
knows(p1010000025,p1090000025) .
knows(p1010000030,p1090000030) .
knows(p1010000035,p1090000035) .
knows(p1010000040,p1090000040) .
knows(p1010000045,p1090000045) .
knows(p1010000050,p1090000050) .
```

Figure A-6. A Portion of the Prolog Knowledge Base Corresponding to the RDF Triples Containing the "knows" Predicate

4. Programmatic Manipulation with Prolog – "Six Degrees of Separation" Redux

The use of PKBs as an alternate representation of the graphs contained in an RDF triple store raises the question of whether or not one can improve the data retrieval

performance of expensive queries by judiciously encoding the statements that make the PKBs. Specifically, can the performance of queries, such as the ones used in the Six Degrees of Separation (SDOS) test case (discussed in the second deliverable), be substantially improved by writing the PKBs in a way that preserves the order implicit in the structure of the SDOS graph, namely, a chain made of the assertions "person T1 knows person T2," "person T2 knows person T3," and so on? (See Figure A-7).

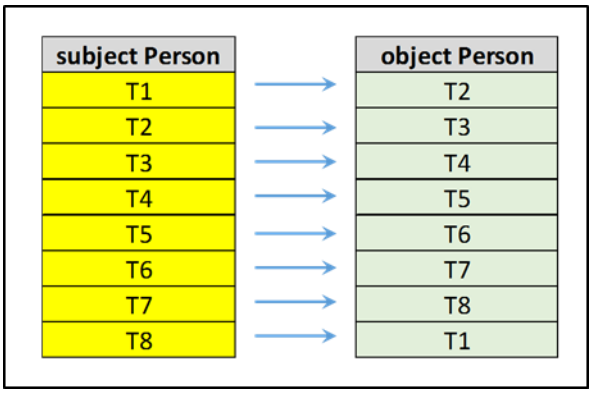

Figure A-7. Schematic Depiction of the Graph Structure for the Six Degrees of Separation (SDOS)

If the PKB is built by directly reading the contents of the **PersonAssociation** table, which is sorted by the key of the **subject Person**, the assertions are written as shown in Figure A-6. In that case after finding that **p1010000005** ***knows*** **p1090000005** the application has to read potentially millions of assertions before finding the assertion that links **p1090000005** to the next instance of **Person**.

If, however, the PKB is written so that the assertions that make up the eight links in each of the subgraphs, as implied by the structure shown in Figure A-7, then once the **subject Person** is found, the application needs only to read at most eight additional assertions to traverse the entire subgraph that begins with that instance of **Person** (See Figure A-8).

```
knows(p1000000005,p1080000005) .
knows(p1080000005,p1160000005) .
knows(p1160000005,p1240000005) .
knows(p1240000005,p1320000005) .
knows(p1320000005,p1400000005) .
knows(p1400000005,p1480000005) .
knows(p1480000005,p1560000005) .
knows(p1560000005,p1000000005) .
```

Figure A-8. Assertions in the PKB Sorted According to the Structure of the Six Degrees of Separation Graph

With the PKB written in this fashion one can then execute Prolog queries that find the instances of **Person** with **fname** = 'TAYLOR' that are separated from instances of **Person** with **fname** = 'DORIAN' by up to six steps (See Figure A-9).

```
sdos01(L) :-
findall(Y,(knows(X,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN')),L) .

sdos02(L) :-
findall(Y,(knows(X,B1),knows(B1,Y),fname(X,'TAYLOR'),
fname(Y,'DORIAN')),L) .

sdos03(L) :- findall(Y,(knows(X,B1),knows(B1,B2),knows(B2,Y),
fname(X,'TAYLOR'),fname(Y,'DORIAN')),L) .

sdos04(L) :-
findall(Y,(knows(X,B1),knows(B1,B2),knows(B2,B3),
knows(B3,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN')),L) .

sdos05(L) :-
findall(Y,(knows(X,B1),knows(B1,B2),knows(B2,B3),knows(B3,B
4),knows(B4,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN')),L) .

sdos06(L) :-
findall(Y,(knows(X,B1),knows(B1,B2),knows(B2,B3),knows(B3,B
4),knows(B4,B5),knows(B5,Y),fname(X,'TAYLOR'),
fname(Y,'DORIAN')),L) .
```

Figure A-9. Prolog Queries for the Six Degrees of Separation Test Case

The execution times using the open source implementation of Prolog SWIPL, and using a PKB containing eight million assertions with the predicate **fname**, and a second PKB containing eight million assertions with the predicate **knows** are shown in Figure A-10.

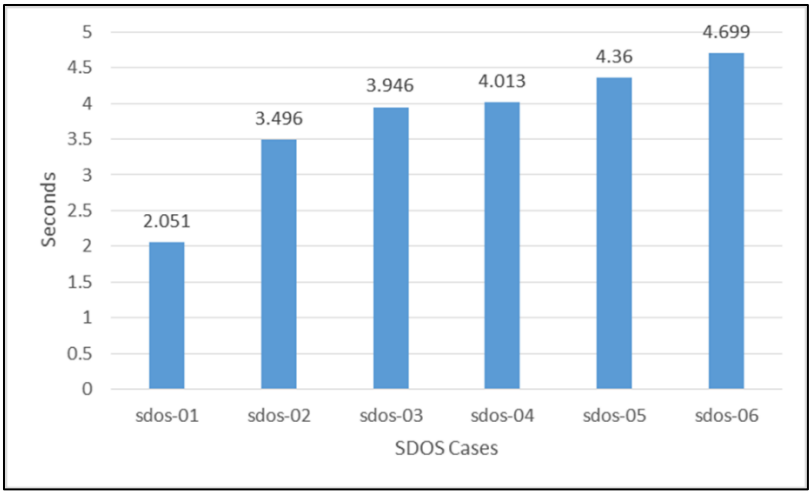

Figure A-10. Performance Results for the SDOS Test Case Using Sorted Assertions

These results suggest that the way in which the PKBs are constructed may have a big impact on the performance of the queries. In particular, the execution times for queries that require testing three or more steps in each subgraph level off (i.e., the execution time only differs by about one second between the shortest and the longest query).

Additional exploration of the effects on performance when using Prolog predicates that have a higher arity is recommended to find out whether this representation of the graphs contained in an RDF triple store further reduces the data retrieval times.

5. Mixing Prolog and Python

As a declarative programming language Prolog makes the formulation of queries to retrieve data from graphs relatively simple. For example the queries in Figure A-9 for the SDOS test case are much simpler than the SQL counterparts needed to retrieve the same results from the tables **Person** and **PersonAssociation** in a relational database such as MySQL.

On the other hand, although some proprietary implementations of Prolog such as SICStus[21] offer support for database manipulation and GUI development for stand-alone applications, the manipulation of strings in Prolog is more complicated than in Python.

Fortunately, libraries such as PySWIP, described as "a Python - SWI-Prolog bridge enabling to query SWI-Prolog in your Python programs," that allow the possibility of having Prolog-like capabilities inside a standard Python script.[22]

[21] https://sicstus.sics.se/

[22] https://github.com/yuce/pyswip

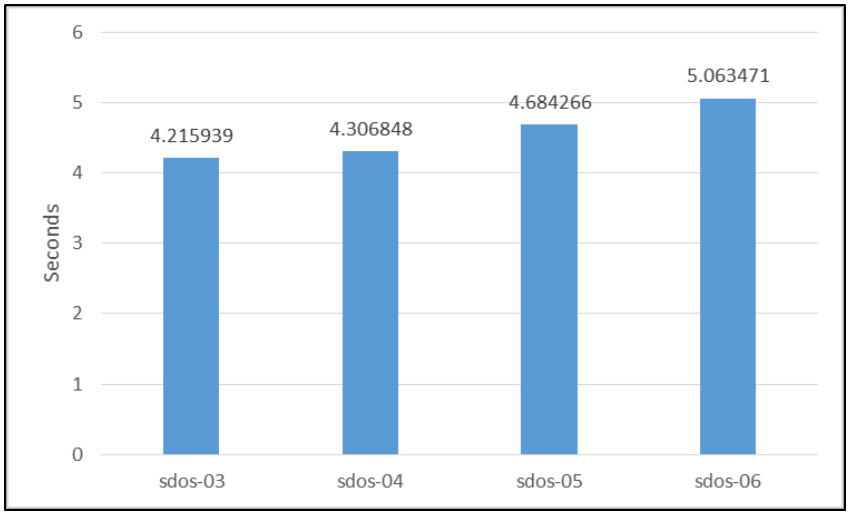

Figure A-11. Performance Results for the SDOS Test Case Using the PySWIP Bridge

Although performance degrades somewhat when using PySWIP as opposed to running the SDOS queries directly in Prolog SWIPL (see **Error! Reference source not ound.**Figure A-11), the availability of the rich set of Python string manipulation functions makes it relatively straightforward to format the results of the queries in a user-friendly manner (see Figure A-12**Error! Reference source not found.**).

```
+-------------------------------------+
|               SDOS-4                |
+--------------------+----------------+
|    TaylorID        |      DorianID  |
+--------------------+----------------+
|  p1323834270       |   p1003834270  |
|  p1163984645       |   p1483984645  |
+--------------------+----------------+

SDOS-4 query_time =  0:00:04.306848
```

Figure A-12. Sample of Formatted SDOS Query Results using Python

Appendix B
Sample Code Used for Testing Conversion of Legacy Relational Data to RDF Triples

The code examples included in this section are provided primarily to facilitate the development of assessment tests similar to those described in this document for graph database manipulation using Prolog.

To eliminate barriers to the reuse of an entire snippet or a portion thereof all the code examples are released under the MIT license shown below.[23] The Institute for Defense Analyses, however, retains the copyright of all the code contained in this appendix.

[23] https://opensource.org/licenses/MIT

A. Preparation of Prolog Knowledge Bases

The Prolog knowledge bases (PKBs) used in this deliverable were generated out of the sample data created previously and stored in the tables **Person** and **PersonAssociation** within a MySQL server.

1. Python Scripts to Generate the PKBs

The script presented below generates a PKB for the **knows** predicate with the assertions ordered consistent with the structure of the SDOS graph (see Figure A-7). The output is of the form shown in Figure A-8. The Python script traverses the first million records in the eight subsets used to create the **PersonAssociation** table (see deliverable 2 for more details).


```
# Author: Francisco Loaiza, Ph.D., J.D.
# Institute for Defense Analyses
# Alexandria, Virginia, USA
```


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

startVal = 1000000005
counter = 1

A = [ ]
B = [ ]
# Replace the values in the connection string below to reflect your configuration
```

```
con = mdb.connect('localhost', 'myuser', 'myuserpwd', 'mydatabase');

with con:

        for j in range(1000000):

        cur = con.cursor()

       # ---------- first record ----------

       # print "First Record"

        sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(startVal)
       # print sqlStr

        cur.execute(sqlStr)
        row = cur.fetchone()
        subj = int(row[0])
        obj = int(row[1])

        prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

        print prologStr

       # cur.execute(sqlStr)

       # counter = counter + 1

       # ---------- second record ----------

       # print "Second Record"

        sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)
       # print sqlStr

        cur.execute(sqlStr)
        row = cur.fetchone()
        subj = int(row[0])
        obj = int(row[1])

        prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

        print prologStr

       # cur.execute(sqlStr)

       # counter = counter + 1

       # ---------- third record ----------
```

```
# print "Third Record"

 sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)
# print sqlStr

 cur.execute(sqlStr)
 row = cur.fetchone()
 subj = int(row[0])
 obj = int(row[1])

 prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

 print prologStr

# cur.execute(sqlStr)

# counter = counter + 1

# ---------- fourth record ----------

# print "Fourth Record"

 sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)
# print sqlStr

 cur.execute(sqlStr)
 row = cur.fetchone()
 subj = int(row[0])
 obj = int(row[1])

 prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

 print prologStr

# cur.execute(sqlStr)

# counter = counter + 1

# ---------- fifth record ----------

# print "Fifth Record"

 sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)
# print sqlStr

 cur.execute(sqlStr)
 row = cur.fetchone()
 subj = int(row[0])
 obj = int(row[1])

 prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."
```

```
 print prologStr

# cur.execute(sqlStr)

# counter = counter + 1

# ---------- sixth record ----------

# print "Sixth Record"

 sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)
# print sqlStr

 cur.execute(sqlStr)
 row = cur.fetchone()
 subj = int(row[0])
 obj = int(row[1])

 prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

print prologStr

# cur.execute(sqlStr)

# counter = counter + 1

# ---------- seventh record ----------

# print "Seventh Record"

sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)

# print sqlStr

cur.execute(sqlStr)
row = cur.fetchone()
subj = int(row[0])
obj = int(row[1])

prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

print prologStr

# cur.execute(sqlStr)

# counter = counter + 1

# ---------- eigth record ----------

# print "Eigth Record"
```

```
        sqlStr = "SELECT subjperID,objperID FROM per08aper08aAssn WHERE subjperID =" + str(obj)

        # print sqlStr

        cur.execute(sqlStr)
        row = cur.fetchone()
        subj = int(row[0])
        obj = int(row[1])

        prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

        print prologStr

        # cur.execute(sqlStr)

        # counter = counter + 1

        # ----------------------------------------------------

        # con.commit()

        startVal = startVal + 5

con.close()
```

The previous script can be substantially simplified if the **PersonAssociation** table is first sorted as shown in the figure below.

Person Association

paID	subjperID	objperID
1	1000000005	1080000005
2	1080000005	1160000005
3	1160000005	1240000005
4	1240000005	1320000005
5	1320000005	1400000005
6	1400000005	1480000005
7	1480000005	1560000005
8	1560000005	1000000005
9	1000000010	1080000010
10	1080000010	1160000010

Note that in MySQL one needs to add a new key attribute (e.g., **paID**) to capture the desired order of the records in that table. Otherwise, the value in the column **subjperID** will be used to sort the records.

```
# Copyright ©2017. The Institute for Defense Analyses (IDA).
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
# associated documentation files (the "Software"), to deal in the Software without restriction,
# including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do
 # so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all copies or substantial
# portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
# SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE
# BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT,
# TORT OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH,
# THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.#

# Author: Francisco Loaiza, Ph.D., J.D.
# Institute for Defense Analyses
# Alexandria, Virginia, USA

#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

counter = 1

# Replace the values in the connection string below to reflect your configuration

con = mdb.connect('localhost', 'myuser', 'myuserpwd', 'mydatabase');

with con:

    cur = con.cursor()

# ---------- retrieve all records ----------

    sqlStr = "SELECT * FROM persAssn"
    cur.execute(sqlStr)
    rows = cur.fetchall()

    for row in rows:

        subj = int(row[1])
        obj  = int(row[2])

        prologStr = "knows(p" + str(subj) + "," + "p" + str(obj) + ") ."

        print prologStr
```

B. Scripts for PySWIP

1. The SDOS Test Case

The Python script shown below executes the Prolog SDOS queries using PySWIP and then produces formatted output as shown in Figure A-12**Error! Reference source not ound.**. The code is not optimized, i.e., code that repeats has not been refactored as a function or method that can be called subsequently by the other portions of the code.

```
# Copyright ©2017. The Institute for Defense Analyses (IDA).
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
# associated documentation files (the "Software"), to deal in the Software without restriction,
# including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do
 # so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all copies or substantial
# portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
# SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE
# BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT,
# TORT OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH,
# THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.#

# Author: Francisco Loaiza, Ph.D., J.D.
# Institute for Defense Analyses
# Alexandria, Virginia, USA

# -*- coding: utf-8 -*-

from pyswip.prolog import Prolog
from pyswip import *
from datetime import datetime

start_time = datetime.now()

X = Variable()
Y = Variable()
B1 = Variable()
B2 = Variable()
B3 = Variable()
B4 = Variable()
B5 = Variable()

prolog = Prolog()

prolog.consult("kb_sdos_1M.pl")
```

```
stop_time = datetime.now()

print "\nTime to load the knowledgebase = " , stop_time - start_time

start_time = datetime.now()

assertz = Functor("assertz")
fname = Functor("fname", 2)
lname = Functor("lname", 2)
dob = Functor("dob", 2)
knows = Functor("knows",2)

# ************************************************

print "+---------------------------------------+"
print "| SDOS-1 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

q = Query(knows(X,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN'))

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()

print "SDOS-1 query_time = " , stop_time - start_time

# ************************************************

start_time = datetime.now()

q = Query(knows(X,B1),knows(B1,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN'))

print "+---------------------------------------+"
print "| SDOS-2 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()
```

```
print "SDOS-2 query_time = " , stop_time - start_time

# ***************************************************

start_time = datetime.now()

q = Query(knows(X,B1),knows(B1,B2),knows(B2,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN'))

print "+---------------------------------------+"
print "| SDOS-3 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()

print "SDOS-3 query_time = " , stop_time - start_time

# ***************************************************

start_time = datetime.now()

q = Query(knows(X,B1),knows(B1,B2),knows(B2,B3),knows(B3,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN'))

print "+---------------------------------------+"
print "| SDOS-4 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()

print "SDOS-4 query_time = " , stop_time - start_time

# ***************************************************

start_time = datetime.now()

q =
Query(knows(X,B1),knows(B1,B2),knows(B2,B3),knows(B3,B4),knows(B4,Y),fname(X,'TAYLOR'),fname(Y,'DORIAN'))
```

```
print "+---------------------------------------+"
print "| SDOS-5 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()

print "SDOS-5 query_time = " , stop_time - start_time

# ***************************************************

start_time = datetime.now()

q =
Query(knows(X,B1),knows(B1,B2),knows(B2,B3),knows(B3,B4),knows(B4,B5),knows(B5,Y),fname(X,'TAYLOR'),fnam
e(Y,'DORIAN'))




print "+---------------------------------------+"
print "| SDOS-6 |"
print "+-------------------+-------------------+"
print "| TaylorID | DorianID |"
print "+-------------------+-------------------+"

while q.nextSolution():
 a = str(X.value)
 b = str(Y.value)
 print "| ",a.ljust(16),"| ",b.ljust(16),"|"
q.closeQuery()

print "+-------------------+-------------------+\n"

stop_time = datetime.now()

print "SDOS-6 query_time = " , stop_time - start_time
```

References

Although graph database technologies are young as compared to their relational database counterpart, a growing secondary literature is readily available. The following are some of the most recent offerings. The URLs point to Amazon.com where the items can be purchased.

Graph Databases: New Opportunities for Connected Data 2nd Edition, by Ian Robinson, Jim Webber, and Emil Eifrem, published by O'Reilly Media; 2 edition (July 9, 2015).
https://www.amazon.com/Graph-Databases-Opportunities-Connected-Data/dp/1491930896/ref=cm_cr_arp_d_product_top?ie=UTF8

Neo4j in Action 1st Edition, by Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, and Jonas Partner, published by Manning Publications; 1 edition (December 21, 2014).
https://www.amazon.com/Neo4j-Action-Aleksa-Vukotic/dp/1617290769/ref=cm_cr_arp_d_product_top?ie=UTF8

Linked Data: Structured Data on the Web 1st Edition, by David Wood, Marsha Zaidman, Luke Ruth, and Michael Hausenblas, published by Manning Publications; 1 edition (January 24, 2014).
https://www.amazon.com/Linked-Data-David-Wood/dp/1617290394/ref=sr_1_2?s=books&ie=UTF8&qid=1474990762&sr=1-2

Linked Data for Libraries, Archives and Museums: How to Clean, Link and Publish your Metadata, by Seth van Hooland (Author), Ruben Verborgh, published by Amer Library Assn Editions (June 25, 2014).
https://www.amazon.com/Linked-Data-Libraries-Archives-Museums/dp/0838912516/ref=sr_1_1?s=books&ie=UTF8&qid=1474991823&sr=1-1

The Great Cloud Migration: Your Roadmap to Cloud Computing, Big Data and Linked Data, by Michael C. Daconta, published by Outskirts Press (October 11, 2013).
https://www.amazon.com/Great-Cloud-Migration-Roadmap-Computing/dp/147872255X/ref=sr_1_1?s=books&ie=UTF8&qid=1474992107&sr=1-1

Information as Product, by Michael C. Daconta, published by Outskirts Press (October 21, 2007).
https://www.amazon.com/Information-as-Product-Michael-Daconta/dp/1432716549/ref=sr_1_2?s=books&ie=UTF8&qid=1474992167&sr=1-2

The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management 1st Edition, by Michael C. Daconta (Author), Leo J. Obrst (Author),

Kevin T. Smith, published by Wiley; 1 edition (May 30, 2003). https://www.amazon.com/Semantic-Web-Services-Knowledge-Management/dp/0471432571/ref=sr_1_5?s=books&ie=UTF8&qid=1474992283&sr=1-5

Joe Celko's Complete Guide to NoSQL: What Every SQL Professional Needs to Know about Non-Relational Databases 1st Edition, by Joe Celko, published by Morgan Kaufmann; 1 edition (October 7, 2013). https://www.amazon.com/Celkos-Complete-Guide-NoSQL-Non-Relational-ebook/dp/B00G4N7HPS/ref=dp_kinw_strp_1

Acronyms and Abbreviations

AI	artificial intelligence
API	Application Program Interface
AQL	ArangoDB Query Language
AWS	Amazon Web Services
CDS	Cross-domain solution
CRUD	Create, Retrieve, Update, Delete
CSV	Comma Separated Values
DDL	data definition language
DFS	Dynamic Force Structure
DML	data manipulation language
DoD	Department of Defense
DSE	DataStax Enterprise
ETL	Extraction, Transformation and Loading
GFM DI	Global Force Management Data Initiative
GUI	Graphic User Interface
IDA	Institute for Defense Analyses
IRC	Internet Relay Chat
JVM	Java virtual machine
LINQ	Language Integrated Query
MQL	Metaweb Query Language
MTO&E	Modified Table of Organization and Equipment
NoSQL	Not only Structured Query Language
OWL	Web Ontology Language

PII	personally identifiable information
PKB	Prolog Knowledge Base
RDF	Resource Description Framework
ReST	Representational State Transfer
SaaS	software as a service
SPARQL	A recursive acronym for SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
TB	Terabyte
TDA	Table of Distributions and Allowances
TSL	Trinity Specification Language
XML	Extensible Markup Language

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YY)	2. REPORT TYPE	3. DATES COVERED (From – To)
29-12-17	Non-Standard	

4. TITLE AND SUBTITLE

Assessment of Graph Databases as a Viable Materiel Solution for the Army's Dynamic Force Structure (DFS) Portal Implementation: Part 3, Risks, Mitigation Approach, and Roadmap

5a. CONTRACT NUMBER

HQ0034-14-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBERS

6. AUTHOR(S)

Francisco L. Loaiza-Lemos, Russell J. Smith

5d. PROJECT NUMBER

BC-5-4277

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

D-8852
H 2017-000092

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Mr. Bruce Haberkamp
Army CIO/G-6 (SAIS-AOD)
5850 23rd Street, Bldg. 220, Ft. Belvoir, Virginia 20060-5832

10. SPONSOR'S / MONITOR'S ACRONYM

CIO/G-6 (SAIS-AOD)

11. SPONSOR'S / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

Project Leader: Francisco L. Loaiza-Lemos

14. ABSTRACT

This document identifies technical risks together with suitable mitigation approaches associated with the use of graph database implementations in the planned Army DFS Portal. The document also describes possible technology insertion milestones that may be adopted to enhance the effectiveness of graph database adoption. Rapid prototyping techniques have been applied, as part of the continuing evaluation of alternatives, to stress the implementations of the graph databases chosen for the study. Data collected during those activities will be used to continue maturing the decision process needed to determine the best-of-breed options. The assessments leverage the metrics elaborated in previous reports provided to the sponsor.

15. SUBJECT TERMS

Graph databases, Resource Description Framework (RDF), RDF triples store, JavaScript Object Notation (JSON), JSON Linked Data (JSON LD), Prolog, data lake, data swamp, data governance, data quality, risk management framework, risk mitigation, cybersecurity, balanced scorecard methodology, National Institute of Standards and Technology (NIST), NIST Special Publication 800-series reports.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mr. Bruce Haberkamp
Unclassified	Unclassified	Unclassified	Unlimited	56	19b. TELEPHONE NUMBER (Include Area Code) 703-545-1464