Available online at http://www.journalcra.com

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

International Journal of Current Research
Vol.3, Issue, 4, pp.015-020, April, 2011

ISSN: 0975-833X

REVIEW ARTICLE

RESPOSITORY INTEGRATION: THE DISCONNECT AND WAY FORWARD THROUGH REPOSITORY VIRTUALIZATION SUPPORTING BUSINESS INTELLIGENCE

***Bernard Shibwabo Kasamani and Ismail Ateya Lukandu**

Faculty of Information Technology, Strathmore University, Kenya

ARTICLE INFO

Article History:

Received 8th February, 2011
Received in revised form
18th March, 2011
Accepted 25th March, 2011
Published online 17th April 2011

Key words:

Integration,
Repository,
Coherence.

ABSTRACT

This paper proposes a presentation-oriented virtualized approach that supports a modern process to merge existing data islands in organizations around the world that results to a merged virtual data repository. The authors explore a variety of data sources and finally present a uniform solution for the common cases. The solution consists of a transparent virtual repository that supports informational intelligence components. By using this approach we not only mask the differences in divergent data repositories but also provide a standard way to access enterprise-wide data. Security is provided through a two-tier mechanism. The result is a single database that percolates and draws data residing in incoherent repositories alongside an intelligence layer. In real-time, there is a determination of the available data repositories. In conclusion, we expect users to exploit this platform and be free to filter information based on their requirements so as to come up with an informed judgment.

INTRODUCTION

The overall observation supports the fact that large volumes of data are continuously being stored in data repositories around the world. With data storage cost per megabit reducing, data should even in the present be stored at a higher rate than ever before as found by Parsaye and Chignell (1995). The potential of unstructured data can be realized by converting Semi-structured or Unstructured data into a more usable, structured form. Entities globally are generally faced with the challenge of developing effective and efficient methodologies for transforming the myriad data forms into structured data as Anandarajan *et al.* (2004) describe. Anandarajan *et al.* (2004), continue to explain that the ability to make effective decisions is crucial to an organizations survival in today's tumultuous business environment. In order for firms to evaluate alternatives and make informed choices they must have reliable and timely data upon which to make their decisions. Liautaud and Hammond (2001) found that the past decade brought a small revolution in the way companies conducted their business. Packaged applications from SAP, PeopleSoft, Oracle and the others were embraced

***Corresponding author:** bshibwabo@strathmore.edu*

as a means of improving operational efficiency. Liautaud and Hammond (2001) continue to explain that the acquired systems increased the productivity of operation and internal processes which, in some cases, had remained unchanged for decades. However the packaged applications introduced a major challenge for IT management – managing the distributed data silos that emerged.

RELATED CONTRIBUTIONS MADE

Background

We adopt the definition of enterprise application integration as "the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise" (Boiling *et al.*, 2008). The point of diversion when embracing Enterprise Application Integration is facilitating sharing of data and processes without applying extensive changes to the existing application and data structures. We will make use of the commonality existing in data sources. We have repository commonality and data commonality. Data commonality only exists if the real world facts that the data repositories represent contains some common elements. It seems not achievable or beneficial to go forward and integrate a database of a space ship with that of a hardware shop. However, integrating a payroll database with a human resource database could well be achievable with sense. This does not imply or conclude that it is infeasible to access unrelated data from a central repository. (Shibwabo and Ateya, 2010).

The Point-to-Point Approach

This approach advocates the deployment of programmers to write low-level communication code between two applications to exchange messages and data. However, such an approach leads to applications spaghetti, which often increases the complexity of the integration solution as the number of interconnected applications rises. This may introduce more problems than solutions. Themistocleous, Irani and Sharif (2000) suggest that for x applications a total of: x*(x-1)/2 connections are required, to piece together all x applications. This means that for 10 applications 45 connections are needed to interconnect these 10 applications. In addition each existing connection requires 2 interfaces (one for each interconnected application).

Web Services

Web services and related technologies are promising, but least mature, set of integration technologies. Web services are standards, protocols, and directory services that enable Web-based clinical applications to share data among themselves regardless of their underlying programming languages or platforms (Shacter, 2007). Web services use XML, which is an open standard for describing data over the Internet. In as much as they enable easier application integration, they also may act as separate mini-applications, enabling simple tasks, such as online randomization or site certification, over the Web. These technology standards allow applications to more easily share data; that's because less proprietary custom coding is used and a single transport mechanism is provided using a standard API to the application. While Web services introduces data sharing standards using XML, security models, and a common Application Program Interface (API) to link loosely coupled disparate systems, they still require integrators to modify existing IT applications and manage key data point synonyms (Shacter, 2007).

Major Weaknesses/Liabilities:

- As XML has been incorporated into many vendors' products, multiple XML standards have proliferated and multiple standards can impede easy integration if a single standard is not agreed upon.
- In addition, XML currently lacks many industry-specific definitions, as well as the security, verification, and the confirmation function necessary for inter-enterprise communication.
- XML can't achieve integration alone; it's just a piece of comprehensive Web-based integration architecture.
- Web services are ideal for asynchronous applications with low transaction volumes.

Message Bus

This approach to integration ordinarily facilitates adding new applications, but requires a common bus interface. It has been applied when you have an integration solution that consists of applications that are provided by different vendors. Because integration solutions usually involve applications that have proprietary interfaces provided by multiple vendors, Message Bus integration is difficult. All applications are connected through a logical component known as a message bus. A message bus specializes in transporting messages between applications. An application that sends a message merely passes the message to the message bus, and the message bus transports the message to all the other applications that are listening for bus messages through a shared infrastructure (Microsoft, 2010). In as much as the bus architecture has benefits such as improved modifiability (ease of adding or removing applications), Reduced application complexity (senders do not interacts with all receivers they need to send messages to), improved performance (no intermediaries between communicating applications) and improved scalability, it suffers from the following liabilities:
Major Weaknesses/Liabilities

- Increased complexity: Integrating through a message bus increases the complexity of the integration solution for the following reasons:
  - Architectural mismatch: The applications of an integration solution typically make conflicting architectural assumptions. Designing the message bus interface and solving the mismatch around the data model is a difficult endeavor.
  - Message bus access policies: Communication through a shared resource such as a message bus requires you to implement policies that ensure fair access and that resolve concurrency conflicts.
- Lowered modifiability when the bus interface breaks compatibility: Changing the message bus interface in a way that breaks compatibility typically affects all the applications that use the bus. In other words, the bus interface represents an extreme example of a published interface. Designing it requires foresight.
- Lowered integrability: All the applications that are hooked to the message bus must have the same message bus interface. Applications that have incompatible interfaces cannot use the message bus. Because the message bus interface includes a common set of command messages, message schemas, and shared infrastructure, these elements together define a common subset that may somewhat restrict the operation of the participating applications.
- Lowered security: A message bus that uses the Broadcast-Based Publish/Subscribe pattern reaches all the applications that are connected to the bus, regardless of the applications that the message is intended for. Broadcasting to all participants may not be acceptable if the messages contain sensitive data.
- Low tolerance for application unavailability: The receiver must be able to process messages when the sender passes the messages to the bus. This solution does not tolerate receiver downtime. In addition, it does not provide direct support for disconnected operation.

Semantic Approach

Semantic Integration is considered as a measure of how closely the schemas of the various data sources have been matched. In other words, how well the types, names, units, meanings, etc. of the data in the sources are matched up. At the very high end of the spectrum, all data matches up to a single agreed-upon schema. At the low end, there is no schema information at all.

The assumption by traditional data integration approaches that there exists intimate familiarity in semantics of the available data sources does not hold in practice. We don't need to have upfront data integration and should only allow for users and administrators to decide whether to invest in identifying semantic relationships or not. The major weakness of the semantic approach is that it requires full semantic integration of the sources in order to provide useful services.

ADDRESSING THE WEAKNESSSES

The Way Forward

A way forward to deal with the existing problem of divergent data islands was to create a mechanism by which an enterprise-wide view point will be prepared that communicates vision for logical data architecture for our enterprises. Techtarget (2010) explains how the Service oriented Architecture (SOA) is entrenched. Use of SOA has not dropped even as 'cloud computing' has emerged as an alternative buzz word of the day. This can be incorporated in our implementation for the model.

Reliability, availability, security and integrity of data should in no way be compromised by a better solution and therefore have to be put under consideration. This was not be the end but was to serve as a roadmap for current and future data systems development in organizations. The integration is implemented in form of a virtual database that not only serves as a virtual storage but also acts as a reliable source to data representation. This mechanism provides for a data management layer that efficiently and reliably enables federated access to a wide range of heterogeneous data sources. The virtual tables can be extended to provide for the use of arbitrary user friendly names in place of the table names in order to facilitate ordinary users to be able to use the tool. This way the organization will not be required to change anything on their databases in order to make the usage as user friendly as possible.

Proposed Approach

The proposed integration framework should provide for an effective and efficient implementation of a real world system implementation but at the same time reducing the complexity in design and code. The proposed model constitutes four layers. The first layer specifically captures the isolated repositories. This might be an Oracle, PostgreSQL, SQL Server or My SQL repository. The second layer, a virtual data store is generated that transparently links to the separate repositories. The third layer handles the rules which implement the transformations. This was implemented in SQL as well as PHP and it has its own MySQL engine that drives it. As shown in figure 1, the final layer deals with Decision making. In this case Information has to be presented in a way that facilitates or enhances decision making. The graphs as well as flexible information navigation capability are both implemented at this layer.

APPLYING THE MODEL

How It Is Accomplished

Access to the data sources requires an initial stage of setting up a connection to active data sources existing for the enterprise. This then allows for storing the connection parameters for later use only if the connection was successful. Having stored

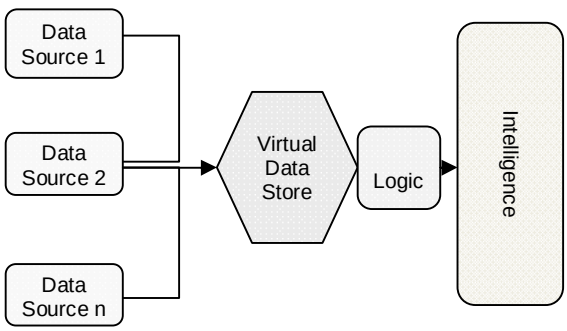

Fig. 1. The Virtualized Integration Reference Model

the connection, it is possible to either come back and describe its objects or browse the data transparently in terms of the underlying repositories. The connection information is part of the metadata that needs to be stored and used frequently. The storage location for this metadata should enable secured updates as connection information changes. In order to guarantee this a database was used. The source models represent the physical structure and characteristics of physical data repositories, whereas view models represent the structure and characteristics of abstract structures the author will want to expose to any available applications. However, the virtual database can be designed to be predictive enough to represent the queried data in a variety of the common formats of data representation that are in use. The virtual database application logic is the engine that can be written in a language that is

preferably web-based like PHP this will both facilitate global access and speedy development process by reusing existing libraries especially for reporting. The need to provide for separation or channeling of requests requires a detailed understanding of logical control structures as well as how they apply on the different repositories. An example global connection function would look like this function dt_connect($datasource, $ipaddress, $portnum, $user, $pass, $debase) { $conn=false; // Initialize the connection variable to false}. The function makes use of $datasource parameter to set the value from an array of repositories (PostgresSQL, Oracle, MySQL, SQL Server and so on), the $ipaddress parameter is used to set the location of the repository server, a $portnum parameter to set the port number of the repository being added to the virtual amalgamated repository. The $user variable sets the authentication username if necessary and the $pass variable for the password. The last parameter, $debase sets the service running on the physical repository. This can be the Service Identifier in oracle databases or the database name in PostgreSQL.

Analyzing the Drivers

PHP installations ordinarily load by default both MySQL and PostgreSQL extensions at startup in LAMP or WAMP environments. These two extensions namely *php_pgsql.dll* and *php_mysql.dll* for PostgreSQL and MySQL respectively can be used to provide the link between PHP and these two repositories from currently different vendors. Through Object Database Connectivity, we can come up with a connection object that is either created or destroyed based on the current database management system. The constructor for each of these objects will vary based on the target DBMS. An example to illustrate the variation in the class constructor is that oracle uses and SID to connect to a service while this does not make sense when using any of the other database systems which mainly refer to it as a database. Other repositories namely Oracle and SQL Server require special drivers mainly in order to deal with the fact that the application need not reside on the same server as the one that physical repositories reside. Microsoft provides its own driver for linking to PHP applications named *php_sqlsrv.dll* which was used for this purpose. For oracle DBMS, the php_oci8.dll was specifically used to achieve the same objective even through remote oracle systems.

Security Provisioning

Security is always a major concern especially now that we are dealing with an enterprise-wide or intra-enterprise data store. This can at the lower level be facilitated through encryption to the connector bindings that bind to the physical data sources. There is a great concern to propagate the restrictions effected on the source repositories. This can be achieved by defining users on the virtual repository who are authenticated by the source parameters. In addition, since this model facilitates access to information by all users, access to information by users is restricted by roles defined by administrators. The role assigned to users and the source repository connection parameters dictate the permitted level of security. This emanated from the concern that ignoring the security parameters on the physical data sources and only applying new rules could be disastrous. However, a possible alternative is to create a user with only select and connect privileges and use the user to for the repository transactions. We therefore ultimately have a two level access layers, the first one will be to the virtual data store while the second one will be used by the virtual database to connect to the real individual data sources.

SUMMARY OF FINDINGS

An important finding is that the storage server for the virtual data store needs not to have the installations of any of the database management systems. Therefore, by using the application that runs the virtual server, relevant drivers should be enough to guarantee connectivity to the repositories. In other words, this can be a solution to the nagging question pertaining why users have to for example install oracle or oracle client just to access a remote oracle database. This is truly not necessary in this age. All that a Systems administrator needs to do is to install a single application for all databases and the rest should be masked. In as much as data has and will always be ours, we need to understand that at some point we

became prisoners of a given repository consequently limiting our potential. By developing a solution that talks to these broad range of vendor repositories and which operates without constraints, the distance between these repositories has been greatly reduced. A coiled term that has been used to describe the solution is a collaborative portal which echoes the domain for this paper as well as the fact that what had been assumed to be stuck on an isolated database server can be turned into value adding information with no more troubles being experienced. The utilization of resources on the server hosting the application is predicted to grow at a rate that cannot be ignored and therefore there is nothing like any machine will do scenario. The ultimate solution to this is to suggest that there is need to use a standard way of connecting to data sources by all vendors. This will really be of benefit to the global community as well as to the vendors since no individual vendor will want to be isolated. The fact that different database management systems use different ways to connect can be one of the first technical hurdles that should be overcome. This can be sufficiently overcome by using Object Database Connectivity (ODBC) since majority of DBMSs are ODBC compliant. Existing Enterprise Applications can be integrated by providing a portal that provides the required services from one point.

CONCLUSIONS AND RECOMMENDATONS

Integration can be described to be more than just a structural or technical problem. Therefore the solution to the problem should be greater than just technical. Existing Enterprise Applications can be integrated by providing a portal supported by a virtual database that provides the required services from one point. In general, information systems are not designed for integration. Thus, whenever integrated access to different source systems is desired, the sources and their data that do not fit together have to be coalesced by additional adaptation and reconciliation functionality. Recall that there is not the one single integration problem. The future areas that are being researched include simulating proper algorithms for ranking of results and adding the ability to learn based on previous access to information so as to impact on the weight of each query result. Mobilizations of support to make DBMS vendors use a common connection standard. Existing repositories have to be fully considered in the system acquisition process with special consideration to the need for complete integration.

Acknowledgements

I would like to express my sincere gratitude and to acknowledge Strathmore University for the Masters Scholarship opportunity awarded to pursue the Masters of Science in Information Technology programme of which this research is part of. Further, my special thanks to Dr. Reuben Marwanga and Dr Joseph Sevilla of the Faculty of Information Technology, Strathmore University, Dr. Cyrus Wekesa of University of Nairobi and finally Professor Faustin Kamuzora from Mzumbe University for their support and suggestions for improvement in the research.

REFERENCES

Anandarajan, M., Anandarajan, A., and Srinivasan C 2004. Business Intelligence Techniques, Berlin-Germany: Springer-Verlag

Boiling, S., Dasika P. and Krisnan S., *Enterprise Application Integration,* 2008. Retrieved 5th October 2010, from http://searchsoa.techtarget. com/s Definition/0,,sid26_gci213523,00.html.

Liautaud, B., & Hammond, M. 2001. e-Business Intelligence, New York: Mc-Graw Hill

Microsoft Corporation, 2010. Prescriptive Architecture: Message Bus. Retrieved 19th May 2010, from http://msdn.microsoft.com/en-us/library/ff647328. aspx,

Parsaye, K., and Chignell, M. 1995. Intelligent Database Tools and Applications. New York: John Wiley & Sons, Inc

Shacter, S. 2007 Data Integration: Past & Future, Advanster, USA.

Shibwabo, B. and Ateya, I. 2010. Systems Integration: The Disconnect and Way forward through Repository Virtualization supporting Business Intelligence. Proceedings of 2nd European Symposium on Progress in Information and Communications Technology (SPICT 2010), Kuala Lumpur, Malaysia, pp.193-202

Techtarget Incorporation, 2010. State of SOA 2010

Themistocleous, M., Irani, Z. and Sharif, A. 2000. Evaluating Application Integration, Proceedings of 7th European Conference on Evaluation of Information Technology (ECITE 2000), Dublin, Ireland, pp.193-202