2018-04-30

A Review of Trusted Broker Architectures for Data Sharing

Talburt, John R.

Monterey, California. Naval Postgraduate School

SYM-AM-18-106

PROCEEDINGS OF THE FIFTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

THURSDAY SESSIONS VOLUME II

Acquisition Research: Creating Synergy for Informed Change

May 9–10, 2018

March 30, 2018

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

A Review of Trusted Broker Architectures for Data Sharing

John R. Talburt—University of Arkansas, Little Rock. [jrtalburt@ualr.edu]

Abstract

Sharing data across organizational boundaries must strike a balance between the competing data quality dimensions of access and security. Without access to data, it can't be used and, consequently, is of no value. At the same time, uncontrolled access to data, especially sensitive personal data, can result in dire legal and ethical consequences. This paper discusses the trade-offs between security and access for three styles of trusted broker architectures in hopes that this will provide guidance for organizations trying to implement data sharing systems.

Background

There are three important topics that provide the background for the discussion of data sharing architectures. These topics are

1. The difference between data sharing and data exchange
2. The need for persistent data integration to enable data sharing
3. The trusted broker as a vehicle for data sharing

Each of these topics is explored in the following sections of this paper.

Data Sharing Versus Data Exchange

Although data exchange is a form of data sharing, it carries a particular connotation with respect to control and time. Data exchanges in which one organization transfers data to another organization are typically one-to-one, episodic exchanges. Even though there may be a standing agreement between the two organizations, the actual exchanges are usually time-bound in the sense that the sourcing organization provides a snapshot of its data at the time of the transfer to the consuming organization with no further interaction or updating of the data by the sourcing organization. Often the data sharing agreement will require the consuming organization to stop using and delete the data within a specific time period. Even in the case that there is not a specific end date, the fact that the quality and utility of unmaintained data tends to decrease over time has essentially the same effect.

The transfer of data also implies a transfer of control. Once in the hands of the consuming organization, the data are processed and manipulated by the consuming organization without the direct participation of the sourcing organization. The sourcing organization only has indirect control vis-à-vis any requirements or constraints in a written "exchange agreement."

Furthermore, the benefits of using the exchanged data accrue primarily to the receiving organization. Even if there is a reciprocal transfer of data from the receiving organization back to the sourcing organization (true data exchange), the benefits derived from the aggregation of the data are not necessarily reciprocal. Each organization may receive some type of benefit from the exchange, but these are not necessarily the same shared benefit.

On the other hand, data sharing posits a system accommodating participation by several organizations all having joint control and continual access and all deriving mutual benefit from the use of the data. Although data sharing can be done for all types of entities (persons, places, and things), the data sharing systems discussed here are those designed to share data about people (person entities) and organizations (legal entities). In the master data management parlance, data about persons and organizations are referred to as “party” data. Examples of organizations benefiting from data sharing systems include healthcare (providers and payers sharing patient data), law enforcement (city, state, and national agencies sharing criminal and persons-of-interest data), and government (service agencies sharing data about their clients).

The primary mutual benefit derived from data sharing systems is an improvement in the data quality dimension of completeness. Whereas each organization holds a “piece of the puzzle,” when the data are shared, all organizations have the complete picture. At the same time, better completeness leads to better data analysis and insights. For example, a physician is able to make a better diagnosis when he or she is able to view all previous tests and treatments rather than just those done within his or her clinic.

In order to gain a complete view of party information over time, the data sharing needs to enable persistent data integration based on shared, persistent identifiers of the parties (people and organizations). The next section discusses the concepts of transient and persistent data integration.

Transient Versus Persistent Data Integration

Data integration processes can be classified into two different categories based on the persistence of the integration. The two processes are illustrated in Figure 1.

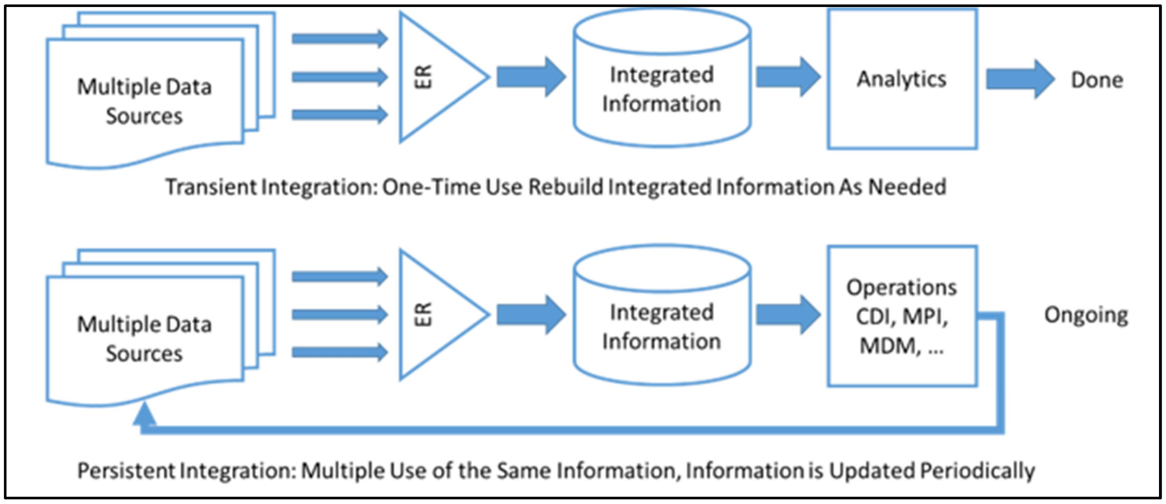

Multiple Data Sources
ER
Integrated Information
Analytics
Done
Transient Integration: One-Time Use Rebuild Integrated Information As Needed
Multiple Data Sources
ER
Integrated Information
Operations CDI, MPI, MDM, …
Ongoing
Persistent Integration: Multiple Use of the Same Information, Information is Updated Periodically

Figure 1. Transient Versus Persistent Data Integration

There is a difference between data integration processes used to support projects such as data analytics as opposed to data integration supporting ongoing operations such as MDM. Both processes rely on ER to link together the information related to the same entity of interest (e.g., the same patient).

In a transient data integration process, ER alone is sufficient. Once the analysis of the integrated information is complete, the process (project) is over. Transient data integration represents data integration on demand. Each time a new analysis or project is needed, the ER process is run again to re-integrate the information. Bringing together data on demand for analysis is sometimes referred to as “data wrangling,” a quick and dirty data integration process not intended to be operationalized.

On the other hand, there are many other data operations for which transient data integration is not well-suited. In particular, MDM cannot be managed effectively using transient data integration. The two motivations for using persistent data integration instead of transient data integration are reuse of effort and changes in characteristic information over time.

Taking the same sources of information and re-integrating them again and again can expend a lot of time and effort, especially if the volume of information is large. For example, suppose you want to generate a report of total sales by customer on a weekly basis. With transient data integration, all of the customer sales transactions must be merged and the ER process run against the combined file. In a persistent data integration process, the groups of sales records from one week are saved, and only the new sales from the next week are incrementally added to the previous groups.

Another reason for persistent data integration is that the characteristic information describing real-world entities changes over time. Characteristic information represents the properties that, taken together, uniquely define a specific entity. Characteristics are also called “identity attributes.” For example, there may be many students in a college with the first name LINDA. Fewer students, but more than one, may have the last name HENDERSON. But only one student will have 1987-11-06 as her date of birth and live on ELM STREET. The values of these attributes taken together identify a particular student.

The problem is these identity attributes are not immutable. LINDA may move to a new address, or she may marry and begin to use her husband’s surname. These changes speak to the need to store and manage identity information in order to consistently identify the same entity over time.

Identity Management and Persistent Identifiers

An Entity Identity Information Management (EIIM) system is an additional layer of ER functionality, namely, the ability to assign the same identifier to references to the same entity over time (i.e., from process to process). EIIM is the collection and management of identity information with the goal of sustaining entity identity integrity over time (Zhou & Talburt, 2011). Entity identity integrity is the state where each entity managed by the system is assigned a unique identifier, and distinct entities are assigned distinct identifiers (Maydanchik, 2007). Entity identity integrity is a basic requirement for MDM systems.

EIIM comprises several processes. Each EIIM process uses aspects of ER and data structures representing the entity identities to accomplish specific goals. These work in concert to maintain the entity identity integrity of master data over time. EIIM is not limited to MDM. It can be applied to other types of systems and data as diverse as RDM systems, referent tracking systems (Chen et al., 2013), and social media (Mahata & Talburt, 2014).

Although ER is a necessary component of MDM, ER is not in itself sufficient to manage the life cycle of identity information. EIIM is an extension of ER in two dimensions, identity knowledge management and time. The knowledge management aspect of EIIM relates to creating, storing, and maintaining identity information. The knowledge structure representing a master data object (entity) is an entity identity structure (EIS).

An EIS is essential to assuring an entity under management in the MDM system is consistently labeled with an identifier from process to process. The EIS stores the identity information of the entity along with its identifier so both are available to future processes. Persistent entity identifiers are not inherently part of ER. At any given point in time, the only goal of an ER process is to correctly classify a set of entity references into clusters where all of the references in a given cluster are equivalent. The labels (identifiers) of these clusters serve as the identifier of the entity. Without storing and carrying forward identity information in the EIS, the identifiers assigned to entities and entity references may change in future processes.

Figure 1 illustrated the problem. It shows three references, References 1, 2, and 3. References 1 and 2 are equivalent references, and Reference 3 is to a different entity. References 1, 2, and 3 are in File 1 along with other references when the first ER process runs. The same References 1, 2, and 3 are also present in File 2, and the same ER process is run against File 2 at a later time. In both cases, if the ER process is accurate, it will classify References 1 and 2 as equivalent and place Reference 3 in a different cluster. The problem is that where there is no communication between the processes, even though the references are clustered properly, they will most likely be labelled differently (i.e., given different identifiers), as shown in Figure 2.

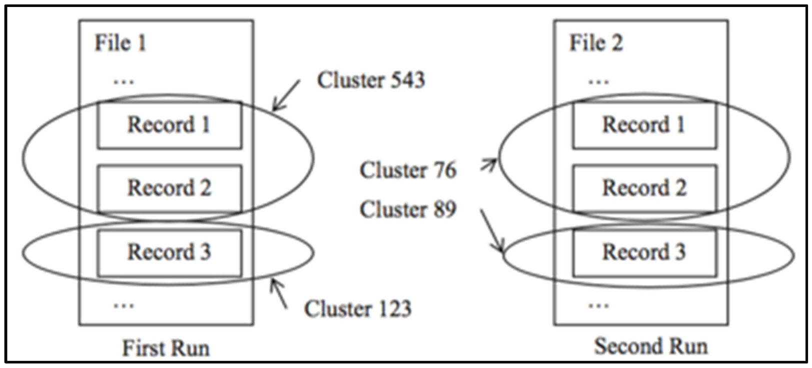


Figure 2. ER With Consistent Classification but Inconsistent Labeling

In the first run, the cluster containing References 1 and 2 is identified as Cluster 543, whereas in the second run, the same cluster is identified as Cluster 76. The purpose of the EIS is to provide the communication between processes so that a reference to an entity in one process is labeled by the same identifier to the same entity in later processes.

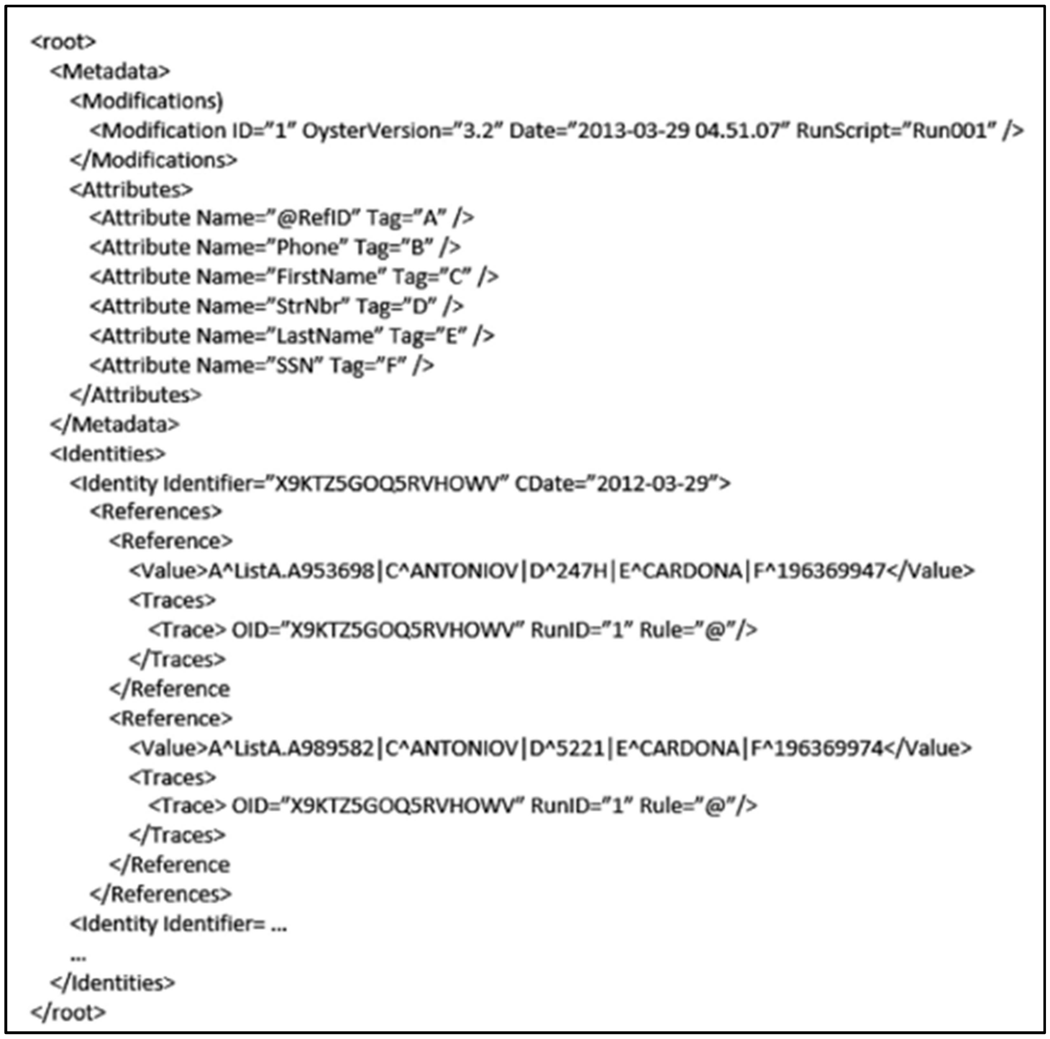
ER that only classifies records into clusters representing the same entity is sometimes called a “merge-purge” operation. In a merge-purge process, the objective is simply to eliminate duplicate entity references. In this context, the term “duplicate” does not mean that the references are exactly the same in content. Instead, it means the references are for the same entity. To avoid this confusion, the term “equivalent” is preferred (Talburt, 2011) to describe references to the same entity.

The term equivalence also avoids confusion arising from designation of “matching” references. References to the same entity do not necessarily have matching information. For example, two records for the same customer may have different names and different addresses. Conversely, matching references do not necessarily refer to the same entity. Two references can be quite similar when important identity information is missing. For example, references to John Doe, Jr., and to John Doe, Sr., may be classified as matching records if the suffix Jr. or Sr. has been omitted from one or both of the references.

Unfortunately, many authors use the term “matching” to mean two references are very similar and also to mean they reference the same entity. The result can often be confusing for the reader. Reference matching and reference equivalence are different concepts and should be described by different terms.

The ability to consistently assign a cluster the same identifier when an ER process is repeated at a later time requires an EIS to carry forward identity information from process to process. The storage and management of identity information in the EIS is what enables the persistence of entity identifiers. Persistent identifiers are the value add EIIM brings to ER.

Different EIIM systems implement different strategies for identity information management. Figure 3 shows one example from the OYSTER open source system (Talburt & Zhou, 2013; Zhou et al., 2010).



```
<root>
  <Metadata>
    <Modifications)
      <Modification ID="1" OysterVersion="3.2" Date="2013-03-29 04.51.07" RunScript="Run001" />
    </Modifications>
    <Attributes>
      <Attribute Name="@RefID" Tag="A" />
      <Attribute Name="Phone" Tag="B" />
      <Attribute Name="FirstName" Tag="C" />
      <Attribute Name="StrNbr" Tag="D" />
      <Attribute Name="LastName" Tag="E" />
      <Attribute Name="SSN" Tag="F" />
    </Attributes>
  </Metadata>
  <Identities>
    <Identity Identifier="X9KTZ5GOQ5RVHOWV" CDate="2012-03-29">
      <References>
        <Reference>
          <Value>A^ListA.A953698|C^ANTONIOV|D^247H|E^CARDONA|F^196369947</Value>
          <Traces>
            <Trace> OID="X9KTZ5GOQ5RVHOWV" RunID="1" Rule="@"/>
          </Traces>
        </Reference
        <Reference>
          <Value>A^ListA.A989582|C^ANTONIOV|D^5221|E^CARDONA|F^196369974</Value>
          <Traces>
            <Trace> OID="X9KTZ5GOQ5RVHOWV" RunID="1" Rule="@"/>
          </Traces>
        </Reference
      </References>
    <Identity Identifier= ...
    ...
  </Identities>
</root>
```

Figure 3. Example of an EIS in XML Format Created by the OYSTER ER System

During processing, the OYSTER EIS exists as in-memory Java objects. However, at the end of processing, the EIS are serialized in an XML document. The XML structure encapsulates extensive metadata to reflect not only the entity identifier (OID value), but also many other aspects of the process such as the sources of the references, the label of the matching rule that linked a reference into the cluster, the identifier of the process (run) when the reference was brought into the cluster, and other information. The XML format also serves as a way to serialize the EIS objects so that they can be reloaded into memory at the start of a later ER process.

Components of Entity Information

To better understand the implementation of EBDI, it is helpful to think of entity information as having three components:

1. A unique and persistent entity identifier. This is a single value assigned as the unique identifier of a particular instance of the entity. It is used to label all information related to the entity. The intent is the identifier should not change (persist) over time. However, processing errors may mandate changing the identifier. An important EBDI goal is to continually improve the identification process and thereby minimize changes to entity identifiers. Best practice is to make the identifier entirely anonymous, such as using a Really Simple Syndication (RSS) hashing algorithm on some identity fields to create an identifier value. An example is the entity identifier “X9KTZ5GOQ5RVHOWV” shown in Figure 2. Encoding information into the identifier can increase the likelihood the value of the identifier may have to be changed to reflect changes in the encoded information.
2. Entity identity specific information. These are identity attributes, sometimes called characteristic information (ISO 8000), that, when taken together, distinguish one entity from another. Another important goal of EBDI is the reuse of identification effort. In the EBDI process, the identity attribute values of entity records are resolved to the entity identifier. The entity identifier is then used to label the record. Later processes can then simply use join or sort records by the entity identifier to aggregate entity information, relieving these processes of the need to apply their own (and possibly less accurate) matching logic.
3. Application specific information associated with an entity (multiple values).

The identity management components are shown in Figure 4.

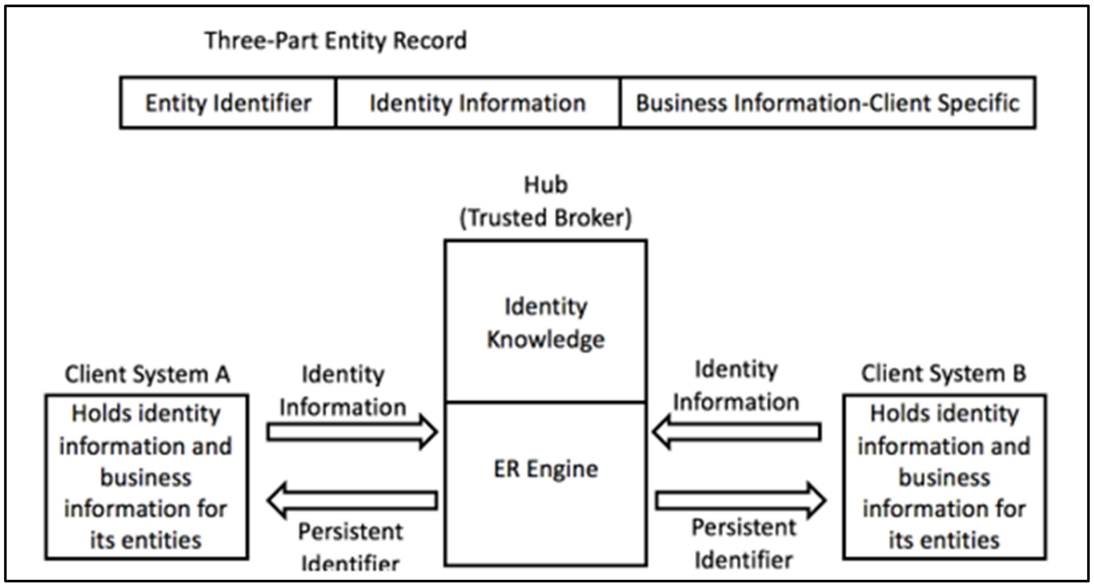


Figure 4. The Identity Management Component

Entity-Based Data Integration (EBDI)

Entity-Based Data Integration (EBDI) is the process of bringing together information about the same real-world entity. From an EBDI perspective, an entity is a set of real-world objects that have distinct identities, for example, customers of a business, patients of a hospital, students of a school system, products of a manufacturer, and locations of service. The fundamental issue with EBDI is that in the modern world, we describe these entities as data records in an information system. As we all know, the blessing and curse of storing information in a computer is how easy it is to copy and replicate information. As a result, there are often many different records in the information system all describing the same entity.

Entity-based data integration has a broad range of applications in areas such as law enforcement (Nelson & Talburt, 2008), education (Nelson & Talburt, 2011; Penning & Talburt, 2012), and healthcare (Christen, 2008; Lawley, 2010). For example, a hospital patient will have separate records for each hospital visit. In addition, each visit or encounter will generate many additional records such as a record for each laboratory test, each treatment, each drug administered, and so on. Aside from the records of medical treatment, there will also be billing records associated with each visit and treatment.

Consequently, EBDI is a two-step process (Talburt & Hashemi, 2008). When integrating entity information from multiple sources, the first step is to determine whether the information is for the same entity. Once it has been determined the information is for the same entity, the second step is to reconcile possibly conflicting or incomplete information associated with a particular entity coming from different sources (Holland & Talburt, 2008, 2010; Zhou, Kooshesh & Talburt, 2012). Master data management (MDM) systems play a critical role in successful EBDI by providing an EIIM process, enabling the MDM system to consistently identify and label references to the same entity.

Three Trusted Broker Architectures for EBDI

- External Reference Architecture
- Registry Architecture
- Transaction Hub
  (Berson & Dubov, 2011)

External Reference Architecture

In the external reference architecture (shown in Figure 5), the IKB is a large cross-reference table connecting equivalent references located in the various client systems. The EIS in the IKB are entirely virtual, only containing pointers to the references to a particular entity. None of the actual entity identity information for a particular entity is stored in the IKB.

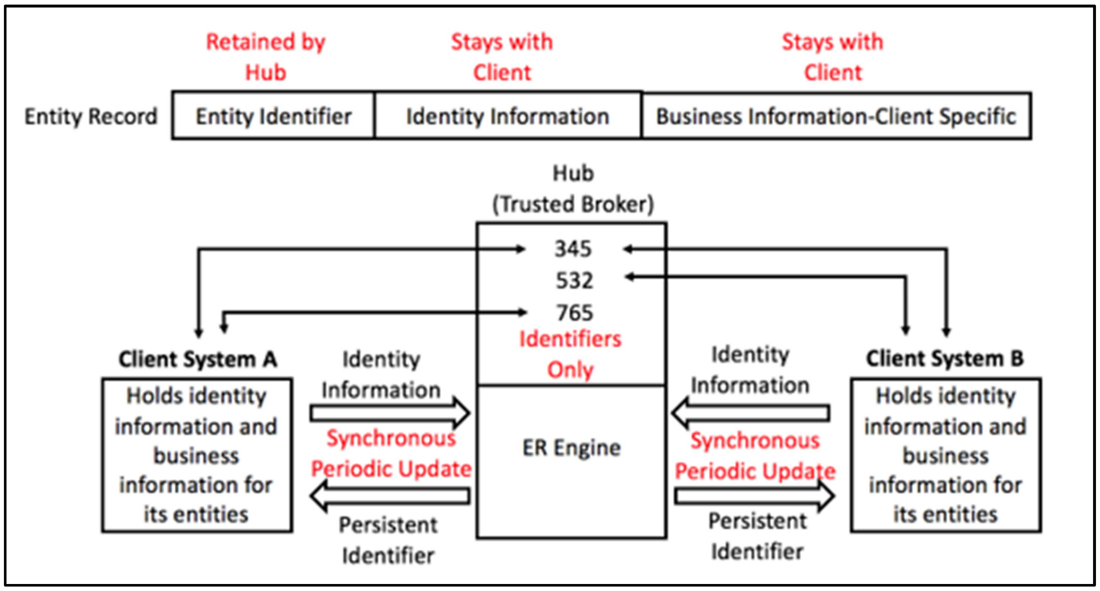


Figure 5. The External Reference Architecture

Both the identity attribute values and the application-specific attribute values of the source record reside in the client system, as shown in Figure 6. The advantage of external reference architecture is that changes to an entity identifier taking place in one system can be more easily propagated to all other client systems where the same entity is referenced.

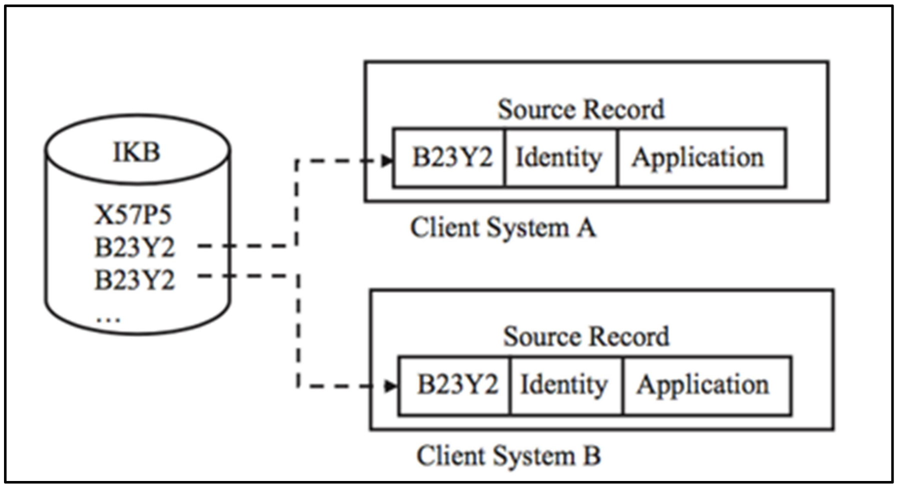


Figure 6. External Reference Schema

The external reference architecture works best when the governance policy allows for distributed authority to make master data changes in several different client systems. It does not work as well in systems where a large number of new source records must be ingested and identified on a regular basis. In systems implementing external reference architecture, the identity information needed for matching must be marshaled on demand from the client systems where it resides.

External Reference Architecture Pluses

- The hub (broker) does not retain identity information between updates, so there is less risk of identity leakage.
- Identifiers are anonymous.
- Lightweight IT Infrastructure, periodic bulk processing, no transactional processing, less staffing.
- Less complex ER processing is required.
- Detection of ER linkage errors is done by client personnel.
- When found, errors are reported to the hub, where they are corrected during periodic processing.

External Reference Architecture Minuses

- Linkage between client records is only as current as the last update.
- All clients must provide all of their identity information at the same time to the hub in order to update the cross-reference table.
- Because the hub does not hold identity information, inquiries for specific entities must be made through a client.
- Only after an entity is identified by a client can the hub can tell if other clients have records for the same entity.
- Analytics is less agile. Must first collect and combine information from each agency onto one platform before starting analysis.

Registry Architecture

A more common architecture for MDM applications ingesting and managing large volumes of input data is registry architecture. In registry architecture, each EIS in the IKB contains a collection of identity attribute values representing the entity under management. Each EIS has an identifier serving as the master identifier for the entity across all client systems. The amount of identity information retained in the EIS will vary from system to system and on the choice of EIS strategy as discussed earlier (e.g., survivor record, attribute-based, record-based, etc.).

In registry architecture, each reference is divided into two parts, as shown in Figure 7. The values for the identity attributes are kept in the IKB. The IKB must retain sufficient identity information so that when a new source record is introduced, the system can correctly determine if it references a previously registered entity already having established entity identifiers or if it references a new entity which must be registered in the IKB with a new entity identifier.

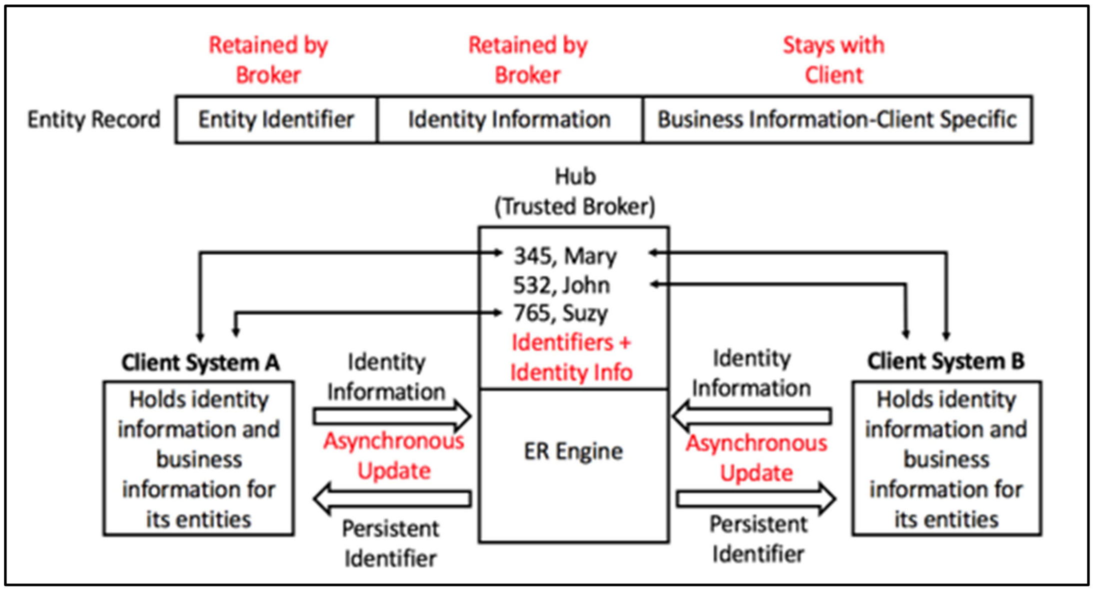


Figure 7. The Registry Architecture

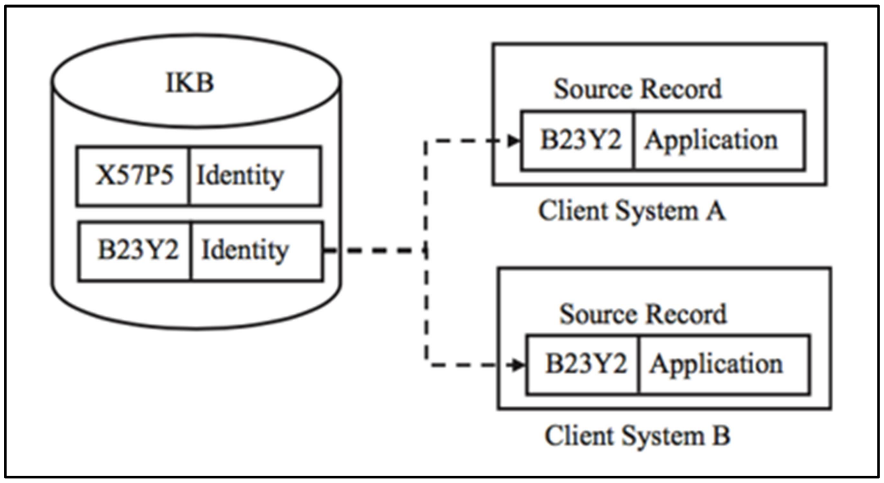


Figure 8. Registry Architecture Schema

A third possibility is if a new source record carries additional identity information providing evidence that two EIS initially thought to represent distinct entities are actually references to the same entity. For example, a source record having both a current and previous address for a customer connects an EIS created for that customer at the previous address and another EIS created for the same customer at the current address. As a result, one of the entity identifiers is retired, usually the most recently assigned, and the identity information in the two EIS are merged.

In registry architecture, the values for client-specific attributes are retained in the source records residing in the client systems. The two halves of the source record the identity values and the client-specific values are linked together by the entity identifier. The shared link identifier also does away with the need to store the identity values of the reference in the client system. The replacement of entity identity values with a unique identifier for the entity is called semantic encoding. Semantic encoding is one of the fundamental principles of the ISO 8000-110:2009 Standard for Master Data Quality.

In registry architecture, the IKB and the systems are loosely coupled. It is usually the responsibility of each client system to synchronize its entity identifiers with the identifiers in the registry through periodic batch or interactive inquiries to the registry. The registry architecture is typical for most CDI systems primarily providing an identity management service (i.e., appends link identifiers to source records on demand).

Registry architecture is sometimes used to provide anonymous entity resolution to several external clients in a trusted broker configuration (Talburt et al., 2005). Trusted broker architecture can be useful when each external client holds some information for entities also known to other clients but also manages information for some entities unique to the client organization. The clients want to collaborate and share information about common entities but do not want their exclusive entity information shared or exposed to other clients. This situation often arises in law enforcement, healthcare, and information sharing among government agencies.

The name comes from the fact that all of the clients must trust one neutral organization to host the hub of the registry. In addition, even though the hub internally maintains only one set of entity identifiers, it issues a different set of entity identifiers to each client. This means even though two different clients hold information about the same entity, the hub will give each client a different identifier for that same entity. The hub mediates the translation between client identifiers. In this way, the trusted broker also incorporates some features of the external reference architecture.

If Client A wants to know whether Client B is holding information about an entity of interest, Client A sends an inquiry to the hub organization. The hub organization can then translate the Client A identifier into its internal identifier and determine if Client B has information on the entity. The hub organization can also mediate policies or regulations on access. If, according to policy, Client A's inquiry is valid, then the hub can send the information from Client B to Client A using the entity identifier of Client A.

Registry Architecture: Pluses

- Clients can update information in the hub according to their own schedule and are not required to coordinate with other clients.
- Depending upon client update schedules, the linkage between clients can be more up-to-date than the external reference.
- Because the hub holds identity information, inquiries for a specific entity can be made at the hub and immediately find which clients hold information for that entity.
- Detection of linkage errors can be done by both hub and client personnel.

Registry Architecture: Trade-Offs

- The hub (broker) retains all identity information between updates, so there is a greater risk of identity leakage.
- Analytics on business information is still less agile.
- Must first collect business information from each agency onto one platform before starting analysis.
- Requires heavier IT infrastructure and staff to service ongoing updates and linkage corrections.
- Requires more complex ER functions including a system for publishing identifier changes (due to linking errors) to clients.

Transaction Hub: Manages Identity and Business Information

The transaction hub architecture is also a hybrid. It attempts to solve both the attribute partitioning problem and the synchronization problem at the same time. In this case, the hybridization is between the IKB and its client systems. In a transaction hub, the IKB stores the complete source record, both identity attributes, and application-specific attributes.

By incorporating the source records into the IKB, the transaction hub is simultaneously an MDM system and an application system. The transaction hub can be a good solution for situations where the system must process large volumes of new source references while at the same time servicing high volumes of inquiries for application-specific information because the application information is immediately at hand. There is no need to fetch the application information from a client system in order to service the inquiry. However, this is only feasible if only one or two applications are integrated with the hub; otherwise, the maintenance of the system becomes too complex to manage. Many financial systems incorporate the transaction hub architecture for MDM.

Transaction Hub Architecture

The transaction hub architecture is shown in Figure 9.

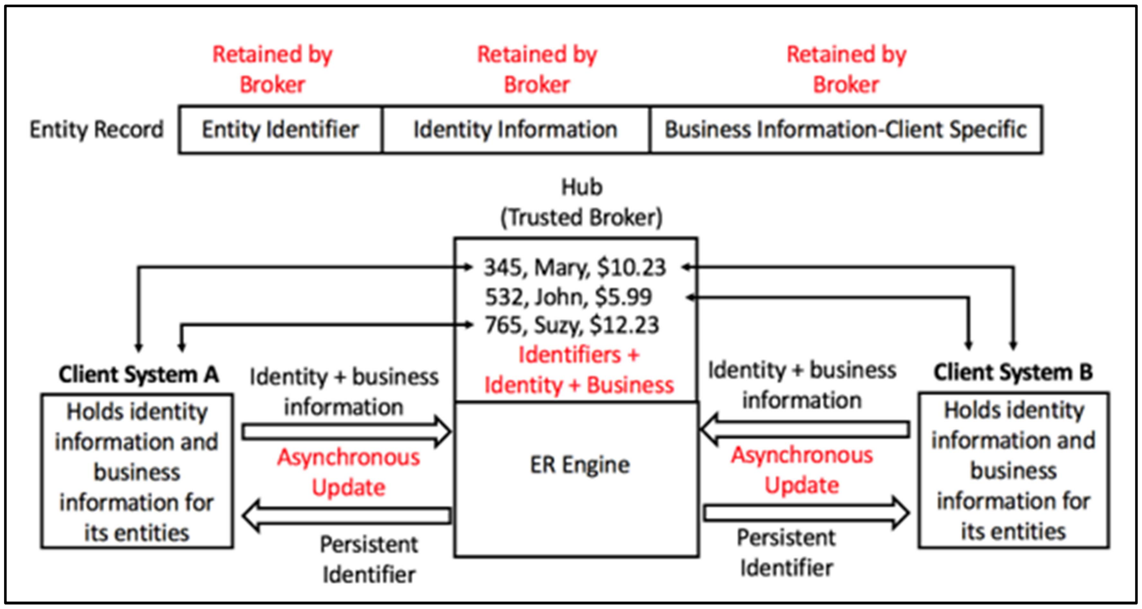


Figure 9. The Transaction Hub Architecture

Transaction Hub: Pluses

- The biggest advantage is that analytics is simpler than with registry or external reference. All of the information to analyze is already in one platform. There is no need to request and join information from clients.
- Clients can update information by their own schedule.
- Depending upon client participation, the linkage between clients can be more up-to-date than the external reference.

Transaction Hub: Trade-Offs

- The hub (broker) retains all identity and business information between updates, so there is a greater risk of not only identity leakage, but also leakage of business information.
- Requires more complex ER functions including a system for publishing identifier changes (due to linking errors) to clients.
- The ER system requires additional functionality to store and retrieve business information with different schemas.

Conclusion

The selection of an appropriate architecture to support data sharing depends primarily on finding an acceptable point of balance between data exposure risk and operational agility. Following the progression from data exchange agreements, external reference, and registry to full transaction hub leads to increasing aggregations of sensitive data in one system for longer periods of time. However, the same progression provides increasing ease of access and agility to support maintenance and data analytics. The real key to successful data sharing is a strong data governance program that implements and enforces compliance with the policies, assuring that the privacy and regulatory compliance of all the participating organizations are satisfied. Data sharing and MDM systems lacking strong data governance programs are doomed to failure.

References

Berson, A., & Dubov, L. (2011). *Master data management and data governance*. McGraw Hill.

Chen, C., Hanna, J., Talburt, J. R., Brochhausen, M., & Hogan, W. R. (2013, July). A demonstration of entity identity information management applied to demographic data in a referent tracking system. *International Conference on Biomedical Ontology (ICBO) 2013* (pp. 136–137). Montreal, Canada.

Christen, P. (2008, January). Febrl-A freely available record linkage system with a graphical user interface. In *Proceedings of the Australian Workshop on Health Data and Knowledge Management (HDKM): Conferences in Research and Practice in Information Technology (CRPIT*; vol. 80).

Holland, G., & Talburt, J. (2008). A framework for evaluating information source interactions. In C. Hu & D. Berleant (Eds.), *The 2008 Conference on Applied Research in Information Technology* (pp. 13–19). Conway, AR: University of Central Arkansas. Retrieved from http://research.acxiom.com/publications.html

Holland, G., & Talburt, J. (2010). An entity-based integration framework for modeling and evaluating data enhancement products. *Journal of Computing Sciences in Colleges, 24*(5), 65–73.

Lawley, E. (2010, March 29). Building a health data hub. *Nashville Post* (online version).

Mahata, D., & Talburt, J. R. (2014, August 13). A framework for collecting and managing entity identity information from social media. In *Proceedings of the 19th MIT International Conference on Information Quality* (pp. 216–233). Xi'an, China.

Maydanchik, A. (2007). *Data quality assessment*. Technics Publications.

Nelson, E., & Talburt, J. (2008). Improving the quality of law enforcement information through entity resolution. In C. Hu & D. Berleant (Eds.), *The 2008 Conference on Applied Research in Information Technology* (pp. 113–118). University of Central Arkansas, Conway, AR. Retrieved from http://research.acxiom.com/publications.html

Nelson, E., & Talburt, J. (2011, July). Entity resolution for longitudinal studies in education using OYSTER. In *Proceedings of the 2011 Information and Knowledge Engineering Conference (IKE 2011)* (pp. 286–290). Las Vegas, NV.

Penning, M., & Talburt, J. R. (2012, July). Information quality assessment and improvement of student information in the university environment. In *Proceedings of the 2012 International Conference on Information and Knowledge Engineering (IKE '12)* (pp. 351–357). Las Vegas, NV.

Talburt, J., & Hashemi, R. (2008). A formal framework for defining entity-based, data source integration. In H. Arabnia & R. Hashemi (Eds.), *2008 International Conference on Information and Knowledge Engineering* (pp. 394–398). Las Vegas, NV: CSREA Press.

Talburt, J., Morgan, C., Talley, T., & Archer, K. (2005). Using commercial data integration technologies to improve the quality of anonymous entity resolution in the public sector. In F. Naumann, M. Gertz, & S. Madnick (Eds.), *10th International Conference on Information Quality* (pp. 133–142). Cambridge, MA: MIT IQ.

Talburt, J., & Zhou, Y. (2013). A practical guide to entity resolution with OYSTER. In S. Sadiq (Ed.), *Handbook on research and practice in data quality* (pp. 235–270). Springer.

Talburt, J. R., 2011. *Entity resolution and information quality*. Morgan Kaufmann.

Zhou, Y., Kooshesh, A., & Talburt, J. (2012). Optimizing the accuracy of entity-based data integration of multiple data sources using genetic programming methods. *International Journal of Business Intelligence Research, 3*(1), 72–82.

Zhou, Y., & Talburt, J. R. (2011, November). Entity identity information management. *International Conference on Information Quality 2011.* Retrieved from http://iciq2011.unisa.edu.au/doc/ICIQ2011_Proceeding_Nov.zip

Zhou, Y., Talburt, J., Su, Y., & Yin, L. (2010, November). OYSTER: A tool for entity resolution in health information exchange. In *Proceedings of the Fifth International Conference on the Cooperation and Promotion of Information Resources in Science and Technology (COINFO '10)* (pp. 356–362). Beijing, China.