OYSTER: An Open Source Entity Resolution System Supporting Identity Information Management

Yinle Zhou
yxzhou@ualr.edu

John R. Talburt
jrtalburt@ualr.edu

Center for Advanced Research in
Entity Resolution and Information Quality (ERIQ)
University of Arkansas at Little Rock
http://ualr.edu/eriq

Abstract

The paper describes the design and operation of OYSTER, an open source, entity resolution system that was specifically designed to support identity information management. OYSTER is a script-driven system that can be configured at run-time to perform record linking, identity resolution, identity capture, or identity update operations. The system assigns and maintains persistent entity identifiers that can be used to support master data management (MDM) applications.

Keywords: Identity information management, entity resolution, OYSTER, EIIM

Introduction

Even as Identity Management (IdM) and master data management (MDM) are becoming increasingly critical components for the security and effectiveness of businesses and government agencies, complete and effective solutions are still elusive. One of the reasons for this is the gap that currently exists between entity resolution (ER) systems that focus on one-time or period alignment of identity and the need to maintain entity identity integrity over time.

For this reason much of the research performed at ERIQ Research Center has focused on building tools that can support the management of identity information. The ERIQ researchers call this new area of research Entity Identity Information Management (EIIM). Three existing areas of research and practice form a context for EIIM. They are entity resolution (ER), master data management (MDM), and identity management (IdM).

Entity Identity Information Management (EIIM) is the collection and management of identity information with the goal of sustaining entity identity integrity (Zhou & Talburt, 2011). Entity identity integrity is one of the basic tenets of data quality that applies to the representation of a given domain of real-world entities in an information system (Maydanchik, 2007). Entity identity integrity has also been described as proper representation (Huang, Lee, & Wang, 1999). Entity identity integrity requires that

- Each real-world entity in the domain has one and only one representation in the information system;
- Distinct real-world entities have distinct representations in the information system.

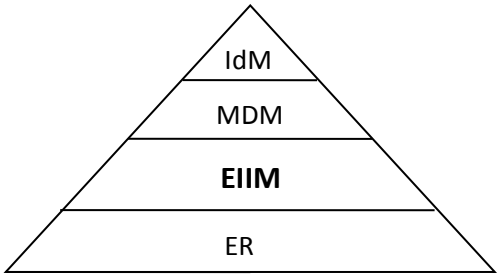

Figure 1: The Role of EIIM

Figure 1 shows the important position EIIM serves as the connection between ER and MDM, which in turn supports IdM. Whereas ER is the process of determining whether two references to real-world objects in an information system are referring to the same object, or to different objects (Talburt, 2011).

Components of EIIM

A high-level view of EIIM components and processes is shown in Figure 2

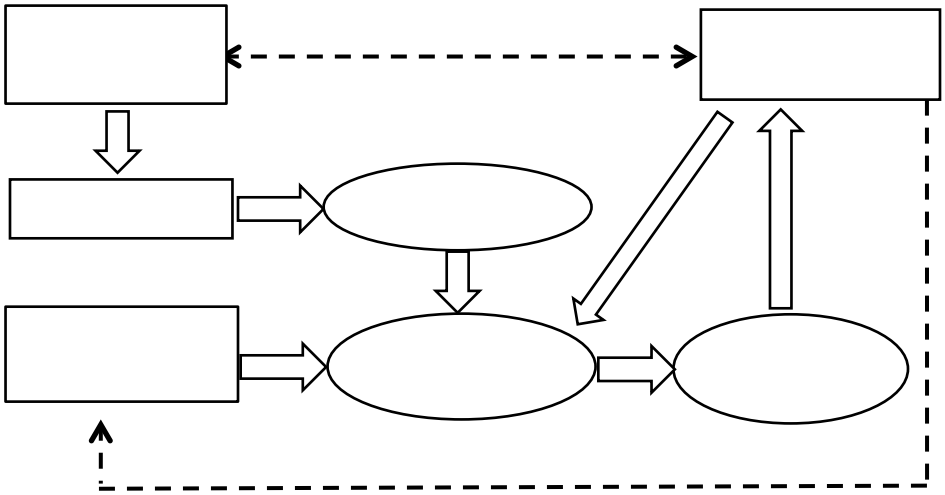
Figure 2: EIIM Components and Interactions

The labeled items in Figure 2 are described as follows (Zhou & Talburt, 2011)

1. The fundamental goal of EIIM is to effect entity identity integrity which is essentially to maintain a one-to-one correspondence between the entity identity structures (EIS) in the information system and the real-world entities in the domain of interest.
2. References are records in the information system that refer to the real-world entities. They may also occur in both structured and un-structured formats.
3. In most cases, entity references undergo a number of preparation steps to improve the quality of the data. These steps may include entity attribute extraction (in the case of unstructured information), reformatting, standardization, correction, and enhancement.
4. After references have been prepared, the next step is to resolve the references against other references and also against existing EIS. For any given input reference the resolution process must decide if that reference is equivalent to any other input reference or a previously created EIS.
5. In the case that two references or EIS are resolved as equivalent, the transitive closure of equivalence requires that they be integrated into a single EIS.

6. The integration process will create a new EIS from a single input reference when the ER process is unable to resolve the input reference to any existing EIS. When an input reference resolves to an EIS, the EIS is updated to include the new information from the reference. In the case that it resolves to more than one EIS, the reference and EIS are all integrated into a single EIS.
7. As described above, the EIS are also inputs to the resolution process as well as outputs from the integration process.
8. Perhaps the most important aspect of EIIM is that it is a cyclical process rather than a one-time process. Just as with any type of information, entity identity information has a life cycle as new identities are created, updated, combined, and eventually discarded. EIIM systems have a continual influx of new reference information, and the resolution and integration of these references will impact the state of entity identity integrity of the system.
9. System configuration has eight modes, which are record-linking mode, identity capture mode, identity resolution mode, identity update mode, reference-to-reference assertion mode, reference-to-structure assertion mode, structure-to-structure assertion mode, and structure-split assertion mode.

This remainder of this paper focuses on the eight EIIM system configurations and how they are designed and implemented in an open source entity resolution system (OYSTER) to support EIIM.

EIIM System Configurations

Shown as Process 9 in Figure 2, the system configuration plays an important role in the EIIM design. The EIIM system configuration defines the input and output elements what the EIIM system is composed of. The input elements include input reference sources, assertion input, and identity input. The output elements are link output and identity output. Each configuration has different requirement for these input and output elements.

Table 1: Eight EIIM System Configurations

	Record-Linking	Identity-Capture	Identity-Resolution	Identity-Update	RefToRef Assertion	RefToStr Assertion	StrToStr Assertion	StrSplit Assertion
Reference Sources	Required	Required	Required	Required	Required	Required	None	None
Assertion Input	None	None	None	None	Required	Required	Required	Required
Identity Input	None	None	Required	Required	None	Required	Required	Required
Link Output	Required	Required	Required	Required	Required	Required	None	None
Identity Output	None	Required	None	Required	Required	Required	Required	Required

Table 1 lists the eight configurations being used in this research and their required input and output elements. Notation “Req” means the configuration requires that element. Notation “None” means the configuration does not have this element.

Record-linking is the basic form of ER which takes input references and produces the link index that assigns the same identifier for equivalent references. Identity-capture is the starting of the EIIM process which takes input references and captures the identities to form the identity knowledgebase. The identity knowledgebase could be updated by identity-update runs, also be used for identity-resolution runs against with. Record-linking, identity-capture, identity-resolution, and identity-update are inferred resolution which takes the information from the input references and makes the decision. The last four configurations in Table 1 are asserted resolution (Zhou & Talburt, 2011) which is when resolution decisions are made based on knowledge from external sources rather than inferences based on values and relationships within the system. Assertions override the equivalence rules in the resolution process to directly create, update, or combine EIS directly. Asserted resolution allows the EIIM process to be

adjusted for drift caused by errors inherent in the inferred resolution process. For most systems not every false positive or false negative issue can be solved by changing an existing equivalence rule or adding a new equivalence rule. Even small rule changes designed to solve one particular error can often create many other unintended errors.

Demonstration of EIIM in OYSTER

OYSTER (Open sYSTem Entity Resolution) is an EIIM system developed by the ERIQ research center. As this writing, the latest version is OYSTER 3.2, and the source code and documentation are available from SOURCEFORGE website (http://sourceforge.net/projects/oysterer/). Written in Java, the reference sources and identity rules used to resolve the references are given by the user in the form of XML scripts. One advantage of OYSTER over similar systems is its support for identity management. The OYSTER EIS are also in the form of XML documents that define and retain the identity information captured during the ER process.

Demonstration context

To set the context of the following examples, assume there is a company named ABC, Inc. trying to integrate and manage their customer data. These examples will refer to three (synthetic) customer data files named List_A, List_B, and List_C. For the interested reader, these files can be downloaded online from the website ualr.edu/eriq/downloads.

The three customer data files have different subsets of the overall set of identity attributes and the records themselves are in different formats. List_A contains 94,306 references and is in a comma delimited file format with quotation marks used as text qualifiers. There are eight attributes in List_A and they are the

- Unique record identifier (RecID)
- Customer name (Name)
- Customer street address (Address)
- City, state, and zip code of the street address (CityStateZip)
- Customer post office box address (POBox)
- City, state, and zip code of the post office box address (POCityStateZip)
- Customer Social Security Number (SSN)
- Customer data of birth (DOB)

A segment of List_A is shown in Figure 2.a.

The List_B file contains 100,777 references and the records are in a pipe-delimited format without a text qualifier character. List_B has ten attributes,

- Unique record identifier (RecID)
- Customer first name (FirstName)
- Customer last name (LastName)
- Street number of the customer's address (StrNbr)
- Street name of the customer's address (Address1)
- Second line of customer address (Address2)
- City name of address (City)
- State name of address (State)
- Zip code of address (Zip)
- Customer telephone number (Phone).

A segment of List_B is shown in Figure 2.b.

The List_C contains 76,059 references in a fixed-length field format. The attributes in List_C have been sent to ABC by a data provider who did not provide a file layout. It has been left to the IT employees of ABC to decide where each field starts and ends and the content of each field. A segment of List_C is shown in Figure 2.c.

```
"RecID","Name","Address","City State Zip", "PO Box","POCity State Zip","SSN","DOB"
 "A953698","antonio v cardona","247H HAHN ST","San Francisco, Cali 94134","PO BOX 280911","SAN FRANCISCO, CA 94128",196-36-9947,""
"A989582","ANTONIO V CARDONA","5221 ZELZAH AVEN APT219","encin, california 91316","PO BOX V19412","encino, ca 91416",196369974,"1913"
```

Figure 2.a: A segment of List_A

```
RecID|FirstName|LastName|StrNbr|Address 1|Address 2|City|State|Zip|Phone
B932797|ANTONIO V|CARDONA |19412|APTDO||encino|ca|91416| 818-453.1558
B949439|ANTONIO V|CARDONA |1207|Miljl Way||stockton|ca|95209|(209)318-1443
```

Figure 2.b: A segment of

```
C967431  ANTONIO V        CARDONA        196-36-9974 1913        (818)453.1558
```

Figure 2.c: A segment of

Demonstration of Identity-Capture Configuration

As shown in Figure 2, the data in the three lists have many data quality issues. First of all, they are in different format. List_A is in comma-delimited format (sometime called comma separated values or CSV format). It also uses quotation marks as a text qualifier. List_B has a pipe-delimited format without a text qualifier. List_C is in fixed-length field format and does not provide a field layout. Another observation is that the attributes are not uniform across the sources. For example, List_A has an attribute for the full name, but List_B has attributes for first name and last name. There are also some other obvious data quality issues, such as different punctuation of telephone numbers and social security numbers. After observing these and other data quality condition in the ABC customer lists, the following actions are taken for this demonstration before attempting the entity resolution process:

1. Make a best guess at which attributes are in List_C and the starting and ending position of each one. Even though the List_C did not come with a file layout, it can be inferred from the data profile reports and through observation of the values and patterns. For example, every value in the first 7 columns has the pattern "C999999" making it a safe assumption that this represents the unique record identifier. Similar assumptions can be made on other attributes. The final attribute names for List_C are
   - Unique Record Identifier (RecID)
   - Customer First Name (FirstName)
   - Customer Middle Name (MiddleName)
   - Customer Last Name (LastName)
   - Customer Social Security Number (SSN)
   - Customer Date of Birth (DOB)
   - Customer Telephone Number (Phone)
2. Perform data cleaning and standardization on the three lists. First, parsing and consolidating the attributes have been done to make the attributes uniform across sources. Next, apply data standardization and transformations as specified in Table 2.

Table 2: Standardization operations for some attributes

Attribute	Standardization Operations
First Name	• Change to all Upper Case • Remove all non-letter characters
Last Name	
Middle Name	
SSN	Remove all non-digit characters
Phone	
PO Box	Extract the PO Box Number (Java Program)
State	Standardize to two-letter USPS state code
PO State	

After the above steps, run the data profiling analysis again. Based on the analysis, the following identity rules were selected for this example:

Rule 1: FirstName values are the same, the LastName values are the same, and the SSN values the same
Rule 2: FirstName and LastName values have a Levenshtein rating of 0.80 or more and the SSN values differ by one transposition of consecutive digits.
Rule 3: FirstName and LastName values have a Levenshtein rating of 0.80 or more and the StrNbr values are the same.
Rule 4: LastName values have a Levenshtein rating of 0.80 or more and the last 6 digits of the Phone values are the same.
Rule 5: FirstName values are Nicknames (from nickname list), LastName values have a Levenshtein rating of 0.80 or more, and the SSN values differ by one transposition of consecutive digits.
Rule 6: FirstName values are Nicknames (from nickname list), LastName values have a Levenshtein rating of 0.80 or more, and the StrNbr values are the same.

An EIIM process that builds and save the clusters created by a record-linking process is called an EIIM identity capture configuration. Figure 3 shows an example of two entity identity structures (EIS) created by an identity capture configuration (Run 1) of OYSTER acting on List_A and List_B using the identity rules above. Note that each EIS is enclosed in <Identity> element of an XML document <root>. OYSTER assigns each EIS a unqiue 16 character identifier. The two EIS shown in Figure 3 have the identifiers EIS "X9KYZ5GOQ5RVHOWV" and "000Z53TVVK0DQXI1"

Furthermore, Figure 3 also shows that the EIS labeled with "X9KYZ5GOQ5RVHOWV" was created from two input records from List_A (A953698 and A989582) that matched by Rule 2 (note that the SSN values have transposed digits). Also note that in order to save storage, the attributes values are in a compressed, tagged format that is part of the Compressed Document Set Architecture (CoDoSA) (Talburt & Nelson, CoDoSA: A light-weight, XML framework for integrating unstructured textual information, 2009). The tags are defined in the <Attributes> section of the document where "A" is the tag for the unique record identifier (RefID), "B" is the tag for the telephone number (Phone), and so on.

```
<root>
   <Metadata>
      <Modifications>
            <Modification ID="1" OysterVersion="3.2" Date="2012-03-29 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="[2]"/>
            </Traces>
  </Reference>
</References>
  </Identity>
  <Identity Identifier="000Z53TVVK0DQXI1" CDate="2012-03-29">
      <References>
            <Reference>
                <Value>A^ListB.B932797|B^8184531558|C^ANTONIOV|D^19412|E^CARDONA</Value>
                <Traces>
                        <Trace OID="000Z53TVVK0DQXI1" RunID="1" Rule="[@]"/>
            </Traces>
            </Reference>
                    </References>
  </Identity>
 </Identities>
…
</root>
```

Figure 3: EIS "X9KYZ5GOQ5RVHOWV" and "000Z53TVVK0DQXI1" after Identity Capture (Run 1)

The EIS labeled with "000Z53TVVK0DQXI1" was created from a single record from List_B (B932797). The "[@]" given and the Rule value indicates that no rule was used, i.e. the record formed its own cluster. The same rule coding shows that Record A953698 was the record that originally formed Cluster "X9KYZ5GOQ5RVHOWV" and that Record A989582 was brought in later because it matched the first record by Rule 2.

In addition to creating the EIS structures, OYSTER also produces a standard Link Index that simply shows the links assigned to each input record processed. Figure 4 shows a segment of the Link Index with the same records capture in Figure 3.

Figure 4: Segment of the Link Index from Run 1

Demonstration of Identity-Update Configuration

Beyond standard record-linking processes, EIIM systems also have the capability to add and modify EIS originally built in a previous process, an EIIM configuration called identity update. Figure 5 shows an EIS originally created in Run 1 of OYSTER that has been updated when the EIS from Run 1 were run against List_C in an identity update configuration (Run 2) of OYSTER. In particular it shows the EIS labeled "000Z53TVVK0DQXI1" that also appears in Figure 3, and is an EIS that was created from a single record in Run 1.

```
<root>
   <Metadata>
      <Modifications>
         <Modifications>
            <Modification ID="1" OysterVersion="3.2" Date="2012-03-29 04.51.07" RunScript="Run001" />
            <Modification ID="2" OysterVersion="3.2" Date="2012-03-29 07.14.44" RunScript="Run002" />
         </Modifications>
      </Modifications>
         …
  </Metadata>
  <Identities>
 <Identity Identifier="000Z53TVVK0DQXI1" 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="[@]"/>
                     <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[2]"/>
               </Traces>
         </Reference>
         <Reference>
         <Value>A^ListA.A989582|C^ANTONIOV|D^5221|E^CARDONA|F^196369974</Value>
               <Traces>
                     <Trace OID="X9KTZ5GOQ5RVHOWV" RunID="1" Rule="[2]"/>
                     <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[2]"/>
               </Traces>
         </Reference>
         <Reference>
         <Value>A^ListB.B932797|B^8184531558|C^ANTONIOV|D^19412|E^CARDONA</Value>
               <Traces>
                     <Trace OID="000Z53TVVK0DQXI1" RunID="1" Rule="[@]"/>
               </Traces>
         </Reference>
         <Reference>
         <Value>A^ListC.C967431|B^8184531558|C^ANTONIO|E^CARDONA|F^196369974</Value>
               <Traces>
                     <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[4, 2]"/>
               </Traces>
         </Reference>
</References>
         </Identity>

  </Identities>
…
</root>
```

Figure 5: EIS "000Z53TVVK0DQXI1" after Identity Update (Run 2)

The story of how this EIS was updated can be read in the <Traces> elements in the EIS. Of the four records now in EIS "000Z53TVVK0DQXI1", two were originally in EIS "X9KTZ5GOQ5RVHOWV" created in Run 1. The other was Record B932797 that originally created EIS "000Z53TVVK0DQXI1" in Run 1. What has happened is that Record C967431 from List_C matched a record in each both of these EIS, Record A953698 by Rule 2 (name and transpose SSN) and Record B932797 by Rule 4 (name and phone). Because of this both EIS and the new record have been merged into a single EIS that has retained the label "000Z53TVVK0DQXI1". Even though there is now no longer an EIS with the label "X9KTZ5GOQ5RVHOWV" there is a record of its existence in the

<Trace> elements that record that the two records originally had that label in Run 1, but in Run 2 were merged into their current EIS.

Demonstration of Structure-to-Structure Assertion Configuration

As discussed earlier, there is a limit to accuracy of the clustering that can be obtained through the use of identity rules. Because these rules infer equivalence based on the information present in the records, they can only be as accurate as the information provided.

```
<root>
   <Metadata>
        <Modifications>
                <Modification ID="1" OysterVersion="3.2" Date="2012-03-29 04.51.07" RunScript="Run001" />
                <Modification ID="2" OysterVersion="3.2" Date="2012-03-29 07.14.44" RunScript="Run002" />
                <Modification ID="3" OysterVersion="3.2" Date="2012-04-02 21.50.55" RunScript="Run003" />
        </Modifications>
  ….
  </Metadata>
<Identity Identifier="KKJYKI0WR7JXQUAO" CDate="2012-03-29">
        <StrToStr>
                <OID>000Z53TVVK0DQXI1</OID>
        </StrToStr>
        <References>
        <Reference>
        <Value>A^ListA.A953698|C^ANTONIOV|D^247H|E^CARDONA|F^196369947</Value>
        <Traces>
                <Trace OID="X9KTZ5GOQ5RVHOWV" RunID="1" Rule="[@]"/>
                <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[2]"/>
                <Trace OID="KKJYKI0WR7JXQUAO" RunID="3" Rule="[@AssertStrToStr]"/>
        </Traces>
        </Reference>
        <Reference>
        <Value>A^ListA.A989582|C^ANTONIOV|D^5221|E^CARDONA|F^196369974</Value>
        <Traces>
                <Trace OID="X9KTZ5GOQ5RVHOWV" RunID="1" Rule="[2]"/>
                <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[2]"/>
                <Trace OID="KKJYKI0WR7JXQUAO" RunID="3" Rule="[@AssertStrToStr]"/>
        </Traces>
        </Reference>
        <Reference>
        <Value>A^ListB.B932797|B^8184531558|C^ANTONIOV|D^19412|E^CARDONA</Value>
        <Traces>
                <Trace OID="000Z53TVVK0DQXI1" RunID="1" Rule="[@]"/>
                <Trace OID="KKJYKI0WR7JXQUAO" RunID="3" Rule="[@AssertStrToStr]"/>
        </Traces>
        </Reference>
        <Reference>
        <Value>A^ListB.B949439|B^2093181443|C^ANTONIOV|D^1207|E^CARDONA</Value>
        <Traces>
                <Trace OID="KKJYKI0WR7JXQUAO" RunID="1" Rule="[@]"/>
        </Traces>
        </Reference>
        <Reference>
        <Value>A^ListC.C967431|B^8184531558|C^ANTONIO|E^CARDONA|F^196369974</Value>
        <Traces>
                <Trace OID="000Z53TVVK0DQXI1" RunID="2" Rule="[4,  2]"/>
                <Trace OID="KKJYKI0WR7JXQUAO" RunID="3" Rule="[@AssertStrToStr]"/>
        </Traces>
        </Reference>
</References>
</Identity>
…
</Identities>
</root>
```

Figure 6: EIS “KKJYKI0WR7JXQUAO” from Structure-to-Structure (Run 3)

Another type of resolution that is based on external knowledge that two records or two EIS are equivalent is called asserted resolution. Rather than inferring equivalence, the equivalence is known to be true. For example, a customer may self-report to ABC Company that he or she has moved and changed mailing addresses, thus making a connection between two different EIS for this customer. The EIIM system was not able to infer the connection because there was not enough evidence in the records to know that it was the same customer at two different addresses.

Figure 6 shows the EIS as example of where this same thing has happened in the ABC customer information. It turns out that in Run 1 there was another EIS labeled "KKJYKI0WR7JXQUAO" built from Record B949439 for the customer named "Antonio Cardona" living at an address with street number "1207". The EIS "000Z53TVVK0DQXI1" shown in Figure 5 also has the name "Antonio Cardona", but living at an address with street number "19412" (Record B932797). When Mr. Cardona reported his change of address to an ABC Customer Representative, she discovered the two different EIS in the system and set out to correct this false negative error by using an assertion.

The OYSTER system supports four types of assertions, reference-to-reference assertion, reference-to-structure assertion, structure-to-structure assertion, and structure-split assertion. In this case since both EIS (structures) already existed, the form of assertion used was structure-to-structure. The representative simply gave the command (Run 3) to OYSTER to merge EIS "000Z53TVVK0DQXI1" into EIS "KKJYKI0WR7JXQUAO". The result of that assertion is shown in Figure 6. The EIS "KKJYKI0WR7JXQUAO" now contains five records, the original Record B949439 this EIS plus the four records that were previously in EIS "000Z53TVVK0DQXI1". Again the <Trace> elements show that in Run 3, these four records were merged migrated into their current EIS by the Rule code "[@AssertStrToStr]" which is the code that indicates a structure-to-structure assertion.

Conclusion

ER has long been recognized as a key process in support of data cleaning for removing duplicate records and in data integration as a way to aggregate information for the same entity across different information sources. Typically the final step is to select one best example (survivor or exemplar record) from each cluster of equivalent records, discard the duplicate records, and pass the results to the next process. The EIIM research expands the scope of the traditional one-time ER processing to maintaining identity information and persistent identifiers. The goal of EIIM is to achieve entity identity integrity which is a key process required for MDM. EIIM research also provides practical guidance to ER and EIIM system designers. The most recent versions of OYSTER incorporate major EIIM design elements and has been successfully used to support EIIM in a number of pilot projects.

References

Huang, K.-T., Lee, Y. W., & Wang, R. Y. (1999). *Quality Information and Knowledge Management.* Prentice Hall.

Maydanchik, A. (2007). *Data Quality Assessment.* Bradley Beach, NJ: Technics Publications.

Talburt, J. (2011). *Entity Resolution and Information Quality.* Burlington, MA: Morgan Kaufmann.

Talburt, J., & Nelson, E. (2009). CoDoSA: A light-weight, XML framework for integrating unstructured textual information. *15th Americas Conference on Information Systems* (p. Paper 489). San Francisco, CA: AIS Electronic Library .

Zhou, Y., & Talburt, J. (2011). The Role of Asserted Resolution in Entity Identity Management. *The 2011 International Conference on Information and Knowledge Engineering (IKE'11).* Las Vegas, Nevada: (accepted for publication).

Zhou, Y., & Talburt, J. R. (2011). Entity Identity Information Management. *Proceedings of the 16th International Conference on Information Quality(ICIQ-11)*, (pp. 327-341). Adelaide, Australia.