IDACT – Automating Data Discovery and Compilation

Kara Nance and Brian Hay

University of Alaska Fairbanks

NASA award AIST-02-0135

Presentation Overview

- IDACT Overview
- A Very Brief Introduction to XML
- Data Transformation Manager (DTM) – Version 1
- Data Transformation Manager (DTM) – Version 2
- Example Transformation
- Data Source Registry
- Summary

NASA award AIST-02-0135

IDACT Overview

- A system that allows data consumers to easily locate, retrieve, and transform datasets from multiple data sources.

NASA award AIST-02-0135

IDACT Overview

- Incorporates intelligent automated processes to provide data access to a wider range of data consumers, with fewer data processing skills.

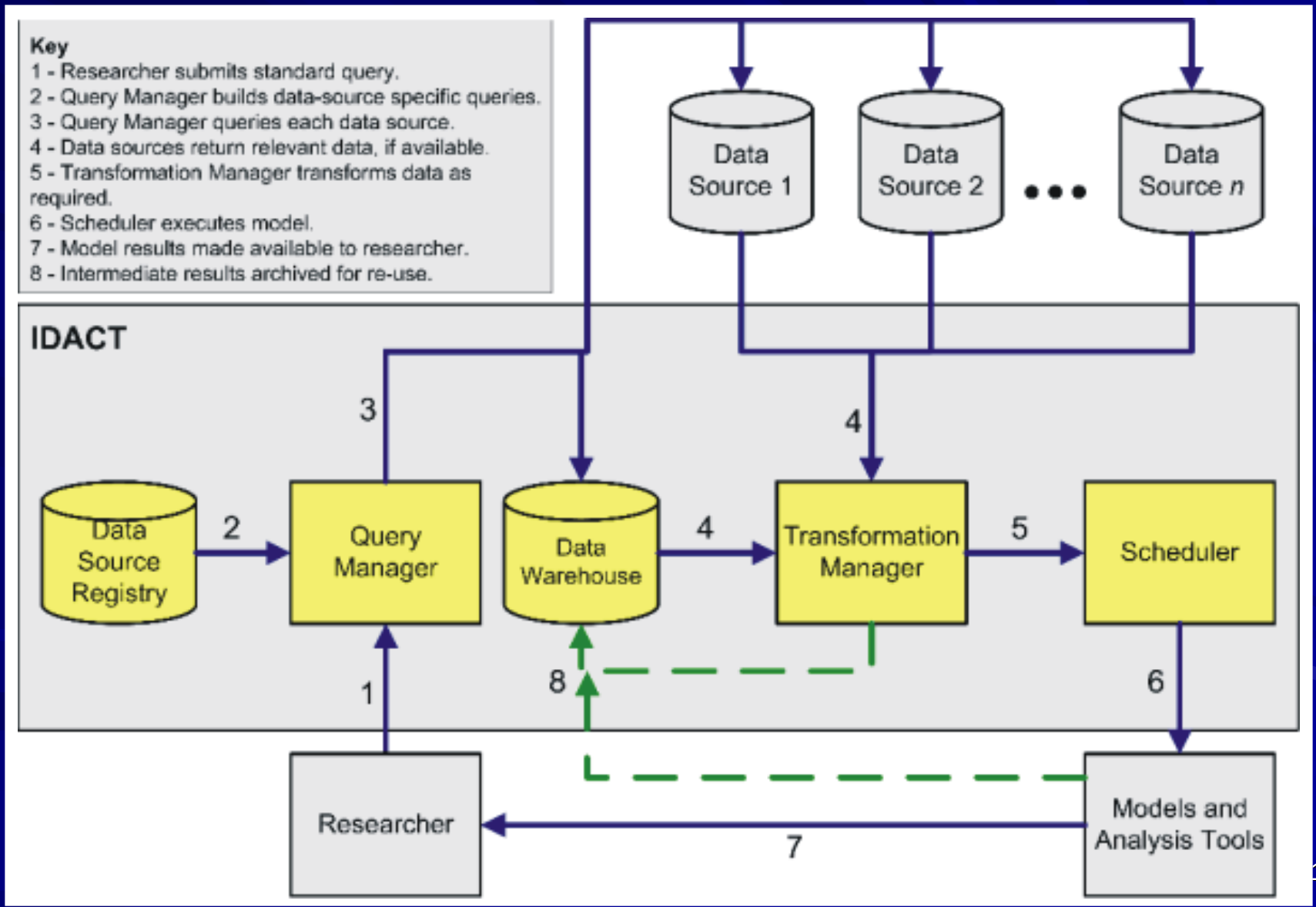
NASA award AIST-02-0135

IDACT Overview

- Built as a modular system, to ensure that it can continue to be used as technology changes in the future.

NASA award AIST-02-0135

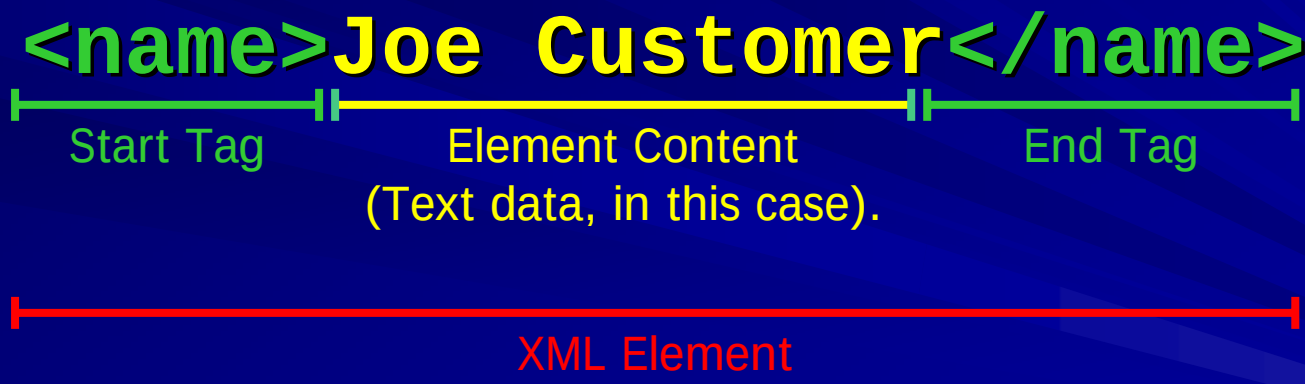
IDACT Overview



A Very Brief Introduction to XML

NASA award AIST-02-0135

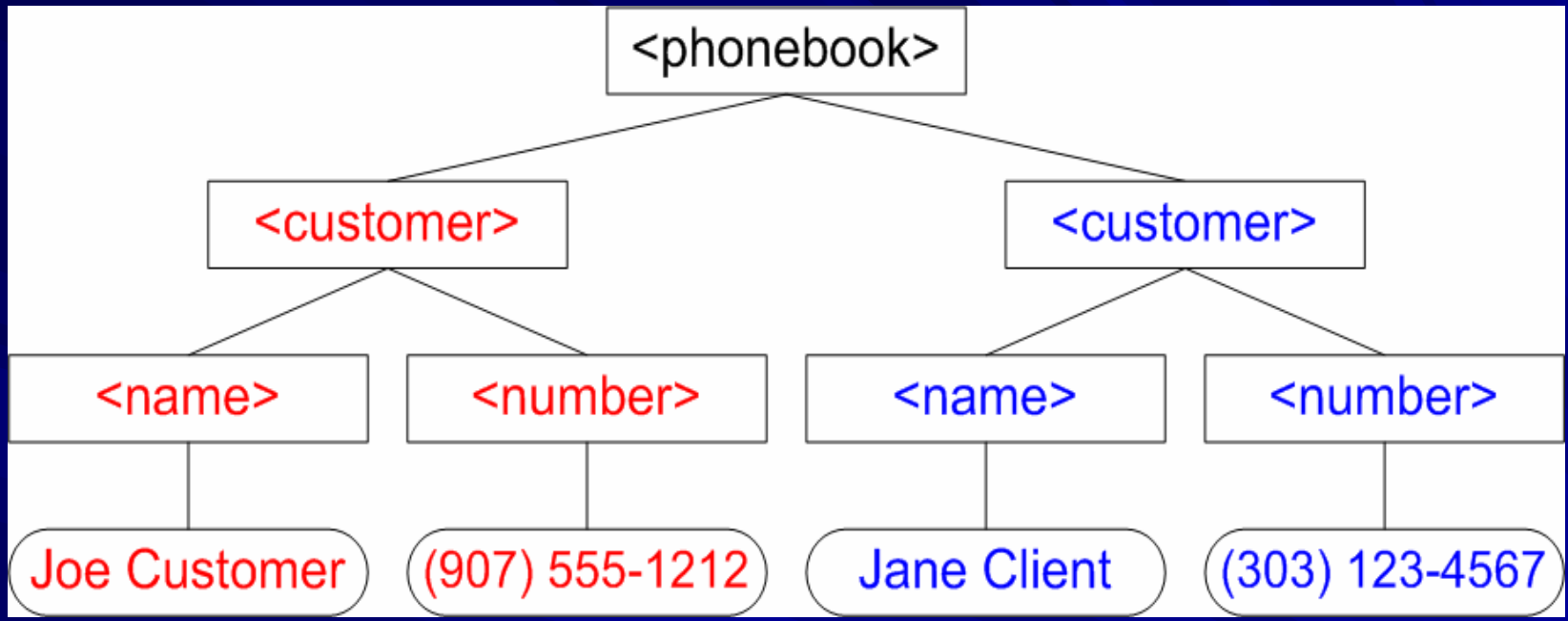
Example of a XML element



NASA award AIST-02-0135

```
<?xml version="1.0" ?>
<phonebook>
    <customer>
        <name>Joe Customer</name>
        <number>(907) 555-1212</number>
    </customer>
    <customer>
        <name>Jane Client</name>
        <number>(303) 123-4567</number>
    </customer>
</phonebook>
```


NASA award AIST-02-0135



A structure for a particular XML document can be defined using an XSD.

Remember this – it will be important!

NASA award AIST-02-0135

IDACT Data Transformation Manager (DTM)

NASA award AIST-02-0135

Step 1 – Accept Input

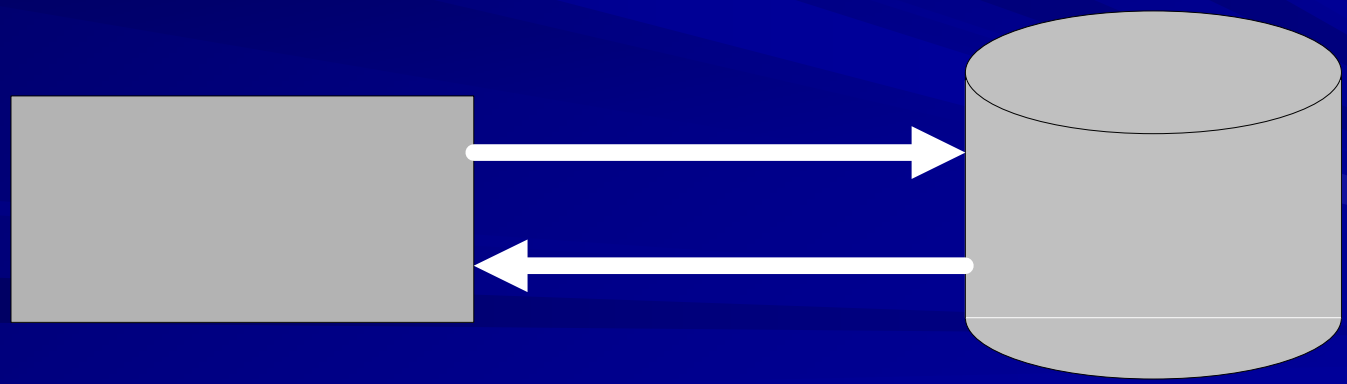
At the most basic level, the transformation process begins with a request that contains:

- The source (input) data location and data format.
- The destination (output) data location and data format.

NASA award AIST-02-0135

Step 2 – Determine Transformation

The DTM queries its transformation database to determine if a transformation, or a a sequence of transformations, exist that perform the required transformation.



NASA award AIST-02-0135

Step 3 – Acquire Source Data

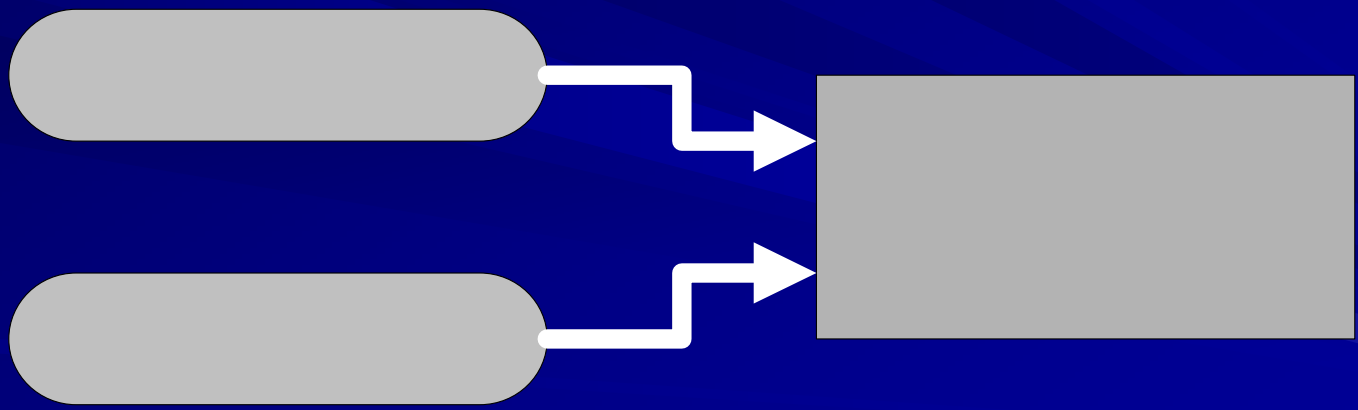
If a suitable transformation sequence is found, the DTM attempts to acquire the source data if it is not already local.

- If the source data is static (e.g. a file, or recordset from an SQL query), the source data can be acquired in its entirety.
- If the source data is a stream (e.g. a real-time data feed), a connection to the stream server is made.

NASA award AIST-02-0135

Step 4 – Apply Transformation

If the source data can be acquired, then the DTM attempts to apply the relevant transformation.

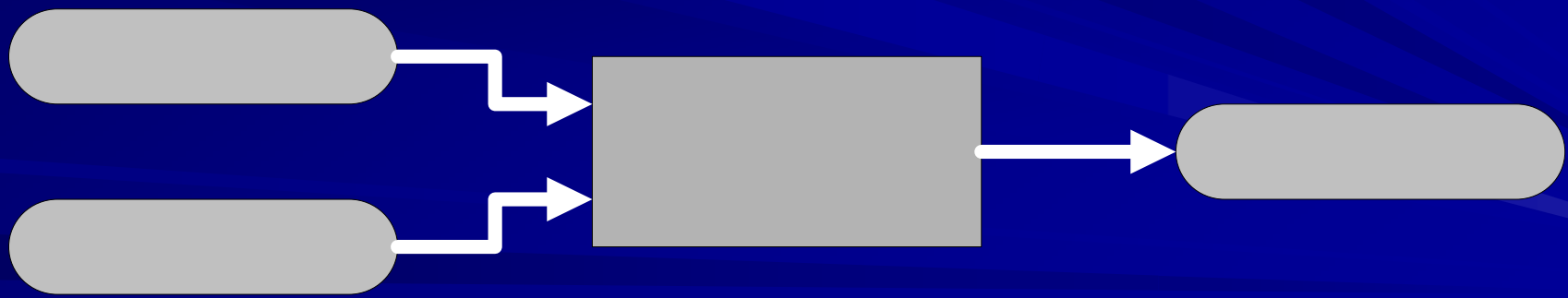


NASA award AIST-02-0135

Step 5 – Store/Transmit Output

If the transformation was successful, the transformed output data is stored or transmitted.

- If the output is static, it is stored for retrieval.
- If the output is to be transmitted, it is made available via a server.



NASA award AIST-02-0135

Optional Input

- In addition to the input already specified, the DTM also accepts some addition input that specify actions to be performed in certain circumstances.
  - For example, an email can be sent after a successful transformation, indicating that the output is available on a given (S)FTP server, or at a web address.
  - Alternatively, an email can be sent in the event of an error during the transformation process, indicating the details of the error.
  - Other actions, such as sending notifications to the IDACT scheduler, are also being developed.

NASA award AIST-02-0135

DTM Current Status

- The DTM version 1 is currently implemented in Java, allowing cross-platform usage.
- The DTM is implemented as a modular framework. This allows
  - updates such as new data formats and new action types to be easily added.
  - components to be re-written in higher performance languages, such as C, if necessary.

NASA award AIST-02-0135

DTM Version 2

- Version 1 of the DTM operates using transformations defined in the transformation database. If a suitable transformation is not found in the database, the DTM cannot transform the data.
- Version 2 of the DTM allows users to add transformations and define new transformations in the event that a suitable transformation sequence cannot be found in the transformation database.

NASA award AIST-02-0135

Adding new transformations

- An existing transformation can be added to the DTM database via a web based interface.
- In this example, a new XSLT transformation is being added, which will be stored in the DTM database and made available to users.

Adding new transformations

- Version 2 of the DTM can also construct new XSLT transformations.
- These newly constructed transformations are then stored in the DTM database, and made available for future users.
- Much of this construction process is automated, although the user remains in control and can approve, reject, or modify the suggestions made by the DTM.

NASA award AIST-02-0135

Creating XSLT transformations

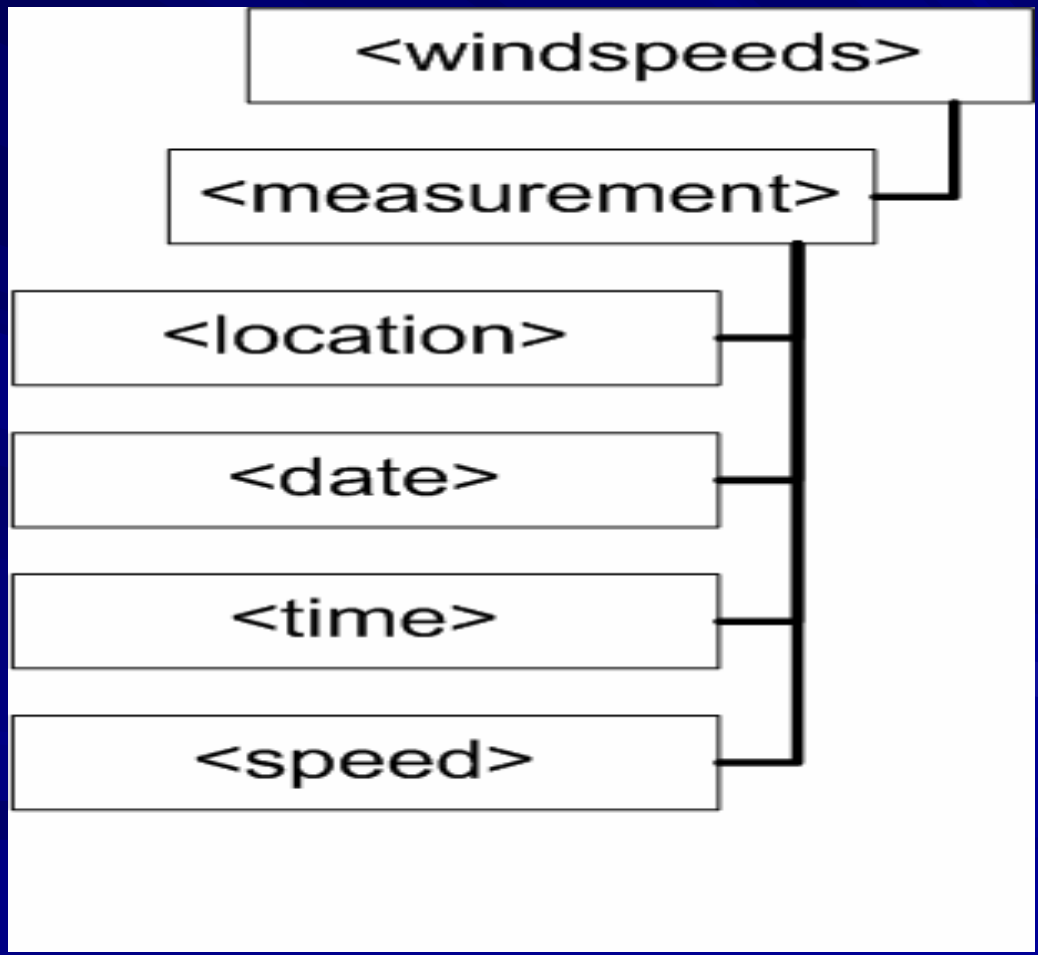
- The DTM starts with the XML Schema Language (XSD) definitions of the input and output formats.
- If such definitions do not exist, they can be created automatically from example input and output data.
- The DTM uses these format definitions to build an internal description of the input and output formats.

NASA award AIST-02-0135

Example
Creating XSLT transformations

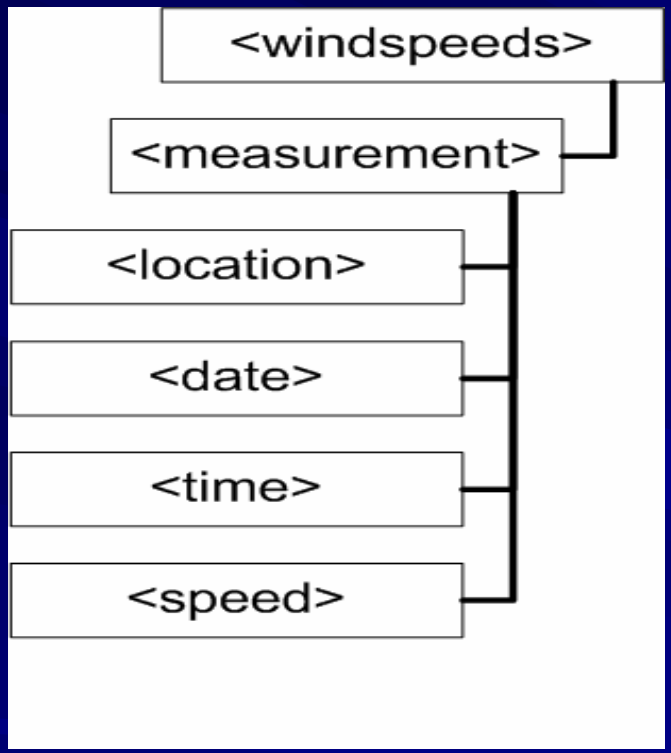
NASA award AIST-02-0135

Example: Creating XSLT transformations



NASA award AIST-02-0135

Example: Creating XSLT transformations



MEASUREMENT : Table

SENSOR	TIME	TYPE	VALUE	UNITS
Windspeed A	1/2/2004 1:20:00 PM	windspeed	20	mph
Windspeed B	1/2/2004 1:25:00 PM	windspeed	40	kph
Windspeed A	1/2/2004 1:30:00 PM	windspeed	29	mph

Record: 4 of 4

NASA award AIST-02-0135

Creating XSLT transformations

MEASUREMENT : Table

SENSOR	TIME	TYPE	VALUE	UNITS
Windspeed A	1/2/2004 1:20:00 PM	windspeed	20	mph
Windspeed B	1/2/2004 1:25:00 PM	windspeed	40	kph
Windspeed A	1/2/2004 1:30:00 PM	windspeed	29	mph

Record: 4 of 4

NASA award AIST-02-0135

Creating XSLT transformations

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
  xmlns:xsi="http://www.w3.org/2000/10XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="MEASUREMENT.xsd">

  <MEASUREMENT>
    <SENSOR>Windspeed A</SENSOR>
    <TIME>2004-01-02T13:20:00</TIME>
    <TYPE>windspeed</TYPE>
    <VALUE>20</VALUE>
    <UNITS>mph</UNITS>
  </MEASUREMENT>

  <MEASUREMENT>
    <SENSOR>Windspeed B</SENSOR>
    <TIME>2004-01-02T13:25:00</TIME>
    <TYPE>windspeed</TYPE>
    <VALUE>40</VALUE>
    <UNITS>kph</UNITS>
  </MEASUREMENT>

  <MEASUREMENT>
    <SENSOR>Windspeed A</SENSOR>
    <TIME>2004-01-02T13:30:00</TIME>
    <TYPE>windspeed</TYPE>
    <VALUE>29</VALUE>
    <UNITS>mph</UNITS>
  </MEASUREMENT>
</dataroot>
```

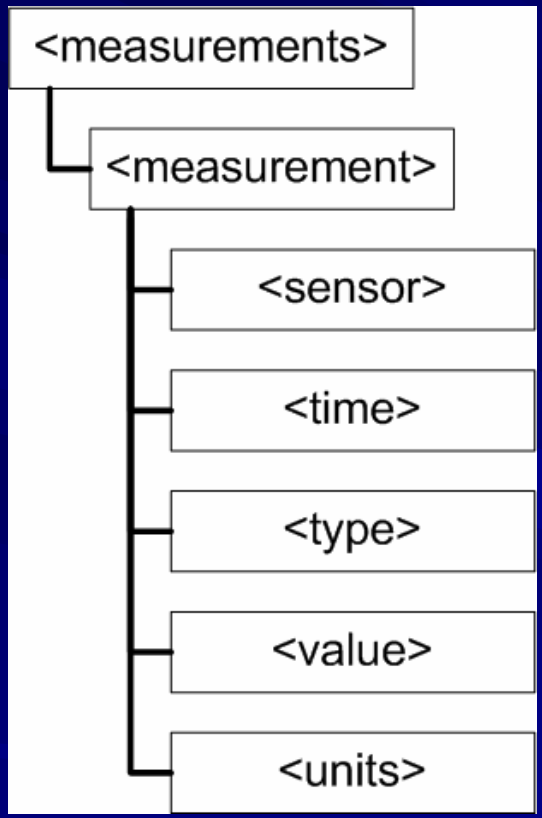
MEASUREMENT : Table

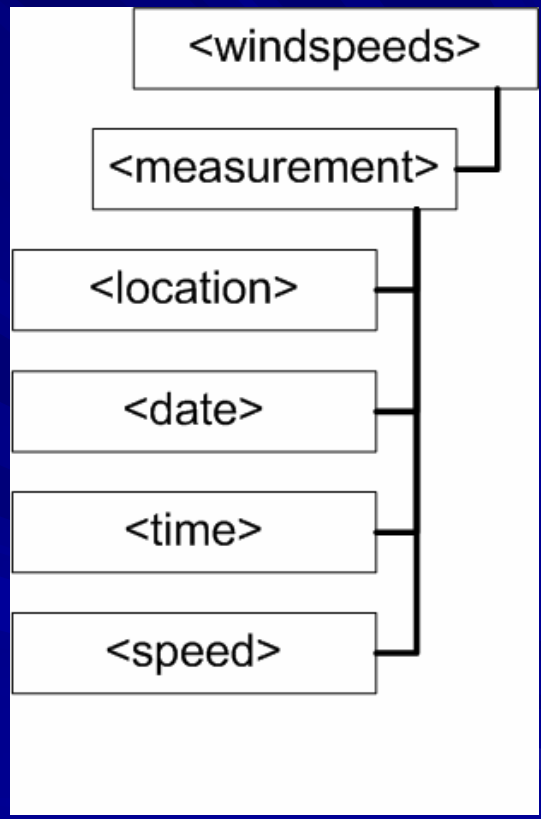
SENSOR	TIME	TYPE	VALUE	UNITS
Windspeed A	1/2/2004 1:20:00 PM	windspeed	20	mph
Windspeed B	1/2/2004 1:25:00 PM	windspeed	40	kph
Windspeed A	1/2/2004 1:30:00 PM	windspeed	29	mph

Record: 4 of 4

NASA award AIST-02-0135

Creating XSLT transformations

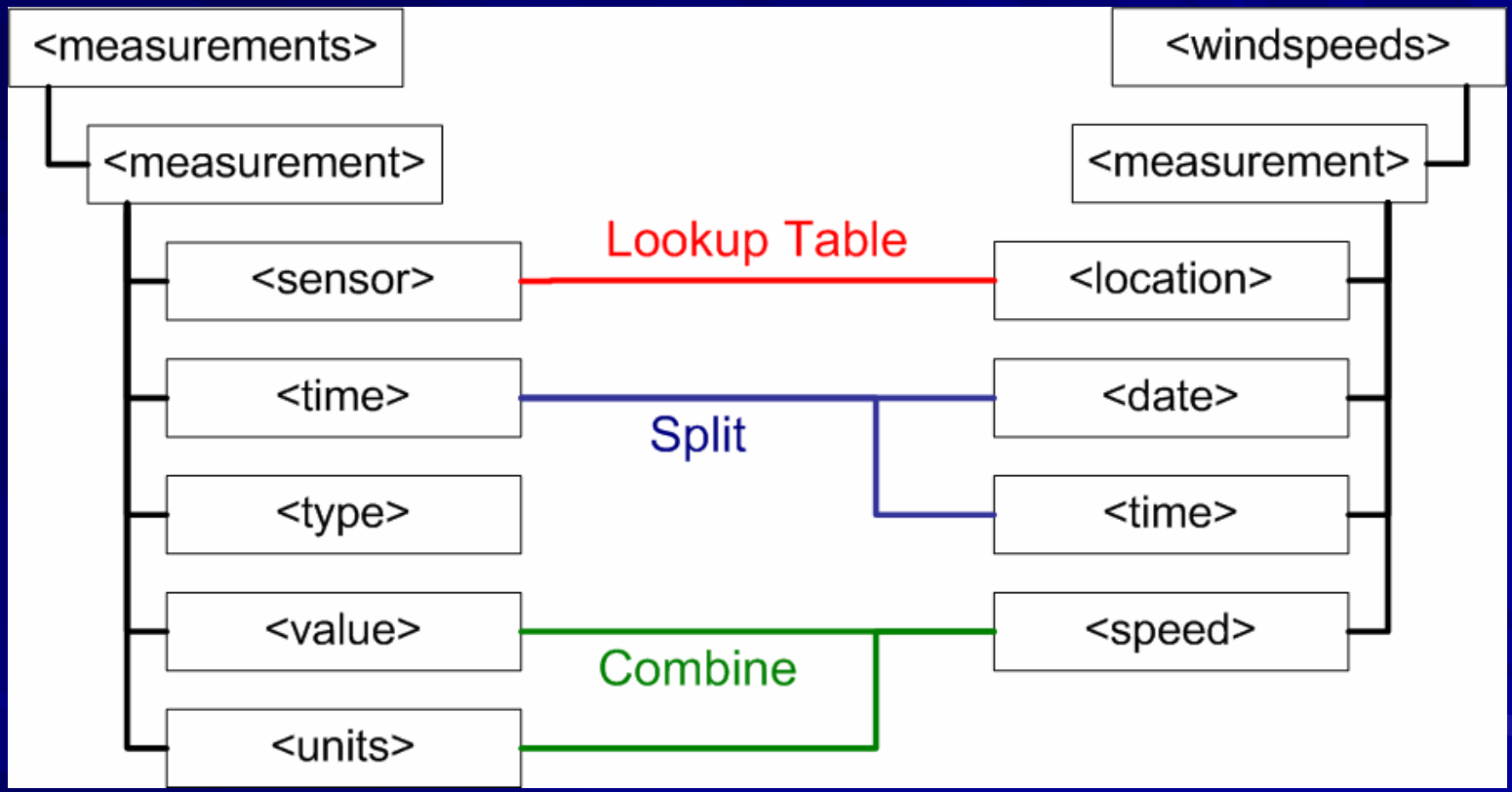




Create a description of input and output formats.

NASA award AIST-02-0135

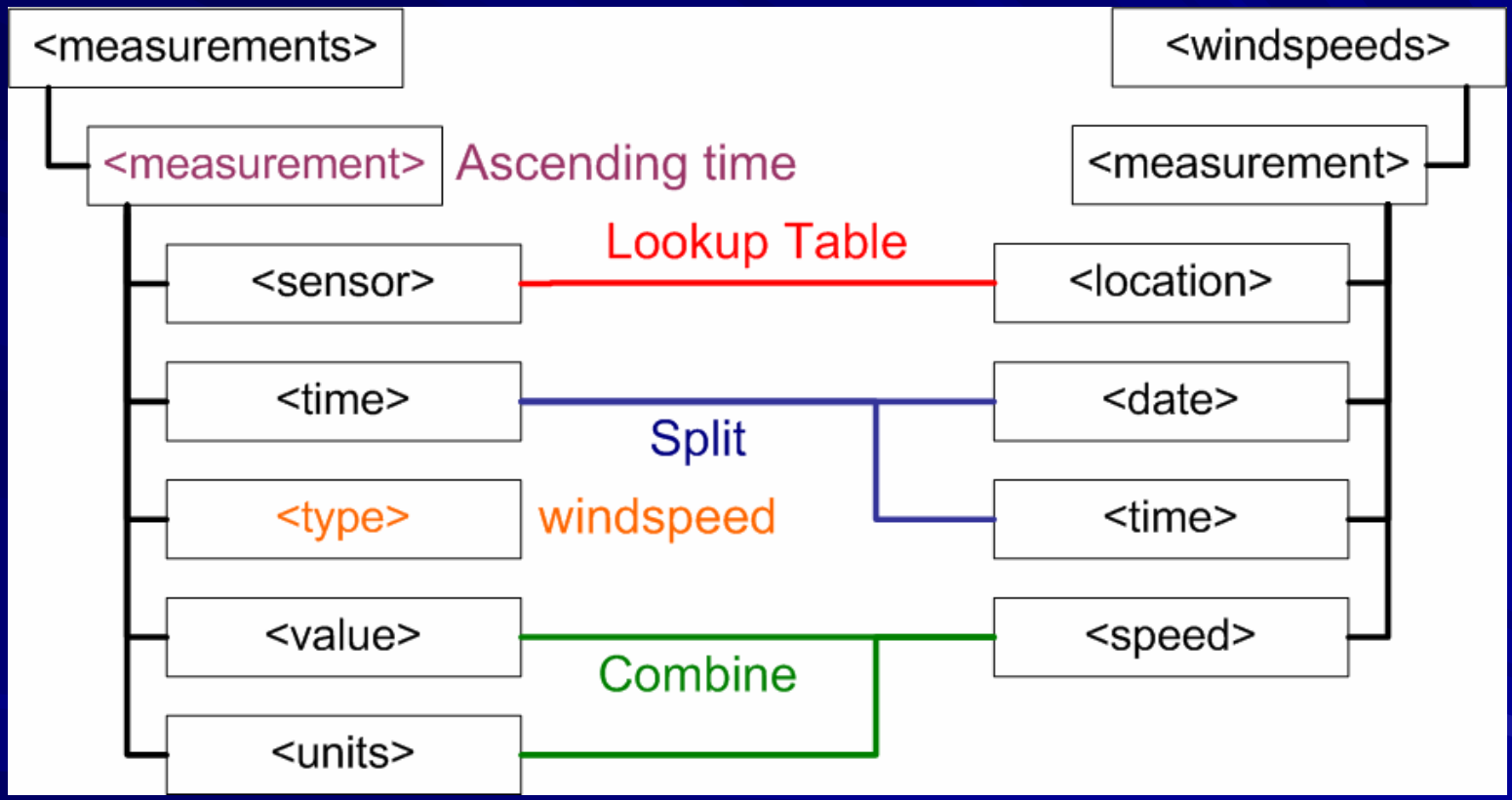
Creating XSLT transformations



Define associations

NASA award AIST-02-0135

Creating XSLT transformations



Define sort and selections where necessary.

NASA award AIST-02-0135

Creating XSLT transformations

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:output method="xml" />

   <xsl:template match="/">
      <xsl:apply-templates select="MEASUREMENTS" />
   </xsl:template>

   <xsl:template match="MEASUREMENTS">
      <xsl:element name="WINDSPEEDS" >
         <xsl:apply-templates select="MEASUREMENT" >
            <xsl:sort select="./TIME" order="ascending" />
         </xsl:apply-templates>
      </xsl:element>
   </xsl:template>

   <xsl:template match="MEASUREMENT">
      <xsl:if test="./TYPE = 'Windspeed'">
         <xsl:element name="MEASUREMENT" >
            <xsl:apply-templates select="TIME" />
            <xsl:apply-templates select="VALUE" />
         </xsl:element>
      </xsl:if>
   </xsl:template>

   <xsl:template match="VALUE">
      <xsl:element name="SPEED" >
         <xsl:value-of select="." />
         <xsl:value-of select="../UNITS" />
      </xsl:element>
   </xsl:template>
```

```
   <xsl:template match="TIME">
      <xsl:element name="DATE">
         <xsl:call-template name="FormatDatetimeToDate">
            <xsl:with-param name="datetime" select="." />
         </xsl:call-template>
      </xsl:element>
      <xsl:element name="TIME">
         <xsl:call-template name="FormatDatetimeToTime">
            <xsl:with-param name="datetime" select="." />
         </xsl:call-template>
      </xsl:element>
   </xsl:template>

   <xsl:template name="FormatDatetimeToTime">
      <xsl:param name="datetime" />
      <xsl:value-of select="substring(substring-after($datetime, 'T'), 1, 8)" />
   </xsl:template>

   <xsl:template name="FormatDatetimeToDate">
      <xsl:param name="datetime" />
      <xsl:variable name="date">
         <xsl:value-of select="substring-before($datetime, 'T')" />
      </xsl:variable>
      <xsl:variable name="d1">
         <xsl:value-of select="concat(substring-before($date, '-'), '/', substring-after($date, '-') )" />
      </xsl:variable>
      <xsl:variable name="d2">
         <xsl:value-of select="concat(substring-before($d1, '-'), '/', substring-after($d1, '-') )" />
      </xsl:variable>
      <xsl:value-of select="$d2" />
   </xsl:template>

</xsl:stylesheet>
```


NASA award AIST-02-0135

XSLT components

- The XSLT file is created from a library of small XSLT components, which are combined to build the XSLT file that will be used for the transformation.
- If a user creates a new XSLT component, it is added to the XSLT component library, and can be used to create new XSLT transformation files.

NASA award AIST-02-0135

Creating XSLT transformations

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:output method="xml" />

   <xsl:template match="/">
      <xsl:apply-templates select="MEASUREMENTS" />
   </xsl:template>

   <xsl:template match="MEASUREMENTS">
      <xsl:element name="WINDSPEEDS" >
         <xsl:apply-templates select="MEASUREMENT" >
            <xsl:sort select="./TIME" order="ascending" />
         </xsl:apply-templates>
      </xsl:element>
   </xsl:template>

   <xsl:template match="MEASUREMENT">
      <xsl:if test="./TYPE = 'Windspeed'">
         <xsl:element name="MEASUREMENT" >
            <xsl:apply-templates select="TIME" />
            <xsl:apply-templates select="VALUE" />
         </xsl:element>
      </xsl:if>
   </xsl:template>

   <xsl:template match="VALUE">
      <xsl:element name="SPEED" >
         <xsl:value-of select="." />
         <xsl:value-of select="../UNITS" />
      </xsl:element>
   </xsl:template>
```

```
   <xsl:template match="TIME">
      <xsl:element name="DATE">
         <xsl:call-template name="FormatDatetimeToDate">
            <xsl:with-param name="datetime" select="." />
         </xsl:call-template>
      </xsl:element>
      <xsl:element name="TIME">
         <xsl:call-template name="FormatDatetimeToTime">
            <xsl:with-param name="datetime" select="." />
         </xsl:call-template>
      </xsl:element>
   </xsl:template>

   <xsl:template name="FormatDatetimeToTime">
      <xsl:param name="datetime" />
      <xsl:value-of select="substring(substring-after($datetime, 'T'), 1, 8)" />
   </xsl:template>

   <xsl:template name="FormatDatetimeToDate">
      <xsl:param name="datetime" />
      <xsl:variable name="date">
         <xsl:value-of select="substring-before($datetime, 'T')" />
      </xsl:variable>
      <xsl:variable name="d1">
         <xsl:value-of select="concat(substring-before($date, '-'), '/', substring-after($date, '-') )" />
      </xsl:variable>
      <xsl:variable name="d2">
         <xsl:value-of select="concat(substring-before($d1, '-'), '/', substring-after($d1, '-') )" />
      </xsl:variable>
      <xsl:value-of select="$d2" />
   </xsl:template>

</xsl:stylesheet>
```


NASA award AIST-02-0135

Datasource Registry

NASA award AIST-02-0135

What is a Datasource Registry?

- Datasource
  - A database or any other organized source of data.
- Registry
  - Keeps track of the datasources and information about them
  - Keeps track of relationships between datasources

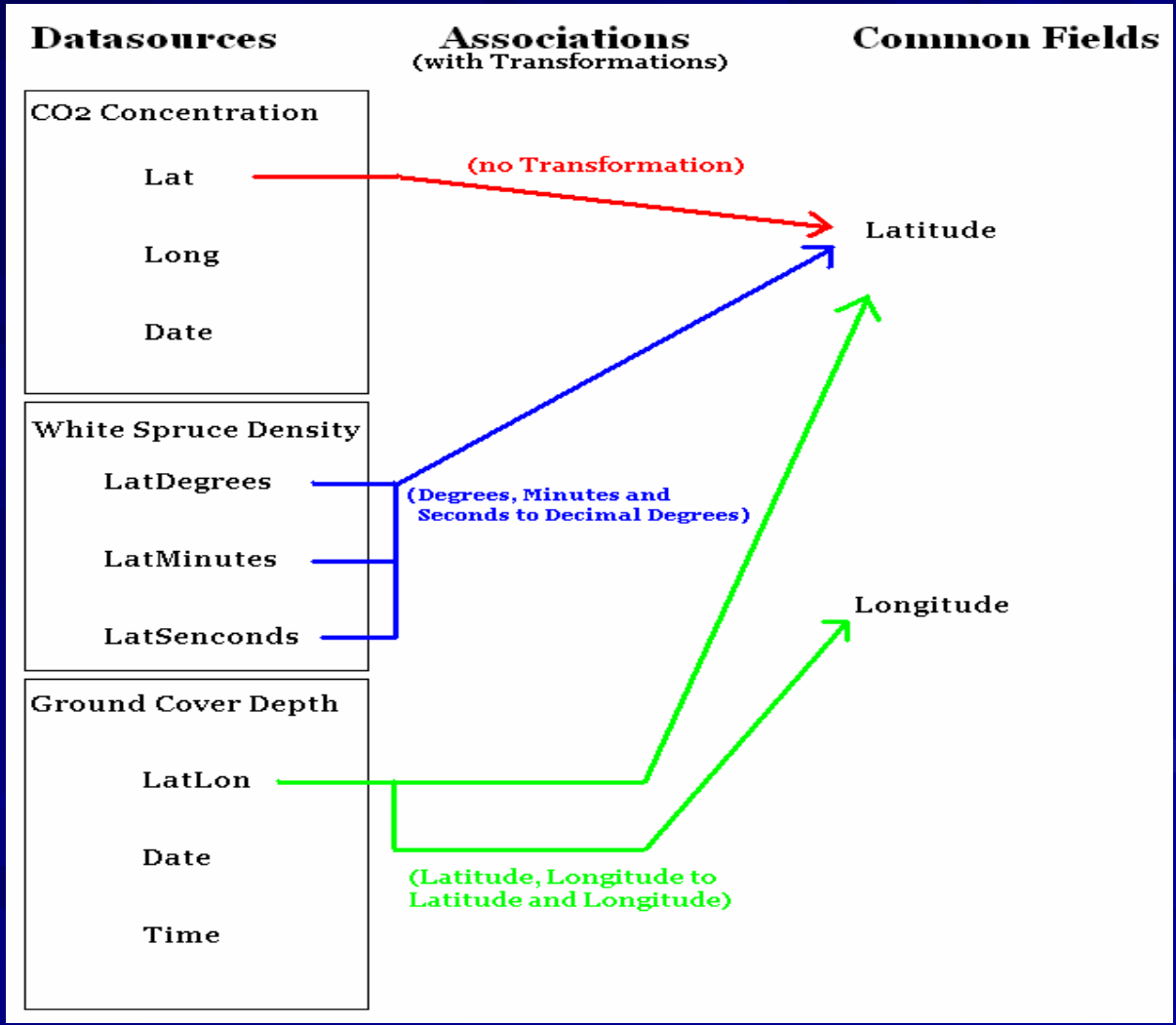
NASA award AIST-02-0135

How it works

- A user submits a datasource by entering information about it into a form
- Relationships to other datasources are described by creating “associations”
- “Associations” are created to relate fields within datasources to “common fields”
- Sometimes these “associations” need a “transformation” to format the field to the format of the “common field”

NASA award AIST-02-0135

How it works (part 2)



NASA award AIST-02-0135

Conclusion

NASA award AIST-02-0135