Ingesting and maintaining data

Scope note

Much of this documentation remains the same across VIVO releases, but some may not have been fully updated to the most recent release. While we will attempt to identify and alert you in such cases, please be aware that your VIVO may look and act slightly differently from what is represented here.

Introduction

This document provides an overview of the data ingest process for VIVO.

- The early sections describe some typical data sources for VIVO and the challenges often associated with multiple values, representing information that is only true for certain periods of time, and the like.
- Later sections describe different technical and workflow options for loading data into VIVO.
- Some more specific examples are provided, but readers should expect to have to modify or extend examples to reflect their local data needs, the format of sources, and the depth of technical skills available to them, such as the ability to write or modify XSLT scripts

Other sources of information

- Karma: A Data Integration Tool

How to plan data ingest for VIVO

How VIVO differs from a spreadsheet, where VIVO data typically comes from, cleaning data prior to loading, matching against data already in VIVO, and doing further cleanup once it's in VIVO

- Ingest tools: home brew or off the shelf? — Major options including the Harvester, semantic ingest tools such as Karma, and XSLT
- Typical ingest processes — Alternative approaches to ingest and making ingest repeatable
- Challenges for data ingest — Challenges in the data, in workflow, in working incrementally, in modeling, and in migration
- Monitoring for quality

VIVO Harvester

- University of Florida Harvester Team
- University of Florida Harvester Documentation Archive
    - Development and Planning
        - Harvester Planned Features
        - Version 1
            - Milestone 1
            - Milestone 2
            - Milestone 3
            - Milestone 4
            - Milestone 5
            - Milestone 6
            - Milestone 7
            - Milestone 8
            - Milestone 9
    - Typical harvest
    - Scheduling
    - Problems and Solutions
    - Web Of Science visual map
    - XSLT Mapping
        - Citation microformat to VIVO XSL Information
        - HCalendar Microformat to VIVO XSL Information
        - HCard Microformat to VIVO XSL Information
        - HGrant Microformat to VIVO XSL Information
        - HResume Microformat to VIVO XSL Information
        - IP to VIVO XSL Information
        - OAI__DC to VIVO Information
        - PubMed to VIVO XSL Information

        - Rel-tag Microformat to VIVO XSL Information
        - XPath
        - XPathTool
    - Harvester Score Ontology
    - Harvester Documentation Procedures
    - New Harvest Workflow Proposal
    - Demonstrations and Examples
        - 20110114 UF Harvester Training
        - CourseIngest Reproducible Harvest Installation Procedure
        - Course Ingest Reproducible Harvest Installation Procedure
        - Harvester 1.0 Demo
        - Image Ingest
        - ImageIngest Reproducible Harvest Installation Procedure
        - IP Example Script
        - JDBC Example Script
        - MeSH Terms to VIVO XSL Information
        - MODS Example Script 1.2
        - Peoplesoft
        - Peoplesoft-Biztalk Reproducible Harvest Installation Procedure
        - Peoplesoft Example Script 1.2
        - Pubmed Example Script
        - Pubmed Example Script (1.1.1)
        - Pubmed Example Script 1.2
        - Scopus
        - UF Data Sources
            - UF New Weekly Publications RSS Feed
            - University of Florida PeopleSoft Harvest
            - Division of Sponsored Research
                - DSR Reproducible Harvest Functional Specification
                - DSR Reproducible Harvest Installation Procedure
                - DSR Reproducible Harvest Project Charter
                - DSR Reproducible Harvest Technical Specification
                - DSR to VIVO XSLT example
                - UF Grant Data to VIVO XSL Information
                - University of Florida Department of Sponsored Research Harvest
                - University of Florida DSR Grants visual map
                - Cornell University Grant Data model
            - University of Florida PubMed Harvest
            - HR Data to VIVO XSL Information
        - Web Of Science Example Script
    - Env
    - Configuration
    - Harvester .tar file
    - Harvester Debian package
    - Harvester in Eclipse
    - Deprecated Harvester Documentation
        - Deprecated__Configuration
        - Deprecated__Controller
        - Deprecated__I O
        - Deprecated__VIVO Harvester SVN Checkout
        - VIVO Harvester User Guide 1.0
    - Harvester vivo configuration file
    - Harvester Source Documentation
        - Diff
        - Fetch — The first step of a typical harvest is the get you data from your target source. We call this the Fetch. For example, let us suppose we have a VIVO installation containing researchers at our university, and we want to harvest from Pubmed http://www.ncbi.nlm.nih.gov/pubmed/ information on publications written by researchers at our university. In this case we would use Harvester's PubmedFetch tool to send a query off to Pubmed, which will return the results of that query to us in its own XML for
            - CSVtoJDBC
            - D2RMapFetch
            - Design of NLMJournalFetch
            - JDBCFetch
            - NIHFetch
                - NLMJournalFetch
            - OAIFetch
            - PubmedFetch
                - Design of PubmedFetch
            - ScopusFetch
            - SOAPMessenger
            - WOSFetch
        - Harvester Architecture Diagram
        - Merge
        - Qualify
        - RecordHandler
        - RenameResources
        - RunBibutils
        - Score — Depending on your data the next step may be to match incoming data with data already in VIVO. For example, if you have just pulled in some publication information from Pubmed, you might want to compare the author names with people in

your VIVO, so that you can link the publications with the authors. This comparison is done via the tool, which compares any values you want between VIVO and the input data, and assigns a number to the comparison.
        - Algorithm
            - CaseInsensitiveInitialTest
            - EqualityTest
            - NameCompare
            - NormalizedDamerauLevenshteinDifference
            - NormalizedDoubleMetaphoneDifference
            - NormalizedLevenshteinDifference
            - NormalizedSoundExDifference
            - NormalizedTypoDifference
    - Smush
    - Translate — The next step of a typical harvest is the translation. The fetched data will be in its own format, and this needs to be converted into VIVO-compatible triples. If the input is an XML format, this can be done using the XSLTranslator tool and a .xsl file containing XSLT code specific to the data format being converted to RDF/XML triples. Included with Harvester in the config /datamaps/ directory are several pre-written XSLT files for frequently-needed formats (including for example Pubmed). Anothe
        - GlozeTranslator
        - RDFTranslator
        - VCardTranslator
        - XSLTranslator
    - Transfer
    - XMLGrep
    - Utilities
        - ArgParser
        - ChangeNamespace — Depending on how your data came in and how you generated triples for it the last step before importing the information into VIVO is to give your data proper URIs via the tool. Prior to this step, URIs may be placeholders provided by the XSLT translation (typically using aspects of the raw data that are expected to be unique, such as an ISBN number) or blank nodes from a SPARQL Construct. If you've generated unique URI's for all of your data using a piece of unique information then you can skip
        - DatabaseClone
        - JenaConnect
            - Jena RDF Model
    - Harvester Tools
    - Match — The Match tool will look at the numbers generated by Score and compare them to a threshold value. Input entities compared by Score that meet or exceed the threshold will have their identities changed to the URI of the person in VIVO, so that when the data is finally pulled into VIVO the new data will be linked to existing data. In this way you can fetch publications for your existing researchers.

Beyond the basics of data ingest: more about tools and techniques

- Name disambiguation and entity resolution
- Advanced PubMed name matching diagram
- Alternative converters from tabular data to RDF
- Ingest Workflow Language
- VIVO PHP Person Data Library

Data ingest guides and workshop materials

- 2011 Conference Workshop on Extended Ingest by Example
- 2012 VIVO Conference workshop: Survey of VIVO Data Ingest Methods
- VIVO 1.2 Data Ingest Guide
- A Generalizable, XSLT Based RDF Ingest Example
    - XSLT Ingest Example: Source Data
    - XSLT Ingest Example: Accumulator Classes
    - XSLT Ingest Example: The Process
    - XSLT Ingest Example: Gather
    - XSLT Ingest Example: Count
    - XSLT Ingest Example: Make URIs
    - XSLT Ingest Example: Create New Persons and Organizations
    - XSLT Ingest Example: Fill in URPs and UROs
    - XSLT Ingest Example: Create RDF
    - XSLT Ingest Example: Final Considerations
    - XSLT Ingest Example: Appendix A
    - XSLT Ingest Example: Appendix B
    - XSLT Ingest Example: Appendix C
    - XSLT Ingest Example: Appendix D
    - XSLT Ingest Example: Appendix E
    - XSLT Ingest Example: Appendix F
- Faculty affiliation ingest example

Populating VIVO from Activity Insight (Digital Measures)