Data Analysis Principles of the Federated Master Data Management Process

This provides some background on many of the challenges in orchestrating the MDM process when defining data domains a Federated data management system.

It is common for organizations to have duplicate information on different systems. For example, student information could be stored in both an SIS and Learning Management System (LMS). As more systems are brought on line, more data is duplicated. Disparate systems aren't necessarily a bad thing but are often a sign that an organization allows groups to acquire systems that meet their needs.

To solve this, organizations build adaptors to extract and transform the data to keep the myriad of systems up to date. This can be done gradually with small subsets of fields and gradually expanded. This process is known as Data Integration or DI. Anyone that has had even limited IT experience understand the pitfalls that follow when trying to keep a company's DI process in check. One of the biggest problems is knowing where the truth for any given record is since it is stored on multiple systems. A truth record for a customer is often spread across multiple systems. This truth record or master record is also known as the Master Data Record or Golden Record.

In terms of storing/accessing data, YOUnite is a hybrid solution that handles data by either:

- Storing it in the YOUnite Data Store
- Accessing data stored in other systems through YOUnite Adaptors - This is known as federated MDM

The data analysts and architects attempt to create a universal schema (data domain or domain) that will work for all systems. For example, if there are ten different applications using a student record the data architect would create a "student" domain that will work for all all ten applications. This is not an easy task and includes analysis techniques and DI/MDM features; many of which will be touched on here.

To further clarify, the YOUnite Data Store that holds domain data in an organization (example domains include students, courses, course-sections, faculty, etc) where federated domains references the data where it lives (College SIS, LMS, Registration, etc.) and extracts/updates it as needed based on the permission of the entity making the request.

The terms MDM, DI and Master Data get used a lot and need clarification:

- MDM is the *process* of describing and cataloging data inside of an organization and understanding which stakeholders value which sources of data.
- DI is the process of integrating disparate data. This ranges from annual CSV exports and imports between systems to real-time connectors between systems.
- Master Data is what is considered the source of truth for a given data domain. See "Establish the Truth" below.

Making DI & MDM Easy is Generally Impossible

However, YOUnite's primary focus has been to make this process as easy and non-intrusive as possible.

Start by Analyzing the Use Cases

If you start by analyzing the data and building data dictionaries of all the systems you plan to work with MDM (source systems) you will quickly feel like you are trying boil the ocean. You will be adding to an already exceedingly arduous process of normalizing data -- by analyzing data that isn't relevant to your MDM process the time to complete the data analysis phase can grow exponentially.

Generally its best to lead with the use cases and limit the initial deployment to just a few use cases and gradually connect more and more of the organization's ecosystem to MDM. Use cases often equate to storyboarding but keep in mind, this is not application storyboarding but data synchronization and notification storyboarding. We want the stakeholders of the applications and data in the organization to specify their realtime needs for "the truth. This includes the following:

- What are the source systems tied to the use cases?
- Who are the stakeholders for the use cases e.g. Data and Application Architects, Business Managers, etc.?
- How do the source systems connect to their data
- What data elements in the source systems matter to the stakeholders.
    - Start building **data dictionaries** of how the various source systems model the data
    - **Stakeholder descriptions:** For each stakeholder, describe the systems where the "truth" data elements live (see next step) and what notifications they need to receive
- Data synchronization and notification storyboarding. We want the stakeholders of the applications and data in the organization to specify their realtime needs for "the truth. This includes descriptions of how the data will be used and which applications need to be notified when changes occur.

From the data dictionaries and stakeholder descriptions a clear picture starts to take shape for:

- Data domains
- Adaptor development and capabilities
- Governance

Establish the Truth

Out of analysis you discover the truth i.e. which systems hold the truth values for a given domain. As you catalogue the data elements in a data dictionary it is important to note which systems hold the truth for the various stakeholders (zones) since knowing this reduces the amount of analysis required by creating a minimum possible set of data elements for a given data domain. Its also important to understand that different zones can have a different take on which systems hold the truth values for a given domain; this too must be documented as data elements for a given data domain are catalogued. Allowing different zones to have a different perspective of where the truth originates is one of the distinguishing features of YOUnite.

As the data governance staff works through the process of MDM, truth is often defined by the DGS but YOUnite provides the flexibility that allows the ZDS to define effective federated master data. In other words, "what may be truth for one zone or, the organization as a whole(what is defined as master data by the DGS) may not be master data for another."

Example: In a college system, the truth for the “name” elements (first, last, etc) for the student attribute, is stored in both the College Application system and the College’s Student Information System (SIS). A learning management system (LMS) at college system should receive name and email address updates when made in the College Application system or SIS but, the converse is not true i.e. the College Application system and SIS do not want student name changes made from the LMS (since name changes made at the college should only be handled by staff with the appropriate permissions to do so).

Knowing this, you no longer need to worry about what issues may arise from sending data from the LMS to other systems and focus primarily on how data will flow from either the Application System or SIS into other systems such as the LMS.

Think in Terms of REST

Asking use case questions in terms of RESTful operations (HTTP GET, PUT, POST and DELETE -- following REST principles) can help keep focus on what can become a very convoluted process of analysis -- if the analysis deviates from this it almost always leads to paralysis. Ultimately YOUnite breaks transactions down into RESTful operations and if you know which operations to avoid then a lot of time can be saved.

Example: The College Application systems never wants to delete a student once they have been added to the system. Since this is the case, analysis for the DELETE request can be ignored with this application.

The MDM Process is a Multi-Dimensional Cross-Cutting Concern

There is no way around it, you must analyze:

- The needs of performing specific operations within each system
- Attributes stored in those systems and their data elements

...for each of the required HTTP operations (GET, PUT, POST, DELETE) in a RESTful context.

This will uncovers most of the challenges and meta data needed (metadata is data that is not part of the data record or that you hoped you would not have to add to the data record but is required to properly store the data).

Example: Incoming freshmen at a college need to take an assessment test to determine which English and Math courses they should be placed into. The assessment holds raw test scores and the SIS system wants to combine the assessment scores with past college and high school course scores from the student’s transcripts and then, create its own score. In other words, the SIS wants the assessment tests but it does not store the assessment test scores - it only uses them as a function of creating a course placement ranking.

Adaptors are DI custom software that connect the application (e.g. SIS, Assessment, etc) to the MDM system. They map data domains (and metadata) to operations in the application and follow protocols about data transformation and data governance i.e. who can see/update what (YOUnite provides fine-grained data governance controls between groups inside an organization).

It is easiest to think in the following terms and build "Data Domain Worksheets" as follows:

DELETE or GET or POST Entity -> {adaptor1, adaptor2...adaptorN}

PUT Entity?attribute=key&value=value -> {adaptor1, adaptor2...adaptorN}

Ultimately, the data architects create a worksheet that contains the required attributes to complete an operation for a given entity for a given adaptor.

Eventhough Data Domains Can Be Modeled as Multi-Demensional Doesn't Mean They Should Be

The JSON modeling tool with YOUnite is very powerful in that it allows a data architect to create very complex inter-dependencies between data domains but this should be avoided. However, when designing data domains, relational database principles should be followed. The following points illustrate a few pitfalls that need to be avoided when building data domains that have complex structure:

- If a domain domain has nested levels of nodes and arrays, its typically a good candidate for being broken out into multiple domains
- Arrays inside of a domain can create scope issues where one zone may not have scope to an entire array. Again, if this is a possibility, the array should probably be broken out into another data domain where governance can be managed.

To summarize, following sound relational database principles will create a master data ecosystem that has data records that are easier to manage and to apply governance to.

If an HTTP Operation Is Not Required for an Adaptor, Don't Analyze It

Example: There is never a situation where the analysts for the College Application system wants YOUnite to create (POST) a new student - they need to maintain control of that process so there is no need to analyze the required elements for a POST /student for the College Application system.

Generally Speaking, All Changes to a Data Record Should Generate a Notification to All Adaptors Interested in That Data Domain

If that application tied to an adaptor has a well written RESTful interface it will allow you to register a callback for changes -- if not then you will need to discover a way to detect changes.

Additionally, all new and deleted resources should generate a notification (this is a YOUnite feature).

Example: A college course catalogue system would not get a notification that a student has been deleted from the system but several others would such as the college application system and the college SIS.

If Data Elements Are Used by Only One System, Then Don't Normalize Them Unless They Are Used Inside Another Data Domain

The job of the data analyst is to create as little work as possible. A single element added to a federated data domain has an exponential effect on the complexity of the overall system.

Example: A college system uses an Ed Planning system that tracks meetings between the student and college faculty and staff. Others systems may use the Ed Planning data but if no other systems in the systems use the scheduling system, then the schedulng data can be ignored in respect to modeling student, faculty or college data domains.

The Process is Iterative

Start small and gradually conect more applications and services in the organization to the MDM ecosystem.

A Couple of Additional Points

- The YOUnite adaptor might need to read and manipulate non MDM data attributes to complete transactions
- When building an MDM worksheet you also need a reference data worksheet too to keep. This is data that infrequently changes (e.g. States, Countries, etc) but is commonly cross-referenced by other domains (e.g. customers). A decision should be made where the reference data should reside and consideration should be made to storing some or all of the reference data in the YOUnite data store for performance reasons.