Framework To find Hairball Structure in Enterprise Data Integration Repositories

Student Name: Shilpi Jain

IIIT-D-MTech-CS-DE-12-050
June 18, 2014

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Vikram Goyal (Chair)
Ashish Sureka
R.K.Agarwal

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Data Engineering

Keywords: Data integration, Sources, Targets, Mappings, Workflows, GUI, Connections, NodeXL, Log Stitching, Centralized hub, Hub and Spoke architecture, and Data integration tools

Certificate

This is to certify that the thesis titled **"Framework To find Hairball Structure in Enterprise Data Integration Repositories"** submitted by **Shilpi Jain** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by her under my guidance and supervision in the Data Engineering group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Professor Vikram Goyal
Indraprastha Institute of Information Technology, New Delhi

Abstract

In the current changing trends, collaboration between different organizations or consolidation between applications of the same organization has become a common phenomenon. In order to achieve strategic business objectives it is necessary to have a unified view of data, which is given by Enterprise Data integration (DI). Based on the requirements of different organizations, large number of tools and technologies are available in the market. Some organizations are already using different integration techniques from past many years. However due to the emerging era of big data and cloud enterprises want to shift from their old integration methods to the new and advance techniques.

In this dissertation, a framework has been developed to retrieve the structure of the enterprises integration repositories and present them visually, so that the enterprises can take completely informed decisions, as they cannot change what they do not understand. This framework can be used to find the connection link information, connection location and the frequency of the repetition of the same sources and links. This tool can also be used to find the amount of data transferred from one geographical area to the other, which will help the organizations in measuring the bandwidth requirements across networks. We conducted extensive experimental study on the available datasets of different organisations and found that approximately 90% of sources and 80% of connections are repeated in an integration environment. We discovered the main reason behind this repetition is the end-to-end connectivity between the creators to the consumers. To the best of our knowledge, our proposed framework is a unique tool of this type.

We have also implemented a Log Stitching utility, which can stitch the logs of a specific duration from various remote and local locations into one file that will help the existing integration tools in faster debugging of their errors which will indirectly reduce the down time for the applications. To achieve this, we have designed a unique method of k-way merges by using Java NIOs, and priority queue.

Acknowledgments

I would like to express my deepest gratitude to my advisor Dr. Vikram Goyal for believing in my work and for all the support and guidance, without which, none of this could be possible. He is a great person and one of the best mentors, I always be thankful to him.

I would also like to thank my Manager in Informatica, Mr. Rahul Gudla for giving me an opportunity to work in B2B Data Exchange Team and being a constant source of motivation and guidance. I would also like to thank my colleague Keshav Veerapaneni for his immense support and guidance throughout the working of the project.

Last but not least I want to thank my friends and parents who have always been with me, no matter where I am.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this age of Data, good decision making has become very critical, as various enterprise/government organizations interact not only within themselves but also with the outside world for their business requirements creating terabytes and petabytes of data. As businesses are spread geographically over various countries/continents as well as in multiple domains, the complexity of application interaction has become very complex. Intelligent decision making is required to have an edge over your competitors. These decisions can only be made in a practical way when all the required data is available at one place for consideration, either physically or virtually, so that we can have a look at all the facts together, analyze them and discover previously unseen insight, aiding the decision making process. In short, a mature, well developed, data integration solution is required [5].

Building warehouses is the industry's standard way to integrate all the data together. The main task during the creation of data warehouse is the collection of data from various number and various types of data sources, transform this data in the desired format and transport to the desired destination. The source data is generally stored in a distributed environment over the large cluster of the servers which can be located geographically separate from each other. Also, the source data can be stored in a combination of different types of data storage systems like relational/NoSQL databases, flat files, live streaming data from various processes. This data needs to be synchronized centrally and checked for quality and integrity. The data can then be transformed into the required format by applying some rules and then transported to the target location. The data transfer should be done in a secure and robust way, in order to protect it from unauthorized access and loss of data. This whole process is also known as ETL, stands for Extract, Transform and Load.

The current data integration tools provide an environment that allows you to create ETL applications easily by using source and target objects which can be connected using a GUI interface. For e.g. Informatica PowerCenter is one of the industry leading tools for data integration. It comes with a designer where users can design mappings (an ETL program) according to their

requirements. In Figure 1.1, we have a representation for a basic mapping in which we use a relational database as the source, an exptrans function which is applied to the table attributes, and a target, also in the form of a relational table.

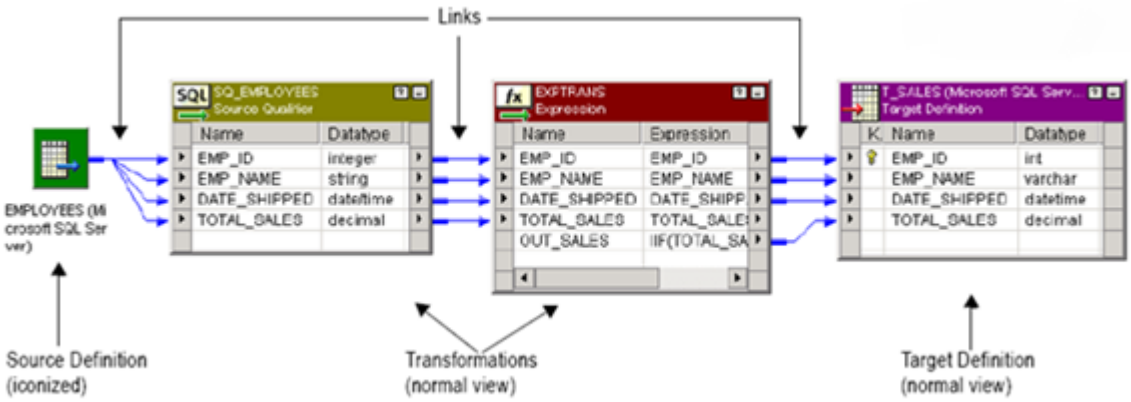


Figure 1.1: Example of Source-Target Mapping

Data Integration was not a big industrial problem, when the generated data was a fraction of quantity as it is now. The number and types of data storage options were also very limited compared to the enormous number of choices we have now. In the past, only few applications were sharing information with each other and it was easy to transfer data from one application to another. But now with the advent of cloud, mobile and social computing, organizations are faced with ever-increasing volumes of new data formats and new storage methods. Each year's total data generation is growing at double the rate of previous year. A fact was discovered that the amount of data generated from the dawn of humanity till 2004, now we generate the same amount of data in every 2 days.

In the current scenario, an average enterprise organization consists of several systems running hundreds of processes that IT need to support. Yet the current data infrastructure is fragmented, leading to the creation of data silos that restrict a company's information potential. Companies need a centralized, efficient way to manage and monitor information. A large part of IT budget in these organizations is being invested in data integration. However, most companies are still focusing on point-to-point integration strategies causing unnecessary expense, delay, risk and waste. However, over the period of time this approach gets complex as the number of inter-connection increases. There interdependencies create the potential for big, unpredictable failure impacts even with the slightest changes. The custom point-to-point data-level integrations has become a tangled web of brittle connections, quickly beginning to look like a hairball structure. Individual systems talk to each other in silos, in different data formats, and at the wrong times, communication isnt consistent and standardized. And now, their integration process is suffering from below mentioned problems:

1. Fragile and complex environment.

2. Complicated change management.

3. Extremely costly to run.
4. Many redundancies.
5. No visibility into relationships between data.
6. Data duplicated, inconsistent and non-trusted.

Let us take an example. Enterprise A consists of thousands of point-to-point connections in which many systems are interacting with each other in every possible way. It is starting to look like a hairball structure which is very clumsy to understand and to make any change, as shown in Figure 1.3 .

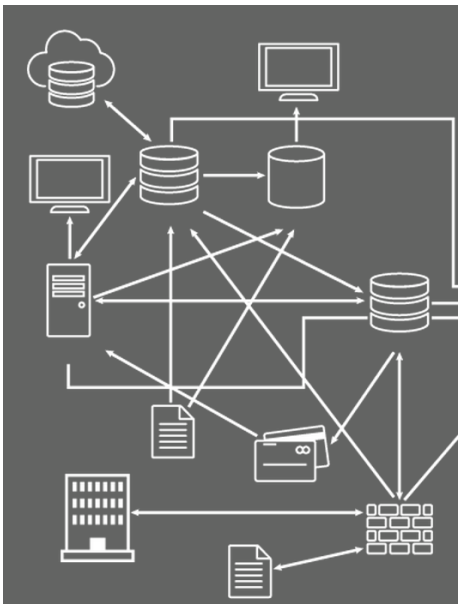


Figure 1.2: Current Structures of Mappings

The objective of this project is to provide motivation for a solution for optimizing the framework of data transfer among places, and find an efficient way to manage the connections between systems that are interacting with each other. We propose an approach to simplify this process and increase its efficiency to handle more and more data with ease. We propose a middle abstraction layer, namely, Hub which reduces dependencies by decoupling systems and providing flexibility. The number of connections is significantly smaller for Hub, which makes it more efficient and also much easier to maintain.

Figure 1.2 represents the resulting architecture after introduction of the middle layer. The following are the reasons for the proposed change in architecture:

1. To provide more agility with the application dependencies so it would be easier to move and upgrade the application to the cloud.
2. To increase large scale enterprise productivity by reducing overlapping significantly.

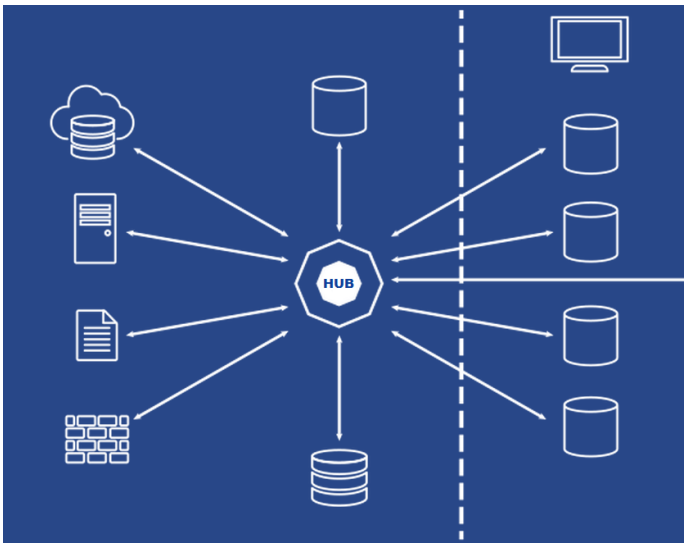


Figure 1.3: Data Integration Overview after introduction of Hub

3. To save time by faster delivery of information.
4. To provide more data control, as you can apply any technology or tool at centralized data server.

1.2 Existing Scenario

Almost all the current data integration systems are point-to-point but different tools have their architecture. Here is a list of existing ETL tools:

1.2.1 InformaticaPowerCenter

Informatica Power Center's architecture is subdivided into two main components:

1. Server
   (a) Informatica Server
   (b) Repository Server
2. Client
   (a) Workflow Designer
   (b) Workflow Manager
   (c) Workflow Monitor
   (d) Repository Manager

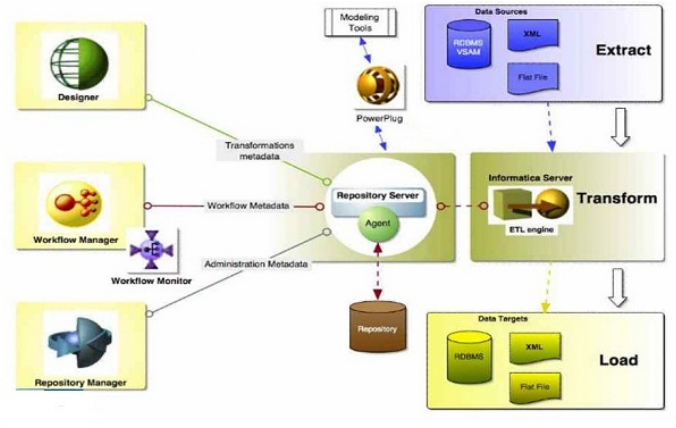


Figure 1.4: Informatica PowerCenter Architecture

How it works:

- Designer component is used to create a program called a "Mapping" which consists of all the information necessary to run our ETL process.
- The created mapping is stored in the repository server for execution.
- A "Session Task" is created which is required for running the mapping. The created "Session Task" is stored in the repository server.
- For running "Session Task", a "Workflow" needs to be created using the Workflow Manager Tool which is then also saved into the repository.
- Workflow monitor can be used to monitor the execution of workflow.

Other Important components are:

Node The machines are represented in a logical way as "Node". There can be numerous "Nodes" for a single "Domain". The Domain is hosted by a "Master Gateway Node". Nodes can be customized to run application services like integration/repository service. Nodes create a kind of Star-Network in which all the messages from one node to another travel through "Master Gateway Node".

Domain It is the fundamental unit for management and administration of powercenter services.

Application Services Group of services run on the Informatica server. For example, Integration Service, Metadata Management Service, Powercenter Repository Service etc.

Service Manager It starts and runs the application services on a node.

Powercenter Repository It is a relational database for storing the metadata related to our ETL applications. The tables in the database contain the instructions to extract, transform and load data.

Powercenter Integration Service The fundamental services of the Informatica tool are extraction, transformation and loading the data.

Powercenter Repository Service This service enables the client to create/modify the metadata related to an ETL task and also allows it to run the workflow related to the task.

Metadata Manager Service It is used to analyze the metadata from multiple repositories.

Informatica Administrator Service It is a web-based service for the administration of informatica domain and security.

1.2.2 SAS DI studio

In SAS DI studio, the data integration layer is divided into three sections:-

1. Staging - It is an area for data to be placed before it is transformed and stored in the warehouse
2. Foundation Layer - a data model which supports the information collected through different business processes.
3. Access and Performance - an optimized data model for information retrieval and analysis.

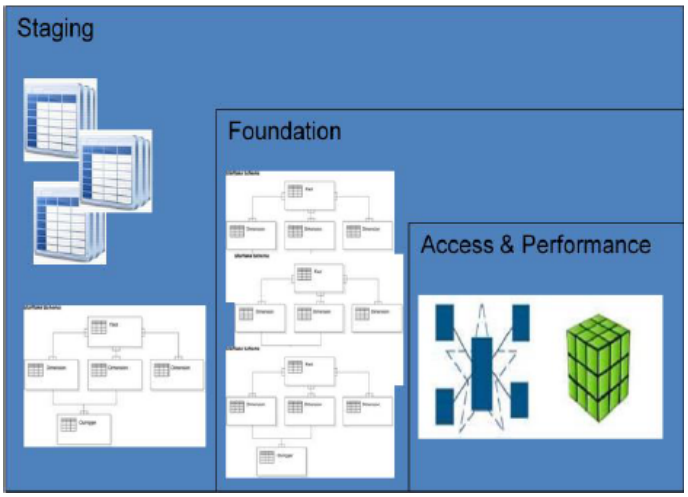


Figure 1.5: Data Integration Layer of SAS

How it works:

- The data is moved from various sources (ERP, CRM) to the staging area.
- The staging area is moved onto the SAS servers and SAS is used for data storage in place of a relational databases. This diminishes the movement of information which is resource consuming.

- This way, the SAS server can do the entire internal server processing without requiring read/write data across different networks, increasing performance.
- Goal of this architecture is to reduce the number of data read/writes, reducing the time to transform and store information [4].

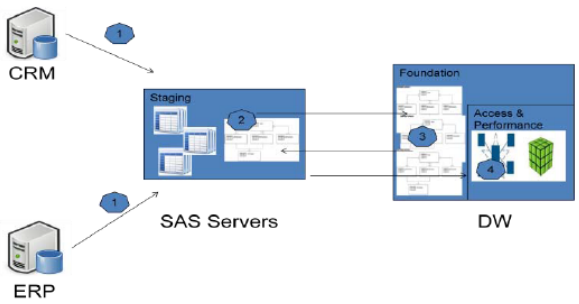


Figure 1.6: Processing Architecture

1.2.3 SAP (Business Objects Data Integrator)

Features:

- SAP data integrator consists of a real-time as well as batch integration system to support analytical and supply-chain management applications.
- It allows organizations to create a unified infrastructure for all kind of data movements including batch/real-time movements.
- Organizations can manage/reuse data as a single entity independent of the type or number of systems present.
- Create a single metadata model to determine the relationships between different extraction/access methods and provide integrated lineage and impact analysis.
- Sharing data/metadata between organizations and SAP Enterprise Platform is possible.
- Optional pre-packaged data solutions are available for faster deployment. These solutions are used to cache historical/daily data from operational systems in relational databases.

Architecture consists of six components:

Designer It is an easy to use GUI tool for creating mappings, transformations, and control logic. It define workflows(job execution flow) and dataflows(data transformation).

Job Server This component starts the data movement engine which integrates data from multiple sources, manages extractions and transactions of data from sources and performs complex data transformations.

Data Engine The job server calls data engine to extract, transform and move the data. Data engine uses parallel processing and in-memory data transformations for scalability and maximum throughput.

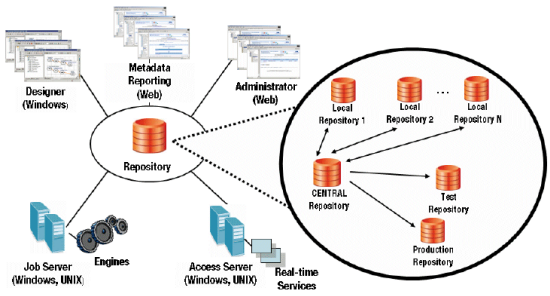


Figure 1.7: SAP Data Services Architecture

Access Server It is a real-time message mediator that collects message requests and propagates a reply in a user specified time frame.

Administrator It is a web-based administration tool which does scheduling/monitoring/executing jobs, real-time service configuration, job/access server configuration, adapter configuration, user management etc.

Metadata Reporting Applications It provides web-based metadata analysis and reporting of data integrator jobs and other associated SAP applications.

It is a distributed architecture. Multiple designers can work on a central repository. The components of the integrator can be distributed across multiple computers for handling scalability and traffic.

1.2.4 Problem Overview

Now there is a big discussion, whether you need a hub or not. As it is already explained that by using centralized location hub enterprises can reduce their system complexity from n(n-1) in case of point-to-point to n(using hub), where n is the no. of connections. Many enterprises are using their tools from years and a huge cost is required to implement such a system or to buy this implementation from any company, so there is a strict requirement of a framework which can give them a clear picture of their system, that can help them to take more significant decisions and provide them an estimate of money and resource savings by moving from point-to-point architecture to centralized hub architecture. To solve this problem and to provide a clear insight of their repositories, we have implemented a framework which can do the following:

1. Scan the repository, to find all the mappings.
2. Find relationship between sources and targets.
3. Show the number of connections involved.
4. Get the amount of data transferred between locations.
5. Get the number of redundant sources, targets and mappings.

The current scenario of data integration requires to process large volumes of data across the distributed environment. The processing of data is carried out over a large cluster of nodes in a network. We need to debug very fast in order to reduce the processing losses. So, the debugging of the errors becomes difficult. We have implemented a log stitching utility to make debugging of logs faster and feasible. This module reads the logs for the given time period from the different sources present in the different nodes over the cluster in a network and stitches them chronologically and produces the output in a single log file.

Chapter 2

Solution Approach

2.1 Solution Framework for Repository Structure

There are two types of approaches through which we can get the structure and the data of the user repository:

- Fetch the data from the metadata tables of the repository and get the upper level information by applying several joins on all the tables. Even after repeated joins on tables, we are not able to fetch table level information. As the PowerCenter and other integration tools store the content in distributed form. All the objects are in the different tables, like no. of folders information is in one table from that we will get an id which we can use to relate all other information with that folder. As in repositories everything is inside a folder. A folder contains workflows, sources, targets and mappings which are stored in their separate tables. This way of retrieval is way to inefficient and less accurate.
- In the second approach we need to develop an API to interact with the tool. After that we define a common structure definition for the all the objects that are needed to be fetched. Then export all the workflows from the repository in the form of XML files and parse each of the XML file to get information about all the sources, targets, mappings and connections. And then after traversing through each connector port, we get the accurate link information.

The first method is really fast and gives the result in very less time with less processing, but the accuracy of this method in the result is less, through which we can lose the customer credibility. In the second method, most of the time is consumed to export the workflows from a given repository, but the accuracy of the result is high and we can show the exact picture of all the links and data transferred from one place to another. So, we have implemented the second method. Figure 2.1 shows the high level architecture of our proposed solution approach. The framework development is divided into three phases:

1. Data Restoration - restore the contents of backup files.

2. Data Export - export objects in form of XML files.

3. Scanning and Visualization of links - parse the files and create excel file to represent the structure.

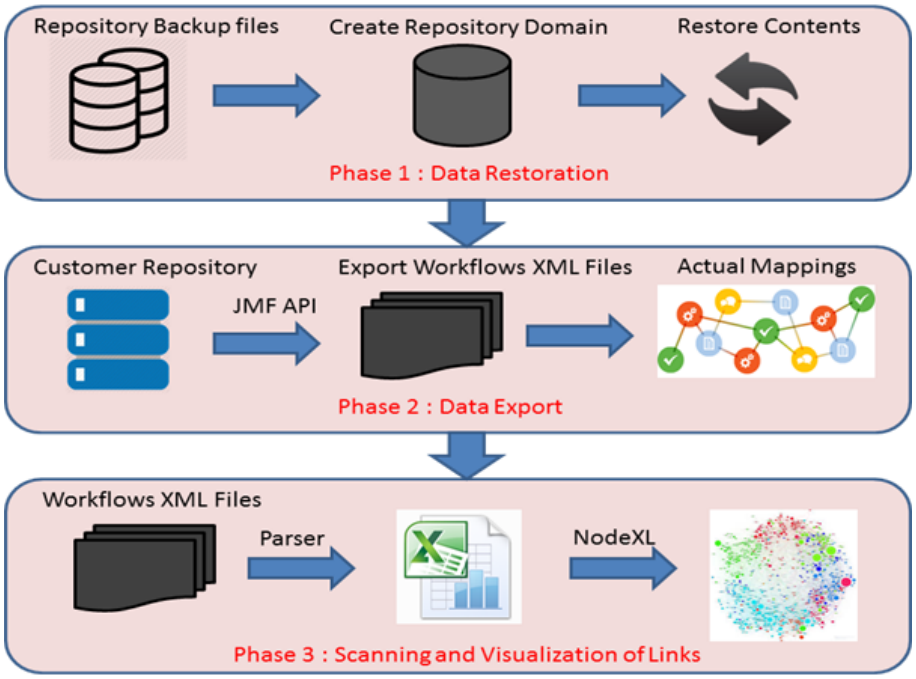


Figure 2.1: Repository Scanning Framework

2.1.1 Data Restoration

To create our framework, and to understand the structure of integration mappings, we have used the backup files of the Informatica PowerCenter Customers, which is considered as the present best tool in terms of data integration. We have backup files of 10 customer repositories, so first we need to restore the contents of those repositories to fetch the workflow files.To restore the contents we need to create a database in which we can store the metadata of the repository. After that, restore the contents in the created database, repositories are restored with all the workflows, mappings and sessions in them.

2.1.2 Data Export

Once we have repository contents, we want to access source, target, mapping and connection objects from the folders. As the source code of the tools are not accessible, so we created an interface which is a Java api through which we can interact with the workflow objects and export the data in the form of XML files of a predefined structure.

Preprocessing

Some of the exported XML files have invalid characters in them, so to overcome this problem we created a function which read all the files character by character, find invalid character and remove them from the files before parsing them.

XML Validation

We require two DTDs for XML validation, we copied both the DTDs in the same folder in which all the XML files are present.

2.1.3 Scanning and Data Visualization

This is the most important step, now we have thousands of workflows XML files, we developed a parser to find all the links between sources and targets of the mappings.

Structure of XML Files

XML file contain several sections:

- Repository
- Folder
- Source
- Target
- Workflow
- Mapping
- Session
  - Session Extension
  - Session Transformation
  - Connector
  - Connection Reference

Parsing Using XPath

Structure of XML files is complex, and there is no direct way to find that which sources are related to which targets, due to all the transformation functions in between. To overcome this difficulty we read all the sources and targets in a data structure and then we store all the

connectors information as vertices of a graph, and connected vertices as neighbors, and finally we applied DFS to find if there is a path between a particular source to target exists or not. We also want the connection information of the sources and the targets, through which we can find the physical location of the resource. As we know, we can have different types of sources, so the connection information varies for each source. Flat files do not have any connection information, but they have folder storage information in their attributes. So in case of files, we read all the attributes to get the storage location of flat files.

To get the amount of data transferred from one location to other, we fetch the data from a view, which is in the repository database. After getting all the data from the files and the database, we created a excel files using XSSF API, which contains following information:

1. Source
   (a) Connection Name
   (b) Connection Type
   (c) Connection SubType
2. Target
   (a) Connection Name
   (b) Connection Type
   (c) Connection SubType
3. Folder Name
4. Workflow Name
5. Mapping Name
6. Session Name
7. No. of Success Source Rows
8. No. of Fail Source Rows
9. Width

The last column width is the frequency of repeated source-target connection.

Visualization Using NodeXL

It is an extension of Excel. To create graph, we first need to populate NodeXl Excel Template with data. It has a number of algorithms to draw a graph. Each algorithm has its own advantages and disadvantages. There is also a feature to create groups based on the vertex attributes and show the layout of groups in separate boxes. Force-directed algorithms are designed to make

all the edges about the same length and to minimize line crossings, which can make for a more aesthetically pleasing and readable graph.

There are two main force directed graph drawing algorithms

- Fruchterman reingold
- Harel koren fast multiscale

Harel-Koren Fast Multiscale Algorithm

This algorithm is the improvement of spring electrical model (force directed). It is very fast and efficient for large graphs. It clusters all nodes of the same group together and gives almost same length edges. It has low computation complexity as compared to other force directed algorithms because it restricts the repulsive force calculation to neighbourhood only, thus ignoring long-range force [3].

2.2 Solution Framework of Centralized Log Stitching

A large number of integration applications are interacting and transferring data to each other from geographically separate location. This scenario requires us to process large volumes of data across the distributed environment. The processing of data is carried out by a number of tools like Data Exchange, Power-Center etc. over a large cluster of nodes in a network. Most of these applications produce their specific log files in their specified folders over the node. The debugging for errors need to be done very efficiently in order to reduce the processing loss. The debugging of the errors becomes infeasible if we are required to access every log folder present in each node on the cluster.

To solve this problem we have implemented a log stitching utility to make debugging of logs feasible and efficient. This module reads the logs for the given time period from the different sources present in the different nodes over the cluster in a network and stitches them chronologically and produces the output in a single log file.

The logs are sorted chronologically from oldest to newest log within the given time period using the priority queue data structure. A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. The priority queue is implemented with a Min-Heap data structure. The min-Heap implementation is highly efficient for this scenario. It takes O (1) constant time for retrieving the log while it takes minimal time of O (log N) time for insertion.

The logs are read from the remote machines over the network using SFTP protocol. The framework used for SFTP protocol is Java Secure Channel (JSCH). The files are copied into the temp folder of the host node and then used for further processing.

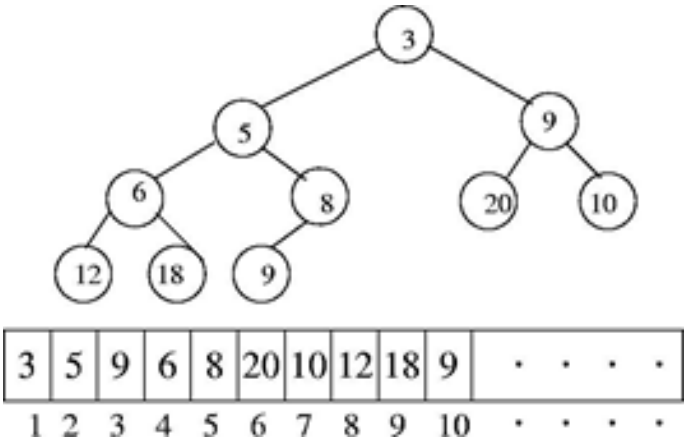


Figure 2.2: Min-Heap Data Structure

Since the logs are continually written by the applications the files are needed to be read in a non-blocking way. For this purpose we are using the framework Java NIO instead of Java IO. Also, NIO framework was found to be twice as fast as the IO framework so making the implementation faster and efficient. Java NIO was used for reading log files instead of Java IO. Java NIO read files in a non-blocking way which is an essential prerequisite for this utility. Also, NIO framework is buffer oriented framework as compared to IO which is stream oriented. Java IO being stream oriented means that you read one or more bytes at a time, from a stream but they are not cached anywhere. In Java NIO's buffer oriented approach, data is read into a buffer from which it is later processed. We can move forth and back in the buffer as you need to. This gives user a bit more flexibility during processing. However, we also need to check if the buffer contains all the data you need in order to fully process it. As there is a storage capacity for each log file after that it will roll over and next logs are written in new files and the old file is renamed with a particular regex. We also handle this condition, when user has given the path of a specific file, then we are reading all the roll-over files of the given file.

The output file contains all the logs produced over the given period of time by all the applications running for the project. For each of the logs we write the source path along with the log entry. Each of the log file contain the time-stamp in a different format. The time for each log entry with the particular format is read using Simple Date Time format (SDF) framework present in Java library. The date pattern for each of the log type is stored in a configuration properties file along with the source path

Chapter 3

Experimental DataSet and Performance Evaluation

3.1 Experimental DataSet

To perform experiments, we collect the dataset from backup files of eight InformaticaPowerCentercustomers. We found the versions of the PowerCenter on which these repositories were created and then we create new databases for each repository, in which we restored their contents. Table 3.1 below is showing the details of the repositories:

Backup Files Name	Size	No. of Folders	No. of Workflows
Backup A	2.40 gb	520	14088
Backup B	7.88 mb	1	246
Backup C	1.69 gb	42	1166
Backup D	527 mb	32	777
Backup E	1.23 gb	14	1093
Backup F	1.53 gb	100	3272
Backup G	5.94 mb	3	30
Backup H	469 mb	17	189

Table 3.1: Experimental DataSet Statistics

First four repositories are of InformaticaPowerCenter 9.5.1 HF2, and the other four are from InformaticaPowerCenter 9.1.0.

In the begining we created a blank database to restore the contents. We also need the server running on our machine, which will restore the contents in the database which we have created for the repository. Then we ran our API program to export all the workflows from each repository one by one in a folder. This process is slow as the data is stored in the database in a very diverse form and to combine all the related data and write it to a single file is a time consuming process. And some repositories are really large. It took approximately 24 hours to export 14088 workflows. After exporting all this data we ran our framework on each folder and retrieve all

the workflows and connections details in workflows. Table 3.2 gives the further details of the data statistics:

Backup Files Name	No. of Edges(Total no. of Source Target Links)	No. of Vertices(Total no. of Ports)	No. of Groups based on connection name
Backup A	38125	2877	180
Backup B	327	271	5
Backup C	15012	1378	12
Backup D	10779	1190	72
Backup E	25999	1333	11
Backup F	23007	1575	18
Backup G	19	16	4
Backup H	1327	915	12

Table 3.2: Excel Sheet Statistics of each repository

3.2 Results Visualization

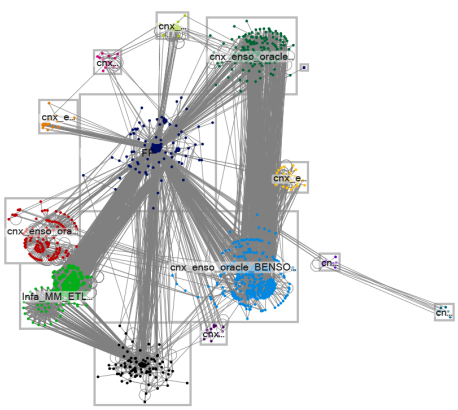
An important part of this project is to visualize results effectively, so that we can explain the insights pattern to the customers. After populating all the data in the form of excel file. We have used NodeXL for representation.

3.2.1 Graphs

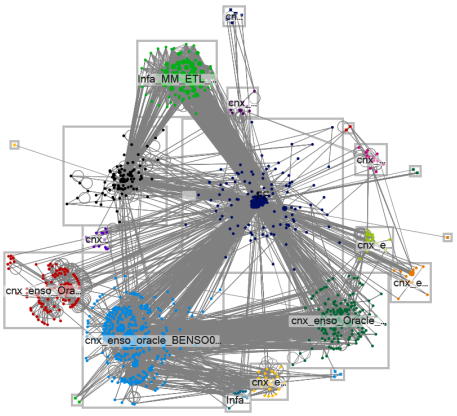
Figure 3.1 is showing the graph representation of different backup repositories. The boxes in the graph represent different groups created on the basis of connection name, edges between the vertices represents mapping, and the vertices represent source-target tables used in the workflows. The graphs below are simply show the hairball structure of the repositories. Different node size represents the frequency of source repetition and variations in edge width show the link frequency difference. These graphs show that huge data is transferred between connections. These connections may be present in geographically separate locations. In the next section, we will give the metrics, which shows the importance of introduction of hub architecture.

Description about Graphs:

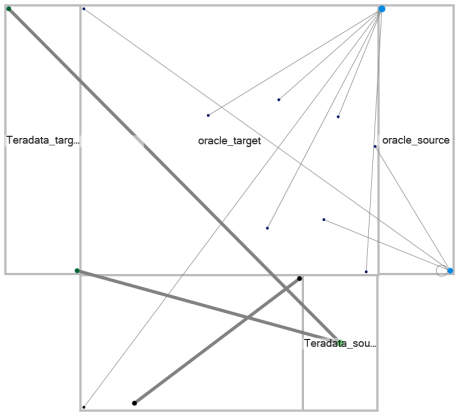
Figure 3.1 showing the results of running tool on the dataset. We have used Force Directed Algorithm to create graph layout of group boxes. Different colors represent different groups. The different size of vertices shows their frequency of repetition. Although the scale is very low, so it is not very clear. Edges are also varies in their width to show the repetition of links. All the groups are labeled with their group names. The groups with less number of vertices are represented smaller size box while the groups with more vertices are larger in size. Some edges



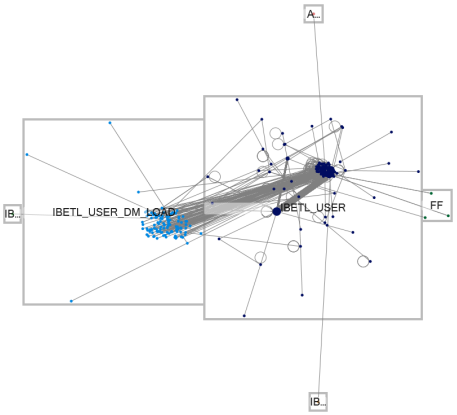
(a) Backup C



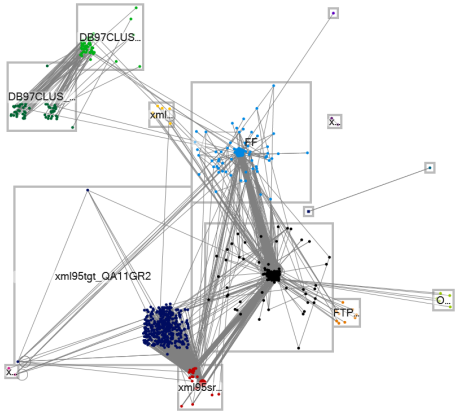
(b) Backup F



(c) Backup B



(d) Backup G



(e) Backup H

Figure 3.1: Graph Representation of Repositories

have both of their vertices in the same box. It shows that both source and target belong to same connection.

3.3 Evaluation Metrics

After running the tool on the dataset we get the following statistics of the data repetition in each of the repository individually.

$$\% \text{ of feature repetition} = \frac{(\text{Total repetition} - \text{unique values}) * 100}{\text{unique values}}$$

Backup repository	% of Link Repetition	% of Source Repetition	% of Target Repetition
Backup A	95	94	75
Backup B	53	43	34
Backup C	69	93	93
Backup D	23	92	93
Backup E	96	96	82
Backup F	94	95	77
Backup G	78	31	31
Backup H	80	46	33

Table 3.3: Statistics of Repeated Data in each Repository

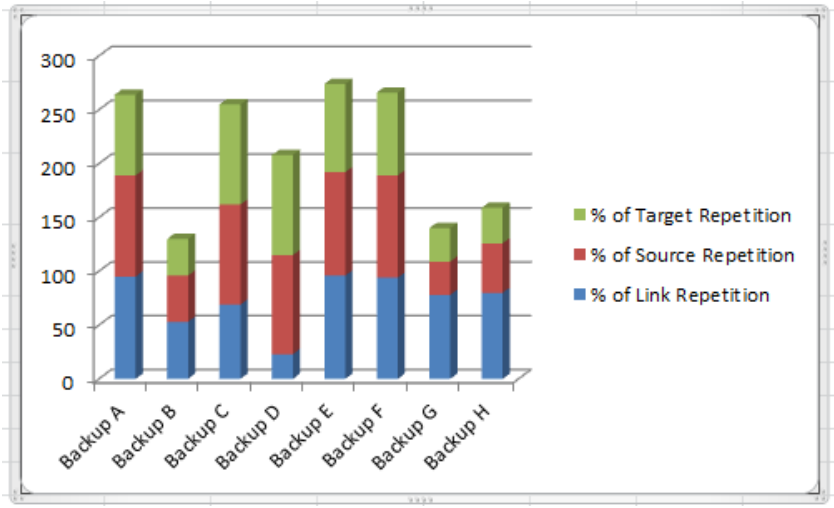


Figure 3.2: Stacked Chart Represent percentage of frequency of repetition of each feature

As above Table 3.3 shows each repository has a different amount of data repetition statistics but most of them has approximately 90% of source - target repetitions. When we manually observe the repositories we found out, that some repositories have the two extra copies of same data integration folders for QA and backup. Sometimes, repetition is required but the above statistics data is clearly showing that we are repeatedly using the same sources and targets and

also same set of links. Most of the repositories are repeatedly using same links to transfer data so it would be better for them to use hub architecture so that we can reduce this link repetition considerably. It will save the cost of data transfer and also save bandwidth.

After using Hub the network bandwidth requirement are reduced to single link between each other to hub and hub to target.

Chapter 4

Conclusions

Mining customer repositories, finding patterns of links, repetition frequencies and the amount of data transferred through them will give more meaningful insights to the concerned enterprises. Now they can take more informed and accurate decisions for their data integration tools requirements, which indirectly help them to reduce their input cost and provide more enterprise-flexibility towards future modifications. Enterprises can also analyze which connections are active and require more bandwidth than others. We conducted a series of experiments on the Informatica PowerCenter Customer Repositories which revealed that most of the repositories have huge numbers of repeated links, sources and targets and the one most important reason behind this is the point-to-point connections. Sometimes, the data resources are highly replicated just to prevent data losses in case of failure. Better knowledge of repository structure will help enterprises to maintain it in a more scalable way and also give them a way to untangle their hairball structure repositories so that they can easily make changes in the future.

The centralized log stitching utility will help the existing systems to debug their errors speedily and efficiently which indirectly saves the time and effort of the IT persons, which they can use in more productive work.

Chapter 5

Limitations and Future Work

Our present system fulfills most of the requirements of the information required by the enterprises, still there are some limitations in the current technique. Currently, we are exporting the workflows first and then parsing them to fetch the information, which is quite time consuming and sometimes it is not possible to export complete workflow due to some internal implementation of the tools. Such limitation can confine the usage of this tool in a limited domain. But in future, if we add a persistent layer in between objects and the framework, so that we can get the data directly in form of objects then we can make this framework functionality more generalized and expand the usage domain. Also, presently we are not getting the complete information of the shortcuts that are created to use same objects at different locations, and are present in some files as sources and targets. After being able to get complete information of all the objects, we can make more strategic decisions to improve the data integration quality of our organization.

Bibliography

[1] Informatica Data Integration Hub. `http://www.informatica.com/Images/02473_data-integration-hub_ds_en-US.pdf`. Accessed: 2014-05-29.

[2] Informatica PowerCenter. `http://www.techtiks.com/informatica/beginners-guide/informatica-system-architecture/`. Accessed: 2014-05-29.

[3] Harel, D., and Koren, Y. A fast multi-scale method for drawing large graphs. In *Proceedings of the 8th International Symposium on Graph Drawing* (London, UK, UK, 2001), GD '00, Springer-Verlag, pp. 183–196.

[4] Nancy, R., and Tim, S. What's new in sas data integration. In *SESUG 2011, The Proceedings of the SouthEast SAS Users Group* (Durham, NC, 2011).

[5] Russom, P. Next generation data integration. In *TDWI research* (Apr 2011).

[6] Smith, M. Nodexl: Simple network analysis for social media. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on* (May 2013), pp. 89–93.