



THE UNIVERSITY OF  
MELBOURNE

---

# ETL Process for Unstructured Geoscience Data

---

## ***Group 3 Members***

Shiv Jitendra Mistry

Jiahui Zhu

Hangzhou Wu

Kanav Sood

## ***Supervisor***

Vivek Katial

Department of Computing and Information Systems,  
The University of Melbourne

## ***Client***

Dr. Fabian Kohlmann

Managing Director, Lithodat Pty Ltd

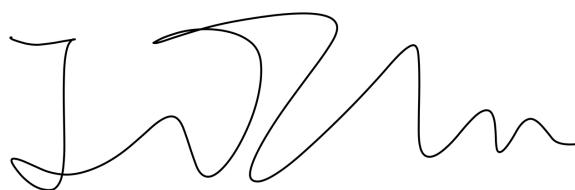
# Declaration of Authorship

We certify that this report does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university, and that to the best of our knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

This report is 9869 words in length (excluding text in images, tables, bibliographies, and appendices).



*Shiv Jitendra Mistry*



*Jiahui Zhu*



*Hangzhou Wu*



*Kanav Sood*

# Abstract

Geospatial data contains vital information regarding features present at various locations of the earth, but most geospatial datasets have discordant formats which make the analysis of a collection of geospatial data a cumbersome process, as it has to be converted into a unified format first. Traditionally used Extraction, Transformation and Loading (ETL) technology which is mostly used for the purpose of unifying various data sources using certain rules can be modified and applied to merge the discordant formats of geospatial data making them coherent. The goal of this project is to assimilate and contrast various ETL tools for the purpose of converting unstructured geospatial data into a unified format. Various ETL tools were tested and a tool was then selected based on the metrics of the tests, which was then used to generate an ETL pipeline to achieve the desired goal of structured geospatial data. Tests were conducted on various popular and industry-leading ETL tools using various testing techniques such as Production Validation Testing, Data Check and Constraint Testing, and Regression Testing to evaluate the performance of the tools and choose the best according to the requirements presented by the client.

# Acknowledgements

We would like to express our sincere gratitude to our client Dr. Fabian Kohlman, for his constant feedback and support throughout the project. We would also like to extend our thanks to Moritz Theile for providing us with the project plan, the required data, and providing timely suggestions.

We want to thank our subject co-ordinator Michael Kirley and project supervisor Vivek Katial for their guidance throughout the project.

Finally, we thank our family and friends for their unfathomable encouragement throughout our course of study.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                              | <b>8</b>  |
| <b>2</b> | <b>Related Work</b>                              | <b>9</b>  |
| 2.1      | What is ETL? . . . . .                           | 9         |
| 2.1.1    | Extraction . . . . .                             | 9         |
| 2.1.2    | Transformation . . . . .                         | 9         |
| 2.1.3    | Loading . . . . .                                | 10        |
| 2.2      | ETL for Geo-Data . . . . .                       | 10        |
| 2.2.1    | Why is it important? . . . . .                   | 10        |
| 2.2.2    | How is it different? . . . . .                   | 10        |
| 2.3      | Approaches to ETL . . . . .                      | 11        |
| 2.3.1    | Code Based-Solution . . . . .                    | 11        |
| 2.3.2    | Software Based-Solution . . . . .                | 11        |
| 2.3.3    | Challenges to performing ETL . . . . .           | 12        |
| 2.4      | Testing techniques for ETL . . . . .             | 12        |
| 2.5      | Comparison of ETL Tools . . . . .                | 13        |
| 2.5.1    | Coding based tools . . . . .                     | 13        |
| 2.5.2    | Software based tools . . . . .                   | 15        |
| 2.5.3    | Cloud based tools . . . . .                      | 16        |
| 2.6      | Database Normalization . . . . .                 | 17        |
| 2.6.1    | Types of Normalization . . . . .                 | 17        |
| <b>3</b> | <b>Methodology</b>                               | <b>20</b> |
| 3.1      | Client's ETL Problem . . . . .                   | 20        |
| 3.2      | Approach to solving Client's Problem . . . . .   | 20        |
| 3.3      | Procedure of the tests to be conducted . . . . . | 21        |
| <b>4</b> | <b>Data (Exploration &amp; Analysis)</b>         | <b>22</b> |
| 4.1      | Data Description . . . . .                       | 22        |
| 4.1.1    | Sample Dataset . . . . .                         | 22        |
| 4.1.2    | Client Dataset . . . . .                         | 23        |
| 4.2      | Data Dictionary . . . . .                        | 25        |
| 4.3      | Data Exploration . . . . .                       | 27        |
| 4.4      | Data Cleaning . . . . .                          | 29        |
| <b>5</b> | <b>Testing ETL Tools</b>                         | <b>30</b> |

|          |  |           |
|----------|--|-----------|
| 5.1      | Pentaho Software . . . . .                                   | 30        |
| 5.1.1    | Using Sample Dataset . . . . .                               | 30        |
| 5.1.2    | Using Client Dataset . . . . .                               | 31        |
| 5.2      | Tableau Prep . . . . .                                       | 33        |
| 5.2.1    | Using Client Dataset . . . . .                               | 33        |
| 5.3      | MS Excel . . . . .   | 35        |
| 5.4      | Google Sheets . . . . .                                      | 36        |
| <b>6</b> | <b>Final ETL Pipeline</b>                                    | <b>37</b> |
| 6.1      | Extraction . . . . .   | 37        |
| 6.2      | Transformation . . . . .                                     | 37        |
| 6.2.1    | Data Normalization and Cleaning using Tableau Prep . . . . . | 37        |
| 6.3      | Loading . . . . .  | 40        |
| 6.3.1    | Lithosurfer Introduction . . . . .                           | 40        |
| 6.3.2    | Loading into Lithosurfer . . . . .                           | 41        |
| <b>7</b> | <b>Conclusion and future work</b>                            | <b>43</b> |
| <b>8</b> | <b>Appendix</b>  | <b>44</b> |
| 8.1      | Team members and Roles . . . . .                             | 44        |

## List of Figures

|    |  |    |
|----|--|----|
| 1  | General ETL Framework . . . . .                  | 9  |
| 2  | Conversion to 1NF . . . . .                      | 17 |
| 3  | Conversion to 2NF . . . . .                      | 18 |
| 4  | Table not following 3NF . . . . .                | 18 |
| 5  | Conversion to 3NF . . . . .                      | 19 |
| 6  | Conversion to BCNF . . . . .                     | 19 |
| 7  | Brief Overview of Client's ETL problem . . . . . | 20 |
| 8  | Sample Data ER Diagram . . . . .                 | 22 |
| 9  | Client Data ER Diagram . . . . .                 | 24 |
| 10 | Sample Dataset Flow - Pentaho . . . . .          | 30 |
| 11 | Client Dataset Flow - Pentaho . . . . .          | 31 |
| 12 | Split into sub-tables . . . . .                  | 33 |
| 13 | sub-tables Normalization . . . . .               | 34 |
| 14 | Data Extraction . . . . .                        | 37 |
| 15 | Data Transformation Overview . . . . .           | 38 |
| 16 | Final Workflow . . . . .                         | 38 |
| 17 | Lithosurfer . . . . .                            | 41 |
| 18 | Data loading in lithosurfer . . . . .            | 42 |

## List of Tables

|   |  |    |
|---|--|----|
| 1 | Data Dictionary . . . . .                                  | 27 |
| 2 | Columns and respective no. of null/empty entries . . . . . | 29 |
| 3 | Test results - Pentaho . . . . .                           | 32 |
| 4 | Test results - Tableau-Prep . . . . .                      | 35 |
| 5 | Team members and roles . . . . .                           | 44 |

# 1 Introduction

Lithodat is one of the hubs of spatial geoscience data and also experts in providing geoscience data solutions for exploration and research through their online platform *Lithosurfer*. With geospatial datasets (structured and unstructured) arriving from various sources, the structured datasets have a fixed format (i.e. common schema) and thus can be easily transferred to a common database, while each unstructured dataset can be of varying input formats (e.g. data from a sensor) which makes it difficult to translate and clean the unstructured datasets through one script into a digestible format and bring the data into a structured and queryable form. Thus, the goal of this project is to find a semi-automated data entry process that could help bring the unstructured geological data into a structured and queryable form by finding, testing, and evaluating various software solutions which can perform the Extract, Transform, Load (ETL) process for Lithodat. The aim of this project is also to provide documentation on how to use specific ETL solutions on sample geological datasets collected by Lithodat.

ETL which stands for Extract, Transform and Load is a method of extracting data from a source arrangement, transforming the data into a unified format fixed by the developing organization, and finally loading the transformed data into an appropriate database from where further research analysis can be performed. Implementing an ETL solution varies depending on the requirements of the type of dataset and the client's requirements. There are three generalized approaches to ETL - one involving a code-based solution, other, involving a software tool-based ETL process and third being a mixture of both. As the solution provided by us is to be used by geoscientists at Lithodat with minimal to no coding experience, we would be focusing mostly on the software-tool-based solutions for the duration of the project as the geoscientist's would not be able to modify if it were a code based solution.

ETL is essential to any organization as a proper functioning ETL would help to gather, read and migrate huge chunks of unstructured data from various sources and platforms into one meaningful platform from where various database operations can be performed, and data scientists can perform appropriate data analysis. In short, ETL is the initial step that helps make more sense of the data. In this report, we would firstly be performing a literature review on the concepts related to ETL, its various testing techniques, a brief comparison of the popular ETL tools, and concepts related to database normalization performed during the data transformation stage. After covering the groundwork, we would be taking a closer look at the client's problems and propose a methodology to solve it. As part of the methodology, we would test out the various ETL tools based on the clients requirements and select the best-suited tool to perform ETL tasks. Finally, Using the best-selected ETL tool we would construct a final ETL pipeline solving the client's problem.



## 2 Related Work

### 2.1 What is ETL?

Organizations rely on data for various purposes to fulfill their business objectives and the data they require may be stored in varying formats, data storage and locations [1]. In order to make sense and use of this, the spread out data requires combining, cleaning, standardizing, and collation from various sources and storing them at a single location. This process of extracting data from the database (or a collection of data sources) and resolving it to a data warehouse after appropriate alteration is known as ETL, which stands for Extraction, Transformation, and Loading. A general overview of the ETL process can be seen in Fig 1. Extracted from various sources, the data is cleansed and transformed into a unified format before finally being loaded into a Data warehouse. The three steps of ETL are detailed in the following sub-sections.

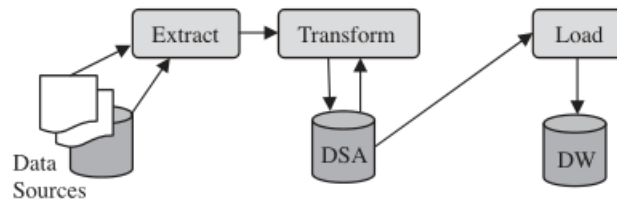


Figure 1: General ETL Framework

#### 2.1.1 Extraction

This is the first step of the ETL process, which is responsible for extracting data from various sources, with the possibility that the data stored at various sources having distinct characteristics which needs to be extracted effectively [2]. Initially, when the ETL pipeline is built the extraction involves collecting data from sources and loading it into the data warehouse [1]. But, once the ETL pipeline is built, an incremental extraction [2] is done every time new/modified data enters the pipeline in order to refresh the data warehouse.

#### 2.1.2 Transformation

The second step of the ETL process involves converting the extracted data into the required format (agreeing on the business logic) which can then be used to seamlessly store it into the data warehouse. The transformation process includes operations like database normalization, joins, performing calculations, adding primary key and foreign key on the data-set along with data cleansing, correction, normalization, and integrity checks [3].

### **2.1.3 Loading**

The final step of the ETL process takes the transformed data and loads it into the target data warehouse which can then be accessed by end-users and applications. Loading dimension tables and fact tables are a part of this process [4]. Disabling any constraints and indexes before loading along with maintaining referential integrity would ensure that the loading process is efficient.

## **2.2 ETL for Geo-Data**

### **2.2.1 Why is it important?**

Geospatial data contains information that describes features with respect to a location on earth. Usually a combination of latitude and longitude along with related information about the location such as weather data, traffic data, population data, etc. Similar to tabular data, geospatial data is stored in a variety of formats, standards, and multiple schemas, platforms, and systems, and are stored in proprietary databases and Geographic Information systems monitored by organizations or government [5]. These geospatial data stored in multiple formats and locations combined with each other can provide vital insights about a particular location and the relation between the various features of that location. This combination of multiple geospatial data can be facilitated using the ETL approach where the geospatial data can be extracted from various organization/government source, transformed to combine various information and converted to unified format as suggested by [6], and store it in a common location from where geospatial analysis can be performed.

### **2.2.2 How is it different?**

But, compared to ETL performed on normal relational DBMS data, ETL to link various geospatial data would be different due to the challenges faced while handling geo-data. The first challenge lies in the complexity of geospatial concepts in which a pair of latitude/longitude serve as a point location, or a combination of them might be used to represent a polygon or curve which makes the data transformation process difficult i.e. whether to convert all points into standalone latitude/longitude or create polygons/curves. Second, apart from the geospatial concept complexity, geospatial data also suffers from non-unique formats, i.e. every geo-data collecting organization/agency depending on the location that is collected and stored, follow varying standards which makes the process of combining two geo-spatial datasets impossible, with some exceptions [5, 6].

## **2.3 Approaches to ETL**

Taking into consideration various factors like the amount of time available to build the ETL process, the budget for the project, the complexity of the ETL process to be implemented, and the number of software developers available, most organizations have the following options to implement ETL [7]:

1. Writing custom code using python, C, or SQL to perform extraction from multiple sources, transforming the data, and load to the data warehouse.
2. Subscribe or purchase a software tool-based ETL solution that performs the required functionality needed to be achieved.
3. A hybrid solution that involves writing a custom code for some aspects of the ETL process while using a software tool for another.

### **2.3.1 Code Based-Solution**

The ETL tools developed through this solution are coded from scratch using a process like CRON job or a code that can be written on top of existing frameworks (programming language-specific or cloud-based) using supported programming languages to implement custom functionality. The ETL tools developed from scratch use a mix of programming languages i.e. Perl, Python, or C for extraction and transformation of data and SQL for loading into the data warehouse [8]. For the framework-based solution, based on the programming language of preference or the current cloud services provider, an ETL framework is chosen based on requirements [9]. Although coding from scratch provides a tailored solution, programs written are slow to execute, hard to document as they are lengthy, and most importantly are not flexible to changes in high volumes of data and require frequent changes [8].

### **2.3.2 Software Based-Solution**

To overcome the challenges of code-based solutions, various vendors have developed ETL tools to perform the extraction, transformation, and loading. An important function of these tools is to generate and maintain centralized metadata. Initially, these tools provided the ability to extract the data from mainframe computers and load it into the target database. But the ETL tools have matured enough to provide user-friendly GUI's, additional functionalities, and performance benefits with the ability to perform real-time ETL. The ETL tools may be code-generators (examples include Talend, Informatica, Oracle Warehouse builder, etc.) or engine-based tools (examples include Pentaho, Tableau-prep, etc.). These GUI-based tools are friendly to use, provide ease of development, by either generating the code or enabling the ETL process through its internal engine. They reduce development, maintenance time

and cost, with the ability of being used by someone with no prior training of the tool.

### 2.3.3 Challenges to performing ETL

Being a complicated process, there are various challenges at every stage of the ETL process, some of the general ones are:

- Data management problems occur at all the stages of an ETL pipeline if the operational data volume is extremely large [10].
- Data pre-processing is required even before performing ETL process.
- ETL pipeline would have to be remodelled every time there is a change in the data structure of the data source. One solution to this could be designing a near real-time ETL environment [11].
- The need for temporary data warehouse during intermediate stages of the data transformation process.
- During the data transformation process, ensuring that when joining the transformed data with the master data, the master data does not get compromised [12].
- Variable loading times for the transformed data to the required data warehouse.
- Simple SQL commands are not sufficient for data loading process as it is not optimized. Parallel loading may be required according to the width of the transformed data.

## 2.4 Testing techniques for ETL

There are various ETL tools out there. Each with its own capabilities and shortfalls. In order to test whether or not a given ETL solution is performing all the required tasks such that the data is transformed correctly according to the business requirements, there is no data loss, the data is loaded within the expected time, and scales according to the increase in data size [13]. Thus, it becomes vital that there be a rigorous and well-defined testing methodology that can be applied to accurately test out the ETL tools. Below are the various types of tests that can be performed [3]:

*Production Validation Testing* - performed on data which is sent to the production system, it helps in maintaining accuracy in the data by information analysis in the system and comparison with the source data.

*Source-to-target Count Testing* - makes sure to match the count of the actual records with the count of records in the target database.

*Source-to-target Data Testing* - it checks after the transformation process to match the data values in the target database with the original database, e.g. performs a string or numeric match.

*Data Check and Constraint Testing* - This testing performs many database checks constraints like data length check, data type check, and index check. In this, the tester checks for Foreign Key, Primary Key, NULL, NOT NULL, and UNIQUE elements in the transformed data.

*Data Transformation Testing* - it verifies the transformation rules applied by running multiple SQL queries for each tuple and compared with the business rules to see if it matches it.

*Data Quality Testing* - to make sure the ETL application rejects, accepts default values, and reports invalid data, syntax tests (invalid characters, pattern, case order), and reference tests (number, date, precision, null check) are run.

*Regression Testing* - in order to confirm that a recent change in the ETL system has not affected the other parts of the ETL pipeline a regression test is performed. It is a full or partial selection of already executed test cases that are re-executed.

*Data Integration Testing* - it checks that data from all the Extraction sources are loaded into the target database.

*Navigation Testing* - it checks the UI/UX features of the front end which would be used by end-users to perform analysis on the extracted data.

## **2.5 Comparison of ETL Tools**

The selection of the right ETL tool is essential to get the right target data set according to the business logic and requirements. In the following sub-sections, we look at various code-based, software-based and cloud-based ETL solutions, and compare them based on various attributes like environment & architecture, level of automation achieved, programmatic control, scalability of data, reliability and repeat-ability.

### **2.5.1 Coding based tools**

With many ETL frameworks available which can be implemented using the most commonly used programming languages (like Python, C, Java, and JavaScript), Python dominates the ETL space [14]. Thus, we would be looking at the top 4 Python ETL frameworks in use in 2021:

1. Apache Airflow - is an open-source workflow automation and management framework created by Airbnb based on python to create and maintain data pipelines. Although airflow was not specifically designed for ETL, it contains Directed Acyclic Graphs (DAGs) which could be used to schedule and manage ETL pipelines. DAGs are used to create data flows that instructs the operation to be performed on the data at every stage of the graph. DAGs are stored in a metadata database, from where the scheduler selects which tasks from the DAGs are to be performed, and

finally, the executor determines the workers to execute the tasks. Workers are processes that execute the logic of the workflow. Apache Airflow is useful when performing ETL which has multiple steps with sub-steps as it allows to restart at any point after the execution of the process. Airflow is also easy to implement in cloud data warehouse scenarios as it contains Google Cloud and AWS hooks and operators. But, airflow is not useful for small ETL processes as it requires to be deployed onto a server and cannot be used just as a python library.

2. Luigi - is an open-source python package created by Spotify to build complex pipelines of batch jobs [15]. Luigi uses a similar concept as Apache Airflow by using DAGs which execute the tasks and dependencies. But, the main difference between Airflow and Luigi is the way they execute these tasks and dependencies. On Luigi, they are called "tasks" and "targets", with each task consuming their target to complete it. Thus, the target-based approach by Luigi makes it suitable for simple ETL pipelines, as it cannot handle complex tasks. Unlike Airflow, interaction with the ETL process is also not possible once Luigi runs the tasks and its UI is not as user-friendly compared to Airflow.
3. Apache Spark - is an open-source distributed data processing framework for performing analytics on large-scale datasets. It supports Java, Scala, R, and Python. PySpark (Spark's python API) can be used to set up Apache Spark ETL integration. PySpark can read the dataset in the form of JSON files, infer the schema automatically, and load it as a DataFrame using Spark SQL. The loaded dataset can then be transformed using python operation with the help of external python libraries (like pandas and NumPy). And load the transformed data into a PostgreSQL. Apache spark does not provide workflow automation like Apache Airflow and Luigi and is mostly used alongside Airflow for the same reasons. Also, Apache Spark can only handle very simple ETL processes as manually coded complex ETL tasks would be cumbersome to manage without any UI.
4. Prefect [16] - is an open-source workflow engine. Similar to Apache Airflow, Prefect can perform the ETL processes using workflows defined using DAGs. But compared to Airflow, Prefect has more unit tests and greater test coverage. Compared to Airflow, Prefect treats workflows as standalone objects that can be run any time, with any concurrency. Prefect overcomes Airflows limitation when you want to execute a DAG with many tasks, and avoids the single point of failure which can occur in Airflow due to the centralized nature of the scheduler by distributing the scheduler tasks. Prefect also allows to run workflows simultaneously and supports dynamic workflows.

### 2.5.2 Software based tools

1. Pentaho Data Integration - is one of the leading ETL tools available today. Pentaho provides its ETL services in the form of data capturing, cleansing, and storing using a unified and consistent format accessible by the end-users [17]. Pentaho supports Data Migration between different types of sources including cloud-based sources, clustered and parallel processing environments, Data warehouse population with surrogate key generation, and the ability to use real-time ETL [17]. Pentaho also has a user-friendly UI, and drag & drop environment [18] which makes it easy for anyone with any non-technical background to perform complex tasks.
2. Tableau-Prep - is a relatively new tool created by the same company that made Tableau Desktop for data visualization. Tableau prep provides a drag & drop-based code-free tool and is composed of two products: 1. Tableau Prep builder for building your data flows; 2. Tableau Prep Conductor for scheduling, monitoring and managing flow across the organization [19]. Tableau-prep provides a user-friendly UI, easy-to-understand workflow diagram, and direct integration with any relational database, but being new it does not provide tools to perform complex data transformation tasks thus making it suitable to create only simple ETL pipelines.
3. Informatica - is a popular data integration tool provider that provides Data Exchange, Cloud Data Integration, Cloud migration, Data management, and Data Visualization services through its InformaticaPowerCenter tools. The PowerCenter consists of three major components [18] - Client tools to enable the data mapping process, Repository to store all metadata of the application, and Server to execute all the data-related tasks and load transformed data into the target database. Informatica's software tool performs ETL using flow diagrams known as mappings [20], a mapping consists of source and target database along with the transformations which are needed to be applied to the data. Informatica also supports transformations written in C or JAVA and works on all OS platforms. The pricing of Informatica varies depending upon the period of licensing or number of users or servers, but its cost is a bit higher than the rest of the competition as it provides a tailored solution to clients.
4. IBM Infosphere Datastage - is an ETL tool created by IBM that supports Massively Parallel Processing architecture which gives it support for parallel ETL job sequences. Datastage provides easy scalability compared to Informatica and supports all OS platforms and, C++ and Perl scripts in terms of programming language. Datastage is less expensive than Informatica's ETL tool and supports real-time ETL [20] with the updated data sent to the Datastage directly, Datastage lacks the workflow reusability feature of Informatica [20] which means that new workflow needs to be made for each project and also does not support common objects shared between projects. But is still quite expensive compared to the competition.

5. Talend Open Studio - is a Freeware (open-source) ETL tool useful for creating basic ETL pipelines for simple data integration tasks, it is monitoring, and performs the role of converting the data transformations into a JAVA script to replicate on other machines [18]. It is usually used to perform minor transformations (like data cleaning) to big data. Talend Open studio is only useful for minor or very simple ETL tasks as it lacks data transformation functionalities to perform complex ETL tasks.

### 2.5.3 Cloud based tools

1. Cloud Data Fusion - is a cloud-based solution by Google providing a drag and drop interface to enable code-free deployment of ETL pipeline. Cloud Data Fusion provides pre-configured or customizable templates to perform ETL tasks. One of the major advantages of using a cloud-based solution over a local solution is that the data being a serverless environment does not face the challenge of scalability to increasing data size. Another advantage of Cloud Data Fusion is the ability to provide real-time ETL solutions with the updated data and provide easy connectivity to other Google Cloud Services. The disadvantage of using Cloud Data Fusion is that it cannot connect to any other cloud services and is the most expensive cloud service compared to its competition.
2. Azure Data Factory - helps build ETL pipeline for Microsoft Azure users. Azure Data Factory is not exactly a full ETL tool, as it helps in orchestrating and migrating data, then performing the complex transformation, but it becomes more complete when combining the data flow and control flow features to migrate data to the data warehouse [21]. The Data flow and control flow features provide the ease to performing ETL without the need to write code and being a cloud platform is not vulnerable to the scalability of data. But just like Cloud Data Fusion, Azure Data Factory cannot connect to other cloud services, and also does not support real-time ETL.
3. AWS Glue - is a serverless data integration cloud-based tool provided by Amazon. AWS glue provides both visual and code-based interfaces to perform ETL tasks. AWS Glue also has inbuilt features like AWS Glue Databrew to normalize data without writing code and AWS Glue Elastic View to use SQL to combine or replicate data [22]. AWS Glue is the least expensive cloud-based ETL solution compared to the other two discussed in this section. Also, combining AWS Glue with AWS Lambda automation service can help perform real-time ETL which gets triggered via AWS Lambda every time new data enters the source.



## 2.6 Database Normalization

In a relational database, normalization is the process of reducing the redundancy in the data by dividing a database into two or more tables in a manner that maintains the relationships amongst the table through the use of keys (e.g. primary key, candidate key, surrogate key, etc.) [23]. The goal of the normalization process is that any changes made to one of the normalized tables would be automatically reflected in the rest of the table through the relationships formed via the keys. One of the tasks that can be performed during the data transformation stage of an ETL process is that of database normalization, which helps in the data cleaning process.

### 2.6.1 Types of Normalization

In the context of a database table, a relation R is defined as a set of tuples or rows in the table, with each item belonging to a particular column (also called an attribute). We would be discussing the four most popular types of database normalization, out of which we would be applying the first three on the client data set for the data transformation task of the ETL process.

1. First Normal Form (1NF) - if all the underlying domains of a relation R contain only atomic values, the given relation R is said to follow first normal form [24] i.e. given a database table all the entries in respective columns of a table should only contain atomic values. In the case where a column does not contain atomic values, split the table into two and use a foreign key to link both the tables.

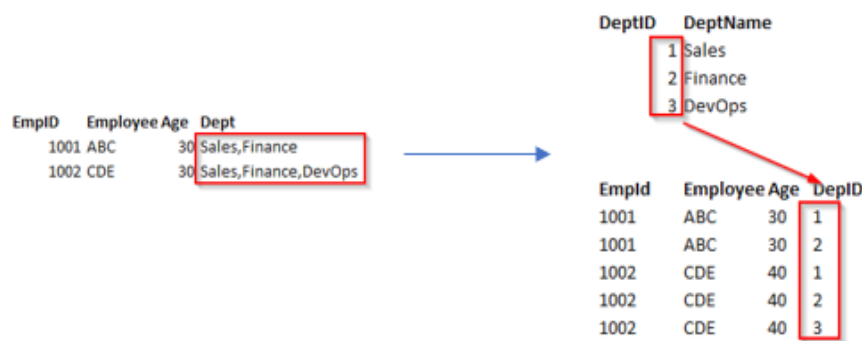


Figure 2: Conversion to 1NF

2. Second Normal Form (2NF) - given a relation R in its first normal form (1NF) and all its non-key attributes are only dependent on the primary key, then the relation R is said to follow the second normal form (2NF) [24] i.e. given a database table in 1NF, should contain all the entries of a respective (non-primary key) columns should only be dependent on primary key column's respective entries. In the case where a database table contains columns that have partial dependencies on columns other than the column forming primary key, split the table into two or more tables,

with each table not containing any partial dependencies.

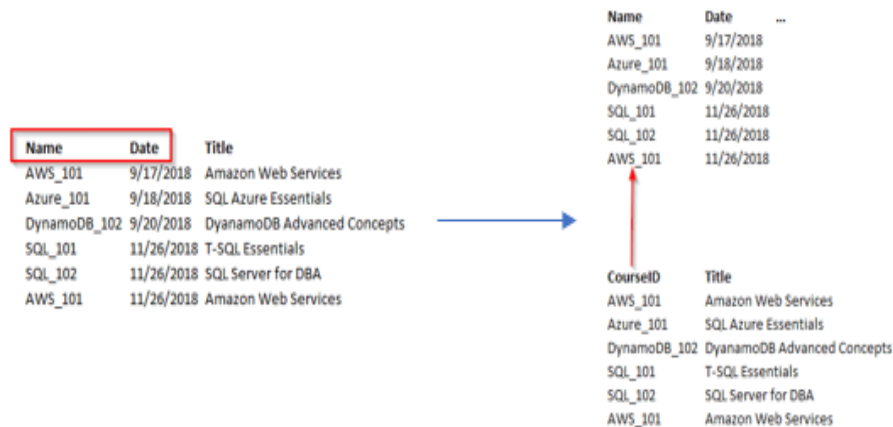


Figure 3: Conversion to 2NF

- Third Normal Form (3NF) - given a relation R in its second normal form (2NF) and all its non-key attributes do not transitively depend on the primary key, then the relation R is said to follow the third normal form (3NF) [24] i.e. given a database table in 2NF, should contain all the entries of a respective (non-primary key) columns to no be transitively dependent on primary key column's respective entries. In the case where a column has a transitive dependency on the primary key column, split the table into two or more tables, with each table not containing any transitive dependencies.

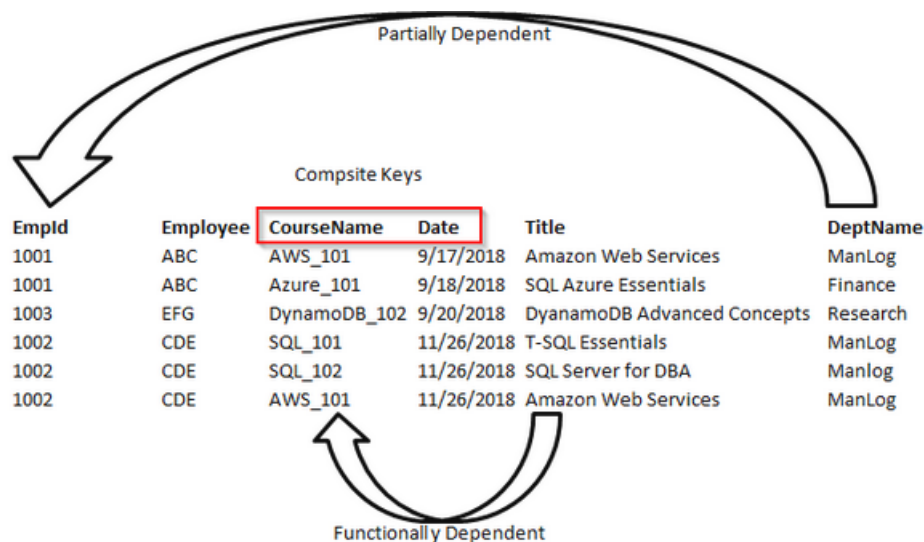


Figure 4: Table not following 3NF

- Boyce Codd Normal Form (BCNF or 3.5NF) - given a relation R in its third normal form (3NF) and if any attribute not in the attribute collection C of R, is functionally dependent on attribute in C, then all attributes in R are functionally dependent on C [24] i.e. given a database table in 3NF is said to be following BCNF if and only if every column in the table is dependent to a column

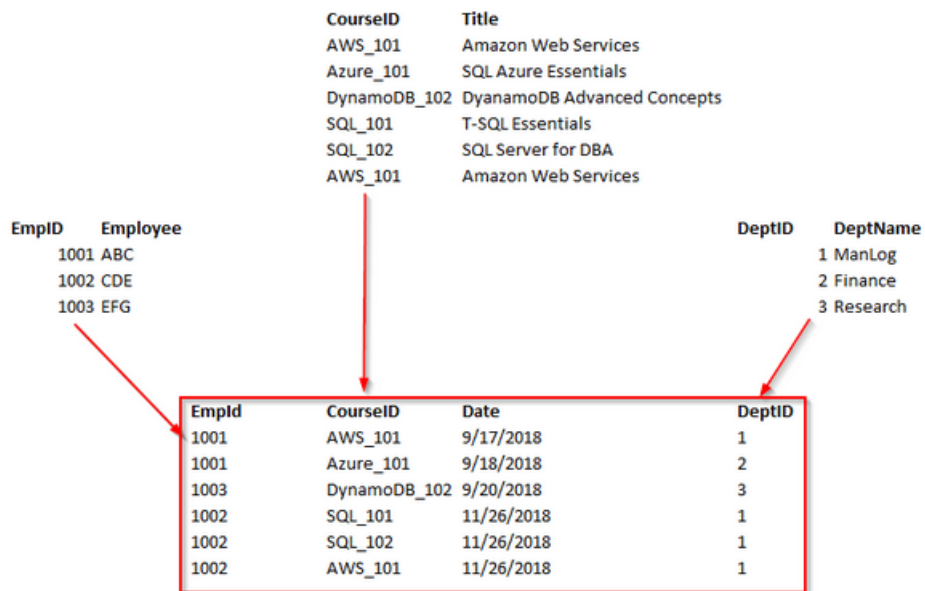


Figure 5: Conversion to 3NF

belonging in the set of super key columns [25]. In the case where a column not belonging to the super key set can derive a super key column, divide the table into two such that each table contains no non-super key column dependency.

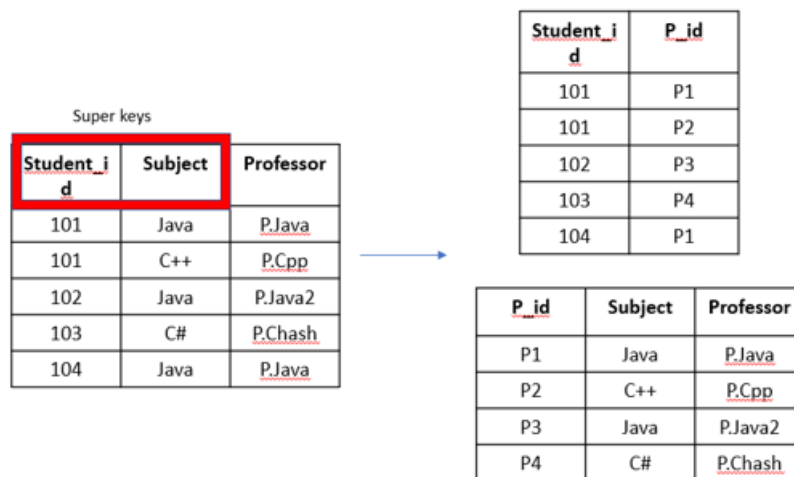


Figure 6: Conversion to BCNF

### 3 Methodology

#### 3.1 Client's ETL Problem

Lithodat provides an online intuitive platform called Lithosurfer to visualize, analyze and extract geospatial data collected from various sources. As the data is collected from various sources, they do not follow a common format i.e. sources do not have a common data schema or in some cases do not follow a supporting data schema. Thus, the collected geospatial data is met with the challenge of conversion to a common schema followed by Lithodat.

The core of the client's ETL requirement is formed by cleaning the collected geospatial data, normalizing and mapping the data to the schema followed by Lithodat, and uploading the transformed data to their common database in a semi-automated manner. The client's ETL problem can be summarized by the below-shown Figure 7.

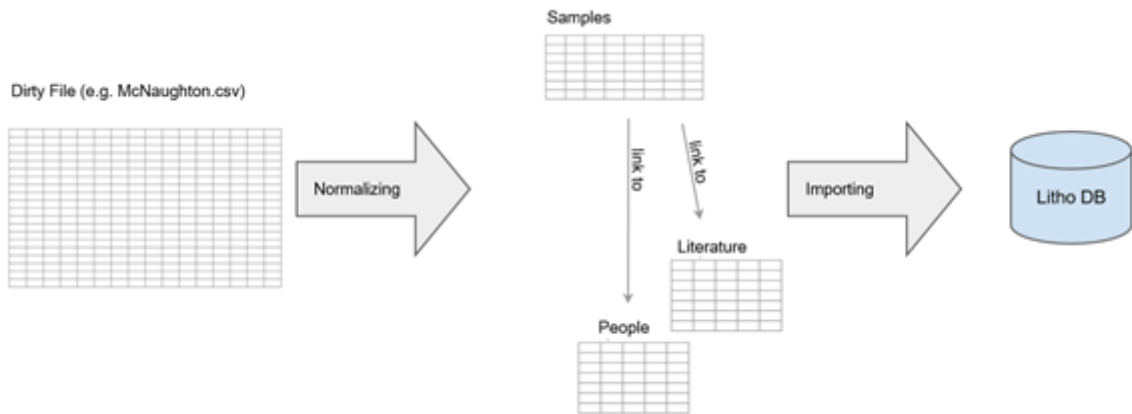


Figure 7: Brief Overview of Client's ETL problem

A part of the data cleaning, normalization, and conversion to data schema tasks would also include splitting of tables into sub-tables, renaming of the columns according to the names in the database schema, and removal of duplicate entities via survivor elements given by the client. As a requirement, the client requested us to only investigate software-based ETL tool for the tasks as the end ETL pipeline solution is to be used by Geo-scientists at Lithodat with minimal to no coding knowledge thereby limiting their ability to modify or extend the pipeline in the future if it were a code-based solution.

#### 3.2 Approach to solving Client's Problem

Taking the client's ETL problem and requirements into consideration, the following approach in terms of steps was considered:

1. Perform a thorough literature review on the available software-based ETL tools including their pros and cons, adaptability to the clients' data set, and matching the clients' requirements.

2. Creating a data schema for the transformed data followed by Lithosurfer database. This task would also include creating an ER (Entity-Relationship) Diagram of the schema.
3. Using the list of potential software-based ETL obtained through the literature review, we would create an ETL pipeline for each tool using a sample dataset (which is different from the original client data) and clients dataset, and test each tool via a subset of the ETL testing techniques as discussed in section 2.4 of the report.
4. Based on the results of the tests conducted, an ETL tool is selected. The ETL pipeline created on the selected tool is then refined for niche details according to business logic to be followed in the data transformation process.
5. The final ETL pipeline is then tested with new client data and the results loaded into the Lithosurfer platform is then tested via source-to-target testing and data check & constraint testing. With the entire process being documented.

### **3.3 Procedure of the tests to be conducted**

As creating an ETL pipeline was a new task undertaken by the client and us, the software tools were not tested directly by creating pipelines for the client data. Instead, a sample bank dataset was created and an ETL pipeline was generated to test out the various data transformation features of the software tools. Once the testing on the sample dataset was performed and the capability of the tools was known, the tools were then tested by creating an ETL pipeline for the clients' dataset using the tools.

In terms of the subset of ETL testing techniques used, Data transformation testing, Regression testing, source-to-target data testing, Data Integration Testing, and Data Check & Constrain testing were selected to assess the tools. The tests, results, and findings on each tool are described in section 5 of the report.

## 4 Data (Exploration & Analysis)

### 4.1 Data Description

#### 4.1.1 Sample Dataset

A sample data was created to initially test out the ETL tools and to get familiarized with the features and functionalities of the tools. The data mirrored what the client expected from us in terms of the ETL tasks to be performed in terms of data normalization. The schema for the sample dataset was created and an ER diagram was generated for the same as shown in Figure 8.

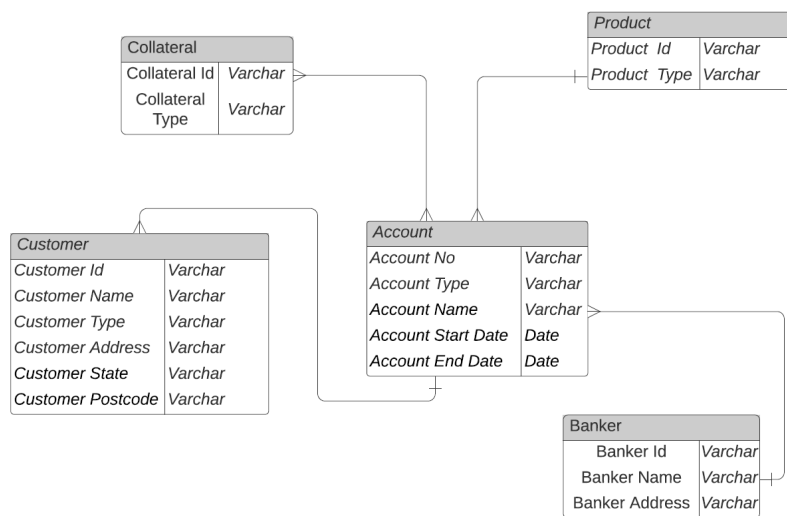


Figure 8: Sample Data ER Diagram

As we can see from the ER diagram there are multiple many to many relationships among various entities. The Entities involved in the above ER diagram are explained below:

- Account* - This is a physical account the bank opens when a customer deals with a financial entity. In our dataset, we had various attributes which were unique to the Account. These attributes include: Account No, Account Type, Account Start date, Account End date.
- Customer* - A human being or an entity(company) that deals with the bank. The attributes for the customer include: Customer Id, Customer Name, Customer Type, Customer Address, Customer Post Code.
- Product* - The financial instrument which the bank offers the customer. The attributes of the product included: Product Id, Product Type.
- Collateral* - The asset against which the customer takes the loan. Collateral may not exist if a

customer is opening a savings account or a term deposit. This entity secures the loan which the bank offers to any customer. Attributes associated with the entity include: Collateral Id, Collateral Type.

- e) *Banker* - A bank employee details who deal with the customer when they come to the bank to buy any financial instrument. The banker may be the clerk or the manager who has opened the bank account for the customer. Banker attributes include: Banker Id, Banker Name.

Apart from the data normalization task, another task was to maintain the referential integrity among the dataset, for example not forming a customer account pair if it did not exist. Also, multiple relationship entity objects were created by the team for this example, which included:

- a) *Account-Collateral* - Relationship between Account and Collateral
- b) *Account-Customer* - Relationship between an Account and Customer
- c) *Account-Product* - Relationship between Account and Product
- d) *Customer-Banker* - Relationship between Customer and Banker

#### 4.1.2 Client Dataset

A geo-data was given to us by the client to test out the selected ETL tools. Before using the data to test out the ETL tools, a data schema was needed to be created according to the attributes stored in various tables in the Lithosurfer Database. This data schema would then be used to perform normalization during the data transformation stage and also be used to perform Data Check and Constrain Testing by verifying the constraints and keys of the transformed data.

Based on the dataset and the description provided by the client, below shown Figure 9 was created, which gives the ER diagram representing the data schema to be followed. The Archive, Sample, Literature, Location, Person, Funding, and Machine sub-tables are formed as part of the database normalization process dividing the original table with 40 columns into their respective sub-table categories.

As we can see from the ER diagram, there are multiple one-to-one relationships and a single one-to-many relationship between the entities of the normalized data. The Entities involved in the above ER diagram are explained below:

- a) *Sample* - Details of the sample where it is collected.
- b) *Literature* - The literature for the sample which includes any reference taken for the sample or when it was last published.
- c) *Location* - The location where the sample was recorded by the Person entity including the latitude

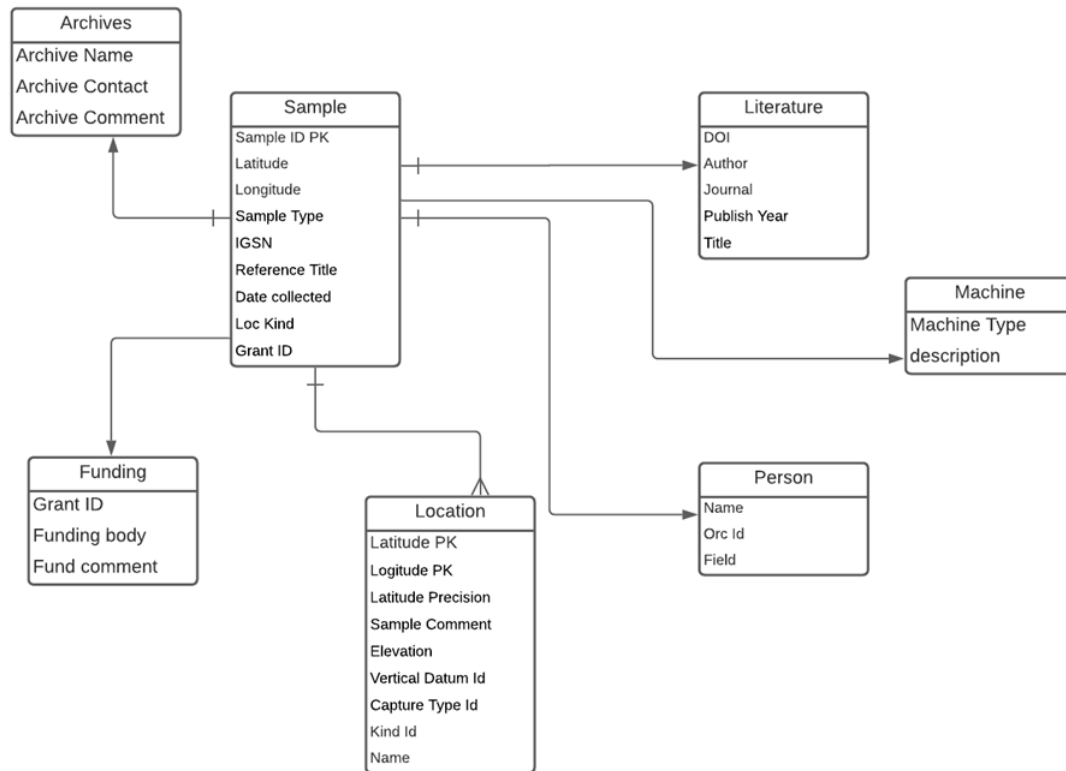


Figure 9: Client Data ER Diagram

and longitude.

- d) *Person* - An entity that was responsible for recording the sample for Lithodat. The entity could be a person or a company.
- e) *Funding* - Funding provided if any for the sample to be collected.
- f) *Archives* - An Archival information present for the sample.
- g) *Machine* - Machine used for collecting the sample.

Some keys can not be provided by the client. Thus, one of the tasks for our team was generating the unique key for the needed entities i.e. for Person and Location. For the Person table, an additional column was created to uniquely identify each person by their ID. And for each location, we consider the combination of the latitude and the longitude as the Primary Key for the respective location.



## 4.2 Data Dictionary

The data provided by the client lacked description, making the process of ER diagram generation harder. Thus, a Data Dictionary was created with inputs from the client. This Data Dictionary was then used to understand the data better and generate a schema for the client based on their requirements.

| Field Name           | Description   | Data Type |
|----------------------|---|-----------|
| SampleID/AnalyteID   | Unique identifier for each sample/analyte row                             | Integer   |
| IGSN                 | International Geo Sample Number (IGSN) for each physical sample           | Varchar   |
| IGSN URL             | URL to handle site for the respective IGSN                                | Text      |
| Reference_DOI        | Digital Object Identifier to identify Literature (document/article/paper) | Varchar   |
| Reference_authors    | Authors of the literature   | Text      |
| Reference_journal    | Name of the journal the reference is published in                         | Text      |
| Reference_year       | Year of publishing of literature  | Integer   |
| Reference_title      | Literature title  | Text      |
| lat; WGS84, decDegr  | latitude of sample/analyte using World Geodetic System 1984               | Double    |
| long; WGS84, decDegr | longitude of sample/analyte using World Geodetic System 1984              | Double    |
| lat/long precision_m | precision accuracy of spatial location in metres                          | Double    |

|                                       |  |         |
|---------------------------------------|--|---------|
| sample comment                        | comments regarding sample                                    | Text    |
| Elevation_m                           | height above sea level for the respective location of sample | Double  |
| Vertical_datum_id                     | ID of datum used to defined elevation reference              | Varchar |
| Location_capture_type_id              | ID of location from where sample/analyte was captured        | Varchar |
| Location_kind_id                      | unique identifier for location                               | Varchar |
| Loc_name                              | Name of location from where the sample/analyte was obtained  | Text    |
| Loc_comment                           | Comments regarding the location                              | Text    |
| Sample_type                           | Type of sample (eg. core, rock, fossil etc.)                 | Text    |
| Lithology_id                          | unique identifier for lithology of sample                    | Integer |
| Mineral_id                            | unique identifier for mineral of sample                      | Integer |
| Stratigraphic unit (name) GA_list     | Geoscience Australia's stratigraphic unit name               | Text    |
| unit_age (Eon, era, period,epoch,age) | Geological age period of stratigraphic unit                  | Text    |
| unit age_min                          | Stratigraphic unit minimum age                               | Text    |
| unit age_max                          | Stratigraphic unit maximum age                               | Text    |

|                                    |   |         |
|------------------------------------|---|---------|
| depth_min                          | Default - Vertical min depth of a sample in a drillhole | Integer |
| depth_max                          | Vertical max depth of a sample in a drillhole           | Integer |
| Date Collected (minimum)           | sample date collected (min)                             | Date    |
| Date Collected (maximum)           | sample date collected (max)                             | Date    |
| Person                             | Name of person who collected sample                     | Text    |
| OrcID                              | unique identifier for person field                      | Varchar |
| Last known sample archive location | Location where sample <sup>c</sup> is/was stored        | Text    |
| Archive contact                    | contact name  | Text    |
| Archive comment                    | comment   | Text    |
| Grant_ID                           | unique identifier for Funding                           | Varchar |
| Funding_body                       | name of funding body                                    | Text    |
| Fund_comment                       | comment regarding funding                               | Text    |

Table 1: Data Dictionary

### 4.3 Data Exploration

After performing an initial analysis on the data, we can see that the original non-transformed dataset provided by the client contained 3152 entries and 59 columns. Table 2 provides a list of the columns and the number of null/empty entries in the respective column. We can see that most column entries are empty, but the columns which form the primary key for the normalized data (e.g. SampleID/AnalyteID) have no null entries. The non-key columns - lat, long, lat/long precision\_m, elevation\_m, unit\_age (min & max), and depth (min & max) are of type float while the rest of type character except Date Collected which has Dated as its datatype.

During our exploratory data analysis task, we found out that there are a lot of missing data across all the entities. The missing data caused the identification of duplicate entries an issue. To resolve this

issue we looked at the importance of the missing value and had frequent catches up with our client to mitigate this risk. The reason for missing this data was explained to us that

- It is possible that there were problems with the extraction process. In such cases based on the evidence and experience, it can be corrected easily.
- These errors occur at the time of data collection and are harder to correct. They can be categorized as - Missing completely at random: This happens when the person who is responsible for logging data misses to log it. This is a human error and is generally one of the most frequent ways of why the data was missing from the file. Missing that depends on unobserved predictors: This is the case when the person who is supposed to log in the data is missing some information regarding the place where the sample was taken and how it was recorded.

| Columns                               | # of null/empty entires |
|---------------------------------------|-------------------------|
| SampleID/AnalyteID                    | 0                       |
| IGSN                                  | 355                     |
| IGSN URL                              | 355                     |
| Reference_DOI                         | 2014                    |
| Reference_authors                     | 253                     |
| Reference_journal                     | 1094                    |
| Reference_year                        | 247                     |
| Reference_title                       | 1148                    |
| lat; WGS84, decDegr                   | 30                      |
| long; WGS84, decDegr                  | 30                      |
| lat/long precision_m                  | 137                     |
| sample comment                        | 3152                    |
| Elevation_m                           | 1447                    |
| Vertical_datum_id                     | 822                     |
| Location_capture_type_id              | 0                       |
| Location_kind_id                      | 0                       |
| Loc_name                              | 3064                    |
| Loc_comment                           | 3086                    |
| Sample_type                           | 0                       |
| Lithology_id                          | 121                     |
| Mineral_id                            | 3152                    |
| Stratigraphic unit (name) GA_list     | 495                     |
| unit_age (Eon, era, period,epoch,age) | 334                     |

|                                    |      |
|------------------------------------|------|
| unit age_min                       | 270  |
| unit age_max                       | 261  |
| depth_min                          | 2156 |
| depth_max                          | 2868 |
| Date Collected (minimum)           | 3152 |
| Date Collected (maximum)           | 3152 |
| Person                             | 434  |
| OrcID                              | 1523 |
| Last known sample archive location | 82   |
| Archive contact                    | 82   |
| Archive comment                    | 3152 |
| Grant_ID                           | 3126 |
| Funding_body                       | 3126 |
| Fund_comment                       | 3152 |

Table 2: Columns and respective no. of null/empty entries

## 4.4 Data Cleaning

The client dataset involved two major data cleaning tasks (both are solved via the ETL pipeline):

- i) The first task is of removal of duplicate entries (or duplicate rows) that includes rows with identical entries for each column. This is done by performing a character match or numeric match (depending on the data type followed by the column) and discarding the redundant values.
- ii) The second task involves pairs of columns - `litDuplicateId` & `litDuplicateSurvivor`, and `persDuplicateId` & `persDuplicateSurvivor`. The 'Duplicate Survivor' column contains a True or False value corresponding to a 'Duplicate Id' column which indicates that the respective literature name or person name spelling associated with it is correct and any other DuplicateId value similar to the true DuplicateSurvivor needs to be replaced with the correctly spelled name.

## 5 Testing ETL Tools

From the literature review performed and based on the client's requirements, it was found that the software tools Pentaho and Tableau-Prep were the ideal solutions because both of them were code-free i.e. software-based solutions, and were able to construct simple ETL pipelines efficiently.

## 5.1 Pentaho Software

### 5.1.1 Using Sample Dataset

The following flow (Figure 10) was created to test out the sample banker dataset. An overview of the Figure shows us that the non-transformed data (in CSV format) is being sent as an input, and various data cleaning and normalization steps are applied to eventually give out the transformed data. The

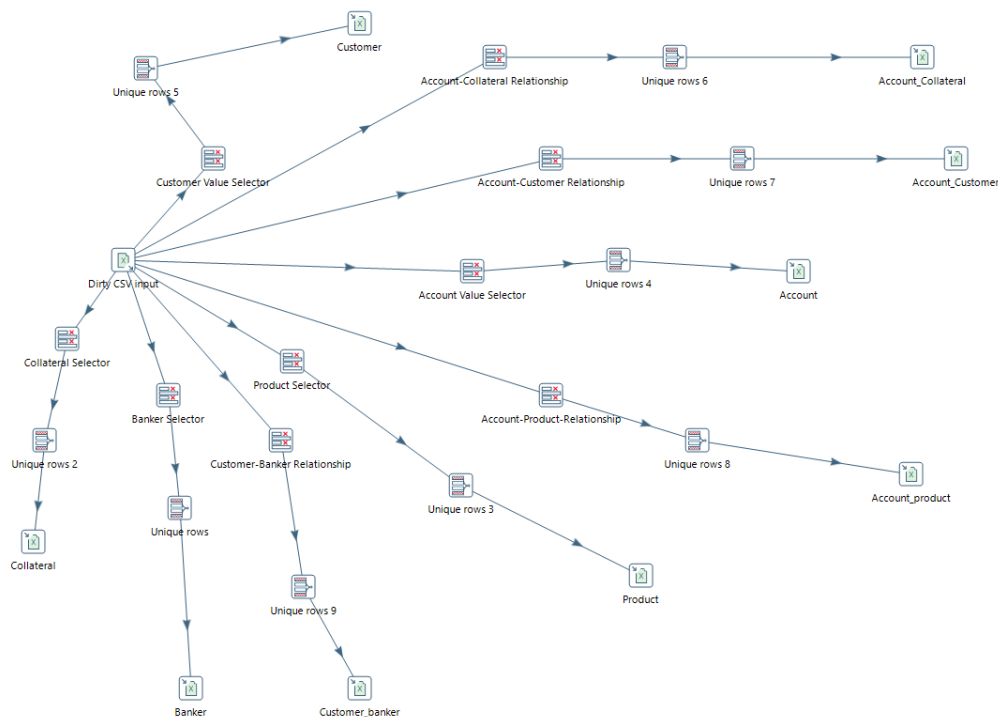


Figure 10: Sample Dataset Flow - Pentaho

entities - Customer, Account, Product, Banker, and Collateral perform the following steps:

- Value selector - Selecting the entries from the original data corresponding to the respective entity it belongs to.
- Unique Rows - Remove duplicate entries present in the entities based on the unique Id generated for each of the entries such that the relationship between entities (e.g. account & customer) have all required entries.
- Output - the extracted data for each entity is then received as a CSV output file.

In order to maintain the Account-Customer, Account-Collateral, Account-product and Customer-Banker relationships between the entities, the following steps were performed:

- Account-Customer, Account-Collateral, Account-product, and Customer-Banker relationships were derived using the extracted values of Account, Customer, Collateral, Product, and Banker respectively by comparing the customer, collateral, and product associated with each account, and matching the bank each customer has an account in.
- Each relationship is then extracted as a CSV output file, to check for correct transformation and perform referential checks.

### 5.1.2 Using Client Dataset

Being more complex than the Sample Dataset, along with primary keys not present for some of the subtables, a more complex flow was created for the client data in Pentaho as shown in Figure 11. Along with the data cleaning and normalization steps, the data transformation steps also included the creation of surrogate keys to maintain relationships between the sample and the rest of the entities. Similar to

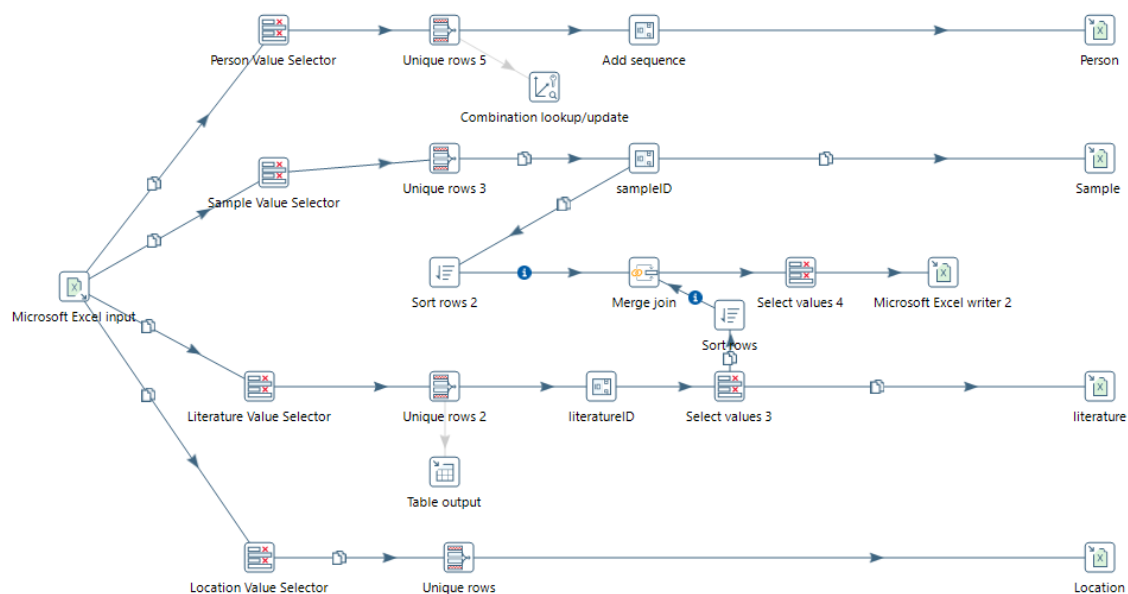


Figure 11: Client Dataset Flow - Pentaho

the sample dataset, the client data flow involves reading in an MS Excel file and then dividing the data into sub-tables. The Person, Sample, Literature, and Location go through the following steps:

- Value selector - Selecting the entries from the original data corresponding to the respective entity it belongs to.
- Unique Rows - Remove duplicate entries present in the entities based on the unique Id generated for each of the entries such that the relationship between entities (e.g. account & customer) have all required entries.

- Add sequence - a random number generated between a defined range, assigned to each value of the corresponding entries of the entity acting as a surrogate key for relationship generation.
- Output - the extracted data for each entity is then received as a CSV output file.

Note that the Location sub-table does not require the additional sequence step, as the latitude & longitude values of the given location act as the surrogate key for the Location sub-table. To maintain the entity-relationship, the surrogate keys were used to:

- Perform an inner join between them e.g. between sample and literature in order to get the ID's for the literature associated with each sample item.
- Select the required columns
- Extract the surrogate keys relationship as a CSV output file to perform referential checks.

Table 3 discusses the results of the tests conducted using a subset of The ETL testing techniques mentioned in section 3.3.

| ETL Tests                   | Results   |
|-----------------------------|---|
| Data transformation test    | Four sub-tables created for the sample and client dataset formed respectively as part of the transformation process according to the created schema.                          |
| Source-to-target data test  | Unique elements from the sample dataset matched with elements present in the sub-tables   |
| Data Check & Constrain test | Relationships between entities tested via the extracted surrogate keys table and relation sub-tables for client and sample dataset respectively to perform referential checks |
| Regression test             | Appending or modifying to the created flows to add/remove transformations did not affect the entire flow, and maintained reference integrity                                  |
| Data Integration Testing    | Loaded the extracted data into the Lithosurfer platform successfully  |

Table 3: Test results - Pentaho



From the tests, we can see that Pentaho passed all the tests. But lacking in UI/UX user-friendliness, having a deep learning curve, and having custom pricing made it costlier than the other tools discussed in this section. Thus, not aligning with the clients' requirements, Pentaho was not utilized for the final ETL pipeline.

## 5.2 Tableau Prep

### 5.2.1 Using Client Dataset

The original data includes all the information that is related to the sample including the sample itself, references that are related to the sample, and the person who collected the sample, and so on. The main process is listed in Figure 12.

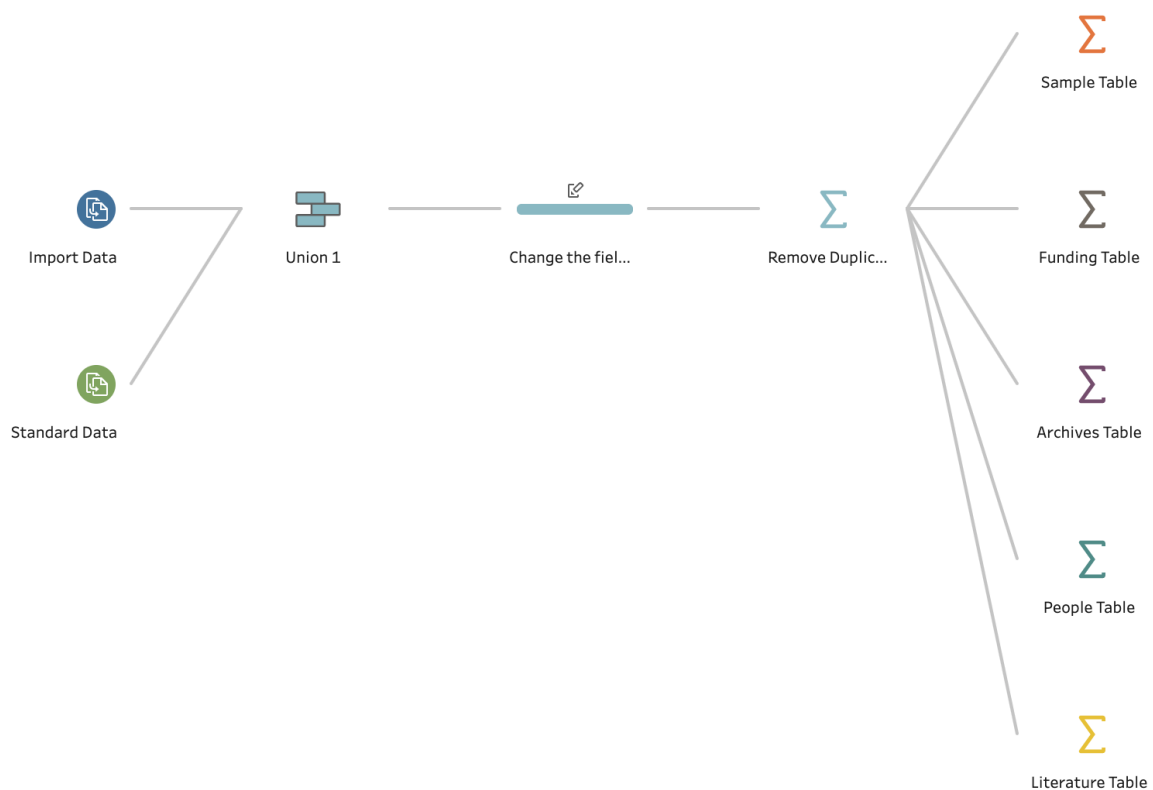


Figure 12: Split into sub-tables

- Merge Standard Table with Original Data - the data provided by the client does not guarantee that all categories exist in the data. There may be some missing columns that prevent it from loading into the lithosurfer database. To avoid missing categories, it is necessary to create a standard table that covers all the columns that need to be loaded into the database. This standard table does not have any data inside. Thus, this step is to compare the standard table with the client's data and then merge them together.
- Handle Duplicate Data - all the duplicate data should be removed. Tableau Prep provides inbuilt functionality to remove duplicate data based on exact string match.

- Generate Sub-tables - dividing the tables into five sub-tables according to the types of entities, namely the Funding table, Sample table, People table, Literature table, and Archives table. Since there are only three categories (Sample, Literature, and People) to load data in lithosurfer at the moment, thus the workflow focuses on processing these three tables while the other two tables have no other operations.
- Select True Value - this step performs data processing on each sub-table. It includes the steps of correcting, merging all the typos, and selecting the True value according to the survivor (as discussed in section 4.4), and then adding the foreign key in the main table before we export the sub-tables. This process is shown in Figure 13.

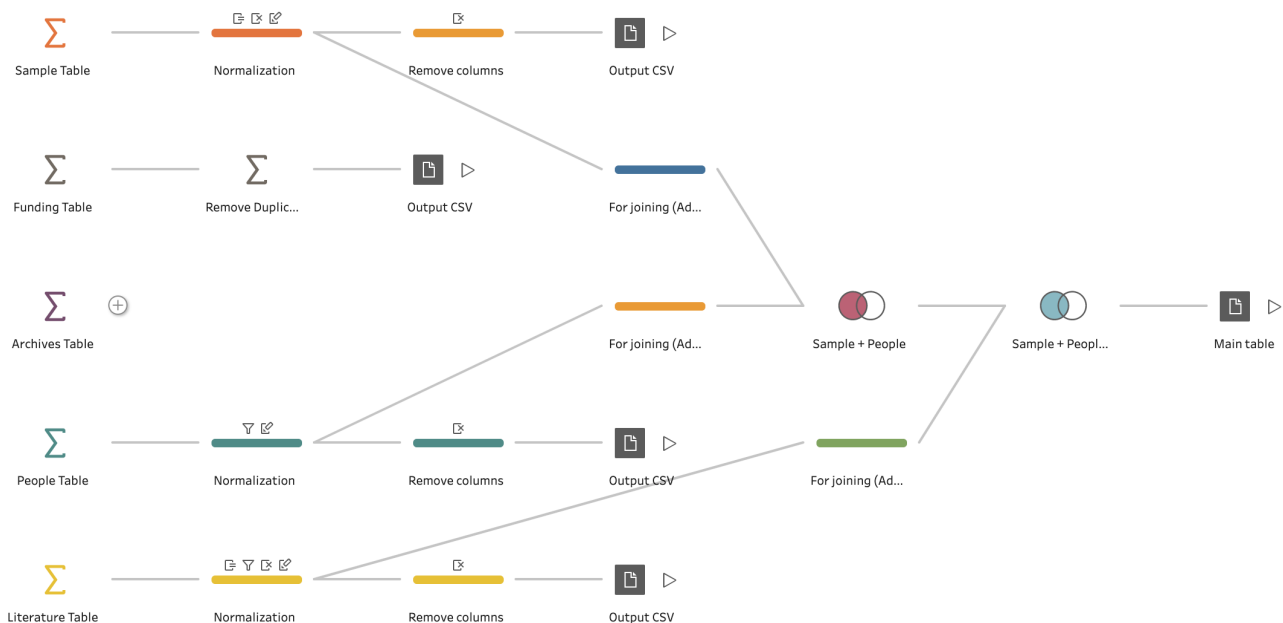


Figure 13: sub-tables Normalization

- By using the "Selected Values" function in Tableau Prep, the "True" values in client's data can be kept, while other information can be filtered out. The "True" values in the people table and literature table are the columns that our clients marked as the correctest one among the other similar contents. This process can turn the typo or wrong content into the correct one. For example, if multiple samples are collected by the same person, but the collector's name related to those samples has slight differences due to typo or format differences such as uppercase, lowercase, and spaces, then the workflow can select only the correctest value among those name. The client provided the true value and the workflow only keeps those "True" values only.
- After selecting the survivor items, a left join of these sub-tables is processed according to their ID's and the main table is generated. The purpose of this main table is to help the client to find the association between these sub-tables conveniently by providing the foreign

keys. Finally, the workflow outputs the sub-tables as .CSV file which is then to load into lithosurfer database.

The ETL flow for the client dataset was then tested using a subset of the ETL testing techniques mentioned in section 3.3. Table 4 discusses the results of the tests conducted.

| ETL Tests                   | Results   |
|-----------------------------|---|
| Data transformation test    | Transformed data matched with the created schema and business rules mentioned by the client                                     |
| Source-to-target data test  | Unique elements from client dataset compared with extracted sub-tables for match  |
| Data Check & Constrain test | Referential integrity maintained by performing unions between sub-tables of the transformed data and checking for relationships |
| Regression test             | Individual flows modified and new flows added did not affect the execution or sub-tables relationships                          |
| Data Integration Testing    | Extracted data loaded into Lithosurfer platform successfully  |

Table 4: Test results - Tableau-Prep

Tableau-prep passed all the tests conducted and overcomes the shortcomings of Pentaho by providing a user-friendly UI/UX experience, optimized flow with lesser execution time, and a subscription plan cheaper than Pentaho. Making it an idea choice for the client, Tableau-Prep was chosen as the tool to create the final ETL pipeline with.

### 5.3 MS Excel

Being a trivial data engineering tool, the client also requested to test out MS Excel for the process of data cleaning. The idea was to use the ETL tools along with MS Excel, where Excel would clean the data and the ETL tools would perform the normalization task. The following data cleaning tasks were performed:

- Removing extra spaces in the data row at the end of the column data and spell check
- Select & develop transformation logic for all blank cells to reduce Null Pointer exception when

uploading data

- Removing Duplicates using the *Transform Data* plugin.
- Change text fields to Lower/Upper case
- Parsing column values into multiple cells wherever necessary

After testing out Excel to perform the various data cleaning tasks, we found that trivial tasks of data cleaning can be manually intensive and can be performed by any of the ETL tools discussed above in a more automated and efficient manner.

## 5.4 Google Sheets

Google Sheets is very similar to MS Excel. It is hard and complex to arrange an automated process for geoscientist's to use without any related backgrounds going against the client's requirements. If some steps do not work well, it is hard to figure out what is wrong quickly and fix it. It is time-consuming to understand and learn how to use Google Sheet to complete the tasks when the tasks can be done easily and smoothly in the professional ETL tools discuss in this section .

Based on the tests conducted, it was found that Tableau-Prep passed all the tests and closely matched the clients' requirements. Thus it was decided to go ahead with it.

## 6 Final ETL Pipeline

### 6.1 Extraction

The first step of the ETL pipeline involves the extraction and collecting of data from various sources and bringing it in one place. In the case of Lithodat, most of the unstructured or non-normalized data is legacy data i.e. data collected from various researchers or laboratory networks. This data is in the form of Tables stored in CSV (comma separated) or .xlsx (MS Excel) format. This collected data is then unified into a single excel worksheet upon which the next step of the ETL process, data transformation is applied in order to perform data cleaning and normalization tasks.

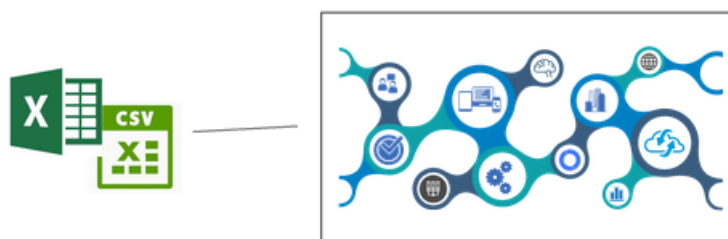


Figure 14: Data Extraction

### 6.2 Transformation

The second step of the ETL pipeline is that of data transformation. The extracted data needed the following data transformation:

1. Removal of duplicate entries that includes rows with identical entries for each column.
2. Normalizing the dataset by splitting it into sub-tables with links (via Primary and Foreign Keys) among them.
3. Pick the 'Survivor' elements from *litDuplicateSurvivor* and *persDuplicateSurvivor*, and replace similar ID's with the correct spelled literature and person name.
4. Cleaning and formatting the normalized data.
5. Output the result normalized data in .CSV format.

Figure 15 provides an overview of the process. After cleaning, formatting, and normalizing the data, the transformed data (in CSV format) is then sent for the data loading process.

#### 6.2.1 Data Normalization and Cleaning using Tableau Prep

After discussing with the client, the final pipeline is updated as below to present a clearer, briefer, and more understandable workflow.

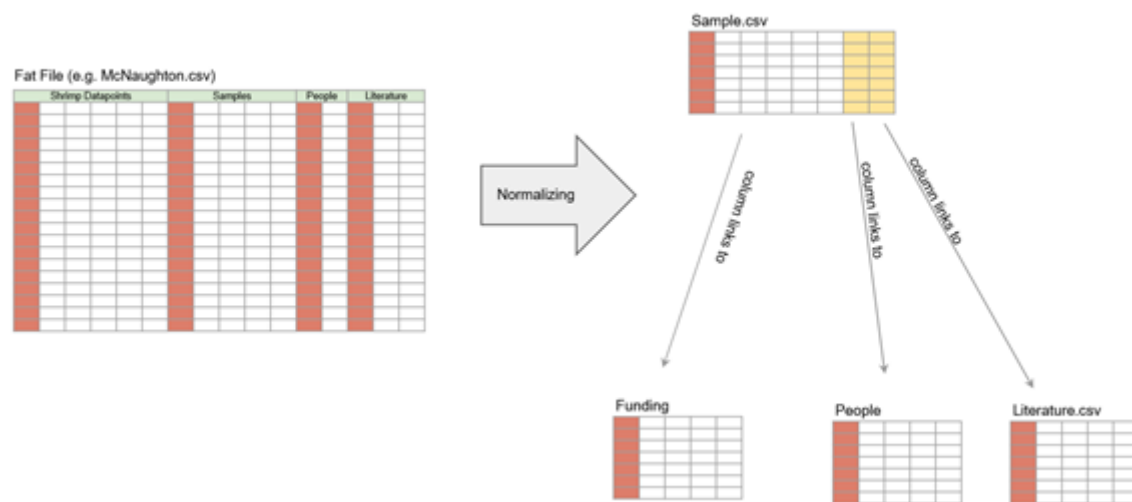


Figure 15: Data Transformation Overview

The biggest difference between the final pipeline and the previous one is we union the Standard Headers Files (Sample headers, Literature Headers, and Person Headers, which contains the headers required by Lithosurfer's database) with the corresponding sub-tables instead of union with the input dirty file at the beginning of the pipeline as it shown in Figure 16.

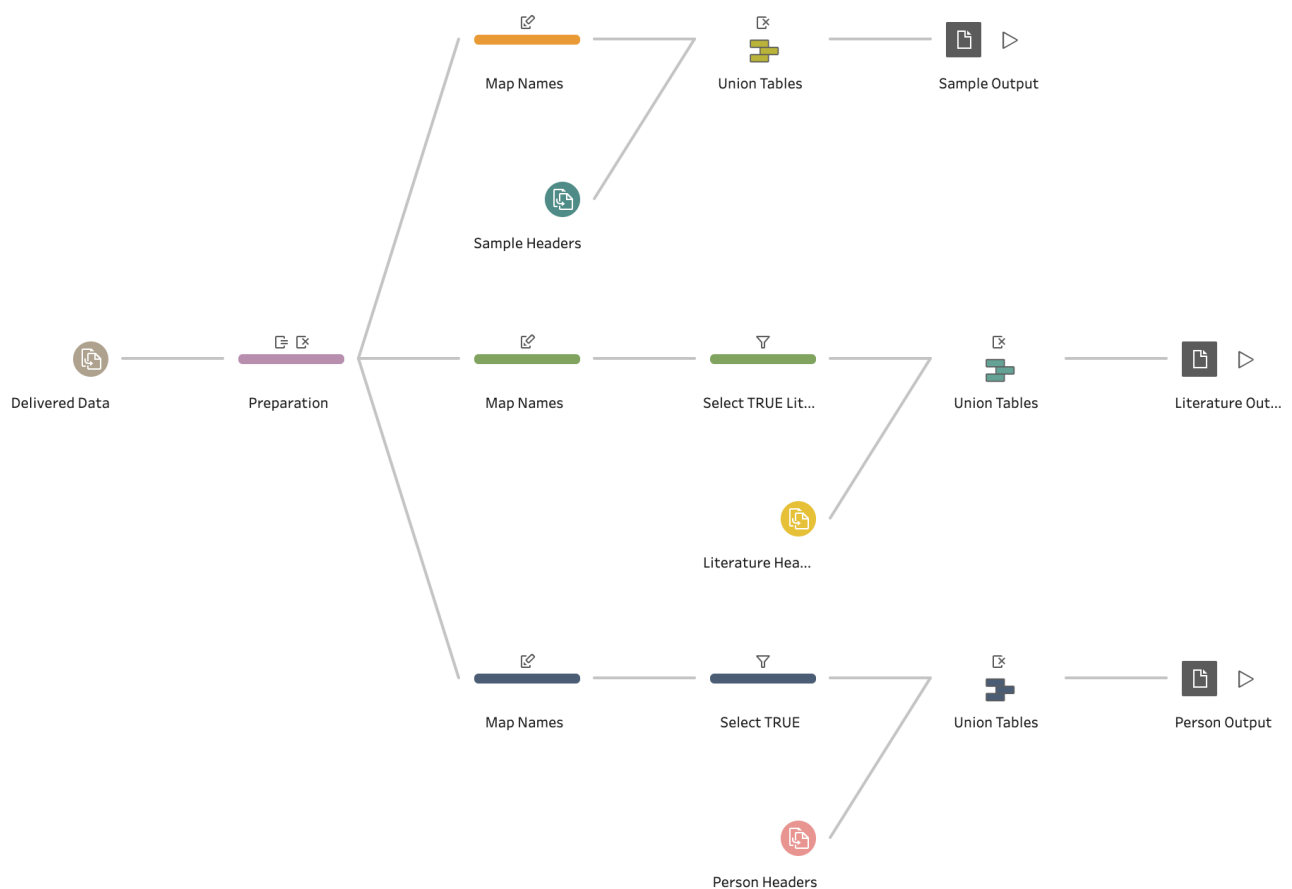


Figure 16: Final Workflow

Following is a description of the data transformation steps:

- *Delivered Data* step - input the dirty file for further cleaning and formatting.
- *Preparation* step - some fields not needed in the sub-tables are removed and a column “person” created which combines the first name and last name columns.
- After these, three branches are built for three sub-tables (Sample table, Literature table, and Person table)
  - a) The first branch:
    - *Prepare* step - rename the needed field headers in the Sample table to match with the standard names in the Sample header file.
    - *Union* step - union the standard headers with the sub-table to check if needed fields for Lithosurfer’s database are contained in the Sample sub-table or not. If not, go back to the *Prepare* step to check and do corresponding operations. Besides, remove not needed fields from the sub-table (i.e. fields corresponding to the Literature sub-table or Person sub-table).
    - *Output* step - output the resulting Sample sub-table as a worksheet named “Sample” in the Excel file named “output”.
  - b) The second branch:
    - *Map Names* step - rename the needed field headers in the Literature table for matching with the standard names in the Literature header file.
    - *Select TRUE* step for Literature - use the function Filter built-in to Tableau Prep to select the Literature marked as TRUE according to the survivor elements (as discussed in section 4.4-ii) and keep the entities associated with the TRUE survivor values or values matching the survivor ID.
    - *Union* step - union the standard headers with the sub-table to check if needed fields for Lithosurfer’s database are contained in the Literature sub-table or not. If not, go back to the *Map Names* step to check and perform corresponding operations. Besides, remove unneeded fields from the sub-table (i.e. fields corresponding corresponding to the Sample sub-table or Person sub-table).
    - *Output* step - output the result Literature sub-table as a worksheet named “Literature” in the “output” Excel file.
  - c) The third branch:
    - *Map Names* step, *Select TRUE* step & *Union* step - similar to the second branch, these

steps generate the “Person” sub-table and outputs it into the worksheet named “Person” in the “output” Excel file.

As the result of running the pipeline workflow file, one .xlsx file named “output” is generated which contains three worksheets corresponding to the three sub-tables, Sample table, Literature table, and Person table. The data is then sent to the data loading stage, onto the client's platform *Lithosurfer* as discussed in the next section.

## 6.3 Loading

### 6.3.1 Lithosurfer Introduction

Lithodat provides an intuitive Data as a Service (DaaS) online platform. LithoSurfer makes it quick and easy to visualize, analyze and extract geospatial data. With the ability to create complex data models, automated IGSN batch minting, and bespoke data extraction, structuring, and mining. LithoSurfer gives you easy access to complex datasets and data layers containing all detailed metadata. Users can upload, disseminate and publicize data within Lithosurfer. They can also purchase the license to use existing data from the company. If Lithodat needs to import new data, they can use the ETL workflow created to structure the customer's data and convert it into a format followed and recognized by the system.

Lithosurfer has three main tabs *Map*, *My Data* and *Documentation* (as seen in Figure 17). *Map* provides a visual interface to overlay a world map with varying geoscience data. This tab is accessible by all users. While, *My Data* and *Documentation* provide the data loading interface and a documentation about the API respectively. These two tabs are only accessible by users with advanced permissions or admins.

*My Data* provides three major functionalities for the data loading stage of ETL (tabs on the left side of Figure 18):

- a) Packages - Any data uploaded by an authorized user or admin is placed inside a data package. This is done to segregate the data imported based on the uploader. The package tab allows to create, delete or update a data package.
- b) Main Data - This tab provides a section to load the Sample data obtained after the data transformation step of the ETL pipeline.
- c) Related Data - This tab provides a section to load the People, Literature, Archives, Reference, Machine, and Funding data obtained after the data transformation step of the ETL pipeline. The IDs present in these entries are used to then match with the Sample data to maintain referential integrity.



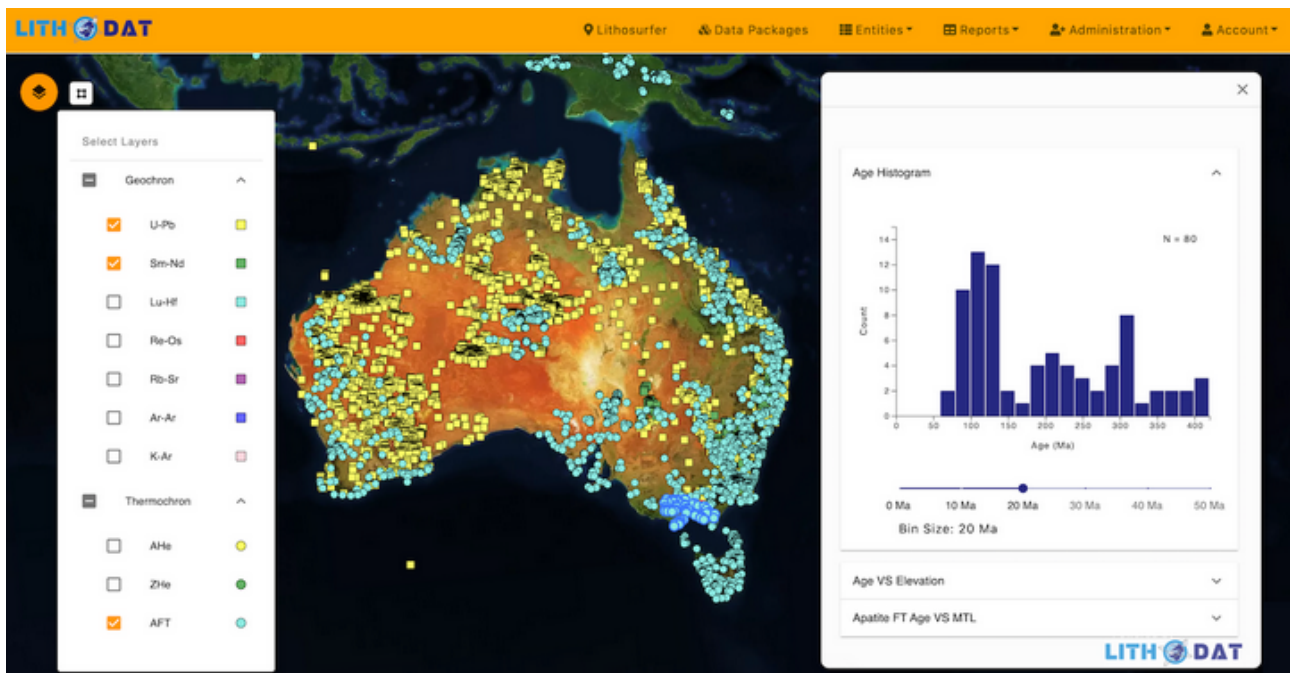


Figure 17: Lithosurfer

Note that the system currently only supports files in .CSV formats, and there is a limit of 50 rows per data file while performing data loading for Sample, People and Literature entities. Thus, the transformed data needs to be broken down into chunks of 50 rows per data file before performing data loading using a python script.

### 6.3.2 Loading into Lithosurfer

This stage of the ETL pipeline is not completely automated, and requires some manual inputs. The data obtained after the data transformation stage is then loaded into Lithosurfer via the following steps:

1. Create a data package (if not created), to store the data for a user belonging to particular data package.
2. Load the People, Literature, Archives, Reference, Machine, and Funding data into their respective section in the Related Data tab via the CSV importer.
3. Import the Sample data into its section in the Main Data section similarly.

After uploading the sample data, Lithosurfer can automatically detect the links between the rows in the Sample section and the rest of the data using the Primary-Foreign Keys relations present in the transformed data, and if any data is not correctly linked with the rest of the section's entries while importing the sample data, an error is shown for the respective entry which can be resolved by clicking on it and choosing an appropriate value matching with it and use its foreign key to link with its respective sub-table.

Map
My Data
Documentation
Account

Samples

Search

Package

Material

Tag

ADD FILTER + CREATE

| <input type="checkbox"/> | Sample Name | Package                     | Material  | IGSN      | Tags | Edited           |
|--------------------------|-------------|-----------------------------|-----------|-----------|------|------------------|
| <input type="checkbox"/> | 8110-137    | Australian Thermochronology | Granulite | IEMTG00JS |      | 2021-10-04 08:28 |
| <input type="checkbox"/> | 8110-125    | Australian Thermochronology | Granite   | IEMTG00JQ |      | 2021-10-04 08:28 |
| <input type="checkbox"/> | FT26        | Australian Thermochronology | Granite   | IEMTG01U6 |      | 2021-10-04 08:11 |
| <input type="checkbox"/> | WI-B        | Australian Thermochronology |           | IEMTG02DK |      | 2021-10-04 08:11 |
| <input type="checkbox"/> | WI-A        | Australian Thermochronology |           | IEMTG02DJ |      | 2021-10-04 08:11 |
| <input type="checkbox"/> | FT25        | Australian Thermochronology | Pegmatite | IEMTG01U5 |      | 2021-10-04 08:11 |
| <input type="checkbox"/> | 8110-118    | Australian Thermochronology | Granite   | IEMTG00JP |      | 2021-10-04 08:28 |
| <input type="checkbox"/> | 8110-132    | Australian Thermochronology | Gneiss    | IEMTG00JR |      | 2021-10-04 08:28 |
| <input type="checkbox"/> | 8110-108    | Australian Thermochronology | Granulite | IEMTG00JN |      | 2021-10-04 08:28 |
| <input type="checkbox"/> | 8110-113    | Australian Thermochronology | Gneiss    | IEMTG00JO |      | 2021-10-04 08:28 |

Rows per page: 10
1-10 of 10220
1
2
...
1022
NEXT

Drag and drop your csv file here

[Download sample template file](#)

[Help with sample data upload](#)

Figure 18: Data loading in lithosurfer

## 7 Conclusion and future work

ETL for geospatial data is on the rise to extract vital information about various geological points on earth from disparate sources with mismatching schema. There are a plethora of ETL tools available to be chosen based on the business logic and requirements i.e. code-based, software-based, or hybrid of both, and various testing techniques to evaluate them. Here, based on the clients' requirements we looked at various software-based ETL tools and tested them out on two datasets. Tableau-Prep was found to be the most suitable tool based on the requirement and passed all the tests performed. A final ETL pipeline was then created using this tool and is now implemented by the client in their workflow to structure their geospatial data and use it on their online platform *Lithosurfer*.

Although the test conducted was rigorous and followed the literature review, some limitations of the test include that they were only conducted on software-based tools due to client requirements as the pipeline had to be used by geoscientists at Lithodat. The tests were conducted on the ETL tools using the sample (banker) dataset and client dataset, these do not represent the complete range of datasets possible. Also, the trade-off between flow complexity and efficiency was considered i.e. although a more complex ETL flow might manage the foreign/surrogate keys relationships between the sub-tables, but are less efficient as it has greater execution time and higher storage demands. Thus, simpler and efficient ETL flows were preferred over complex.

The current pipeline generates the Sample, Literature, and Person sub-tables as its output. The final pipeline can be appended to extract Archives, Funding, and Machine sub-tables adding surrogate/foreign keys for respective sub-tables and checking for duplicates. A recommendation made to the client is to test the final pipeline on another geo-dataset. Finally, Geo-scientists can use the final ETL pipeline to structure their data and view or analyze it on the Lithosurfer platform.

## 8 Appendix

### 8.1 Team members and Roles

| Name                 | Contribution  |
|----------------------|---|
| Shiv Jitendra Mistry | Introduction, Related work, Testing Pentaho, Data exploration, Data dictionary, Data cleaning |
| Jiahui Zhu           | Methodology, Testing Tableau-Prep, Final ETL pipeline   |
| Hangzhou Wu          | Testing Tableau-Prep, Final ETL pipeline  |
| Kanav Sood           | Data architect, Data description, Testing Pentaho, Client communicator                        |

Table 5: Team members and roles

**System Repository:** <https://github.com/shivmistry605/MAST90106-Data-Science-Project-Group-3>

#### Created ETL Flows:

- **Pentaho:** <https://github.com/shivmistry605/MAST90106-Data-Science-Project-Group-3/tree/main/Pentaho>
- **Tableau-Prep:** <https://github.com/shivmistry605/MAST90106-Data-Science-Project-Group-3/tree/main/Tableau>

**Meeting Summaries:** [https://github.com/shivmistry605/MAST90106-Data-Science-Project-Group-3/tree/main/meeting\\_minutes](https://github.com/shivmistry605/MAST90106-Data-Science-Project-Group-3/tree/main/meeting_minutes)

## References

- [1] R. Katragadda, S. S. Tirumala, and D. Nandigam, "Etl tools for data warehousing: an empirical study of open source talend studio versus microsoft ssis," 2015.
- [2] S. H. A. El-Sappagh, A. M. A. Hendawi, and A. H. El Bastawissy, "A proposed model for data warehouse etl processes," *Journal of King Saud University-Computer and Information Sciences*, vol. 23, no. 2, pp. 91–104, 2011.
- [3] S. Vyas and P. Vaishnav, "A comparative study of various etl process and their testing techniques in data warehouse," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 753–763, 2017.
- [4] P. Woodall, T. Jess, M. Harrison, D. McFarlane, A. Shah, W. Krechel, and E. Nicks, "A framework for detecting unnecessary industrial data in etl processes," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 2014, pp. 472–476.
- [5] K. Patroumpas, M. Alexakis, G. Giannopoulos, and S. Athanasiou, "Triplegeo: an etl tool for transforming geospatial data into rdf triples." in *Edbt/Icdt Workshops*. Citeseer, 2014, pp. 275–278.
- [6] I. B. Sani and A. C. Aydinoglu, "Producing spatial etl tool for transforming between geo-data models," *International Multidisciplinary Scientific GeoConference: SGEM*, vol. 1, p. 867, 2013.
- [7] M. Zode, "The evolution of etl," *Retrieved on*, vol. 6, no. 06.
- [8] P. Amanpartap Singh, J. S. Khaira *et al.*, "A comparative review of extraction, transformation and loading tools," *Database Systems Journal BOARD*, vol. 42, 2013.
- [9] N. Biswas, A. Sarkar, and K. C. Mondal, "Empirical analysis of programmable etl tools," in *International Conference on Computational Intelligence, Communications, and Business Analytics*. Springer, 2018, pp. 267–277.
- [10] P. Vassiliadis and A. Simitsis, "Near real time etl," in *New trends in data warehousing and data analysis*. Springer, 2009, pp. 1–31.
- [11] —, "Near real time etl," in *New trends in data warehousing and data analysis*. Springer, 2009, pp. 1–31.
- [12] A. Wibowo, "Problems and available solutions on the stage of extract, transform, and loading in near real-time data warehousing (a literature study)," in *2015 international seminar on intelligent technology and its applications (ISITIA)*. IEEE, 2015, pp. 345–350.
- [13] P. Bindal and P. Khurana, "Etl life cycle," *International Journal of Computer Science and Information*

Technologies. Descargado de <http://bit.ly/2UXaXX6>, 2015.

- [14] A. Dearmer. (2021) Top 6 python etl tools for 2021. [Online]. Available: <https://www.xplenty.com/blog/comparison-of-the-top-python-etl-tools/>
- [15] T. L. Authors. (2011) Luigi. [Online]. Available: <https://luigi.readthedocs.io/en/stable/>
- [16] C. White. (2019) Why not airflow? [Online]. Available: <https://medium.com/the-prefect-blog/why-not-airflow-4cfa423299c4>
- [17] W. Christopher. (2017) Pentaho data integration. [Online]. Available: [https://help.hitachivantara.com/Documentation/Pentaho/7.1/0D0/Pentaho\\_Data\\_Integration](https://help.hitachivantara.com/Documentation/Pentaho/7.1/0D0/Pentaho_Data_Integration)
- [18] N. Mali and S. Bojewar, "A survey of etl tools," *International Journal of Computer Techniques*, vol. 2, no. 5, pp. 20–27, 2015.
- [19] Combine, shape, and clean your data for analysis with tableau prep. [Online]. Available: <https://www.tableau.com/products/prep>
- [20] R. Mukherjee and P. Kar, "A comparative review of data warehousing etl tools with new trends and industry insight," in *2017 IEEE 7th International Advance Computing Conference (IACC)*. IEEE, 2017, pp. 943–948.
- [21] D. Tobin. (2021) Understanding microsoft etl with azure data factory. [Online]. Available: <https://www.xplenty.com/blog/microsoft-etl-understanding-etl-with-azure-data-factory/>
- [22] I. Amazon Web Service. (2017) Aws glue: How it works. [Online]. Available: <https://docs.aws.amazon.com/glue/latest/dg/how-it-works.html>
- [23] M. Demba, "Algorithm for relational database normalization up to 3nf," *International Journal of Database Management Systems*, vol. 5, no. 3, p. 39, 2013.
- [24] G. McMurdo, "Database file normalization as an information science related activity," *Journal of Information Science*, vol. 4, no. 1, pp. 9–17, 1982.
- [25] A. Silberschatz, H. F. Korth, S. Sudarshan *et al.*, *Database system concepts*. McGraw-Hill New York, 1997, vol. 4.