

SMD Data Vault Project

Norah Alshahrani
nxa001@student.bham.ac.uk
2300301

Aadam Varsani
Aav118@student.bham.ac.uk
1877118

Arun Chumber
axc1220@student.bham.ac.uk
1802260

Cihang Yan
cxy108@student.bham.ac.uk
2357508

Salman Fatahillah
sxf181@student.bham.ac.uk
2318081

1. Abstract

This report documents the building of a Data Vault, built primarily to store multimodal data collected from a Neuro Imaging experiment. Data vault is a methodology used to build scalable data warehouses, allowing data analysis, and ensuring scalability and flexibility as the data stored in the data vault is raw and unprocessed. Using python programs, the data files were lightly processed to convert them into the .csv format to allow easy insertion into the data vault. The schema of tables was implemented using PostgreSQL, and the data was inserted using the module psycopg2 in python. The data vault that was produced as part of this project is functional, albeit basic, primarily lacking an easy-to-use GUI for the end-user to use to interact with the data vault.

2. Contributions of each Author

Norah Alshahrani:

- Participated in schema design
- Inserted data into tables
- Contributed to methods and results sections in this report
- Contributed to the Technical Documentation

Arun Chumber

- Participated in making the DataVault schema
- Helped to create the python script to insert data into database
- Contributed to the Methods, Results, Discussion and Conclusion sections of the Report
- Contributed to the Technical Documentation
- Helped created the SQL queries to query the database

Salman Fatahillah:

- Constructed the file combiner for fNIRS data
- Contributed to the Technical Documentation
- Contributed to the state-of-the-art section of the report
- Provided consult on some data insertion code
- Contributed to data processing section of methods

Aadam Varsani:

- Wrote the file combiner program for Dataset 1 VM
- Wrote the file converter for Dataset 2 EEG

- Contributed to the Abstract, Introduction, Discussion and Conclusion sections in this report
- Contributed to the Technical Documentation

Cihang Yan:

- Participated in schema design
- Inserted data into tables with python script
- Contributed to queries in the Result section
- Contributed to Methods, Results and Conclusion sections of the report
- Contributed to the Technical Documentation

3. Introduction

A data warehouse is a large collection of data that users or a business can retrieve to inform decision-making^[i]. These differ from databases; databases are optimized for read/write access to record and access data quickly. Data warehouses store large datasets from multiple databases with an additional layer that adds capabilities to aid in analyzing the data.

Data vaults are one method of implementing a large-scale data warehouse that builds upon other methods of implementing such warehouses, such as the 3rd normal form and dimensional design approaches. These changes generally improve flexibility and change over time, as raw data and metadata are stored separately and never lost when changes are made, and the system logs any changes.^[ii]

Data vault architecture generally has three layers. The staging area extracts data from the source as quickly as possible and generally does not store any historical data. The data warehouse layer is where all the data is stored for long term use, including all historical data. This data is raw and unmodified. The information delivery layer is between the end-user and prevents them from directly accessing the raw data in the staging area. This layer provides information to the end-user, meaning that the data is given context and processed.^[iii]

The data that this Data Vault was built to store had two main data sets. Dataset 1 is titled Visuomotor functional connectivity, whose primary purpose was to test and validate the processing and analysis algorithms for Functional Near Infrared Spectroscopy (fNIRS). The method behind collecting this data is that infrared light is shone upon the scalp. When leaving the scalp, the light contains information about the relative concentration of oxy- and deoxy-

haemoglobin.^[iv] For each exercise each patient participated in, the relative concentration oxy- and deoxy- haemoglobin was collected and stored in various files in the csv format.

Dataset 2, titled Multimodal pre-autism, had data collected from two modes, functional near-infrared spectroscopy (fNIRS) and electroencephalography (EEG). EEG measures the electrical activity of firing neurons within the brain^[iv]. The EEG data contains .mat files with two sessions for each patient, whereas the fNIRS data contains many different file formats for each patient and session.

4. State of the Art of Data Vaults for Medical Imaging

The efficient management and examination of large-scale scientific file repositories has become critical to scientific progress^[v]. Ivanova, M et al. (2013) implements the data vault which allows a user to attach an external file repository to a main database management system (DBMS) and let it perform efficient and flexible query processing over the data. Since the data vault only reads through the data, the original data is kept in its original format, preserving the ability to use current external tools if necessary. The data vault architecture proposed in this paper adds three new components to the MonetDB software architecture: a wrapper, an optimizer, and a cache manager. The Data Vault wrapper is in charge of communicating with the file repository and is based on data model mapping, as well as file format conversions to database schema conversions.

Unifying data found in medical domains should be done to deal with the “heterogeneity and diversity” of structures and sources of data. This will help with the machine learning and artificial intelligence (AI) algorithms which can detect diseases, segment images, assess organ functions, and other research tasks, close to human-like level of performance^[vi]. Vázquez-Ingelmo, A et.al (2020) proposes a platform architecture that can accept these diverse data through an architecture of data management that supports the “plain” upload of data, which also can be analyzed, visualized, and processed through integrated tools. HTML, CSS, and JavaScript are used at the front end. The platform's core features are supported by the back end, which includes data storage, data processing, and an AI environment. Django, a Python-based web framework, is used to create the web application. The platform's entry point is the web application, which provides users with a variety of services. To manage the various data structures that will be part of the projects, this web application is connected to other technologies (databases, file systems).

The implementation of data warehouses in medical imaging is an increasingly important and powerful tool that is used to diagnose many kinds of diseases. Neurovault is an open-source repository of statistical maps of the human brain^[vii]. Although it does not implement the data vault method, it still gives insight on how a data warehouse is implemented to visualize medical images. It uses a web interface, not needing for any software to be downloaded or installed, and users may upload their own research data to the repository. The fact that an abundance of data is available for other researchers and doctors to access, thousands of dollars and effort is saved.

Neurovault also tackles other problems in the field such as accepting unthresholded statistical maps to avoid bias when discarding information below a threshold, thus “skewing perception of accumulated knowledge”. Another problem is the difficulty of placing a researcher's findings in the perspective of others' findings. Because of the large number of brain imaging studies published each year, manual comparison is both impossible and biased.

5. Methods

5.1 Data Vault Schema

The data vault was designed specifically to store and manage data from the two medical studies. It is a relational database, and its Entity-Relationship (ER) Model is shown in Figure 5.1.1 in the next page.

A data vault consisted of three types of tables: Hubs, which contained a unique hub key for each entity; Links, which determine the relationship between related hubs; and Satellites, which store descriptive information in the attributes.^[iii]

5.1.1 Schema Overview

5.1.1.1 Hubs

Based on the structure and features of the original datasets, hubs were created: Patient, Session, Study, Factor, Treatment, EEG, DataSource, Analysing. These hubs were the aspects of the data that a use contained the keys and IDs of the objects in each category.

Data Vault entities such as links and satellites reference objects using keys and IDs in hubs.

Hub_***_Key was the primary key of the Hub, which would be a foreign key in the Link. The reference key in the Hub was ***_ID, which was the foreign key in the Satellite.

5.1.1.2 Satellites

As part of the satellite architecture, the descriptive information was stored in attributes that belong to objects that are part of the hub. The satellites were divided up to make the updating process more efficient and easier.

In a Satellite table, there was a foreign key whose reference is the hub's reference key.

When one attribute of the object changes considerably, but other attributes remain the same, all the attributes should not be kept in the same satellite. Therefore, one hub might connect to multiple satellites. For example, the hub patient has several satellites for patient name, sex and age.

There were two rules to distribute attributes into different satellites, one based on data type and the other based on rate of change. For instance, Hub DataSource stored massive experimental data. It is linked to Hub Analysing, and analysis information was divided into three satellites based on Data Type (measurement units). DataSourceChange and

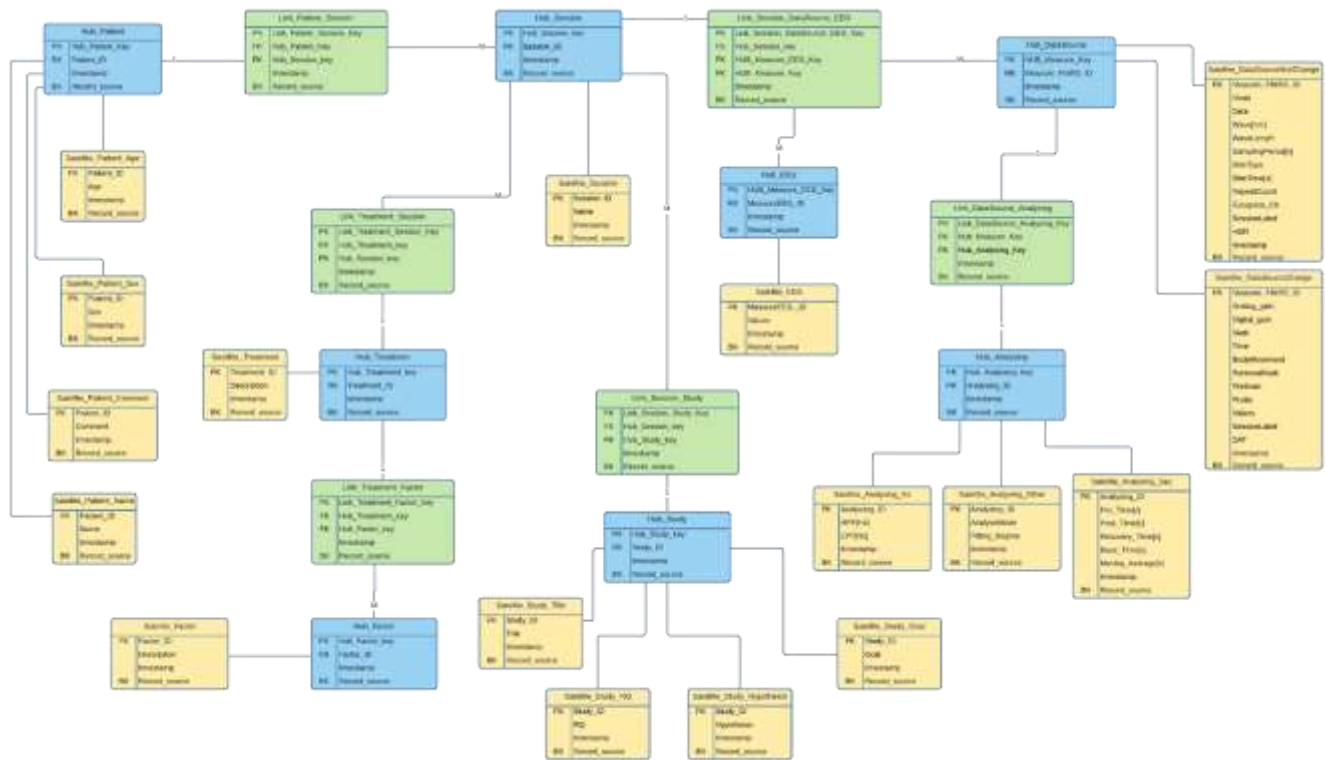


Figure 5.1.1 The schema of the proposed ER Model

DataSourceNotChange were two satellites of Hub DataSource that were split by Rate of Change.

5.1.1.3 Links

In addition to connecting hubs, Links indicate the relationship between objects within each hub. In the Link table the primary key was a unique Link ID that identified a link relationship between two or more Hub Keys. The Hub Keys are considered foreign keys.

5.1.1.4 Business Key

The Record Source is the business key for all the Hub/Satellite/ Link tables, which gives information about the record's insertion.

This addition makes it possible to see historical updates and versioning of the tables. In the satellite, a record is unique by pairing both the reference key and business key. For instance, a patient's comment could have several versions during the study, hence it is necessary to allow inserting multiple records under one patient ID (reference key) in the table. When updating the information, instead of deleting the previous record with a "version 1" business key, a new row could be inserted with a new "version 2" business key.

5.1.2 Schema Decision

5.1.2.1 Patient Hub

Hub_Patient has an identifier as Hub_Patient_Key. Each attribute is stored in one satellite. There were 4 different satellites connecting to Patient Hub: Patient_Age, Patient_Sex, Patient_Comment, and Patient_Name. The satellites were divided up to make the updating process more efficient. This design could make it possible to store the information of other studies in this data vault, because the

attributes of personal information are universal for most of the research study objects.

5.1.2.2 Analysing Hub

Hub_Analysing has an identifier as Hub_Analysing_Key. It stores the Analysis information from the VM dataset. This hub has three satellites. They are divided by the Data Type (measurement units), and are as follows: Analysing_Hz, Analysing_Sec and Analysing_Other. The three satellites ensured same type of data stay together, which provides convenience for the end user.

5.1.2.3 Session Hub

Hub_Session has an identifier as Hub_Session_Key. This hub has only one satellite - Satellite_Session, and it stores the name/description of the session.

5.1.2.4 Study Hub

Hub_Study stored the descriptive information of each study and is identified by its Hub_Study_Key. Each satellite stores one attribute of each study. There are 4 satellites in total: Title, Goal, Hypothesis and RQ (Research Question). This satellite design makes it possible to store information from other studies because the attributes are universal.

5.1.2.5 Treatment Hub

Hub_Treatment has an identifier as Hub_Treatment_Key, it identifies the treatments in the studies. The descriptions of the treatments are stored in Satellite_Treatment. Again, as these attributes are universal, this design can be used to store other study information.

5.1.2.6 Factor Hub

Hub_Factor has an identifier as Hub_Factor_Key, it identifies the factors in the studies. The descriptions of the factors are stored in Satellite_Factor. Again, as these attributes are universal, this design can be used to store other study information.

5.1.2.7 EEG Hub

Hub_EEG has the identifier as Hub_EEG_Key, and it identifies each data point from dataset 2. Satellite_EEG is used to store attributes from the data. In the satellite, the arrays of data are stored in the 'Values' attribute. The data was stored as an array to make the data retrieval process easier. The decision was made to store the EEG data in a separate hub as it contains a different data type to the rest of the data in both datasets.

5.1.2.8 DataSource Hub

Hub_Datasource has the identifier as Hub_DataSource_Key to store the arrays of fNIRS data from both dataset 1 and dataset 2. There was no need to create another hub because the data types are the same for both sets of data. The attributes of each fNIRS experiment were split by whether the data changes from session to session and thus the need for the two satellites, DataSourceChange and DataSourceNotChange.

5.1.2.9 Relationship sets [cardinality ratios] between two or more hubs

- Each Treatment is allotted with many Factors. [Look from Hub_Treatment to Hub_factor]
- Each Patient is allotted with many Sessions. [Look from Hub_Patient to Hub_Session]
- Each Treatment is allotted with many Sessions. [Look from Hub_Treatment to Hub_Session]
- Each Study is allotted with many Sessions. [Look from Hub_Study to Hub_Session]
- Every experiment's metadata (Analysing information) is allotted with exactly one outcome/result (DataSource). [Look from Hub_Analysing to Hub_DataSource]
- Each DataSource is allotted with exactly one Session and Each EEG is allotted with exactly one Session. [Look from Hub_DataSource and Hub_EEG to Hub_Sesstion]

5.2 Data processing for insertion - Staging Layer

Raw data was pre-processed for easier insertion before creating the database. In dataset 1, for each of the 16 different file names the raw data under the banner "Data" were combined. So, one of these combined files contained every patient's data.

For data set 2, the EEG data was converted from the raw .mat files into .csv by using the mat73 and numpy packages. Similarly, the fNIRS data files from dataset 2 (.w11, .w12, .dat, .hdr, and .inf) were also converted and then later combined .csv files. This pre-processing was done to allow for easy parsing and insertion.

The structure of VMData from dataset 1 and EEG data from dataset 2 was not altered. However, the fNIRS data from

dataset 2 was slightly modified. The .w11, .w12, and .dat files were each transposed (change the rows of data into columns and vice versa). The .w11 and .w12 files consistently had 96 columns but each file had a different amount of rows. Similarly, .dat files had a consistent number of rows (48) and a different amount of columns. By transposing the data, appending onto the .csv file, insertion into the database was much easier to implement since the rows are divided evenly.

Figure 5.2.1 shows an example of a combined .csv file for the VM data, showing 28510 rows. Figure 5.2.2 is the result of the conversion of the EEG data from .mat to .csv and shows that there are a total of 86 files combined with a size of 5.50 GB. Figure 5.2.3 is the result of the combination of the first session of fNIRS .w11 data files converted to .csv format, showing that the file accumulates up to 4128 (43 files x96) columns transposed into rows. Figure 5.2.4 accumulates up to 2064 (43 files x48) columns transposed into rows. Figure 5.2.5 shows the metadata collected from the first session .inf files. Figure 5.2.6 shows the metadata collected from the second session .hdr files.

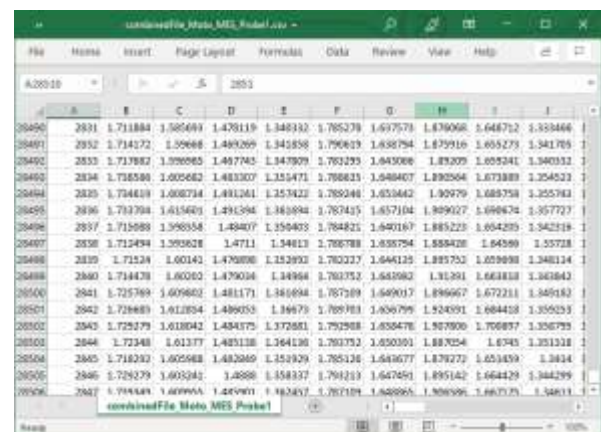


Figure 5.2.1 An example of a combined file from VM data

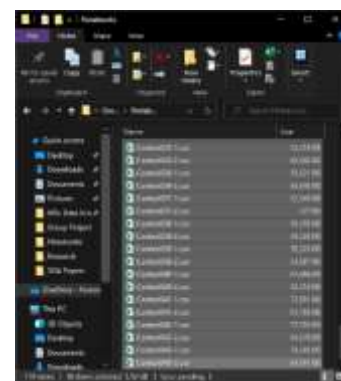


Figure 5.2.2 Conversion of EEG data from .mat to .csv

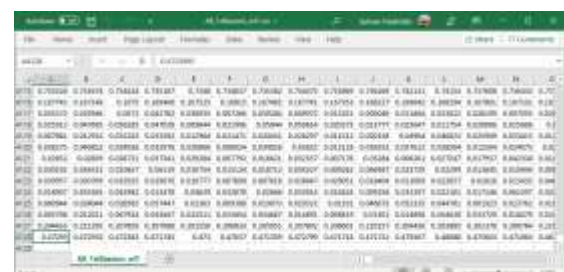


Figure 5.2.3 The first session .w11 data

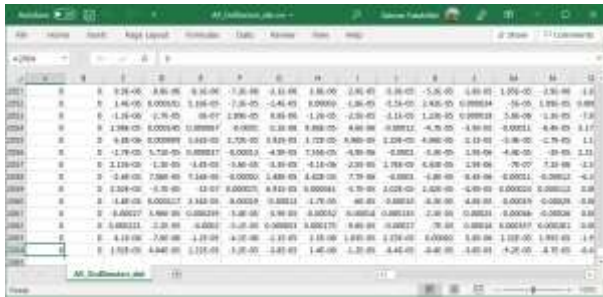


Figure 5.2.4 The second session .dat data

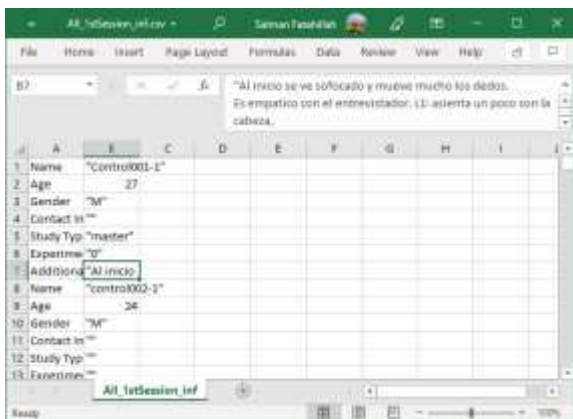


Figure 5.2.5 Metadata collected from the first session .inf files

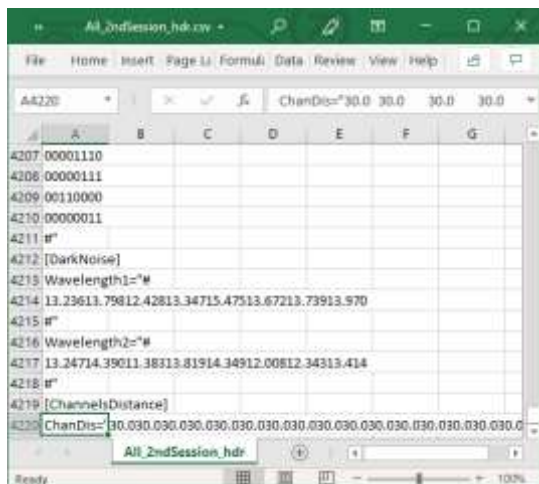


Figure 5.2.6 Metadata collected from the second session .hdr files

5.3 Data Vault Population – Enterprise Data Warehouse

The data was inserted into the tables in two ways. The first method was to manually write each row of data. The second method was to use a combination of the python packages ‘pandas’, ‘numpy’ and ‘csv’ to parse through each of the pre-processed data files and extract information (please see the technical documentation for more details).

Method 1 was mainly used for small hubs/satellites where there would only be a few rows of data. For the larger hubs/satellites, method 2 was used in combination with many for-loops to insert thousands of rows worth of data.

This process, however, is time consuming as it takes roughly 20 minutes to fully populate.

The main package used to insert data is “psycopg2”.

This library allows python to connect to Postgres and it was extensively used to insert and query data from python to the data warehouse in PostgreSQL.

The data was taken from arrays created from parsing through the pre-processed csv files. Using a for loop, each element from each array was entered into a SQL statement and this statement was then executed. The SQL statement was then updated with new data and again this was executed. This carried on until all data was entered into the database.

The subsections 5.3.1, 5.3.2 and 5.3.3 show examples of how this process works throughout the entire insertion process.

5.3.1 EEG Satellite Insertion – Method 2

The .mat files of EEG data was first converted to .csv during the pre-processing stage. The data in EEG file is a 2-dimensional numeric array in which rows are channels and columns are samples. The .csv was read into a python list and inserted into the table in the attribute “Values”. With the glob function, each file in the fold is read and inserted one by one. Figure 5.3.1 shows the python code of reading and inserting of the EEG data into the satellites, and figure 5.3.2 shows the EEG data in list format.



Figure 5.3.1 Python code of reading and inserting the EEG



Figure 5.3.2 The EEG data in the list format

5.3.2 Patient Satellite Insertion – Method 2

The metadata containing the patients’ ID, name, sex, age, and comments are sorted into a combined .csv file, made during the pre-processing stage. The .csv is read into a pandas DataFrame and converted to a numpy array. Since the information is arranged in a certain order, the rows are read where the required value is located. For the satellite “Satellite_Patient_Sex”, since there is a enumerate type attributes, the original value as “F” and “M” would be identified by the program, and the program would insert values “Female” and “Male” into the table. Figure 5.3.3 shows the python code of reading and inserting the satellite table, “Satellite_Patient_Sex”.



Figure 5.3.3 python code of reading and inserting the table satellite_patient_sex

5.3.3 Analysing Hub Insertion – Method 1

For this hub, a list was created containing the range of numbers from 1-333. The range 1-160 represents dataset 1 and the range from 161-332 representing dataset 2. These ranges represent the ID's and the keys. Then by use of a for-loop this was inserted into an SQL query which was then executed.



Figure 5.3.4 python code of reading and inserting the table Hub_Analysing

5.4 Data Vault Querying - Information Delivery Layer

Queries were written for the information delivery layer. In this report, 4 queries were made to demonstrate the data retrieving abilities from the data vault. The first two queries return information regarding the same hub and the other two are multi-level nested queries retrieving data across multiple tables.

The command was executed in python, and the retrieved data would be in a tuple format which was then converted into a pandas dataframe.

6. Results

The following features were tested and proved to be valid:

6.1 Intermediate files

Most of the experimental data in the given both datasets has been sorted into a processable format for storage in the data vault.

This includes: (1) VM data of 160 .csv files, each file containing Patient Information, Analysing Information, Measure Information, and experimental results data. (2) EEG data of 86 .mat file, each file containing a numeric array of experimental results data. (3) fNIRS data of 86 session folders, each folder containing files of several formats and the content of .wl1, .wl2, .dat, .hdr, and .inf files is stored.

Figures 6.1.1 to 6.1.6 are screenshots showing a successful query of these different data types:

Measure FNIRS ID	Session Label	Values
0	1 Probe1(Decoxy)	[[0.1523127, 0.1531544, 0.1564775, 0.1574434, ...
1	5 Probe1(Decoxy)	[[0.3503567, 0.3546688, 0.3582221, 0.358452, 0...
2	9 Probe1(Decoxy)	[[0.2309925, 0.2322225, 0.2382927, 0.2347602, ...
3	13 Probe1(Decoxy)	[[0.610120, 0.5537068, 0.5781247, 0.6085341, 0...
4	17 Probe1(Decoxy)	[[0.1712919, 0.1714161, 0.1727203, 0.1723045, ...

Figure 6.1.1 VMData CSV Files

Measure EEG ID	Values
0	1 [[0.0000000000000000e+00, 3.906250000000000...

Figure 6.1.2 EEG Files

Measure FNIRS ID	Contents of HDR File
0	161 [[[GeneralInfo]], [FileName="NIRS-2019-10-10_0...
1	162 [[[GeneralInfo]], [FileName="NIRS-2019-10-16_0...
2	163 [[[GeneralInfo]], [FileName="NIRS-2019-10-17_0...
3	164 [[[GeneralInfo]], [FileName="NIRS-2019-10-17_0...
4	165 [[[GeneralInfo]], [FileName="NIRS-2019-10-17_0...

Figure 6.1.3 .hdr Files

Measure FNIRS ID	Contents of DAT File
0	161 [[0.0, 0.0, 2.05e-05, 8.19e-05, -1.04e-05, -1...
1	162 [[0.0, 0.0, -7.81e-05, 6.34e-05, -0.0001467, ...
2	163 [[0.0, 0.0, 5.2e-06, -0.0001064, 2e-07, 8.83e...
3	164 [[0.0, 0.0, -0.0001248, 0.0001151, 0.0001322, ...
4	165 [[0.0, 0.0, 6.36e-05, 8.59e-05, -3.3e-06, -9.6...

Figure 6.1.4 .dat Files

Patient ID	Names from INF File
0	control001 Control001
1	control002 control002
2	control003 control003
3	control004 control004
4	control005 control005

Figure 6.1.5 .inf Files

Measure FNIRS ID	Session Label	Contents of WL1 & WL2
0	217 Session1_wl2	[[0.7007903, 0.7640430, 0.7713492, 0.7703189, ...
1	218 Session1_wl2	[[0.1971750, 0.2042922, 0.1958234, 0.2018211, ...
2	219 Session1_wl2	[[0.6766214, 0.6598759, 0.6606942, 0.677299, 0...
3	220 Session1_wl2	[[0.2253933, 0.227439, 0.2281479, 0.2252912, 0...
4	221 Session1_wl2	[[0.2780361, 0.2729575, 0.2720815, 0.2737829, ...

Figure 6.1.6 .wl1 and .wl2 Files

For further information, please see the Group_4_DataTypes.ipynb file in the technical documentation.

6.2 Tables of the Schema

All the tables in the schema were implemented and populated, and the screenshots of the tables were provided in supplement material Result folder.

6.3 Query traversing Hubs and Satellites

The data warehouse can display data from multiple satellites all from the same hub.

Figure 6.3.1 shows a query which will display the patient's age, sex and comment when given the Hub_Patient_Key.

Patient ID	Age	Sex	Comment	
0	control013	24	Male	"Actitud tranquila y neutra. Contestaciones co...

Figure 6.3.1 Query of patient's age, sex and comment with given hub patient key

Similarly, figure 6.3.2 shows a query which when given a Session Label will display the corresponding Measure fNIRS ID, Session Label and Value.

Measure	fNIRS ID	Session Label	Values
0	2	Probe1(Deoxy)	[[0.4625012, 0.4429649, 0.4377473, 0.4207151, ...
1	6	Probe1(Deoxy)	[[0.1246009, 0.1228095, 0.1230936, 0.1258359, ...
2	10	Probe1(Deoxy)	[[0.6436276, 0.6435532, 0.6412718, 0.6459306, ...
3	14	Probe1(Deoxy)	[[0.6996979, 0.6973091, 0.690679, 0.6968778, 0...
4	18	Probe1(Deoxy)	[[0.2658096, 0.2632664, 0.2625043, 0.2652858, ...

Figure 6.3.2 Query of retrieved Measure_fNIRS_ID with given session label

6.4 Query traversing Hubs, Satellites and Links

The data warehouse can be traversed to look for specific information. The value in the satellite could be queried with given condition of another satellite providing there is a relationship of [Satellite_A – Hub_A – Link_A_B – Hub B – Satellite_B].

For example, factor description of one treatment could be queried with given treatment ID, shown in figure 6.4.1.

Factor ID	Factor Description	Treatment ID	Treatment Description	
0	5	Without syllabic stress	5	Normal conversation

Figure 6.4.1 query of factor description with given treatment ID

Another example is of a query selecting some metadata from the Hub Analysing based on the Session Label "Probe1(Deoxy)" from the Hub DataSource, shown in figure 6.4.2.

Session Label	PreTime[s]	PostTime[s]	RecoveryTime[s]	BaseTime[s]	MovingAverage[s]
0 Probe 1(Deoxy)	10.0	10.0	10.0	25.0	5.0
1 Probe 1(Deoxy)	10.0	10.0	10.0	25.0	5.0
2 Probe 1(Deoxy)	10.0	10.0	10.0	25.0	5.0
3 Probe 1(Deoxy)	10.0	10.0	10.0	25.0	5.0
4 Probe 1(Deoxy)	10.0	10.0	10.0	25.0	5.0

Figure 6.4.2 Query of retrieved metadata with given session label

7. Discussion

One of the main requirements for a data vault is scalability and flexibility. However, for the file processing programs for Dataset 1 and Dataset 2 EEG, the programs are rather inflexible and tailored specifically for this given data set. These programs work well with the supplied files and their given naming idiosyncrasies but would give the user many errors if they attempted to use the program on a different set of files, even if they were still trying to merge files, in the case of dataset 1, or converting the files from .mat to .csv in

the case of dataset 2 EEG. These programs are tailored to these files such that they would need to virtually need to be rewritten entirely for any other set of files. This is not scalable or flexible for any other experiment.

The insertion code itself is not scalable or flexible, as there would be too many variable names and loops to keep track of. There's also the issue with the method of entering the many file paths for each file, and at a large scale this process is not feasible. Additionally, the time in which it takes to parse through each file, extract relevant information and then insert into the database is too time consuming.

Unfortunately, some of the Link tables do not work as required, and the information mart layer is directly affected by this. As a result, only some basic querying can be performed. Querying the whole data warehouse by traversing tables will not work. However, out of the few Link tables working, performing more complex and non-trivial queries can be done (see Results section). If all Link tables worked as desired, a wider variety of queries could have been made and visualizations of the data could have been shown in more relevant ways.

Another limitation is that the queries are static. The consequence of this is a user cannot specify what data they would like to view, and the only way to view it is to change the static query. This is problematic as a user might accidentally change the wrong part of the query and now the output will not work, or it will display an incorrect result.

8. Conclusion

Data vaults are required in a wide variety of industries to store historical data that will be required to conduct analysis and drive decision-making. The data vault produced for this project was optimised for data collected as part of a Neuroimaging experiment but had been designed with the aim of facilitating sufficient flexibility that it could be used to store data from other experiments.

This project was ultimately successful in that a data vault was built that was capable of storing data from multimodal brain scan data, in addition to the extensive metadata associated with the experiments and the modes of scanning. Historical updates and versioning were made possible by using "record_source" as a business key. The data warehouse layer was fully implemented using the data vault model with basic staging and information mart layers. In addition, rudimentary data analytics is possible by querying the data stored.

While successful in achieving the broad aims of the project, there are areas for improvement if it were to be undertaken again, given greater time and resources. The largest source of improvement would be the current lack of a Graphical User Interface (GUI). Django would be used to implement this to create a web-based application that facilitated the GUI. This would also support user profiling and authentication.

Additionally, a more comprehensive data mart would have been produced to support more complex querying of the data.

Further, visualisation of the data would ideally have been supported in this implementation.

While there are aspects that leave room for improvement, the project is successful in that it facilitated the contributors learning about Data warehouse architecture and the data vault methodology and the associated three-layer architecture and Hub-Link-Satellite structure. Additionally, processing raw data in different formats and using psycopg2 to populate the database, all in python, and designing the data vault schema have strengthened the contributors' knowledge and expertise in these areas.

9. References

[i] *talend. (n.d.). What is a Data Warehouse and Why Does It Matter To Your Business? Retrieved December 4, 2021, from <https://www.talend.com/resources/what-is-data-warehouse/#:~:text=A%20data%20warehouse%20is%20a,systems%20that%20reveal%20business%20intelligence>*

[ii] Ryan, J. (2021, July 12). When should I use Data Vault? <https://www.analytics.today/blog/when-should-i-use-data-vault> Retrieved December 4, 2021

[iii] Linstedt, D., & Olschimke, M. (2015). *Building a Scalable Data Warehouse with Data Vault 2.0 (1st ed.)*. Morgan Kaufmann Publishers Inc. <https://dl.acm.org/doi/10.5555/2901170>

[iv] Orihuela-Espina, F. (2021). *Storing and Managing Data Module 06-32245 Course project*.

[v] Ivanova, M., Kargin, Y., Kersten, M., Manegold, S., Zhang, Y., Datcu, M., & Molina, D. E. (2013). Data Vaults: A Database Welcome to Scientific File Repositories. *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. <https://doi.org/10.1145/2484838.2484876>

[vi] Vázquez-Ingelmo, A., Sampedro-Gómez, J., Sánchez-Puente, A., Vicente-Palacios, V., García-Peñalvo, P. I., Sanchez, P. L., & García-Peñalvo, F. J. (2020). A Platform for Management and Visualization of Medical Data and Medical Imaging. *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 518–522. <https://doi.org/10.1145/3434780.3436652>

[vii] Gorgolewski, K. J., Varoquaux, G., Rivera, G., Schwarz, Y., Ghosh, S. S., Maumet, C., Sochat, V. v, Nichols, T. E., Poldrack, R. A., Poline, J.-B., Yarkoni, T., & Margulies, D. S. (2015). NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. *Frontiers in Neuroinformatics*, 9. <https://www.frontiersin.org/article/10.3389/fninf.2015.00008>