Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

Processing FHIR in modern Data Lakehouse

Author: Shreyas Patil (VU ID: 2759591)

1st supervisor:	A.S.Z. (Adam) Belloum
daily supervisor:	Marcel Hofman (LOGEX)
2nd reader:	Dr Ana Ana Oprescu

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

July 4, 2024

"I am the master of my fate, I am the captain of my soul"
from Invictus*, by William Ernest Henley*

Abstract

Healthcare data exchange standard, FHIR is becoming the widely adopted standard for data exchange and storage of health data. However, there is still some learning curve involved when it comes to using the data for analytical purpose. In our research we have devised a generic approach to ingest FHIR data into tabular format that is suited for analytical purpose. Our approach reduces the learning barrier involved in processing FHIR and formatting it for secondary usage.

Contents

List of Figures **ii**

List of Tables **iii**

1 Introduction **1**

1.1 Research Question(s) 2

2 Background **4**

2.1 Clinical Registry and FHIR in the Netherlands 6

2.2 Data lakes in Healthcare 6

3 Related Work **8**

4 Design and Implementation **11**

4.1 Synthetic Data 11

4.2 Infrastructure Setup 12

4.3 Implementation 13

5 Evaluation **15**

6 Discussion **19**

7 Limitations **21**

8 Conclusion **22**

References **23**

List of Figures

1.1 Stages of HoF initiative by HL7 Europe (Henrique Martins et al 2022)(2) . 2

2.1 Combinatorial increase in number of different interactions and interfaces required to exchange data between different systems due to lack of single widely used standards (Benson, T., & Grieve, G., 2016)(5) 4
2.2 Contributors of data and applications of data lakes in healthcare (Gentner et al. (2023) (8) 7

4.1 Architecture of Synthea (Walonoski et al. 2018) 12
4.2 Mechanism involved in processing FHIR 13

5.1 Sample Flattening of Patient Resource 15
5.2 Multiplication of rows from single FHIR resources 16
5.3 Distribution of resources generated by Synthea 17
5.4 Proportion of all resources vs Observation resources 17
5.5 Computation time to flattening of resources on a 3-Node Spark Cluster(12 GB, 5 cores) 18
5.6 Storage of Observation resources after processing in Parquet format vs all resources in the input JSON format 18

List of Tables

1

Introduction

Analytics of healthcare data has been challenging due to the lack of interoperability of data received from different healthcare IT sources which are built using legacy standards and software. However, with the introductionand increasing usage of modern data format FHIR (Overview - FHIR v5.0.0 (hl7.org)), the possibility of aggregating data from multiple sources for the purpose of analytics has increased a lot. FHIR is a term that refers to the API specification, the data format specification wherein each data point is represented as a JSON/XML object of entities representing patient, their interaction with healthcare, including financial aspects such as billing, as well clinical aspects such as patient conditions and treatments received and so on (Lehne et ., 2019)(1).

With more and more EMR vendors (Electronic medical records, health IT systems for capturing and storing) leaning towards the usage of standard format FHIR, it still being in its nascent stages poises the question how it can be modelled into data warehouses for analytics.

To accelerate the adoption of FHIR in the context of European Healthcare, HL7 Europe has started an initiative Hospitals-on-FHIR(HoF) (Henrique Martins et al 2022)(2). It is based on maturity model that enables the hospitals to assess their status as to on what level they are on with respect to usage of FHIR. FHIR is a relatively new concept to many healthcare organisations and it fosters the collaboration of knowledge between hospitals and relevant stakeholders. Broadly the different levels as seen Figure 1.1 in can be categorised into 4 distinct phases to describe the usage of FHIR by a healthcare org. The first phase is Aspiring which implies that the healthcare organisation has expressed its interest in participating in the adoption of FHIR. Second phase is Preparing which means that the hospital has made the necessary assessment and requirement specification to start with implementation. Third phase is Using which means that there is a live FHIR project on

going and it has formalised on its capabilities and offerings which are accessible to other entity that is going to interact with it. Fourth stage is Collaborating which implies that the organisation has started sharing of FHIR data with external entities within country.

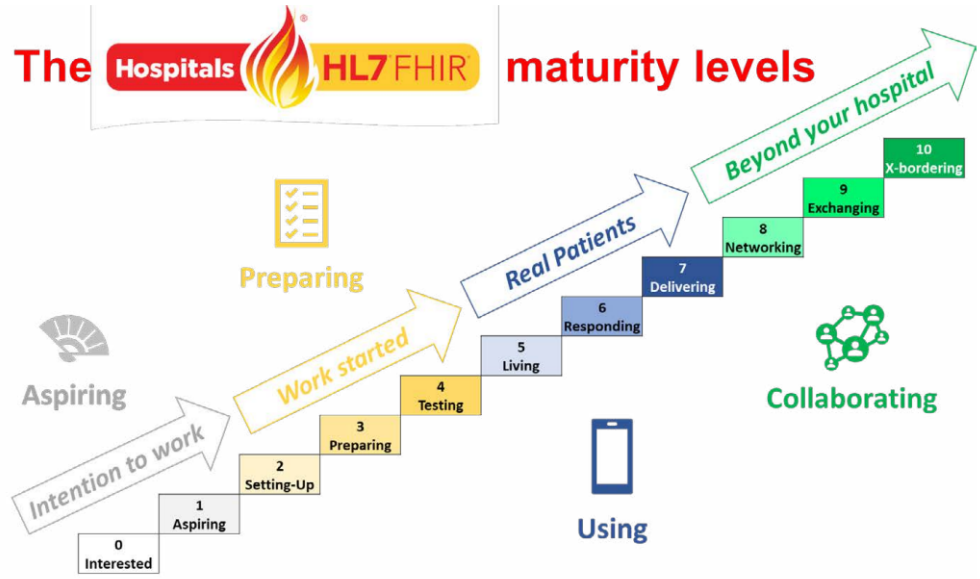

Figure 1.1: Stages of HoF initiative by HL7 Europe (Henrique Martins et al 2022)(2)

The aim of our research is to devise a generic approach that is capable of processing any FHIR resource with minimal knowledge of FHIR specification compared to existing approaches which we discuss further in Section 3 With emphasis on scalability of processing using modern datalakehouse architecture using technologies such as PySpark [1] for distributed processing and Iceberg table format [2] for facilitating better management of data to be analysed. Below we discuss the specific research questions that we aim to address in this thesis.

1.1 Research Question(s)

RQ1 : How can FHIR data be effectively processed into a tabular structure in a datalakehouse ?

Existing approaches to process FHIR data for analytics as discussed further in this document under Section 3 rely on usage of FHIRpath [3] which is a query language to access

[1] https://spark.apache.org/docs/latest/api/python/index.html
[2] https://iceberg.apache.org/
[3] http://hl7.org/fhirpath/N1/

and process different elements of FHIR data. This introduces learning curve involved in being able to process FHIR data into tabular structure that can be further utilised for analytics. A more generic approach to process is required and is the main focus of research here.

RQ2 : How does usage of processing engine Spark perform in terms of processing speed and data format Iceberg perform in terms of storage ?

With data table format Iceberg gaining popularity purpose of our research is to perform benchmarking on how it affects performance in terms of storage of data. Iceberg is a table format that provides the combined features of data management using data warehouse and data lakes.

2

Background

Healthcare standards have evolved over years and HL7 is an organisation that is responsible for the development and usage of standards in industry. Efforts to achieve data interoperability has not been successful due to the presence of different standards and their usage has been customized to suit different cases local to the region (Bender, D., & Sartipi, K. , 2013)(3). HL7 is an organisation that is responsible for development of healthcare data exchange standards in 1987. It creates standards for messaging between hospital information systems(HIS) and the format of electronic documents. HL7 v2 and v3 are the popular and widely used messaging standards. HL7 v2 was started in 1987, it was a messaging protocol at the application level in the OSI networking model (4) like HTTP. Although it had been widely used since then, it has drawbacks which limited its use in a consistent manner by different developers of HIS. It was designed and utilised in an adhoc manner to allow IT systems of different departments such as clinical, billing and laboratory to exchange data and is widely supported by HIS in North America. Due to lack of consistency in the standard's usage, the standard could not scale well to multiple systems in healthcare institutions. (Bender, D., & Sartipi, K. , 2013)(3).

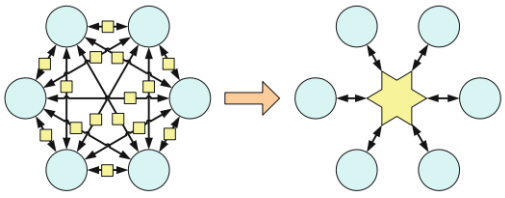

Figure 2.1: Combinatorial increase in number of different interactions and interfaces required to exchange data between different systems due to lack of single widely used standards (Benson, T., & Grieve, G., 2016)(5)

To overcome the drawback of HL7 v2 protocol, a new version of HL7 version 3 (v3) was

started in 1995. It was based on principle of design by constraint where in large complex health data models were created as specifications and the developers had to constrain them to implement their requirements. However, this approach is very expensive to implement and there was no guarantee that different implementations would be interoperable in data exchange. HL7 v3 was adopted by governments in Canada, United Kingdom, Denmark, Australia, Germany but the projects using it has failed (Bender, D., & Sartipi, K. , 2013)(3).

The major challenge has been the lack of consensus around what data should be exchanged between to HIS. The overall problem is due to the lack of understanding about which parts of data exchange are standardised and which are left to the choice of implementers. With Service Oriented Architectures, protocols such as SOAP, REST were gaining popularity in 2000s in other industries, and the idea was to utilise them for creating healthcare data exchange interfaces. This led to creation another standard HL7 FHIR(Fast Healthcare Interoperability Resources) in 2011 (Grahame Grieve et ., 2016)(5), wherein different data elements of a healthcare process are represented as resources as JSON or XML and the servers containing them can be interacted with by means of REST APIs (Braunstein., 2018)(6). The specification is not final and is improved in an incremental manner in an agile manner wherein experts update the specification based on experiences and grievances of the implementers using it for their projects.

Clinical and observational data in healthcare is represented using code which are provided by different coding systems such as SNOMED CT [1], LOINC [2]). Usage of codes is a challenging task as different codes could be possibly used by different implementers of EMRs differently and being able to consistently use the same code and interpret it correctly is the crucial aspect of data exchange necessary of semantic interoperability. While FHIR provides a standard structure for different data points it only guarantees structural interoperability. Semantic interoperability is the ability to correctly interpret the data that is being sent from another system, both the sender and receiver need to have the same interpretation of medical concept that is being exchanged and the consistent usage of codes plays a key role. Although this has majorly been a challenge which currently is not the focus of FHIR but it does support the exchange of data using codes, however the usage and interpretation of codes by different systems determines semantic interoperability (Benson, T., & Grieve, G., 2016) (5).

[1] https://www.snomed.org/

[2] https://loinc.org/

2.1 Clinical Registry and FHIR in the Netherlands

In the Netherlands usage of FHIR by different HIS and EMRs is gradually increasing and there have been initiatives and incentives in place to promote its usage (Henrique Martins et al., 2022)(2).

Clinical registries are systems that store minimal data about patients that have undergone a medical treatment or therapy. It acts as a source of evidence to evaluate how effectively the treatment has been delivered and if the clinical guidelines were followed during the process. It also provides with data to evaluate the result of a treatment for the patient also referred to clinical outcome. Clinical registries facilitate comparison of treatment for the same diseases or conditions across countries which can be used as a reference by physicians and hospitals to improve quality of care. Introduction of clinical registries have proven to improve the quality of care and lower the cost of care. With their rising need and popularity to deliver better healthcare across different countries they are also being used in the Netherlands (Hoque et al., 2017)(7).

2.2 Data lakes in Healthcare

Gentner et al. (2023) (8) reviews literature about the contributors of healthcare data and how they can benefit from the data that is managed on data lakes. It argues that data warehouses have long been used for data analytics use cases but due to lack of scalability and lack of ability to store heterogeneous data in different formats makes them not suited for healthcare analytics purpose. Only 20 percent of data stored on EHRs (Electronic Health record) is in structured format and as such data in healthcare is not used to its full potential. Having data from different contributors as seen in Figure 2.2 will also benefit the sources in form of different applications of the stored data.

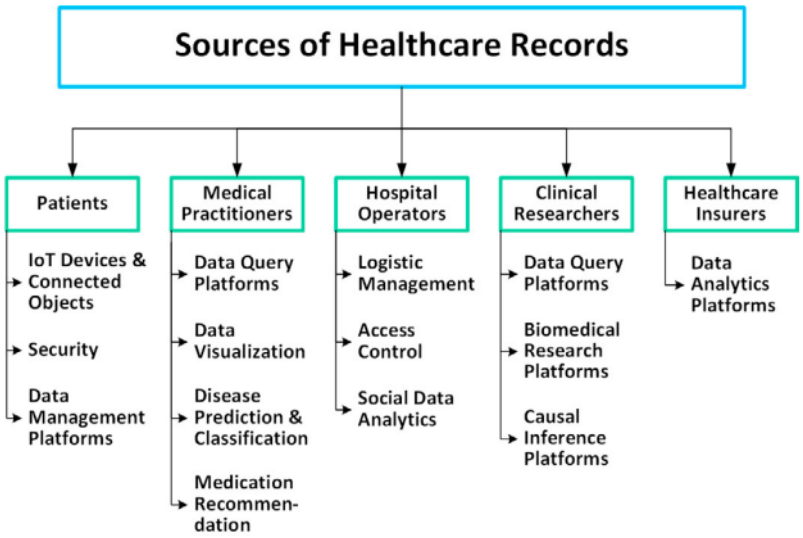

Figure 2.2: Contributors of data and applications of data lakes in healthcare (Gentner et al. (2023) (8)

3

Related Work

A data lake architecture has been proposed by Manco et al., 2023 (9), along with a proof of concept implementation wherein medical vital signs data in the format of time series waveforms and natural language reports has been processed and stored. The data lake, however has been implemented on a local single machine using HDFS file system [1] for storage and Apache NiFi [2] has been used for data ingestion. It is based on a zone based architecture Giebler et al., 2020 (10) wherein data is moved between stages landing zone, raw zone, process and refined zone, which each zone containing a more processed version of data progressively. The architecture however does not consider the problem of metadata management which is a critical component given that medical data is not particularly interoperable and not having metadata could result in erroneous analyses. The architecture also does not take into account the security aspect of data.

Pathling is a tool that has been developed to enable performing analytics on FHIR data, it is based on Apache Spark and provides capabilities to perform ETL(extract transform load) operations and analysis,(Grimes, John et al. 2022) (11). It provides feature to incorporate querying clinical terminology server during analysis. The library however depends on FHIRPath[3] which is a conceptual language to traverse the FHIR resources. There is dependency on usage of FHIRPath to extract the desired data points and results are finally stored as nested structures and this introduces complexity and learning effort to analyse data. In our approach however, we aim to flatten the nested structure entirely and reduce the barrier involved in having to deal with the nesting of data which will enable doing analytics on tabular structure. The flattening is also based on a schema that has been

[1] https://cwiki.apache.org/confluence/display/hadoop
[2] https://nifi.apache.org/
[3] http://hl7.org/fhirpath/N1/

derived from a FHIR profile.

A3[1] is a project that is developed to prepare FHIR data for data analysis. It enables loading the FHIR data to a database BigQuery[2] which is a data warehousing platform on Google Cloud[3]. However, it stores the data in nested format and does not fully flatten the nested data and this adds complexity and knowledge required to analyse the data. Additionally it relies on BigQuery to store and process further. Usage of cloud in context of European healthcare project as in our project is not feasible due to data privacy concerns.

The FHIR Bulk Data API has been developed to extract data in bulks to analytical systems (Liu, D. et al. 2019) (12). The core FHIR API only supported extraction of single patient level data and there was a need to have a mechanism to export population level data, hence to simplify the process the Bulk data API was designed and it supports extraction in NDJSON data format wherein each record is a json object on a newline in a file. To address the scalability of data in terms of storage and compute with respect to file format it provides a comparison using file formats AVRO [4], Parquet[5] and NDJSON. It finds that Parquet is most efficient with storage and read write operations of data. The API however does not address the problem of flattening the nested structure of FHIR data.

Fhircrackr is a package developed in language R to process FHIR data, (Palm, J. et al. 2023)(13). It is capable of flattening the nested FHIR into a table, such that for each type of resource it creates a table in R dataframe. It relies on usage of XPath[6] expression language to achieve the extraction of desired data attributes from the FHIR json. However, it is not developed with the concept of scalability across a distributed computing cluster such as Spark and is primarily suited for usage on single machine with focus on biostatistical research.

FHIR-PYrate is python package that is designed to extract from FHIR servers and process it further to create tabular structure as Dataframes in Pandas python package[7]. It works with all FHIR resources and user has control over the attributes that are required to be extracted from the FHIR json resource (Hosch et al. 2023)(14).However, the package relies on extensive use of FHIRPath query syntax [8] similar to Pathling tool described earlier.Its focus is mainly on extraction from FHIR server and requires sufficient knowledge

[1]https://github.com/sync-for-science/a3
[2]https://cloud.google.com/bigquery
[3]https://cloud.google.com/
[4]https://avro.apache.org/docs/current/
[5]https://parquet.apache.org/
[6]https://www.w3.org/TR/1999/REC-xpath-19991116/
[7]https://pandas.pydata.org/
[8]http://hl7.org/fhirpath/N1/

of FHIRPath to work with. Additionally, to improve performance it utilises multiple processes in Python to speed up the retrieval and querying, but it does not leverage any distributed architecture for processing. It also does not take into account scalability of storing data after tabularisation.

FHIR Python Analysis and Client and Kit (FHIRPACK) is another python tool[1] that has been developed to analyse FHIR data (Jayson et al 2022)(15). However, in the process of extracting relevant information from the FHIR server, FHIRPath syntax is used and it requires sufficient learning effort to analyse FHIR data. It does so using pandas with not much focus on scalability of processing over a distributed architecture.

[1] https://github.com/fhirpack/fhirpack

4

Design and Implementation

In this chapter the design and implementation of project are described. The existing approaches to processing FHIR data that have already been discussed in the Section 3, the key features of our implementation are that we do not rely on the usage of FHIRPath [1] to process FHIR data. We develop a generic code logic that is capable of processing any FHIR resource and thus is resource type agnostic. It makes use of the FHIR profiles defined on the FHIR registry [2] to be used as reference. The chapter is organised as follows.First, we describe the details and need of synthetic data in Section4.1 as real healthcare data is not readily available for the project implementation. Next Section4.2 describes the infrastructure setup that is available for the implementation and testing.

4.1 Synthetic Data

Healthcare data is not readily available for secondary analytical usecases and this hinders the progress that can be made on any development that requires data. Privacy and lack of consent prevents the usage of real world patient data, even if it is available and deanonymized there is still the risk that data could be reidentified to the individual and thus anonymization of data is also not the most secure approach to using data. Several data simulation tools have been developed that are capable of generating healthcare data, however the tools are not open sourced. Moreover they use real world data to generate similar dataset which is not risk free approach of generating data,(Walonoski et al. 2018) (16) discusses the drawbacks and risk involved with synthetic data generators. It also presents a synthetic healthcare data generation tool that is widely being used in industry

[1] http://hl7.org/fhirpath/N1/
[2] https://simplifier.net/

for developments involving the usage of FHIR data format. It generates data based on models of clinical workflow and health data that is publicly available.It generates data for the entire lifetime of a patient and not just pertaining to a specific incidence. It is based on the PADARSER framework that specifies an approach for generating realistic synthetic EHR data (Dube, K., Gallagher, T., et al. 2014)(17). Figure 4.1 illustrates the mechanism of data generation in Synthea.

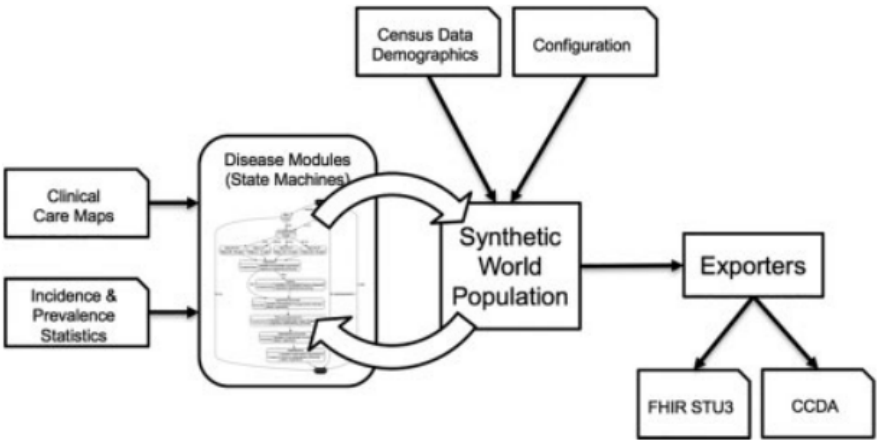

Figure 4.1: Architecture of Synthea (Walonoski et al. 2018)

4.2 Infrastructure Setup

A 3 worker nodes(12 GB Memory, 5 cores) Spark cluster has been used for processing FHIR data from raw JSON format to a tabular format. Spark is an in memory cluster computing framework that is designed for distributing the processing over multiple nodes (Zaharia, M. et al. 2016)(18). MinIO [1] is an object store that is used for storage of data. A modern data storage format, Apache Iceberg[2] is used to save data. It is a table format that abstracts the object stores and exposes them as tables that can be worked on like a typical database table. It provides features such as dynamic partitioning of data, time travel of data allowing viewing of different versions of data across time[3]

[1] https://min.io/
[2] https://iceberg.apache.org/
[3] https://www.dremio.com/resources/guides/apache-iceberg-an-architectural-look-under-the-covers/

4.3 Implementation

The processing logic followed in our implementation has bypassed the usage of FHIRPath query syntax as with other existing libraries and thereby simplifying the complexity of code logic involved in processing FHIR. Figure 4.2 illustrates the logic involved in flattening FHIR json data to a tabular structure. As described in section 4.1 synthetic data has been generated using Synthea tool. The tool generates a json file that consists of all data for the entire duration of patients lifetime. The patient data is generated as a FHIR bundle which is a resource type that consists of all resources associated with the patient such as Observation, Encounter, Condition and so on as described on FHIR documentation[1].

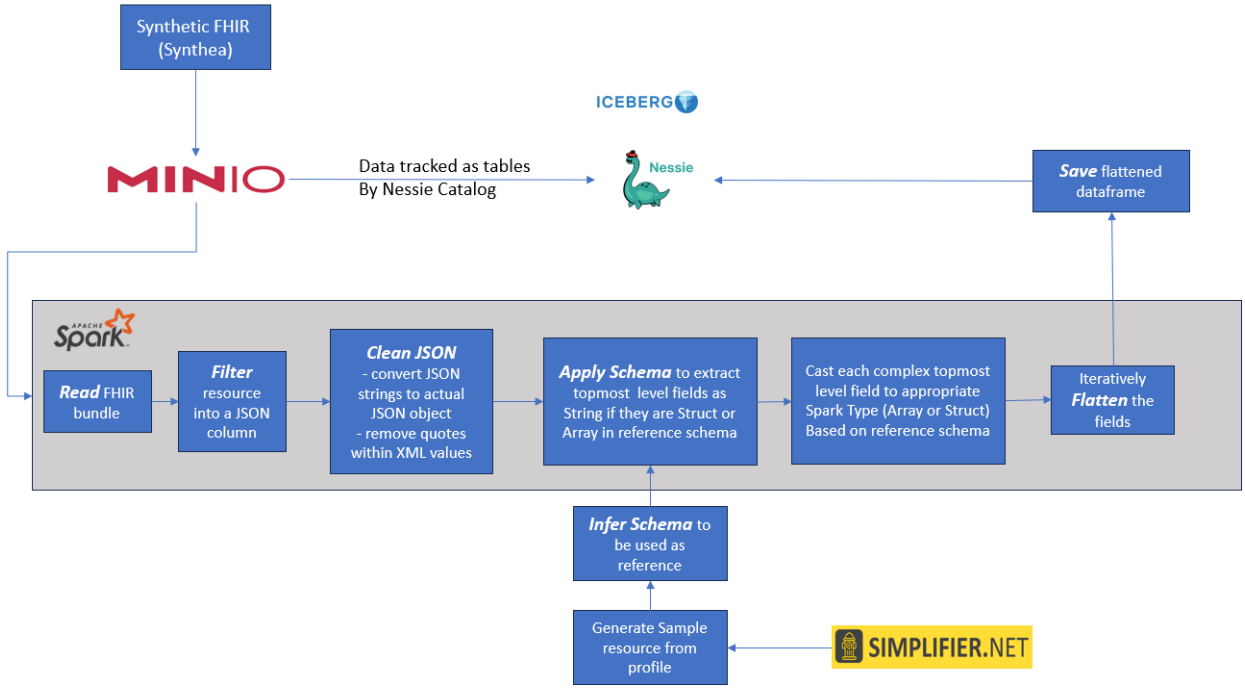

Figure 4.2: Mechanism involved in processing FHIR

The generated synthetic patient records are saved in the datalake MinIO. The synthetic data is read in Spark using the python version, PySpark[2]. The logic is developed to tabularise a FHIR resource based on resource type passed as parameter. The next step is to filter out all the resources of the particular resource types. Once all the resources of a particular type are filtered out, the next step is to clean up the json resources, essentially converting json strings to pure json object values.

[1] https://www.hl7.org/fhir/resourcelist.html
[2] https://spark.apache.org/docs/latest/api/python/index.html

The data/resources are brought in proper json format and now a reference schema is required to map each field to a proper datatype in Spark. To generate the reference schema, a sample resource of the particular resource type being processed is generated using profiles on FHIR registry [1]. The generated sample resource is used to generate the desired Spark schema.

The reference Spark schema is then applied to extract the topmost level fields of the json such that if they are of Struct or Array type then they are first extracted as Stringtype since casting the json data of resources directly to Struct and Array type did not produce consistent results. Each field ws then cast to the appropriate Sparktype, this ensures that non primitive FHIR types as described in FHIR official documentation [2] are represented in equivalent Spark type.

The data once in the correct FHIR type is iteratively flattened such that each nested attribute inside Struct type results in creation of new column and each element inside Array type creates a new row in the final tabular data.The tabular data is then persisted in Iceberg format[3], which is tracked as table by Nessie catalog[4].

[1] https://simplifier.net/
[2] https://www.hl7.org/fhir/datatypes.html
[3] https://iceberg.apache.org/
[4] https://projectnessie.org/

5

Evaluation

In this chapter we discuss the results with respect to the flattening mechanism and metrics on computation and storage required for processing. Figure 5.1 depicts the mechanism and result of flattening. Every innermost nested attribute in the resource creates a new column. Additionally another point to note here is that the presence of null in the output post flattening is due to absence of the particular attribute in the input FHIR resource,

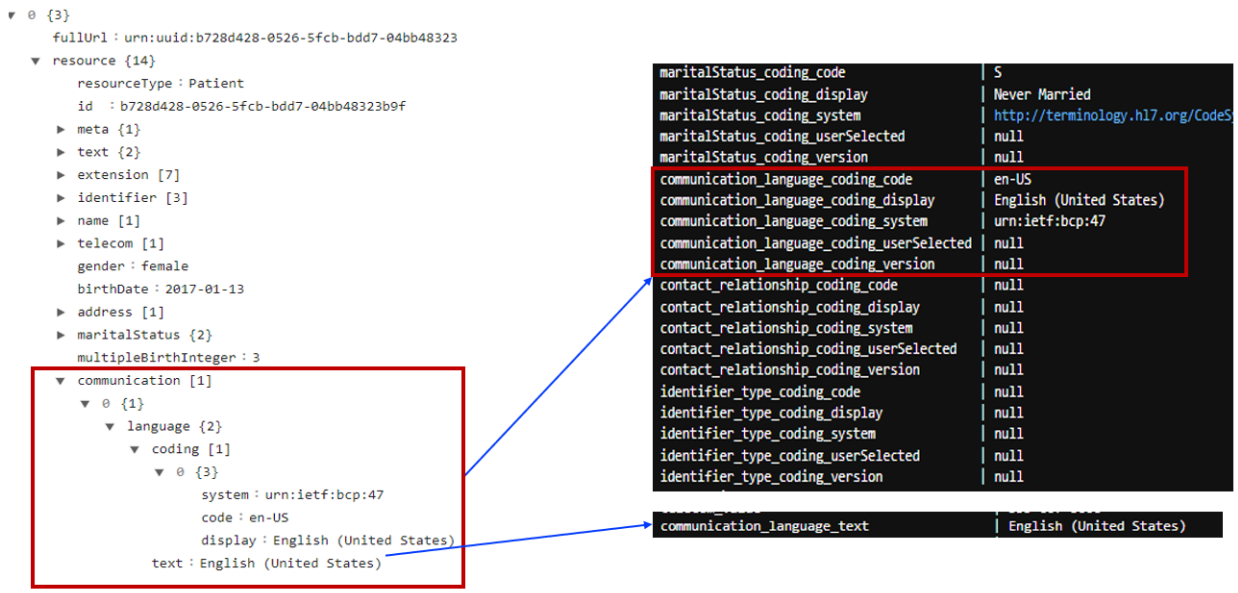

Figure 5.1: Sample Flattening of Patient Resource

Another key observation is that presence of array elements in the input FHIR resource results in creation of multiple rows in output, and in total number rows created is a combinatorial of the distinct values in each of the nested array resource. As seen in Figure

5.2 since in the profile that was used contains identifier and name given attributes on those 2 array elements have result in creation of multiple rows. Here identifier has 3 distinct values and name given has 2 distinct values so in total 6 rows have been created. Hence every row is a unique combination and we indicate that by creation of primary key which is concatentation of id attribute with a incrementing number for all rows of that id. Id attribute of FHIR resource uniquely identifies a FHIR resource on a FHIR server.

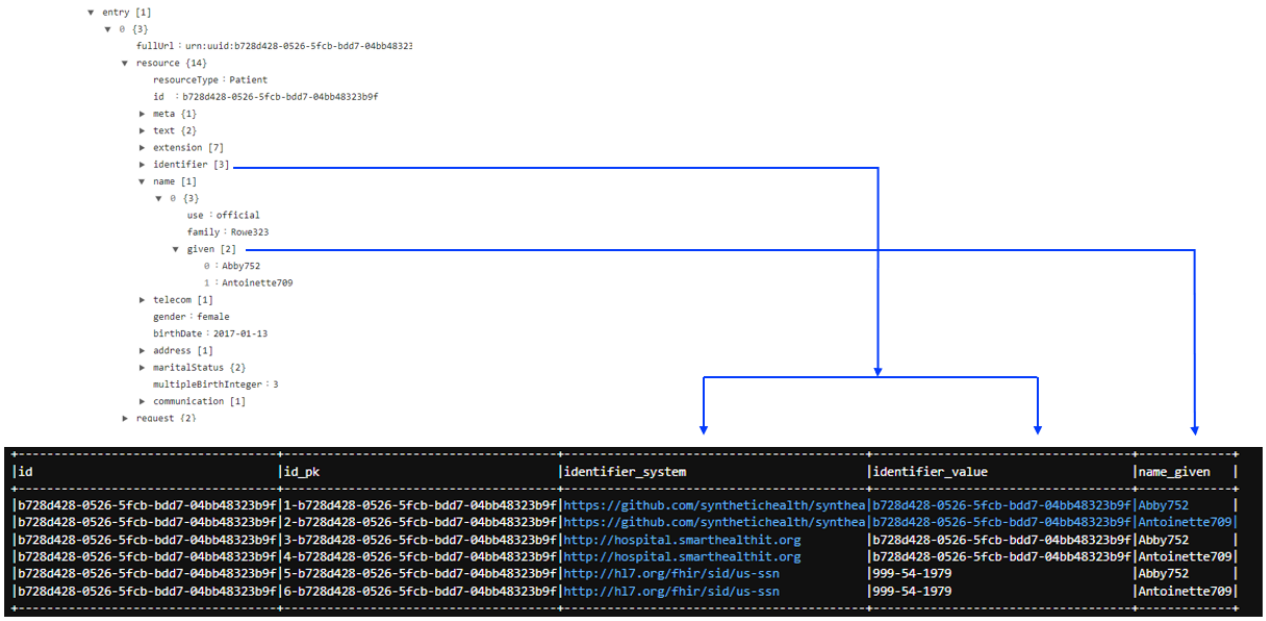

Figure 5.2: Multiplication of rows from single FHIR resources

A distribution of resources as generated the simulator tool Synthea, which we discuss in the Section 4.1 previously, can be seen in Figure 5.3. We notice that the Observation resource is the most frequently occurring resource making approximately half the percentage of all resources generated.Hence it was chosen as the resource to benchmark the computation time for the script. It gives an indicative measure on computation time to flatten large number of resources and get better idea on scalability. Scalability is an important aspect to consider since most of the processing of healthcare data cannot be readily done on cloud resources due to privacy constraints.

Figure 5.4 reveals the total number of all resources in comparison with the observation resources generated with respect to the number of patients as generated by Synthea. Both the measures increase linearly with the number of patients and for 10000 patients the total resources touch 16 million and observation resources are 6.9 million. It takes 2 hours 13 minutes to flatten 6.9 million observation resources. The variation of computation or more

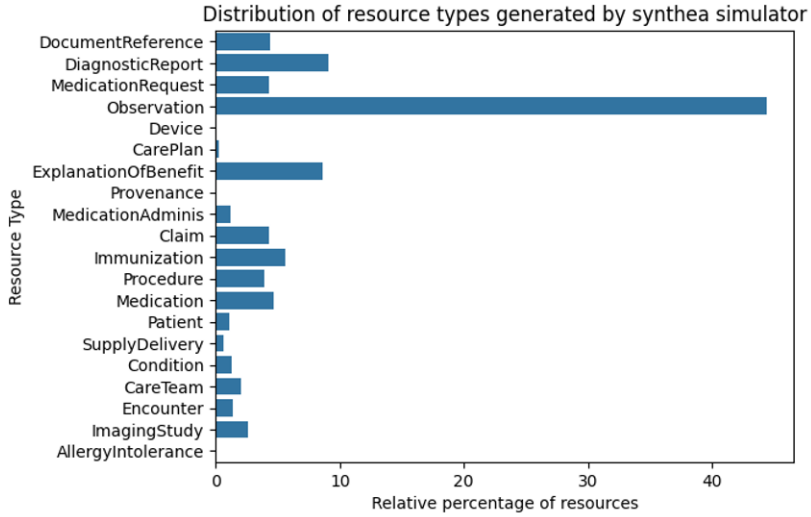

Figure 5.3: Distribution of resources generated by Synthea

precisely the flattening time can be seen in Figure 5.4. This is indicative of the scalability of the solution and that having more nodes or cores with higher RAM would speed up the process. The effective compression of data post processing can be seen in Figure 5.6.After tabularization of FHIR Observation resources and storing it in Iceberg(which underneath stores data in Parquet format) results in tabular FHIR Observation resources which takes approximately 1 percent of original size of input JSON data.

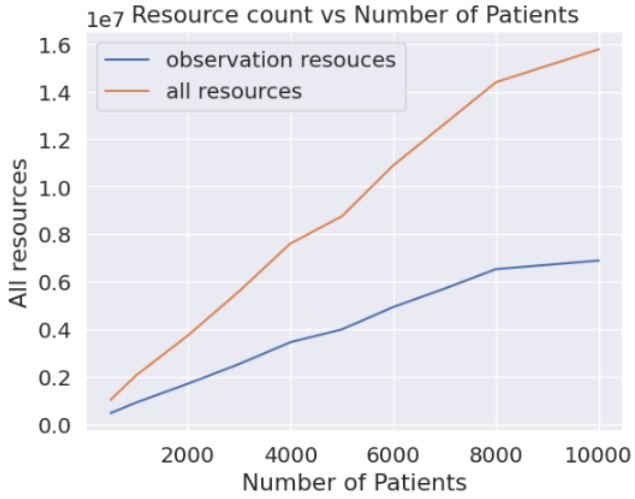

Figure 5.4: Proportion of all resources vs Observation resources

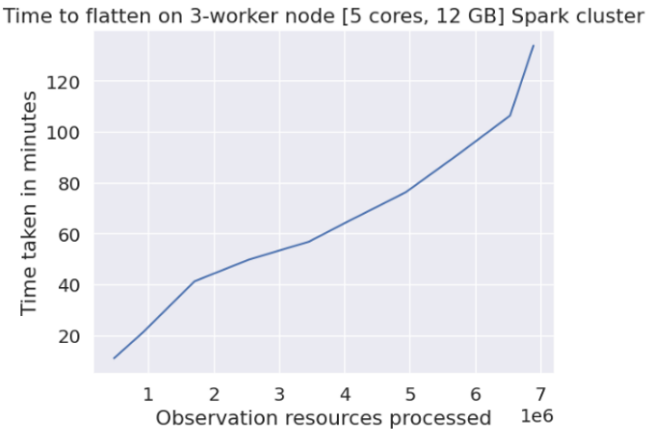

Figure 5.5: Computation time to flattening of resources on a 3-Node Spark Cluster(12 GB, 5 cores)

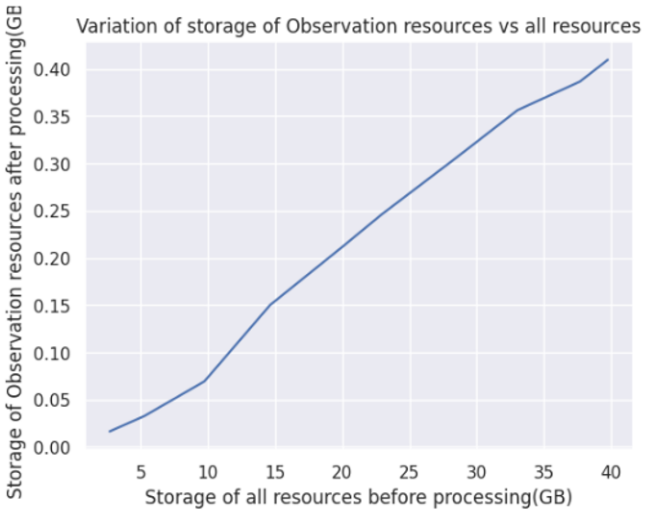

Figure 5.6: Storage of Observation resources after processing in Parquet format vs all resources in the input JSON format

6

Discussion

With respect to research question RQ1 : **How can FHIR data be effectively processed into a tabular structure in a datalakehouse ?** discussed in detail previously in Section 1, We have been able to successfully tabularize the FHIR data without the usage of FHIRPath [1] unlike the existing methods discussed in detail in Section 3. Our approach makes usage of a Spark schema for each resource which has been derived from a sample instance of the FHIR resource generated from using profiles on FHIR registry SIMPLIFIER [2]. The schema can be generated from a generic sample instance of the particular FHIR resource. Important point to note here is that only the fields that are present in the profile get captured from the input data, meaning if the field is not available in the schema then the output data after flattening will not have the field even if it is present in input. Another key aspect to note here is that some fields have variable data type and if the schema does not have the particular data type represented in the sample then the field is not captured in the tabularised form. Fields in the FHIR with variable type are marked by [x]. It is advised to ensure that all possible variations of values corresponding to the datatype are present in the sample instance used to generate reference schema. This way the chance of accidental data loss during tabularization is minimized. As part of future research automation of schema generation using SIMPLIFIER could be possibly looked into, as with our current approach we manually generate the sample FHIR resource and use that to derive the reference schema. Automation could possibly involve maintaining a repository of schemas from which the could be fetched as and when required to process a FHIR resource type.

[1] http://hl7.org/fhirpath/N1/
[2] https://simplifier.net/

As far as research question RQ2: **How does usage of processing engine Spark perform in terms of processing speed and data format Iceberg perform in terms of storage ?** We observe a linear increase in computation time to flatten observation resource. Since observation are most frequently occurring resource they serve as an indicative choice to determine processing time using Spark. The results is seen previously in Results section, Figure 5.5, although the graph is not entirely linear since the cluster is also used by others users in a shared environment. For the storage requirements Iceberg uses the parquet data format [1] to store the data files, which is highly compressed and storage of observation resources which make up nearly half of all resources in the input take 1 percent of space in comparison to the original input data in JSON format.

[1] https://parquet.apache.org/

7

Limitations

In this section we discuss a few key limitations of our research with respect to the results achieved. The Spark cluster we have used for development is a shared environment, meaning that it could have been used by other spark jobs that might be running as we execute our spark jobs. Hence, the performance the results could be vary on an actual real world project where the cluster could be possibly dedicated only to this particular processing at the time of execution.

The time to process is also dependent on the actual size of FHIR bundle resource. In our project we have used synthetic data generator Synthea (Walonoski et al. 2018) (16) to generate FHIR resources, which generates FHIR resources for a patient with most commonly occurring resource types, in our current project the number is 20 while as per FHIR specification in release R5 [1] there are 157 FHIR resource types. Hence the time to flatten which we discuss in Section 5, and storage requirements could be possibly higher. The time to flatten could also vary based on the size of the cluster in use. The cluster we have used is a 3-worker node cluster with 5 cores and storage 12 GB.

[1] https://hl7.org/fhir/R5/resourcelist.html

8

Conclusion

In our project we have been successfully been able to flatten FHIR resource following a generic approach that does not require prior knowledge of FHIRPath that is being followed in existing approaches so far in the FHIR community. We discuss the pros and cons of existing approaches in Section 3. With our current generic approach the only key aspect that requires careful attention to detail is that of the schema that is being used to process the FHIR resource. As long as the schema has all the desired fields with corresponding appropriate FHIR data types the output get generated accordingly. In our approach we have generated the Spark schema based of an example FHIR resource sample in JSON format which we derive using SIMPLIFIER profile [1] that we had created. As part of future research a more automated method of generated schema could be possibly looked into that will simplify the process of flattening FHIR resource. The generic logic we followed can be used to fully flatten any of the FHIR resource types, which are 157 in FHIR release R5 [2]. Another key aspect that could be possibly looked into to improve performance is if all the resources for a particular type to be ingested are extracted beforehand from FHIR server that would possibly speed up the processing since in that case step to extract them from bundle would be eliminated which is followed in this research since input is a FHIR bundle. This would avoid having to read in entire bundle as part of processing which is currently followed in our approach.

[1] https://simplifier.net/

[2] https://hl7.org/fhir/R5/resourcelist.html

References

[2] Henrique Martins et al. **Hospitals on FHIR: Preparing Hospitals for European Health Data Space**, 2022. ii, 1, 2, 6

[5] Tim Benson and Grahame Grieve. *Principles of health interoperability: SNOMED CT, HL7 and FHIR*. Springer, 2016. ii, 4, 5

[8] Tobias Gentner, Timon Neitzel, Jacob Schulze, Felix Gerschner, and Andreas Theissler. **Data Lakes in Healthcare: Applications and Benefits from the Perspective of Data Sources and Players**. *Procedia Computer Science*, **225**:1302–1311, 2023. ii, 6, 7

[1] Moritz Lehne, Sandra Luijten, Paulina Vom Felde genannt Imbusch, and Sylvia Thun. **The Use of FHIR in Digital Health-A Review of the Scientific Literature.** *GMDS*, **267**:52–58, 2019. 1

[3] Duane Bender and Kamran Sartipi. **HL7 FHIR: An Agile and RESTful approach to healthcare information exchange**. In *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pages 326–331. IEEE, 2013. 4, 5

[4] John D Day and Hubert Zimmermann. **The OSI reference model**. *Proceedings of the IEEE*, **71**(12):1334–1340, 1983. 4

[6] Mark L Braunstein. *Health Informatics on FHIR: How HL7's New API is Transforming Healthcare*. Springer, 2018. 5

[7] Dewan Md Emdadul Hoque, Varuni Kumari, Masuma Hoque, Rasa Ruseckaite, Lorena Romero, and Sue M Evans. **Impact of clinical registries on quality of patient care and clinical outcomes: a systematic review**. *PloS one*, **12**(9):e0183667, 2017. 6

[9] Carlo Manco, Tommaso Dolci, Fabio Azzalini, Enrico Barbierato, Marco Gribaudo, and Letizia Tanca. **HEALER: A Data Lake Architecture for Healthcare**. 2023. 8

[10] Corinna Giebler, Christoph Gröger, Eva Hoos, Holger Schwarz, and Bernhard Mitschang. **A zone reference model for enterprise-grade data lake management**. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 57–66. IEEE, 2020. 8

[11] John Grimes, Piotr Szul, Alejandro Metke-Jimenez, Michael Lawley, and Kylynn Loi. **Pathling: analytics on FHIR**. *Journal of Biomedical Semantics*, **13**(1):23, 2022. 8

[12] Dianbo Liu, Ricky Sahu, Vlad Ignatov, Dan Gottlieb, and Kenneth D Mandl. **High performance computing on flat FHIR files created with the new SMART/HL7 bulk data access standard**. In *AMIA Annual Symposium Proceedings*, **2019**, page 592. American Medical Informatics Association, 2019. 9

[13] Julia Palm, Frank A Meineke, Jens Przybilla, and Thomas Peschel. **"fhircrackr": An R Package Unlocking Fast Healthcare Interoperability Resources for Statistical Analysis**. *Applied Clinical Informatics*, **14**(01):054–064, 2023. 9

[14] René Hosch, Giulia Baldini, Vicky Parmar, Katarzyna Borys, Sven Koitka, Merlin Engelke, Kamyar Arzideh, Moritz Ulrich, and Felix Nensa. **FHIR-PYrate: a data science friendly Python package to query FHIR servers**. *BMC Health Services Research*, **23**(1):734, 2023. 9

[15] Jayson Salazar Rodriguez. **FHIRPACK (FHIR Python Analysis Client and Kit)**. 2022. 10

[16] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. **Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record**. *Journal of the American Medical Informatics Association*, **25**(3):230–238, 2018. 11, 21

[17] Kudakwashe Dube and Thomas Gallagher. **Approach and method for generating realistic synthetic electronic healthcare records for secondary use**. In *Foundations of Health Information Engineering and Systems: Third International Symposium, FHIES 2013, Macau, China, August 21-23, 2013. Revised Selected Papers 3*, pages 69–86. Springer, 2014. 12

[18] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. **Apache spark: a unified engine for big data processing**. *Communications of the ACM*, **59**(11):56–65, 2016. 12

Appendix