MHDP: An Efficient Data Lake Platform for Medical Multi-source Heterogeneous Data

Peng Ren[1], Shuaibo Li[2](✉), Wei Hou[2], Wenkui Zheng[2], Zhen Li[2], Qin Cui[2], Wang Chang[2], Xin Li[3], Chun Zeng[4], Ming Sheng[1], and Yong Zhang[1]

[1] BNRist, DCST, RIIT, Tsinghua University, Beijing 100084, China
{renpeng,shengming,zhangyong05}@tsinghua.edu.cn
[2] Henan University, Kaifeng 475004, China
houwei@henu.edu.cn, lz@vip.henu.edu.cn
[3] Beijing Tsinghua Changguang Hospital, School of Clinical Medicine, Tsinghua University, Beijing 102218, China
Horsebackdancing@sina.com
[4] Viewhigh Technology (Beijing) Co., Ltd., Beijing 100062, China
zengchun@viewhigh.com

Abstract. In medical domain, huge amounts of data are generated at all times. These data are usually difficult to access, with poor data quality and many data islands. Besides, with a wide range of sources and complex structure, these data contain essential information and are difficult to manage. However, few existing data management frameworks based on Data Lake excel in solving the persistence and the analysis efficiency for medical multi-source heterogeneous data. In this paper, we propose an efficient **M**ulti-source **H**eterogeneous **D**ata Lake **P**latform (MHDP) to realize the efficient medical data management. Firstly, we propose an efficient and unified method based on Data Lake to store data of different types and different sources persistently. Secondly, based on the unified data store, an efficient multi-source heterogeneous data fusion is implemented to effectively manage data. Finally, an efficient data query strategy is carried out to assist doctors in medical decision-making. In-depth analysis on applications shows that MHDP delivers better performance for data management in medical domain.

Keywords: Data Lake · Medical multi-source heterogeneous data · Efficient · Persistence of data storage

1 Introduction

With the advancement of information technology, most hospitals and other medical institutions have realized large-scale informationization. However, the architecture of hospital information system (HIS) is complex. Furthermore, without uniform standard, the information systems used by different medical institutions are distinctive, resulting in different types and structures of medical data. These data include structured data in MySQL, Oracle, SQL Server and other related databases, semi-structured data in the format of CSV, JSON, XML, and unstructured data, such as EMRs, ECGs, CTs, MRIs, etc. In front of massive multi-source heterogeneous data in medical domain, how to acquire, store, and provide methods for unified management is the primary key problem [1, 2].

The works of medical data management in the early stage adopted Database [3, 4]. Database technology can cater to the needs of rapid Insert, Delete and Query in the case of relatively few medical data to deal with online transaction affairs in medical domain. However, Database technology cannot well deal with the analysis tasks which are characterized by reading a large amount of data. To address this issue, Data Warehouse technology is applied to medical data management [5, 6]. The medical data management frameworks based on Data Warehouse provide unified data support for medical workers' data management analysis and treatment decision-making. More recently, due to the demand for data flow keeps growing, Data Warehouse technology cannot cope with the remarkable challenge of data management.

Confronted by the challenges brought by Database and Data Warehouse in medical data management, researchers began to pay attention to Data Lake technology. Data Lake can integrate complex data by multiple means, and there are related researches in medical data management [7, 8]. However, the current data management frameworks in medical domain based on Data Lake technology show three challenges as follows:

1. Framework extendibility. In the real scene, the data generated in medical domain is multi-source, multi-structured and massive. Without broader data source range, existing works obtain these original data by a single approach, which demonstrates less extendibility.
2. Data persistence. When data is stored, it is sometimes lost. Though some data storage methods have been well studied, most of them ignore the data persistence over a longer time.
3. Efficiency. Usually, medical workers frequently need to obtain real-time results when analyzing and querying data based on their illness. However, most frameworks leave efficiency optimization to be desired, affecting the real-time analysis of illness for medical workers.

Contributions. The main contributions are summarized as follows:

1. An efficient multi-source heterogeneous data storage persistence method is proposed. With the support of distributed computing, medical multi-source heterogeneous data is converted into one unified data model to store data efficiently and permanently. Meanwhile, it considers recording the data sources and changes to ensure the traceability of the data sources.
2. An efficient multi-source heterogeneous data fusion method is presented. The data fragmentation triggered by the discrete distribution of medical data after storage cannot depict the patient's condition completely, which may lead doctors to make wrong decisions on the condition of patients. The proposed method can effectively tackle this problem and concentrate the data efficiently.

3. An efficient multi-source heterogeneous data query method is put forward. Different from the traditional query method, the proposed method takes into account the distributed computing power as well as space and time required for query, thus achieving high efficiency to underpin doctors' rapid medical decision-making.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 gives the platform architecture. Section 4 presents persistence of data storage. Section 5 describes data fusion. Section 6 implements data query. Section 7 summarizes the paper and points out future research direction.

2 Related Work

The concept of Data Lake was first put forward by Dixon to deal with the challenges brought by Data Warehouse. [9]. In practical applications, Data Lake technology has been employed in academia and industry. The companies that have deployed the Data Lake in the industrial community include AWS, Huawei, Alibaba, Azure, etc. The achievements of the deployment in academia include CLAMS [7], CM4DL [8] etc. Inspired by these Data Lake frameworks, the research work of Data Lake management framework in medical domain began to emerge constantly. Due to the flexibility of Data Lake technology and the diversity of diseases, the existing researches work on Data Lake management framework in medical domain can be divided into two categories, namely, Data Lake technology research and disease analysis research.

With regard to Data Lake technology research, Mesterhazy et al. [10] aimed at the medical image data such as X-Ray, MRI, CT, etc., and adopted cloud-based distributed computing technology to carry out rapid turnover of medical imaging, thus generating a medical image Data Lake processing framework. Hai et al. [11] designed a Web Data Lake framework called Constance, providing users with unified query and data exploration by applying an embedded query rewrite engine. They have been applied in the open source medical engineering project *miMappa*, which demonstrates the practicability of the framework in medical data management. Walker et al. [12] regarded metadata management as the core part of the Data Lake system, and proposed a personal Data Lake framework for personal customization, which could be used to store, analyze and query personal medical data as well as generate personal medical reports conveniently and quickly. Bozena et al. [13] integrated fuzzy technology and declarative U-SQL language into data analysis, and developed a fuzzy search scheme for big Data Lake, which could analyze a large amount of medical data in a distributed way. This scheme is equipped with good scalability, which is a successful step to realize the large-scale medical data declaration.

In terms of disease diversity research, Alhroob et al. [14] designed a data management framework based on Data Lake technology for semi-structured data of cardiovascular and cerebrovascular diseases by using k-means clustering with categorical and numerical data with big data characteristics. Maini et al. [15] proposed a solution to optimize data storage and analysis, applied it to the prediction of cardiovascular diseases, and constructed a prediction framework for cardiovascular diseases based on Data Lake.

Kachaoui et al. [16] came up with a Data Lake framework combining semantic web services (SWS) and big data features in order to predict the case of COVID-19 in real time. This framework extracts crucial information from multiple data sources to generate real-time statistics and reports.

3 The MHDP Architecture

This section introduces the platform architecture from two aspects: software architecture and deployment architecture.

3.1 Software Architecture

As shown in Fig. 1, the software architecture of MHDP is bottom-up, with a total of 5 layers, namely data storage layer, calculation layer, function layer, API layer and application layer.

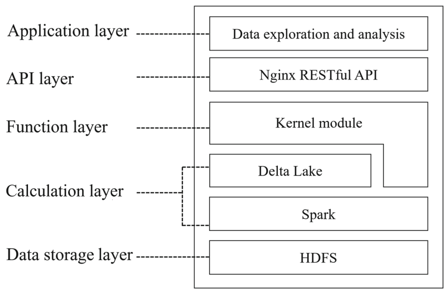


Fig. 1. MHDP software architecture

Data Storage Layer: The bottom layer of MHDP is the data storage layer with storage function provided by HDFS. HDFS is a very popular distributed file system, which provides high fault tolerance and high throughput storage capacity.

Calculation Layer: The calculation layer above the HDFS is supported by Spark which is a fast and general computing engine designed for large-scale data processing, and has formed an ecosystem with rapid development and wide application. Delta Lake is an open source storage layer provided by the Databricks, which brings reliability to the Data Lake. Delta Lake provides ACID transaction, extensible metadata processing, and unifies streaming and batch data processing. Delta Lake is fully compatible with Spark API.

Function Layer: Based on Delta Lake and Spark API, we modify and encapsulate them with code, and form various functional methods, such as multimodal data fusion, data query and other functions. These methods are the core components of MHDP.

API Layer: This layer encapsulates the functions of MHDP into Restful API. Nginx forwards the received requests to the master nodes of different clusters according to their types, and then uses cluster computing power to complete the corresponding tasks.

Application Layer: The top layer is the data exploration and analysis layer, which will package Restful APIs into a visualization module providing various services, interacting with users directly, and assisting users to complete various scientific research tasks of data analysis and data exploration.

3.2 Deployment Architecture

Figure 2 shows the deployment architecture of MHDP. First, a large number of servers are used as DataNode in Hadoop. Meanwhile, the servers are divided into two groups of Worker to provide computing power support for the two groups of Spark clusters. Then, several servers assume the identity of NameNode and Master with Zookeeper, building HA (Highly available) cluster. Once the NameNode and Master are down, the platform will automatically convert the server state from ready to activation.

There are several servers in the Restful Server, which split the services of MHDP. Each server will assume different responsibilities and provide different services. The requests sent by the client will be forwarded by the Nginx cluster to perform reverse proxy according to the category, so as to improve the reliability of the platform by improving the concurrency.

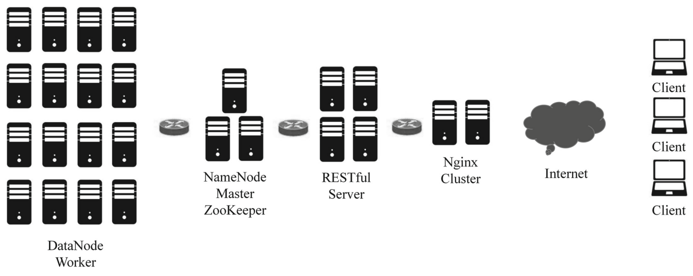


Fig. 2. MHDP deployment architecture

4 Data Persistence

Facing massive multi-source heterogeneous data in medical domain, the primary key problem is how to acquire, store, and provide methods for unified management. In this section, we introduce how to solve the problem mentioned above.

4.1 Data Acquisition Mode

In MHDP, these data sources are integrated through transportation mode and pipeline mode.

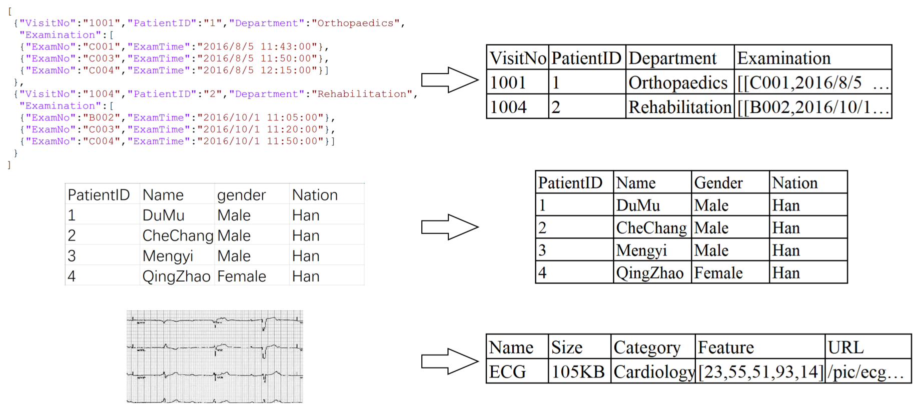
Transportation Mode. The interface is used to upload the data files to the reserved space of the Data Lake with SFTP protocol. Meanwhile, during the upload process, the interface will collect the meta information of the data files, such as data source, data structure, data type, data address, etc., and store them in a meta information database. In the future, when the data is traced due to medical safety problems, this information will provide support.

Pipeline Mode. In the pipeline mode, the data will not be store in MHDP. This mode will establish a session for the target data address. When we need to use the data, we will directly put it into the memory of the server cluster by network transmission, not in the disk. When we read the data of a certain pipeline type for the first time, there is also a method to obtain the meta information of the data and save it to the meta information database. The biggest difference between this mode and the Transportation Mode is that we will not save it in the Data Lake. We do not provide storage services for this type of data, and we do not need to be responsible for the security problems such as the leakage of this type of data.

4.2 Efficient Data Persistence

For data from different sources, the persistence methods are different. The data file is uploaded to the HDFS of the Data Lake for persistence with the Transportation Mode, whereas they will be directly loaded into the memory of the Data Lake by the Pipeline Mode when needed. According to different types of data, there are also differences about persistence methods. For structured data and semi-structured data, no additional changes are needed. But for unstructured data, it is difficult to directly obtain valuable information from itself, and we need to further process it. Therefore, in terms of persistence strategy, the original data are first stored. In addition, different embedding models are used to generate different feature vectors according to the requirements, and the information such as the address and size of the original data are stored structurally.

MHDP employs two kinds of models, Dataframe and DeltaTable, as the data models in the data processing stage. These data models can provide unified data models for efficient management of data with different structures and different sources. These data models and methods will transform the computing process of data into the distributed computing task of Spark, and fully mobilize the cluster computing power. DataFrame is a regulation data model in the platform, and DeltaTable is similar to DataFrame. It is the data structure that will be employed when Delta Lake is used. There are mutual transformation methods between DataFrame and DeltaTable. The DataFrame will be discussed as the main body in the following. The Fig. 3 shows an example of using DataFrame as data model for data in the format of CSV, JSON and image data. Whether it is semi-structured data in the format of CSV, JSON, or unstructured data like images, the platform provides an appropriate way to convert them into DataFrame. Structured and semi-structured data are supported by Spark and Delta Lake. For unstructured data, we will convert the feature vectors of these data and the path information for data traceability, as well as data size, data classification and other information into DataFrame for storage.



```
[
 {"VisitNo":"1001","PatientID":"1","Department":"Orthopaedics",
  "Examination":[
  {"ExamNo":"C001","ExamTime":"2016/8/5 11:43:00"},
  {"ExamNo":"C003","ExamTime":"2016/8/5 11:50:00"},
  {"ExamNo":"C004","ExamTime":"2016/8/5 12:15:00"}]
 },
 {"VisitNo":"1004","PatientID":"2","Department":"Rehabilitation",
  "Examination":[
  {"ExamNo":"B002","ExamTime":"2016/10/1 11:05:00"},
  {"ExamNo":"C003","ExamTime":"2016/10/1 11:20:00"},
  {"ExamNo":"C004","ExamTime":"2016/10/1 11:50:00"}]
 }
]
```

VisitNo	PatientID	Department	Examination
1001	1	Orthopaedics	[[C001,2016/8/5 …
1004	2	Rehabilitation	[[B002,2016/10/1…

PatientID	Name	gender	Nation
1	DuMu	Male	Han
2	CheChang	Male	Han
3	Mengyi	Male	Han
4	QingZhao	Female	Han

PatientID	Name	Gender	Nation
1	DuMu	Male	Han
2	CheChang	Male	Han
3	Mengyi	Male	Han
4	QingZhao	Female	Han

Name	Size	Category	Feature	URL
ECG	105KB	Cardiology	[23,55,51,93,14]	/pic/ecg…

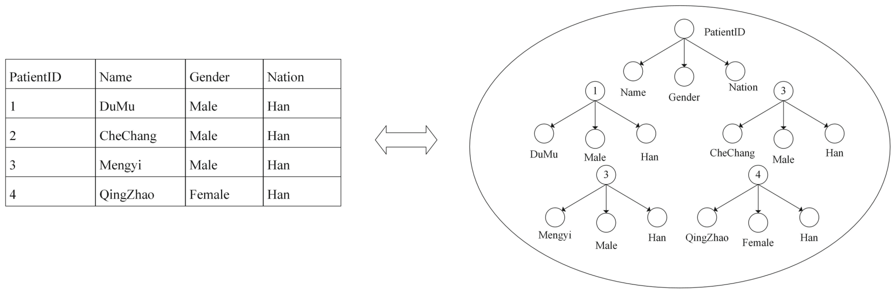
Fig. 3. The conversion from various types of data to DataFrame

5 Data Fusion

The data fragmentation cannot depict the patient's condition completely, which may lead doctors to make wrong decisions on the condition of patients. Aiming at this situation, data fusion is one of the key problems that MHDP solves.

5.1 Integration Method

In general, we can treat data as a set, and every piece of information in the data is an element in the set. This information can be represented by a tree structure, so every piece of data can be regarded as a set of trees. As shown in Fig. 4, it is an ordinary two-dimensional table, and the set of transformed trees is on the right side. The root node of the tree is generally assumed by the primary index.

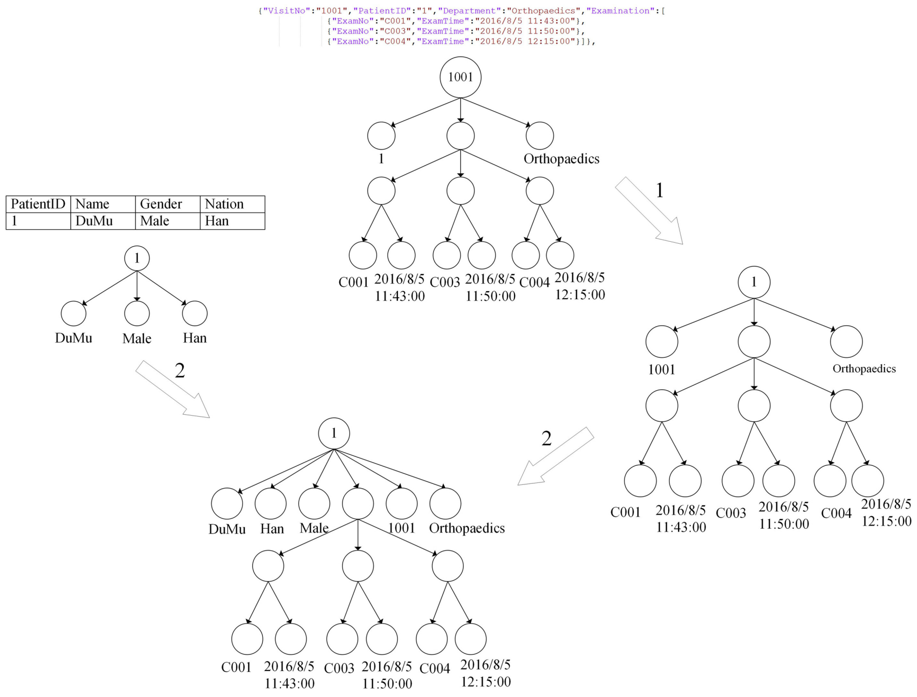


PatientID	Name	Gender	Nation
1	DuMu	Male	Han
2	CheChang	Male	Han
3	Mengyi	Male	Han
4	QingZhao	Female	Han

Fig. 4. The transformation between 2D table and tree

We use two cases to illustrate the merging process.

Case I. The first case is tree merging. If the Merged Key is the root node of the normal tree, then the merging process is the same as the special tree. If the Merged Key is the leaf node of the second layer in the normal tree, because the root node is adjacent to the second layer node, simply convert the position between the Merged Key and the root node of normal tree. Taking the fusion of patient information and treatment information as an example as shown in Fig. 5, the two datasets are associated with each other by the PatientId, which is used as the Merged Key. When the PatientID nodes in the treatment information dataset are located in the second layer, they should be converted to root node and be special tree, then the two trees are merged to get the merged result according to the root node.



```
{"VisitNo":"1001","PatientID":"1","Department":"Orthopaedics","Examination":[
                {"ExamNo":"C001","ExamTime":"2016/8/5 11:43:00"},
                {"ExamNo":"C003","ExamTime":"2016/8/5 11:50:00"},
                {"ExamNo":"C004","ExamTime":"2016/8/5 12:15:00"}]},
```

PatientID	Name	Gender	Nation
1	DuMu	Male	Han

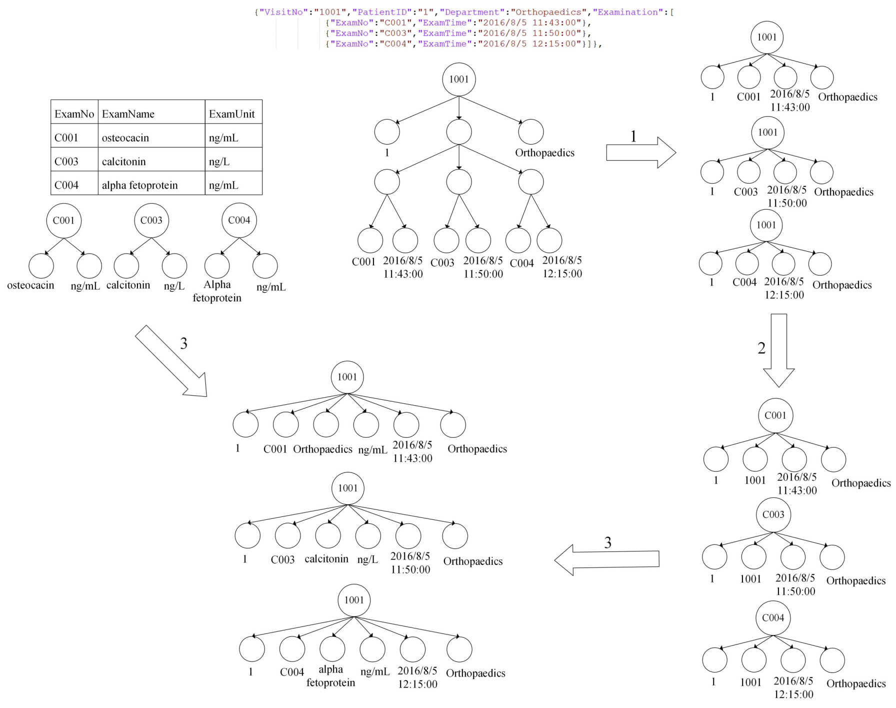
Fig. 5. Tree merging case I

Case II. The case that normal tree with the Merged Key below the second layer, occurring when the Merged Key is nested within a certain data value. The value is often an object or a list, or even a complex structure that both exist. As shown in Fig. 6, taking the merging of treatment information and examination information as an example, the two datasets need to use the examination number ExamNo as the Merged Key. There are three steps:

Step 1: Extracting these nested data. In view of this nesting situation, we design a special method depending on two basic processes. One is to split the Object existing in the nested data to expand the nodes. The other is to split the Array in the nested data to expand the elements. We can always turn the splitting process of a data into a finite combination of these two processes. These two processes can be used to process the data until the Merged Key appears at the root node or the leaf node on the second layer.

Step 2: Like the **Case I** above, changing the position between Merged Key and root node after getting the split tree, a special tree can be constructed.

Step 3: Merging the node to get the final result.



```
{"VisitNo":"1001","PatientID":"1","Department":"Orthopaedics","Examination":[
          {"ExamNo":"C001","ExamTime":"2016/8/5 11:43:00"},
          {"ExamNo":"C003","ExamTime":"2016/8/5 11:50:00"},
          {"ExamNo":"C004","ExamTime":"2016/8/5 12:15:00"}]},
```

ExamNo	ExamName	ExamUnit
C001	osteocacin	ng/mL
C003	calcitonin	ng/L
C004	alpha fetoprotein	ng/mL

Fig. 6. Tree merging case II

5.2 Implementation Mechanism

In MHDP, we use DataFrame as the collection of trees, that is, the loading container of data. Each row of records in DataFrame corresponds to a tree.

Using DataFrame, we can call computing power of the clusters to speed up the process of data fusion. According to our method, we can solve the problem of merging ***N*** sets. No matter whether the merging node is nested or non-nested, we can always merge several data sets into a new data set, and use computing power of the clusters to speed up the process. When the merge node ExamNo in the treatment information is nested in Examination, we use two splitting methods to transform the DataFrame to get the following DataFrame. It can be seen that the ExamNo has been exposed to the outside. Next, only merging with the ExamNo in the examination data can get the final result. In the final result, it can be clearly understood what tests each patient has done and the relationship between these tests in time.

We encapsulate the function of data fusion into a Restful API to support the implementation of various functions of the platform. The API will return different values according to the running situation of the code. When the code runs as expected, it will return the execution situation: true and the corresponding data. Once the code is wrong, it will return the corresponding error information according to the error situation, so as to help medical workers understand the current situation.

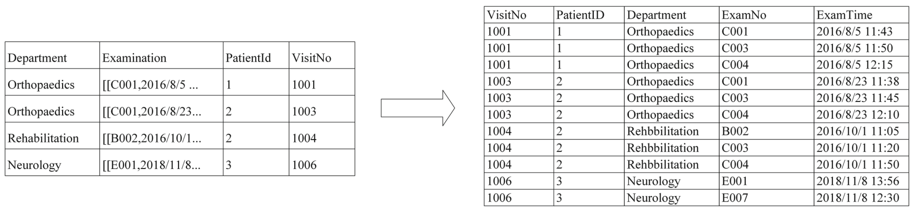
6 Data Query

The data management framework in medical domain not only needs to store and manage a large number of multi-source heterogeneous data, but also has the query function to assist doctors to make in-time medical decisions. This section introduces how to efficiently implement data query in MHDP based on above mentioned.

6.1 Query on Medical Data Lake

In the traditional situation, joint query is the most time-consuming whether by Data Warehouse or data container, which will consume a lot of space and time. Most of the inefficient query were due to too many Joins. Our query mechanism is to do single table query based on data fusion. When query conditions exist in both set A and set B, we fuse set A and set B into set C. We only execute single table query for table C. Due to using Spark SQL and RDD mechanism, parallel computing can be used to accelerate the queries, so our query mechanism is very efficient in MHDP.

Generally, the data sets have certain columns. For the data sets with uncertain columns, we design a method. This method obtains the maximum structure of data first, and then uses it as the basis of data query. For the data sets with deep nesting, we also provide the function of data splitting in data fusion, releasing the data to be queried, as shown in Fig. 7.



Department	Examination	PatientId	VisitNo
Orthopaedics	[[C001,2016/8/5 ...	1	1001
Orthopaedics	[[C001,2016/8/23...	2	1003
Rehabilitation	[[B002,2016/10/1...	2	1004
Neurology	[[E001,2018/11/8...	3	1006

VisitNo	PatientID	Department	ExamNo	ExamTime
1001	1	Orthopaedics	C001	2016/8/5 11:43
1001	1	Orthopaedics	C003	2016/8/5 11:50
1001	1	Orthopaedics	C004	2016/8/5 12:15
1003	2	Orthopaedics	C001	2016/8/23 11:38
1003	2	Orthopaedics	C003	2016/8/23 11:45
1003	2	Orthopaedics	C004	2016/8/23 12:10
1004	2	Rehbbilitation	B002	2016/10/1 11:05
1004	2	Rehbbilitation	C003	2016/10/1 11:20
1004	2	Rehbbilitation	C004	2016/10/1 11:50
1006	3	Neurology	E001	2018/11/8 13:56
1006	3	Neurology	E007	2018/11/8 12:30

Fig. 7. The splitting of nested data

6.2 Query API

MHDP encapsulates the query method of data, and packages it into a Restful API to provide services for the exploration and analysis of medical data. This API requires two parameters, one is the address of the data file, the other is the SQL statement. When the code runs as expected, it will return the execution status: true, and the corresponding data. Once there is an error in the code, it will return the corresponding error information according to the error situation, so as to help users understand the current situation.

Taking the query of treatment information and examination information as an example, MHDP will fuse the two first, and then only need to query the results after fusion. When enter SQL standard query codes: 'SELECT ExamUnit, count (ExamUnit) as times from d6 group by ExamUnit order by times', the number of times that the current patient's examination unit appears will be displayed.

6.3 Application

MHDP has been deployed to XXX hospital to support data analysis in knee osteoarthritis domain. These knee osteoarthritis data, spanning from 2008 to present, involving 128,000 people and a total of 230,000 visits, include structured EMRs and unstructured medical images and texts.

Compared with the traditional Data Analysis System (DAS) of XXX hospital, MHDP queries data of different types uniformly and has higher query efficiency. We compare the performance of the Oracle-based DAS and MHDP by comparing the time consumed to execute the same query on the same dataset. The experiment dataset is 200 GB of structured data extracted from EMRs. The query task is to count the effective rate of patients with different genders and ages after PRP treatment. The experiment results show that MHDP-based queries outperform the Oracle-based queries by $3\times$.

7 Conclusion and Future Work

This paper proposes an efficient and medical-oriented Data Lake platform to manage massive medical data. Different from the traditional medical data management framework, MHDP adopts two unified data models, DataFrame and DeltaTable, to provide scalable and persistent storage capacity for medical multi-source heterogeneous data, coping with the increasing amount of data and the analysis needs of medical related data. During the construction of the platform, Spark and Nginx are used to store, transform and query medical data to provide fast computing capabilities, making the analysis of medical data more efficient. In the future, we will focus on optimizing the traceability of data. Not only do we record when the data is loaded into the Data Lake, but also track the changes of data during data fusion, so as to ensure that each piece of data has a clear provenance.

Acknowledgements. This work was supported by National Key R&D Program of China (2019YFC0119600).

References

1. Lee, C., Yoon, H.: Medical big data: promise and challenges. Kidney Res. Clin. Pract. **36**(1), 3–11 (2017)
2. Kalkman, S., Mostert, M., Beauvisage, N., et al.: Responsible data sharing in a big data-driven translational research platform: lessons learned. BMC Med. Inform. Decis. Mak. **19**(1), 283 (2019)
3. Mitchell, J., Naddaf, R., Davenport, S.: A medical microcomputer database management system. Methods Inf. Med. **24**(2), 73–78 (1985)
4. Mohamad, B., Orazio, L., Gruenwald, L.: Towards a hybrid row-column database for a cloud-based medical data management system. In: 1st International Workshop on Cloud Intelligence, pp. 1–4. ACM, New York (2012)
5. Sebaa, A., Chikh, F., Nouicer, A., et al.: Medical big data warehouse: architecture and system design, a case study: improving healthcare resources distribution. J. Med. Syst. **42**, 59 (2018)
6. Farooqui, N., Mehra, R.: Design of a data warehouse for medical information system using data mining techniques. In: 5th International Conference on Parallel Distributed and Grid Computing, pp. 199–203. IEEE, New York (2018)
7. Farid, M., Roatis, A., LLyas, F., et al.: CLAMS: bringing quality to Data Lakes. In: 2016 International Conference on Management of Data, pp. 2089–2092. ACM, New York (2016)
8. Alserafi, A., Abello, A., Romero, O., et al.: Towards information profiling: data lake content metadata management. In: 16th International Conference on Data Mining Workshops, pp. 178–185. IEEE, New York (2016)
9. Dixon, J.: Pentaho, Hadoop, and data lakes. https://jamesdixon.woedpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Accessed 25 May 2021
10. Mesterhazy, J., Olson, G., Datta, S.: High performance on-demand de-identification of a petabyte-scale medical imaging data lake. In: CoRR abs/2008.01827 (2020)
11. Hai, R., Geisler, S., Quix, C.: Constance: an intelligent data lake system. In: 2016 International Conference on Management of Data, pp. 2097–2100. ACM, New York (2016)
12. Walker, C., Alrehamy, H.: Personal Data Lake with data gravity Pull. In: 5th International Conference on Big Data and Cloud Computing, pp. 160–167. IEEE, New York (2015)
13. Bozena, M., Marek, S., Dariusz, M.: Soft and declarative fishing of information in big data lake. IEEE Trans. Fuzzy Syst. **26**(5), 2732–2747 (2018)
14. Alhgaish, A., Alzyadat, W., Alfayoumi, M., et al.: Preserve quality medical drug data toward meaningful data lake by cluster. Int. J. Recent Technol. Eng. **8**(3), 270–277 (2019)
15. Maini, E., Venkateswarlu, B., Gupta, A.: Data lake-an optimum solution for storage and analytics of big data in cardiovascular disease prediction system. Int. J. Comput. Eng. Manag. **21**(6), 33–39 (2018)
16. Kachaoui, J., Larioui, J., Belangour, A.: Towards an ontology proposal model in data lake for real-time COVID-19 cases prevention. Int. J. Online Biomed. Eng. **16**(9), 123–136 (2020)