# Research on power grid topology data management technology for multi-temporality

Sen Pan*[,a,b], Aihua Zhou [a,b], Jing Jiang [a,b], Hongbin Qiu [a,b], Junfeng Qiao [a,b], Xingde Huang [c]
[a]State Grid Smart Grid Research Institute Co.,LTD.,nanjing, Jiangsu,210003, China; [b]State grid key laboratory of information & Network Security, Nanjing,210003, China; [c]State Grid Shanghai Municipal Electric Power Company, Shanghai,200025, China

## ABSTRACT

Aiming at the typical characteristics of constant changes in the power grid topology caused by multiple factors such as large-scale access of distributed power supply and diversified loads, this paper analyzes the shortcomings of existing power grid topology data management methods in temporal management of topology data, designs a power grid topology data model integrating time dimension based on big data and graph computing technology, and establishes a multi - temporal power network topology data management method. This method realizes the importing, processing, storage and reading of multi-temporal power grid topology, which provides a new solution for the power grid topology data management with multi-temporal version attribute, and also provides a powerful data support for the subsequent analysis and application of multi-temporal power grid topology data.

**Keywords:** Multi-temporal; power grid topology; graph calculation; data management

## 1. INTRODUCTION

The grid topology data is mainly derived from the internal data of power enterprises, including the equipment model connection relationship in the power grid production management system (PMS), the power grid common information model (CIM) in the grid dispatching automation system, and the topology location information in the grid geographic information system (GIS). At present, the general process of power grid topology data management is: establish the graph data model of the power grid topology network, use the graph database software to import the topology information data of the current time section from the relational database, and then form the grid topology network to support the subsequent topology analysis, and synchronize and update in the graph database when the grid topology information changes. With the development and application of new energy [1], the new construction and migration of power stations, the expansion and transformation of power equipment, etc., the power grid topology information, especially the connection relationship of equipment, the number of equipment access, the spatial location information, etc. are also changing with time, especially the large-scale access of distributed power generation and diversified loads [2], which directly affects the original power grid topology network structure, and the current mode of online update of topology data after importing into the graph database can no longer meet the management needs of large-scale changes in power grid topology data in the current complex environment. Therefore, the time characteristics of the data need to be considered when managing and applying grid topology information. Taking photovoltaic energy as an example [3-6], large-scale household photovoltaic power supply is continuously connected, making the grid topology more and more large, and its intermittent power characteristics make it constantly change between load and power supply, which further aggravates the complexity of the power grid topology, and then brings certain inconvenience to the existing power grid topology analysis, reducing the real-time and accuracy of power grid topology analysis. Therefore, the current single grid topology data management method and mode without considering the time attribute can no longer meet the topology analysis needs of complex power grids in the new situation.

*yingzi108@126.com

## 2. TOPOLOGY DATA MANAGEMENT IDEAS

In order to solve the data management requirements of complex power grid topology in the new situation and overcome the modeling and storage problems of multi-time versions of grid topology data, this paper designs a multi-temporal power grid topology data management method.

In terms of data storage, Hadoop is comprehensively used to support multi-temporal grid topology data storage through the integration of Hbase and Spark. As a distributed database based on Hadoop Distributed File System (HDFS), HBase can be easily expanded horizontally by adding nodes, expanding the storage scale, and meeting the capacity requirements of massive data storage [7]. At the same time, HBase has the characteristics of multi-version storage, which can control data versions through timestamp labels, and support custom configuration of timestamps according to the needs of business scenarios to meet the needs of multi-temporal data storage. Through the integration of Spark and HBase, the use of Spark's memory-based large-scale fast parallel processing characteristics, the underlying storage system to achieve the support of high concurrency data storage and query requirements, Spark itself provides rich operators, reduces the complexity of data processing transformation, simplifies the development of data storage and access interfaces, and improves ease of use [8].

In terms of data model construction, based on graph data modeling theory [9], the equipment properties in the power grid topology and the connection relationship and information between the devices are analyzed, and a multi-temporal power grid topology data model including time labels is formed, including vertex collections and edge collections, to meet the model requirements of multi-time versions of power grid topology data. The GraphX framework [10] is introduced and developed twice, the multi-temporal grid topology data model is technically realized, and the integration of timestamp information is realized by reconstructing edge EdgeRDD and vertex VertexRDD, and Graph objects are generated in real time to support subsequent data storage and reading. At the same time, GraphX provides rich graph calculation and graph mining algorithms and corresponding interfaces, which can provide strong support for the subsequent analysis and application of multi-temporal grid topology data.

## 3. TOPOLOGY DATA MANAGEMENT DESIGN

### 3.1 Topology data management framework

Based on the traditional grid topology data management method, this paper integrates the time dimension data of the grid topology and constructs a multi-temporal grid topology data management framework, including grid topology data model management, grid topology data import, multi-temporal grid topology data storage and multi-temporal grid topology data access as shown in Figure 1.

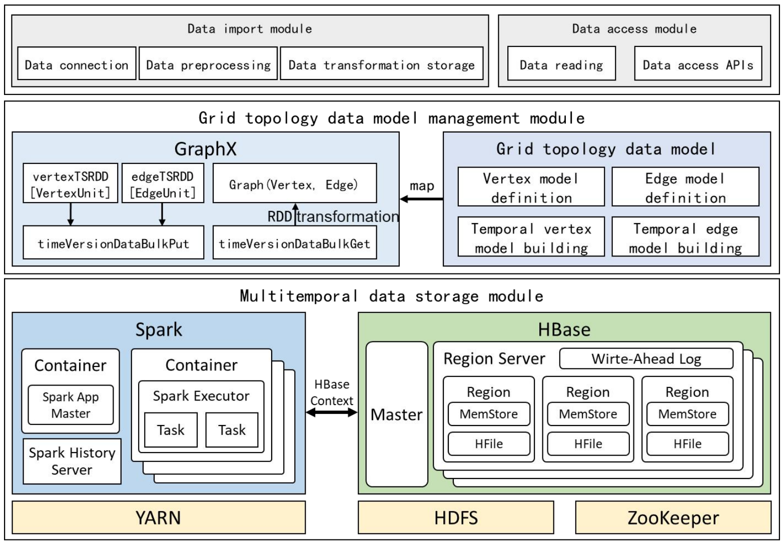


Figure 1. Multitemporal grid topology data management framework

## 3.2 Topology data model

In view of the fact that the original grid topology data model is difficult to support the management of multi-temporal grid topology data, this paper designs a grid topology data model that integrates time dimensions based on the basic theory of graph computing and graph data model, and realizes the model support of multi-temporal grid topology data management.

(1)Multi-temporal grid topology data model definition. Sort out the key equipment and components in the power grid topology, analyze the typical characteristics of the power grid topology, abstract the interconnected equipment and components such as power points, busbars, substations, circuit breakers, and load points in the power grid into vertices, and abstract the power lines connecting these vertices into edges to form vertex entities and edge entities, and define and describe them through entity definitions, entity properties, time labels, etc. respectively. Entity definitions mainly include entity numbers, entity names, entity space characteristics, etc., entity attributes mainly include entity types, entity information, power attributes, management information, entity descriptions, etc., and time labels describe the time version characteristics of entities in the form of timestamps. Table 1 shows the detailed model of vertex solid models, including vertex definitions, vertex properties, and timestamps. Table 2 describes the detailed model of edge entity models, including edge definitions, edge attributes, and timestamps.

Table 1. Vertex entities

| category | name | illustrate |
|---|---|---|
| Vertices definition | Id | numbering |
| | Name | name |
| | Coordinate_x | Longitude value x |
| | Coordinate_y | The latitude value y |
| Vertex properties | Type | Node type |
| | Info | Node information |
| | Electricity_attributes | Power properties |
| | Manager | administrator |
| | Ower | Affiliation |
| | Address | Installation address |
| | Desc | Node description |
| timestamp | Timestamp | Time version |

Table 2. Edge entities

| category | name | illustrate |
|---|---|---|
| Edge definition | Id | numbering |
| | Name | name |
| | Starting_node | Edge start |
| | Terminal_node | Edge end |
| Edge properties | Type | Edge type |
| | Info | Edge information |
| | Electricity_attributes | Power properties |
| | Manager | administrator |
| | Ower | Affiliation |
| | Length | Edge length |
| | Desc | Node description |
| timestamp | Timestamp | Time version |

(2)Design of multi-temporal grid topology data model based on GraphX. GraphX defines a Property Graph data model that creates graphs in vertices and edges. GraphX stores graph data in the form of Resilient Distributed Datasets(RDD) distributed on the nodes of the cluster, where the vertex RDD name corresponding to Vertics is VertexRDD, and the attributes include vertex ID and vertex attribute for storing vertex collections. The RDD name corresponding to Edges is EdgeRDD, and the attributes include source vertex ID, target vertex ID, and edge attribute to store edge collections. This paper designs a multi-temporal grid topology data model based on GraphX to support the management of multi-temporal grid topology data. The specific steps are as follows:

1）Define the vertex model: def VertexUnit(keys: Long, attribute:String, timestamp: Long)

2）Mapping the vertex model, mapping the ID of the vertex to keys, using the timestamp of the vertex as timestamp, building an attribute with other properties of the vertex, and "|" to form a property string with the expression:Name|Coordinate_x|Coordinate_y|Type|Info|Electricity_attributes|Manager|Ower|Address|Desc

3）Create a vertex RDD with a timestamp: vertexTSRDD[VertexUnit]

4）Define the edge model: def EdgeUnit(src: Long, dst: Long, attribute: String, timestamp: Long)

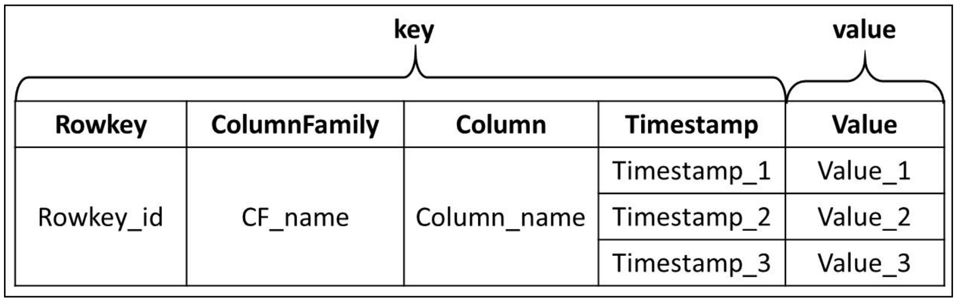
5）Perform mapping transformations of edge models, map Starting_node to src, map Terminal_node to dst, use the timestamp of the edge as timestamp, build an attribute with other attributes of the edge, and use the | A concatenation is made to form a property string with the expression:Id|Name|Type|Info|Electricity_attributes|Manager|Ower|Length|Desc

6）Create an edge RDD that contains the timestamp: edgeTSRDD[EdgeUnit]

# 4. DATA MANAGEMENT FUNCTION REALIZATION

## 4.1 Multitemporal data storage implementation

(1) The underlying storage system. Hbase divides the storage of data by column family, and sets several columns under the column family to achieve flexible data access, and uniquely distinguishes the data of a row by introducing Rowkey as the primary key. HBase uses regions for partitioned data storage, allocating the data of a large table to different regions based on different ranges of Rowkey, and each region is responsible for a certain range of data access and storage, providing access speed and reducing data latency. HBase's time version data storage is achieved through timestamps, by using different timestamps to identify the same Rowkey row corresponding to the different versions of the data, the same Rowkey data according to timestamp reverse order, users can read the old version of the data with the specified timestamp value. According to HBase, Key-Value(KV) mode is adopted, and the data storage mode of the underlying storage system of the multitemporal data storage module is (Rowkey, Family, Column, Timestamp) -> Value, where (Rowkey, Family, Column, Timestamp) is the key, which is used to identify the storage and reading of data, as shown in Figure 2.



| Rowkey | ColumnFamily | Column | Timestamp | Value |
|---|---|---|---|---|
| Rowkey_id | CF_name | Column_name | Timestamp_1 | Value_1 |
| | | | Timestamp_2 | Value_2 |
| | | | Timestamp_3 | Value_3 |

Figure 2. Key-value model of the multitemporal data storage module

(2) Data storage and access methods. The way to store and access is to combine Spark with HBase, and use the HBase-Spark connection tool class provided by HBase to support Spark to read and write HBase in batches in RDD by creating HBaseContext objects. The HBaseContext creation process is as follows:

```
val conf = new SparkConf().setAppName("SparkOnHBase").set("spark.master", "yarn")
```

```
.set("spark.submit.deployMode", "client")
val spark = SparkSession.builder().config(conf).getOrCreate()
val hbaseconf = HBaseConfiguration.create()
Val hbaseContext = new HBaseContext(spark.sparkContext, hbaseconf)
```

The HBaseContext object provides an interface for batch processing of HBase data, where bulkPut method is used to send put operations to HBase massively parallel, bulkDelete method is used to send delete operations to HBase massively parallel, and bulkGet method is used to send get operations to HBase massively parallel. Based on the interface provided by HBaseContext, secondary development and encapsulation are carried out to form a batch processing method that supports temporal data storage and reading, as shown in Table 3.

Table 3. Temporal data processing methods

| Method name | Underlying interface | illustrate |
|---|---|---|
| timeVersionDataBulkPut | bulkPut | Temporal data is stored in bulk |
| timeVersionDataBulkGet | bulkGet | Temporal data is read in batches |
| timeVersionDataBulkDelete | bulkDelete | Temporal data is deleted in bulk |

## 4.2 Power grid topology data management implementation

(1) GraphX-based data storage and reading. Storage methods include storage for vertex collections and storage for edge collections. Vertex storage is done by creating a vertex RDD (vertexTSRDD[VertexUnit]) containing a timestamp and calling the timeVersionDataBulkPut method to store the collection of vertices in the HBase data table. Edge storage is done by creating an edge RDD (edgeTSRDD[EdgeUnit]) containing timestamps, and calling the timeVersionDataBulkPut(rdd:RDD, rdd_table:String) method of the underlying storage to store the collection of edges in the specified table (rdd_table) in HBase. For different graph objects, vertex data RDDs and edge data RDDs are stored in different HBase tables, vertex data RDDs with different temporals of the same graph object will be stored in the same HBase table (such as vertex_table), and edge data RDDs with different temporal states will also be stored in the same HBase table (such as edge_table), and the data time version will be defined by timestamp. The process of data access is to obtain the RDD of vertex data and RDD of edge data by calling the underlying storage method, and then build a graph object (Graph) after conversion to provide access to the graph. The specific steps are: first call the timeVersionDataBulkGet(rdd_table:String, timestamp: Long) method to get vertexTSRDD[VertexUnit] and edgeTSRDD[EdgeUnit], respectively, and then perform Map conversion to vertexRDD[keys: Long, values:String] respectively. and edgeRDD[src: Long, dst: Long, values: String], and finally create a graph object (Graph) via Graph (VertexRDD, EdgeRDD).

(2) Data import. The data import module includes a data connection part, a data preprocessing part and a Data transformation storage part, as shown in Figure 3.

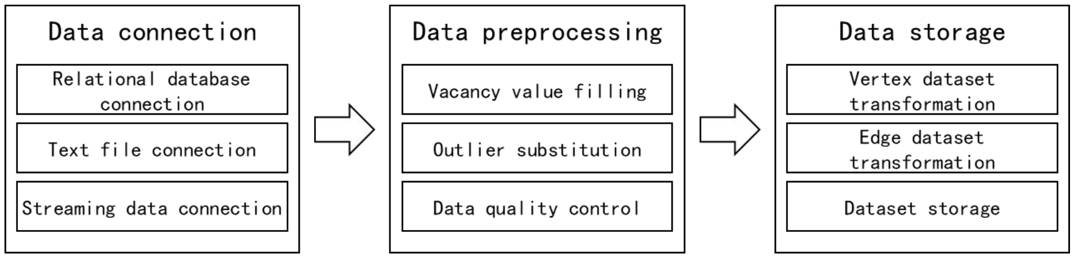


Figure 3. Flow chart of the data import module

Data connection. Based on Spark components, data connection modules for different data storage forms such as relational databases, non-relational databases, text data, and data streams have been developed, JDBC is used for database connection, text data is directly read, and SparkStreaming is used for data streaming. The main methods of JDBC-based database connections are as follows:

```
Val jdbcDF = spark.read.format("jdbc").option("url", "jdbc:xxx:xxxserver").option("dbtable", "schema
```

.tablename").option("user", "username").option("password", "password").load()

Data preprocessing. Based on the DataFrame formed after data connection, using the rich DataFrame operator provided by Spark, retrieve duplicate data and deduplicate them, find missing data and fill them according to the average value of neighboring data or set default values, analyze abnormal data and replace outliers according to the change trend of neighboring data. Then, the DataFrame.rdd method is used to convert the DataFrame into RDD, and finally the vertex RDD dataset and edge RDD dataset that meet the data quality requirements are formed.

Data transformation storage. After obtaining the preprocessed RDD dataset, the multitemporal grid topology data model is applied to add timestamp information to the RDD dataset based on GraphX, in which the structure of the vertex RDD is converted from (keys: Long, attribute:String) to (keys: Long, attribute:String, timestamp: Long), and the structure of the edge RDD is converted by (src: Long, dst: Long, attribute: string) is converted to (src: Long, dst: Long, attribute: String, timestamp: Long) to form vertex datasets and edge datasets containing temporal attributes, and then stored into the multitemporal data storage system using the data storage interface provided by the data model management module.

(3) Data access. Based on GraphX and multi-temporal grid topology data model, develop a data reader, according to the given power topology object and time version, read the specified temporal grid topology data through the data access interface provided by the data model management module, and provide it in the form of grid topology diagram (Graph), and the attribute data of vertices and edges of the grid topology map is restored and read by parsing the attribute strings of vertices and edges.

## 5. CONCLUSION

This paper analyzes the management requirements of existing grid topology data in multi-temporality, designs a grid topology data model covering time information based on big data and graph computing technology, and proposes a multitemporal grid topology management method that integrates time dimensions. This method realizes the whole life cycle management of multi-temporal grid topology data, provides a new solution for improving the efficiency of grid topology data management with multi-time version attributes, and strongly supports the subsequent analysis and application of multi-temporal grid topology data.

## ACKNOWLEDGMENTS

This work is supported by the Science and Technology Project of State Grid Corporation of China (Research on deep data fusion and resource sharing technology of new distribution network, No. 5400-202258431A-2-0-ZN).

## REFERENCES

[1] Gandhi, O., Rodríguez-Gallegos, C. D., Zhang, W., Srinivasan, D., Reindl, T., Economic and technical analysis of reactive power provision from distributed energy resources in microgrids[J]. Applied Energy, 210, 827-841 (2018).

[2] Bai H., Wu K. H., Sun C. B., Kong W. Y., Qiu Y. L., Research on adaptability of distribution network access standard under high proportion distributed power supply and diversified load conditions[J]. Electrotechnical Application, 38(05):81-88 (2019). (In Chinese)

[3] Chang, J. L., Research on the Influence of Distributed Photovoltaic Power Access to the Distributed Network and Grid-connected Planning Study [D]. School of Electrical Engineering and Electronics (2016). (In Chinese)

[4] Kai Qiao,Qiao Kai,Zhang Guomin,Liu Ningchao,Man Hongwei. Analysis of Abnormal Low-voltage Line Loss in the Transformer Area Caused by Distributed Photovoltaic Access[J]. Journal of Physics: Conference Series, (Vol. 1654, No. 1, p. 012091). IOP Publishing (2020).

[5] Singh, M., Protection coordination in distribution systems with and without distributed energy resources- a review[J]. Protection and Control of Modern Power Systems, 2(1): 1-17 (2017).

[6] Chandraratne, C., Naayagi Ramasamy, T., Logenthiran, T., Panda, G., Adaptive Protection for Microgrid with Distributed Energy Resources[J]. Electronics, 9(11), 1959 (2020).

[7] Zhu, L., Cai, X., Le, Y., Research on Performance Optimization for Power Big Data Storage based on HBase[J]. Journal of Physics: Conference Series,(Vol. 2033, No. 1, p. 012181). IOP Publishing (2021).

[8] Park K., Peng L., A development of LDA topic association systems based on spark-hadoop framework[J]. Journal of Information Processing Systems, 14(1):140-149 (2018).

[9] Ma, Y. s., Wu, Z. G., Modeling and analysis of big data for power grid based on Neo4j[J]. Advanced Technology of Electrical Engineering and Energy, 35(02):24-30 (2016). (In Chinese)

[10] Dharavath, R., Arora, N. S., Spark's GraphX-based link prediction for social communication using triangle counting[J]. Social Network Analysis and Mining, 9(1):1-12 (2019).