An Orm-Based Method and Platform to Uniformly Access Heterogeneous Data in China National Earthquake Precursor Network

Chen Wang, Yuntian Teng*,Xiaomei Wang, Xiaoyong Fan and Jiemei Ma
Institute of Geophysics, China Earthquake Administration, Beijing100081, China.
Email: xiaomei7978@163.com

Keywords: Object relation map, heterogeneity, access data uniformly, national earthquake precursor network

Abstract: The distribution and heterogeneity of data sources in a national earthquake precursor network make uniform data access difficult. To resolve this problem, we propose a uniform data access method based on object relation mapping, and then design a uniform data access platform. The proposed uniform data access method adopts a view of the data in three layers with two layers of mapping. The former includes the application subject view (ASV), a uniform access view (UAV) and a local data view (LDV), and the latter includes mapping between the application subject view (ASV) and the uniform data access view (UDAV), and mapping between the uniform data access view (UDAV) and the original data view (ODV). This design has potential practical applications which validate that a platform based on the proposed method effectively serving the China National Earthquake Precursor Network.

1 INTRODUCTION

In China National Earthquake Precursor Network, regionally distributed stations are responsible for the collection and the storage of basic precursor data, which enables the current data source distribution situation, in which the data have diversity, heterogeneity and autonomy properties. These properties make it difficult to manage the data uniformly, as well as share the data concurrently. Consequently, this situation impedes further development of businesses carrying out earthquake precursors such as earthquake observing, earthquake broadcasting and epicenter analysis(Wang et al., 2011). Therefore, intensive investigations of uniform access to earthquake data are needed.

Based on the above reasons, we propose a UDAM based on ORM, and design a uniform data access platform. Our proposed method realizes uniform access based on the creation of three layer views and mapping between pairs of layers. The three layer views include the ASV, the UAV and the LDV. The two-layer mapping consists of mapping between the ASV and the UDAV, also the UDAV and the ODV.The practical applications validate that the platform based on the proposed method can effectively serve national earthquake precursor network in China.

2 RELATED WORK

With the ongoing development of computer and wireless transmission technologies, characteristics of data distribution and heterogeneity are becoming increasingly significant. Accordingly, data integration issues have aroused increasing research interest because they facilitate the application-to-application exchange of standard business documents between different subjects(Andrea et al., 2013; Len et al., 2013).

Traditional data integration studies mainly focus on how to eliminate data distributed effects and heterogeneity; they usually accomplish this by using a certain persistent integration method (Lnmon, 1996; Guo et al., 2008). Recently, the view-based data integration method (VBDIM) has become increasingly popular. In this method, each data source creates a uniform model based on local information integration rules; a local information

Wang, C., Teng, Y., Wang, X., Fan, X. and Ma, J.
An Orm-Based Method and Platform to Uniformly Access Heterogeneous Data in China National Earthquake Precursor Network.
In *Proceedings of the International Workshop on Environment and Geoscience (IWEG 2018)*, pages 434-439
ISBN: 978-989-758-342-1

view (LIV) is then developed. Finally a global information view (GIV) is constructed based on this view. The entire application program conducts information access operations based on this global view. Compared with other methods, the view-based method (VBM) has advantages that include good extendibility and dynamic properties.

It has been a common belief that creating XML vocabularies was sufficient to achieve data interoperability, yet this assumption goes far beyond reality. XML by itself does not guarantee that XML expressed business information exchanged in the span of business processes across different enterprises will be understood equally well by all systems. This is because the XML syntax only creating markup languages used as metadata, it does not address how the underlying business information must be modeled, named and structured. Semantics come to cover this gap by attaching meaning to data in a structured and technical way that both humans and machines can understand and process(Lampathaki et al., 2009; Nurmilaakso et al., 2004).

Object relational mapping (ORM) is a popular uniform data method, which provides a method and mechanism for object-oriented systems to hold their long-term data safely in a database, while maintaining transactional control over it, and having it expressed in program objects when needed(Elizabeth , 2008).

3 ORM-BASED METHOD USING THREE LAYER VIEWS AND TWO-LAYER MAPPING

3.1 Layers

To provide a simple method for accessing certain business data, screening for heterogeneity, and decreasing the coupling between the database structure and the application program, we identify three data layers, which include the ASV, a UAV and a LDV (Figure 1).

The functions of the each view are as follows.

(1) Application Subject View. This view is a kind of virtual view for various kinds of application demands, including the data collection, data exchange and data application views. The application subject view screens for difference in the underlying database, which makes database access more effective.

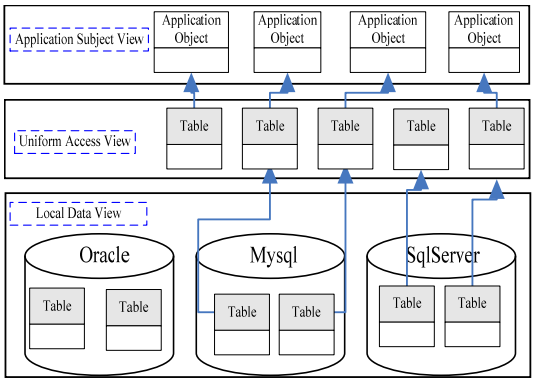

Figure 1: Diagram of the three layer views.

(2) Uniform Access View. This refers to the virtual table used to represent the data structure in the database. A uniform access view acclimatizes changes in database products and structure. The uniform access view includes original views of the national center, the subject center, and the regional center. It is represented by a virtual table and virtual fields; and, in this case, the application program does not need to be concerned with the actual storage manner. As a result, this view can increase integration and extension abilities as well as flexibility.

(3) Local Data View.This view is a kind of lowest level view for various kinds of database, which could access measuring instruments easily.

3.2 Rules for the Two-Layer Mapping

The two-layer mapping undertaken involves mapping between the ASV and the UDAV, the UDAV and the ODV(Figure 2). We have designed and implemented two mapping rules for the two-layer mapping: an application subject view mapping rule and a local data source mapping rule.

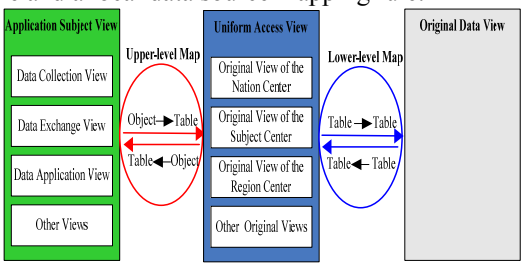

Figure 2: Two-layer mapping.

3.2.1 Application Subject View Mapping Rule

The application subject view mapping rule is used to map from the application subject view to the uniform access view; i.e., to mapping the object and property in the application subject view to a virtual table and field in the uniform access view.

When mapping the application subject view to the uniform access view, the application subject view maps the table to the corresponding view by creating a database view. Additionally, the application subject view accesses the information by reading the corresponding field in the uniform access view. Accessing the uniform view is different from accessing the ordinary table because it is not a major feature in the view. For example, there is an object (*O*) that has three properties: *x, y,* and *z*, i.e., *O(x, y, z)*. There are two virtual tables *A* and *B* in the uniform access view, where *A* has *N* fields: a_1, $a_2 \ldots\ldots a_n$, and *B* has *M* fields: b_1, $b_2 \ldots\ldots b_m$. Then we will have the following mapping results: $O.x \rightarrow A.a_1$; $O.y \rightarrow A.a_2$; $O.z \rightarrow B.b_2$.

From the above mapping results, we can see that an object could correspond to several tables. So once the application design corrects an object, it can operate several tables through this object; the data access then will be more convenient and effective.

3.2.2 Local Data Source Mapping Rule

The local data source mapping rule is used for mapping from the uniform access view to the local original data view; i.e., to mapping a virtual table and field from the uniform access view to the actual table and field in the local origin data view. For example, for a virtual table A that has N fields: A_1, A_2, … AN, there is an actual table a that has M fields: a_1, a_2, … am. We then will get the following mapping results: $A.A_1 \rightarrow a.a_1$; $A.A_2 \rightarrow a.a_4$; $A.A_N \rightarrow a.a_M$.

From the above mapping results, we can see that the mapping relationship between the tables in the uniform access view and the tables in the original data view is a one-to-one relationship. However, the fields in the two views do not have a one-to-one relationship. The uniform access view can hide some protected fields in the original data from the users.

3.3 Information Description Methods

we consider the information description methods, which include descriptions of the data source, the application view mapping rules and the local data source mapping rules.

3.3.1 Data Source Description Method

In this paper, we adopt a five-fold description of the data source.

DataSource= {DBName, URL, Username, Password, Driver},

where *DBName* is the name of the data source, which is the unique identifier for the underlying data source; *URL* is the universal resource locator access address of the data source; Username and Password are the name and the password used to access the data source, respectively; Driver is the name of the driver program that needs to be loaded to access the data source. An example description is shown in Figure 3.

```
<DataSourceInfoList>
    < DBSource name=”ORACLE”>
        <URL value="
    jdbc:oracle:thin:@localhost:1521:PDBQZ
    “></URL>
        <USER value=" qzdata”></USER>
        <PSW value=" dataqz”></PSW>
        <DRIVER value="
    oracle.jdbc.driver.OracleDriver”></
    DRIVER >
    </ DBSource >
</DataSourceInfoList>
```

Figure 3: Example description of the data source

3.3.2 Application View Mapping Rules Description Method

For the application view mapping rules, we adopt a three-fold description.

AppObject={ObjectName, ReferView, PropertyList}

where *ObjectName* is the name of the unique application object; *ReferView* is the name of the virtual table referred to by the application object in the middle logic view; *PropertyList* contains the names of the fields included in the application

object, and each field is represented by another three-fold definition:

Property= {PropertyName, ReferViewName, ViewFieldName}

where *PropertyName* is the name of the property of the application object; *ReferViewName* is the name of the virtual table according to the object property; *ViewFieldName* is the property name in the virtual name according to the property.Figure 4 gives an example of this description method.

```
<mappingList>
        <Object name="TEST1">
                <ReferView
ViewName="V_TEST1" />
                <Property name="SID"
ViewName="V_TEST1" FieldName="V_SID" />
                <Property
name="SNAME" ViewName="V_TEST1"
FieldName="V_SNAME" />
                <Property name="AGE"
ViewName="V_TEST1" FieldName="V_AGE"
/>
                <Property name="AA"
ViewName="V_TEST1" FieldName="V_AA" />
                <Property name="BB"
ViewName="V_TEST1" FieldName="V_BB" />
        </Object>
</mappingList>
```

Figure 4: Example description of the application view mapping rule.

3.3.3 Local Data Source Mapping Rule Description Method.

We adopt a three-fold description of the local data source mapping rule:

View={ ViewName，ReferTable，FieldList}

where *ViewName* is the name of the virtual table in the logical view, which is unique in the logical view; *ReferTable* is the name of the underlying table referred to in the virtual table; *FieldList* is a set of the table fields included in the virtual table, with each table field being expressed by a three fold description:

Field ={ FieldName, ReferTableName ， TableColumnName}

where, *FieldName* is the name of the field in the virtual table; *ReferTableName* is the corresponding name of the field in the underlying data table; *TableColumnName* is the corresponding property name of this field in the underlying data table. Figure 5 gives an example of this description method.

```
<mappingList>
        <View name="V_TEST1">
                <ReferTable
TableName="TEST" DBname=”ORACLE”/>
                <Field name="V_SNAME"
TableName="TEST" ColumnName="SNAME"
/>
                <Field name="V_AGE"
TableName="TEST" ColumnName="AGE" />
                <Field name="V_AA"
TableName="TEST" ColumnName="AA" />
                <Field name="V_BB"
TableName="TEST" ColumnName="BB" />
        </View>
</mappingList>
```

Figure 5: Example description of local data source mapping.

4 PLATFORM FOR UNIFORM ACCESS TO DATA

Based on the above analysis, we designed a uniform data access platform based on the three layer views and two-layer mapping. The main idea of this platform is to create a globally logical view for all heterogeneous data coming from different data sources and to screen the heterogeneity of the data. Then, based on this uniform logical view, each application program can design a different application object according to its demand; this can result in the addition, deletion and revised operation of a certain data object. The architecture of our platform is shown in Figure 6.

The data access platform consists of four discrete parts: mapping management, execution engine, application program interface, and database access. The function of the platform is the encapsulation of the underlying database and files, which provides data access service for all kinds of application programs.

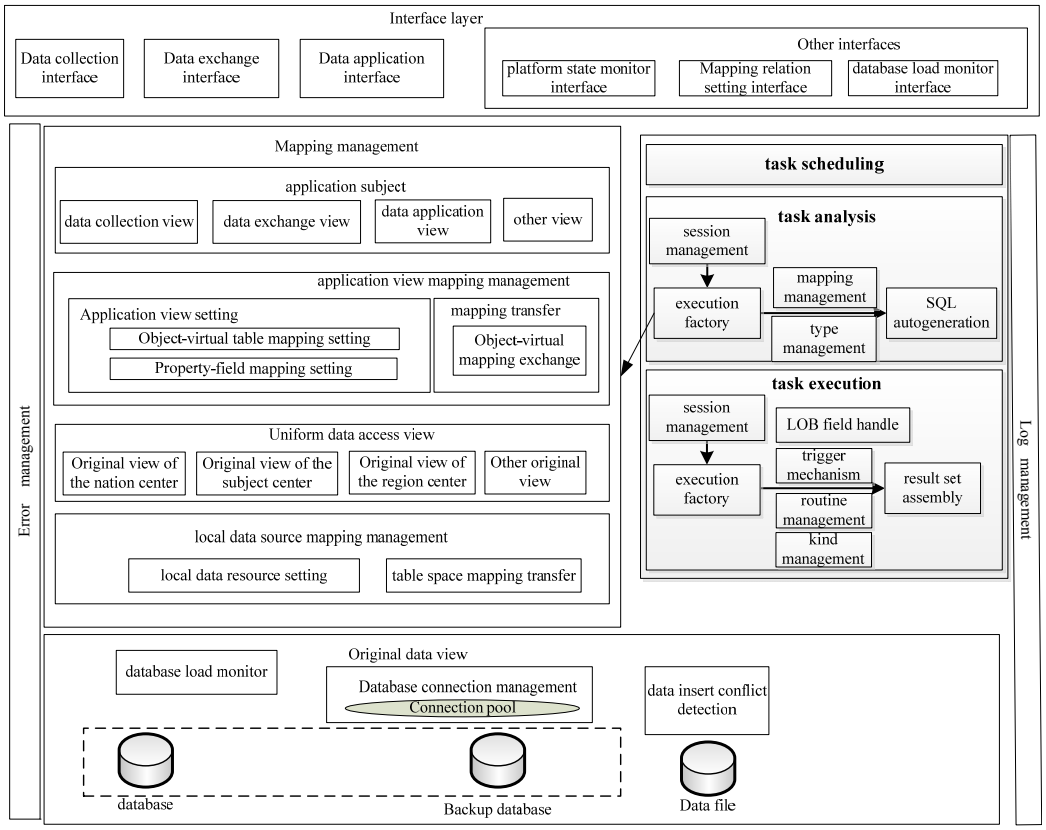

Figure 6: Platform architecture.

4.1 Task Execution Engine

Based on the mapping view structure, the main functions of the task execution engine are to analyze and execute the submitted tasks.

When the engine receives a task, the scheduling module first schedules the task to be executed, and then inserts the task into the task list.

The task analysis module first analyzes the submitted task by assessing related information, including the task type and the operation object. Based on the uniform view generated by the mapping and type management, the task analysis module then translates this task into structured query language (SQL) commands that can be recognized by the database. Finally, these commands are submitted to the task execution module.

When the task execution module receives the information, the execution factory first creates a corresponding execution class based on the different task types; then, it executes the SQL commands. During execution, it is responsible for managing exceptions, logs, sessions and big objects (CLOB fields), and so on. Finally, the module encapsulates all the execution results into a uniform result set and returns.

4.2 Application Programming Interfaces

The application programming interfaces can be divided into three categories: exchange application interfaces, collection application interfaces and the data application interfaces. The main functions of these interfaces are as follows.

4.2.1 Exchange Application Interface

This interface is responsible for the following work: data and information exchange between national, regional and local centers; consistency checking and issuing the error warning during the exchange; and managing the parameters used in the exchange

strategy. In the exchange module, the regional center forwards the data to the national center, and the national center distributes the data to the subject center, which means that the data will generally have multiple copies. Therefore, the exchange method must maintain the consistency of the data and be responsible for checking data versions. The data must be unique for the whole platform; therefore, the exchange module must be responsible for change management.

4.2.2 Collection Application Interface

The data collection module is responsible for the following work: collecting original observational data coming from non-IP instruments, IP instruments and the artificial observation instruments, collecting data sourced from instrument logs and the event; checking the data validation and data storage; collecting and managing the metadata and the basic data. Figure 7 shows the functions of the collection application interface.

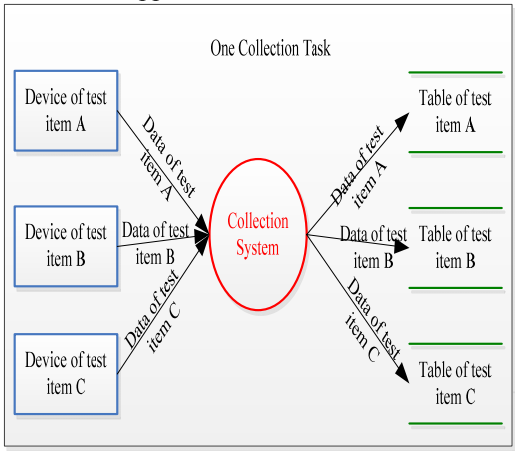

Figure 7: Functions of the collection application interface.

4.2.3 Data Application Interface

This interface is responsible for observing the state of the platform, and managing information on the nodes, stations and system basics. The database is responsible for the management of the data version and the user authority, and can provide different users with different versions and authorities with different functions. This interface is also responsible for recording user operations, such as user login.

5 CONCLUSIONS

To deal with the difficulties in uniformly accessing data under circumstances affected by the data distribution and heterogeneity properties of the data source, we propose a uniform data access method based on object relation mapping(ORM), and have designed a platform based on this method. We give the detail of the platform architecture and function implementation method, and also talk about the task execution engine and application programming interfaces.This design has potential practical applications serving the China National Earthquake Precursor Network and play a role in the future.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 41404141) and the National Key Scientific Instrument and Equipment Development Project of China (Grant No. 2014YQ100817-2).

REFERENCES

Andrea C, *et al.* 2013 Data Integration under Integrity Constraints. *Seminal Contributions to Information Systems Engineering* 335-352

Elizabeth (Betty) O'Neil. 2008 Object/Relational Mapping 2008: Hibernate and the Entity Data Model (EDM). *Proceedings of SIGMOD* **08** 1351-1356

Guo H M, *et al.* 2008 Heterogeneous data integration and coordination in national geology grid. *Journal of Beijing University of Aeronautics and Astronautics* **34(2)** 179-182

Lampathaki F , *et al.* 2009 Business to business interoperability: A current review of XML data integration standards. *Computer Standards & Interfaces* **31** 1045-1055

Len W , *et al.* 2013 Incorporating Recovery from Failures into a Data Integration Benchmark. *Lecture Notes in Computer Science* **7755** 21-33

Lnmon W H 1996 Building the Data warehouse, Second Edition. *John Wiley & Sons Ine* 27-55

Nurmilaakso J M, *et al.* 2004 A review of XML-based supply-chain integration, *Producti on Planning and Control* **15** 608–621

Wang C ,*et al.* 2011 Research on Coordinative Fusion and Uniform Integration for Earthquake Precursor Hybrid Data. *Computer & Digital Engineering* **39(12)** 44-46