Available online at www.sciencedirect.com

SciVerse ScienceDirect

AASRI Procedia 1 (2012) 33 – 39

AASRI Procedia

www.elsevier.com/locate/procedia

2012 AASRI Conference on Computational Intelligence and Bioinformatics

The Realization of CIM Data Access Mechanism Based on FastDB*

Cao yulin[a,b],Wang Xiaoming[a]

[a]*School of Computer Science, Shaanxi Normal University, Xi' an Shaanxi, 710062, China*
[b]*School of Computer Science, Qinghai Normal University, Xining*, ***,*** *810008*,***China***

Abstract

FastDB is an open source real-time database with characteristics of high efficiency and the object oriented, supporting the various object-oriented, running on different operating system platform, convenient for realizing CIM data access mechanism. Through the analysis of CIM data access mechanism, this paper states the methods of the resources location, data query, data describing in the process of data access. The realization code, the function and exception handling methods of CIM data access interface ResourceIDService, ResourceQueryService code based on FastDB are given. FastDB can effectively use CPU cycles and memory, to some extent, meet real-time requirements of the system on the data, and increase the development, maintenance efficiency of the whole system in the different industries.

Keywords: FastDB;DAF;Data access;CIM;Interface

1. Introduction

With the rapid development of computer technology, database system is everywhere, and although relational database is powerful, open well, the general application for the realization mechanism is difficult to satisfy EMS application for real-time/quasi real time request. IEC 61970 standardization technology is a kind of criteria achieving the different manufacturer, different control center Information sharing [1] and system integration, including that CIM (Common Information Model, CIM) describes data of exchange and CIS

Corresponding author. Tel.: 15111716327; fax:.
E-mail address: caoyulin@126.com

explains how to exchange CIM data. CIS gives detailed descriptions of the information exchange according to the standard way among the applications and interface of public data access, insertion and using among the application systems. In recent years, the interoperability tests have been operated several times based on the model CIM XML at home and abroad , and has been successfully used in the actual system [2]. But with the continuously increasing of system size and application complexity, these are difficult to meet the requirements of various systems and the system function expansion exists a lot of difficulties. At the same time the CIM model is a typical object-oriented model, in the development process, after detailed comparison research, using open source real-time database FastDB as development prototype system. FastDB is the real-time database of the object oriented by using C/C + + [3], supporting SQL query, Web page query, export and import of XML data files, online backup of transaction process and automatic recovery and so on, running on Win32 / Unix/Linux and other kinds of operating system platform, which has brought convenience for the realization of the CIM data access mechanism.

2. Data access facilities

Data Access facilities (Data Access Facility, DAF) is one of the main contents of IEC61970, in order to improve the EMS application and other systems, and application of collaborative work ability as the goal, which accesses CIM public data for different suppliers to provide a public use application program interface (API) and public service mechanism [4]. The design concept of the DAF is to provide a simple application easy API. DAF is one of the most basic CIS interface standard, including DAFDescriptions, DAFIdentifiers, DAFQuery, DAFEvents module, the core of the module is corresponding interface. For the external system of internal system CIM data acquisition system, DAF provides enough interface in the function [5]. This paper only uses DAFDescriptions. Idl, DAFQuery. Idl and DAFIdentifiers. Idl .

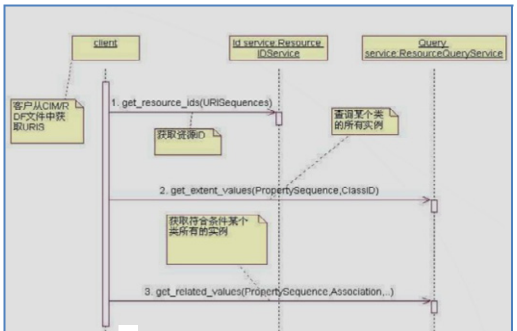
DAF makes use of resources, properties, and values to inquiry the formulation results. A resource is anything of an unmistakable identity. Typically, That a unified resource identifier (URI) pluses an optional fragment identifier may label any resource. But, in the DAF inquires and the search results, a more compact ResourceID of URI should be used. A property is the aspect a resource may describe. When appearing in inquires, nature expresses through the type PropertyID, and nature itself is a resource, so PropertyID is defined as a ResourceID. A property has a domain (resource collection) and a range (value collection). The relationship between the resources through some properties is created, and the nature of the domain and range is also a collection of resource.

In the resource description framework, the value is a basic data unit, and can be a string, integer or resource reference. In the DAF, value type SimpleValue, is actually a combination of several basic types and ResourceID.

3. The realization of CIM data access

3.1 Data access process

DAF provides a series of standard data access service of orientation, inquiry, description and inform event and so on, and , through this public data access interface, realizes the integration between application system. DAF mainly includes positioning, and inquiry, describing services The below figure describes a basic DAF data access process. First clients input a series of string, and the server will return to the string in the database of location. Because the string is based on URI format, the access is a standard form, namely resources positioning. And then in the query service, the information users input is a ResourceID positioning and PropertyIDSequence obtained in resource orientation, which is the corresponding with ResourceDescriptione, such as data query. At the last, ResourceDescriptione in query is analyzed and then get the needed data, such as data description.


client
Id service:Resource IDService
Query service:ResourceQueryService
客户从CIM/RDF文件中获取URIS
1. get_resource_ids(URISequences)
获取资源ID
查询某个类的所有实例
2. get_extent_values(PropertySequence,ClassID)
获取符合条件某个类所有的实例
3. get_related_values(PropertySequence,Association,..)

Data access process maps

3.2 The realization of ResourceIDService interface

ResourceIDService provides conversion service between the ResourceID and URI. The goal of ResourceIDService design is to reduce a price of repeatedly analyzing, comparing and searching for URI.

1) get_resource_ids () method

The input of Get_resource_ids () method is a standard URI sequence, and its output is a ResourceID sequence. Function prototype is:

```
DAFIdentifiers::DAFIdentifiers::ResourceIDSequence*get_resource_ids(
const DAFIdentifiers::URISequence & uris)
throw (DAFIdentifiers::LookupError);
```

Parameter uris is a standard URI sequence, and its return value is a ResourceID sequence.

Function: CIM/CIS server analyzes URI sequence, and the elements will be transformed into the corresponding ResourceID. Finally, return sequence in the same position is assigned. For no analytical URI, corresponding ResourceID is empty (Container and Fragment are 0). The realization HTML code is as follows:

```
dbCursor<UrlToname> UrlTonam_Cursor; // Define the UrlTonam_Cursor cursor of UrlToname table
char  name[256]; // Define the string array, used for storage of resources to inquiry
dbQuery q=”name=”,name; // Find the resource identifier portion after uris colon;Copy this part of the
  string into //the string array name;
If (UrlTonam_Cursor.select(q)>0) // If the corresponding ResourceID is found in UrlToname table
return UrlTonam_Cursor->fragment; // Return result
    else{
    }// Not found
```

In order to conveniently implemente ResourceIDService interface, in the FastDB, a extra storage table UrlToname stores the mapping of resource ResourceID to resource. With this form and FastDB strong inquiry function, the realization of the get_resource_ids method will be simple and clear, and its analytical process isdescribed as the program.

2) get_uris () method

Get_uris method is the inverse process of the get_resource_ids method. A standard ResourceID sequence is input , and then a URI sequence is output. Function prototype is:

```
DAFIdentifiers::URISequence*ResourceID ServiceImpl::get_uris(const DAFIdentifiers::
    ResourceIDSequence& _ids)
    throw (DAFIdentifiers::LookupError);
```

Function: CIM/CIS server analyzes ResourceID sequence, and the elements will be transformed into the corresponding URI. Finally, return sequence in the same position is assigned. For no analytical ResouceID, corresponding position returns to an empty string. When this method errors in inquires (class_id is found, but not a class (or form) or check error), an DAFIdentifiers: : LookupError shows abnormal.

The realization HTML code is as follows:

```
DAFIdentifiers::URISequence  uris; // Definite the URI sequence to return
dbCursor<UrlToname> UrlTonam_Cursor; Definition the UrlTonam_Cursor cursor of UrlToname table
dbQuery q="+fragment=", _ids;  //_ids is a standard  ResourceID sequence transferred
 if(UrlTonam_Cursor.select(q)>0) // If the corresponding resource ResourceID name is found in UrlToname
  list
    {
   if(Resources is class of)
        uris="http:"+ "#"+ "C"+":" +"Resources name";
    else if(Resources for attribute)
            uris="http:"+"#"+"P"+":"+"Resources name";
  else // Resources for example
    uris="http:"+"#"+"I"+":"+"Resources name";
      return uris;
      }
```

3.3 *ResourceQueryService interface*

ResourceQueryService interface DAF is a central part of the DAF that defines basic resources service inquires interface. In ResouceQueryServiceImpl class the interface is implemented.

1)interface definition

```
  interface ResourceQueryService{
  ResourceDescription get_values(
        in ResourceID resource,
        in PropertySequence properties)
  raises( UnknownResouce, QueryError );
  ResouceDescriptionIterator get_extent_values(
        in PropertySequence properties,
        in ClassID class_id)
    raises(UnknownResource, QueryError);
    ResourceDescriptionIterator get_related_values(
        in PropertySequence properties,
        in Association association,
        in ResourceID source)
      raises(UnknownResource,UnknownAssociation, QueryError);
    ResourceDescriptionIterator get_descentdent_values(
        in PropertySequence propertiesm,
        in AssociationSequence path,
        in ResourceID parent,
        out AssociationSequence tail)
    raises(UnknownResource, QueryError);
    };
```

2) get_values () method

Query returns to a given ResoureID resource description, which is one of the most simple data request. If the terminal and name of a switch, function prototype is:

```
 DAFDescriptions::ResourceDescription* ResourceQueryServiceImpl::get_values(
 const DAFIdentifiers::ResourceID& _resource,
 const DAFIdentifiers::ResourceIDSequence& _properties)
 throw(CORBA::SystemException,
 DAFQuery::UnknownResource,
 DAFQuery::QueryError);
```

The parameter resource is ResourceID of the resource record to inquiry; Properties are the attribute logo sequence to query, and returns to ResourceDescription.

Data service throws UnknownResource abnormal when no finding input resources; and throws QueryError abnormal when Database query fails.

The realization of the get_values method, only according to the ResourceID and given resource attribute ResourceID sequence in the form of inquires directly Link, returns to the results.

The realization HTML code is as follows:

```
 DbCursor<Link> Link_cursor; // Define the cursor of Link table
 DAFDescriptions::ResourceDescription rd； //Resource description object to return
 Dbquery q="objectID=",_resource;
      //Write the query classes， instantiate query object, the parameter _resource is transferred ResourceID
  If(Link_cursor.select(q)>0)  //Find all the objecID and query resource record ResourceID
     {
    rd.iD= Link_cursor->objectID; //Assign the object objiectID to the resource object resourceID
    do {//According to transferred attribute sequence identify, initialize each sequence attribute value of
//resource  description objiect attribute value
 for(CORBA::ULong I=0; I<=_properties->length; I++)
    {       if(Link_cursor->propertyID==_properties[I])
          {
                 rd..value[I].property=_properties[I];
                 rd.value[I].value= Link_cursor->value;
           }
          }
  }while(Link_cursor.next())
  return rd; //Return to resource description
  }
  Else
     {//No finding
     }
```

3) get_extent_values () method

When get_extend_values method gives a ClassID, inquiry returns to all the resources description of a given ClassID. Function prototype is:

```
DAFDescriptions::ResourceDescriptionIterator_ptr  ResourceQueryServiceImpl::get_extent_values(
ConstDAFIdentifiers::ResourceIDSequence&_properties
const DAFIdentifiers::ResourceID& _class_id)
throw(CORBA::SystemException,
DAFQuery::UnknownResource, DAFQuery::QueryError);
```

Parameter PropertySequence properties is the attributelogo sequence of query. ClassID class_id is the ResourceID of inquiry, and returns a iterators (ResourceDescriptionIterator).

If class_id isn’t found within the scope of the data dictionary, an UnknownResource is thrown abnormal. If class_id is found, but not a class (or form) or check error, an QueryError is thrown abnormal.

In order to realize get_extent_values () method, first according to a given type of ResourceID, the ResourceID of the all instances are returned; then the table Link, by example ResourceID and the given attributes of the query results returns to ResourceID sequence.

The realization HTML code is as follows:

```
/*In order to query in the link table and instance table, define two table cursors*/
DbCursor< Instance > Instance _cursor;
DbCursor<Link> Link_cursor;
vector<unsigned long long> v; //Define vector，which is used for temporary storge of all ResourceID of query class
DAFDescriptions::ResourceDescription rd;
ResourceDescriptionIterator iter  //Define the iterator to return
char objectID[256];
Dbquery q1=”objectID=”, _class_id;
Dbquery q2=”objectID=”, objectID;
If(instance_cursor.select(q1)>0)  //According to class_id，//find all instances in class Instance
{
 do{ v.push_back(instance_cursor->objectID)  //Put all such instances in the class instance table into a vector
    }while (instance_cursor.next())
          while(!v.empty())
          {  objectID=v.begin(); //The instance resourceId in  //the vector is assigned to objected of q2
           if(Link_cursor.select(q2)>0){
     rd.iD= Link_cursor->objectID;//ObjectID is assigned to resource description object resourceID to return
 for(CORBA::ULong I=0; I<=_properties->length; I++)
          {           if(Link_cursor->propertyID==_properties[I])
        {
     rd..value[I].property=_properties[I];
                   rd.value[I].value= Link_cursor->value;
           }
          }
          v.pop_back();
    }
   }
   iter = &rds;
 return iter;  //Return to an interator
}
else
 {  //No finding
 }
```

4) get_related_values () method

First inquiry the correlation table, and return to ResourceID sequence of the given instances according to instance and relation ResourceID. Then check attribute list, and based on instance ResourceID sequence and a given property ResourceID sequence, return to several attribute values of all such instances, specific method is similar to section 3) .

4. Conclusions

Various industries often need operations of fast storage, query, update, and delete for a large amount data, particularly outstanding in some energy management and other large enterprise applications. In these cases, the high performance real-time database should be chosen. Through the in-depth analysis and comparison, this paper chooses real-time database after FastDB as a prototype system and analyzes the realization process of the CIM data access. Detailed methods of realization of the main interface function, code and exception handling are given. At present there are many real-time databases of the strong performance, but the price is expensive, in not large system construction, the free open source real-time database can achieve good effect, especially in equipments of physical memory enough big.

According to various the specific application of the demand of energy management system in the future, combining with FastDB open source database, and gradually improving various functions and performance, we will make all-round analysis and extended application for the CIM data access mechanism, so that the CIM mechanism has better inclusiveness and versatility.

Acknowledgements

This paper was from National Natural Science Foundation of China (Grant No. 61065009) funding.

References

[1]Wang Kun, Fan Xiaoli, Pan Jianyue. The electric power enterprise application based on the CIM/CIS integrated platform [J]. Zhejiang power, 2011, 26 (5):55-57.
[2]Xing Jialei, Yang Honggeng. A CIM data processing method orienting application system [J]. Automation of electric power systems, 2010 (special issue 18): 46-50.
[3]CaoYulin. Study on memory database realization mechanism based on FastDB [J].Journal of Sichuan university (natural science edition), 2011 Chinese (3):561-565.
[4]Lin Changnuan, Wu jian. CIM data storage research and implementation scheme based on RDF [J]. Science, technology and engineering, 2007, 7 (24):6339-6343.
[5]BE CKER D.I nter operability test # 10 of the common information model (CIM) and the generic interface definition (GID) standards: the power of the common information model (CIM) to exchange power system data. PaloAlto, CA, USA: EPRI, 2007.
[6]QuYang.280 examples of Visual C++ effectiveness programming [M]. Beijing: people's posts and telecommunications publishing house, 2009.
[7] Zhang Shenming, Bu Fanjiang. Follow the real-time database management system of IEC 61970 standard [J] Automation of electric power systems, 2002, 26 (24):26-30.
[8]CaoYulin, Wu Junsheng. The design and research based on the network file management information system [J]. Computer engineering and design, 2009 (6): 1532-1535.
[9]Ma Kuitao, Cai Ying. The realization method of Win32 process the information sharing [J]. Computer application and software, 2007, 24 (12) :119-121.