Use ORM Middleware Realize Heterogeneous Database Connectivity

Hongjun Song
Dept. of Computer Science and Technology
Northwest University
Xi'an, P.R.China
songhongjun@nwu.edu.cn

Ling Gao
Dept. of Computer Science and Technology
Northwest University
Xi'an, P.R.China
gl@nwu.edu.cn

***Abstract*—Using ORM middleware can easily shield the differences of databases, and realize the connection of heterogeneous databases, effectively solve the problem of Information Island, facilitate information sharing. Because Hibernate can better realize ORM framework, so this paper introduces the implementation steps of heterogeneous database connectivity through Hibernate.**

Keywords- ORM; heterogeneous database; Hibernate

I. INTRODUCTION

ORM is the abbreviation of Object Relation Mapping. Why would the ORM arise? First, from the development progress of programming language and database technology to analyze, the programming language is responsible for data processing and transmission, while the database is responsible to organize, store and manage data in accordance with data structure. In the past decades, from machine language to assembly language, to structured language, up to object-oriented language, the programming language becomes more abstract, more human. Especially, the structured programming language converts to the object-oriented language smoothly, which is a qualitative leap in program design. With the development of object-oriented programming language, the first commercial object-oriented database appeared in 1990, but it is different to that the object-oriented language completely replaces the structured language, the object-oriented database has not fully replaced the relational database, which is because the object-oriented database has many technical defects, now the mainstream database is still relational database. Because the objects in object-oriented languages and the tables in relational databases are two completely different concepts, ORM is a thoroughly solution to solve the impedance mismatch between program objects and database tables, promoting relational databases with object-oriented features, this has prompted the perfect combination between object-oriented programming languages and relational databases.

II. THE NECESSITY OF HETEROGENEOUS DATABASE CONNECTIVITY

Along with computer network and database technology have rapid development, information sharing plays an important role in people's work and learning, the demand for information sharing doesn't confine in a database system, but in several different database systems. For historical reasons, information resources exist in all kinds of database systems, these systems built by different databases are like information islands, there are two important problems to be solved that each isolated database system is to be connected perfectly and the user is to be provided transparent database access.

The heterogeneities of heterogeneous databases mainly show in the following aspects:

a) According to data models, heterogeneous databases can be divided into: hierarchical model, network model, relational model, object-oriented model.

b) According to different operating systems, heterogeneous databases can be divided into: Unix，Windows NT, Linux, OS / 2 and so on.

c) According to different database management systems, heterogeneous databases can be divided into：MySQL, SQL Server, Oracle, DB2 and so on.

d) According to different computer operating system platforms, heterogeneous databases can be divided into: supercomputer, mainframe, midrange, minicomputer, workstation, PC and embedded system.

e) According to the different control methods of database system, heterogeneous databases can be divided into: centralized database system, distributed database system and parallel database system.

In order to achieve heterogeneous database connectivity, International Standards Organization and database vendors have made unremitting efforts, for example: published in 1992, Microsoft ODBC (Open Database Connectivity), which provides standard APIs to access different DBMSs (Database Management System). Later, X / Open organization and ISO based on ODBC 3.0 specification，and further standardized, so in 1995, SQL / CLI standard published, which has become a part of the SQL standard. At the same time, IBM had developed the Distributed Relational Database Architecture (DRDA), which is the industry standard of heterogeneous database connectivity that has cross-platform access and follows the SQL standard.

978-1-4577-1964-6/12/$31.00 ©2012 IEEE

III. THE SCHEMES OF PREVIOUS HETEROGENEOUS DATABASE CONNECTIVITY

A. Through Gateway to Realize the Connection of Heterogeneous Database

Through gateway, database systems developed by different vendors can be transparently connected. At present, Sybase, Informix, Oracle etc. all have their own gateway products; they can support different datasource connections and run on multiple operating system platforms [1], as shown in figure 1. In this respect, Oracle has done more outstandingly. Through transparent gateway, Oracle is able to achieve heterogeneous database connectivity; the databases include SQL SERVER, SYBASE, DB2, Informix and other, and through optimization can provide excellent performance.

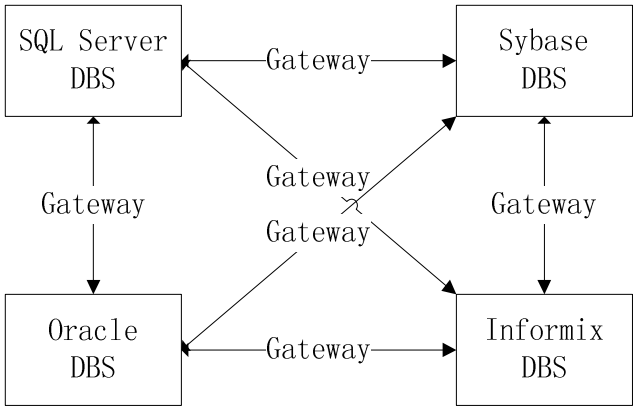

Figure 1. Through gateway to realize the connection of heterogeneous databases

B. Through ADO Components to Realize the Connection of Heterogeneous Databases

Microsoft ActiveX Data Objects (ADO) is a set of Component Object Model (COM) objects for accessing datasources. ADO exists between programming languages and OLE DB; through OLE DB, the details of database access are shielded. The biggest advantage of ADO components is that it can access any database. At the same time, ADO components are senior, which provide developers with a convenient object-oriented programming interfaces, developers don't have to care about the details of databases, as long as providing uniform SQL statements, the transparent access to heterogeneous databases can be achieved, thus using bottom connection pool technology can provide dynamic data exchange, finally realize the interconnection of heterogeneous databases.

C. Through JDBC to Realize the Connection of Heterogeneous Databases

JDBC is the abbreviation of Java Data Base Connectivity, a database connection technique achieved by Java language, and a Java implementation of ODBC (Open Database Connectivity) [2]. JDBC Contains two different APIs, one is the JDBC API for application developers, the other is the JDBC Drive API facing specific databases. Program developers call JDBC components through the upper JDBC API, JDBC components access specific databases through the bottom JDBC Drive API, so as to achieve different database access. JDBC supports concurrent control, can also connect to several different database systems, and distinguishes different database management systems through each URL. In this way, through using SQL statements we can easily achieve the interconnection of heterogeneous database systems.

D. Through XML Intermediate Conversion to Realize the Connection of Heterogeneous Databases

XML (Extensible Markup Language) is modified from the Standard Generalized Markup Language (SGML), however, XML provides unified approaches to describe and exchange structured data independent of specific applications or vendors, and therefore, XML has become a document processing and database processing combination. XML provides standard, but customizable ways to describe the contents of document, so it can be used to describe database views according to standard methods. In addition, when using XML Schema, data from the database can be automatically built to XML documents, and also we can automatically extract database data from XML documents, and it also has standard methods to define the corresponding relationships between document elements and database elements [3]. Heterogeneous database interconnection architecture based on XML is composed of three parts: in the lower layer, the data service layer composed by a variety of database systems is providers of system data; in the middle, the application service layer realizes data conversion between the database data and the XML documents; at the top is the user layer, which achieves transparent access to the underlying heterogeneous databases through the XML documents provided by the application service layer.

IV. THROUGH ORM MIDDLEWARE TO REALIZE THE CONNECTION OF HETEROGENEOUS DATABASES

By mapping entity classes to specific database tables, ORM realizes object persistence, thereby realizing the connection between the object-oriented programming and the relational database, as shown in figure 2. At present, the product that supports ORM framework is very many, like the traditional EJB, Oracle TopLink, Apache Software Foundation's subproject IBATIS, now more popular Hibernate, etc. Generally, the ORM framework includes the following four functions:

a) Having API for CRUD operations of entity class;

b) Through a special language or API, special operations are used to classes or class attributes;

c) Realizing the mapping provisions between entity classes and database tables;

d) The ORM framework realizes the corresponding relationship between classes; and realizes transaction control at all levels, and other optimization programmes.

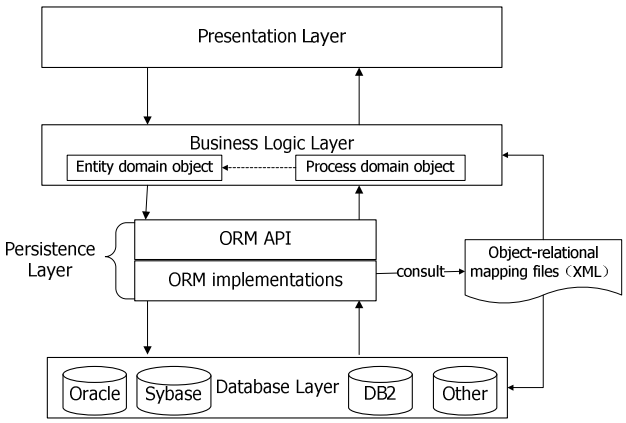

Figure 2. ORM model diagram

Hibernate is a typical representative of ORM, the following describes the steps of realizing the connection of heterogeneous databases through Hibernate, and the theme for the connection realization of other ORM frameworks is the same.

A. Establish Entity Classes Based on Corresponding Database Tables

The entity class in Hibernate is defined as POJO (Plain Old Java Object), which do one-to-one mapping to the database table, POJO is literally simple Java object, and generally has the following characteristics:

a) Create a default no-argument constructor;

b) Providing an identifier property (optional);

c) Use non-final class (optional);

d) Declare access methods for persistent fields and mark whether persistent fields could be modified.

In development projects, the traditional approach is to create database tables according to design documents, and then create corresponding entity classes. Using Hibernate, as long as the entity classes needed for projects have been established, basing on ORM configuration files, the corresponding tables in databases are automatically generated by Hibernate, and shielding the differences on field types in different databases. Assume that the databases to be connected are MySQL and Oracle, and I respectively define the corresponding POJO objects that are stored in different folders.

B. Create Object-Relational Mapping Files

Almost all of ORM middleware need mapping files between database tables and entity objects. Hibernate uses XML format files to specify the mapping relationship between the object and the relational table, used to be named “the main file name. Hbm.xml", in general, the mapping file and the corresponding POJO object are in the same folder, so easy to be found. In Hibernate mapping configuration files, you can specify basic data type mapping, primary key mapping, set mapping, entity mapping, and specify the component relationship between database tables: inheritance, one to one, one to many and so on. Hibernate ORM framework as shown in figure 3.

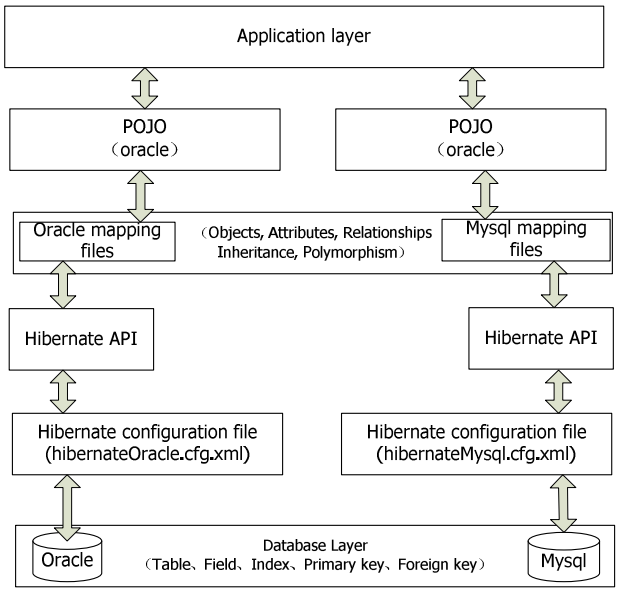

Figure 3. Hibernate ORM framework

C. Hibernate Configuration Files

Finished the object-relational mapping files of entity classes, you must also configure Hibernate configuration files. The functions of Hibernate configuration files are to tell database system address, user, user password, database dialect and so on, the operations of underlying databases are entirely managed by Hibernate. There are three kinds of Hibernate collocation methods: one is using XML format file, and now we mostly use the collocation method; another is the use of properties file format; the last one is using programmable configuration. Here, using XML format files, we connect a local MySQL database and a remote Oracle database, the main code of configuration files is as follows:

MySQL configuration file:hibernateMysql.cfg.xml

<property name="hibernate.connection.url">jdbc:mysql://local host/hibernate_student</property>

<property name="hibernate.connection.driver_class">com.mys ql.jdbc.Driver</property>

<property name="hibernate.dialect">org.hibernate.dialect.MyS QLDialect</property>

Oracle configuration file:hibernateOracle.cfg.xml

<property name="hibernate.connection.url">jdbc:oracle:thin: @219.245.18.231:1521:xe</property>

<property name="hibernate.connection.driver_class">oracle.jd bc.driver.OracleDriver</property>

<property name="hibernate.dialect">org.hibernate.dialect.Orac leDialect</property>

D. Through Entity Classes Export Corresponding Database Tables

Hibernate provides hbm2ddl that could reversely generate database tables from entity classes. According to POJO mapping files, hbm2ddl can not only create the corresponding database tables, but also establish the relationships between tables, such as one to many, many to one, one to one, mapping, inheritance and other relations, so that we can transparently access the database. As long as caring about the corresponding entity class and complying with Hibernate rules, users can use the database by object-oriented approach.

Here is the main code that can export POJO to different databases:

```
Configuration cfgMysql = new Configuration().configure("/hibernateMysql.cfg.xml");  //Mysql

Configuration cfgOracle = new Configuration().configure("/hibernateOracle.cfg.xml");  //Oracle

SchemaExport exportMysql = new SchemaExport(cfgMysql);

SchemaExport exportOracle = new SchemaExport(cfgOracle);

exportMysql.create(true, true);

exportOracle.create(true, true);
```

E. How to Use the Data in Heterogeneous Databases

After having established database tables, we could be able to use the database through object-oriented approach. Through generating different SessionFactories link different datasources to achieve database CRUD operations. For the operation of a single datasource, Hibernate has a good transaction control mechanism, as follows:

```
Configuration cfgMysql = new Configuration().configure("/hibernateMysql.cfg.xml");//hibernateOracle.cfg.xml (Oracle)

SessionFactory factoryMysql = cfgMysql.buildSessionFactory();

Session sessionMysql = null;

try {

        sessionMysql = factoryMysql.openSession();

        sessionMysql.beginTransaction();

        ……..//data processing

        sessionMysql.getTransaction().commit();

}catch(Exception e) {

        sessionMysql.getTransaction().rollback();

}finally {

      if (sessionMysql.isOpen()&&sessionMysql != null) {

              sessionMysql.close();

       } }
```

For heterogeneous databases, it is necessary to consider distributed transaction processing, JTA transaction declared by Hibernate API doesn't support distributed transaction，JTA API must be used to ensure the implementation of distributed transaction [4]. To ensure the ACID properties between different databases, it is mainly using the two-phase commit protocol to ensure that transaction correctly commit or rollback. If your persistence layer runs in an application server (e.g. behind EJB session beans), every datasource connection obtained by Hibernate will automatically be part of the global JTA transaction. You can also install a standalone JTA implementation and use it without EJB [5]. Hibernate global transaction management as shown in figure 4.

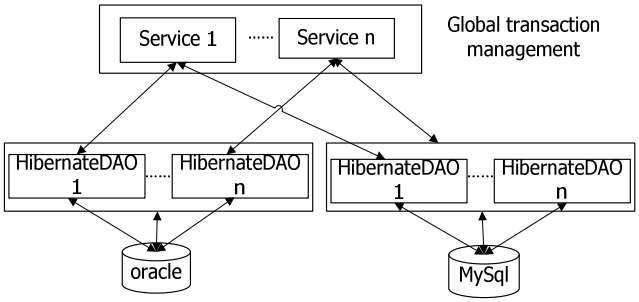

Figure 4. Global transaction management

V. CONCLUSIONS

This article describes heterogeneous database connectivity by ORM, and using Hibernate ORM mapping mechanism analyzes the steps of heterogeneous database connectivity, proves that it is convenient to realize the connection of heterogeneous databases by ORM middleware. Especially now, considering the object-oriented programming language and the relational database occupy major positions，the realization of ORM is essential condition to solve the object-relational impedance mismatch. Of course, only relying on one technology in the actual development, it is difficult to achieve the perfect connection of heterogeneous databases; only ORM combines with other frameworks can we better play its functions. For example, you can use spring+hibernate+jotm to realize multi-database transaction management, so as to realize the perfect connection of heterogeneous databases.

REFERENCES

[1] X. H. Ye, "Investigate on the connection of heterogeneous databases," Computer Era, 2002(08): pp. 7-9.

[2] Y. Y. Zhang, "JAVA database connection(JDBC) technique and its realization," Journal of College of Disoster Prevention Technique, 2003(04):pp.10-14.

[3] D. M. Kroenke and D. J. Auer, Database Processing, 2011, BEIJING: Publishing House of Electronics Industry, pp. 406-438.

[4] W. Q. Sun, Mastering Hibernate:let Java objects hibernate in the relational database, 2rd ed., 2010: Publishing House of Electronics Industry, pp. 526-530.

[5] K. Gavin, B. Christian, R. A. Max, B. Emmanuel, E. Steve and F. Hardy, Hibernate Reference Documentation 3.6.8.Final, Copyright © 2004 Red Hat, Inc, October 26, 2011 .