"Knowledge Scholar"
An International Peer Reviewed Journal Of Multidisciplinary Research
Volume: 02, Issue: 02, March–April 2015 *eISSN NO. 2394-5362*
Impact Factor (IIFS) - 0.776

AN ALGORITHMIC APPROACH FOR DATA BINDING TO FEDERATED DATABASES

Dr. Moussa Demba
Assistant Professor,
Aljouf University,
Kingdom of Saudi Arabia

Mohammed Afzal Bashu
Senior Lecturer,
Aljouf University,
Kingdom of Saudi Arabia

Abstract

This research article describes binding method for different database in to a single procedure or DLL file to connect the heterogeneous federated database. Due to a modern technologies use a huge applications with different platform .An algorithm shows the approach to bind the different software to a federated database. A detailed analysis and discussion of the data obtained through questionnaires. This algorithm approach shows developer to feel free to combine heterogeneous level of data. The architecture of the FBD is one solution for common binding on global application processing for different application. This study helps the scholar to find the binding over a global application in modern computing environment.

Introduction

A federated database system is a collection of meta-database managed by many other local and component databases. In this modern world the organizational data is spreading over various parts of the world. There are many methods to bind the database for the connection of frontend and backend applications. But we need to provide a common connecting database which would be helpful to connect the databases in a single method. And it should be connect over all these localized databases and support the global data processing application. The further architectural work for FDB management system is a solution for all common databases in global application processing system. In this article we have made an algorithm which will describe the common connectivity for the database and it will be very useful. Further we have taken a survey and issues which will define all the connection of our connection of database.

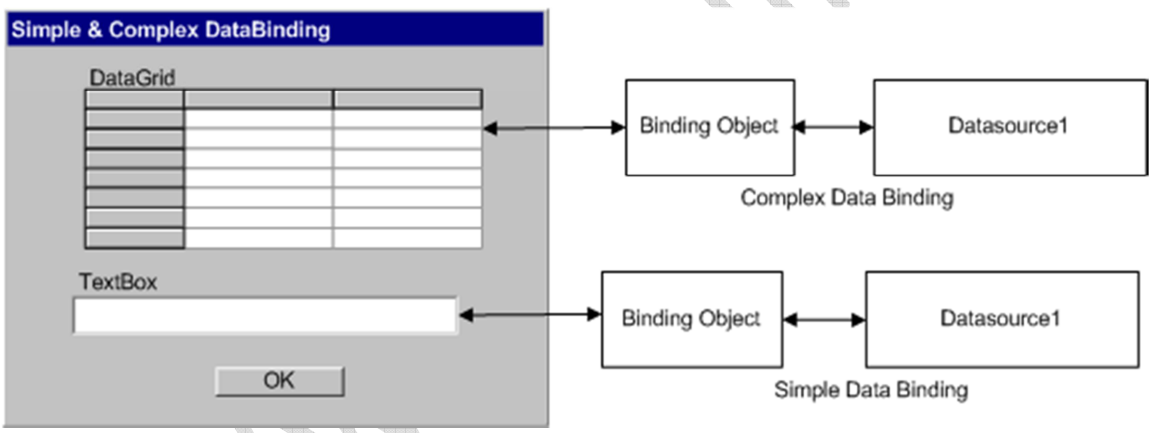
Definition

1. algorithmic approach
2. federated databases
3. Heterogeneous Database
4. Binding

Present Binding Heterogeneous Database

There are uncountable databases which are different from each other in the case of language program. By connecting one programing language to another programing language a problem arises in the connectivity of two databases from which our whole program gets distracted. From this following survey we can see some databases with other programing languages which connect with other database.

VB 6




```
Private Const afz_CONN_STRING = "Provider=sqloledb;Data
Source=%SERVER%;Database=%DataBase%;User
ID=%UserID%;Password=%Password%"
Public Sub OpenConnection(ByVal Server as String, ByVal Database as String, ByVal
UserId as String, ByVal Password as String)
 strConn = afz_CONN_STRING
 afz_CONN_STRING = Replace$(strConn, "%SERVER%", Server, , , vbTextCompare)
strConn = Replace$(strConn, "%Database%", Database, , , vbTextCompare)    strConn =
Replace$(strConn, "%UserID%", UserId, , , vbTextCompare)    strConn =
Replace$(strConn, "%Password%", Password, , , vbTextCompare)
```

Java

```
public class SimpleForm extends javax.swing.JFrame {
   Connection afz_CONN_STRING;      Statement stmt;      ResultSet rs;
   /** Creates new form SimpleForm */
   public SimpleForm() {         initComponents();
      afzConnect();      }
   public void afzConnect( ) {
      try {            //connect to the database
         afzconnvar=
DriverManager.getConnection("jdbc:postgresql:mydb","testuser","testpassword");
          stmt = con.createStatement();
          String sql = "select firstname, lastname, name from employees, departments where
employees.department = departments.id";
        rs = stmt.executeQuery(sql);
  rs.next();
```

Oracle

```
static void Main(string[] args)
{  string constr = "User Id=hr; Password=hr; Data Source=oramag";
  OracleConnection con = new OracleConnection(constr);
  con.Open();
  StringBuilder sbSQL = new StringBuilder();    sbSQL.Append("select    country_name ");
  sbSQL.Append("from     countries ");   sbSQL.Append("where     country_id =
:country_id");
  OracleCommand cmd = new OracleCommand();
  cmd.Connection = con;   cmd.CommandText = sbSQL.ToString();
  OracleParameter p_country_id = new OracleParameter();
  p_country_id.OracleDbType = OracleDbType.Varchar2;
  p_country_id.Value = "AA";
```

asp.net

```
<asp:LinqDataSource ID="LinqDataSource1" runat="server"
   ContextTypeName="AdventureWorksLTDataClassesDataContext"
   EnableDelete="True" EnableInsert="True" EnableUpdate="True"
   TableName="SalesOrderDetails">
</asp:LinqDataSource>
<asp:GridView ID="GridView1" runat="server"
   AutoGenerateColumns="False"
   DataKeyNames="SalesOrderID,SalesOrderDetailID"
   DataSourceID="LinqDataSource1">
   <Columns>
      <asp:CommandField ShowDeleteButton="True"
         ShowEditButton="True" />
      <asp:BoundField DataField="SalesOrderID"
         HeaderText="SalesOrderID" ReadOnly="True"
         SortExpression="SalesOrderID" />
      <asp:BoundField DataField="SalesOrderDetailID"
         HeaderText="SalesOrderDetailID" InsertVisible="False"
         ReadOnly="True" SortExpression="SalesOrderDetailID" />
      <asp:BoundField DataField="OrderQty"
         HeaderText="OrderQty" SortExpression="OrderQty" />
      <asp:BoundField DataField="ProductID"
         HeaderText="ProductID" SortExpression="ProductID" />
      <asp:BoundField DataField="UnitPrice"
         HeaderText="UnitPrice" SortExpression="UnitPrice" />
      <asp:BoundField DataField="ModifiedDate"
         HeaderText="ModifiedDate" SortExpression="ModifiedDate" />
   </Columns>
</asp:GridView>
```

Php

```
<?php
# FileName="myconnect.php"
$hostname = "localhost";
$database = "customers";
$username = "root";
$password = "";
?>
```

Now we have our file to do the connection for us and we need to create the file that will run the query and bring the data so our Grid can be populated. We will call the file data.php.

```
<?php
#Include the connect.php file
include('connect.php');
#Connect to the database
//connection String
$connect = mysql_connect($hostname, $username, $password)
or die('Could not connect: ' . mysql_error());
//select database
mysql_select_db($database, $connect);
//Select The database
$bool = mysql_select_db($database, $connect);
if ($bool === False){
  print "can't find $database";
}
// get data and store in a json array
$query = "SELECT * FROM customers";
$from = 0;
$to = 30;
$query .= " LIMIT ".$from.",".$to;

$result = mysql_query($query) or die("SQL Error 1: " . mysql_error());
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
```

```
        $customers[] = array(
       'CompanyName' => $row['CompanyName'],
       'ContactName' => $row['ContactName'],
        'ContactTitle' => $row['ContactTitle'],
        'Address' => $row['Address'],
        'City' => $row['City']
     );
} echo json_encode($customers);
```

Serverlight

```
UserControl x:Class="SampleCode.BindingModesExamples"
    xmlns:local="clr-namespace:SampleCode"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"    >
  <UserControl.Resources>
     <local:Employee x:Key="employeeInfo" FirstName="Aaaaah" LastName="Kaaaa"/>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot"
    DataContext="{StaticResource employeeInfo}"
    Background="LightCyan" Width="500" Height="300"
    HorizontalAlignment="Left" >
     <Grid.ColumnDefinitions >
        <ColumnDefinition Width="100" />
        <ColumnDefinition Width="100" />
        <ColumnDefinition Width="200" />
         <Grid.RowDefinitions>
       <RowDefinition Height="100"/>
       <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
     </Grid.RowDefinitions>
```

```
<TextBox x:Name="tbFirstName" Text="{Binding Path=FirstName, Mode=OneTime}"
  Grid.Column="2" Grid.Row="1"></TextBox>  </Grid>  </UserControl>
```

Sql server

```
CREATE REMOTE SERVICE BINDING binding_name [ AUTHORIZATION
owner_name ] TO SERVICE 'service_name' WITH USER = user_name [ ,
ANONYMOUS = { ON | OFF } ][ ; ]
```

Python

```
SQLALCHEMY_DATABASE_URI = 'postgres://localhost/main'
SQLALCHEMY_BINDS = {
   'users':       'mysqldb://localhost/users',
   'appmeta':     'sqlite:////path/to/appmeta.db'
}
```

Creating and Dropping Tables

The create_all() and drop_all() methods by default operate on all declared binds, including the default one. This behavior can be customized by providing the bind parameter. It takes either a single bind name, '__all__' to refer to all binds or a list of binds. The default bind (SQLALCHEMY_DATABASE_URI) is named None:

```
>>> db.create_all()
>>> db.create_all(bind=['users'])
>>> db.create_all(bind='afzmeta')
>>> db.drop_all(bind=None)
```

Referring to Binds

If you declare a model you can specify the bind to use with the __bind_key__ attribute:

```
class User(db.Model):
    __bind_key__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
```

Internally the bind key is stored in the table's info dictionary as 'bind_key'. This is important to know because when you want to create a table object directly you will have to put it in there:

```
user_favorites = db.Table('user_favorites',
    db.Column('user_id', db.Integer, db.ForeignKey('user.id')),
    db.Column('message_id', db.Integer, db.ForeignKey('message.id')),
    info={'bind_key': 'users'}
)
```

C++

```
d_Database db;
d_Transaction txn;
try {
   db.open("addressDB");
   txn.begin();
   // perform query
    d_OQL_Query query(
       "select x from Person x where x.name = \"Doug Barry\"");
   d_Bag<d_Ref<Person> >   allDougBarrys;
   d_oql_execute(query, allDougBarrys);
   d_Iterator<d_Ref<Person> > iter = allDougBarrys.create_iterator();
   // iterate over the results
   d_Ref<Person> p;
   while( iter.next(p) ){
     // do some addition processing on the person (not shown)
     // now traverse to the address object and update its value
     p->address.street = "13504 4th Avenue South";
   }
   txn.commit();
   db.close();
}
```

Vc++

```
namespace DataBinding2
{
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;
    public __gc class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
        }
    protected:
        void Dispose(Boolean disposing)
        {
            if (disposing && components)
            {
```

```
                components->Dispose();
            }
            __super::Dispose(disposing);
        }
    private: System::Windows::Forms::DataGrid *  dataGrid1;

    private:
        System::ComponentModel::Container * components;

        void InitializeComponent(void)
        {
            System::Resources::ResourceManager *  resources =
new System::Resources::ResourceManager(__typeof(DataBinding2::Form1));
            this->dataGrid1 = new System::Windows::Forms::DataGrid();
(__try_cast<System::ComponentModel::ISupportInitialize *  >(this->dataGrid1))-
>BeginInit();
            this->SuspendLayout();
            this->dataGrid1->Anchor = (System::Windows::Forms::AnchorStyles)
(((System::Windows::Forms::AnchorStyles::Top |
System::Windows::Forms::AnchorStyles::Bottom)
                | System::Windows::Forms::AnchorStyles::Left)
                | System::Windows::Forms::AnchorStyles::Right);
        this->dataGrid1->DataMember = S"";
```

```
this->dataGrid1->HeaderForeColor = System::Drawing::SystemColors::ControlText;
        this->dataGrid1->Location = System::Drawing::Point(16, 16);
        this->dataGrid1->Name = S"dataGrid1";
        this->dataGrid1
```

Present Database Connectivity Issues

The users of different programing languages use different connectivity which is difficult to connect. If the organization wants to begin a change in the connectivity of programing database and languages means they need to do some extra efforts to develop their connecting programs easily.

Modern data users demand control over how data is presented; their needs are somewhat in conflict with such bottom-up approaches to data integration.

An Algorithm for Common Database Connectivity

Input：Name of the plate form with type of and name of the database

Logic: ConntBinding(PLF,D,TBL,LOOKUPTABEL)

Output：Establish the connection to desire connection

Method:

1. Read Nameof plateform PLF , Database name DB , Tblname TBL ;

2. DO ConntBinding(PLF,DB,TBL,LOOKUPTABEL)

3. Set Numbers for all Plateform according to front end

4. Set all Database and its table according to input

5. End

1. Return ConntBinding(PLF,DB,TBL,LOOKUPTABEL) ;

2. Input PLF , DB,TBL ,LOOKUPTABEL;

3.If PLF== [1, N] GO TO PLFNAME , LOOKUPTABLE

Else Invalid PLF

4.If DB== [1, N] GO TO DBNAME , LOOKUPTABLE

Else Invalid DB

5. if TBL== [1, N] GO TO TBLNAME , LOOKUPTABLE

Else Invalid TBL

6. Returns PLF,DB,TBL;

LOOKUPTABLE → A table which consists whole details about any database binding is said to be Lookuptable and it establishes with connection, table path, assign number etc. According to lookuptable all the above program languages platform will have numbers which will assign each programing languages with a unique number. Lookuptable is very

much useful for database binding and if a user gives the perametres like names of platform, table, database which is already assign a number by lookuptable and it will match and the binding will bind automatically.

Conclusions

This paper describes designer to develop the Federated database system. Hence the database uses to store and retrieved the information due to the database system. They have to be taken into account differently compared to major changes whose impact sphere is an important part of the system for each type of change; we define one or several techniques to be used in order to ensure that the evolution is propagated both in the system and in its documentation. We propose a framework matching three dimensions of changes, and database life-cycle phases, to evolution techniques such as forward engineering, reverse engineering migration process, etc. Further work will consist in defining a guidance tool in order to help database designers and maintenance teams in managing database unanticipated evolution. This guidance tool will be based on our change typology and on our framework. Prototyping this tool will allow us to validate this typology and/or enrich it.

References

1. Computer Algorithms: Introduction to Design and Analysis Sara Baase, Allen Van Gelder, Algorithm Design Éva Tardos, Jon Kleinberg
2. Akoka J, Comyn-Wattiau I, MeRCI: , Algorithms Sanjoy Dasgupta, Umesh Vazirani, Christos PapadimitriouAn Expert System for Software Reverse
3. Batini C., Lenzerini M. and Navathe S., A Comparative Analysis of methodologies for Database Schema Integration. ACM Computing Surveys Journal 18 (4), 1986.Management of ER'99,LNCS 1727, Paris, France, 1999.
4. Yugopuspito P. and Araki K., Evolution of Relational Database to Object-Relational Database in Abstract Level. Proceedings of the InternationalWorkshop on Principles of Software Evolution, July 1999, pp 103-107.
5. Zicari R., A Framework of Schema Updates. Proceedings of the 7 □_ International Conference on Data Engineering, Japan, 1991.