Introduction To SharePoint XQuery Reporting Solution

作者：王昕
http://www.cnblogs.com/starcrm

1. Projects

1.1 XmlToDB

The project use to export the SharePoint form library's XML files to new database table, or update the XML files to existing database field(XML data type). And it can save the associate XML nodes value to database fields with easy configure.

Major files:

- App.config: Configure file for control SharePoint site URL / DB Connection string / XML- Database mapping…
- Program.cs : Console entry file for this project , Loop the SPListItem and run the function Common.SaveXMLFileToDB();
- Common.cs : Contain the function: SaveXMLFileToDB(SPListItem listItem) , there are two mode to save database:insert and update,control by ConfigurationManager.AppSettings["Mode"]

Code:

```
public static void SaveXMLFileToDB(SPListItem listItem)
        {
            Stream stream = null;
            MemoryStream fileStream = null;
            try
            {
                //throw new Exception("test");
                SPFile file = listItem.File;
                if (file != null)
                {
                    XmlDocument xDoc = new XmlDocument();
                    byte[] bytes = file.OpenBinary();
                    stream = new MemoryStream(bytes);
                    xDoc.Load(stream);
```

```
                    XmlNamespaceManager xNsMgr = new
XmlNamespaceManager(xDoc.NameTable);
                    string strNamespace = Common.GetNamespace(file.Name);
                    xNsMgr.AddNamespace("my", strNamespace);
                    SaveXML(xDoc, xNsMgr);
                }

            }
            catch (Exception ex)
            {
                WriteLog.LogErrMsg(ex);
                Console.WriteLine(ex.Message);
            }
            finally
            {
                if (stream != null)
                    stream.Close();
                if (fileStream != null)
                    fileStream.Close();
            }
        }

        public static void SaveXML(XmlDocument xDoc, XmlNamespaceManager
xNsMgr)
        {
            string sql = "";
            string xml = xDoc.InnerXml;
            string DbTableName =
ConfigurationManager.AppSettings["DbTableName"];
            string keyColumn =
ConfigurationManager.AppSettings["keyColumn"].Split(',')[0];
            string keyValue =
xDoc.SelectSingleNode(ConfigurationManager.AppSettings["keyColumn"].Split(','
)[1], xNsMgr).InnerText;
            string XmlColumn = ConfigurationManager.AppSettings["XmlColumn"];

            Dictionary<string, string> dicColumnAssociate = new
Dictionary<string, string>();
            Dictionary<string, object> valueAssociate = new Dictionary<string,
object>();
            string sColumnAssociate =
ConfigurationManager.AppSettings["ColumnAssociate"];
            string[] arrColumnAssociate = sColumnAssociate.Split(';');
            for (int i = 0; i < arrColumnAssociate.Length; i++)
            {
                dicColumnAssociate.Add(arrColumnAssociate[i].Split(',')[0],
arrColumnAssociate[i].Split(',')[1]);
                valueAssociate.Add(arrColumnAssociate[i].Split(',')[0],
xDoc.SelectSingleNode(arrColumnAssociate[i].Split(',')[1],  xNsMgr).InnerText);
            }

            //Mode is insert
            string mode = ConfigurationManager.AppSettings["Mode"];
            if (mode == "insert")
            {
                sql = "INSERT INTO [{0}]";
```

```
                sql += "([{1}]";

                foreach (KeyValuePair<string, string> kvp in
dicColumnAssociate)
                {
                    sql += ",[" + kvp.Key + "]";
                }

                sql += ",[{2}]) VALUES ";

                sql += "('{3}'";

                foreach (KeyValuePair<string, object> kvp in valueAssociate)
                {
                    sql += ",'" + kvp.Value.ToString() + "'";
                }

                sql += ",'{4}')";


                sql = string.Format(sql, DbTableName, keyColumn, XmlColumn,
keyValue, xml);
                if (Common.ExecSQLForNonQuery(sql) == 1)
                {
                    WriteLog.LogMsg(keyValue+" Exported");
                    Console.WriteLine(keyValue + " Exported");
                }

            }
            else if (mode == "update")            //Mode is update
            {
                //todo
            }

           }
```

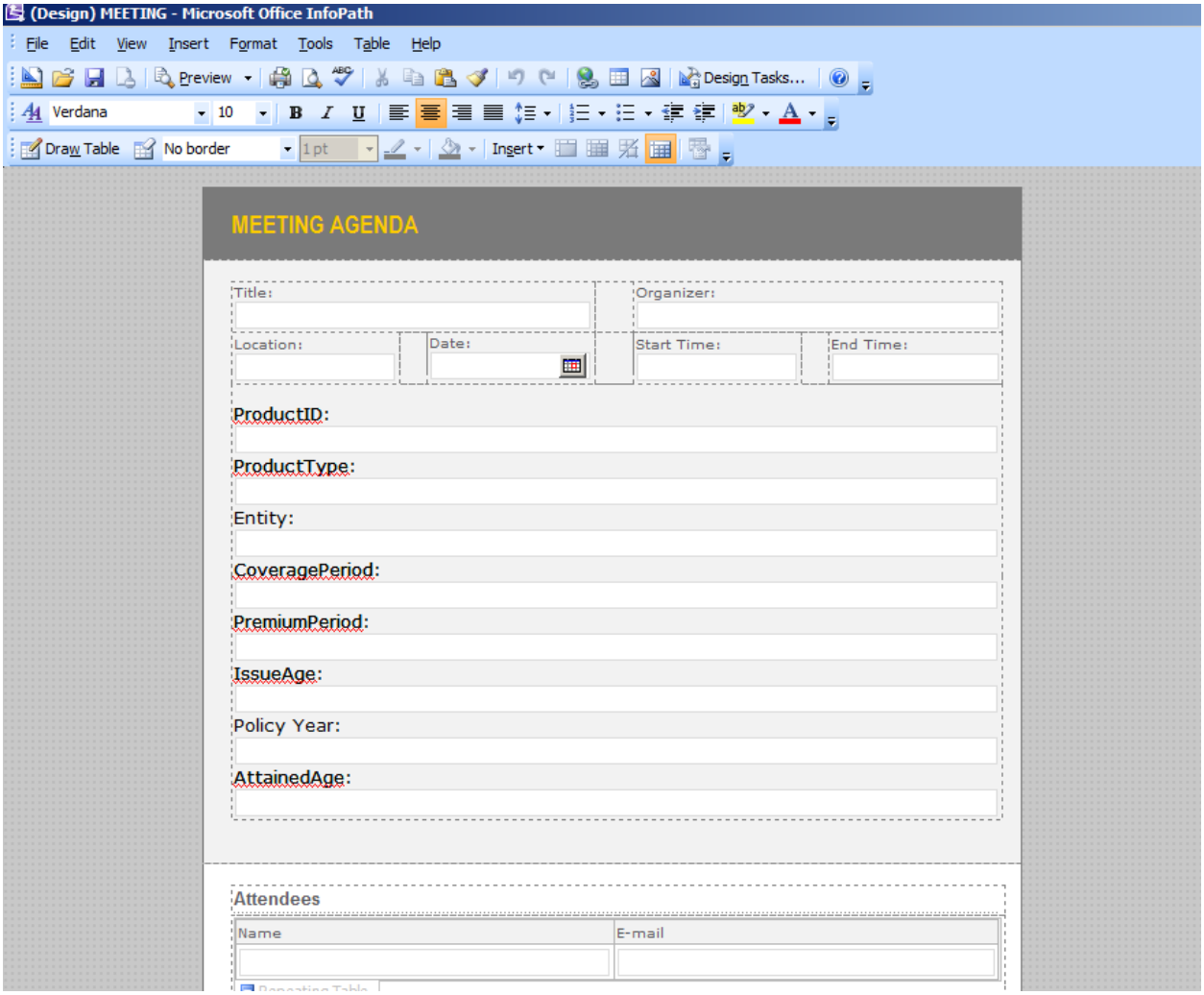
Database and test form library introduction

Database major field:

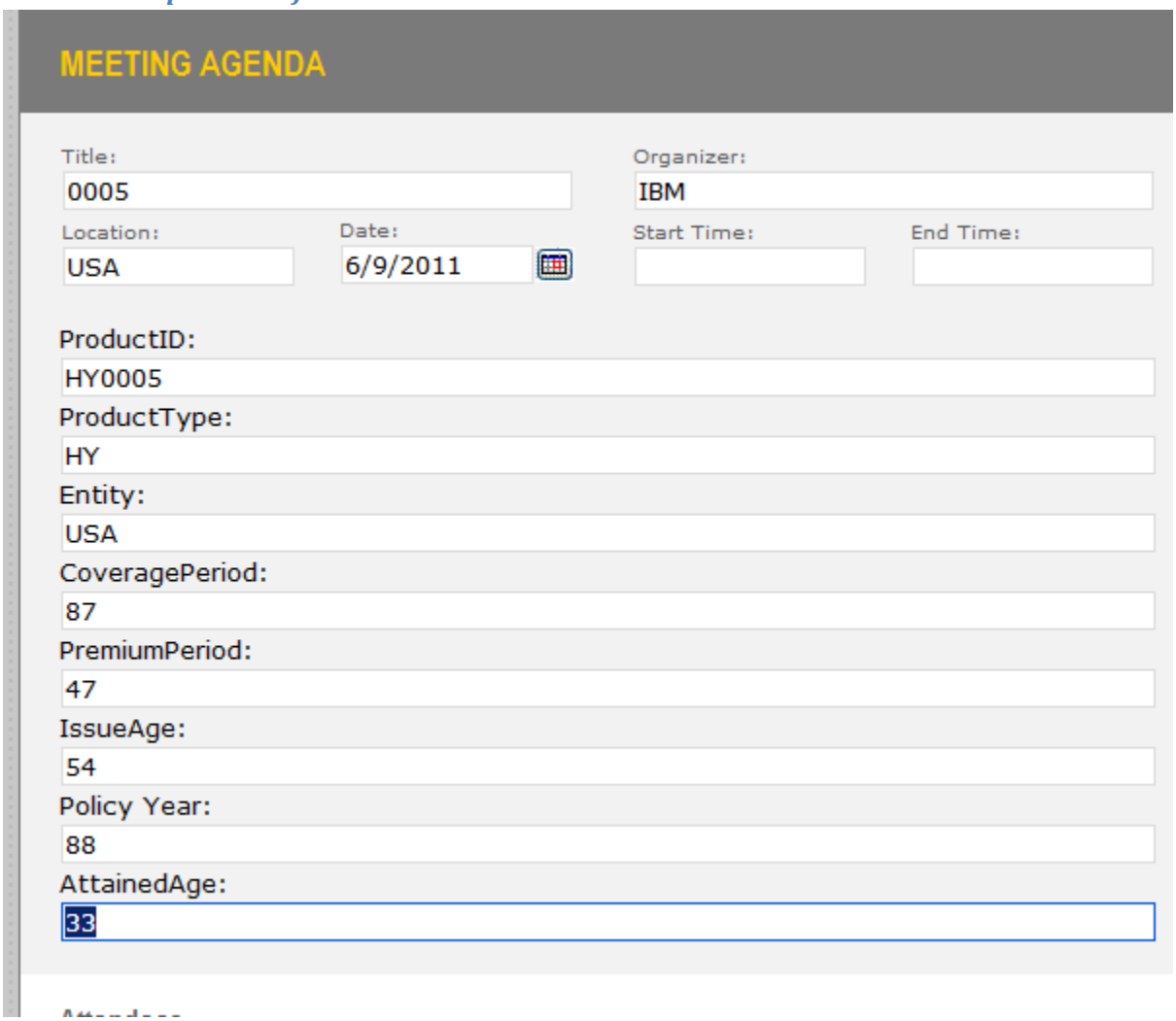
- ProductName
- XmlInfo: XML data type field, to store SharePoint form library's XML file

DEMO:

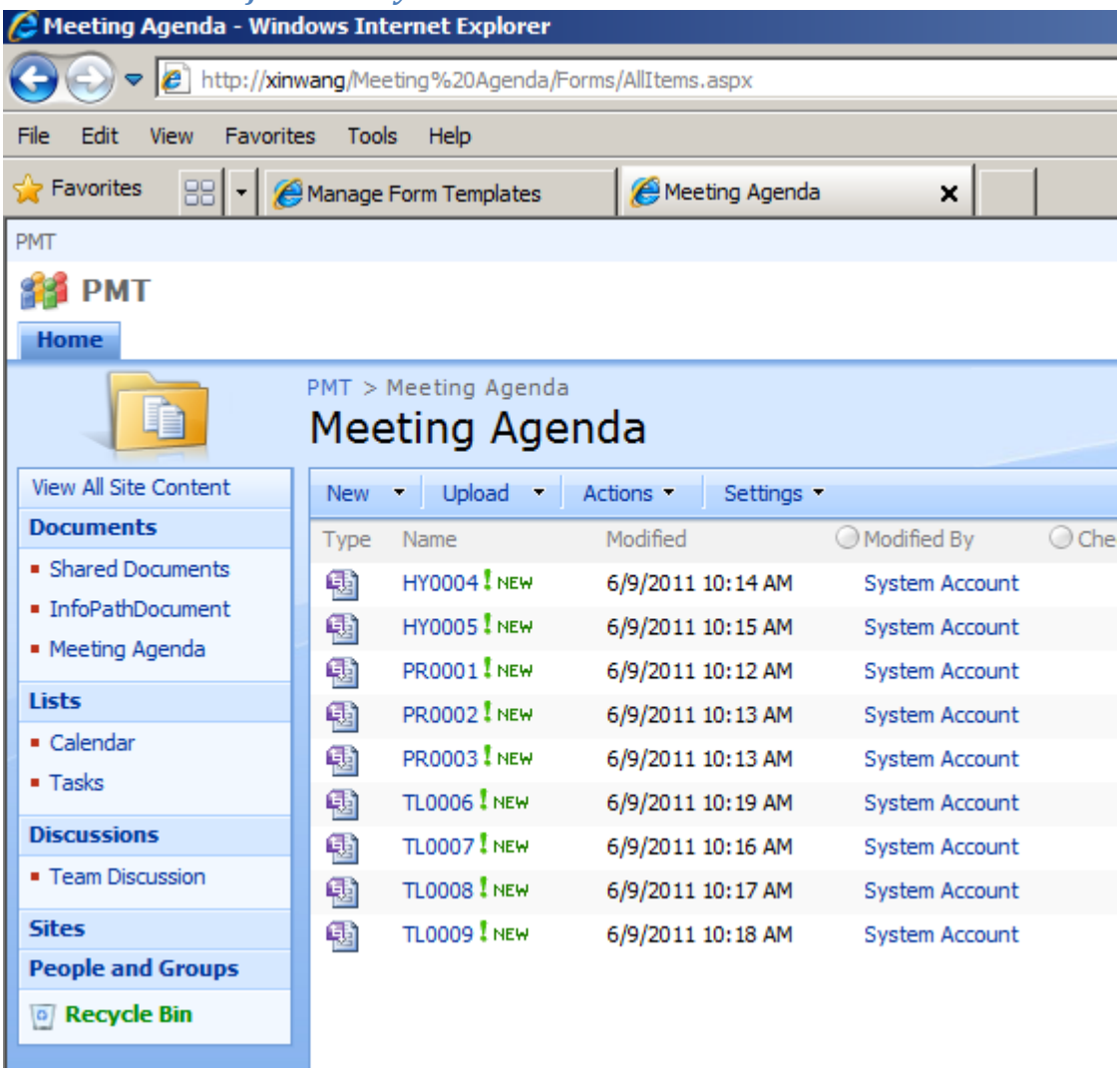
Step1:design the demo infopath template:



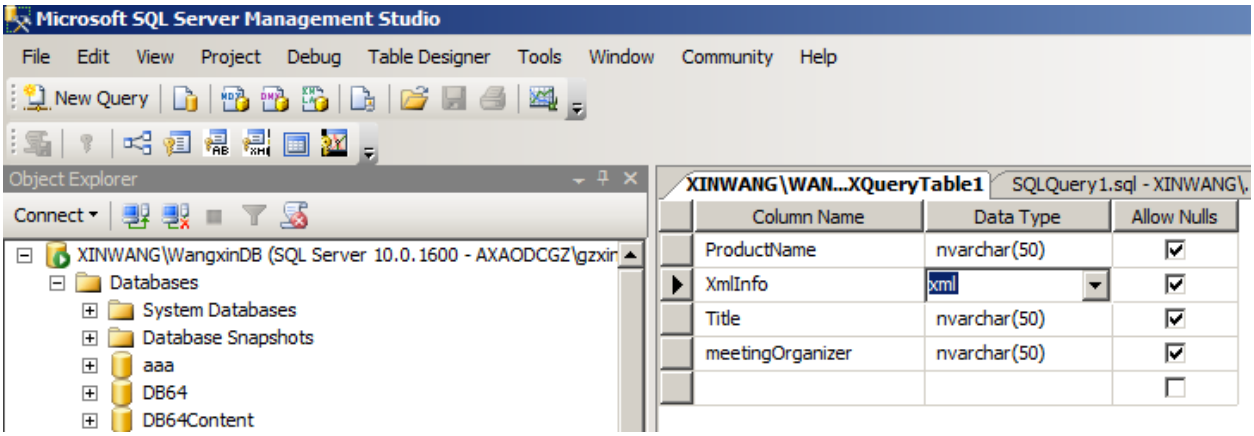
Step2:Pulish template and fill some dome data:



Step3:Save the data to form library:



Step4: Prepare the demo datebase,



Step5:Config the XMLtoDB project

XQueryDemo - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug XML Data Tools 测试(S) Window Help

Debug | Any CPU | 3.1

Default.aspx.cs | Web.config | Default.aspx | SearchElement.ascx | sql.txt | Common.cs | WriteLog.cs | ExcelHelper.cs | App.config | Program.cs | web.config | Start Page

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <!--insert or update-->
    <add key="Mode" value="insert" />
    <add key="SiteUrl" value="http://xinwang" />
    <add key="LibraryName" value="Meeting Agenda" />
    <add key="DbTableName" value="XQueryTable" />
    <!--dbcolumn,xpath-->
    <add key="keyColumn" value="ProductName,/my:meetingAgenda/my:meeting/my:meetingLocation" />
    <!--the db column used to store xml,db type is XML-->
    <add key="XmlColumn" value="XmlInfo" />
    <!--save the associate XML nodes value to db,format is:dbcolumn,xpath-->
    <add key="ColumnAssociate" value="title,/my:meetingAgenda/my:meeting/my:meetingTitle;meetingOrganizer,/my:meetingAgenda/my:meeting/my:meetingOrgar
    <add key="LogFilePath" value="D:\XmlQueryDemo\" />
    <!-- LogFileType 1:excel 2:csv 3:txt -->
    <add key="LogFileType" value="1"/>
    <add key="ConnString" value="Data Source=xinwang\wangxindb;database=XmlQueryDemo;uid=sharepoint;pwd=ABCdef123" />
    <add key="UsingUpdated" value="true" />
  </appSettings>
</configuration>
```

Step6: Run the XMLtoDB.exe program

File Edit View Project Build Debug Data Tools 测试(S) Window Help

Debug | Any CPU | 3.1

Default.aspx.cs | Web.config | Default.aspx | SearchElement.ascx | sql.txt | Common.cs | WriteLog.cs | ExcelHelper.cs | App.config | Program.cs | web.config

PMT24.Common | SaveXML(XmlDocument xDoc, XmlNamespaceManager xNsMgr)

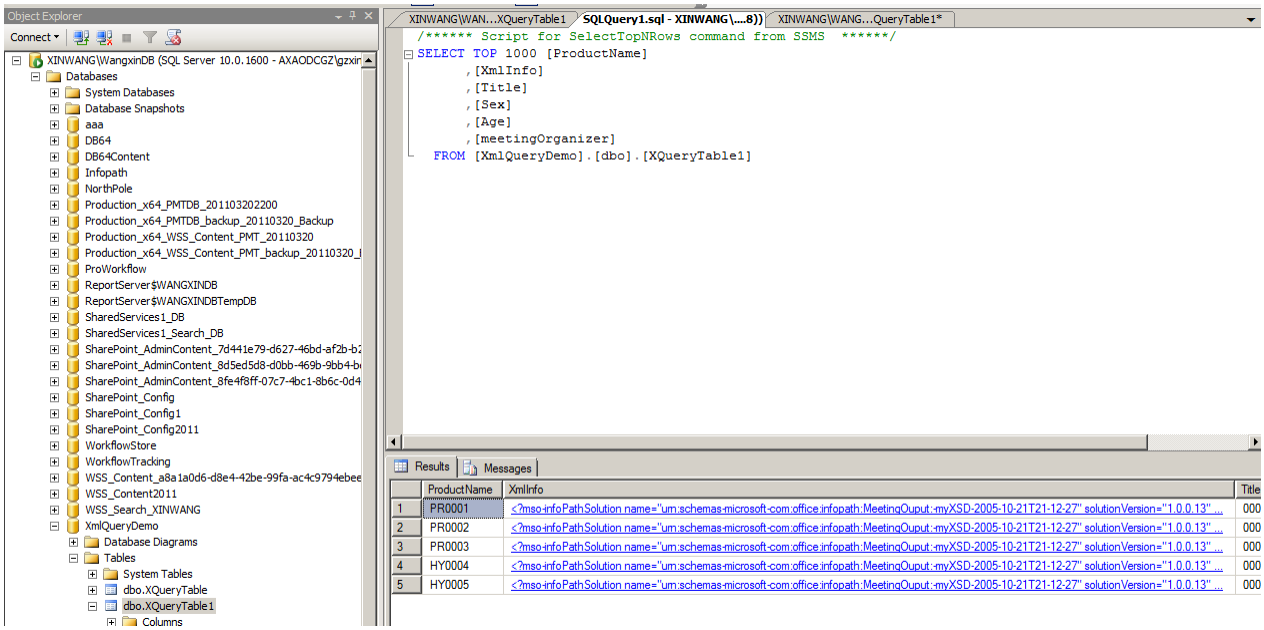
```
public static void SaveXMLFileToDB(SPListItem listItem)
{
    Stream stream = null;
    MemoryStream fileStream = null;
    try
    {
        //throw new Exception("test");
        SPFile file = listItem.File;
        if (file != null)
        {
            XmlDocument xDoc = new XmlDocument();
            byte[] bytes = file.OpenBinary();
            stream = new MemoryStream(bytes);
            xDoc.Load(stream);

            XmlNamespaceManager xNsMgr = new XmlNamespaceManager(xDoc.NameTable);
            string strNamespace = Common.GetNamespace(file.Name);
            xNsMgr.
            SaveXMI
        }

    }
    catch (Exceptic
```

file:///C:/Users/gzxinw/Documents/Visual Studio 2008/Projects/XQueryDemo/XmlToDB/bin/Debug/X...

```
=== 6/9/2011 10:39:59 AM 'http://xinwang' Xml data
PR0001 Exported
PR0002 Exported
PR0003 Exported
HY0004 Exported
HY0005 Exported
```

file:///C:/Users/gzxinw/Documents/Visual Studio 200

Step7:Check database. Xml has saved to database.



1.2 XQuery Web Demo

Query the database XML type field.

Major files:

- Default.aspx : start page;
- SearchElement.ascx : user control for auto generate query conditions;

Demo:

Step1:Try to query xml field

http://localhost:49381/Default.aspx - Windows Internet Explorer

http://localhost:49381/Default.aspx

File Edit View Favorites Tools Help

Favorites http://localhost:49381/Default.aspx

Page Safety Tools

```
SELECT xmlinfo from dbo.XQueryTable1
```

XML data type field

Basic Search | Ignore Blank Search

xmlinfo


```
<?mso-infoPathSolution name="urn:schemas-microsoft-com:office:infopath:MeetingOuput:-myXSD-2005-10-21T21-12-27" solutionVersion="1.0.0.13" productVersion="12.0.0.0" PIVersion="1.0.0.0" hr
><my:meetingAgenda xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/200:
US"><my:meeting><my:meetingTitle>0001</my:meetingTitle><my:meetingOrganizer>AXA</my:meetingOrganizer><my:meetingLocation>CN</my:meetingLocation><my:meetingDate>2011-06-09</my:
xsi:nil="true" /><my:meetingObjective /></my:meeting><my:attendees><my:attendee><my:attendeeName /><my:attendeeEmail /></my:attendee></my:attendees><my:discussionItems><my:discussionItem
xsi:nil="true" /></my:discussionItem></my:discussionItems><my:actionItems><my:actionItem><my:actionItemTitle /><my:actionItemAssignedTo /><my:actionItemDueBy
xsi:nil="true" /></my:actionItem></my:actionItems><my:ProductID>PR0001</my:ProductID><my:ProductType>PR</my:ProductType><my:Entity>CN</my:Entity><my:CoveragePeriod>20</my:Covera

<?mso-infoPathSolution name="urn:schemas-microsoft-com:office:infopath:MeetingOuput:-myXSD-2005-10-21T21-12-27" solutionVersion="1.0.0.13" productVersion="12.0.0.0" PIVersion="1.0.0.0" hr
><my:meetingAgenda xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/200:
US"><my:meeting><my:meetingTitle>0002</my:meetingTitle><my:meetingOrganizer>AXA</my:meetingOrganizer><my:meetingLocation>CN</my:meetingLocation><my:meetingDate>2011-06-09</my:
xsi:nil="true" /><my:meetingObjective /></my:meeting><my:attendees><my:attendee><my:attendeeName /><my:attendeeEmail /></my:attendee></my:attendees><my:discussionItems><my:discussionItem
xsi:nil="true" /></my:discussionItem></my:discussionItems><my:actionItems><my:actionItem><my:actionItemTitle /><my:actionItemAssignedTo /><my:actionItemDueBy
xsi:nil="true" /></my:actionItem></my:actionItems><my:ProductID>PR0002</my:ProductID><my:ProductType>PR</my:ProductType><my:Entity>CN</my:Entity><my:CoveragePeriod>53</my:Covera

<?mso-infoPathSolution name="urn:schemas-microsoft-com:office:infopath:MeetingOuput:-myXSD-2005-10-21T21-12-27" solutionVersion="1.0.0.13" productVersion="12.0.0.0" PIVersion="1.0.0.0" hr
><my:meetingAgenda xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/200:
US"><my:meeting><my:meetingTitle>0003</my:meetingTitle><my:meetingOrganizer>IBM</my:meetingOrganizer><my:meetingLocation>HK</my:meetingLocation><my:meetingDate>2011-06-09</my:m
xsi:nil="true" /><my:meetingObjective /></my:meeting><my:attendees><my:attendee><my:attendeeName /><my:attendeeEmail /></my:attendee></my:attendees><my:discussionItems><my:discussionItem
xsi:nil="true" /></my:discussionItem></my:discussionItems><my:actionItems><my:actionItem><my:actionItemTitle /><my:actionItemAssignedTo /><my:actionItemDueBy
```

Step2:Try XQuery. It is work!

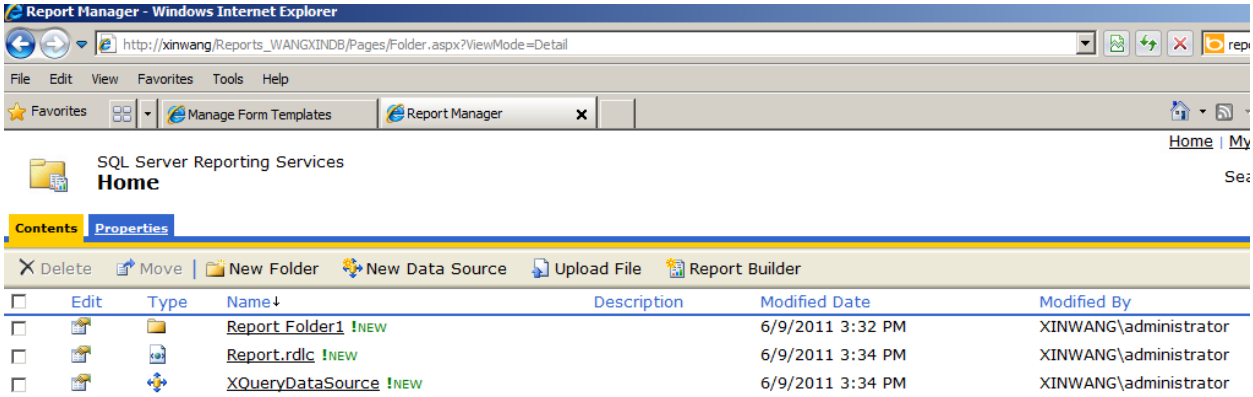
```
WITH XMLNAMESPACES
( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27
AS "my")
SELECT
xmlinfo.value('(/my:meetingAgenda/my:ProductID)[1]','nvarchar(50)') as
ProductID,
xmlinfo.value('(/my:meetingAgenda/my:ProductType)[1]','nvarchar(50)') as
ProductType ,
xmlinfo.value('(/my:meetingAgenda/my:Entity)[1]','nvarchar(50)') as Entity ,
xmlinfo.value('(/my:meetingAgenda/my:CoveragePeriod)[1]','int') as
CoveragePeriod ,
xmlinfo.value('(/my:meetingAgenda/my:PremiumPeriod)[1]','int') as
PremiumPeriod ,
xmlinfo.value('(/my:meetingAgenda/my:IssueAge)[1]','int') as IssueAge ,
xmlinfo.value('(/my:meetingAgenda/my:AttainedAge)[1]','int') as AttainedAge
FROM dbo.XQueryTable1
```

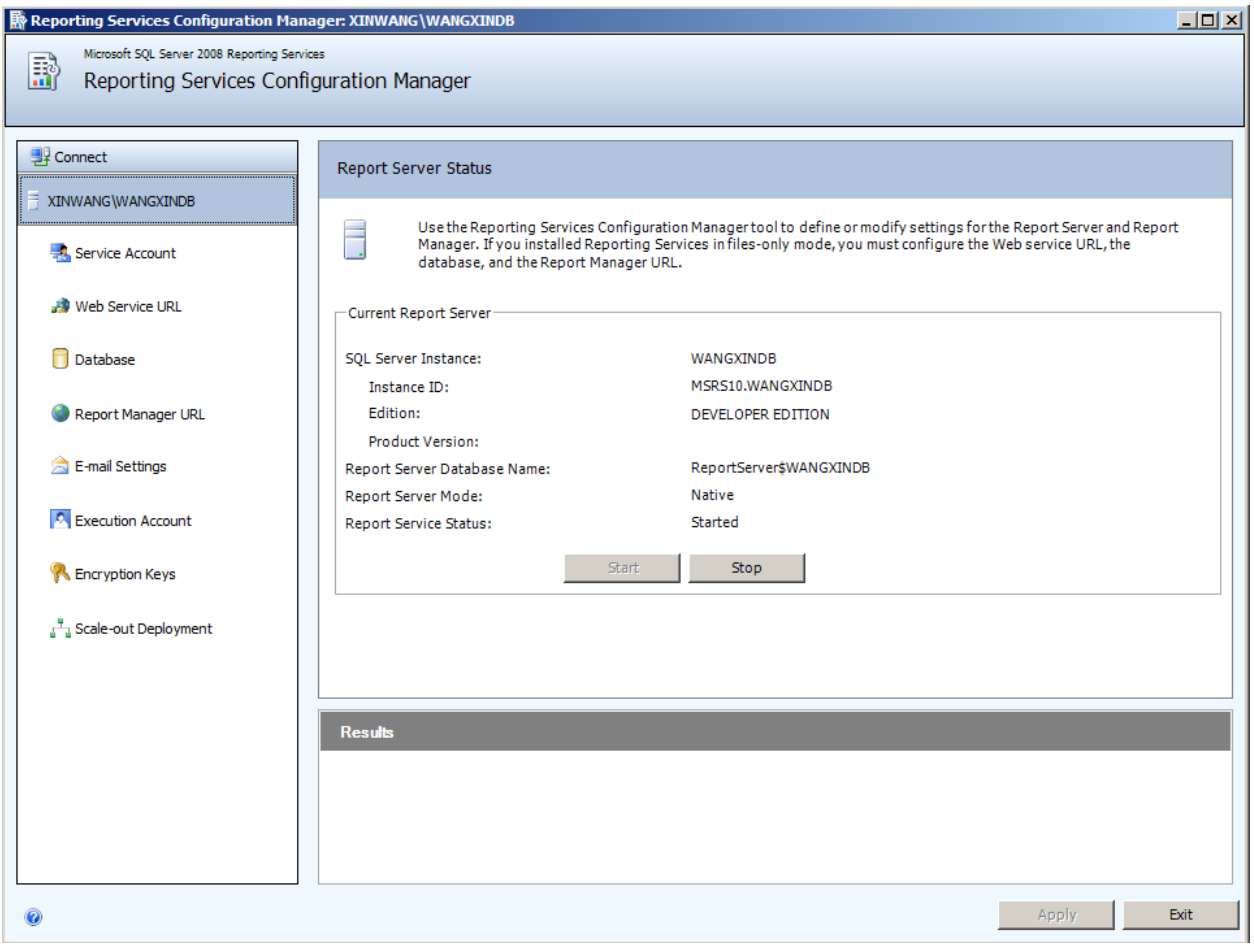
Basic Search | Ignore Blank Search

ProductID	ProductType	Entity	CoveragePeriod	PremiumPeriod	IssueAge	AttainedAge
PR0001	PR	CN	20	40	10	23
PR0002	PR	CN	53	23	85	66
PR0003	PR	HK	84	84	28	27
HY0004	HY	HK	20	83	64	34
HY0005	HY	USA	87	47	54	33

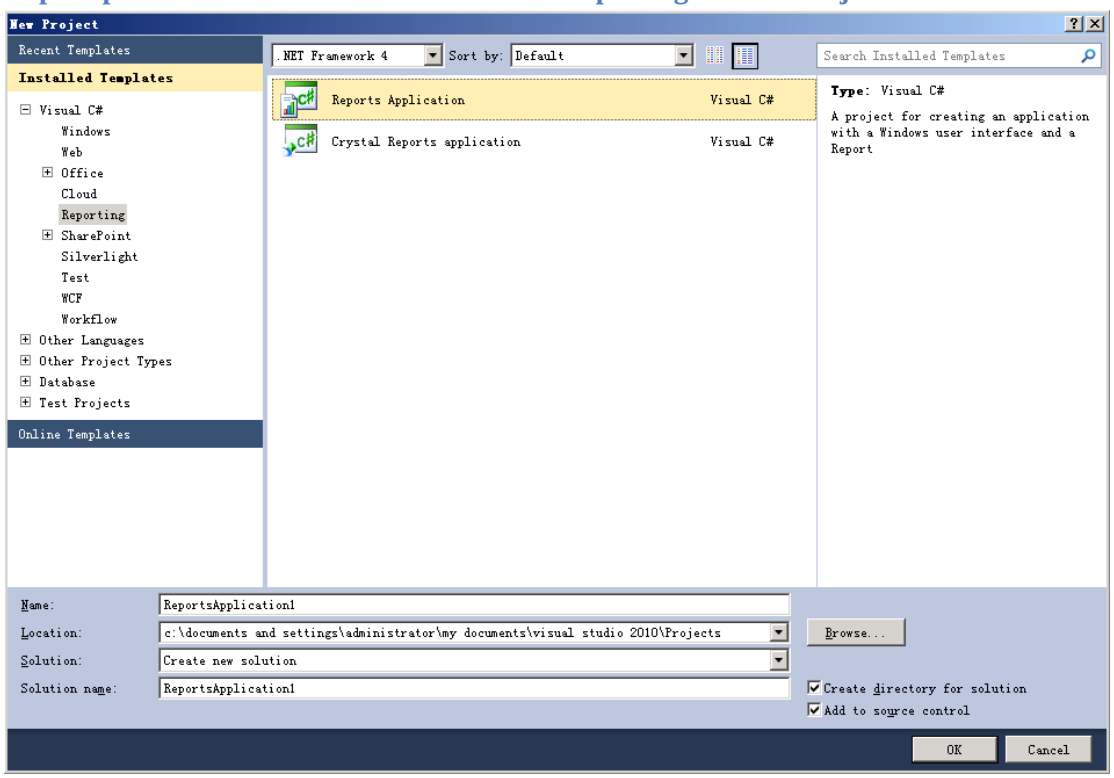
1.3 XQuery Report Project

Step1:Install and config SQL Server Reporting Service

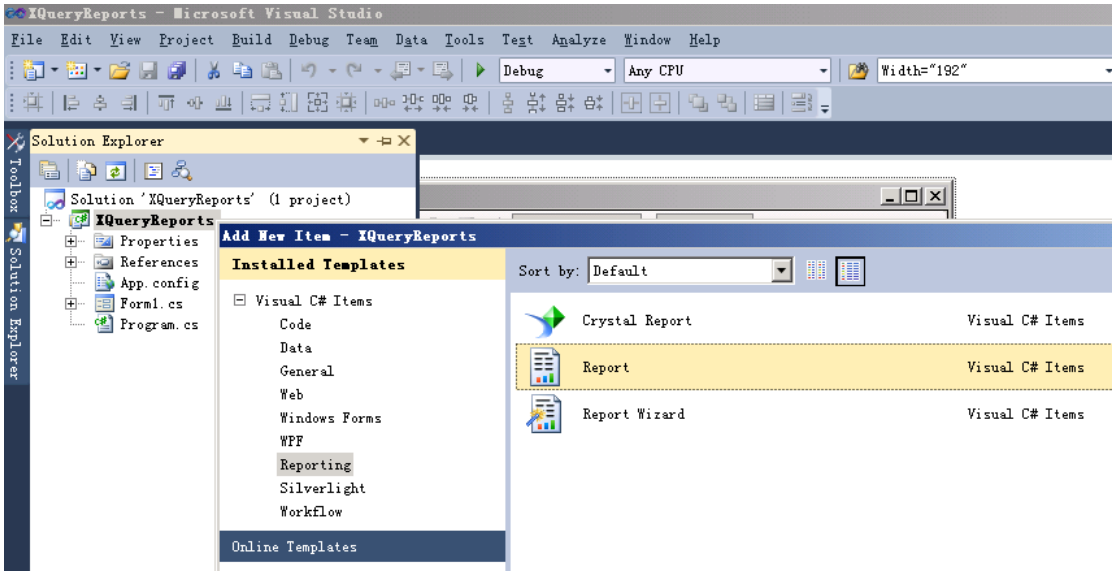



Reporting Services Configuration Manager: XINWANG\WANGXINDB
Microsoft SQL Server 2008 Reporting Services
Reporting Services Configuration Manager
Connect
XINWANG\WANGXINDB
Service Account
Web Service URL
Database
Report Manager URL
E-mail Settings
Execution Account
Encryption Keys
Scale-out Deployment
Report Server Status
Use the Reporting Services Configuration Manager tool to define or modify settings for the Report Server and Report Manager. If you installed Reporting Services in files-only mode, you must configure the Web service URL, the database, and the Report Manager URL.
Current Report Server
SQL Server Instance: WANGXINDB
Instance ID: MSRS10.WANGXINDB
Edition: DEVELOPER EDITION
Product Version:
Report Server Database Name: ReportServer$WANGXINDB
Report Server Mode: Native
Report Service Status: Started
Start
Stop
Results
Apply
Exit

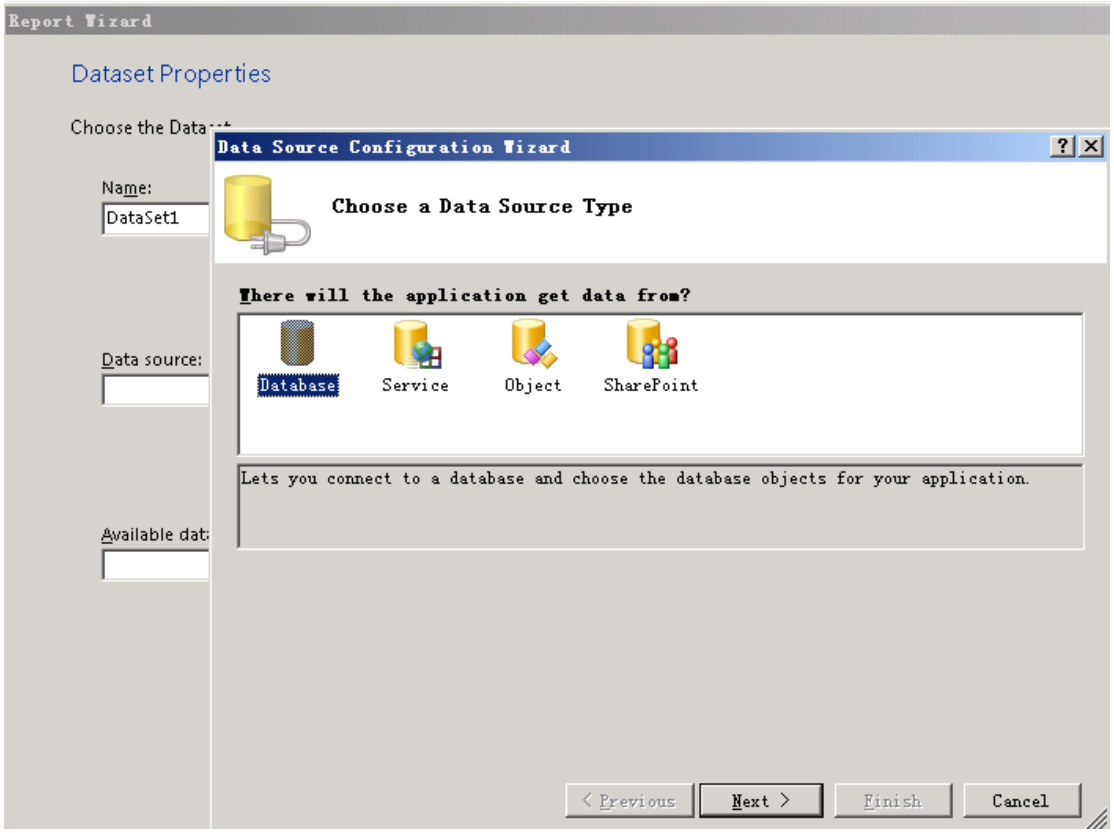
Step2:Open .Net Visual Studio Create a new Reporting Service Project



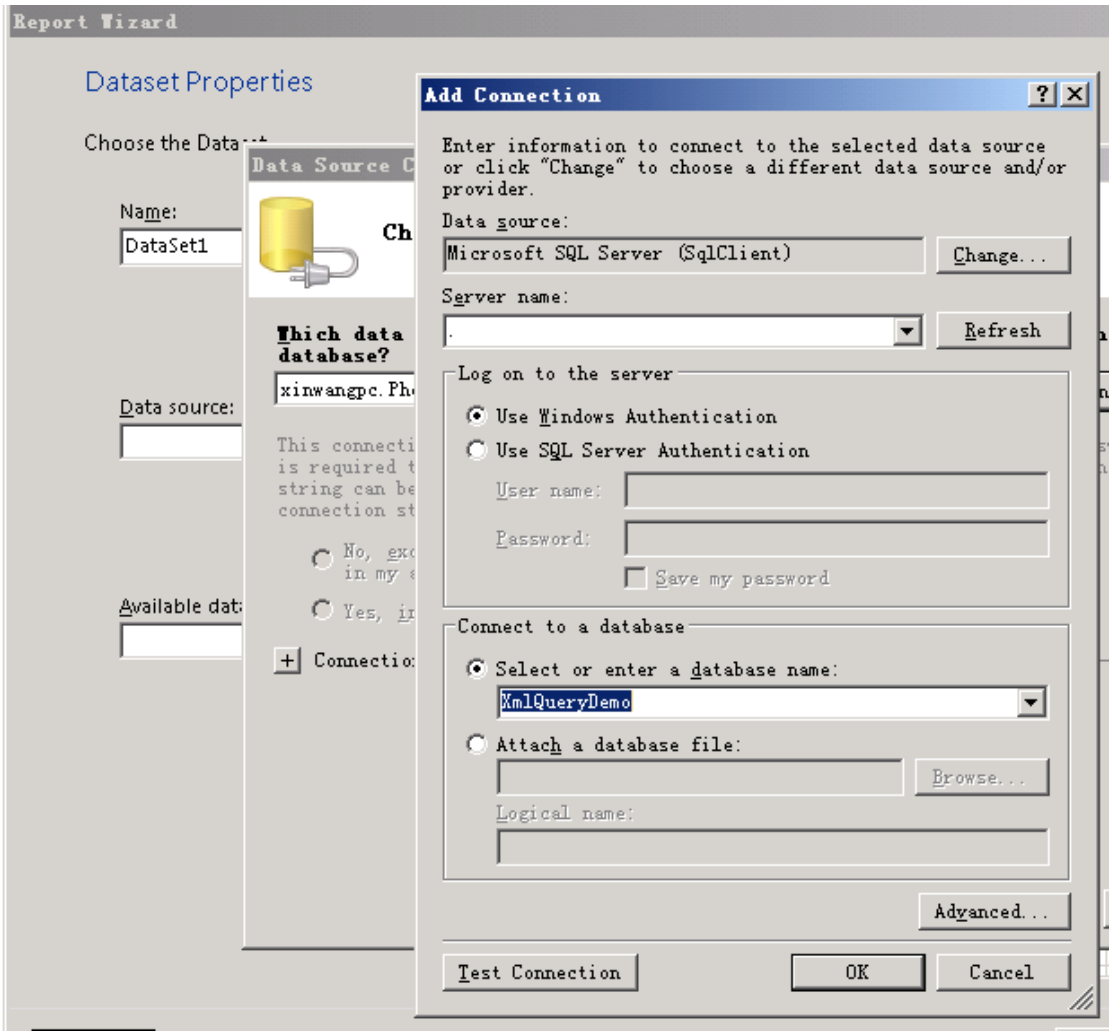
Create a new report item



Step3:Datasource config



If choose Database as datasource:



Note:

- XQuery can't refers to SQL Server View and Produce,So if use Reporting Service with XQuery,Must not choose Database data source;
- SharePoint data source is appoint a SharePoint List as data source,Xml file nodes must map to list field,5 MC ,total more than 1000*5 fields,so SharePoint data source could not be choosed also.
- So must choose Object or Service as data source, Service return value must to convert to Entity Object,so in this demo the best choice is Object as data source.

Step4:Create business objects to use as a data source.

Business Objects code here:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;

namespace XQueryReportDemo
{
    public class Product
    {
        private string _ProductID;

        public string ProductID
        {
            get { return _ProductID; }
            set { _ProductID = value; }
        }
        private string _ProductType;

        public string ProductType
        {
            get { return _ProductType; }
            set { _ProductType = value; }
        }
        private string _Entity;

        public string Entity
        {
            get { return _Entity; }
            set { _Entity = value; }
```

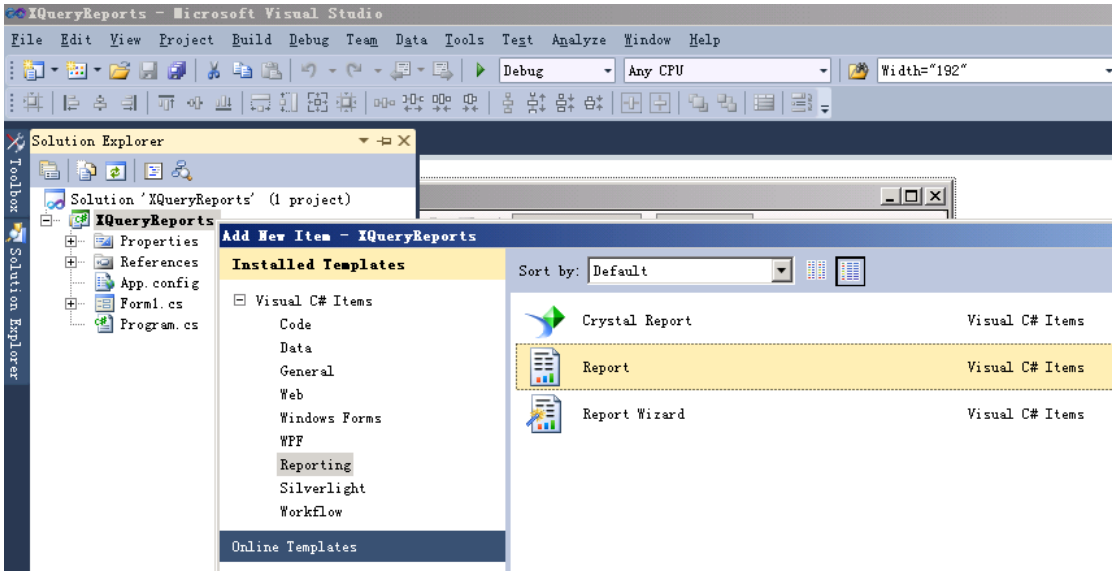
```
}
private int _CoveragePeriod;

public int CoveragePeriod
{
    get { return _CoveragePeriod; }
    set { _CoveragePeriod = value; }
}
private int _PremiumPeriod;

public int PremiumPeriod
{
    get { return _PremiumPeriod; }
    set { _PremiumPeriod = value; }
}
private int _IssueAge;

public int IssueAge
{
    get { return _IssueAge; }
    set { _IssueAge = value; }
}
private int _AttainedAge;

public int AttainedAge
{
    get { return _AttainedAge; }
    set { _AttainedAge = value; }
}
```

```
        public Product()
        {
        }



    }

    public class AXAProducts
    {
        private List<Product> m_products;

        public AXAProducts()
        {

            m_products = new List<Product>();
            string sql = @"WITH XMLNAMESPACES( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27' AS ""my"") ";
            sql += @"SELECT ";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:ProductID)[1]','nvarchar(50)') as ProductID,";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:ProductType)[1]','nvarchar(50)') as ProductType , ";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:Entity)[1]','nvarchar(50)') as Entity , ";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:CoveragePeriod)[1]','int') as CoveragePeriod ,";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:PremiumPeriod)[1]','int') as PremiumPeriod ,";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:IssueAge)[1]','int') as IssueAge , ";
            sql += @"xmlinfo.value('(/my:meetingAgenda/my:AttainedAge)[1]','int') as AttainedAge";
            sql += @" FROM dbo.XQueryTable1";
            DataSet ds = new DataSet();
            ds = DataCommon.GetDataSet(sql);
```

```
        m_products = DataCommon.GetList<Product>(ds.Tables[0]);

    }

    public List<Product> GetProducts()
    {
        return m_products;
    }
  }

}
```

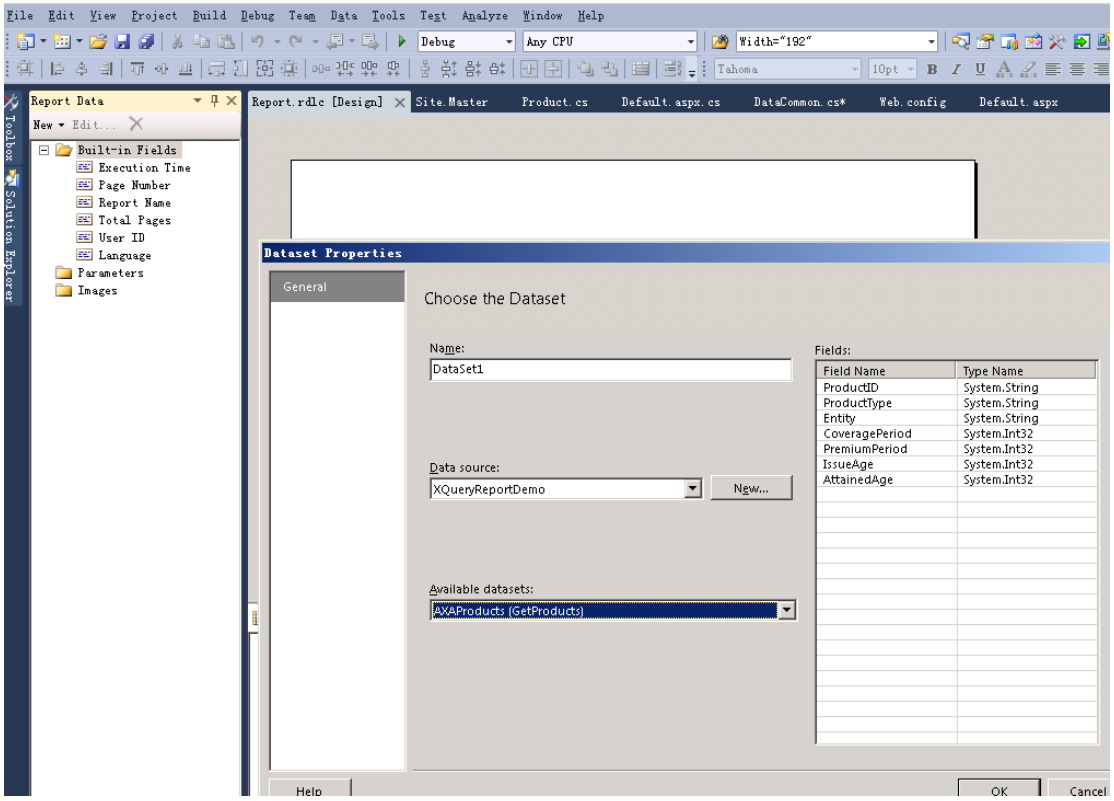
Step5:Add a report to the project

In the Add New Item dialog, select Report. Type a name for the report and click Add. The report is added to the project and automatically opened in Report Designer. The default name for the report is Report.rdlc.



Step6:Check the Website Data Sources

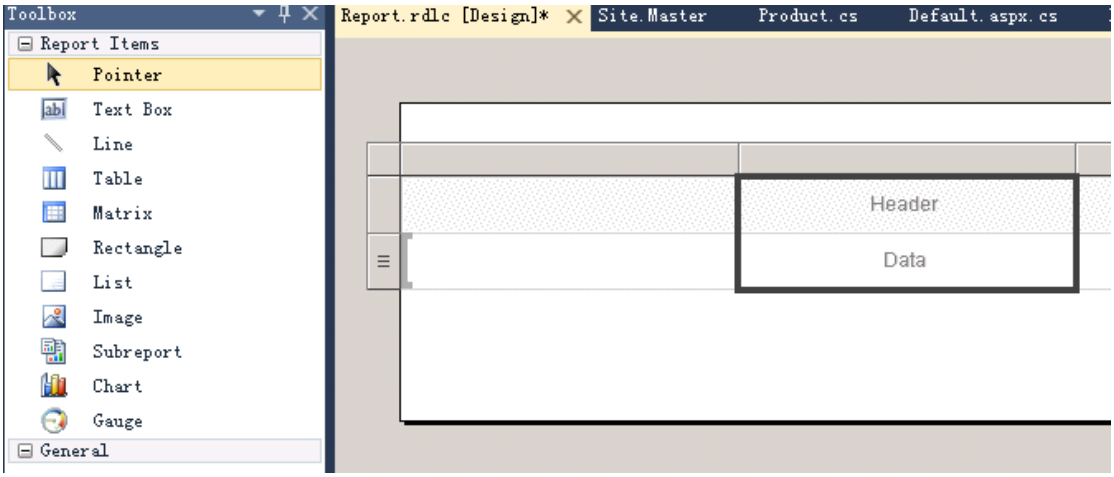
1.Click the Report.rdlc [Design] tab. In the left pane, click the Website Data Sources tab. If the Website Data Sources tab is not visible, from the Data menu, select Show Data Sources.



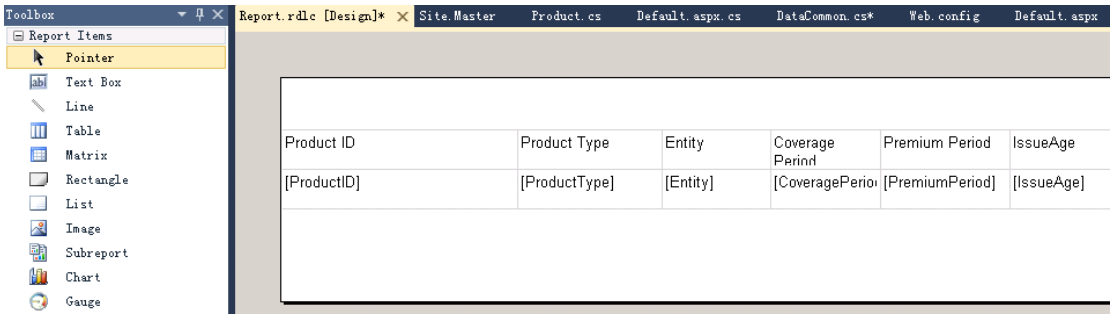
2.Confirm that the Product object and its public properties, appear in a hierarchy in the Website Data Sources window.

Step7:Design the report

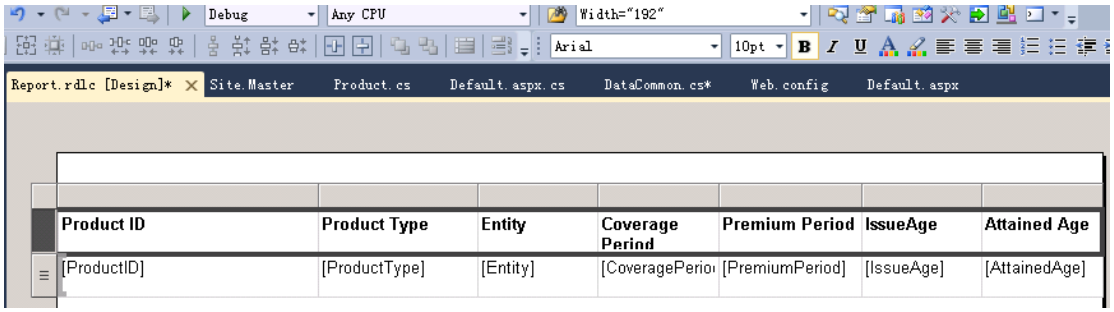
1. With the report open in Design mode, open the Toolbox. From the Toolbox, drag a Table control onto the report. The table control opens in a tabbed Design window.



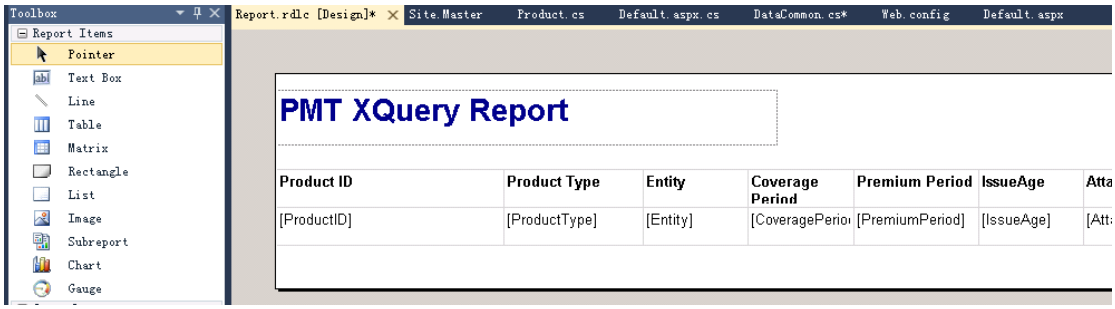
2.From the Data Sources window, drag the field from the Product data source onto the column of the Detail row of the table. The Detail row is the middle row. Notice that the Header row automatically fills in for you when you specify the Detail row.



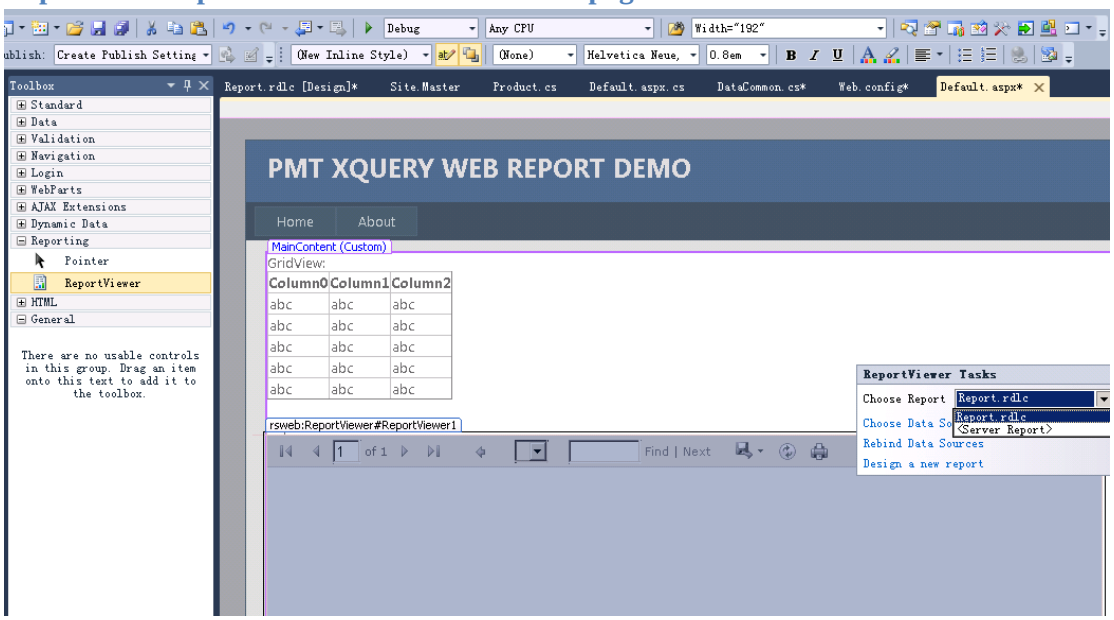
3.(Optional) Select the header row of the table by clicking on the left table header icon and apply the Bold font style.



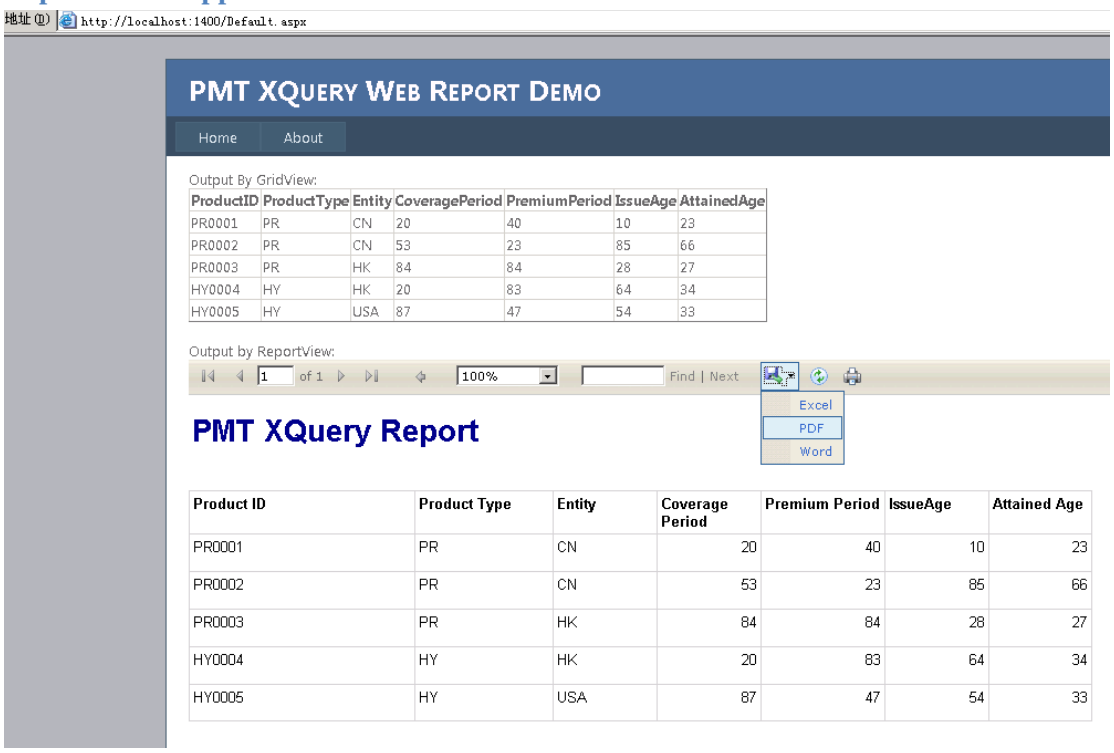
5.To add a title to the report, open the Toolbox and drag a Textbox onto the report. Position the Textbox above the table. Type the report name.



Step8:Add a ReportViewer control to the Web page



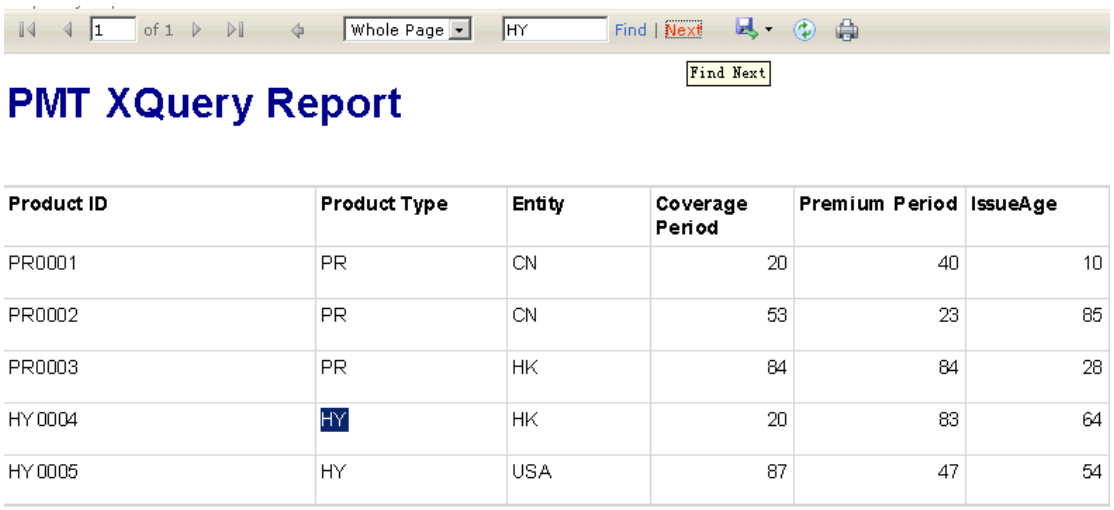
Step9:Run the application



ProductID	ProductType	Entity	CoveragePeriod	PremiumPeriod	IssueAge	AttainedAge
PR0001	PR	CN	20	40	10	23
PR0002	PR	CN	53	23	85	66
PR0003	PR	HK	84	84	28	27
HY0004	HY	HK	20	83	64	34
HY0005	HY	USA	87	47	54	33

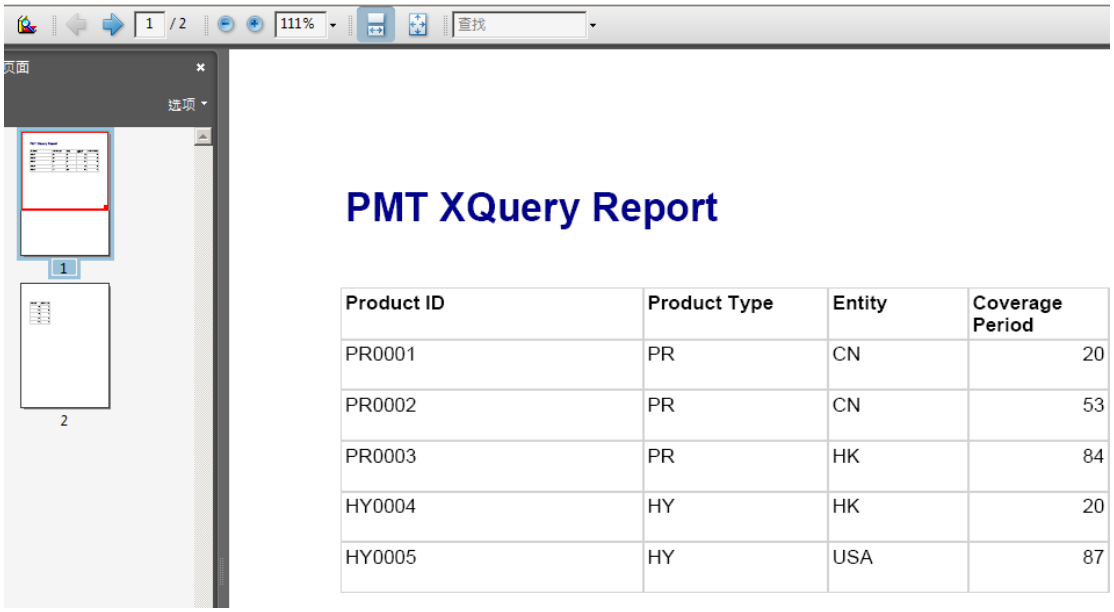
Product ID	Product Type	Entity	Coverage Period	Premium Period	IssueAge	Attained Age
PR0001	PR	CN	20	40	10	23
PR0002	PR	CN	53	23	85	66
PR0003	PR	HK	84	84	28	27
HY0004	HY	HK	20	83	64	34
HY0005	HY	USA	87	47	54	33

Find some word:



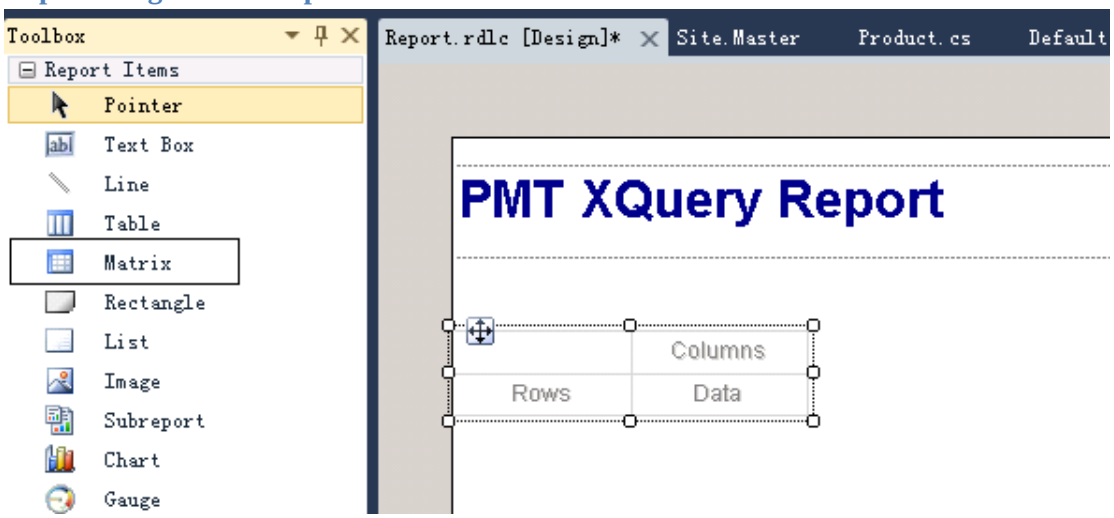
Product ID	Product Type	Entity	Coverage Period	Premium Period	IssueAge
PR0001	PR	CN	20	40	10
PR0002	PR	CN	53	23	85
PR0003	PR	HK	84	84	28
HY0004	HY	HK	20	83	64
HY0005	HY	USA	87	47	54

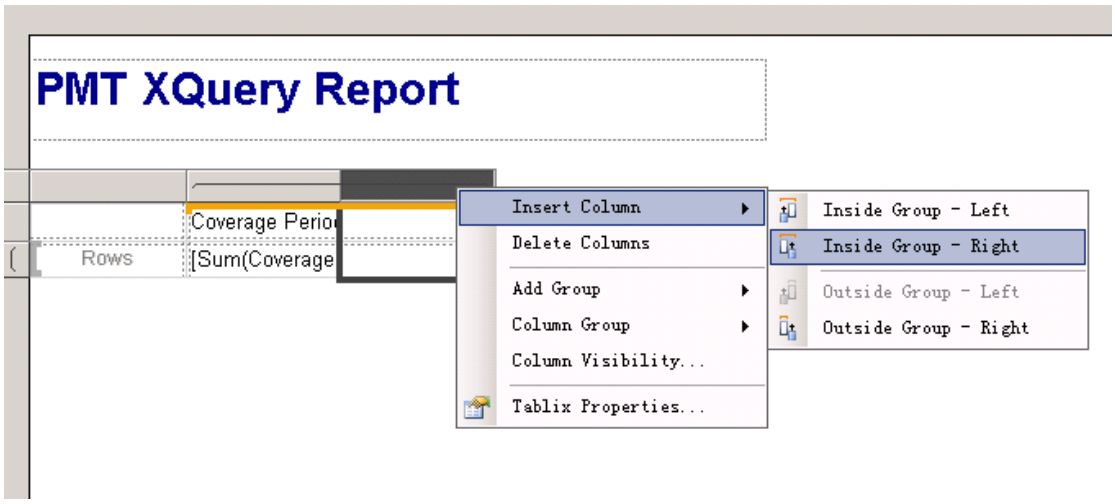
Save to PDF

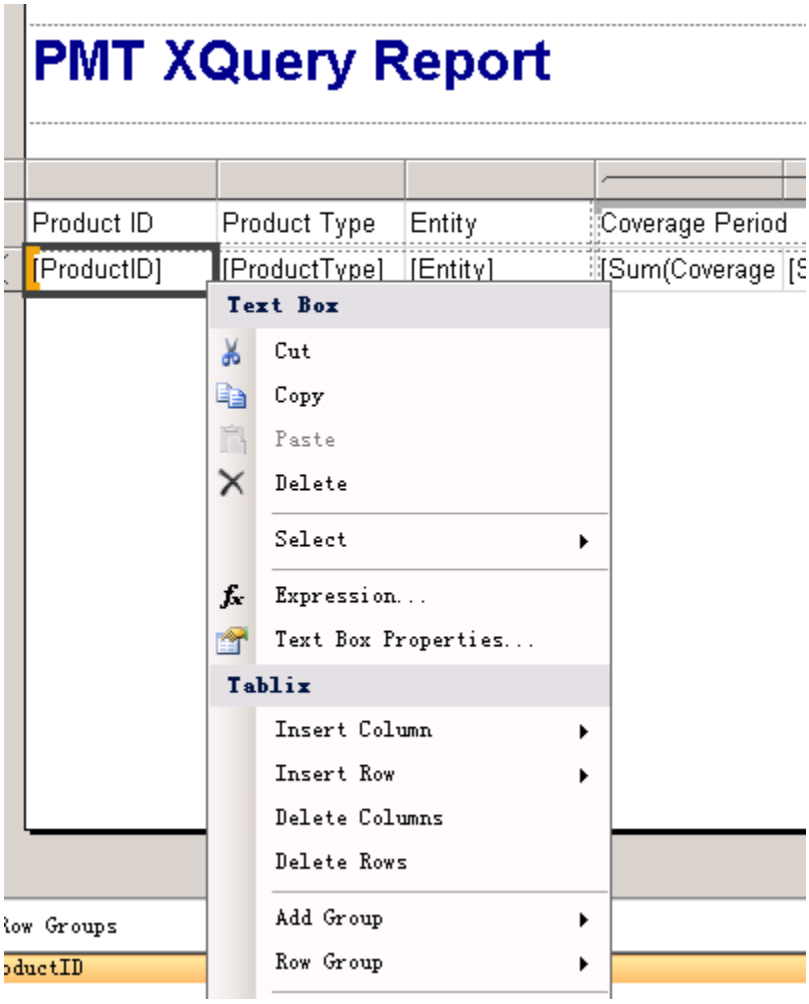


Product ID	Product Type	Entity	Coverage Period
PR0001	PR	CN	20
PR0002	PR	CN	53
PR0003	PR	HK	84
HY0004	HY	HK	20
HY0005	HY	USA	87

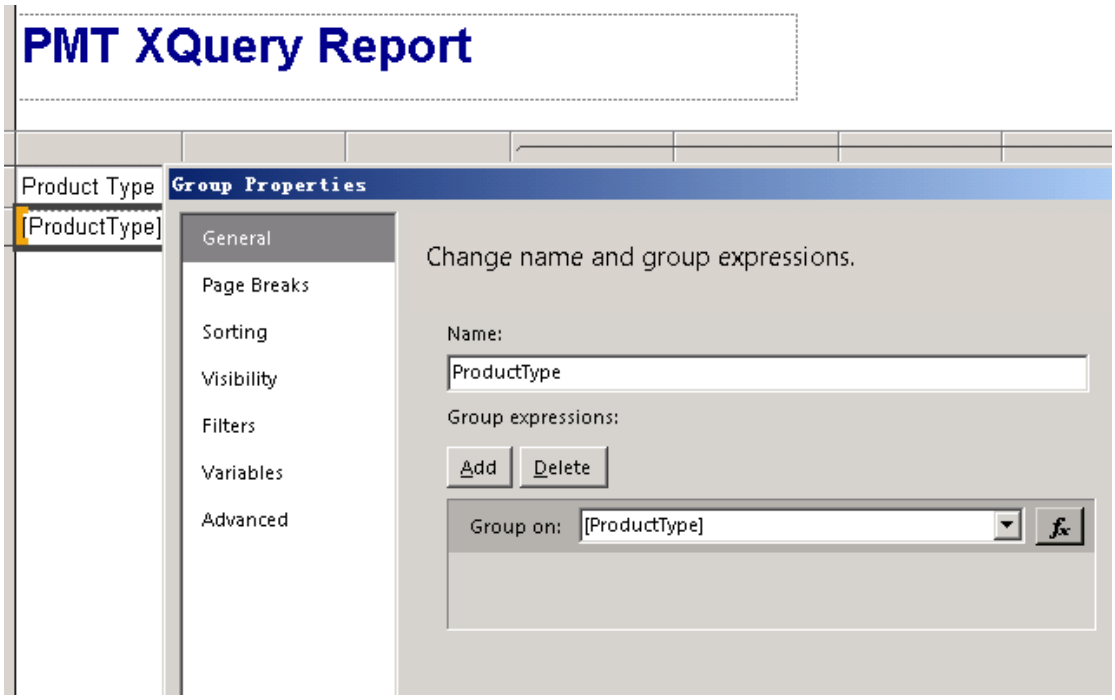
Step10:Design matrix report



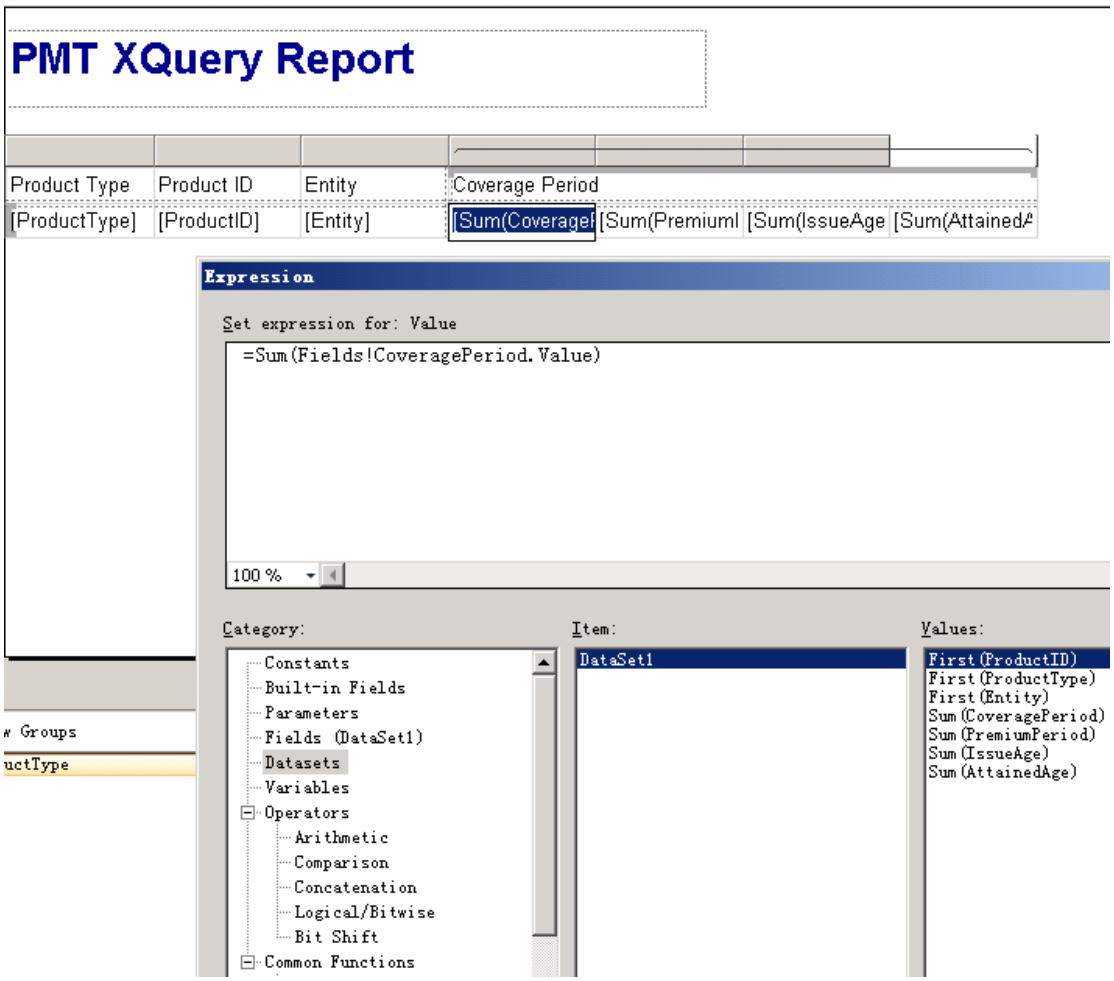

PMT XQuery Report
Coverage Perio
Rows
[Sum(Coverage
Insert Column
Delete Columns
Add Group
Column Group
Column Visibility...
Tablix Properties...
Inside Group - Left
Inside Group - Right
Outside Group - Left
Outside Group - Right


PMT XQuery Report
Product ID
Product Type
Entity
Coverage Period
[ProductID]
[ProductType]
[Entity]
[Sum(Coverage
Text Box
Cut
Copy
Paste
Delete
Select
Expression...
Text Box Properties...
Tablix
Insert Column
Insert Row
Delete Columns
Delete Rows
Add Group
Row Group

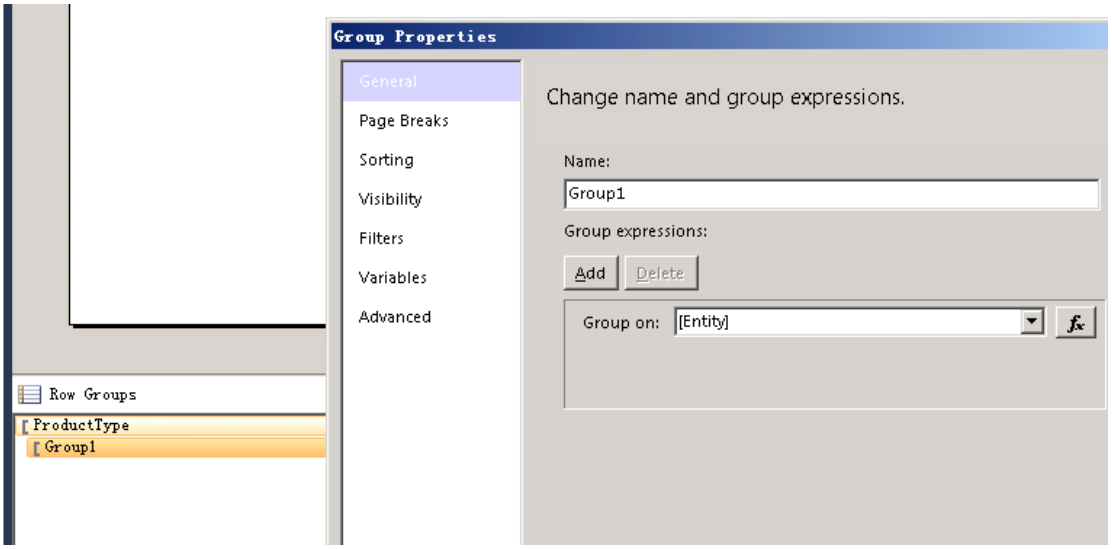
Group:


PMT XQuery Report
Product Type
[ProductType]
Group Properties
General
Page Breaks
Sorting
Visibility
Filters
Variables
Advanced
Change name and group expressions.
Name:
ProductType
Group expressions:
Add
Delete
Group on:
[ProductType]

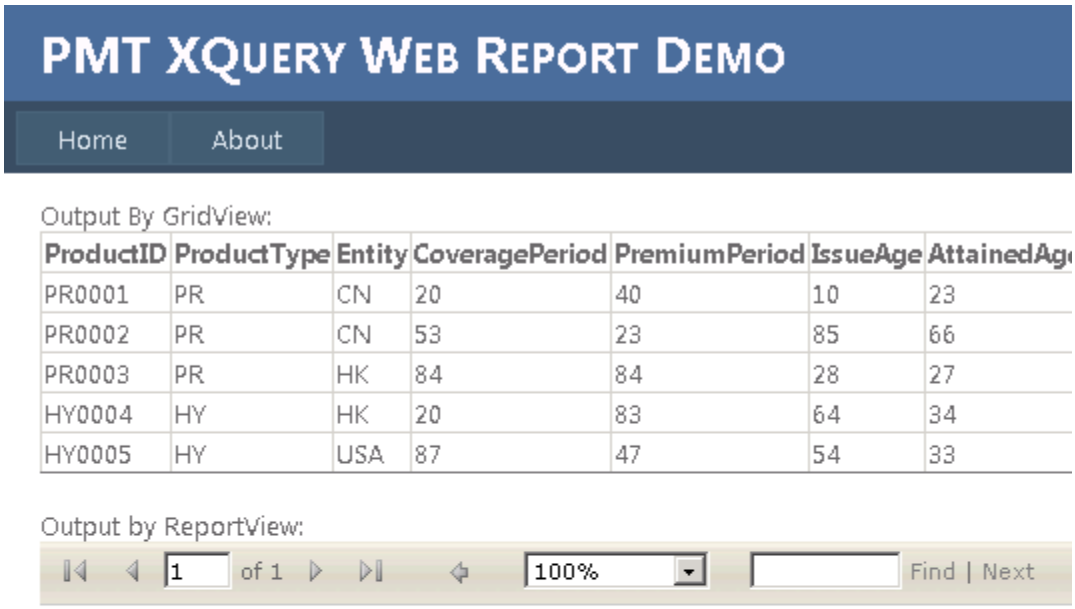
Expression:


PMT XQuery Report
Product Type
Product ID
Entity
Coverage Period
[ProductType]
[ProductID]
[Entity]
[Sum(Coverage
[Sum(Premiuml
[Sum(IssueAge
[Sum(Attained
Expression
Set expression for: Value
=Sum(Fields!CoveragePeriod.Value)
100 %
Category:
Constants
Built-in Fields
Parameters
Fields (DataSet1)
Datasets
Variables
Operators
Arithmetic
Comparison
Concatenation
Logical/Bitwise
Bit Shift
Common Functions
Item:
DataSet1
Values:
First(ProductID)
First(ProductType)
First(Entity)
Sum(CoveragePeriod)
Sum(PremiumPeriod)
Sum(IssueAge)
Sum(AttainedAge)
Groups
uctType

Child group:


Group Properties
General
Page Breaks
Sorting
Visibility
Filters
Variables
Advanced
Change name and group expressions.
Name:
Group1
Group expressions:
Add
Delete
Group on:
[Entity]
Row Groups
ProductType
Group1

Step11:Run



ProductID	ProductType	Entity	CoveragePeriod	PremiumPeriod	IssueAge	AttainedAge
PR0001	PR	CN	20	40	10	23
PR0002	PR	CN	53	23	85	66
PR0003	PR	HK	84	84	28	27
HY0004	HY	HK	20	83	64	34
HY0005	HY	USA	87	47	54	33

PMT XQuery Matrix Report

Product Type	Entity	CoveragePeriod	PremiumPeriod	IssueAge
HY	HK	20	83	64
	USA	87	47	54
PR	CN	73	63	95
	HK	84	84	28

Chart report:

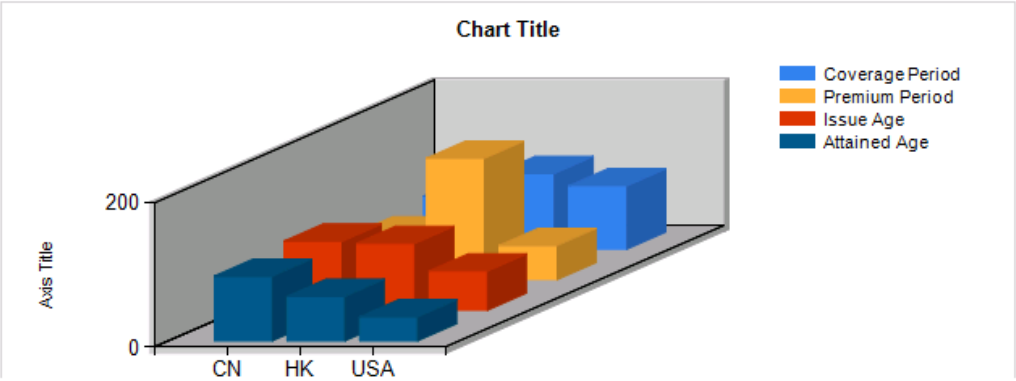
Output By GridView:

ProductID	ProductType	Entity	CoveragePeriod	PremiumPeriod	IssueAge	AttainedAge
PR0001	PR	CN	20	40	10	23
PR0002	PR	CN	53	23	85	66
PR0003	PR	HK	84	84	28	27
HY0004	HY	HK	20	83	64	34
HY0005	HY	USA	87	47	54	33

Output by ReportView:

1 of 1 100% Find | Next

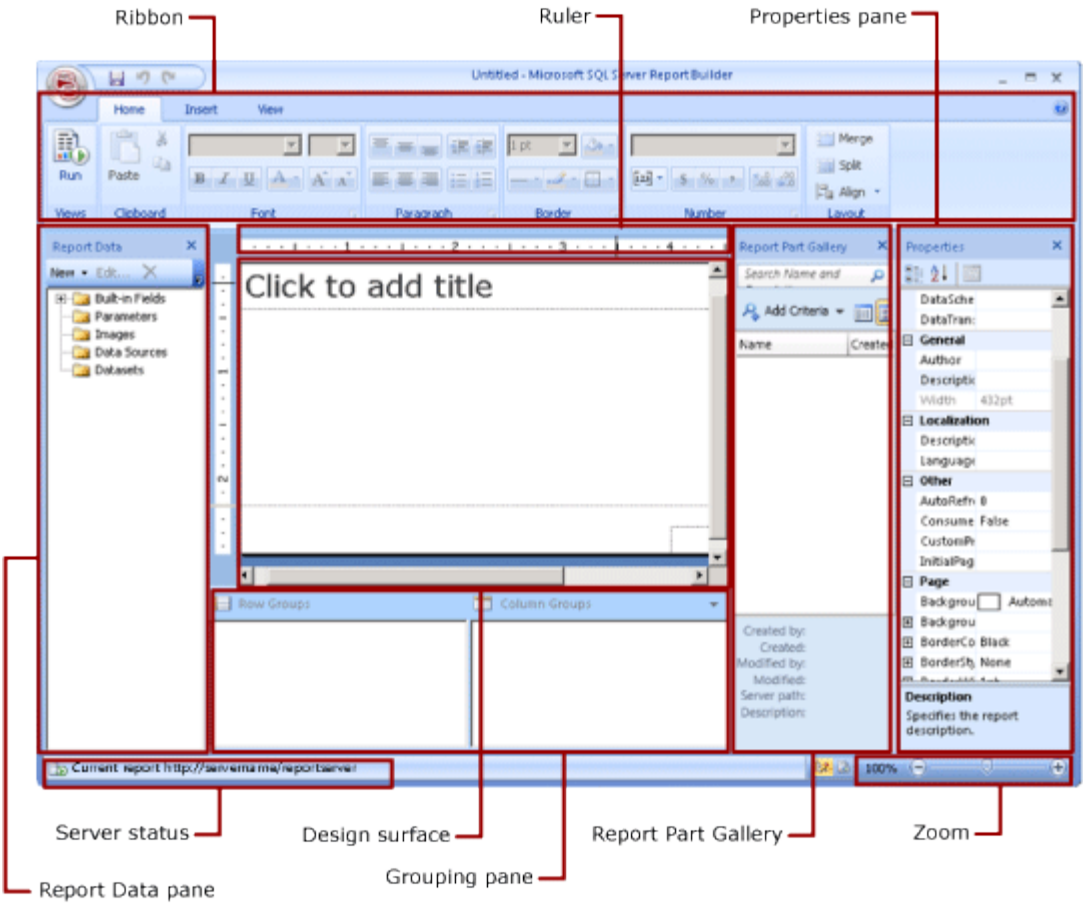
Product Type	Entity	CoveragePeriod	PremiumPeriod	IssueAge
HY	HK	20	83	64
	USA	87	47	54
PR	CN	73	63	95
	HK	84	84	28



2. Create Report By Report Builder

Report Builder 3.0 is a report authoring environment for business users who prefer to work in the Microsoft Office environment. When you design a report, you specify where to get the data, which data to get, and how to display the data. When you run the report, the report processor takes all the information you have specified, retrieves the data, and combines it with the report layout to generate the report. You can preview your reports in Report Builder, or you can publish your report to a report server or a report server in SharePoint integrated mode, where others can run it.

The report in this illustration features a matrix with row and column groups, indicators, and a summary pie chart in the corner cell, accompanied by a map with two sets of geographic data represented by color and by circle size.



3. XQuery Introduction

3.1 Methods of the XML Data Type

The XML data type supports five methods that can be used to manipulate XML instances. The methods of the XML data type can be described as follows:

- The **query()** method takes an XQuery expression that evaluates to a list of XML nodes and allows the user to extract fragments of an XML document. The result of this method is an instance of untied XML.
- The **value()** method is useful for extracting scalar values from XML documents as a relational value. This method takes an XQuery expression that identifies a single node and the desired SQL type to be returned. The value of the XML node is returned cast to the specified SQL type.
- The **exist()** method allows the user to perform checks on XML documents to determine if the result of an XQuery expression is empty or nonempty. The result of this method is 1 if the XQuery expression returns a nonempty result, 0 if the result is empty, and NULL if the XML instance itself is NULL.
- The **nodes()** method accepts an XQuery expression and returns a rowset in which each row represents a context node identified by the query expression. Methods of the XML data type such as **query()**, **value()**, **exist()**, and **nodes()** can also be invoked on the context nodes returned from the **nodes()** method.
- The **modify()** method can be used to modify the content of an XML document. It accepts XML DML statements to insert, update, or delete one or more nodes from an XML instance. It raises an error if applied to a NULL value.

3.2 XPath 2.0 Expressions

XQuery uses XPath 2.0 expressions to locate nodes in a document and to navigate from one location to another within a single document or across documents. Navigation paths defined using XPath consist of a sequence of steps separated by /. A single step comprises an axis, a node test, and zero or more step qualifiers.

The *axis* specifies the direction of movement, relative to the context node. Supported axes in SQL Server 2005 are **child**, **descendant**, **parent**, **attribute**, **self** and **descendant-or-self**.

This example expression uses the column Resume which is of type XML in table [HumanResources].[JobCandidate] from the AdventureWorks database for the purpose of illustrating the concept of path expressions. The following path expression selects all address nodes for which the address type is set to Home.

```
//child::ns:Addr.Type[.="Home"]/parent::node()
```

In the preceding path expression,

- **child** is the axis specifier.
- **::** is the axis separator.
- **ns** is the namespace prefix.
- **Addr.Type** is the node test.
- **[.="Home"]** is the predicate expression where . refers to the context node.

XQuery also supports abbreviated syntax for specifying the axis. The following table shows the axis and corresponding abbreviated syntax.

Table 2 Abbreviated syntax for axes

Axis	Abbreviated form
Attribute	@
Child	
descendant-or-self::node()	//
parent::node()	..
self::node()	.

3.3 Query Example

3.3.1 Easy Query

```
SELECT
xmlinfo.query('/stuInfo[age>17]') as Result ,
xmlinfo.query('/stuInfo/name') as name ,
xmlinfo.value('(/stuInfo/age)[1]','nvarchar(3)') as age
From [XQueryTable]
```

Reslut:

Results | Messages

	Result	name	age
1			NULL
2	<stuInfo><name>Tom</name><age>18</age><sex>Male</sex><specialty>HY</specialty></stuInfo>	<name>Tom</name>	18
3	<stuInfo><name>Jack</name><age>18</age><sex>Male</sex><specialty>VA</specialty></stuInfo>	<name>Jack</name>	18
4	<stuInfo><name>Jack</name><age>18</age><sex>Famale</sex><specialty>VA</specialty></stuInfo>	<name>Jack</name>	18
5			NULL
6	<stuInfo><name>Jane</name><age>23</age><sex>Famale</sex><specialty>HY</specialty></stuInfo>	<name>Jane</name>	23
7			NULL
8			NULL
9			NULL

```sql
SELECT
      xmlinfo.query('/stuInfo[age>18]') as Result ,
      xmlinfo.query('/stuInfo/name') as name ,
      xmlinfo.value('(/stuInfo/age)[1]','nvarchar(3)') as age
From [XQueryTable]
```

Reslut:

Results | Messages

	Result	name	age
1			NULL
2		<name>Tom</name>	18
3		<name>Jack</name>	18
4		<name>Jack</name>	18
5			NULL
6	<stuInfo><name>Jane</name><age>23</age><sex>Famale</sex><specialty>HY<...	<name>Jane</name>	23
7			NULL
8			NULL
9			NULL

Notice: I write a procedure to filter blank rows, the effect is:

```sql
SELECT xmlinfo.query('/stuInfo[age>18]') as
Result,xmlinfo.query('/stuInfo/name') as name , xmlinfo.value
('(/stuInfo/age)[1]','nvarchar(3)') as age from XQueryTable
```

Basic Search | Ignore Blank Search

Result
<stuInfo><name>Jane</name><age>23</age><sex>Famale</sex><specialty>HY</specialty></stuInf

Notice: if don't want to show the xml format strings ,just show the node value only,must use value method. query like this:

```sql
SELECT
      xmlinfo.value('(/stuInfo/name)[1]','nvarchar(max)') as name ,
      xmlinfo.value('(/stuInfo/age)[1]','nvarchar(3)') as age,
      xmlinfo.query('/stuInfo[age>18]') as Result
From [XQueryTable]
```

```
SELECT
xmlinfo.value('(/stuInfo/name)[1]','nvarchar(max)') as name ,
xmlinfo.value('(/stuInfo/age)[1]','nvarchar(3)') as age,
xmlinfo.query('/stuInfo[age>18]') as Result
From [XQueryTable]
```

Basic Search | Ignore Blank Search

name	age	Result
Jane	23	<stuInfo><name>Jane</name><age>23</age><sex>Famale</sex><spec:

Other query:

```
SELECT xmlinfo.value('(/stuInfo/name)[1]','nvarchar(max)') as name from
XQueryTable
SELECT xmlinfo.query('/stuInfo[name="Jean"]') as NameisJean from XQueryTable
SELECT xmlinfo.exist('/stuInfo/name') as result from dbo.XQueryTable
```

3.3.2 Statement With namespace

Namespace

```
WITH XMLNAMESPACES( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27'
AS "my")
SELECT
xmlinfo.query('/my:meetingAgenda/my:meeting[my:meetingOrganizer="AXA"] ') as Result
FROM dbo.XQueryTable
```

```
WITH XMLNAMESPACES
( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/20
05-10-21T21:12:27' AS "my")
SELECT
xmlinfo.query('/my:meetingAgenda/my:meeting
```

Basic Search | Ignore Blank Search

Result
<my:meeting xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27"><my:meetingTitle>a</my:meetingTitle><my:meetingOrganizer>AXA</my:meetingOrganizer><my:meetingLocation>001</my:meetingLocation><my:meetingDate>2011-05-30</my:meetingDate><my:meetingStartTime xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" /><my:meetingEndTime xmlns:xsi="http://www.w3.org/2001/XMLSchema xsi:nil="true" /><my:meetingObjective /></my:meeting>
<my:meeting xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27"><my:meetingTitle>b</my:meetingTitle><my:meetingOrganizer>AXA</my:meetingOrganizer><my:meetingLocation>002</my:meetingLocation><my:meetingDate>2011-05-30</my:meetingDate><my:meetingStartTime xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" /><my:meetingEndTime xmlns:xsi="http://www.w3.org/2001/XMLSchema xsi:nil="true" /><my:meetingObjective /></my:meeting>

For

The **for** clause in a FLWOR expression enables users to define a declarative iteration of a bound variable over an input sequence. The input sequence can be specified using XPath expressions, sequence of atomic values, a sequence constructed using literals, or constructor functions. It is therefore analogous to the SQL SELECT FROM clause and is not like a programming language "for" construct.

Variable binding is also specified in the **for** clause.

Example: Selecting all home address elements from resume using the for clause

The following query selects all **Address** nodes where the type of address is set to **Home** and the **JobCandidateID** is 3:

```
SELECT Resume.query('
  declare namespace RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
  for $A in /RES:Resume/RES:Address/RES:Addr.Type[.="Home"]/..
  return
  $A
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3

WITH XMLNAMESPACES( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27'
AS "my")
SELECT
xmlinfo.query(' for $A in /my:meetingAgenda/my:meeting[my:meetingOrganizer="AXA"]/.. return $A ') as Result
FROM dbo.XQueryTable
```

where

The **where** clause filters the results of an iteration by applying the expression specified with the **where** clause.

Example: Selecting all home address elements using the where clause

The following query selects all **Address** nodes where the type of address is set to **Home** and the **JobCandidateID** is 3:

```
SELECT Resume.query('
  declare namespace RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
  for $A in /RES:Resume/RES:Address
  where $A/RES:Addr.Type[.="Home"]
  return
  $A
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3
```

```
WITH XMLNAMESPACES( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27'
AS "my")
SELECT
xmlinfo.query('
for $A in /my:meetingAgenda/my:meeting
  where $A/my:meetingOrganizer[.="AXA"]
  return
  $A
 ')
as Result FROM dbo.XQueryTable
```

order by

The **order by** keyword enables you to sort the values in the returned result set. The **order by** keyword accepts a sorting expression, which should return an atomic value. Optionally, you can also specify ascending or descending for the sort order. The default sort order is ascending.

Example: Selecting employment history in ascending order using the order by clause

The following query selects all **Employment** nodes in ascending order of employment starting date for a candidate whose **JobCandidateID** is 3:

```
SELECT Resume.query('
  declare namespace RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
  for $EMP in /RES:Resume/RES:Employment
  order by $EMP/RES:Emp.StartDate
  return
  $EMP
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3
```

return

The **return** clause, which is analogous to the SELECT clause in SQL, enables users to define the result of a query. You can specify any valid XQuery expression in the **return** clause. You can also construct XML structures in the **return** section by specifying constructors for elements, attributes, etc.

Example: Selecting specific elements of employment history using the return clause

The following query selects StartDate, EndDate, OrgName, JobTitle elements of **Employment** node for a candidate whose **JobCandidateID** is 3:

```
SELECT Resume.query('
  declare namespace RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
  for $EMP in /RES:Resume/RES:Employment
  order by $EMP/RES:Emp.StartDate
```

```
  return
    <Employment>
     { $EMP/RES:Emp.StartDate }
     { $EMP/RES:Emp.EndDate }
     { $EMP/RES:Emp.OrgName }
     { $EMP/RES:Emp.JobTitle }
    </Employment>
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3
```

3.3.3 Composite condition

```
WITH XMLNAMESPACES(
      'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27' AS
      "my")
SELECT
xmlinfo.query(' for $A in /my:meetingAgenda/my:meeting
where
   ($A/my:meetingOrganizer[.="AXA"] or $A/my:meetingOrganizer[.="IBM"])
   and
   ($A/my:meetingLocation[.="001"] )
   return $A/.')
as Result
FROM dbo.XQueryTable
```

3.3.4 Value comparison operators

Value comparison operators help compare atomic values. Value comparison operators supported by SQL Server are **eq**, **ne**, **lt**, **gt**, **le**, and **ge**

The following query selects all **Education** nodes where the **GPA** is greater than 3.5 for the candidate whose **JobCandidateID** is 2:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $ED in /RES:Resume/RES:Education
   where xs:decimal($ED/RES:Edu.GPA) gt 3.5
   return
   $ED
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```

3.3.5 The if-then-else Construct

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $A in /RES:Resume/RES:Address
   return
      if ( $A/RES:Addr.Type eq "Home" )
      then
         <Result>Home Address</Result>
      else
         <Result>Other Address</Result>
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3
```

```
WITH
XMLNAMESPACES( 'http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-
10-21T21:12:27' AS "my")
SELECT
xmlinfo.query('
if ( /my:meetingAgenda/my:meeting/my:meetingOrganizer[.="AXA"] )
then
      <abc>MeetingOrganizer AXA</abc>
else
      /my:meetingAgenda/my:meeting/my:meetingOrganizer
      ') as Result,ProductName
FROM dbo.XQueryTable
```

Results | Messages

	Result
1	<abc>MeetingOrganizer AXA</abc>
2	
3	
4	
5	<abc>MeetingOrganizer AXA</abc>
6	
7	
8	
9	<my:meetingOrganizer xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27">Microsoft</my:meetingOrganizer>

3.4 Built-in XQuery Functions

Data Accessors

You can use the data accessor functions to extract values of nodes as strings or typed values. XQuery supports two types of data accessor functions: **string()**, which extracts the string value of an item and **data()**, which gets the typed value. If the node is not a text node, an attribute node, or an element node then the **data()** function throws a static error. If the node is a document node of an untyped XML instance, then **data()** returns a string value of the document. The **data()** function returns a static error if the node is a complex typed element.

Example: Using the data() function

The following query generates the employment history of a candidate by using the **data()** function and computed element constructors:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $ED in /RES:Resume/RES:Education
   where xs:decimal( data($ED/RES:Edu.GPA) ) gt 3.5
   return
     element Education
     {
      element Level { data($ED/RES:Edu.Level) },
      element Degree { data($ED/RES:Edu.Degree) },
      element GPA { data($ED/RES:Edu.GPA) },
      element GPAScale { data($ED/RES:Edu.GPAScale) }
     }
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```

String Manipulation

XQuery supports four string manipulation functions:

- **concat()** Helps concatenate two or more strings.
- **contains()** Helps determine whether or not a string specified as the first operand contains another string specified as the second operand. The length of the search string is limited to 4,000 Unicode characters.
- **substring()** Helps extract portion of a string from another string known as source string.
- **string-length()** Helps calculate the length of a string.

Aggregate Functions

Aggregate functions operate on a sequence of items and return the aggregate values of the sequence. Aggregate functions currently supported in the XQuery support in SQL Server 2005 are **count()**, **min()**, **max()**, **avg()**, and **sum()**. The functions **min()** and **max()** accept only base types that support the **gt** operator (i.e., the three built-in numeric base types, the date/time base types, xs:string, xs:boolean, and xdt:untypedAtomic). A sequence of mixed types is not supported in these functions. Furthermore, xdt:untypedAtomic is treated as xs:double.

For **avg()** and **sum()**, the type of the passed expression needs to be a subtype of one of the three built-in numeric base types or untypedAtomic (but not a mixture, xdt:untypedAtomic is treated as xs:double).

The **count()** function returns the number of items in a sequence.

Example: Using the avg() function

The following query calculates weekly average high and low temperatures for the cities of New York and Boston using the **avg()** function:

```
DECLARE @Weather xml
SET @Weather = '
<WeatherInfo>
   <NewYork>
      <Temp Date="2004-11-01" High="55" Low="45" />
      <Temp Date="2004-11-02" High="58" Low="42" />
      <Temp Date="2004-11-03" High="60" Low="40" />
      <Temp Date="2004-11-04" High="51" Low="47" />
      <Temp Date="2004-11-05" High="54" Low="41" />
      <Temp Date="2004-11-06" High="55" Low="43" />
      <Temp Date="2004-11-07" High="58" Low="47" />
   </NewYork>
   <Boston>
      <Temp Date="2004-11-01" High="53" Low="45" />
      <Temp Date="2004-11-02" High="56" Low="42" />
      <Temp Date="2004-11-03" High="54" Low="41" />
      <Temp Date="2004-11-04" High="52" Low="45" />
      <Temp Date="2004-11-05" High="52" Low="36" />
      <Temp Date="2004-11-06" High="54" Low="41" />
      <Temp Date="2004-11-07" High="56" Low="44" />
   </Boston>
</WeatherInfo>'

SELECT @Weather.query('
   <WeatherInfo>
      <NewYork>
         <AvgHigh>{ avg(/WeatherInfo/NewYork/Temp/@High) }</AvgHigh>
         <AvgLow>{ avg(/WeatherInfo/NewYork/Temp/@Low) }</AvgLow>
      </NewYork>
      <Boston>
         <AvgHigh>{ avg(/WeatherInfo/Boston/Temp/@High) }</AvgHigh>
         <AvgLow>{ avg(/WeatherInfo/Boston/Temp/@Low) }</AvgLow>
```

```
      </Boston>
   </WeatherInfo>
') as Result
```

Context Functions

You can use the context functions to obtain the contextual properties of a context item. SQL Server 2005 implements two context functions—**last()** and **position()**. The **last()** function can be used to determine the number of items in a sequence and the **position()** function can be used to obtain the position of a context item. Both the **last()** and **position()** functions without an argument can only be used in the context of a context-dependent predicate (i.e., inside []) in SQL Server 2005.

3.5 Type-Related Expressions

XQuery supports various types of expressions or operators that are based on the type information. These expressions can be classified as type assertion expressions, type inspection expressions, and type casting expressions. These expressions are discussed briefly in the following sections.

3.5.1 Type Assertion Expressions

You can use the **as** clause to specify the type for the binding variable used in the **for** statement.

When a type is declared for the binding variable, binding values that are not of the declared type would result in type error. The xs:TYPE clause is not a cast expression but it serves as a type assertion.

Example: Using "as xs:TYPE" clause with for statement

The following query binds the address node sequence to a variable $A which is defined as type element(RES:Address):

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $A as element(RES:Address) in /RES:Resume/RES:Address
   return
      $A
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```

3.5.2 Type Casting Expressions

Implicit Type Casting

The XQuery engine performs implicit type casting for numeric types and untypedAtomic values in expressions that contain arithmetic operations or function invocations. This process is known as type promotion. Type promotion occurs when an expression results in a numeric type that is incompatible with the expected numeric type. Type promotion is performed by casting the resulting expression to the required type.

Example: Implicit type casting

The following query performs an arithmetic operation on a decimal value and a double value. In the current scenario, the values in the expression are added only after promoting the xs:decimal value to xs:double.

```
DECLARE @Result xml
SET @Result = '<Result />'
SELECT @Result.query('
   <Result>{ xs:decimal("10.55") + xs:double(1.5e1) }</Result>
') as Result
```

Explicit Type Casting

3.5.3 Typed value constructors

XQuery provides constructor functions for all built-in types defined in the XML Schema specification. These constructors are useful for constructing typed values and also for casting values from one type to another. XQuery also makes constructors available for types that are defined in imported schemas.

Example: Using a value constructor for constructing values

The following query returns all **Employment** nodes for which the **StartDate** is greater than a value constructed using constructor for type xs:date:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $EMP in /RES:Resume/RES:Employment
   where $EMP/RES:Emp.StartDate gt xs:date("1995-01-01")
   return
      $EMP
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 3
```

Example: Using a value constructor for typecasting

The following query selects all **Education** nodes where the **GPA** is greater than or equal to 3.8 for the candidate whose **JobCandidateID** is 2. This query uses the value constructor for xs:decimal for typecasting the value of Edu.GPA from xs:string to xs:decimal:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $ED in /RES:Resume/RES:Education
   where xs:decimal( data($ED/RES:Edu.GPA) ) ge 3.8
   return
     element Education
     {
      element Level { string($ED/RES:Edu.Level)},
      element StartDate { string($ED/RES:Edu.StartDate)},
      element EndDate { string($ED/RES:Edu.EndDate)},
      element Degree { string($ED/RES:Edu.Degree)},
      element GPA { string($ED/RES:Edu.GPA)},
      element GPAScale { string($ED/RES:Edu.GPAScale)}
     }
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```

3.5.4 cast as xs:TYPE ? Operator

XQuery in SQL Server 2005 supports the **cast as TYPE ?** operator, which is useful for performing explicit type casting. Explicit type casting can also be performed using the **xs:TYPE()** constructors, which are more convenient to write than the **cast as TYPE ?** operator.

Example: Using "cast as xs:TYPE ?" operator

The following query generates an XML that contains typed values of selected elements from **Education** node set for a candidate whose **JobCandidateID** is 3:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $ED in /RES:Resume/RES:Education
   return
     element Education
     {
      element Level { $ED/RES:Edu.Level cast as xs:string? },
      element StartDate { $ED/RES:Edu.StartDate cast as xs:date? },
      element EndDate { $ED/RES:Edu.EndDate cast as xs:date? },
      element Degree { $ED/RES:Edu.Degree cast as xs:string? },
      element GPA { $ED/RES:Edu.GPA cast as xs:decimal? },
      element GPAScale { $ED/RES:Edu.GPAScale cast as xs:decimal? }
     }
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```

Example: Using "xs:TYPE()" operator

The following query generates the same results as the query in the previous example using the **xs:TYPE()** operator instead of the **cast as xs:TYPE ?** operator:

```
SELECT Resume.query('
   declare namespace
RES="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
   for $ED in /RES:Resume/RES:Education
   return
     element Education
     {
      element Level { xs:string($ED/RES:Edu.Level) },
      element StartDate { xs:date($ED/RES:Edu.StartDate) },
      element EndDate { xs:date($ED/RES:Edu.EndDate) },
      element Degree { xs:string($ED/RES:Edu.Degree) },
      element GPA { xs:decimal($ED/RES:Edu.GPA) },
      element GPAScale { xs:decimal($ED/RES:Edu.GPAScale) }
     }
') as Result
FROM [HumanResources].[JobCandidate]
WHERE JobCandidateID = 2
```