基于元数据的自定义数据结构设计与应用

王晓雯 [1]
WANG Xiaowen

摘 要 针对数据操作中对结构数据的频繁提取或修改而引起的加载缓慢问题，提出一种元数据和自定义数据结构相结合的设计方案。通过对传统的树形结构控件进行改进，封装构建成自定义数据模型，设计符合一定规则的数据结构，使主程序的运行只绑定少量参数，即可完成对层次结构数据的增加、删除、修改、查询。这种数据管理机制实现了对层次结构数据的快速提取、操作和回传，提高了数据的操作效率。通过实际项目实例的引入，验证了方法的有效性，它能够有效地提取数据并加工处理，最大程度降低了数据的调整对整体结构的影响。

关键词 元数据；自定义数据结构；数据管理机制；快速提取；操作效率

doi：10.3969/j.issn.1672-9528.2024.04.029

0 引言

随着当今应用软件系统功能的不断深化，对数据库中数据的访问及操作需求也越来越重要。针对现有的管理系统，数据库中存储的结构数据的存取和更新已作为衡量系统响应速度的重要指标之一。传统的树形结构控件（TreeView）可以很好地体现数据的树形结构，但该控件显示和操作树形数据需要大量代码支持，且每次输出树形数据都要编写重复代码，进行局部数据修改时会频繁刷新页面。这使得很多编码者在使用该控件时，只注重功能实现，不考虑系统开发的整体效率，工作量巨大，代码冗余，一旦完成很难再进行修改。特别是当需求变更时，往往牵一发而动全身，导致整个页面的后台代码修改困难，难以更新和维护。针对上述问题，为实现树形结构数据的快速提取、操作和回传，缩减代码量，从基于元数据的技术角度进行分析，结合实际的应用需求，研究并构建了一种针对层次结构数据的结构模型，对原有的树形数据结构进行改进，引入元数据机制，建模封装成自定义数据结构，实现对数据库中结构数据的增加、删除、修改和查询，以达到数据的有效管理。

1 元数据简介

元数据是描述数据的数据，它通过描述信息资源或数据本身的特征和属性，来规定数字化信息的组织 [1]。在关系数据库中，元数据是对数据库、表、列和其他对象的定义，在信息系统中一般把元数据看成是独立的信息单元 [2]，对元数据进行管理可以灵活地实现外部数据的加入和退出，降低数据管理的难度。元数据是对信息资源结构化的描述，在数据库中可以更好地存储树形结构数据的节点信息 [3]。在元数据应用上，资源分析可以说是明确元数据所对应描述的数字对象以及相关的需求等。在系统建模基础上，用属性提取来分析每一个资源实体的属性以及资源实体的管理、描述、应用的功能需求，从而可以得到每个资源实体的元数据描述。只要为所需要描述和管理的元素在其他的一种或几种通用元数据规范中找到了相应的对应实体，元数据元素便可以取自一个或多个命名域，并且可以自定义相关的编码体系与编码规则和重新定义所引用的概念元素。

2 整体设计

2.1 设计思想

结构分为一元结构、线性结构和塔式结构。一元结构由多个相对独立、互不相关的节点构成，它的结构最简单。线性结构由一系列平行或者具有顺序关系的指标构成，当指标较多时，这种结构很难准确描述指标之间的关系，由此产生了常用的塔式结构。塔式结构也称为递阶结构，其由多个具有层级关系、相互联系的节点构成。层次结构数据就是一种塔式结构，它通常采用树状的关系结构，能较全面地描述指标之间的关系。

层次结构数据即节点层次结构，数据中的每一项称为一个节点，并可以按照展开或折叠的方式显示父节点或包含子节 [4]。传统的层次结构典型代表树形控件（TreeView）作为一个独立的输出数据控件使用，树中的每一项称为一个节点并由 TreeNode 对象表示，可以为用户显示节点层次结构，但无法满足输入数据要求。若要实现输入功能，则需要借助其它控件共同完成（即需要将控件结合其它控件来共同实现此结构数据的输入、输出一体化操作 [5]），这便需要封装成自定义数据控件。自定义数据控件是将用户界面和其他功能都

1. 中国石油天然气第七建设有限公司 山东青岛 266061

封装到可复用的包中，与其他标准控件相比，除一个不同的标记前缀，并且必须进行注册和部署以外，并没有什么不同。此外，自定义控件还拥有自己的对象模型，能够触发事件，并支持 Microsoft Visual Studio 所设计的特性，例如属性窗口、可视化设计器、属性生成器和工具箱 [6]。

为实现此数据结构数据输入、输出一体化操作，利用元数据理念构建开发一套自定义数据结构模型，将 TreeView 树形控件结合 TextBox 输入控件来共同实现树形结构数据的提取、操作和回传，并将其封装成一类新的自定义树形控件。在使用时输入树形结构数据节点的名称和属性，将元数据信息填入数据库，然后配置这些信息就可以根据树形结构数据生成相应的目录树 [7]。重复使用时，只需传递此控件的少量参数（包括父节点字段名、节点编号字段名、节点名称字段名、数据表名），即可完成对数据的获取。通过这种元数据思想，可以建立起自定义数据编码，采用基于元数据的设计模式，“忘记”数据库中有哪些具体的表，熟悉元数据表的内容，搞清楚元数据中的记录包括哪些类别，每一类对应于数据库的哪些对象，在设计系统的应用功能模块时，尽量针对一类数据库对象，而不是一个具体的表或字段。

2.2 改进理念

考虑到信息数据的记录方式和表现方式经常改变，采用元数据机制配置数据库中的树形结构数据，通过元数据描述数据库表及其字段属性，当数据库信息或实体信息发生改变时，只需修改相应的元数据，而无需更改程序。利用 .Net 平台为开发各种应用程序提供的自定义控件技术支持，通过对现有树形 TreeView 控件改进并与 TextBox 控件组合而成新的自定义控件并对其进行封装 [8]。设计形成基于元数据的自定义树形结构数据，根据元数据机制的描述，结合控件开发技术，利用此读取元数据信息显示输出相关数据及属性值。此结构数据包括对树形结构数据的增加、删除、修改和查询。针对不同的元数据表在调用该数据结构时，只传入几个参数便可使用，减少了代码量 [9]。通过在元数据表中执行查询，此时的工作转化为针对数据类型而不是针对数据具体值，提高了数据访问提取效率，同时减少了编程工作，实现了整体的封装集成，有效地解决了数据操作的冗余问题。

3 关键技术

3.1 数据库封装

元数据信息主要包括数据库中数据实体各个字段的描述信息，一般包含编码、名称、类别、标识、字段类型、是否主键、对应外键等字段 [10]。系统采用基于元数据的设计模式，“忘记”数据库中有哪些具体的表，熟悉元数据表的内容，搞清楚元数据中的记录包括哪些类别，每一类对应于数据库的哪些对象 [11]。元数据逻辑结构设计的核心是两个元数据表“YSJ_数据表”和“YSJ_ 数据集字段描述”。前者描述数据库中数据表的主要信息，后者描述数据库的数据集中每个字段的主要信息，二者间是一对多关系 [12]。

使用自定义结构数据表主要包括 5 个字段，即树节点编号、节点名称、级数、父节点和节点属性。树节点编号是主键；节点名称是该节点要显示的文本信息；级数表示该节点位于整个树的深度；父节点是标志该节点所属上级节点的字段编号，其中根节点的父节点编号为 0；节点属性是用于描述与节点相关的备注信息。此数据表即为一个数据源的存储结构，设计结果如图 1 所示。

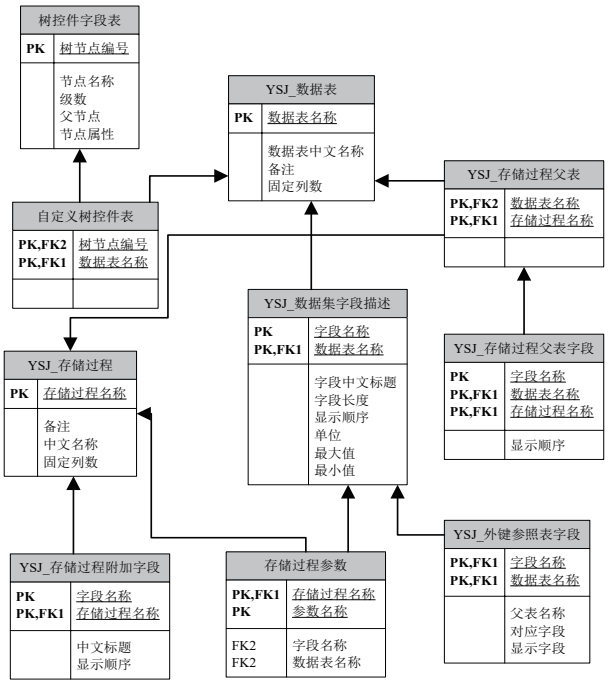

图 1 数据库设计

3.2 类封装

元数据模型下的自定义数据结构设计主要用到 5 个自定义类：数据库连接字符串类、数据集操作类、树型导航操作类、自定义树形结构类和树形结构类，代码中分别为 ConnectManager 类、DataBase 类、MyCommonTool 类、UserTreeView 类和 treelist 类。设计的自定义数据结构模型在继承了传统 TreeView 控件的基础上，添加了一些属性。这些属性的设置是填写元数据表的相关信息，将自定义类封装在结构模型中，不需要再编写树目录形成的代码 [13]。类之间的关系如图 2 所示。由图 2 中关系可以看出，MyCommonTool 类继承 treelist 类的属性并显示树形结构数据，类中的主要方法 FillList()、ListItemIsFather()ListItemIsNull()、ListItemIsRoot()、ListItemIsExsit()ListFillFatherFlag()FillTreeNode()、FillTreeFatherNode() 和 ShowTreeView() 依次实现如下功能：获取元数据，判断该节点是不是父节点，判断该节点有没有记录，判断是不是根节点，判断该节点是否存在，填充父节点下的所有子节点，填充树根节点，展示树型结构内容，在当前节点下新增一个节点，修改、删除当前节点等。

DataBase 类调用 ConnectManage 类用来承载获取数据集并对其进行操作，UserTreeView 类是当单击 TreeView 控件时用来显示自定义结构数据生成的树目录。它调用 MyCommonTool 类并依赖 DataBase 类获取数据集，同时完成对自定义结构数据的增加、删除、更新记录，并存回数据表中 [14]。

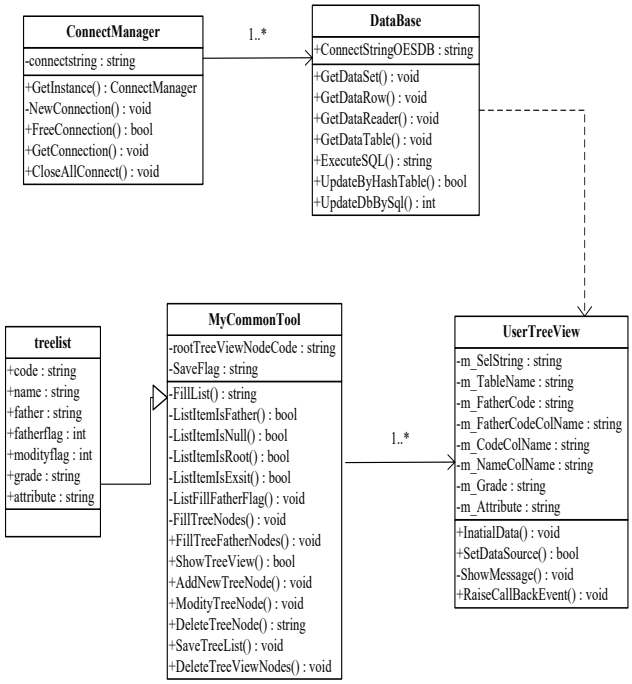

图 2 类设计

3.3 调用机制

自定义数据结构的调用机制流程如图 3 所示。

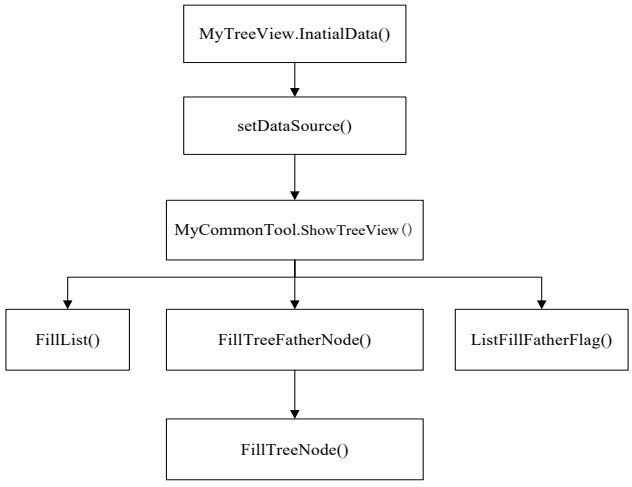

图 3 调用流程图

首先调用自定义树形结构 MyTreeView 获取元数据的函数 InatialData()，传递参数形成 SQL 语句；然后 UserTreeView 类加载函数中调用 setDataSource() 函数设置数据源；setDataSource() 函数调用 MyCommonTool 类的 ShowTreeView() 函数以展示树形结构；最后 ShowTreeView() 函数调用三个函数：FillList() 用数据源填充列表结构，ListFillFatherFlag() 填充链表的父标志，FillTreeFatherNode() 填充树根节点，其中 FillTreeFatherNode() 函数调用 FillTreeNode() 函数，填充该树根节点下的所有子节点。

当对自定义数据结构进行新增修改操作时，点击自定义控件中的新增、保存、修改和删除操作按钮，触发单击事件。即由 RaiseCallbackEvent 函数分别调用 MyCommonTool 类的 GetNextCode()、AddNewTreeNode()、ModifyTreeNode() 和 DeleteTreeNode() 函数来实现输入、更新数据操作。

根据上述设计方案，创建自定义建模数据，在左侧设计树形 TreeView，用来显示数据表中的树形结构字段名称，右侧放置 TextBox 和 Button 组合控件，用来显示单击树节点存储的各个字段的属性值，包括父编号、编号、级数等。具体布局如图 4 所示。

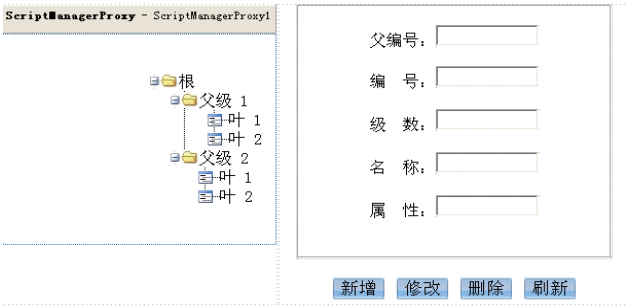

图 4 界面设计

自定义数据结构在增删改查及遍历过程中，需要先对信息格式、通信协议等方面进行约定，元数据模型下的自定义数据结构在互操作中对各个功能信息进行了相关描述，包括每个操作的具体功能，对外提供的接口和输入、输出参数等。为此，本文建立了语言模型，描述服务在操作上的基本信息。

```
protected void Page_Load(object sender, EventArgs e)
    {  // 注册回调函数
        string cbRe = Page.ClientScript.
GetCallbackEventReference(this, "arg", "CallServe", "context");
        string callbackscript;
      callbackscript = "function CallPageServe(arg,context)" +
"{" + cbRe + ";}"; // 调用 CallPageServe() 实现回调
        Page.ClientScript.RegisterClientScriptBlock(this.
GetType(), "CallPageServe", callbackscript, true);
        if (IsPostBack)
        {
          mytool.DeleteTreeViewNodes(myTreeView);
         mytool.FillTreeFatherNodes(myTreeView, m_Fath-
erCode);  // 填充节点
          myTreeView.ExpandDepth = 1;
          return;
        }
          if ((m_SelString == string.Empty)||(m_TableName
```

```
== string.Empty)||(m_FatherCode==string.Empty))
            {
              mytool.ShowErrorMessage(“设置 select 语句或者数据表！”, this.Page);
              return;
            }
            else
            {
              if (!SetDataSource())
              {
                ShowMessage();
              }
            }
          }
```

4 应用案例

该模型设计技术在各类信息管理系统的树形结构存取中得到了应用和验证。图 5 给出在某一管理系统中，应用基于元数据设计的自定义数据结构模型，配置数据连接，传递表名和字段名参数，便可实现层次结构数据的快速映射。

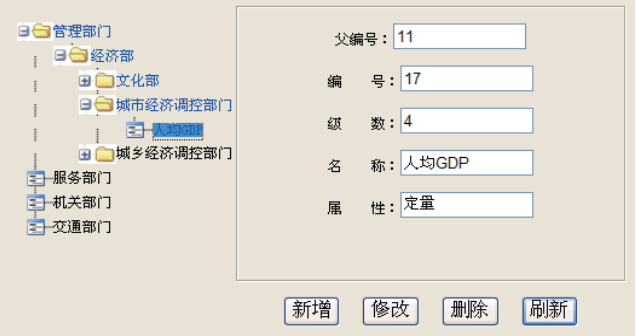

图 5 应用实例

此技术是在自定义树形控件基础上实现的，将相关的数据库操作代码封装到该控件中，只传递少量参数值（父节点字段名、节点编号字段名、节点名称字段名、数据表名）。通过所传递参数读取元数据信息，生成整个部门及其指标列表。在实际的项目开发中，此方法不仅保证了数据的快速读取操作，而且由于大量的数据库操作语句及实现代码被封装到自定义控件中，每个界面中需要编写的代码量减少了，较好地实现了代码复用。

5 结语

元数据模型下的自定义数据结构的设计与应用充分发挥了元数据的作用，在层次结构数据的输入输出操作中得到了成功验证。数据的逻辑独立性也得到了较好保证，在数据库结构发生变更时，只要更新对应的元数据，自定义数据模型就可以读取更新后的信息。这便提高了数据库应用程序的灵活性和扩展性，最大限度地降低了数据库结构的调整对程序的影响。而自定义控件作为一种开发技术，广泛应用于实际系统的层次结构数据的查询、更新等操作中，利用此技术所设计的自定义控件来实现数据操作和检索功能，既高效，又便捷。

参考文献：

[1] 杨阔，李海涛，张雪梅．基于可信云计算的非集中式元数据存储结构优化 [J]. 计算技术与自动化 ,2023(1):183-187.

[2] 金澜涛，黄志球．混合 P2P 海量小文件元数据模型研究与实现 [J]. 计算机与数字工程 ,2013,41(4):598-601+634.

[3] 吴军，李昭，陈鹏．基于元数据的流程模型相似性度量方法 [J]. 计算机技术与发展 ,2019,29(1):49-54.

[4] 谭笠志，李常宝，贾宏．基于元数据的动态数据需求定制中间件设计 [J] 信息技术 ,2021,2(1):242-244.

[5] 蒋炎华．计算资源共享平台中非集中式的元数据管理 [J]. 计算机应用，2011, 31(2):462-465.

[6]MATEO C M, CORRALES J A, MEZOUAR Y. Hierarchical, dense and dynamic 3D reconstruction based on VDB data structure for robotic manipulation tasks[J]. Frontiers in robotics and AI, 2021,2:1-5.

[7] 陈颖颖，陈秉堰．高校图书馆数据库元数据模型及元数据共享平台设计 [J]. 科学技术与工程 ,2020,20(36):41-42.

[8]YAN D L, LI J X, LEI S N, et al. Public sentiment big data query processing and optimization with unified storage of source and meta data[J]. Journal of physics: conference series, 2021,18(1):1-7.

[9]BITTNER M I. Rethinking data and metadata in the age of machine intelligence[J]. Patterns, 2021,2(2):1-2.

[10]HANNU J, BERNHARD T, YASUSHI K, et al. Metadata as support for data provenance[M].Netherlands: IOS Press, 2017: 12-13.

[11]ZHOU L, LI Q, TU W, et al. A heterogeneous key performance indicator metadata model for air quality monitoring in Sustainable Cities[J]. Environmental modelling & software, 2020,13(2):1-3.

[12] 王璐．基于元数据的自定义决策支持系统的分析设计与实现 [D]. 南昌：南昌大学 ,2020.

[13] 马映梅．融合元数据和社会标注的数字教育资源特征模型研究 [D]. 昆明：云南师范大学 ,2020.

[14]CRISTIANO F, PAOLO T A A, ALESSANDRO O, et al. Decentralized geospatial metadata management delegating properties in the web of data[J].Earth science informatics, 2021, 14: 1579-1596.

【作者简介】

王晓雯（1990—），女，山东青岛人，硕士，工程师，研究方向：软件工程、数据库、信息管理等。

（收稿日期：2024-02-21）