电子CAD框架系统的数据模型和DDL*

潘雪增 寿宇澄 赵余平 平玲娣

(浙江大学计算机系,杭州 310027)

摘要 随着电子产品迅速发展,设计涉及的信息量日益增大,数据格式种类繁多,工程和设计数据管理系统已越来越成为电子产品CAD框架的关键构成部分.本文提出以面向对象方法管理框架系统中的工程和设计数据,对其中的关键部分数据模型作了研究,并给出了DDL实现.该系统已应用到电子CAD框架系统中.

关键词 计算机辅助电子设计,面向对象,框架系统,数据模型,DDL.

应用CAD工具的目的是尽可能缩短产品设计周期,增加竞争能力.但是在产品设计过程中,各设计阶段牵涉到种类繁多的信息和大量的数据.近几年的应用研究表明:工程设计师在设计数据管理和过程集成方面花费的时间,随着大型设计工程项目的展开,成倍地增长.因此,能否有效地管理如此大量、复杂的设计数据,将严重地影响整个产品的设计周期.此外,在一个CAD系统中,各CAD工具处理的数据类型不同,要成功地开发工程项目,一个维护集成数据管理的系统正成为运行可靠安全的基础.更重要的,随着CAD技术的应用发展,只强调同一CAD系统中各工具之间的数据共享,已不能满足应用的实际需要.人们期望在新的CAD框架系统中具有更好的开放性和灵活性,能把各种CAD工具紧密地集成到公共的平台上.在下一代的电子产品CAD框架系统中,EDDMS就是为解决这一关键问题而设计的.在CAD框架系统中,EDDMS将作为公共的数据共享基础.另外,在产品的设计过程中,有各种各样的人员需要操作设计数据,例如工程总管负责整个方案;数据管理员负责设计数据的组织,复制后备,意外事件后的恢复;工程技术人员则参于各实际阶段的设计操作.因此,为保证数据库安全可靠,必须为各类人员设计不同的操作功能.面对如此大量和复杂的数据,必须有强有力的管理方法,其中最关键的技术是数据模型.

1 数据模型

电子产品设计是一个复杂的对象.传统的面向记录的数据模型(如关系数据模型)已无法表达该领域大量的信息和设计对象的复杂语义关系.面向对象的方法已在许多CAD系

* 本文1994-08-08收到,1994-10-22定稿

本研究得到国家863支持.作者潘雪增,1944年生,副教授,主要研究领域为ECAD,工程数据库,电子CIMS,电子CAD框架.寿宇澄,1968年生,博士,主要研究领域为工程数据库,人工智能.赵余平,1967年生,硕士,主要研究领域为CAD框架.平玲娣,女,1946年生,副教授,主要研究领域为电子CAD/CAM,电子CIMS.

本文通讯联系人:潘雪增,杭州 310027,浙江大学计算机系

统和数据库系统中用以建立数据模型[1]. 面向对象方法支持数据抽象和继承,便于描述电子产品设计对象不同层次的抽象[2].

本文提出的数据模型的主要特点体现在 DDL 中. DDL 文本在制定时借鉴了 C++的继承性、封装性和重载等特性,及 STEP 标准中 EXPRESS 语言中对对象的描述手段,再根据电子 CAD 的实际应用要求和系统自身的需要将各种内容融合在一起. 在 DDL 中,只承认类(Class)和对象(Object),而不承认类型(Type)和值(Value)具有独立意义. 在处理上,类型只是作为类的某种终结性的描述,而不再象类那样具有深层语义. DDL 主要内容如下:

(1)Explicit Attribute 描述

Explicit Attribute 是所描述的对象的基本部分,每形成一个对象,它的 Explicit Attribute 都必须有确定值,每个 Explicit Attibute 定义了对象的一个侧面的性质. Explicit Attribute 分为可索引和不可索引两类. 隐含值为不可索引,每个可索引的 Explicit Attribute 由用户在定义时显式指定.

(2)Derived Attribute 描述

Derived Attribute 是对象的可通过某种方式产生的属性,因此对它不光要描述其静态结构,还要说明其产生方式. 在 DDL 中,产生方式以 Method 形式说明,在每个对象生成或被修改时,若某 Derived Attibute 不存在值或其值已过时,则调用其产生方式,形成最新值.

(3)Inverse Attribute 描述

Inverse Attribute 主要用于不同对象的关系,在实现上,它有助于表达对象共享;在语义上,它记录了本对象在何处被引用.

(4)Unique 规则描述

Unique 规则可描述一个 Explicit Attribute 或几个 Explicit Attibute 的 unique 约束,即只要此 Attribute 或此组 Attribute 的值为确定,那么整个对象就决定. 也就是说在同一类中,各个对象的这一 Attribute 或这组 Attribute 的值是不重复的. 当描述为 Unique 的 Attribute或 Attributes 在应用时发生重复时,系统将发出警告,并由用户选择取消前一或后一对象. Unique 中描述的 Attribute 隐含为 Indexable.

(5)Where 规则描述

Where 规则为每一对象的属性间提供了一种值的约束. 当对象生成或被修改时,都进行 where 约束的检查,若有违反,系统将发出警告并拒收该对象. Where 检查的返回为 True、False 二值.

(6)继承性描述

Supertype 和 Subtype 用于描述类层次性,是一种"概括"关系. 它使类的属性和方法可从 Supertype 类中继承来,也可被 Subtype 类所继承. 继承允许用户通过改变属性从已有的类导出新类.

为适应工程需要,DDL 实现多重继承,即可以有多个 Supertype,当发生冲突时,以排在前面的 Supertype 为准并保证方法继承的一致性. 实现多重继承的关键在于冲突解决. 由于在多重继承中类的合成失去了唯一途径,因此可能出现二义性. 主要表现在:

(1)属性、方法的重名问题;

(2)间接公共父类的多次被合成问题;

(3)二义的多重继承问题，即一个父类同时是直接父类和间接父类的问题.

这些问题的解决往往是与领域有关的. 根据工程领域的特点，解决方法是：

(1)在父类中分出优先级；

(2)由于同时存在部分继承，所以不强调间接公共父类的被合成时的唯一性，即类在合成时只从直接父类中得到要继承的内容；

(3)在工程中情况(3)的实际意义较弱，可将此父类作为比直接父类优先级都低的父类处理. 若有几个这样的父类，它们之间的优先级次序与它们作为直接父类时相同. 实现时还要建立类中属性和方法的依赖关系，以防方法继承中出现不一致问题. 另外，定义了一个虚类"root"作为所有类组成的树的根.

DDL 中支持 Attribute Redeclaration，即当所继承的属性与本地说明中相重时，以本地说明的优先权为高. Supertype/Subtype 关系是可传递的，不允许存在环.

(7)Method 描述

对每一对象中属性的使用均通过 Method 进行，即外部只能调用所定义的 Method 对对象进行操作，这是为了实现对象的封装性，也是实现继承性的基础之一，它体现了 O—O 方法的基本内容.

(8)类型聚合描述

用于建立新类型，包括 SET、ARRAY、BAG 和 LIST.

(9)自动定义

每当一个类被定义时，相应地自动产生对这个类的一些基本操作，用户可以和调用其它定义的方法一样调用这些操作. 这些操作主要是形成此类和访问此类的方法. 这是为方便用户使用而考虑的. 同时也是封装性的实现机制.

(10)封装与重载

封装性是面向对象系统的重要特征之一. 它往往与模块性、安全性、可重用性等联系在一起. 在我们的系统中，封装性表现在 3 个层次：

第 1 层次就是一般意义上的类封装性. 外部对类的操作用消息发送形式进行；

第 2 层次是模式封装. 所谓模式定义简单地说就是一组类定义. 在模式中，允许或不允许其他模式对其中的某些类进行引用(reuse). 它可以更充分地体现面向对象模型"软插件"特点，对模式进化也提供了支持；

第 3 层次是数据库层次上的封装，即允许某一数据库通过激活另一数据库中某些类来引用(Read Only)那一数据库中的数据. 这不但提高了数据共享性，更重要的是它将成为异构的面向对象数据库间通讯的基础.

重载意为同一方法名可以定义于不同的类上. 重载中一个关键问题是冲突问题. 为解决冲突问题我们规定：方法在关系最近类中定义.

所谓关系最近类的排列顺序是：(1)当前类；(2)排名在前的直接父类.

在这个规定下，不会出现方法名冲突的问题.

总之，在 DDL 中实现了对象的静态描述、动态生成属性和动态约束，可索引性、唯一性、与其它对象的引用关系、操作方法的定义，以及对象标识、封装、继承、重载等特征.

DDL 的 BNF 参见附录.

根据 DDL 产生的内容及流程可由图 1 示出.

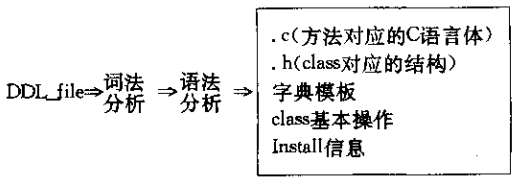

图1 DDL到库结构的转化过程

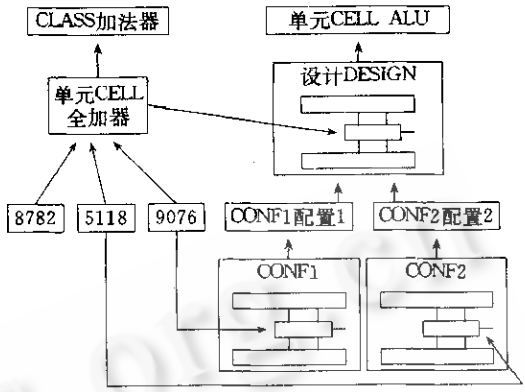

图2 各类对象的引用关系

以下为一个 DDL 定义的例子. 它定义了电子 CAD 中的一个典型模型.

在本文的框架系统中,对象分为 4 种类型:类对象,单元对象,设计对象和配置对象.

(1)类对象:描述一组性质相同的对象的类型. 设计者可以应用类属性访问一组对象,例如加法器表示具有加法功能的器件,但是加法器可以由各种具体的全加器或半加器实现. 因而加法器是类对象,而全加器是元件实例.

(2)单元对象:在一个复合对象中,它可以作为一个元件实例引用. 单元对象是一个逻辑抽象,与具体的技术实施无关. 按照定义的接口约束条件,可以输入到下一级抽象层次设计对象中. 它作为元件来说,设计者既能应用库中的单元对象,也能自己设计一个新的单元对象. 然而不管怎样,它都必须满足预先定义的约束条件.

(3)设计对象:它可以看作为设计项目功能体系结构的抽象描述. 通过设计验证,能动态地检测设计对象的约束条件. 设计对象的约束条件也可以继承到低级配置对象. 设计对象必须满足的一个约束条件是它的父亲单元对象的接口描述.

(4)配置对象:是某一设计对象的实例装配[3]. 它的创建是通过引用单元对象的实例,或其他的配置实例. 从单元库或用户设计的单元中选择不同单元的组合,一个设计对象则能产生不同的配置对象. 在创建配置过程中,每次引用单元对象,对约束条件都进行动态检测. 例如,如果元件的接口条件与配置对象的接口要求不匹配,引用的元件就不能插入到相应的位置上.

图 2 是对象和配置对象之间的引用关系的一个实例.

对应的 DDL 描述如下:

```
class TYPE:root
  {
   explicit{
     int cell_num;
     CELL c[cell_num];
   }
  }
class ADDER:TYPE
  {
  }
```

```
class CELL:root
  {
   explict
     {
      int design_num;
      DESIGN d[design_num];
     };
  }
class FULLADDER:CELL
  {
```

```
    where (Exist_cell(8782)
          & Exist_cell(5118)
          & Exist_cell(9076))
  }

class DESIGN:root
  {
  explicit{
     int config_num;
        CONFIG conf[config_num];
     }
  method
       {
        GetConfig(num);
       }
  }
class config:root
  {
  explict
    {
     int cell_num;
     CELL cell[cell_num];
    }
  }
CONFIG methoddef DESIGN:GetConfig(num)
    {
     if(num<config_num)
              return(conf[num]);
     else
              return NULL;
    }
```

2　结束语

本文提出了面向对象设计的数据模型及实现机制,在各个级别上对设计数据提供了集成支持. EDDMS是新一代框架系统的重要组成部分. 相信随着EDDMS投入实际应用,设计工程师将能节省大量的数据管理时间和系统集成时间. 它与设计过程管理系统协同工作,也有利于整个设计过程的管理.

参考文献

1　Rajiv Gupta, Horowitz E. Object—oriented databases with applications to CASE networks and VLSI CAD. Prentice Hall Inc., 1991.

2　Kemper A *et al*. Object oriented database management system. Prentice Hall Inc., 1994.

3　Steve V. A configuration management system in a data management framework. 28th ACM/IEEE DAC, 1991.

附录

```
This file defines the BNF for DDL (Version 1.1)
⟨DDL_file⟩::={⟨ClassDef⟩|⟨MethodDefinition⟩|⟨FunDef⟩}*
⟨ClassDef⟩::=Class⟨Classname⟩:{⟨Superclassname⟩}*
        Explicit{
          {[Indexable][Optional] ⟨classname⟩⟨attr_name⟩ [[arrayindex ]];}*};
          [Unique{{⟨attr_name⟩,}* ⟨attr_name⟩;}*};]
          [Derived{
             {⟨Classname⟩⟨attr_name⟩[[arrayindex ]] By ⟨MethodHead⟩;}*};]
          [Inverse{
          {⟨Classname⟩⟨attr_name⟩[[arrayindex]] For ⟨attr_name⟩;}*};]
          [Method{
                [Public{
                    {⟨MethodHead⟩;}*
                };]
                [Private {
                    {⟨MethodHead⟩;}*
                };]
          };
          [Where {
                {⟨Condition⟩;}*
          };]
```

```
            [Trigger {
                {If〈Condition〉 Then 〈Condition〉;} *
            };]
        }
  | Set of 〈Classname〉
  | Array [arrayindex] of 〈Classname〉
  | Bag of 〈Classname〉
  | List of 〈Classname〉
〈MethodDefinition〉::=Primitive 〈MethodHead〉|
                    〈MethodHead〉{
                        〈MethodBody〉
                    }
〈FunDef〉::=Function 〈Func_name〉([〈Para_list〉])
                    {
                        {〈c_statement〉;} *
                    }
〈MethodHead〉::=MethodDef 〈Classname〉:: [{〈type〉}] 〈MethodName〉([Paralist])
〈MethodBody〉::=[〈VarDeclaration〉]
                        {〈c_statement〉 | 〈MethodStatement〉;} *
〈MethodStatement〉::=[〈MethodVar〉=]
                    〈MethodVar〉. 〈MethodName〉([〈Para_list〉])
〈MethodVar〉::=@〈attr_name〉
```

说明：

1. unique 的属性必为 indexable；
2. MethodStatement 中可随处包含合法的 c 语句；
3. 〈Condition〉定义同一般逻辑表达式.

DATA MODEL AND DDL WITHIN A FRAMEWORK SYSTEM

Pan Xuezeng Shou Yucheng Zhao Yuping Ping Lingdi

(*Department of Computer Science, Zhejiang University, Hangzhou* 310027)

Abstract As electronic products developing rapidly, the information of the design is increasing and requires a more powerful data management system. The engineering design data management system is a crucial component in an ECAD framework when the design involves large volume of data in diversified format. This paper identifies the requirements for such a system and presents the approaches that support these requirements. This paper presents a data model and its corresponding DDL. This system is currently under development and will be released in the next generation of the electronic products ECAD framework.

Key words Computer aided electronic design, object oriented, framework system, data model, DDL.