一种面向对象的开放式超媒体引擎*

龚育昌 周学海 文栋辉 赵振西

(中国科学技术大学计算机科学与技术系 合肥 230026)

E-mail: ycgong@ustc.edu.cn

摘要 从讨论超媒体引擎的设计原则和设计方法入手,提出了一种基于面向对象数据库管理系统的开放式超媒体引擎.由于引入了元对象建模,该引擎不仅可以表达复杂的超媒体语义,而且可以直接支持开放性链接协议,从而实现了对第三方应用的开放性.

关键词 超媒体引擎,元对象,语义约束,语义剪裁.

中图法分类号 TP311,TP393

超媒体引擎是支持超媒体应用并发的抽象机,它提供访问和操作超媒体的接口以及用来维护超媒体应用语义的复杂操作.已有的超媒体引擎或者只提供一个相对固定的超媒体数据模型,需要由应用的开发者通过增加属性来扩充语义并处理引擎的维护(如HAM,HyperBase等[1]);或者只简单地提供存储功能,其他功能均要求应用的开发者自行实现(如HB_3[2]和Devise[3]等),使得超媒体应用的建模复杂化,而且不支持超媒体系统与第三方应用的集成.

一个优良的开放式超媒体引擎应该满足下述4项基本要求:(1) 可剪裁的数据模型,能够抽象而准确地表达各种应用的超媒体对象;(2) 灵活的语义表达,能够方便地描述超媒体应用复杂而丰富的语义;(3) 充分支持超媒体系统与第三方应用和外部数据管理系统的有效集成;(4) 支持对持久对象的多用户访问、并发控制、恢复和版本控制.

以上述4项要求为设计原则,我们利用功能强大的面向对象数据库管理系统Shore,设计并实现了一个开放式超媒体引擎.该引擎引入了元对象建模技术,有效地实现了上述的(1)~(3)项要求.而利用Shore强大的数据库管理功能,自然能较好地满足上述第(4)项的要求.

本文从说明SohEngine(Shore-based open hypermedia enginc)的设计方法入手,详细地介绍了元对象建模中的语义关系和SohEngine的数据模型,文章最后给出了SohEngine的实现和应用情况.

1 SohEngine的设计方法

SohEngine的基本设计方法是在Shore之上建立一个独立于应用的抽象原始超媒体数据模型,使其具有语义表达的灵活性、可任意剪裁性,并充分支持超媒体应用的建模.

原始超媒体数据模型是通过元对象建模构造的,元对象建模的主要工作包括:

(1) 定义独立于应用的语义关系.为了描述原始超媒体概念间的语义关系,根据对超媒体引擎提出的要求,定义以下4种语义关系是充分必要的:

- 平整的超媒体网络结构(加约束规则),如允许使用不同类型的链和结点.
- 元素关联和集合关联,如组合结点和原子结点间的组合关系.

---

* 作者龚育昌,女,1943年生,教授,博士生导师,主要研究领域为数据库和算法设计.周学海,1966年生,博士,副教授,主要研究领域为多媒体技术,面向对象技术.文栋辉,1970年生,硕士,主要研究领域为数据库技术.赵振西,1937年生,教授,博士生导师,主要研究领域为软件工程环境.

本文通讯联系人:龚育昌,安徽 230026,中国科学技术大学计算机科学与技术系

本文1998-12-21收到原稿,1999-04-13收到修改稿

• 类别指定,如建造具有结点行为的链.

• 角色指定,如指定对象的角色及角色转换.

通过定义合适的 Shore 对象类型可以建造这些语义关系.

(2) 创建元对象.为了描述原始超媒体建模概念的结构和行为,SohEngine 将上述语义关系结合起来,并加入约束,定义了 8 种元对象类型,并对每个元对象类内置了一些常用的元对象.这些元对象扩展了数据模型,体现了 SohEngine 强大的表现能力.由于元对象本身就是 Shore 的数据描述语言 SDL(Shore description language)对象类型定义的实例,因此,可以实时、方便地创建新的元对象或通过建模原语来改变其属性.这不仅保证了SohEngine的可剪裁性和语义表达的灵活性,而且这种剪裁不需要编写额外的程序和重新编译.

(3) 提供建模元语.为了构造复杂的超媒体应用,支持对引擎自身的剪裁并实现开放性,超媒体引擎必须提供丰富的建模元语.我们在文献[4]中提出了一个较完备的开放性超媒体链接协议.SohEngine 对该协议中的消息都提供了对应的建模元语作为实现函数.此外,还提供了一组约束元语,用于对引擎进行剪裁.

在开发实际的超媒体应用时,可以将不同元对象的语义组合起来,描述超媒体应用系统的结构和行为.为了给开发者提供灵活的机制以创建所需要的实例对象,SohEngine 利用 SDL 在 Shore 中内置了一组实例对象类.当用户创建表达具体语义关系的应用超媒体对象时,可利用原有的和自定义的元对象,在相应的实例对象类中创建实际的应用实例对象.上述过程中重用了引擎中已有的对象和类定义,而且保证了对象的完整性约束.

SohEngine 的可重用和可任意剪裁的特性极大地简化了超媒体应用的开发工作.

2 元对象建模中的语义关系

定义原超媒体模型中的语义关系是实现 SohEngine 的基础,需要对其进一步地加以描述.

2.1 平整的超媒体网络结构

我们把一个包含单向和双向链的原始超媒体网络定义为图 $G_0=\{N_0,DL_0,BL_0\}$,其中 N_0,DL_0 和 BL_0 分别为顶点集、单向链和双向链.对该结构进行以下两方面的扩展.

(1) 为了定义链结点,对顶点的概念作如下扩展:

设 $N_{i+1}=N_i\cup DL_i\cup BL_i$;$DL_i\subseteq N_iN_i$;$BL_i\subseteq N_i\times N_i$,据此定义图 $G=\{N,DL,BL\}$,其中 $N_i=U_i\in nN_i$;$DL=U_i\in nDL_i$;$BL=U_i\in nBL_i$.为了超媒体结构的平整性,定义如下约束:① 图中不允许有回路;② 定义不允许悬链.

(2) 为了允许不同类型的链和结点,定义分类图 $TG=\{N,DL,BL\}$,并引入 3 个获取结点和链的类型的函数 $Type\ N:N\to NT$;$Type\ DL:DL\to DLT$;$Type\ BL:BL\to BLT$,其中集合 NT,DLT 和 BLT 分别包含了元素 N,DL,BL 的所有类型并且是互不相交的,而 DLT' 和 BLT' 则分别为 DLT 和 BLT 的子集,表示可以当作结点来操作的链的集合.

为了建立链和结点连接的约束,引入约束函数:

$$ConsHT:DLT\cup BLT\to\rho((HT\cup DLT'\cup BLT')\times(HT\cup DLT'\cup BLT')).$$

利用该函数可以指定某种链允许连接某类结点,测试超媒体网络中是否有某种链或者测试某链是不是合法的.例如,为了限定类型为 SUPPORTS 的二元有向链只允许连接类型为 datum 和 claim 的结点或者,在 claim 类结点之间建立连接时,定义

$$ConsHT(SUP\text{-}PORTS)=((datum,claim),(datum,claim)).$$

2.2 元素关联和集合关联(element association and set association)

为了建立原始超媒体模型中的组合机制,我们引入关联(association)的概念,并建立了元素关联和集合关联两类约束机制.

元素关联引入了集合对象来描述一组元素对象的特征.元素关联建立了 Element-of($Element\in S$)关系,因而在元素对象和集合对象间建立了 1-层次结构.

集合关联也引入了一个集合对象(Superset)来描述另一个集合对象(Subset)的属性.集合关联在 Superset

和 Subset 之间建立 Subset-of($S' \subseteq S''$)关系,它可以被递归地应用于建立一个 n-层次结构.依照 Dexter 模型的思路,我们把超媒体的结构和内容划分为两个层次.在结构层,内容结点被认为是原子,它与内容层的内容对象存在 Element-of 关系,或者说它是内容对象的容器.利用集合关联可以建立组合结点.组合结点可以包含原子、链和其他组合结点,因此,可以将其看作超集,并建立相应的子集关系.

为了对元素关联和集合关联进行分类,引入类型函数:

$$Type\ Ass: EO \cup SO \rightarrow ET \cup ST, ET \cap ST = \varnothing,$$

其中 EO 和 SO 分别表示原子和组合对象集合,ET 和 ST 表示可能的类型.将该函数与关联捆绑,对于针对应用来剪裁超媒体模型是十分有用的.

我们对集合关联建立如下约束:

• 子集关系中集合对象的结构必须是无环的,即在同一集合关联中作为子集的集合对象结点只能有一个父结点.原子内容对象的共享是很有用的,故原子关联不受这一约束影响.

• 利用前述的 Type Ass 函数,我们对关联的结构引入约束函数:

$$Cons\ Ass: ST \rightarrow \rho((ET \cup ST) \times N_0 \times (N_0 \cup (\infty)),$$

其中第 1 个参数表示 ST 的子集中可以包含的元素关联类型,后两个参数分别表示该类型在 ST 中出现次数的上下限.

例:设一个类型为 Path 的组合结点包含一个起始结点,并允许有任意多个内容结点,且只能指向一个原子文本内容结点(该例用于对结点加注释).该例的约束函数描述为

$$ConsAss: (Path) = \{(start,1,1),(Content,0,\infty),(FollowBy,0,\infty),(Textcontent,0,1)\},$$

其中 $Start, FollowBy, Content \in ST$,而 $Textcontent \in ET$.这隐含说明:如果产生了一个 Path 结点实例,则超媒体引擎就会自动地生成一个起始结点.

2.3 类别指定(category specialization)

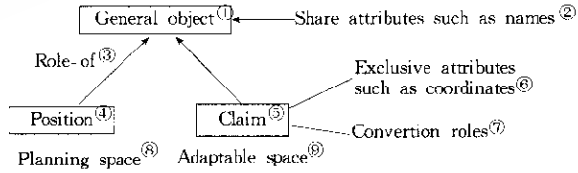
类别指定用来建立对象集合间属性与方法的继承关系,是与类型定义无关的"外部概念".

我们用 $S \prec T$ 表示类 S 是类 T 的类别指定关系,并建立以下一致性约束:(1) 关系 $\prec$ 必须是一个偏序;(2) 只允许类的单重继承,即类关系是树结构;(3) 设 $ext(S)$ 表示类 S 的扩展,则需满足以下条件:如果 $S \prec T$,则 $ext(S) \subseteq ext(T)$,这意味着 $\bigcup_{S \prec T} ext(S) \subseteq ext(T)$;(4) 若类 S 和类 S' 间不存在类别指定关系,则 $ext(s) \cap ext(S') = \varnothing$.

类别指定可用于建造那些需要继承其他类的属性和方法的类,例如,可用来建造有结点行为的链又毋需引入有关结点链的额外概念.

2.4 角色指定(role specialization)

在构造超媒体网络时,有时需要把一个组合结点中某结点的某个(些)属性拷贝到另一组合结点中的另一个类型不同的结点中,使这两个不同类型的结点共享同一个(或一些)属性,实现对象类型的转换.SohEngine 中定义了角色指定关系 Role-of 来表示这种共享.如图 1 所示,建立一个共享属性结点 General,通过 Role-of 关系把名字等共享属性从计划空间中的 Position 结点拷贝到修改空间中的 Claim 结点上.为了说明这种类型转换,引入了一个转换函数 $TransR: R \rightarrow \rho(R)$,其中 R 是类的集合,可以作为一个普通对象角色.此外,我们对类别指定所建立的一致性约束,除约束(4)之外均适用于角色指定.



①普通对象,
②共享属性,如名字等,
③角色,
④位置,
⑤确认,
⑥共享属性,如坐标等,
⑦转换规则,
⑧计划空间,
⑨修改空间.

Fig. 1 Role specialization and convertion
图1 角色指定和转换

3 SohEngine 的数据模型

3.1 SohEngine 中的对象分类

SohEngine 中的对象类型分为元对象类(Meta-Class)和实例对象类(Inst-Class)两类.元对象是第 1 类对象,用以描述实例对象的属性格式和语义约束.一个元对象可以对应于多个实例对象,而相同类型的实例对象只能对应于一个元对象,如图 2 所示.SohEngine 允许应用开发者用 SDL 定义新的元对象,以描述应用所需的某些实际应用对象.图 2 中 T_IsBaseOn 是新创的链元对象,用以描述实际的应用链对象 T_IsBaseOn1 和 T_IsBaseOn2 的属性和语义约束.

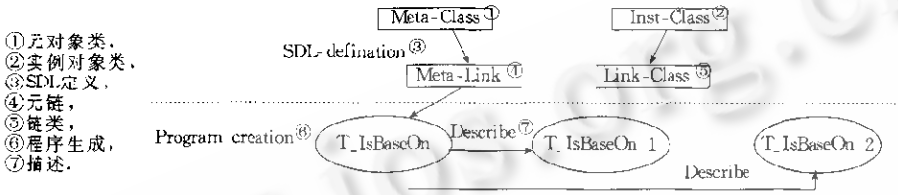
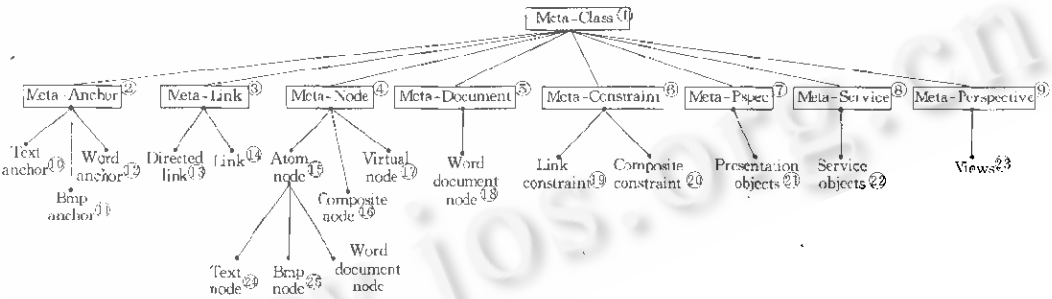


Fig. 2 Relationship between meta-object and instance object
图2 元对象和实例对象的关系

3.2 元对象

SohEngine 综合了所有可想到的超媒体抽象概念,定义了 8 种元对象类型:Meta-Anchor(支持创建各种锚元对象)、Meta-Link(支持创建各种链元对象)、Meta-Node(支持创建各种结点元对象)、Meta-Document(支持创建各种超媒体文档元对象)、Meta-Constraint(支持创建各种语义约束元对象)、Meta-Pspec(支持创建各种展示元对象)、Meta-Service(支持创建各种服务元对象)和 Meta-Perspective(支持创建各种视图元对象).

SohEngine 对这 8 个元对象类内置了一组常用的元对象,如图 3 所示,这些元对象体现了 SohEngine 的表达能力.



①元-对象,②元-锚,③元-链,④元-点,⑤元-媒体文档,⑥元-语义约束,⑦元-展示,⑧元-服务,⑨元-视图,⑩正文锚,⑪位图锚,⑫Word锚,⑬有向链,⑭无向链,⑮原子结点,⑯组合结点,⑰虚拟结点,⑱文档结点,⑲链约束,⑳组合约束,㉑展示对象,㉒服务对象,㉓视图,㉔文本结点,㉕位图结点.

Fig. 3 Meta-Classes and Meta-Objects
图3 元对象类和元对象

SohEngine 还提供了元对象之间的继承机制,允许子元对象继承父元对象的属性和语义约束.例如,图 4 中链 1 和链 2 都继承了有向链特征,又分别捆绑了语义约束$[A,B]$和$[C,D]$,链 3 除了继承有向链特征外,还隐含地继承和捆绑了语义约束$[A,B]+[C,D]$.此外,SohEngine 还提供了灵活的语义约束定义和捆绑机制,以增强引擎的可剪裁性.例如,对图 4 中的示例,可自定义一种语义约束 Constraints,并可动态地向 Link1 进行捆绑或释放捆绑:

```
Meta-Link * pConst1=newMeta-Link(Constraints);
Link1. BindConstraint(Con. S);
Link1. UnBindConstrain:(Con. S).
```

3.3 实例对象类

当超媒体应用的开发者对 SohEngine 进行剪裁或编辑超媒体文档时,SohEngine 会创建各种实例对象.为了支持实例对象的创建,SohEngine提供了与元对象类一一对应的8种实例对象类,分别为 Anchor_Class,Link_Class,Node_Class,Document_Class,Constraint_Class,Pspec_Class,Service_Class 和 Perspective_Class.这些实例对象类可以实例化为多个实例对象.在创建实例对象时,SohEngine 通过元对象理解 Viewer 发来的数据,在 Shore 数据库中建立持久的实例对象,如

Link_Class * IpA1=new Link_Class(T_IsBaseOn).

其中,T_IsBaseOn 是开发者定义的链元对象,它描述了实际的链实例对象 IpA1 的属性格式和语义约束规则.

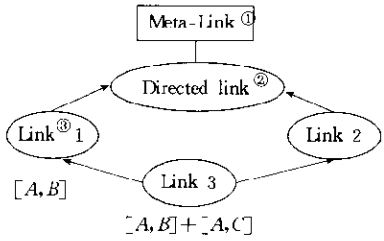


Fig. 4 Inharitance of constraint rules
图4 约束规则的继承

3.4 SohEngine 的建模元语

SohEngine 的建模元语是一组基于 Shore 事务操作的 C++接口函数,客户引用建模元语时利用下列函数与 SohEngine 通信:

- 发送类名,取回一个类的 OID.
- 开始、提交和终止一个 Shore 的上层事务.
- 调用建模元语,取回执行结果.
- 发送 Shore 对象查询语句(OQL),取回查询结果.

当客户调用建模元语进行一系列复杂操作时,每个操作缺省地作为 Shore 的一个上层事务.通过 SohEngine 中的事务命令,开发者可以构造一个新的复杂操作,由 Shore 的事务处理机制来维护其完整性.

我们在文献[4]中提出了一个较完备的开放性超媒体链接协议,该协议是开发超媒体应用(包括第三方应用)所需的通信约定.SohEngine 建模元语的主要部分几乎是一对一地实现该协议中的消息.这部分元语分为7类,共38个元语,分别为

- 锚元语:DefAnchorType,CreateAnchor,GetAnchorDef,UpdateAnchor,DeleteAnchor,GetAnchorList,ExecuteAnchor,GetAnchorPspe;
- 链元语:DefLinkType,CreateLink,GetLinkDef,UpdateLink,DeleteLink;
- 结点元语:DefNodeType,CreateNode,GetNodeDef,UpdateNode,DeleteNode,ClosingNode,CheckNode,GetNodeContent;
- 展示元语:CreatePspec,GetPspecDef,UpdatePspec,DeletePspec;
- 服务元语:GetService,SendService,ExecuteService;
- 视图元语:DefPerspective,GetPerspectiveDef,UpdatePerspective,GetPerspectiveList,SelectPerspective,ClosePerspective;
- 文档元语:RegisterDocument,UpdateDocument,DeleteDocument,GetDocumentContent,CheckDocument.

为了支持对引擎和文档进行剪裁,SohEngine 还提供了4条语义约束元语,分别为 CreateConstraint(创建新的语义约束)、GetConstraint(根据约束的 id 取到其定义)、UpdateConstraint(根据约束的 id 修改其定义)、DeleteConstraint(根据约束的 id 删除其定义).

应用建模元语开发超媒体应用变得相对简单了.例如,只要给出接口函数:

CreateNode(Meta_Node_Class:OID,Composite:OID,name:Cstring,position:CPoint):OID,

就可创建一个元类型为 Meta_Node_Class 的结点,引擎将自动检查该结点是否满足组合约束(是否属于组合结点),执行后引擎将返回一个新结点的 id,或告知由于不能满足语义约束而无法创建该结点.上述创建过程无需编写额外的程序.

4 实 现

依据上文中介绍的超媒体引擎设计技术,我们在SPARC工作站上构造了一个SohEngine的原型系统,实现了SohEngine的大部分功能,并以该引擎原型为核心开发了一个开放式超媒体实验系统,通过《数据结构》课程的计算机辅助教学应用验证了SohEngine在可剪裁性、语义表达灵活性和可简化超媒体应用开发的优点,以及对第三方应用和外部文档管理系统的开放性.

参考文献

1 Cambell B, Goodman J M. HAM: a general purpose hypertext abstract machine. Communications of ACM, 1988,31(7): 856~851

2 Legget J J, Schnase J L. Viewing dexter with open eyes. Communications of ACM, 1994,37(2):76~86

3 Gronbak K, Hem J A, Madsen O L *et al*. Cooperative hypermedia system: a dexter based architecture. Communications of ACM, 1994,37(2):67~74

4 Li Guang-ya. Research and implementation of open hypermedia system [Ph. D. Thesis]. Hefei: University of Science and Technology of China, 1998
(李光亚.开放式超媒体系统的研究与实践[博士学位论文].合肥:中国科学技术大学,1998)

An Object-Oriented Open Hypermedia Engine

GONG Yu-chang ZHOU Xue-hai WEN Dong-hui ZHAO Zhen-xi

(*Department of Computer Science and Technology University of Science and Technology of China Hefei* 230026)

Abstract The design principles and methods of the hypermedia engine are discussed in this paper, and an open hypermedia engine based on OODBMS (object oriented database management systems) is presented. By introducing meta-object modeling, the engine can not only express complex hypermedia semantic, but also support the open hypermedia link protocol for integrating the third-part applications.

Key words Hypermedia engine, meta-object, semantic constraint, semantic tailoring.