面向对象的语义关联数据模型查询语言在C语言中的嵌入

周立柱 王小京 衣丰超 王健斐

(清华大学计算机科学与技术系, 北京, 100084)

EMBEDDING OBJECT-ORIENTED AND SEMANTIC ASSOCIATED DATA MODEL QUERIES IN C LANGUAGE

Zhou Lizhu, Wang Xiaojing, Yi Fongchao and Wang Jianfei

(Department of Computer Science and Technology, Tsinghua University, Beijing, 100084)

ABSTRACT

This paper discusses the issues on embedding an object oriented and semantic associated query language–OSDL in C language, and presents a complete C host language interface for OSDL. This C interface, called OSDL-C, suggests a method for embedding semantic data model queries in a procedure language in general.

摘 要

在这篇论文里, 我们重点讨论在C 语言里嵌入面向对象的查询语言OSDL 所遇到的矛盾及处理方法, 完整地提出了在C 中支持使用OSDL 的宿主语言接口OSDL-C. 文中提出的宿主语言接口设计思想, 对于如何在过程性语言里嵌入面向对象及语义数据模型的查询语言具有一定普遍意义.

1990 年4 月16 日收到, 1990 年7 月16 日定稿.

§1. 引言

近年来，面向对象数据库系统的研究产生了一大批数据模型及有关的查询语言．这些数据模型都具有面向对象的若干特征[1]，而它们的查询语言多为一次一个集合(即每次查询都得到所有满足条件的客体) 的非过程性语言．OSAM* 与它的查询语言OSDL 就属于这样一种数据模型与查询语言[2]．

由于OSDL 主要应用于查询，因此它在功能上并不完整．这一局限性使得程序员无法单独使用OSDL 开发应用系统．为此，我们为程序设计人员提供了在C 中使用OSDL 命令的宿主语言接口．

由于非过程性查询语言与过程性语言之间存在的本质上差别，因此当把OSDL 嵌入到C 这样的语言里时必然要遇到一些矛盾．

首先，C 语言对数据的处理是采用一次一个记录的方式，而OSDL 语言每执行一个查询语句会得到所有符合条件的对象集合，因此两者之间存在着“失配”问题．“失配”引起的突出矛盾需要宿主语言接口妥善地解决数据导航问题．

其次，OSDL 语言支持时间、日期、集合、有序集、向量、矩阵这样一些复杂数据类型．如何在C 语言里支持这些扩充的数据类型，也是C 宿主语言接口应当解决的问题．

为了解决上述矛盾，我们采用了指针及指针层次[3] 的概念，把它用于对查询结果进行导航．同时又在OSDL-C 中引入了“伪类型”，为用户在C 的变量里存贮数据库数据提供了极大的方便．

§2. 面向对象的语义关联数据模型OSAM* 及其查询语言OSDL

OSAM* 是为CIMS(计算机集成制造系统) 设计的高级数据模型．OSAM* 通过对象类(object class) 以及它们之间的语义关联(semantic association) 来描述现实世界．对象类又分作域类与实体类．域类只用于定义一定的取值范围，例如整数、字符串等．而实体类则表示客观存在的事物或概念．每一个实体类都包含若干个实例．每一个实例即为一个对象(object)．

OSAM* 支持下列四种语义关联：A(Aggregation) 关联，G(Generalization) 关联，I (Interaction) 关联，以及C(Composite) 关联．在上述四种语义关联中，A 关联用来表示一个实体类由哪些域类、实体类组成．G 关联表示实体之间的子类与超类的继承性关系．I 关联类似于E-R 模型里实体集之间的relationship 关系；而C 关联则用于说明某一实体类整体被当作另一个实体类里的一个实例来处理．

OSAM* 对现实世界的描述(数据库模式) 可以用语义图来表示．图1 是OSAM* 语义图的一个示例．

图1 中的矩形表示实体类，圆圈表示域类．域类又分简单域、复合域．简单域是指图中最下面的叶子结点域，而复合域则由简单域组合而成．例如position．

OSAM* 的查询语言OSDL 是一种集合型语言．它所提供的查询语句RETRIEVE 具有下述格式：

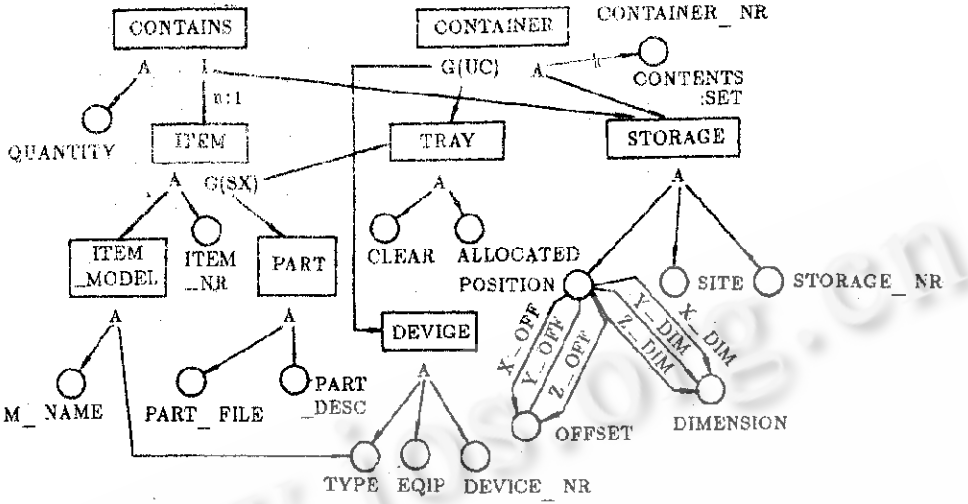

图1 语义图表示的数据库模式

RETRIEVE 域类名
CONTEXT 关联表达式
[VIEWPOINT 实体类名]

Retrieve语句中的“域类名”表明用户要输出的具体对象. “关联表达式”则说明了通过哪条语义关联路径检索以及对象的具体限制条件. 而VIEWPOINT 任选项则决定了是以INF 形式还是以嵌套关系的形式将结果输出. 例如, 图1 中的查询“输出所有运输小车的设备号以及它们要运输的组件所在仓库的仓库号及位置”可以表示成

RETRIEVE DEVICE_NR, STORAGE_NR, POSITION
CONTEXT DEVICE [TYPE='cart']*STORAGE
VIEWPOINT DEVICE

DEVICE_NR	STORAGE_NR	POSITION					
		X_OFF	Y_OFF	Z_OFF	X_DIM	Y_DIM	Z_DIM
D1	S10	100	150	120	0	0	50
	S7	50	100	120	0	100	50
D2	S5	100	150	120	10	100	100
	S22	50	50	100	300	500	0
⋮							

图2 retrieve 查询实例

由于在上面的RETRIEVE 语句中将DEVICE 指定为输出的VIEWPOINT, 因此输出结果将以DEVICE 为立足点组织成嵌套表格的形式, 图2 示意了具体的查询结果.

§3. 确定指针及指针层次

OSDL 的C 预编译程序规定，必须通过下述语句为retrieve 的结果定义一个指针

DEFINE　指针名　for RETRIEVE　域名

CONTEXT　关联表达式

[VIEWPOINT　实体类名]

我们把这样定义的指针称之为retrieve 语句的“根指针”．当retrieve 语句没有指定viewpoint 的实体类时，其输出是1NF 关系形式，这时该指针即为输出的1NF 关系的指针．而当retrieve 语句指定了作为viewpoint 的实体类时，输出表现为嵌套关系．这时，“根指针”仅仅是指向viewpoint 给出的实体类，为了引用查询结果中其它实体类的域类，用户还必须用下列语句为这些实体类定义指针及它们的双亲指针

DEFINE　指针名　FOR　实体类名

WITHIN　双亲指针名

在为非viewpoint 的实体类定义指针时，OSDL 的C 预编译程序的唯一限定是，这样的指针有且只能有一个双亲指针．下列语句为图2 查询结果及它的RETRIEVE 语句确定由C0及C1 组成的指针层次．

```
DEFINE C0 FOR
  RETRIEVE DEVICE_NR, STORAGE_NR, POSITION
  CONTEXT DEVICE [TYPE='cart']*STORAGE
  VIEWPOINT DEVICE
DEFINE C1 FOR STORAGE
  WHTHIN C0
```

指针具有下述语义约束：(1) 任一指针在任一时刻至多指向它所代表的实体类中的某一个实体．(2) 当一个双亲指针已确定地指向某一个实体e 时，它的子女指针只能指向它所代表的实体类中与e 有语义关联的实体．

§4. 通过指针及指针层次的数据导航

OSDL-C 预编译程序提供了下述FETCH 语句

FETCH　指针名　参数1, 参数2, ···, 参数n

INTO　变量1, 变量2, ···, 变量n

这里指针名后面的参数必须是在RETRIEVE 语句中出现的域类名．变量必须是由OSDL-C 提供的DEFINE VAR 语句定义的变量(见下节讨论)．执行FETCH 语句的结果是将参数1, 参数2, ···, 参数n 分别存入变量1, 变量2, ···, 变量n 之中，并且使得指针指向与它的双亲指针实体相关联的下一个实体．下面我们以为上节查询结果定义的指针层次为例说明FETCH 语句的上述功能．

(1) open C0

首先使用open 语句执行与C0 有关的RETRIEVE 语句．

(2) FETCH C0 DEVICE_NR INTO deviceNumber

将C0 指向DEVICE 实体类的第一个实体D1, 同时将D1 存入deviceNumber．

(3) FETCH C1 STORAGE_NR POSITION

INTO StorageNumber PositionVar

将与D1 有关的第一个STORAGE 实体S10 的STRORAGE_NR 及POSITION 数据分别存入变量StorageNumber 及PositionVar.

(4) FETCH C0 DEVICE_NR INTO deviceNumber

将查询结果中DEVICE 的第二个实体D2 存入deviceNumber.

(5) FETCH C1 STORAGE_NR POSITION

INTO StorageNumber PositionVar

将与D2 有关联的第一个STORAGE 实体S5 的STORAGE_NR 及POSITION 数据分别存入变量StorageNumber 及PositionVar.

§5. 有关变量类型的说明

上节介绍的FETCH 语句允许用户将一个指针所指向的实体的域存入变量. OSDL-C 为这些变量提供了以下形式的类型说明.

```
DEFINE VAR SECTION
    变量说明1;
    变量说明2;
      ...
    变量说明n
END VAR SECTION
```

而每一个变量说明的格式为变量类型：变量1, 变量2, ···, 变量m, 这里的变量类型可以是

(1) C 的基本数据类型int, float, short, double, char 等；

(2) 扩充基本数据类型time(时间), date(日期);

(3) 由基本数据类型或扩充基本数据类型组成的vector(向量)、matrix(矩阵)、set(集合)、oset(有序集).

(4) OSAM* 数据库中的域类名.

上面的各种数据类型，除了C 的基本类型外，其余的被称之为OSDL_C 伪类型，它们都将由OSDL-C 预编译程序在进行预编译时转换成C 的结构、数组等内部形式. 而在C 的程序里，OSDL-C 允许用户象使用C 的结构那样，使用这些伪类型.

参考文献

[1] O.Nierstrasz, "A Survey of Object-Oriented Concepts" in Object-Oriented Concepts, Databases, and Applications, ed. by W.Kim and F.Lochovsky.

[2] S.Su, V.Krishnamurthy, and H.Lam, "An Object-Oriented Symatic Association Model (OSAM*)", in: A.I. in Industrial Engineering and Manufacturing: Theoretical Issues and Applications, S.Kumara, A.L. Soyster, and R.L.Kashyap(eds.), American Institute of Industrial Engineering, 1988.

[3] R.Erbe, N.Sudkamp, "An Application Program Interface for a Complex Object Database" Procedings of 3rd. Int. Conf. on Data and Knowledgebases, June, 1988, Jerusalem Israel.