MIDS P++语言*

车敦仁　麦中凡

（北京航空航天大学计算机科学与工程系，北京 100083）

摘要　P++是智能数据库系统MIDS提供给用户的一体化的语言，它在C++基础上扩充了8种重要的语言机制，即对象模式定义、持久性、迭代、查询、版本、约束、触发器和知识表示等.

关键词　语言，持久性，查询，迭代，版本.

MIDS[1]是一个正在开发中的多媒体智能数据库系统，P++是它面向DBA、应用程序员和最终用户提供的一个一体化的通用语言. P++与MIDS中的模型与模式不无关系，但就语言本身而论，P++又是一个独立的语言. 系统通过一些预定义的媒体对象类来提供多媒体支持，多媒体特色没有直接反映到P++语言中.

P++包含两个可分离的子语言：ODL和OQL，分别用于对象模式定义和对象查询. “可分离”意味着DBA和最终用户可把ODL和OQL语句分别单独抽取出来当命令来使用. 这就需要为ODL和OQL实现独立的解释器.

MIDS的内核是一个ROODBMS(Reduced OODBMS). 在MIDS中，“类既是对象工厂，又是对象仓库”，任何对象都是由其类的构造函数(Constructor)创建的，并被自动收藏在类的外延(Extent)之中.

每个类对应一个概念(Concept)，类的定义本身是概念的内涵，类的逻辑上所包含的所有实例的集合是概念的外延(Extention)；在MIDS中，我们把类的外延(Extent)定义为该类当前实际拥有的实例的集合，这与通常意义下的外延(Extention)是不完全一样的(Extent≠Extention).

由于子类/超类间的继承语义是一种外延包含关系(Inclution－Semantics)，对于类的外延需区分下列两种情况：

小外延：指一个类的全体直接实例所构成的对象集合(不含它的任何子类的实例).

大外延：指一个类及其所有的子类、子子类…，所包含的全体实例的集合.

在不至引起误解的情况下，我们就用类的名字(比如C)指称其小外延，而把它的大外延记为：“all C”.

下面侧重于论述P++所扩充8种语言机制中的前5种，即对象模式定义、持久性、迭

* 本文1993－09－10收到，1994－05－31定稿

作者**车敦仁**，1964年生，讲师，现为清华大学计算机系博士后，主要研究领域为人工智能和新一代数据库系统等. **麦中凡**，1935年生，教授，主要研究领域为软件工程，程序设计语言，数据库.

本文通讯联系人：车敦仁，北京100084，清华大学计算机系

代、查询和版本.关于约束、触发器和知识表示的详细论述,受篇幅所限将另文给出(也可参见文献[2]).

1　对象(模式)定义

ODL 对于 MIDS 就如同 DDL 对于传统数据库系统一样,用于建立数据库模式.P++的 ODL 子语言是应 MIDS 系统的对象模型和库模式的需要,在C++的类型系统的基础上扩充后形成的.

ODL 包括四类基本语句,本节仅就最重要的对象模式定义加以阐述.

对象模式定义的核心就是在 MIDS 所支持类层次结构的 DBClass 分枝下派生所需的对象类,ODL 派生子类的语句格式与C++的类声明语句很相似,但有所扩充.下面是采用类C++语法给出的 univObject(即 MIDS 的对象模型)模式的非严格 BNF 表示:

```
class ClassName [WithVers]; [ParentList]
{
public:|protected:|private:          // Attrs defining section
   [static] CT_Class AttrName;    // C++ like attribute definition
   [static] CT_Class AttrName     // univObject specific attribute
            {
             [:AttrCons BoolExpr;]
             [:Default DeftValue;]
             [:IFNeeded ECAs;]
             [:IFAdded ECAs;]
             [:Inverse AttrName; ]
             [:Exclusive Boolean; ]
             [:Dependent Boolean; ]
            };
            ……
   IndKeys: [AttributeList];  //Index key list defining section
public:|protected:|private:   // Methods defining section
   [static] Method_Signature;
       ……
   [soft|hard] Constraint:
            Cons_Signature { Condition⇒Action };
                // With deferred or immediate execution mode
       ……
   [read|write] Trigger: TrigNameList;
       ……
}
```

其中,选项 WithVers 是对有无版本的声明;选项 AttrCons 是定义在属性上的约束;Constraint 定义状态约束(通常涉及到多个属性间、以及本对象与其它对象间关系的约束).

2　持久性

MIDS 实现其持久性策略的技术关键是在C++的临时对象标识(C++指针)和P++的持久对象标识 UID(Unique IDentifier)间所进行的很好的统一,系统专门为持久对象预定义了与C++中的 new 和 delete 相应的操作,即 pnew 和 pdelete.pnew 创建一个持久对象并返回其标识(UID),pdelete 删除一个持久对象并回收其标识(UID).

此外,由于 UID 是由系统生成的,指称不直观,程序易懂性差,这是用户所不希望的,因

而P++还提供了允许用户按自己的习惯定义持久对象标识的语句：

```
persisting(dbPointer) ClassName * VarName1;
volatile(dbPointer)   VarName1;
persistent(dbPointer) ClassName * VarName3;
```

其中，第1个语句声明一个新的持久对象标识 VarName1；该语句被解释后，ODL 解释器会自动地在 PNT(Persistent Name Table)表中为之建立一个入口表项(亦称为持久性根——Persistence root)，一旦有对象联编到其上，用户即可直接从该持久性根入手，遍历这个对象的内容，而不必先经过查询. PNT 表中不允许同名的表项.

第2个语句是第一个语句的逆操作语句，旨在删除 PNT 表中的一个入口表项.

第3个语句声明一个临时的持久对象标识 VarName3(即指向持久对象的临时指针)；临时的持久对象标识不在 PNT 表中维护，它的作用域服从于普通 C++变量的作用域规则.

持久性名不同于 UID，它是可以被赋值的，即可以改变联编到其上的对象，持久性名和对象间的关联语义是与应用相关的，语义的一致性由程序员来保证.

3 迭代(Iteration)

传统 DBMS 引起"阻抗失配"的根源是：

(1)数据库系统(DDL)支持的数据模型和程序语言的数据模型不一致；

(2)数据库系统(SQL)的计算范型和程序语言的计算范型不一样：前者的特点是面向集合的(a－set－at－a－time)且计算不完全，后者的特点是面向元组(a－record－at－a－time)而且是计算完全的；

(3)持久性缺少正交性：两种语言(SQL 嵌入 PL)混合使用，在同一个应用中，SQL 只能操作持久对象，PL 只能操作临时对象.

传统语言支持的迭代仅限于循环控制语句，如 C 及 C++的 for、while 和 do－while 语句. 支持集合类结构上的迭代是数据库系统(SQL)的特色，旨在消除"阻抗失配"的 DBPL 同时需要这两种迭代机制.

P++直接支持下列两种面向集合的迭代语句：

(1)for_in 语句：

```
for elem in 〈collection_or_extent〉
    [〈suchThat_Expr〉] 〈execution_statement〉;
```

(2)for_in_all 语句：

```
for elem in all 〈extent〉
    [〈suchThat_Expr〉] 〈execution_statement〉;
```

其中：〈collection_or_extent〉::=〈collection〉|〈extent〉

〈collection〉::=〈expression_of_set_or_list_or_array_et_al〉

〈extent〉::=〈class_name〉

〈suchThat_Expr〉::=〈boolean_expression〉

利用上述迭代语句可完成下列5种迭代功能：

(1)无条件迭代:不用〈suchThat_Expr〉选项的迭代;

(2)有条件迭代:有〈suchThat_Expr〉选项的迭代;

(3)非层次迭代:用第1种语句格式(for_in),即仅在〈extent〉所指类的小外延上进行迭代;

(4)层次迭代:用第2种语句格式(for_in_all),意即在〈extent〉所指类的大外延上进行迭代.

可见层次迭代和非层次迭代的区别仅在于迭代基的不同:前者在大外延上迭代,后者在小外延上迭代.

(5)递归迭代:是否递归迭代取决于〈execution_statement〉语句是否对迭代基〈collection_or_extent〉进行动态改变.递归迭代能够动态地改变循环执行的次数.

迭代语句的实现需要迭代基〈collection_or_extent〉支持 TheFirst()和 TheNext()操作;这两个操作预定义在 MIDS 预定义类库的 Collective[3]之中,因为〈collection_or_extent〉的实现依赖于 Collective 类.

4　对象查询

理论和实践都已表明,查询是最重要的数据库特征之一,"无论好坏,SQL 是个星际数据库语言".为克服语义断层和阻抗失配,OQL 被设计成 P++的一个可分离的子集,使 P++实际上成为一个 PPL. OQL 支持两种查询方式:ad hoc 方式和 programming 方式.

Aho 和 Ullman 早在1979年就提出递归查询和最小不动点算子对查询语言(特别是演绎能力)的重要性.鉴于此,OQL 也提供了递归查询的表示.

4.1　OQL 基本语句

OQL 的基本查询语句 Select …From…Where 在语法格式和关键字上仍延用 ANSI SQL 标准,而对各子句的语义和其中所允许的表达式之种类进行了适当的扩充,以应 MIDS 对象模型和 OQL 设计目标的需要.在对于查询与封装性的关系上,OQL 采用了与 O_2Query 相似的原则:唯有 ad hoc 查询能够绕过封装.因此在 OQL 中,我们从形式上废除了 ANSI SQL 标准规定的 insert、update 和 delete 语句.其功能则为相应的 DBClasses 中特定的预定义方法所取代[2].

(1)From 子句.广义而言,ANSI SQL 中的每一个 Selection 是把一组或多组元组变换成另一组元组,一般将前者称为论域(Discourse Domain——DD),后者称为目域(Target Domain——TD).From 子句的作用是引出论域,Select 子句规定目域.From 子句的形式如:From〈DD〉

举例:　From City　//DD 即 City 类之外延;
　　From Beijing.hotels

(2)Select 子句.Select 子句的语法格式如:Select〈TD〉

例如:　Select name,map,garrison
　　Select *
　　Select Name,Thefirst(Beijing.Hotels)

DD 与 TD 有一本质差别:DD 所描述的 Aggregate 都是库中确实存在的,TD 则未必.

(3)Where 子句. 语法格式如:Where 〈WhereExpr〉

例如:Select Name

From City

Where Population>1000000

ANSI SQL 允许在 Where 子句中嵌套查询,OQL 当然也允许.

很显然,欲用 OQL 语句完成 insert,update 和 delete 功能,相应的 DBClasses 必须提供必要的预定义方法给予配合. 这样的好处是,可有效地防止 OQL 语句对对象库的随意操作:insert、update 和 delete 均不是 ad hoc 查询,应遵循对象的封装性.

4.2 OQL 的查询功能

P++也支持很强的查询功能,用户除了可利用 P++迭代功能(尤其是条件迭代)进行间接查询外,还可直接使用 P++支持的具有"ad hoc"特色的 OQL 子语言进行即席查询.

(1)非层次查询

非层次查询是指仅对某一个类进行的查询,是基于该类的小外延上的查询. 例如:

Select Name,Score From s in Student Where s.Score>90;

表示从 Student 类中选择成绩大于90分的学生 ,并且仅对 Name,Score 两个属性进行投影.

(2)层次查询(Hierarchy Query)

层次查询是指对于以某个类为根的整个子类层次进行的查询,是基于该类的大外延上的查询. 例如:

Select all from S in all Student

表示从以 Student 类为根的整个类层次中进行查询,其中第一个"all"表示欲对每个 Student 对象的全部属性进行投影,第二个"all"指明所进行的查询是层次查询.

(3)递归查询

递归在查询与迭代中的思想是相似的. 例如:

```
q=Select all from S in all teacher;
p=q.evaluate();  //查询是对象,evaluate()是计算查询结果的方法
for d in p p.append(d.children);
```

P++的递归查询语义是某种关系的传递闭包.

查询也是对象,可直接对之发消息 evaluate(). 例如:

设命题"老子英雄儿好汉!"成立,并且"英雄"在语义上与"好汉"等价,下列查询语句即可求出所有的英雄:

```
q=(Select From h in Heros).evaluate();
for d in q q.append(d.sons);
```

上面查询均具有下列共同的语法格式:

Select 〈TD〉 From [〈variable〉 in] 〈DD〉 Where 〈WhereExpr〉

其中:〈TD〉::= all|all but att1,att2…|att1,att2…|ε

〈DD〉::=〈Collection〉|〈Extent〉

〈Extent〉::=〈ClassName〉|all 〈ClassName〉

〈WhereExpr〉::=〈Boolean_Expression〉|ε

P++借鉴了VBase中Object SQL[4]的作法，在实现上把每个查询也当作对象，即类Query的实例.根据查询对象与名字的不同约束情况，P++查询也可从另一角度(dimension)分为三种:(1)无名查询,(2)有名查询,(3)持久查询.

其中持久查询意指查询对象被显式约束到某个持久名字(变量)上，使得该查询对象亦可入库，从而可以历经多次(事务)运行.

5 版　本

存在大量应用，例如CAD,CASE、司法和金融业等，从本质讲都希望数据库提供创建数据对象的不同版本的能力，传统数据库几乎对此无能为力，面向对象范型在这方面为DBMS带来了新的生机.版本可以粗分为两类:库模式的版本和数据对象的版本.模式的版本主要是面向数据库系统的，用户(特别是终端用户)更感兴趣的是数据对象的版本，一个先进的数据库语言应该支持创建并操纵对象版本的能力.

我们在对版本的应用需求进行仔细研究之后，将MIDS和P++对版本的支持建在下列假设之上:

(1)版本是对象所具有的一种性质，不是作为一个独立的对象而存在;因而版本化对象(不管拥有多少个版本)只有一个统一的UID(Unique IDentifier);

(2)与持久性相似，版本应与对象类/型正交，即任何类都可被声明具有或不具有版本要求，并且应在类定义时予以声明，而且不被子类自动继承.考虑到实现的简化，MIDS的先期目标不拟支持单独对象的版本请求(声明)，即MIDS第1版支持的版本正交性是不完全的.

(3)只有持久对象的版本才有意义，MIDS P++不支持临时对象的版本.

(4)版本按自然数(线性化)编号，用作版本标识.

(5)由于版本标识是供多用户、多事务共用的，因此同一对象的各版本标识应唯一，且在其生存期内保持不变，就象对象UID的不变性一样.

(6)对于任何对象，一次事务运行最多只产生它的一个新版本.即不允许一个对象具有两个以上的未提交(新)版本，如果事务管理系统支持一个事务的多次(物理)提交，则在一个事务内就有可能产生同一个对象的多个版本.

(7)MIDS支持版本的树形模型，但按提交的时间戳进行线性化标识;

P++共支持下列6个版本操作:

```
(1)NewVersion(p);   //创建p对象的一个新版本，并使之成为当前版本
(2)Version(p,n);    //指定p对象的第n个版本为当前版本
(3)Delete(p);       //若p是一个版本化对象，删除p及其所有版本
(4)DelCurrent(p);   //删除对象p的当前版本
(5)Pre(p,n);        //使当前版本指针前移n个位置
(6)Suc(p,n);        //使当前版本指针后移n个位置
```

6 约束、触发器和知识表示

在P++中，约束和触发器有本质的区别:

(1)二者引发的动作具有完全不同的执行方式:触发器引发的动作可以不依赖于当前事务是否能够成功结束;

(2)约束定义在类上,触发器定义在对象上.一般而言,相同类的对象具有相同的约束,但可关联到不同的触发器上.

(3)职能分工不同:约束偏于负责"内务"(即完整性和协调性),触发器偏于负责"外务"(监控外部事件).

在 univObject 及 P++中,对象和框架是统一的,框架是最一般的对象,也是最一般的知识表示形式.

除此之外,P++还支持两种规则形式的知识表示:ECA 规则(Production Rules)和谓词规则(Horn Clauses).特别是 ECA 规则,当事件模式为空时,即是最一般的产生式规则.而一个谓词(规则)类正与一个 PROLOG 过程相对应,包含一组同名同秩(Name/Arity)的规则和事实.

MIDS 系统内含的推理机能够利用框架和规则知识,自动地组织多种形式的推理,如前向、后向推理等.

7 库操作辅助语句

P++提供两种库访问途径:利用持久性根直接(导航)访问和通过 OQL 查询访问.完成库操作还需要下列辅助语句:

(1)库定义语句,格式如: DatabaseCreate("DB_File_Name");

(2)簇定义语句,格式如: ClusterCreate("Cluster_File_Name",ClassList);

库定义和簇定义是模式定义的一部分.本语句的目的在于接受用户关于一个簇的描述,并创建一个物理文件与之对应,除此之外,不能再对簇施以任何操作;

(3)库指针声明语句,格式如: Database *db,*db1;

(4)库打开语句,格式如: DB_Identifier=DatabaseOpen("DB_File_Name");

(5)库关闭语句,格式如: DatabaseClose(DB_Identifier);

(6)事务开始语句,格式如: Tr_Identifier=TransactionBegin();

(7)事务提交语句,格式如: TransactionCommit(Tr_Identifier);

(8)*事务流产语句,格式如: TransactionAbort(Tr_Identifier);

(9)事务回退语句,格式如: TransactionRollback(Tr_Identifier);

(10)*Tr_Id=TransactionCreate();

(11)*TransactionSchedule(Tr_Id);

(12)*return Tr_Id;

其中语句(10)、(11)和(12)是触发器和 ECA 规则的执行模型所需要的[2].

8 结束语

与 MIDS 和 P++相近的系统和语言有:Vbase 和 Object SQL[4],ODE 和 O++[5],ObjectStore[6]和 ObjectStore C++等,这些系统皆是单纯的面向对象系统或环境.作为一

个PPL或DBPL,P++比O++和ObjectStore C++在迭代查询和版本支持方面均有较强的功能.不仅如此,P++还在知识表示方面进行了探索.

据我们追踪研究的结果表明,目前国内外试图将OO、AI、DB和多媒体等多项重要的软件技术同时进行集成研究的课题还不多见,可供借鉴的东西甚少,因此MIDS课题的开展是带有一定挑战性的.

MIDS的配套语言,P++集OOPL、PPL、DBPL和AI等多种语言特色于一体,其研究也同MIDS一样,多在艰难中进行.

本文如实反映了MIDS P++研究所取得的初步成果.目前,P++的两个子语言ODL和OQL已初步完成,但仍在进一步完善之中.MIDS的核心子系统是对象管理子系统OMS,我们拟配合OMS的完成,使P++作为一个完整的语言尽快予以实现.

参考文献

1 Che D R, Mai Z F. The overall design of MIDS/BUAA: a multimedia intelligent database system. In: Wu J P, Yang J, Gao W *et al*. eds. Proc. of 3rd International Conference for Young Computer Scientists, Beijing, 1993, Beijing: Tsinghua University Press, 1993. 5. 67—5. 70.

2 车敦仁.面向对象的智能数据库系统:研究、设计与实现[博士论文].北京航空航天大学研究生院,1994.

3 车敦仁,麦中凡.MIDS/BUAA模式管理器与对象管理器的设计.计算机研究与发展,1994,**31**(4):14—19.

4 Harris C, Duhl J. Object SQL. In: Gupta R, Horowitz E eds. Object—Oriented Databases with Applications to CASE, Networks and VLSI CAD, Prentice Hall, 1991. 199—215.

5 Agrawal R, Gehani N. H. ODE: the language and the data model. In: Gupta R, Horowitz E eds. Object—Oriented Databases with Applications to CASE, Networks and VLSI CAD, Prentice Hall, 1991. 365—386.

6 Lambetal C. The objectstore database system. Communication of The ACM, 1991,**34**(**10**):50—63.

MIDS P++ LANGUAGE

Che Dunren　Mai Zhongfan

(*Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, Beijing* 100083)

Abstract　P++ is a uniform language provided by the multimedia intelligent database system MIDS/BUAA for users. P++ extends C++ by eight important language facilities, i. e., object schema definition, persistence, iteration, query, version, constraint, trigger, and knowledge representation.

Key words　Language, persistence, iteration, query, version.