计算机科学 1995Vol. 22 №. 4

对象数据库标准——介绍与分析

张成洪 周傲英 施伯乐
(复旦大学计算机科学系 上海 200433)

摘 要 Now, some object database relevant standards have been developed or are being developed by ANSI, ISO or some other groups. Among them, the most influential standards are SQL3 by ANSI, CORBA and COSS by OMG, and ODMG-93 by ODMG. Because OMG Object Services and SQL3 go somewhat beyond the goals of object database, we choose the ODMG-93 to introduce and analyse the date model, the query language and the language bindings in ODMG standard.

关键词 Database System, Object model, Object database.

一、引 言

随着应用领域的扩大和对象数据库的纷纷涌现，对象数据库的用户和厂商都强烈需要一个对象数据库标准。为此，国际标准化组织(ISO)和美国国家标准局(ANSI)以及一些联合体作了大量和对象数据库有关的标准化工作，其中最有影响的标准包括：由 ANSI 和 ISO 制定的 SQL 系列，主要由 ANSI X3H2 和 ISO DBL 委员会联合开发的数据库语言标准 SQL3；对象管理组(OMG)的 CORBA 和 COSS；对象数据库管理组(ODMG)的 OCMG-93。

SQL3 和当前的 ANSI/ISO 数据库语言标准 SQL-92 兼容，并对 SQL-92 作了很多扩充，其中最重要的扩充是增加了面向对象的类型系统。这个类型系统是基于"抽象数据类型"(ADTs)的，ADT 对应有一组属性和过程的定义，它包含封装、继承和重载等和面向对象概念一致的性质。

OMG 为系统软件厂商提出分布式对象系统标准，有两类：CORBA (Common Object Request Broker Archirecture)和 COSS(Common Object Services Specifications)。CORBA 是所有 OMG 对象使用的核心通讯机制，使得分布式对象能互相操作，并定义了一个对象的界面定义语言(IDL)，IDL 允许设计人员说明一组操作和属性，CORBA 还定义了类型化的对象引用，对象引用和对象访问一致。COSS 定义了支持用 OMG IDL 说明的分布式对象集成的标准服务，而且可以在 CORBA 环境中操作。

ODMG 是由一些面向对象数据库系统厂商组成的，已经为它们的产品开发了一个标准界面 ODMG-93，这个标准包括对象数据库管理系统的一个公共体系结构和定义，还包括公共对象模型，有对象定义语言，公共对象查询语言和标准的程序设计语言联编，由 ODMG 定义的面向对象数据库系统提供与编程语言的直接联编，而不象传统数据库管理系统那样，把 SQL 嵌入主语言，公共对象模型允许数据通过编程语言被共享，这个模型包括了对象标识、封装、方法、类型、多继承、关系、列表、集合、数组等特征。

ODMG 成员正在努力与 SQL3 组合作，许多 ODMG 厂商已经在它们的产品中支持 SQL2，ODMG 很容易把 SQL3 安置在它们的体系结构中，可作为一个 SQL3 binding，与 C++、Smalltalk 及其它语言 binding 互相操作和共享。ODMG 和 OMG 更为密切联系，ODMG 基于 OMG 的对象模型和定义语言的标准。OMG 也接受了 ODMG-93 界面作为 OMG 的对象服务中的持久服务的一部分。和 C++，Smalltalk，SQL3 及其它 binding 一样，通过把 OMG 的基于 IDL 的界面作为 ODMG 框架的另一个语言 binding，OMG 界面也可以很好地安装在 ODMG 体系结构中。因此 ODMG 和 SQL3 组、OMG 在很多方面的工作是一致的，由于 SQL3 组和 OMG 的工作超出了对象数据库的范围，本文选择介绍和分析 ODMG-93。

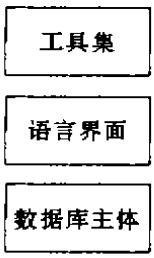
二、ODMG-93

1991 年秋天，由五个面向对象数据库厂商(O2，

Object Design，Objectivity，Ontos 和 Versant）组成了 ODMG（ObjectDatabase Management Group），并出版了一个对象数据库标准的原始版本。1993 年底又加入了 2 个厂商（Poet 和 Servio），于 1994 年初产生了“ODMG-93：对象数据库标准”的最终版本，以后又有 HP、TI、Anderson Consulting、EDS、Sybase 等公司加入。

ODMG-93 是一种可移植性标准，也就是说，它保证在一个系统上运行的应用，可以很容易地移植到另外一个系统上；而不象 CORBA 一类的可互操作性标准，允许一个应用同时在不同的系统上运行。

ODMG 标准基于这样的事实：所有的对象数据库都有一个公共的如下图所示的包括三层的基本系统体系结构：



数据库主体提供所有的数据库功能，支持所有（或部分）对象数据库模型，其界面是一个函数库，是系统编程的内部界面，也可以由应用开发人员直接访问和利用，支持数据存放、索引、安全性、并发控制、恢复、Client/Server 和分布性。

语言包括对象定义语言、对象查询语言和一组编程语言。对象定义语言（ODL）相当于传统数据库中的数据定义语言 DDL，它描述数据库模式。对象数据库模式包括一组类、每个类的结构和方法、方法的基调（Signature）、类层次、复合结构，而没有方法代码和应用代码，所以 ODL 只描述用户或开发者看待模式的方式，而不是实现方式。对象查询语言（OQL）允许访问数据库中的对象，从数据库中抽取信息，以及向对象调用方法。它可以用交互式命令行方式查询，也可以嵌入在编程语言中。编程语言用于编写与对象相联的方法，同时用于编写使用和管理对象的应用。编程语言分成两种形式：内部的和外部的。内部形式指开发和运行环境是 DBMS 的一部分；外部形式指开发和运行环境在数据库系统之外，它需要考虑怎样用外部语言编写数据库中对象的方法，以及怎样在语言中表示数据模型，怎样说明、建立、操纵持久数据。

工具集在各个系统中是不一样的，可以用来开发应用程序和方便地使用数据库。

ODMG-93 标准集中在语言界面，以后的版本将会考虑数据库主体的界面，工具层超出了 ODMG 范围，各个产品正好可以在工具层中反映其特色。由于近年来编程语言界和 OMG 组织花了大量精力来定义和统一了一个普遍接受的“对象模型”，所以 ODMG-93 的对象模型是 OMG 对象模型的简单扩充，其编程语言也不是创造“另一个编程语言”，而认为利用已有的 OOPL 更现实和更有吸引力。ODMG-93 标准重点考虑了对象定义语言、对象查询语言和编程语言。

三、定义对象数据库模式

在 ODMG-93 中，定义数据库模式可以直接用 ODL，它是对 OMG 的界面定义语言 IDL 的扩充，也可以使用 Smalltalk 或 C++来定义。本文介绍的是用 C++格式定义对象数据库模式。ODMG 对象模型和 OMG 对象模型一样，支持类、带有属性和方法的对象、继承和特化，也支持经典的型（type），可处理日期型、时间型、字符串。

3.1 引用

ODMG-93 中使用 Ref 表达一个对象对另一个对象的引用，Ref 相当于 C++的指针，但比指针具有更丰富的语义。C++的指针只是一个内存概念，如果直接通过 C++指针跟踪引用关系，则需要用激活方法把被引用对象从外存调入内存，以后又要用挂起（deactivation）方法把对象放到外存中去；Ref 能把内外存交换工作交给系统自动完成，而且还能用于表达引用完整性，并可以由系统来维护，例如“人”（Person）和“房间”（Apartment）的类定义如下：

```
Class Person{
  Ref<Apartment>lives_in inverse is_used_by;
};
Class Apartment{
  Ref<Person>is_used_by inverse lives_in;
};
```

其中的 inverse 是 ODMG 对标准 C++的类定义的扩充。表示了“人”和“房间”之间的对应关系，从而允许系统维护其引用完整性：当人搬家时，房间的 is_used_by 属性变成 Null；当房间拆了，人的 lives_in 属性变成 Null。

3.2 集合（collection）

有效管理大量数据集合是数据库的一个基本特征，数据库的查询语言一般也是从集合中抽取信息，因此 ODMG-93预定义了一些集合：Set<T>、Bag<T>（允许重复）、Varray<T>（变长数组）、List<T>（变长

且可插入的数组)。

这些集合可以作为同一个类的元素的容器,由于类层次,一个集合中可以有不同类的元素,例如假设雇员是人的子类,则类型为 Set(Ref(Person))的一个集合可以包括所有人和雇员的对象。在 ODMG-93 C++ binding 中,一个类的外延不是自动维护的,需要应用程序显式地创建和维护,对类的外延的维护可以封装在类的方法中(比如创建对象时),而且应用程序可以有任意多的集合,以存放有某种性质的一组对象。

3.3 表达各种关系

如果使用 Ref 和 inverse,能表达类和类之间的一对一关系,现在如果加上集合,就能表达类和类之间的多对多关系,例如每个人有双亲,每个人可以有多个孩子,可表达如下:

```
Class Person{
Set<Ref<Person>>parents inverse children;
List<Ref<Person>>children inverse parents;
}
```

假设 Person 类和很多类都有关系,如和 Apartment 类有居住关系,和其它人有父子关系,和 Company 类有工作关系等,如果在关系数据库中,对应每个类有一个表,对应类之间的每个关系也要一个表;而在对象数据库中,一个类就把它本身的信息和与其它类的关系都同时在其类定义中表达出来。避免了关系库中的 join 操作;但与其它类的关系同时也要在相应的那个类中表达出来,又有信息冗余的问题,所以 ODMG-93中引进 inverse,使系统维护这个冗余信息。

3.4 命名

ODMG-93能显式地为任何对象或集合命名,通过名字可以直接访问对象,然后对对象操作,或者往下找其它对象。在程序中一般只用变量指代对象,但变量只是一个内存概念,没有持久性,名字才是在数据库中指代对象的方法,从这些入口通过查询或导航可以访问数据库中的对象,因此在 ODMG-93中,一般每个类的外延要命名,对类查询就是对类的外延查询。

命名的另一方面作用是涉及到持久模型,即决定什么东西该持久化,在原来的对象数据库系统中就有两种做法:一种以 Object store 和 O2为代表,通过命名实现持久,命名对象、命名集合可达的部分是持久的,另一种是以 Ontos 和 Versant 为代表,系统定义一个持久类(Ontos 中用 Object 类,Versant 中用 Persistent 类),然后这个持久类的所有子类的实例都是持久的。这种做法由类定义决定其对象的持久化,使得持久性和类型相互依赖,违反正交性,所以 ODMG-93选择第一种做法。

四、C++ binding

要实现上面定义的模式,需要为每个方法编写方法体,这些方法体可以用 C++很容易地写出来,其中 Ref<T>就相当于指针(T *),不过是操纵持久对象的指针,它可以和一般指针的用法完全一样。

利用 C++ binding 还可以编写应用程序,ODMG-93 提供了 Database 和 Transaction 类。Database 类处理数据库操作,如打开数据库、关闭数据库方法;Transaction 类处理事务操作,如事务开始、事务提交和事务夭折的方法。应用创建的对象可以是挥发性对象,即对象在程序运行结束之后就消失,也可以是持久对象,在程序运行结束之后仍然存在,并且可以被多个程序所共享。下面的例子程序创建了一个持久的房间对象,并让"john"住进去。

```
Transaction move;//move 是一个事务对象
move.begin( );
   Ref <Apartment> home = new (database) Apartment; // 在 database 中创建一个房间,让
                home 指向它
   Ref <Person> john = database -> lookup_object
        ("john");//检索一个命名对象
   Apartments.insert_element(home);
      //把这个新房间放在类的外延集合中
   john->lives_in=home;
      //持久对象和标准 C++对象一样处理
move.commit( );
```

五、对象查询语言 OQL

ODMG-93还定义了查询语言 OQL,和 SQL 的风格一样,可以利用它方便地访问对象。如果只有前面介绍的对象定义语言和 C++ binding,还不足以表达数据库应用的需求。

5.1 OQL 的特点

①交互式即席查询。OQL 是纯说明性的,不需要用户去书写、编译、连接及调试 C++程序,其语法简单灵活,很容易学习和使用。

②可以通过嵌入查询简化编程。如果把 OQL 嵌入在 C++等编程语言中,由于其表达能力强,一句 OQL 足以替代一长段 C++程序;而且因为 OQL 直接支持 ODMG 模型,避免了"阻抗失配"问题。

③OQL 查询可以由系统优化。关系数据库中的查询优化技术基本上都可以扩充到对象数据库中来。

④逻辑/物理的独立性。正因为由系统进行查询

优化，决定查询策略。所以改变物理的数据结构、索引或聚集策略时，不影响查询语句本身。

⑤高层构造。OQL 提供高层操作符，如为对象排序、分组、聚集、或进行统计，而不必为此编写大量的 C++程序。

⑥动态性。C++需要动态联编，OQL 本身就是动态计算的，即查询计值时决定调用什么方法，而不用做很多编译工作。

⑦对象服务器的体系结构。现在的对象数据库系统大都支持 client/server 的体系结构，有了 OQL 之后，才能真正实现对象级的 client/server，即 client 接收用户需求，向 server 发 OQL 查询命令，server 处理查询并向 client 返回结果。

⑧与 SQL 相似。OQL 尽可能地象 SQL，这主要是为了方便 OQL 的学习和使用。

⑨支持先进特征，包括视图、完整性约束、触发器。它们都需要说明性的语言，因此不用 OQL 很难实现。

5.2 OQL 用于数据检索

正如前面提到的，用户可以通过命名对象进入数据库，并且由于 ODMG 模型中允许对象引用和对象嵌套，用户需要从命名对象根据对象引用关系导航到正确的数据，OQL 中可用“.”或“->”跟踪对象引用关系，进入复杂对象内部，例如：

```
p.lives_in.building.address.street
```

从人 p 出发，得到他所住房间所在的楼房的地址中的街道名。

上例跟踪的是一对一关系，如果其中有多对多关系，比如一个人有多个小孩，要得到其小孩的名字，不能写成 p.children.name，因为 p.children 是一个 List，它后面的“.”没意义，OQL 可使用 select 语句，如：

```
select c.name
from c in p.children
```

它得到的结果类型为 Bag(String)。

和 SQL 类似，OQL 中可以使用“where”子句，其中用谓词作判断条件，谓词中可以使用路径表达式，如果把每个类对应于关系数据库中的关系，那么 OQL 中的路径表达式则隐式地意味着要进行多个类之间的 join 操作。OQL 同时允许显式地联接，即在“from”子句中可以使用没有直接关系的集合。

5.3 复杂数据管理

管理复杂数据是 OQL 区别于 SQL 的主要方面。在关系数据库中，SQL 可以理解为关系代数表达式，其结果也是关系。由于关系模型简单，只需在 select 子句中指定要查的属性名，就能把结果关系的模式确定了，但 ODMG 模型比关系模型复杂得多，有时需要把结果组装成复杂值的集合，因此 OQL 中可以使用 struct，set，bag，list 和 array 等构造子，生成复杂值。例如：

```
select struct(me:p.name, my_address:p.lives_in.
              building.address,
  my_children:(select struct(name:c.name,
    address:c.lives_in.building.address)
    from c in p.children))
from p in Persons
```

其结果值类型为：

```
struct result_type{
  String me;
  Address my_address;
  Bag(struct{String name;Address address}) my_
    children;
}
```

在 ORION 一类的纯粹面向对象数据库系统中，可以没有集合概念，用类代替集合，查询语言就是对类进行查询，它的每个结果也都是类，这样上例则需要构造一个结构如上的类 Newclass，但是类比集合具有更丰富的语义，要考虑类的属性、方法、以及它在类层次中的位置，而且因为 my_children 属性还是复杂值，也要为它再构造一个类。由于新类是对原来的类的信息的重新表达，新类和原类之间存在冗余的信息，因此数据的更新容易出现异常，给系统维护和用户使用带来负担。这是 ODMG 模型中包含面向值概念的一个原因。

OQL 中还可创建复杂对象，但它必须使用类名作为构造子，例如：

```
Building(address:struct(number:10,street:"Main
street"),
apartments:List_apart(Apartment(number:1),A-
partment(number:2)))
```

其中的 Apartment(number:1)和 Apartment(number:2)同时又生成了两个对象，如果把这些对象命名或加入持久集合中，就相当于 SQL 中的 insert 语句。

5.4 方法调用

OQL 中允许出现方法，方法可出现在其结果类型与查询中的相应类型一致的任何地方，方法可以有参数，也可以没有参数。没有参数的时候看起来和使用属性一样，例如：

```
select max(select c.age from c in p.children)
from p in Persons,
where p.name="Paul"
```

而其中 age 是方法。

方法也可以返回复杂对象，从而可以继续往下导航，如：

```
b. less_expensive. is_used_by_name
```

其中"less_expensive"是一个返回 Apartment 对象的方法。

在 OQL 中允许方法调用的原因，主要是为了利用导出属性方法，这类方法的返回值相当于一个属性值，只不过这个值是计算出来的，而不是存放在数据库中的。尤其是不带参数的方法。如上例的 age，用户根本不用关心它是方法(计算出来的)还是属性(存放起来的)。

5.5 多态

在 OODB 中对一个类的查询经常意味着对一个类层次的查询，而方法在各个类中可以有不同的含义，因此需要滞后联编机制。

假设类 Employee 和 Student 是 Person 的子类，可以查询：

```
select p. activities
from p in Persons
```

其中这个 activities 方法在类 Person，Employee 和 Student 中分别有定义。

也可以显式地说明所关心对象所属的类：如：

```
select (Student)p. grade
from p in Persons
where"course of study"in p. activities
```

这种指明类的方式主要是基于程序没有 Students 集合的考虑，只能从 Persons 集合中查询 Student 的信息。

5.6 OQL 复合

OQL 是一个纯粹的函数型语言，其所有操作符都可以自由地组合，因此 OQL 简单而又功能强大。OQL 还可使用 oql 函数与 C++连接，用法如：

```
oql (result, "select p. name from p in $1c",
persons);
```

其中"$1c"是参数格式，参数格式一般形式为$(位置)(类型)，其中位置表示后接的第几个参数，类型为"o"表示对象，"c"表示集合，"i"表示整数等等。

六、结 论

由于面向对象概念被广泛应用于各个领域，所以在前段时间出现了各种各样的定义和术语，对对象数据库的标准化一直是个大问题。以前对面向对象数据库的指责也主要集中在其缺乏理论基础和统一的标准方面。近年来，关于对象数据库的理论基础作了很多研究，对象数据库标准也日益成熟。ODMG-93提供了一个设计对象数据库，用 C++或 Smalltalk 编写可移植的应用，以及用一个简单有力的查询语言查询数据库的完整框架。这个标准的出现是对象数据库市场的一件大事，其对于对象数据库的重要性就相当于关系库中的 SQL。对象数据库厂商都保证以后的版本支持此标准。

和对象数据库有关的标准化工作仍在继续，SQL3预期在1996年或1997年完成，而且现在每半月会议产生一个新的草案；ODMG 还在与 OMG 建立组织联系，与 SQL3和 OMGOSA 进一步协调，将产生 ODMG-95；还有 ANSI X3H7的对象信息管理(OIM)标准。另外，关于对象数据库的性能测试标准也日趋成熟，象001 Benchmark、HyperModel Benchmark 和007 Benchmark 已被广泛接受，作为对象数据库评价的标准。这些工作将对对象数据库系统的研究、实现和应用产生巨大影响。

参考文献

[1] Rick Cattell, et al., The Object Database Standard: ODMG-93, release 1.1. Morgan Kaufmann, 1994

[2] Alfons Kemper et al., Object-Oriented Database Management, 清华大学出版社 and Prentice-Hall, May 1994

[3] Jose A. Blakeley et al., Object Database Management: Database Design and Open Systems, Tutorial of Intl. Conf. on VLDB, 1994

[4] Andrew Ensenberg et al., SQL-92 and SQL3, Tutorial of Intl. Conf. on VLDB, 1993

[5] Michael J. Carey et al., The 007 Benchmark, In Proc. of ACM SIGMOD Conf., May 1993

[6] Francois Bancilhon et al., (eds.), Building an Object-Oriented Database System, The Story of O2, Morgan Kaufmann Publishers, San Mateo, California, 1992

[7] ANSI/ASC/X3/SPARC/DBSSG/OODBTG, OODBTG Final Report, 1991