# ONet 对象数据库管理系统架构设计

董颖涛　陈　奇　俞瑞钊
（浙江大学人工智能研究所　杭州 310027）

**摘　要**　本文主要讨论基于面向对象数据库标准 ODMG 3.0 的数据库管理系统 ONet 的系统架构设计。介绍了 ONet 系统的页面服务器和对象服务器相结合的客户端服务器体系结构，和它的对象模式服务器以及对象组件的设计。

**关键词**　对象数据库管理系统，ODMG，ONet，对象模式

## Design of Architecture of ONet—An Object Database Management System

DONG Ying-Tao　CHEN Qi　YU Rui-Zhao
(Insititute of Atificial Intelligence, Zhejiang University, Hangzhou 310027)

**Abstract**　This paper introduces the architecture of ONet, an object database management system based on ODMG 3.0 standard. ONet adopts an architecture combining the advantage of page-server and object-server, two kinds of client-server architectures. The design of object component and object schema server is also discussed.

**Keywords**　ODBMS, ODMG, ONet, Object schema

## 1　引言

在传统的关系数据库占据数据库主流近 30 年的今天，面向对象数据库(OODB)为了满足日益增长的新需求已经不是什么陌生的概念了。OODB 因其相对于传统关系数据模型独特的优势，已经在诸如 GIS(地理信息系统)、CAD(计算机辅助设计)、数字图书馆等领域得到广泛的应用。

目前对于面向对象数据存储的研究主要有两个方向：一是以关系数据库为基础，向其中加入对象的特性，使之具有面向对象的表达能力，这种方式的数据库产品发展中比较普遍，各个主流的关系数据库平台如 Oracle、DB2、SQL Server 等都在向这方面发展。SQL3 标准也引入了对象模型的表示[5]。而这种数据库就是通常所说的对象关系数据库(ORDB)。

另一个方向是以面向对象程序设计语言为基础，加入数据库相应的特性。这种方式就是这里讨论的 OODB。主要的产品有 GEMSTONE、ObjectStore、ONTOS、Matisse 等。现有的 OODB 的主要标准是 ODMG 标准，目前的版本是 3.0。本文介绍的 ONet 系统就是依照 ODMG 3.0 标准实现的对象数据库管理系统(ODBMS)。

ODMG (Object Database Managerment System Group) 是由各大对象数据存储厂商组成的组织，在 1993 年推出 ODMG 1.0 标准以来，一直是对象数据存储的主要标准组织。

1999 年，ODMG 组织推出 ODMG 3.0 标准[1]，其主要内容如下：

1. 对象模型，是 ODBMS 支持的公共数据模型，它参照了 OMG 的对象模型，并对它做了必要的扩充。

2. 对象定义语言(ODL)，它区别于传统的数据库数据定义语言(DDL)，而以 OMG 接口定义语言(IDL)作为 ODL 语法的基础。

3. 对象查询语言(OQL)，使用关系数据库标准 SQL92 作为对象查询语言 OQL 的基础。

4. C++语言绑定，ODMG 3.0 规定了如何写管理持久对象的可移植的 C++代码，这称为 C++ OML(对象操纵语言)。C++绑定也包括了一个符合 C++语法的 ODL 版本、执行 OQL 的机制以及有关数据库操作和事务管理的成分。

5. ODMG 3.0 还分别定义了 SmallTalk 绑定和 Java 绑定。

ONet 系统是由浙江大学人工智能所智能软件实验室自主开发的面向对象数据库管理系统，由该实验室早期的面向对象智能决策支持开发环境 Z 系统中对象存储模子系统演变而来，经历了 TR-OODB 原型系统的开发，在 2002 年新版更名为 ONet。ONet 系统遵循 ODMG3.0 标准，主要实现其中的 C++绑定接口。

作为一个 ODBMS，ONet 最直接的功能就是它能够允许用户操纵任意复杂的对象模型以及它们之间任意复杂的相互关系。ONet 可以让应用程序编写者管理简单的数据对象如整数、字符串等；也可以定义复杂的数据结构；同时还可以管理除了 1-1，1-*N* 之外的双向关系，这些关系用 list(链表)、set(集合)等来表示。复杂的数据类型和关系不仅丰富了程序的表达能力，同时也提高了程序获取数据的效率。

在体系结构上，ONet 采用典型的 C/S 架构，服务器能够大数据量、高负荷地同时为多个客户端服务。ONet 提供单个数据库容量 32TB 存储能力和同时允许数十个客户端同时连接的吞吐量。在客户端提供对象、页面双重缓冲区，大大提高了数据访问的效率，同时自动维护对象完整性。

ONet 系统还提供数据库管理系统本身所具有的基本功能，如并发控制、故障恢复等功能。提供一个良好的对象数据存储解决方案。目前 ONet 系统已经在我们与浙江成功软件开发公司合作开发的 GIS 平台 GNet2003 中得到了良好的应

**董颖涛**　硕士研究生，主要研究方向：面向对象技术数据库。**陈　奇**　副教授，主要研究方向：面向对象技术、ERP 系统、人工智能、地理信息系统。**俞瑞钊**　教授，博士生导师，主要研究方向：智能软件技术、数据挖掘、数据库和知识库系统。

用。

## 2 ONet 系统体系结构

ONet 系统采用 C/S 体系结构。ONet 应用体系主要由对象模式服务器、数据服务器和客户端进程组成。图 1 所示的是 ONet 系统的体系结构。

整个系统中有一个对象模式服务器，用来存储数据库的对象模式信息，这些模式信息使用 XML 进行存储和交换。对象模式服务器以独立进程方式运行在网络上任意位置，可以通过 TCP/IP、Web Service 的方式被访问。

系统中可以分布多个数据服务器，用来存储对象数据，它们在运行的时候向对象模式服务器询问对象模式信息，并向客户端是供对象数据访问。

每一个数据服务器可以同时供多个客户端连接。客户端进程中有一部分是 ONet 提供的运行时库，客户端进程就通过这个库和服务器进行交互。服务器和客户端之间通过 TCP/IP 进行通讯。其中 ONet 运行时实现了 ODMG 3.0 OML 接口和经过简化的部分 OQL 查询接口。

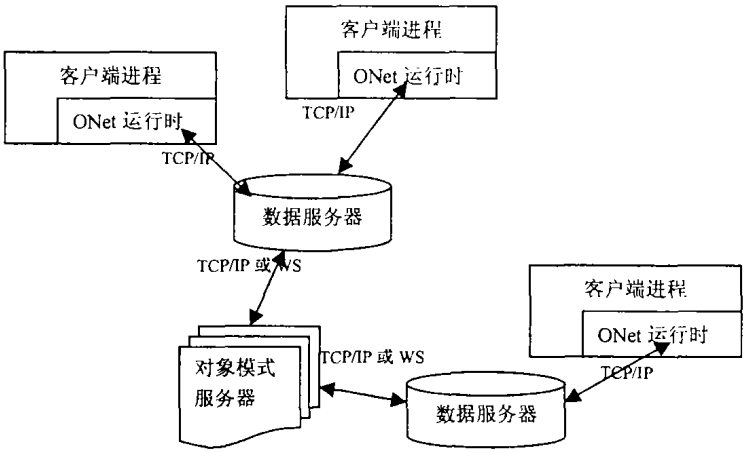


图 1

## 3 关键技术和实现策略

### 3.1 客户端/服务器体系结构

3.1.1 *传统客户端/服务器体系结构* 在传统的面向对象数据库管理系统实现中，主要有三种体系结构，它们是：对象服务器、页面服务器和文件服务器结构。表 1 对这几种结构进行了比较[2,3,6]。

表 1

| | 对象服务器 | 页面服务器 | 文件服务器 |
|---|---|---|---|
| 数据传输单位 | 对象 | 页面 | 数据库文件 |
| 服务器缓冲区 | 对象、页面双缓冲 | 页面缓冲区 | 文件缓冲区 |
| 客户端缓冲区 | 对象缓冲区 | 对象、页面双缓冲 | 文件缓冲区 |
| 并发控制粒度 | 对象锁 | 页面锁 | 文件锁 |
| 查询执行位置 | 服务器 | 客户端 | 客户端 |
| 数据聚簇性 | 差 | 好 | 好 |
| 网络消息 | 频繁 | 不频繁 | 不频繁 |
| 对象模式信息 | 服务器 | 客户端 | 客户端 |

从上面的表中可以看出，在性能上，文件服务器通过网络文件系统直接读、写数据文件，有比较好的性能。但文件服务器数据操纵能力较弱，并且难以实现有效的并发控制。

传统的对象服务器在服务器端管理对象，而客户端相对轻型，不必知道任何有关存储的细节。而同时对象模式信息存储在服务器上，客户端是纯粹的对象访问功能。服务器保留对象模式信息使得服务器可以对象执行查询处理，也使得集中式的并发控制很自然地得以实现。其缺点也很明显，过小的网络传输粒度会造成频繁的网络通讯，在最坏的情况下，这种开销可能迅速造成系统瓶颈。

传统的页面服务器结构中，在服务器没有对象管理模块，这使得页面服务器和对象服务器有本质的区别，因为服务器不再知道任何有关对象的逻辑模式信息。这种模式下，由于可以有效利用数据库的聚簇性，在性能上比对象服务器要好。但是，这种方式下，由于服务器不知道对象如何组织，因此很难实现对象级别的并发控制。而查询放在客户端执行会使得海量数据处理变得效率低下。

3.1.2 *ONet 系统的客户端/服务器体系结构* 上面对三种传统的 ODBMS 实现的结构进行了比较。事实上，我们在实现 ONet 系统的时候，综合了页面服务器、对象服务器架构的优点，采取了一种较为折中的方案。

ONet 采用了页面服务器和对象服务器结合的方式。服务器端，是一个对象服务器，它能够完整地操纵数据库中的所有对象，插入、删除对象、执行查询、完成对象级别的并发控制。服务器和客户端之间数据交换的粒度是页面。但是 ONet 的服务器并不是传统意义上的页面服务器，它可以看成是一个轻量级的对象服务器，它不具有对象存储、故障恢复等功能，但是它具有对象级别的操作能力以及客户端内部的对象级别的并发控制，它操纵的数据就是位于客户端的缓冲区中的数据。

ONet 系统的缓冲区结构有别于传统的页面服务器结构和对象服务器结构，传统的结构要么在服务器端处理对象、要么在客户端处理对象，因此只会有一端有对象缓冲区。而 ONet 体系中，服务器端和客户端都有操纵对象的能力，它们分别有自己的对象缓冲区和页面缓冲区。服务器端的对象缓冲区主要用来完成服务器端的对象查询，而客户端的对象缓冲区则用来完成客户端的对象查询。

通过对 ODMG 3.0 标准[1]的研究，我们发现，应用程序使用 ODMG OML 查询对象时，这些查询适合于在客户端处理；而当应用程序使用 ODMG OQL 查询对象时，这些查询请求适合于在服务器处理。因此，ONet 底层为 ODMG OML 和 ODMG OQL 提供了不同的查询接口，服务器的对象缓冲区主要负责 OQL 的查询处理需要，而客户端的对象缓冲区主要负责 OML 的查询处理。而这两个缓冲区之间并不直接交

互，服务器和客户端之间的数据交互主要通过它们各自的页面缓冲区完成。这样做一方面是效率的原因，页面可以充分利用数据库的聚簇性；另一方面是考虑到客户端可能运行在不同的平台上，使用不同的面向对象语言。因此，在服务器端对象缓冲区中的对象内存结构和客户端的对象内存结构并不一定相同，这样它们之间的交互缺乏实际意义。图 2 描述了 ONet 系统在这样的缓冲区结构下的查询处理过程。

ONet 系统的并发控制采用集中式的多级锁管理，并实现了严格可串行化的事务隔离级别。在服务器端集中管理所有的锁，基本的锁粒度分两级：页面上的锁和对象上的锁。在页面上的锁分为 IS、IX、SIX、SH、UD、EX 六种，而在对象上的锁分为 SH 和 EX 两种。其它类型的锁，如外延锁、集合锁等，均基于上述几种基本的锁类型实现。考虑到完全在服务器管理锁的性能瓶颈，ONet 系统在客户端缓冲页面级别的 SH 锁，锁缓冲算法采用目前很多 DBMS 厂商都采用的 CallBack 算法[4]。

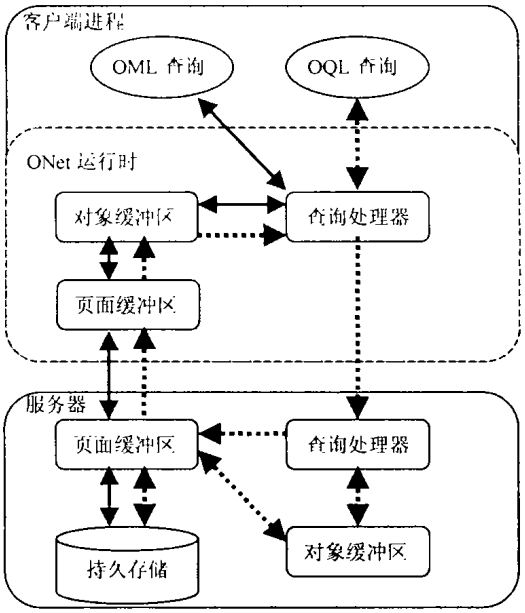


图 2

### 3.2 对象模式服务器

对象模式信息是对象数据库系统中至关重要的部分，它记录了所有持久对象的描述信息，如类型、类型之间的关系、存储结构等。对象模式信息在应用程序访问数据库时，起着指导 ODBMS 如何操纵持久对象的作用。ONet 系统在遵循 ODMG 标准 C++绑定的基础上，对 ODMG 对象模式的管理模式做了一些扩展。

3.2.1 ODMG C++绑定的处理流程 图 3 描述了 ODMG ODBMS 的整个使用流程[1]。在 ODMG 的数据库模型中，数据模式信息来源于 ODMG ODL 说明。这个说明类似于编成语言中的类声明，它描述了应用程序需要持久化的类型、这些类型之间的关系等信息。这个 ODL 说明在被 ODBMS 提供的 ODL 预处理器处理错之后，就会生成 ODBMS 所需的对象模式信息和为了支持持久对象执行的相应的 C++头文件和源文件。生成的对象模式信息被存储到 ODL 仓库(ODL Repository)中去，而生成的 C++头文件和源文件和应用程序开发者编写的 C++ OML 程序一起通过 C++编译器编译成目标代码，然后和 ODBMS 的运行库链接成应用程序可执行文件。应用程序运行的时候，通过 ODBMS 运行时和数据库进行交互。

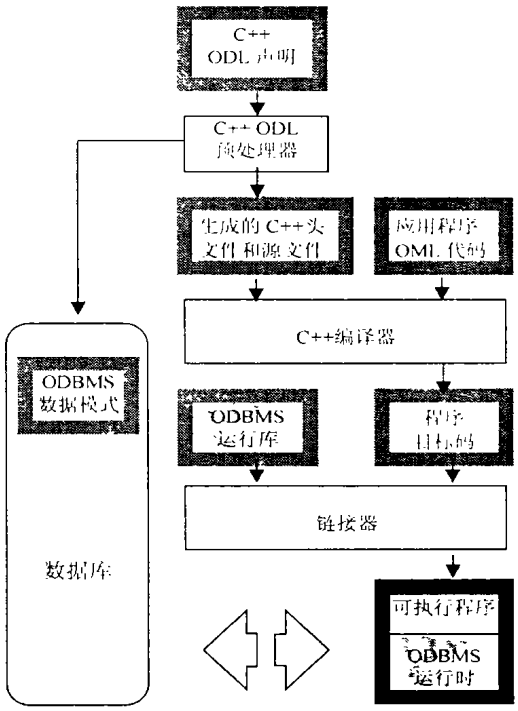


图 3

3.2.2 ODMG C++绑定流程的缺陷 ONet 系统实现了 ODMG C++绑定的基本流程，但在实际应用中遇到一些问题。

首先，ODMG 的整个流程事实上可以使应用程序开发者不必关心底层的对象如何被存储，如何被装入内存，只需要像普通的 C++对象一样使用就行了。但是由于 C++语言的特殊性，C++语言的应用程序二进制接口(ABI)还没有一个统一的标准，因此 ODMG ODBMS 的实现者在不知道用户使用的 C++编译器的情况下，无法仅仅通过 C++ ODL 中的信息直接创建其中定义的 C++对象。具体地说，在没有编译器帮助的情况下，ODBMS 一旦将对象数据存入磁盘，一些非持久性的信息就丢失了，比如对象的虚函数表等，这样也就没有办法再让这些数据还原成一个在内存中可以执行的 C++对象了。所以，ODL 预处理器必须生成一些辅助的 C++头文件和源文件来完成这些工作。而生成 C++头文件和源文件会直接导致一旦持久对象的模式被修改，使用这些对象模式的应用程序就必须重新编译，使得系统设计变得非常不方便，这个缺陷在大型的系统中表现尤其严重。

事实上，这种情况很普遍。举个简单的例子，一个雇员薪金统计的应用程序，在设计的时候程序在数据库中定义了可持久类型接口 Employee 类，它有一个多态方法 void Salary() 用来计算年薪。同时定义了 Employee 类的派生类 Manager、Clerk 等类型，分别重载了相应的 Salary 方法。当系统运行一段时间后数据库中已经有一些 Employee 类型的实例。这时，公司结构调整，有了新的职位 Product Manager，而数据库也加入了相应的 Product_Manager 类，从 Employee 派生，并重载了 Salary 方法，这时使用原有的薪金统计程序由于无法将 Product_Manager 类型的数据转换成 C++对象，这就无法统计所有的 Product Manager 的薪金了。只有将 Product_Manager 类型的实现编译到系统进程中才能使用。

其次是服务器端查询问题。由于 C++绑定中，应用程序定义的持久类型的对象在执行方法的时候必须是以一个完整的 C++对象出现。而持久对象的实现代码是在客户端的进程

空间中执行的，所以凡是涉及到对象方法的查询就都只能在客户端执行。这大大限制了ODBMS的数据处理能力，对海量数据的数据库，将查询放在客户端进行，其效率是无法令人接受的。所以，ONet为了提供服务器端查询功能，必须提出一套可行的方案。

3.2.3 对象组件　针对上面这个缺陷，ONet提出了一种有限的解决方案——对象组件。图4描述了ONet扩展之后的整个流程。

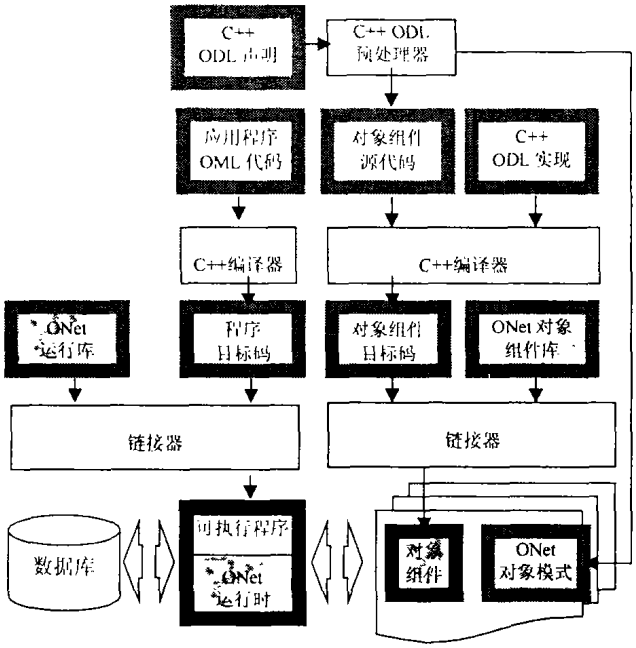


图4

这个流程和ODMG原来的流程的根本不同之处在于，它将应用程序的ODL类型实现封装成对象组件，而不是放到原来的应用程序可执行文件中去。考虑到执行效率，ONet对象组件是进程内的组件，它通过动态链接库的方式被装入可执行程序的进程空间，进而在应用程序中提供持久对象的运行时行为。而应用程序只需要将使用到的接口声明包含到代码里编译生成可执行文件，就可以通过ONet运行时访问对象组件，ONet运行时提供了一整套ODMG OML和OQL的接口。

有了对象组件，ONet就可以在服务器端执行查询了，当ONet的查询进程（为了保证服务器的安全，ONet服务器端查询被单独的进程处理，这里称为查询进程）获得查询请求之后，它就可以向对象模式服务器请求与服务器平台相符的相应版本的对象组件，动态装入，执行查询。客户端得到的查询结果仍然是数据库中对象的内部格式，客户端则使用与自己平台相符的对象组件来访问对象数据。因此，客户端和服务器不一定需要在相同的平台上运行，只要应用程序开发者为它们相应平台编译生成了对象组件，就可以执行相应端的查询。

有了对象组件，应用程序在很多情况下不需要重新编译，就能够访问修改数据模式之后的数据组件。在前面3.2.2节的例子中，ONet运行时会自动装入Product_Manager类的对象组件，自动重构该类对象，并使应用程序得到Employee类型的引用，而应用程序不需要任何修改。

3.2.4 对象模式服务器　为了统一注册、管理对象组件，同时由于对象模式信息的频繁读、不频繁改的特点，ONet引入了对象模式服务器，将整个ONet应用系统中的对象模式信息和所有版本的对象组件放在一个专门的服务器上进行管理。所有的数据库服务器在需要相应的对象模式时，以数据库为单位向对象模式服务器索取，同时在需要时索取相应版本的对象组件。

应用程序在ODMG的访问接口中不能修改对象模式，必须通过ODL预处理器或者ONet提供的专门可视化设计工具定义、修改对象模式，得到的对象模式信息存储在对象模式服务器。ONet在XML上定义OSchemaML表示对象模式信息，并通过TCP/IP或者Web Service和数据服务器进行交互，可以适应主流的网络环境。同时其他分布式计算环境协作打下基础。

对象模式服务器的引入，使得应用程序编写者能真正做到编写一次、多次使用，达到对象组件的共享功能，真正发挥面向对象设计的威力。

3.2.5 存在的不足　前面提到对象组件方案仍是个有局限性的方案，主要是因为存在以下的不足：1）对象组件在不同的平台必须有不同的版本，需要相应平台的编译器；2）对象组件或对象组件群内部必须是类型封闭的，无法引用到应用程序中对象组件以外的类型；3）对象组件依赖于动态链接库技术，在没有该技术的系统上无法执行。

目前的ONet也同时支持不使用对象组件，而遵循ODMG标准流程应用。惟那些在以上几个方面有特殊要求的应用系统提供另一个选择。ONet也将不断改进对象组件的模式，使其有更广泛的应用领域。

**结束语**　本文主要讨论了面向对象数据库管理系统ONet的体系结构以及选择这样的体系结构的原因。对象访问采用页面服务器和对象服务器相结合的方式，提供高效的对象查询功能。而对象模式访问则采取对象模式服务器的方式，运用对象组件，提供高度的灵活性。

从1999年至今，ONet系统发展的时间并不长，已经得到有效的应用。但其中遇到许多问题，如本文提到的对象组件方案也只是个有限的解决方案，希望在今后的研究和发展中能有所突破。目前我们正在OQL、对象模式演化、XML数据管理等方向上进行深入的研究，许多已经解决和尚未解决的问题将在后续的文章中进行讨论，希望为对象存储领域做出更大的贡献。

## 参考文献


1　The Object Data Standard: ODMG 3.0. Morgan Kaufmann Publishers, 1999

2　Dewitt D, Futtersack P. A Study of three alternative workstation-server architectures for object oriented database systems. In: Proc. of the 16th VLDB Conf. 1990. 107～121

3　马明．高性能面向对象数据库ONet服务端存储管理的设计与实现：[浙江大学硕士学位论文]. 2002

4　Franklin M J, Carey M J. Client-Server Caching Revi-sited. University of Wisconsin - Madison, Computer Science Technical Report #1089, May, 1992

5　曹渝昆，等．SQL3对面向对象技术的支持．计算机工程与科学．2002,24(4)

6　于戈，等．一个NT平台上分布式对象数据库系统．软件学报，2002,13(4)

7　Wietrzyk S, Orgun M A. VERSANT architecture: Supporting High-Performance Object Databases