第一章 概 述

1.1 面向对象技术概述

随着计算机科学的发展、应用领域的不断扩大，对计算机技术本身的要求越来越高。特别是当计算机硬件有了飞速发展之后，各种应用领域对软件提出了更高的要求。例如，提高软件质量，缩短软件开发周期，提高软件的可靠性、可扩充性和可重用性等。计算机软件技术的发展严重滞后于硬件的发展，在60年代后期产生软件危机，使得软件的结构和简明性成为语言设计的基础，而不是软件表达能力。软件工程学应运而生，提出了一些结构化程序设计语言和结构化分析与设计方法作为软件设计的基础。对于软件，人们认识的重点逐步从组成程序的语句序列转到了构成软件的模块序列。但并未从根本上解决软件危机，世界各国的软件界人士仍在不断地研究新的方法、新的技术，探索新的途径。面向对象技术的研究及应用即是近20年来在学术界及工业界逐渐形成的一大热点。

面向对象(Object-Oriented)的思想最初出现于挪威奥斯陆大学和挪威计算中心共同研制的仿真语言Simula67中，其后，随着位于美国加州的Xerox研究中心推出SmallTalk-76和SmallTalk-80语言，面向对象的程序设计方法得到比较完善的实现。面向对象技术的出现引起了计算机界的极大关注，因为面向对象的技术对于软件工程学面临的困境是一个很有希望的突破口，对这项技术的研究和应用得到了迅猛发展。

面向对象的研究遍及计算机软、硬件的各个领域，如面向对象的程序设计语言，面向对象的程序设计方法，面向对象的设计，面

向对象的分析，面向对象的操作系统，面向对象的 DBMS，面向对象的开发工具，面向对象的开发环境等。目前不仅在研究领域已取得了丰硕的成果，而且有些软件产品已经投放市场。

面向对象的方法是一种分析方法、设计方法、思维方法和程序设计方法，面向对象方法学的出发点和所追求的基本目标是使我们分析、设计和实现一个系统的方法尽可能接近我们认识一个系统的方法，也就是使描述问题的问题空间和解决问题的方法空间在结构上尽可能一致。其基本思想是：对问题空间进行自然分割，以更接近人类思维的方式，建立问题域模型，以便对客观实体进行结构模拟和行为模拟，从而使设计出的软件尽可能直接地描述现实世界，构造出模块化的、可重用的、维护性好的软件，并能控制软件的复杂性和降低开发维护费用。面向对象方法中，对象作为描述信息实体的统一概念，把数据和对数据的操作融为一体，通过方法、消息、类、继承、封装和实例化等机制构造软件系统，并为软件重用提供强有力的支持。

1.2 面向对象的基本概念

面向对象技术为软件开发提供了一种新的方法学，引入了许多新的概念，这些概念是理解和使用面向对象技术的基础和关键。

1. 对象、方法及消息(Object, Method, Message)

客观世界的问题都是由客观世界中的实体及其相互之间的关系构成的。我们将客观世界中的实体抽象为问题空间中的对象。由于我们需要解决的问题不同，我们面向的对象也就不同，因此对象是不固定的。一本书可以是一个对象，一家图书馆也可以是一个对象。从动态的观点看，对象及其操作就是对象的行为，一个对象的通常定义是：对象是对一组信息及其上面的操作的描述。对象由以

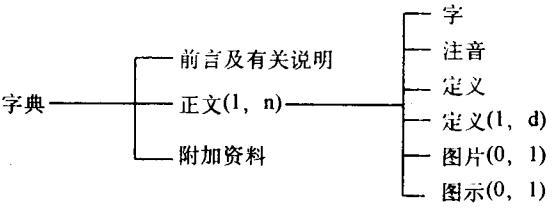
下部分组成：

• 私有数据结构和处理，这些处理又称为操作(Operation)或方法(Method)，包括控制和过程。其中私有数据表示了对象的状态，该状态只能由私有操作来改变，每当需要改变对象的状态时，只能由其它对象向该对象发送消息。

• 消息是用来请求对象执行某一操作或回答某些信息的要求，消息统一了数据流和控制流，程序的执行是靠在对象间传递消息来完成的。表示消息的形式是消息模式。对同一消息模式的不同消息，同一对象所作的解释和处理都相同，但是会由于对象状态的不同而导致操作结果不同。一个消息模式定义对象的一种处理能力，所有消息模式及相应于消息模式的处理能力，定义了对象的外部特征。

我们用一个例子来说明对象操作及消息之间的关系。如下页图 1.1 所示。

现实世界的对象"字典"被映射为软件实现时，包括私有数据结构和一组相关操作，其私有数据结构如下：



有关数据结构的一组操作包括：Add_word，Delete_word……；外部对象通过字典对象的接口向对象发消息(如 Find_word(word))，字典对象才可以执行相应的操作。

我们对对象的初步概念可以理解为：对象是一个封装数据和操作的实体。对象的结构特征由属性表示，数据描述了对象的状

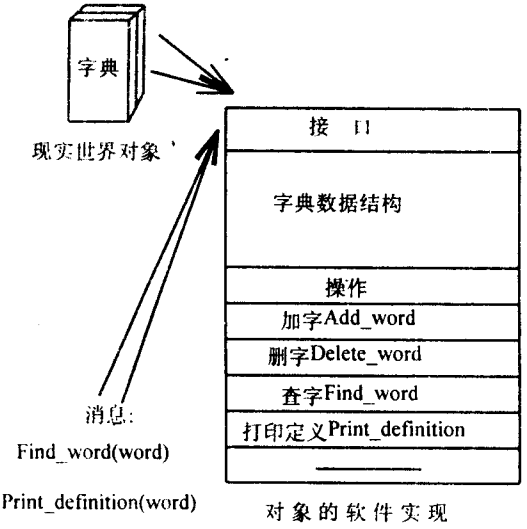


图 1.1 对象操作及消息之间的关系

态，操作可操纵私有数据，改变对象的状态，每当其它对象向本对象发出消息，本对象响应时，其操作才得以实现。

2. 类(Class)、实例(Instance)、继承性(Inheritance)

在客观世界中，有许多具有相同特征的事物，如：小轿车、大客车、卡车等，可以归类为机动车。从对象观点看，具有共同的属性、共同的操作性质的对象的集合就是类，而单个对象则是对应类的一个实例。例如：书是一个类，而某一本具体的书如《面向对象的分析》则是该类的一个实例，任何一个对象都是某一个类的实例，并继承该类定义的私有数据和操作。这就是继承性，一个类实质上定义的是一种对象类型。

我们仍可以用有关书的例子来说明类、实例与继承性的关系。

如图 1.2 所示：

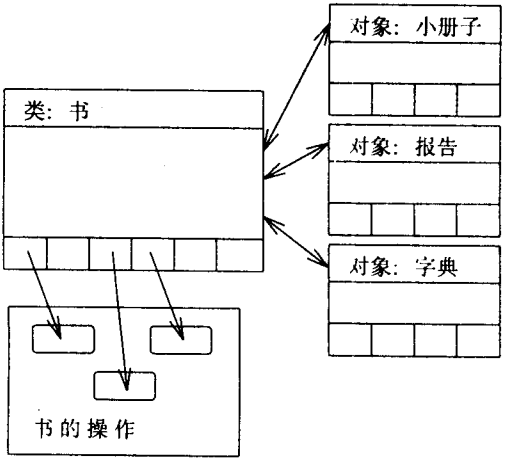


图 1.2　类、实例与继承性的关系

书是一个类，其中定义了私有数据结构及一组有关的操作，而对象小册子、报告、字典等则是该类的实例，它们继承了该类中定义的数据结构及操作。

类构成层次结构，相对上层的是超类，相对下层的是子类，子类在继承超类的私有数据结构及操作的同时可以拥有自有的私有数据结构及操作。如果一个子类只有一个超类，则称为单继承性；如果一个子类具有多个超类称为多继承性或多重继承，这时该类共享多个超类的属性及操作。

类和继承性是现代软件工程中的重要概念，软件的可重用性、程序成分的可重用性是通过继承类中的属性和操作而实现的。许多工业观察家相信可重用软件不是通过建立传统的过程库（子程序库），而是通过建立“类库”实现的。

3. 封装性(Encapsulation)

对象的封装性是面向对象技术的一个重要特征。对象本身的定义即提供了封装性。

对象的封装性是一种信息隐蔽技术，对象的使用者只能看到对象封装界面上的信息，对象的内部对使用者是隐蔽的，其目的在于将对象的使用者和设计者分开。对象的封装性体现在以下几个方面：

- 对象具有清楚的边界：对象的内部软件(数据结构及操作)的范围，限定在这个边界之内。
- 对象具有统一的外部接口：对象的接口(消息模式)描述该对象与其他对象间的相互作用。
- 对象的内部实现是不公开的：对象的实现给出了对象提供的功能细节，外部对象是不能访问这个功能细节的。

信息隐蔽是软件开发过程中强调的一个重要概念，对象的封装性很好地体现了这一概念。这就使得用面向对象技术所开发设计的软件的可维护性大为改善，这也是软件技术追求的目标之一。

第二章　面向对象技术在数据库系统中的应用

2.1　概述

数据库系统是信息系统的核心，一般地说，综合的信息系统就是大型数据库应用系统。将面向对象技术应用到数据库系统中，这是数据库应用发展的迫切需要，也是面向对象技术和数据库技术发展的必然结果。面向对象技术在数据库系统中的应用主要体现在数据库管理系统和数据库应用开发工具两个方面，即面向对象的数据库系统和面向对象的数据库应用开发工具。

数据库管理系统是建立信息系统的基础。将面向对象技术应用到数据库管理系统中，使数据库管理系统能够支持面向对象数据模型，这对于提高数据库系统模拟客观世界的能力，扩大数据库应用领域具有重要的意义

数据库应用开发工具是信息系统开发的必备环境，将面向对象技术应用到数据库应用开发工具中，使数据库应用开发工具能够支持面向对象的开发方法并提供相应的开发手段，这对于提高应用开发效率，增强应用系统界面的友好性、系统的可伸缩性、可扩充性等具有重要的意义。

2.2　面向对象的数据库系统

1. 应用的需求

数据库技术自 60 年代后期问世以来，无论从理论上，技术上，

还是应用上，都经历了一个飞速发展的过程。现在，大型信息系统一般都是以数据库系统作为其核心的。

从数据库系统采用的数据模型来看，70年代广为流行的是网状模型和层次模型的数据库系统。它们采用记录的汇集，以及记录之间一对多(一对一)的联系来描述现实世界客观事物，用过程化的数据操纵语言来提供数据管理功能。自80年代以来，关系模型的数据库系统逐步取代了网状模型和层次模型的数据库系统。关系模型建立在集合代数的基础之上，用关系(二维表)来描述现实世界客观事物，用面向集合的非过程化的数据操纵语言来提供数据管理功能。由于关系模型有严格的数学基础，概念简单清晰，非过程化程度高，数据独立性强，因此关系型数据库系统的发展非常迅速，80年代以来，计算机厂商新推出的数据库管理系统几乎都是支持关系模型的。

随着数据库技术的发展，数据库应用领域已从传统的商务数据处理扩展到许多新的应用领域，例如计算机辅助设计(CAD)、计算机辅助软件工程(CASE)、图象处理、超文本应用等，关系数据库管理系统很难适应这些新应用领域中模拟复杂对象，模拟对象的复杂行为的需求。甚至在传统的商务数据处理应用中，也提出了新的处理需求，例如存储和检索保险索赔案件中的照片、手写的证词等，这些要求也是传统的关系数据库系统难以满足的。

新的应用需求推动了数据库新技术的研究，其中最重要的研究方向之一就是将面向对象技术与数据库技术相结合，研究新型的数据库管理系统——面向对象的数据库系统。利用类的设施来描述复杂对象，利用对象中封装的方法来模拟对象的复杂行为，利用继承性来实现对象结构和方法的重用。

2. 面向对象数据库系统的特性

面向对象数据库系统的研究始于80年代中后期，对于什么是

面向对象的数据库系统，目前尚缺乏权威性的统一标准。然而，对于面向对象数据库系统应该具备的基本特性，国际数据库学术界已取得了大体一致的共同认识。

首先，面向对象数据库系统必须支持面向对象的数据模型，具有面向对象的特性。这些特性主要包括：支持复杂对象，具有对简单对象运用各种对象构造符组成复杂对象的能力；具有对象标识，对象独立于它的值而存在；具有封装性，数据库对象中既封装数据又封装程序，从而达到信息隐蔽，同时也是逻辑数据独立性的一种形式；支持类型和类的概念，类型概括具有相同特性的一组对象的共同特性；支持类或类型的层次结构，从而支持继承性这一有力的建模工具；允许过载，即将同一名字用于不同类型上的数据操作；通过与现有程序设计语言的合理连接来达到计算完备性；并具有可扩充性。

其次，面向对象数据库系统必须是一个数据库管理系统，具有数据库管理系统的基本功能。主要包括：持久性，数据库中的数据是持久保存的；外存管理，包括索引管理、数据聚集、数据缓冲、存取路径选择、查询优化等；并发性，系统应该提供和目前的数据库管理系统同样级别的，对多个用户并发操作数据库的支持；故障恢复，系统应该提供和目前的数据库管理系统同样级别的，将数据库从故障后的错误状态恢复到某一正确状态的功能；以及即席查询功能，查询功能应该是非过程化的，高效的，独立于应用的。

面向对象的数据库系统除了必须具备上述面向对象特性和数据库管理系统基本功能外，最好还能具备新应用领域所需要的一些进一步的特性，例如模式演化、版本管理、长事务和嵌套事务、分布式计算等。

3. 面向对象数据库系统的优越性

面向对象数据库系统将面向对象的能力赋予了数据库设计人

员和数据库应用开发人员，从而可以大大扩展数据库系统的应用领域，并且提高开发人员的工作效率和应用系统的质量。

(1) 复杂对象构造能力使得对于客观世界的模拟能力强，方式自然

关系数据库系统强迫用户用多个关系的元组来表示层次数据、嵌套数据、或复合数据。例如，职工有职工号、姓名、性别、工资、部门等属性，而部门又有部门号、部门名、部门性质、部门经理等属性。关系数据库中属性的取值只能是基本数据类型，这样，职工元组中的部门属性取值只能是部门号。要查询某职工及其所在部门的信息就需要做"职工"和"部门"这两个关系的连接。这样的表示方式既不自然，又影响查询的速度。面向对象数据库中对象的属性的取值可以是另外一个对象，一个职工对象的部门属性的取值就可以是该部门对象，实际储存的是该对象的对象标识，这样的表示方式自然、易理解，而且在查询某职工及其所在部门的信息时可以通过该部门的对象标识直接找到那个部门，提高了查询的速度。

(2) 封装性向开发人员和最终用户屏蔽复杂性和实现细节，降低了数据库应用系统开发和维护的难度

对象封装将程序和数据封装在一起作为存储和管理的单位，也是用户使用的单位，从外部只能看到它的接口，而看不到实现的细节，对象内部实现的修改不影响对对象的使用，因此使应用系统的开发和维护都变得更加容易。关系数据库系统现在支持存贮的过程，即允许程序用某种过程性语言编写并存入数据库中以备以后装载和执行。但是，存储的过程并不和数据封装在一起，即它们不和任何关系或关系的元组相关联，构成一个整体，其信息隐蔽和易维护性显然不如面向对象数据库系统中封装起来的对象。

(3) 继承性使得数据库设计和应用编程成为可重用的

在面向对象的数据库系统中，类的定义和类库的层次结构体现了系统分析和数据库设计的结果，即体现了客观世界中对象的

内部结构及对象之间的联系。同时，类定义中封装的方法保存了数据库应用编程的结果。应用开发人员可以在已建立的类库的基础上派生出新的类，继承已存在的类的属性和方法。例如定义“销售人员”类作为已存在的“职工”类的子类，“销售人员”可以继承“职工”的职工号、姓名、性别等属性，重用了数据库设计的结果，还可以继承“职工”的计算工资额、显示奖惩记录等方法，重用了应用编程的结果。数据库设计和应用编程的重用对于建立大型复杂的数据库应用系统具有重要意义。

2.3 面向对象的数据库应用开发工具

1. 应用的需求

为提高应用开发人员的生产率，增强数据库应用系统的界面友好性、可维护性、易扩充性，就必须有数据库应用开发工具来支持数据库应用系统的开发。

十余年来，数据库厂商和工具开发商在数据库应用开发工具上投入了大量的人力和物力，推出了若干个建立在关系数据库系统之上的应用开发工具。早期的开发工具多是字符界面的、集中式的(非客户/服务器结构的)，一般是与特定的 DBMS 配套的。

随着客户/服务器体系结构的发展，以及对全企业范围数据库应用系统的需求，数据库应用开发人员对应用开发工具提出了新的要求，要求它们支持图形化用户界面(GUI)开发，软件部件重用，开发组的工作方式，应用系统的可伸缩性、可扩充性等。与这些要求相呼应，数据库厂商和工具开发商开始研究将面向对象技术应用到数据库应用开发工具中，并开始推出面向对象的数据库应用开发工具。

2. 面向对象的数据库应用开发工具的特性

与关系数据库查询语言有 SQL 标准不同，数据库应用开发工具五花八门，没有统一的标准，当然更没有面向对象的数据库应用开发工具的统一标准。然而，我们还是可以列举出它应该具备的一些基本特性。

首先，作为数据库应用开发工具，它应该提供对应用开发的全面支持，包括图形化的界面描绘工具，应用建立工具，调试工具，图示工具，以及强有力的数据库访问能力和浏览工具等。

其次，作为面向对象的开发工具，它应该支持面向对象的开发方法，包括一个可扩充的面向对象编程语言定义、类的层次结构、继承性、多态性等。

此外，这样的开发工具还应该是客户/服务器结构的，最好具有与多种数据库服务器的开放联接。以及支持开发组的工作方式，支持应用分割等进一步特性。

面向对象的数据库应用开发工具形成了这样一种环境，允许先进的面向对象技术与成熟的关系数据库技术在同一个环境中工作，支持数据库应用系统的开发。其优越性我们将在本系列讲座的以后几讲中详述。

2.4 现状与未来趋势

1. 面向对象的数据库系统

近 10 年来，面向对象数据库系统一直是数据库学术界和工业界研究的热点之一。

自从 1987 年以来，已陆续有多个面向对象的数据库产品投入市场，此外，还有不少实验室的原型系统。这些系统在功能，体系结构，实现技术等方面没有统一的标准。比较典型的实现途径有

两种。

面向对象的数据库管理系统的一种实现途径是，以一种面向对象的程序设计语言，例如 C++或 Smalltalk 为基础，增加数据库功能，主要是持久对象和数据共享。这样实现的面向对象数据库系统与面向对象的程序设计语言紧密结合，容易被熟悉 OO 语言的开发人员所接受，而且一般具有比较高的执行效率。其不足之处是缺乏某些数据库基本特性，很重要的是缺乏与 SQL 兼容的查询设施，在安全性，完整性，并发控制，开发工具等方面也比关系数据库产品差。这类面向对象数据库产品适合于复杂数据，简单查询的应用。对于某些应用领域，例如 ECAD(Electronic Computer Aided Design)非常适用。目前市场上的这类面向对象数据库产品主要有 ObjectStore，O2，Objectivity/DB，Matisse，ONTOS，Gemstone 等。

面向对象的数据库管理系统的另一种实现途径是，从关系数据库管理系统扩展，增加面向对象特性，主要是对基类扩充，复杂对象继承性，规则系统等的支持。这样实现的面向对象数据库系统是关系数据库技术与面向对象技术的融合，支持查询语言 SQL 的超集，具备关系数据库系统的数据库功能，同时又支持面向对象特性。这类数据库系统又称作对象-关系数据库系统。对象-关系数据库产品适合于复杂数据、复杂查询的应用，随着新的多媒体应用的计算机化，随着 Web 应用的迅猛发展，对象-关系数据库系统有着广阔的应用前景，目前市场上的对象-关系数据库产品主要有 UniSQL，Illustra 等。

面向对象技术应用到数据库系统中，不会完全取代关系数据库。今后的发展趋势是：关系数据库系统，面向对象数据库系统，对象-关系数据库系统同时存在。用户根据自己的应用特色去选择适合于自己的数据库产品。

2. 面向对象的数据库应用开发工具

面向对象数据库系统现在应用还很不广泛，将来的趋势也不是取代关系数据库系统。关系数据库系统在现在和将来的相当长时间内仍将是应用的主流。因此，基于关系数据库系统的面向对象应用开发工具对于数据库应用的发展具有十分重要的意义。

目前，已有一些面向对象的数据库应用开发工具推向市场，并由于它们出色的特性而受到用户的欢迎。例如 Informix 公司的 Informix-NewEra，Borland 公司的 Delphi，PowerSoft 公司的 PowerBuilder 等。预计还继续会有新的面向对象的数据库应用开发工具陆续推出，并且这些工具在面向对象特性的支持、与多种数据库服务器连接的能力、高效性、易用性等方面将进一步改进和提高，从而为在关系数据库系统的应用开发中采用面向对象技术提供更有力的支持。

第三章　Client/Server 结构的数据库系统

3.1　Client/Server 结构概述

随着计算机网络技术的飞速发展以及一些功能强大的工作站、PC 机的出现，实际应用中的计算机系统大多采用 Client/Server(客户/服务器)模式。这样的应用模式为信息处理提供了一个高效的、成本经济的、易于扩充的解决方案。

Client/Server 结构既可指硬件的体系结构，也可指软件的体系结构。在 Client/Server 计算模型中，要完成的功能，在 Client 和 Server 间进行划分。硬件的 Client/Server 结构，通常是指把某项任务在两台或多台机器之间进行分配，其中 Client 机用来运行提供用户接口和前端处理的应用程序，Server 机向 Client 机提供所需要的服务，提供可供 Client 机使用的共享资源，例如：硬盘、打印机等。对软件系统来说，Server 将根据 Client 的请求提供相应的服务，其典型的服务有数据管理、图象处理、科学计算等。对数据库管理系统而言，Client 与 Server 的功能划分可以有多种方案，例如，其中方案之一可以是：Client 负责数据的表示和应用，Server 负责数据的存储和检索。

随着计算机应用的发展，迫切需要支持 Client/Server 结构的 DBMS，这种需求主要表现在下列两方面。

1. 大量的联机事务处理(OLTP)

在联机事务处理中，不仅需要数据的查询，更需要实时的事务

处理，进行大量的数据更新（包括：插入、删除和修改），并要求系统吞吐量高、响应时间短。联机事务处理中，通常都有上百个用户在同时使用系统。对传统的 DBMS 来说，用户数目增加时，系统性能就显著下降。必须把数据的表示及应用处理任务分配给各个 Client，来提高运行效率，满足 OLTP 的需求。

2. 企业计算机管理模式的发展

随着计算机应用的发展，企业的计算机管理模式正在从集中式走向分布式、从部门级管理走向全企业管理、从各自孤立的系统走向集成的系统。

原来采用集中方式，一台主机带多个终端的多用户系统，数据存放在主机上，所有的处理任务都由主机来完成。这种模式已无法适应信息系统的发展。不断购买更大的主机不是一种合理的方案，而且这种模式可靠性也较差，必须走向分布式，把处理和数据进行分布。原来企业中各个部门的机器管理本部门的数据，部门之间的机器是孤立的，无法实现数据的共享，随着网络技术的发展和企业计算机管理水平的提高，要求把各部门的机器联网，实现数据共享，集成各部门的数据和服务，建立全企业的管理信息系统。企业计算机管理的这一发展，迫切需要支持 Client/Server 结构的 DBMS。

3.2 Client/Server 结构的 DBMS 的功能划分

在 Client/Server 结构的 DBMS 中，在 Client 和 Server 之间必须有明确的功能划分。功能划分可以有多种方案。下面介绍三种划分方案：

一种功能划分方案（称为 CS 体系结构）是：应用的处理（包括实现用户界面与表示逻辑）由 Client 完成。数据访问和事务管理由

Server 完成。这种方案实现了功能分布，即由 Client 承担了部分处理任务，但数据都集中在 Server 中。在查询处理过程中，Client 提出查询请求，而 Server 完成对数据库的查询任务，把查询结果返回给 Client。这种方案使 Server 有更多能力完成数据访问和事务处理，从而支持更多的用户，提高系统性能，但因数据集中于 Server，因而并未解决 Server 上的主要问题，如磁盘输入/输出。

另一种功能划分方案是：Server 上主要是执行一些底层的操作，如封锁、页面读写；而由 Client 来完成查询优化和查询处理。在查询处理过程中，Client 向 Server 申请它所需要的页面，在网络上传输的是页面。在这种体系结构中，Client 分担了更多的处理任务，但数据仍集中在 Server 中。

再一种功能划分方案是：每个 Client 通过对查询结果进行缓存建立其局部数据库，而且在 Client 端都有一个功能完整的 DBMS，它与 Server 上 DBMS 相互协作来完成数据库访问。这种体系结构，不仅有功能分布，而且有数据分布，因而可以避免 Client 向 Server 再申请相同的数据，减少了网络传输和 Server 端的数据输入/输出。但在这种数据分布的环境下，数据更新要通过复杂的协议来保证数据的一致性。

当然，还可以有别的功能划分方案，因为合理的功能划分，其目的是求得 Client 和 Server 上负载的均衡分配，并减少网上的传输。例如：当 Server 端具有强大的处理能力（多个 CPU，多个硬盘驱动器等）时，那么，Server 可以承担更多处理任务，把原来由 Client 端完成的应用处理部分地移到 Server 端，由 Server 实现，例如：通过存储过程，或者采用面向对象的开发工具等。所谓存储过程就是把予先写好的，编译过的代码段（过程）存放在 Server 上，这段代码能完成一定的功能，可以在 Client 端的应用程序中直接调用该过程，这样既发挥了 Server 端机器的强大处理能力，又减少了网络的传输量，该过程中可以包含频繁的 SQL 语句的使