北京大学本科生学位论文

深圳天音通信公司购销存系统

设计与部分实现

姓名： 俞诗鹏

学号： 09601127

院系： 数学科学学院

专业： 信息科学

导师： 林作铨 教授

2000 年 6 月

目　录

摘 要

深圳天音通信公司是一个销售网络遍布全国的大型公司。由于公司管理机构庞杂，购、销、存数据在公司内部流动缓慢，严重影响了公司的日常运作以及工作效率。企业迫切需要一个方便、快捷的工具来对所有销售数据进行管理，以便为计划部门提供真实可靠的依据。

针对天音公司的具体需求，我们采用三层客户机/服务器的模式，为公司设计了一个基于广域网的购、销、存管理系统。我们力求在系统设计上充分考虑到公司运作的实际情况，并且包含必要的通用性和扩展性，为公司今后发展 B2B 的电子商务平台做好准备。

客户端到服务器端的数据传输是系统设计成败的关键。为了解决数据传输问题，我们设计了一个基于 Agent 分布式组件模式的传输组件管理，让它全权代理系统的传输工作。传输组件将负责传输数据的提取、打包、压缩、加密等一系列工作，并在对等传输端能够将数据正确恢复出来。传输组件为整个购销存系统的稳定运行提供了可靠的保障。

为了保证系统客户端的安全，我们还设计了一个基于菜单的用户权限管理。通过对不同用户定义不同的菜单权限，我们可以有效的控制各类用户对系统的使用。系统管理员可以管理用户信息，还可以直接在系统内部设置各种用户类型的权限。

关键字

三层客户机/服务器（Client/Server）结构，传输组件管理，Agent 组件，基于菜单的权限管理，用户权限管理

一、客户机/服务器计算模式

1.1 C/S 结构概述

客户机/服务器计算模式（Client/Server 结构，简称 C/S 结构）是当今 IT 产业的核心，它从逻辑上把计算过程分为两部分：客户机和服务器。

- **客户机** C/S 结构中服务的请求端，在一般的商业系统开发中主要是 GUI（Graphic User Interface）用户界面和接口的设计，负责用户数据的输入和输出、前端数据的处理和实现与服务器的接口等功能。
- **服务器** 作为接收端，响应用户的数据请求和过程调用。在接受用户的数据请求之后，服务器将执行成功或失败的结果返回给用户相应的应用程序，或通过 GUI 的形式回显给用户。

采用 C/S 结构之后，系统结构变得清晰了，客户机和服务器各司其职，相互配合以完成用户的请求。响应时间缩短了，用户的效率提高了，而且用户不必知道后台发生了什么变化。对于系统管理员来说，维护整个系统的复杂性以及费用也比较低，一般只要保证服务器的高性能以及网间联线的畅通即可。

1.2 两层 C/S 模型

1.2.1 结构简图

传统的企业级应用软件一般都采用两层 C/S 结构，即所有客户端都直接将请求发送到服务器端，服务器端对所有的客户端请求予以相应。其结构简图如下：

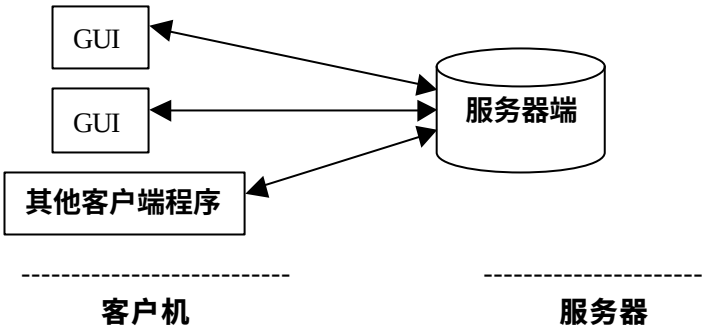

图 1-1 两层 C/S 模型

1.2.2 结构优缺点

这种两层结构的优点是结构简单，易于实现，非常适合小型系统的需要。但是对于一个大型系统，这种简单的结构就显得不适用了。它的局限性表现在如下几方面：

（1）系统的可收缩性：

在这种模型里，所有的商业逻辑必须物理地驻留在客户端或后端 DBMS 里以触发器或存储过程的形式实现。虽然数据访问得到了简化，但却缺乏灵活性，对数据源的交互作用无法进行完全的控制。实际上，在这种模式中所有的远程数据交互都是直接对远程数据库进行操作，我们无法对传输的数据进行进一步的处理，例如加密处理。对 Client 端的应用系统来说，GUI 界面和数据库访问操作是捆绑在一起的，对于商业逻辑或后台数据库的哪怕一丁点改变，前端的应用系统都必须作相应的改变，给系统的进一步开发和以后的升级、维护造成比较大的困难。另一方面，当系统扩大、用户增加或计算负荷的变化时，这种系统也无法提供灵活的系统负载从新配置或进行系统服务器的扩充。

（2）系统效率：

对于数据库应用系统来说，系统的效率问题是最关键的问题之一。采用两层 C/S 结构不利于系统总体效率的提高，这主要有两方面的原因：一方面是在数据传输的环节上，当前中国网络信息化建设相对较为落后，可用带宽较小，网络拥塞状况较为严重。况且网络状况的稳定性相对较差，对于通过远程数据库访问进行直接通信的应用系统来说，这更是一个致命的弱点。另一方面是在数据库的操作上，随着数据库中存储数据量增加，数据访问耗费的 CPU 资源将越来越多，数据操作时间加长，反应速度下降。同时，随着业务规模的扩大，用户数量的增加，使用区域范围的延伸，对数据库的同时并发操作将越来越多，前端的反应速度由于服务器资源的使用的过度拥塞将变得越来越低，甚至造成服务器的瘫痪。

（3）安全性：

随着业务范围的扩展和信息系统进一步使用，数据库中数据所包含的商业价值将越来越高，系统和数据的安全性问题很快被作为一个重要的问题给提了出来。安全性问题一般来说主要包含两方面的要求：一方面是系统和数据的安全性，另一方面是数据传输的安全性。系统和数据的安全性是指我们使用的系统架构必须是安全的结构，对于关键的数据和服务必须能设置在一个相对安全的地方，所有对它进行的访问都必须经过严格的访问控制，能阻止任何非法或无授权的访问。所谓数据传输的安全性是指所有通过公网传输的数据在传输过程中是安全的，除了解密的办法以外，任何人都无法通过网络监听等手段得到有用的信息。这种安全性的要求一般通过数据加密来获得。

对于上面提到的两方面的安全性，基于两层 C/S 结构的系统都比较难于满足。由于需要对数据库进行直接的访问，数据库必须放在从 Internet 的任何一个角落都能访问到的前台，这显然不利于数据库的安全维护。同时，所有的数据交互都是通过远程的直接写库操作来完成的，不存在任何的传输数据处理过程，所以说，在数据传输上也是非常不安全的。

1.3 三层 C/S 模型

1.3.1 结构简图

由于两层 C/S 结构存在着以上诸多的不足，目前大多数大型数据库应用系统都是采用三层 C/S 结构，即在客户端和服务器端之间增加了一层，一般称为“组件服务器”层。三层 C/S 结构的简图如下：

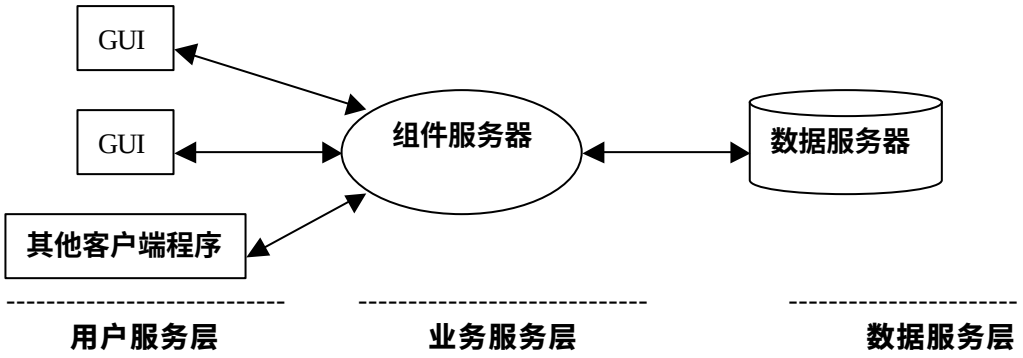

图 1-2　三层 C/S 模型

1.3.2　结构描述及优缺点

三层 C/S 体系结构一般将系统功能分为用户服务层、业务服务层和数据服务层三层，并将其抽象成对象或组件的形式。这些组件配置于一个分布式的网络环境中，通过某种通讯机制进行通讯、协调和分工，以实现相应的系统功能。

- **用户服务层**　这一部分与两层 C/S 结构中的客户端类似，也是负责读取用户的输入请求，直接与用户进行交互。不同的是，用户服务层不直接将用户请求送交服务器端处理，而是先将请求送至组件服务器，由组件服务器进行预处理；数据服务器回应的信息也是先经过组件服务器，再送交用户服务层。
- **业务服务层**　这一层正是三层 C/S 体系结构的核心部分。为了不直接使用户服务层直接与数据服务层交互，业务服务层被放置于它们之间，它的作用有两方面：对于用户服务层传送过来的请求信息，先由它进行分析，然后以一种更方便、更安全的方式送交后台的数据服务层，起到一种过滤器的作用；对于从数据服务层回传的应答信息，由它进行重新组织，并按照一定的方式回传给用户服务层，起到一种发送器的作用。
- **数据服务层**　这一层就类似于两层 C/S 结构中的服务器端。不同的是，这时整个数据服务层位于后台，对用户服务层来说是不可见的，它只与业务服务层进行交互。数据服务层的性能以及安全性不再由不可预知的客户端请求来决定，而是由业务服务层的设计保证的。只要组件服务器设计得当，系统的整体性能会比两层 C/S 结构要强很多。

三层 C/S 结构具有两层 C/S 结构不可比拟的优点，除了能够克服前面提到的两层 C/S 结构的不足之外，还有其他许多值得称道之处，我们将其简单罗列如下：

- **可重复利用**。组件设计在不同的应用程序中能够共享。
- **性能提升**。由于能在除客户机以外的其他机器上配置组件，所以可以将计算负担从功能不强的客户机转移到功能强大的服务器上，利用这种配置和设计上的灵活性，作为开发者，可充分利用计算资源使应用程序达到最佳状态。
- **容易管理**。将应用程序封装在各种组件以后，可将大型、复杂的应用程序划分为更易管理的模块。
- **容易维护**。由于将组件集中起来，以便重复使用。所以一旦需要对某个组件进行修改，可更容易的进行重新配置，以便随时适应商业需求的变化。

1.3.3 最近发展

三层 C/S 结构中的“组件服务器”是一个范围很广的概念，也是近些年来很热门的一个研究课题。许多大型软件提供商都在致力于这方面组件服务器的开发，比如 Microsoft 公司的 MTS 组件（Microsoft Transaction Server）等等。目前的组件服务器一般都分为若干个模块，每一模块实现系统的一部分功能，然后需要将各个模块有机的结合在一起，达到协同运作的目的。在具体实现上，一般需要采用软件 Agent 技术和分布式组件技术，在各个模块中设计相应的功能 Agent，然后由一个管理 Agent 来协同诸多功能 Agent 的运作。正是因为组件服务器结构的复杂性，近年来又出现了所谓 N 层 C/S 的体系结构，不过从本质上来说无非是将组件服务器的功能进一步细分，只能说是三层 C/S 结构的一种细化。

二、深圳天音通信公司购销存系统

2.1 系统背景

深圳天音通信公司主要经营 Motorola 手机及其零配件，经过短短几年的发展，目前已经成为一个营销网络遍布全国的大型公司。随着这种强大的营销平台的建立和不断完善扩大，传统的信息交换方式（传统的信息交换主要有传真、电话和 email 等等）已经远远无法满足公司这种营销平台的需要。前端的营销信息（主要是进销存信息）对公司的营销决策起着至关重要的作用，而信息反馈的快慢又是公司能否成功保住原有市场和开拓新市场的一个极重要的因素。与此同时，公司的物流系统、资金流系统两条生命线的建立和完善工作也越来越紧迫。

为了克服企业现有各个环节中的不利因素，缩短购销存数据的周转时间，充分提高企业运作效率，降低销售风险，公司决定在企业内部建立一个基于三层 C/S 结构的购、销、存管理系统，力求充分利用现有公司业务流程中的有利因素，达到提高物流效率、节约物流成本的目的。

系统建设的目标如下：

- 改善现有购、销、存系统的管理模式，实现公司内部信息高速传递和高效共享。
- 建立企业内部 WEB 信息站点，为公司顺利实现电子商务建立系统基础环境。
- 建立具有高可靠性的电子商务交换平台、业务处理平台和信息数据存储平台。
- 建立企业内部决策支持系统，为企业内部决策层提供科学、可靠、实用的企业决策息，更好地为企业高速发展提供服务。
- 改变现有单一的产品销售模式，实现集产品网络行销、销后服务、客户跟踪和市场开拓于一体的综合管理模式。

2.1.1 公司营销网络

根据深圳天音通信公司的具体运作模式，公司的营销网络框图如下图所示：

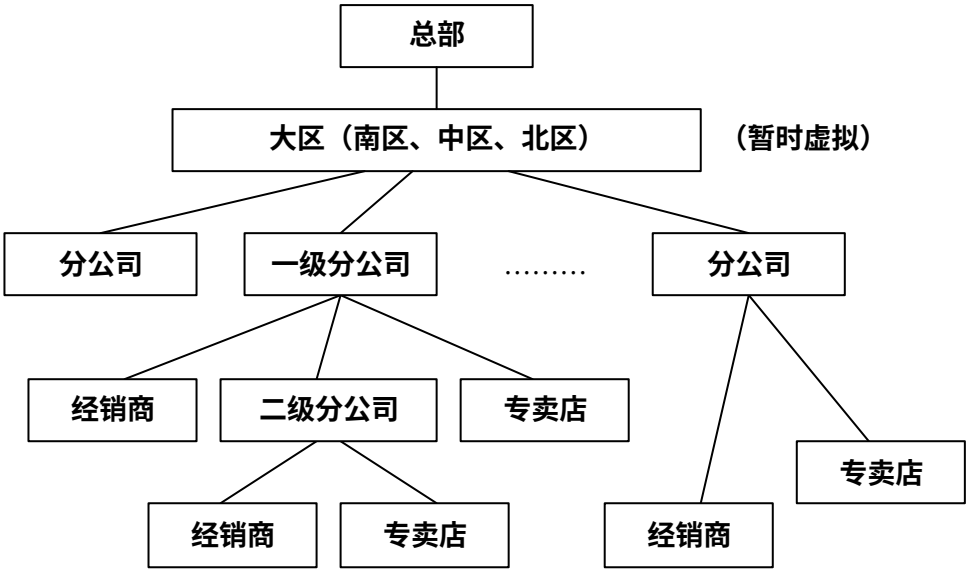

图 2-1　公司营销网络

从图中可以看出，公司的主要营销管理主体分为三级，即总部、分公司以及专卖店。

- 公司总部控制三个大区，每个大区负责管理其下的一些分公司。不过它们暂时还只是一个虚的机构，今后它的作用会逐步加强。
- 大区下设分公司，不过这些分公司不是彼此相互独立的，有些分公司还受其他分公司的管理，比如深圳分公司就可以管理东莞、惠州等几个分公司。我们姑且称被管理者为二级分公司或经营部，称管理者为一级分公司。
- 所有的分公司都可以管理自己的经销商和专卖店，包括一级分公司和二级分公司。

2.1.2　系统流程

整个系统的流程图如下：

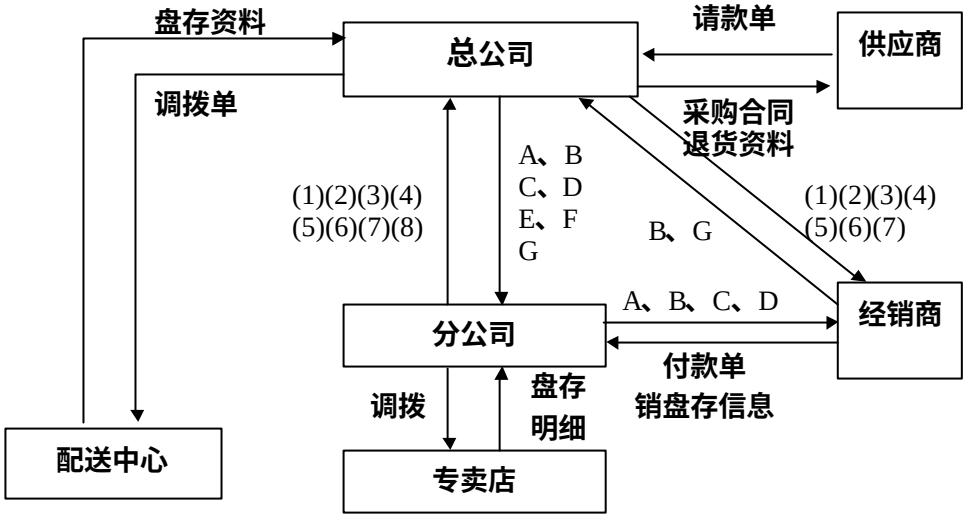

图 2-2　系统流程图

图中有关项的详细说明如下：

分公司 ——→ 总公司

（1）市场价格信息反馈；
（2）销售（明细）；
（3）库存（收到货、调出货等）非合格品；
（4）往来款；
（5）价格保护，现金折扣、折让；
（6）各分公司经销商的销售、库存；
（7）串号信息（收货、发货）；
（8）经销商资料（资信调查表）；

总公司 ——→ 分公司

A．销售指导价、审批价；
B．审批的返利；
C．新产品信息；
D．价格通知；
E．调拨指令；
F．货运信息；
G．销售计划;

根据系统业务流程，可以发现系统在销售层次上具有如下特点：

- 总部不直接负责销售，但是所有有关经销商、供应商以及配送中心的信息都由总部进行管理，总部还负责产品信息的确定以及价格指导。
- 分公司主要负责产品的销售。除此之外，它还必须管理子分公司以及专卖店，指导它们进行销售。专卖店的销售计划由分公司控制。
- 专卖店是最底层的销售主体，它只负责按照分公司的销售计划进行销售，并及时反映销售情况。专卖店可以采用 POS 机等形式进行销售。

2.2 系统设计

2.2.1 系统开发构架

在本文的第一部分，我们已经谈到在目前的企业级大型应用软件的开发中客户机/服务器结构所起的重要作用。根据公司的实际情况，两层 C/S 结构已经明显不适合系统的实际需要，因此我们决定在本系统中采用三层 C/S 结构（C/S/S- Client/Service/Server）和 B/S（Browser/Server）相结合的客户服务模式。对于重要用户，可以采用三层 C/S 结构的方式操作系统，这样大量数据计算和数据处理可以集中在中间件部件中，由它来统一管理数据库连接、数据接收、数据同步以及事务处理、线程调度等工作，最大限度地发挥网络的性能，避免客户机和服务器端的各种冲突。对于一些查询型的网络用户则可采用 B/S 的方式进行系统浏览，避免进行大量的库操作，提高效率。根据这种方案，我们将客户机/服务器系统中各种各样的部件划分成三“层”服务，它们共同组成一个应用程序。这三层服务包括：

- 客户端服务程序
- 业务服务和其它“中间层”服务程序
- 数据服务（数据库）

根据天音公司的具体情况，我们决定充分采用 Internet 技术来建设该公司的企业管理信息网络系统。先建设总公司、天音通信公司本部和仓库之间的 Intranet 网络系统，建立公司内部的信息发布 Web 站点，实现公司本部的信息充分共享，增强管理的透明性、实时性，提高公司的竞争力；接着再利用公司的 Web 服务器向公司的下属公司、用户提供服务，公司内部的所有分公司、办事处和各个分销商都可以通过当地的 Internet 通道直接向该公司提取各种信息。

我们力图建立如图所示的连接方式：

图 2-3　系统连接方式

对于上图，我们作如下说明：

- Internet 支撑平台包括整个系统运行平台，网络连接设备及工作站以及 Web 服务器、邮件服务器、DNS 服务器以及各种协议；
- 数据库则是指 SQL Server 7.0 数据库，Web 与数据库的中间接口是 Web 服务器与数据库服务器互连的技术关键。

2.2.2　系统信息流构架

从上面的系统背景中可以看出，由于公司管理依照总公司——分公司——（办事处、专营店、分销商）的三层管理模式，所以系统可分为总公司——分公司——（办事处、专营店、分销商）的三层模式。公司内部管理信息网络系统除要支持总公司本部的办公、管理和各个业务系统的正常运作外，还要收集所有下属分公司（办事处、专营店）的各种信息。

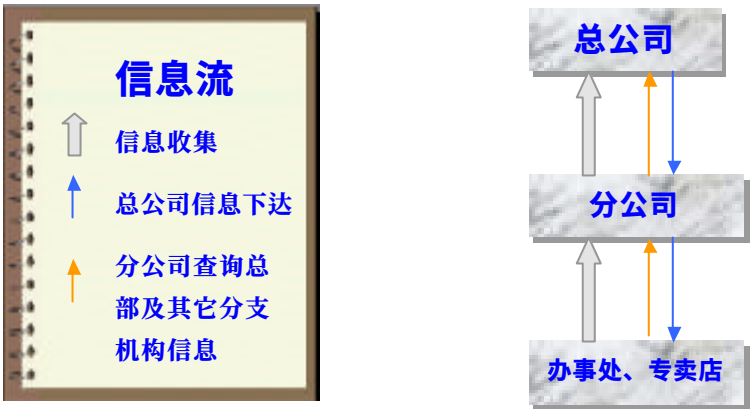

图 2-4　系统信息流

2.2.3　系统模块设计

根据公司管理的这种结构，我们将整个系统分为三个主要模块：服务器端、胖客户端以及瘦客户端。

（1）服务器端：对应总公司一级。

主要是接受胖客户端上传来的信息，做汇总以后存入后台数据库，供客户端查询及下载。服务器端应放在总部，并设有适当的安全机制。

（2）胖客户端：对应分公司一级。

主要是放在分公司，兼起一个客户端和一个服务器端的功能。相对于总公司的服务器端，它相当于一个客户端，负责将最新的销售、库存等信息上传给总公司，并下载相应的产品信息以指导销售；相对于瘦客户端，它则相当于一个小的服务器端，负责接受瘦客户端上传来的信息，作适当汇总再进一步上传。

（3）瘦客户端：对应专卖店一级。

主要设在专卖店，处理的信息量不大，主要包括当天专卖店的销售情况以及库存情况。每天要将最新数据上传给胖客户端，并下载新的销售计划。

我们将系统分为三个模块是基于以下几个原因：

- 配合公司物流。公司管理体制要求三个单独的模块设计。
- 将客户端的功能一分为二，对分公司和专卖店量体裁衣。由于专卖店不需要过多的管理功能，我们可以将瘦客户端设计的十分简单，只要实现相应功能即可。
- 必须要让分公司来管其下属的专卖店。我们不可能让所有的专卖店都将每天的销售数据直接上传到总公司，因为那样会使总部的服务器不堪重负。经过分公司处理过的数据再上传将会大大提高系统效率。
- 通用性更强。考虑到将来公司的发展，分公司有可能逐步变为一个管理机构，管理其下属的专卖店以及其他经销商。三模块的结构有利于系统的进一步完善。
- 分公司与专卖店之间类似 B2B 的关系，这为今后公司发展与经销商的电子商务系统打下了良好的基础。

进行如上的模块划分之后，系统结构更清晰了，也更符合公司的实际运营情况。但是，复杂的模块设计必将带来复杂的库表设计以及很多问题。下面列出的是其中的一些：

- 库表结构复杂。在设计库表时，必须考虑到每一个库表对于每一个模块是否可以访问，是否可读可写等问题。
- 系统管理难度大。系统管理员必须对三个模块之间的关系有很好的把握，才有可能协调好彼此之间的关系。
- 对广域网的依赖大。模块间信息的互传效率直接依赖于网络的畅通与否和速度高低。
- 系统安全控制难度加大。必须在系统设计时很好的考虑用户权限问题。
- 传输的困难性。由于存在三个模块，就需要在瘦客户端和胖客户端之间以及胖客户端和服务器端之间互传信息，传输的高效性与正确性是保证整个系统正常运作的前提。

在所有上面这些问题中，传输问题以及安全性问题是比较重要的。下面几节，我就从我个人对这两个问题的理解开始，尝试设计一些解决办法。

三、传输组件设计与实现

3.1 传输组件概述

纵观整个深圳天音购销存系统，传输部分在其中起了决定性的作用。无论是胖客户端与

服务器端交互，还是瘦客户端与胖客户端交互，传输的方便、高效、安全都成为系统成败的关键。只要传输的某一个环节出了问题，整个系统都有可能崩溃。

利用三层 C/S 体系结构的优点，我们在这里设计了一个传输组件，全权负责客户端与服务器端的连接以及数据的上传下载问题。每当客户端需要与服务器端交互时，只要将传输命令交给传输组件，其他的一切事情都由传输组件来做。整个传输组件的细节对用户以及系统其他部分的设计者来说都是透明的。

考虑到这里有两个对等传输系统，我们需要使用这种传输组件两次，一次是在瘦客户端与胖客户端之间，另一次是在胖客户端与服务器端之间。整个系统的三层架构如图所示：

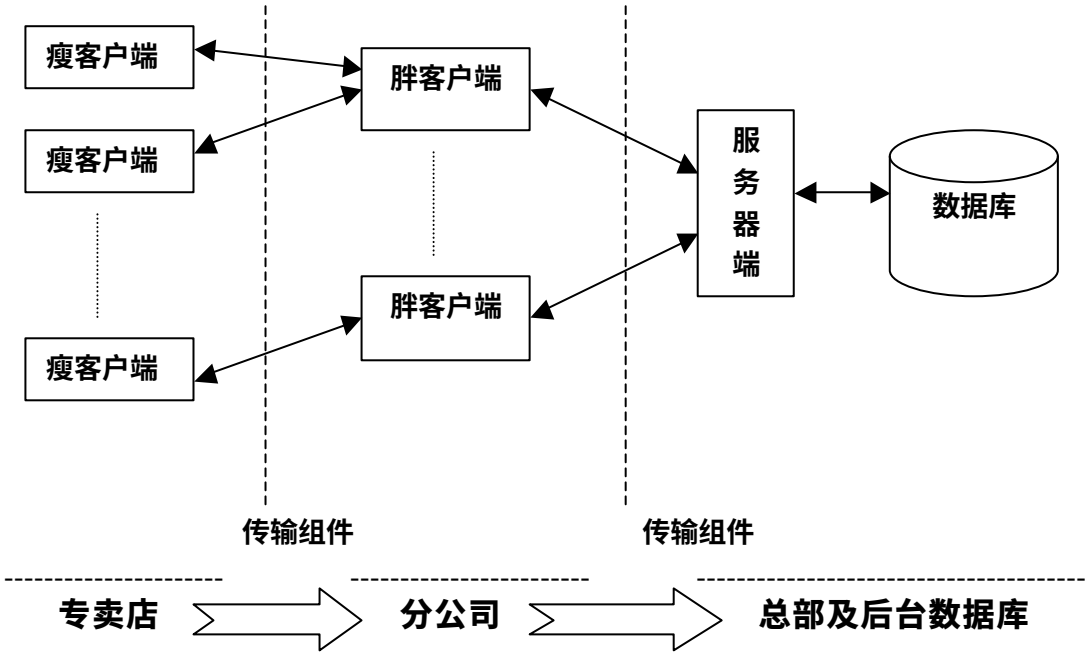

图 3-1　利用传输的系统三层架构

在具体设计传输组件时，我们充分使用了 Agent 组件技术。整个传输组件实际上是由若干个分布式 Agent 组件组成的，包括打包组件、解包组件、加解密组件、压缩解压组件、信息互传组件以及一个负责协调各 Agent 组件工作的管理 Agent 组件。客户端和服务器端都有这些组件在协同工作。每一个组件都会根据其他组件的状态以及环境的变化调整自身的状态，执行相应的动作，并通知其他组件采取后续的行为。正是因为采用了这种 Agent 组件技术，传输部分才能够正常工作，用户也能够在系统传输出问题的时候得到系统的提示。

3.2　组件工作流程

我们这里设计的传输组件在对等传输的客户端和服务器端各有一个传输 Agent 在运行。服务器端的传输 Agent 一直在运行，它时刻监听着客户端发来的信息。每当客户端有数据要上传时，双方就会按照如下的流程进行通讯：（为叙述方便，省去“传输 Agent”用语）

1. 客户端发出连接请求，并等待确认。
2. 服务器端监听到连接请求。
3. 如果服务器一切正常，服务器端向客户端发送连接确认信息，并做好接收数据的准备。
4. 客户端收到连接确认。这时，双方已经建立好了一条点对点的数据传输通道。开始

数据上传。

5. 客户端将事先准备好的数据传送至服务器端。
6. 服务器端接收客户端传送过来的数据。
7. 客户端在传送完数据之后发送结束帧，等待服务器端确认。
8. 服务器端对客户端传送过来的数据做校验，如果校验无误，发确认帧，并标志是否有新数据下载，转第 10 步；否则，发校验失败帧。
9. 客户端收到校验失败帧，转第 5 步重发数据。
10. 客户端收到确认帧，在库中作相应标记，数据上传结束。
11. 如果无新数据下载，传输结束，转第 20 步。
12. 有新数据下载，客户端做好接受下载数据的准备。开始数据下载。
13. 服务器端将事先准备好的数据下载至客户端。
14. 客户端接收服务器端下载的数据。
15. 已经下载全部信息后，服务器端发送结束帧，等待客户端确认。
16. 客户端校验下载数据，如果无误，发确认帧，转第 18 步；否则，发校验失败帧。
17. 服务器端收到校验失败帧，转第 13 步重发数据。
18. 服务器端收到确认帧，在库中作相应标记，并通知客户端结束传输。
19. 客户端收到通知。数据下载结束。
20. 双方断开连接，传输结束。

上面的整个过程可以用下面的图表简洁的表示：

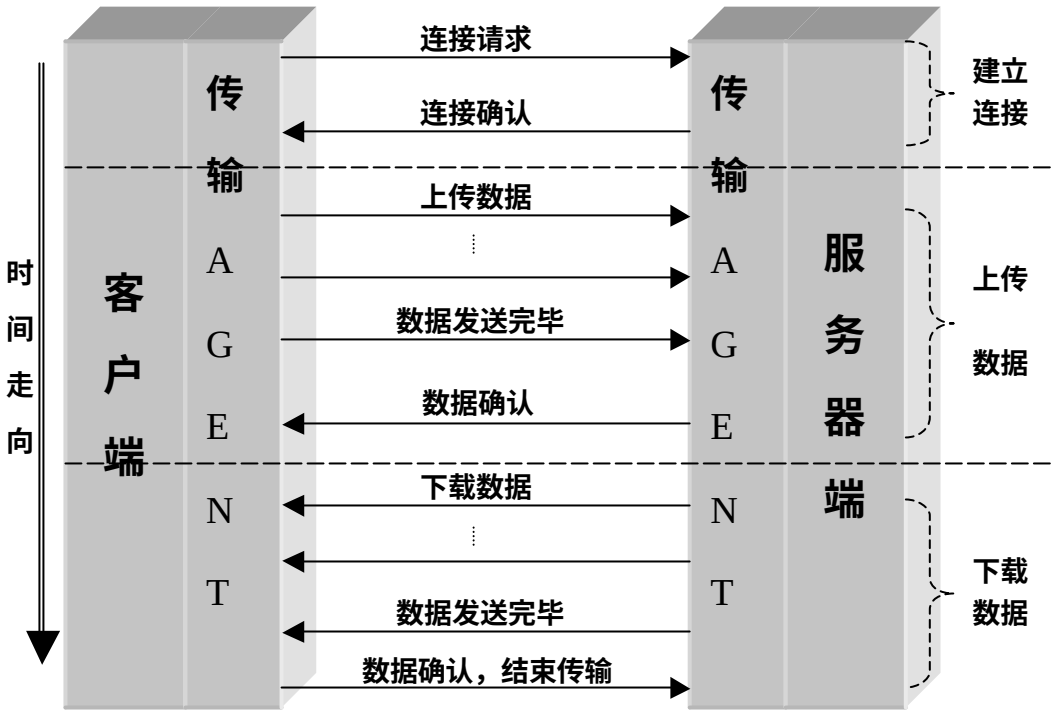

图 3-2　传输组件工作流程

3.3　传输预处理

数据在上传、下载之前是要经过预处理的。这种预处理一般要达到以下几个目的：

- 正确提取要传输的数据。这个问题比较复杂，我们下面会专门讲到。

- 精确定义传输文件格式，以便文件传输后能够被接收端正确识别。
- 如果传输安全对系统来说很重要，就要在传输前执行相应的安全策略，比如说对传输文件加密。接收端应该有相应的解密程序。
- 如果网络带宽有限，为了提高效率，可以考虑对传输文件进行适当压缩，不过必须保证在接收端能够对文件进行解压。

在我们设计传输组件中，上面几个问题都考虑到了。

首先，客户端与服务器端约定了一种数据打包格式，这也相当于是一种底层协议。数据上传之前，数据打包程序将所有要上传的数据从库中提取出来，按规定好的格式写入一个文本文件 up.txt。

然后，文件压缩程序将 up.txt 压缩为 up.zip。我们这里采用的是一种类似 Winzip 的压缩方法。

最后，文件加密程序将 up.zip 变为 up.zae，作为最后要传输的文件。加密算法我们这里采用的是 64 位的 DES 加密，密钥就嵌入程序中。

经过这三个步骤，既保证了上传数据的正确预处理，又考虑到了打包文件的安全性，还顾及到网络传输的带宽限制，可谓是考虑周全，尽量做到在保证传输质量的前提下增加安全系数，提高效率。

由于我们在设计传输组件时采用的是 Agent 组件技术，上面提到的文件压缩程序以及文件加密程序可以在需要时随时更换。我们可以单独设计一种压缩算法以及加密算法，替换我们现有的 Agent 组件。

经过这些预处理之后的数据文件在传到服务器端之后，还要经过一个逆过程，将最初的数据文本文件读出来，写入后台数据库。由于数据打包格式以及压缩、加密都是事先约定好的，只要传输中不出问题，数据的解包以及写库都应该正确无误。整个数据上传的处理过程见下面的框图。由于数据下载正好与数据上传相反，整个数据处理过程都是一样的。

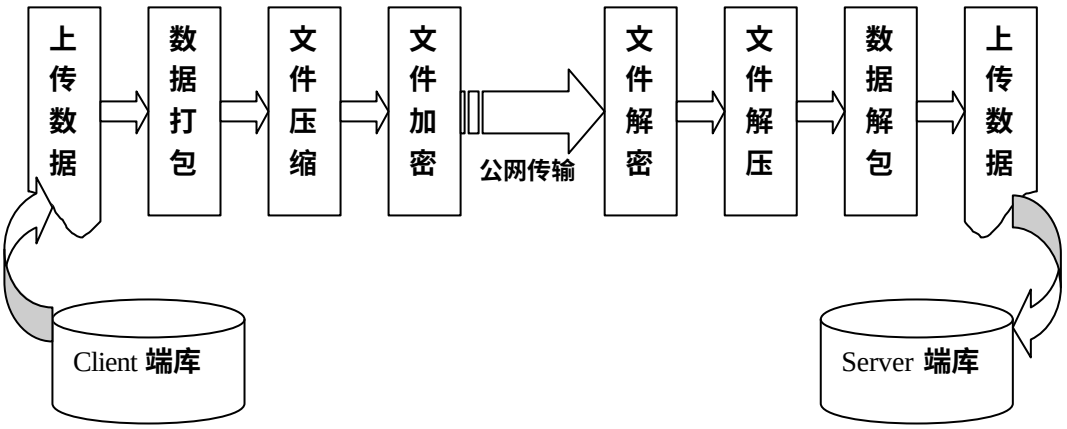

图 3-3　传输预处理

3.4　传输库表设计

前面已经提到了一个重要的问题，即如何在预处理阶段进行数据提取。为了标志出需要上传以及需要下载的数据，我们需要在库结构设计中充分考虑数据的组织问题。传输组件与数据库的结合是很紧密的，为了使传输组件能够正常的工作，我们需要在库表设计时充分考虑传输带来的问题，增加一些与传输有关的库结构设计。这一节我们就来探讨一下库表设计

的技巧。

3.4.1 服务器端传输表设计

根据深圳天音通讯系统的需求，系统的服务器端除了能够接受客户端上传过来的信息并作必要的处理之外，还必须维护许多有关物品、用户以及经销商信息的表单。为了做到这些信息在整个系统中的统一，必须由服务器端维护这些信息。这样，客户端为了得到相应的信息，就必须从服务器端下载相应的表单，存储在本地的数据库中。因此，服务器端的库表主要可以分为如下两个部分：

- **下载基础库** 这部分库表是要求客户端下载的，客户端必须维护与服务器端的数据统一，否则系统将无法正常运作。
- **非下载基础库** 这一部分是服务器端所独有的库表，主要作用是接受客户端上传的信息，并作相应的统计分析，为查询提供方便。

如此一来，服务器端的下载基础库就十分重要了。只要这些表单中有一个发生了变化，服务器就要通知客户端下载。只有这样，客户端才能够维持与服务器端的一致性，这也是系统最基本的要求。

要做到这一点，我们需要在系统的服务器端设计一些管理库，管理所有的应用库，并且与传输组件一起完成数据的传输工作。我们首先来看下面的“应用库管理库”：

字段名称	字段描述	字段类型	字段大小
KDM	库代码	Varchar	10
KMC	库名称	Varchar	30
KLX	库类型	Varchar	20

表 3-1 应用库管理库

这个表将服务器端所有表单的代码、名称以及类型作一记录，以方便管理。其中库类型包括前面提到的下载基础库和非下载基础库，还有专门为传输设计的中间库、传输库，介绍如下：

- **传输库** 这是为传输部分专门设计的表单，用于存放从客户端上传来的数据。系统要利用这些传输库中的数据来进行分析和统计，并将分析结果存入其他基础库中。
- **中间库** 多个客户端同时上传数据时，如果都往传输库里写，做插入操作，效率很低。我们可以设计一些中间库表，客户端上传时直接将数据添加到中间库的末尾。等到所有客户端都上传完毕之后，系统再利用存储过程对中间表的数据进行统计，将结果写入传输库里。处理完之后，应将中间库清空。

为了更好的描述哪些数据需要下载以及下载到何处，我们在服务器端建立了两个库表，描述如下：

字段名称	字段描述	字段类型	字段大小
SDKDM	上端库代码	Varchar	20
XDKDM	下端库代码	Varchar	20
KMC	库名称	Varchar	30
KLX	库类型	Char	1
QKLX	取库类型	Tinyint	1

表 3-2 下载库名库

字段名称	字段描述	字段类型	字段大小
SDKDM	上端库代码	Varchar	20
SDZDDM	上端字段代码	Varchar	20
XDZDDM	下端字段代码	Varchar	20
ZDCD	字段长度	Tinyint	1
ZDLX	字段类型	Char	1
SFGJZD	是否关键字段	Char	1

表 3-3　下载库字段名库

“下载库名库”描述了下载数据时服务器端和客户端的库表对应，其中“库类型”字段有添加库、修改库、添加和修改库三种选择，表示对库表的三种操作。“下载库字段名库”是在“下载库名库”的基础上进一步定义了下载时字段的对应关系，以及字段的一些基本信息。在数据传输前的打包操作中，这两个表中的信息是最关键的。

有了这两个表，服务器端如果要下载数据，就需要先到“下载库名库”中寻找客户端对应的库表，然后再到“下载库字段名库”中找寻相应的字段。这样服务器端就知道数据下载以后应该放在哪里了。通过对这两个表的定义，管理员可以轻松的管理数据下载操作，并可随时做出调整。

与下载维护表类似，数据上传也有相应的维护表，我们这里就略去了。在这里需要注意的是，“上传库名库”以及“上传库字段名库”是下载基础库，需要客户端随时更新。这也就是说，客户端要上传哪些信息不是由客户端自己定义的，而是由服务器端预先定义好的。这样可以保证上传信息的一致性，避免混乱局面。

涉及传输的另外一个很重要的库是“用户信息库库”，我们将它描述如下：

字段名称	字段描述	字段类型	字段大小
JGBH	机构编号	Integer	4
JGDM	机构代码	Varchar	10
CCLJ	存储路径	Varchar	200
is_have_new_data_up	是否有新数据上传	Char	1
is_have_new_data_down	是否有新数据下载	Char	1
is_successful_down	是否成功下载	Char	1
Successful_up_check	成功上传确认	Char	1
Successful_down_check	成功下载确认	Char	1
new_data_time	新数据时间	Datatime	8
down_data_time	最后下载数据时间	Datatime	8
JBCS	解包次数	Tinyint	1

表 3-4　用户信息库库

这个表的字段内容与传输组件的细节密切相关：

- **存储路径**　表示上传、下载文件存放在服务器上的绝对路径，便于查找。
- **解包次数**　表示数据上传的次数，如果数据上传之后发现有误，则需要重传，而且这个字段的值加一。我们可以人为地规定解包次数不能超过三次，超过三次自动断线，提示客户端重新连接。
- **是否有新数据上传、是否有新数据下载、是否成功下载**　传输前以及数据传输之后需要填入的字段，作为信息标志位。

- **成功上传确认、成功下载确认** 是在客户端发回确认信息之后填入的，只有这个字段回答“Y”的时候，数据才真正成功传输。

其他与传输有关的表单还有“错误日志库”、“错误类型库”、“数据处理时间库”等，分别记录传输时产生的错误信息以及数据打包解包处理的时间等。

为了及时告诉客户端哪些下载基础库有数据更新，我们必须在所有的下载基础库中都定义一个“最近更新时间”字段，标志当前记录最后的更新时间。在数据下载之前，系统将对比上次传输时间和最近更新时间，如果最近有更新，就要将数据打包下载。否则，可以不必下载客户端已有的信息。经过这样处理，每次客户端连接服务器端时，系统可以只将该客户端需要更新的数据下载到客户端，尽可能的减少冗余，提高传输效率。

另外，考虑到服务器端有许多库表需要下载，如果每次客户端建立连接后都直接到服务器端去寻找哪些表最近更新过，势必会降低效率。为了解决这个问题，我们在服务器端建立了一个“系统更新库”，记录最近有哪些下载基础库做了更新。

字段名称	字段描述	字段类型	字段大小
KDM	库代码	Varchar	10
ZJGXSJ	最近更新时间	Datetime	8

表 3-5 系统更新库

这样当有下载基础库最近作了更新时，系统就会将相应库代码以及最近更新时间记录在此表中。在下载数据打包之前，系统先去读此表，然后根据“库代码”字段的值再去相应的下载基础库中寻找更新数据。这样系统的效率就会大幅度提高。

3.4.2 客户端传输表设计

对应于服务器端，客户端也有与传输组件相对应的表单，我们在这里简单介绍一下。

首先，为了确定哪些数据需要上传，客户端需要将服务器端的“上传库名库”以及“上传库字段名库”下载到本地。每天在数据上传之前，先要将上传数据放入各类上传表中，然后在数据打包时对应两个上传库描述表整理数据。这里的工作与服务器端下载数据时的操作历程一样。

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
SKMC		上端库名称	Varchar	20	
CKMC		下端库名称	Varchar	20	
KMS		库描述	Varchar	30	Y
KLX		库类型	Char	1	
CZDM		操作代码	tinyint	1	

表 3-6 上传库名库

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
CKMC		下端库名称	Varchar	20	
SZDDM		上端库字段名	Varchar	20	
CZDDM		下端库字段名	Varchar	20	
ZDCD		字段长度	Tinyint	1	

ZDLX		字段类型	Char	1	
SFGJZD		是否关键字段	Char	1	

表 3-7　上传库字段名库

对应于服务器端的“用户信息库库”，在客户端我们设计了“传输信息库”如下：

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
JGDM		机构代码	Integer	4	Y
JGMC		机构名称	Varchar	30	Y
SJJGMC		上级机构名称	Varchar	30	Y
CCLJ		存储路径	Varchar	200	Y
Is_have_new_data_up		是否有新数据上传	Varchar	1	Y
Is_successful_up		是否成功上传	char	1	Y
Is_have_new_data_down		是否有新数据下载	char	1	Y
Successful_up_check		成功上传确认	char	1	Y
Successful_down_check		成功下载确认	char	1	Y
JGKL		机构口令	Varchar	10	Y
Sql_add_new		服务器新地 IP 地址	Varchar	15	Y
Sql_add_now		服务器当前 IP 地址	Varchar	15	Y
SXJZSJ		生效截止时间	Datetime		Y
JBCS		解包次数	Tinyint		Y

表 3-8　传输信息库

这里面大部分字段的含义与“用户信息库库”中的相同，有一些新的字段解释如下：

- 机构口令　在客户端上传数据时的口令。不过这个口令一般不需要客户端人员直接输入，而是由系统直接附着在上传数据中。服务器端将验证口令是否正确。
- 服务器新的 IP 地址　标志更新后的服务器地址。
- 服务器当前 IP 地址　表示当前服务器地址。
- 生效截止时间　表示服务器新地址生效的日期。

客户端在上传数据之前总是要查一下当前时间是否在生效截止时间之后，如果是，则采用新的服务器地址；否则，采用当前的服务器地址。

3.5　传输组件应用

采用前面的设计思想，我们利用 Visual C++ 6.0 作为工具，设计出来了一个功能比较完备的传输组件。我们实验室的一个同学将该传输组件应用于 TCL 公司的 JXC 系统中，取得了良好的运行效果。

受深圳天音购、销、存系统开发进度所限，目前该传输组件还没有真正付诸实用。我们只是在瘦客户端和胖客户端之间传输数据时作了一个试验，效果不错。但与 TCL 公司的系统不同，深圳天音购销存系统有三层结构，需要两次应用该传输组件，可能在很多地方会出问题。我们会在今后的工作中进一步测试该组件，力争设计出一个较通用的、能够直接加载在一般的企业级系统之上的传输组件来。

四、基于菜单的权限管理

4.1 权限管理综述

纵观目前的企业级软件，如 MIS 系统、企业内部网管理系统等等，系统权限管理一直是一个比较头疼的问题。企业级管理软件的使用者类型多种多样，可以是系统管理员，可以是一般的数据录入员，还可以是查询系统的查询员。每一种类型的使用者都应该有适当的权限，这样既可以保证他们能够完成他们的各项工作，也可以防止他们查看到他们不应该知道的信息。这对于企业的正常运作以及安全保密工作都具有十分重要的意义。

基于系统设计的不同，系统权限管理可以有很多种，例如：如果数据库中的表单相对于不同用户来说权限不同，就可以设计基于表单的用户权限管理；如果不同的文件对于区分用户权限有明显的作用，就可以设计基于文件的用户权限管理；如果各个功能模块相对独立，而且也要定义各个模块对于不同用户的权限，就可以将功能模块定义为主菜单，设计基于菜单的权限管理。下面我就以深圳天音通信公司的内部网管理系统为例来谈谈我设计的基于菜单的权限管理。

对于深圳天音通信公司的购销存管理系统，权限管理问题十分重要。对于该系统的三个模块，胖客户端功能最强，结构最为复杂，因此我们应该先考虑胖客户端的权限管理问题。这个模块的权限管理问题解决了，其他模块就水到渠成了。根据天音公司的需求，胖客户端的用户主要可以分为三类，即我们前面提到的系统管理员级、数据录入员级以及查询员级。在这三级之下还可以继续细分，我们为了简单考虑姑且只考虑这样三类用户，其他类型的用户我们可以在系统中逐步添加。

根据需求，深圳天音公司购销存管理系统的胖客户端主要由以下几个功能模块组成：系统管理，收、提、退货管理，库存管理，数据采集，往来帐管理，对帐，结帐以及报表与统计分析。如果按菜单项写出，主要的菜单项如下：

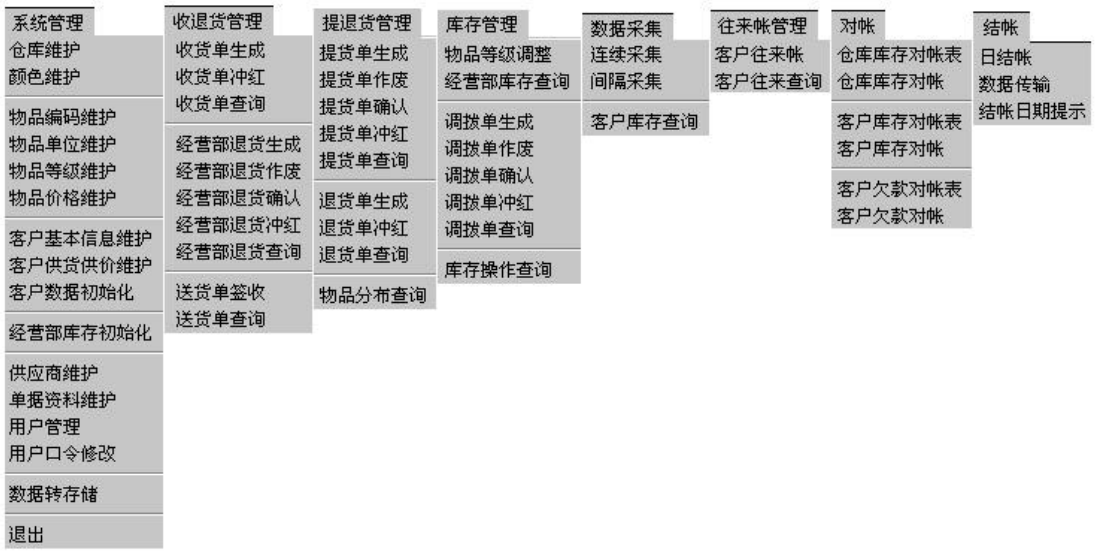

图 4-1 系统菜单

从上面的菜单项可以看出，这些功能模块彼此之间相对独立。因此对于该系统的权限管理，一个比较好的设计方案是：将这些功能模块设计成系统的主要菜单，然后对菜单进行编

号，在菜单编号与用户类型之间建立一个对应关系。这样只要给每个用户指定一个用户类型，就可以标志出当前用户的权限了。

4.2 库表设计

首先明确一个问题：对于一个企业级的客户端系统，使用的人员比较多，而且其中大部分人员可以归类处理，就像我们前面提到的系统管理员、数 据录入员以及查询员等等。因此，我们没有必要给每一个用户定义他的权限，我们只需要对某一类用户定义他们的权限，然后在对某一个用户指明他属于哪一类的即可。

根据这一点，我们就需要在“用户管理表”中添加一项标志“用户权限类型”的字段，用以定义每一用户的权限。另外，考虑到计算机中存储和查询的方便，以及添加用户权限类型的需要，这个字段我们用“用户权限类型编码”来表示，然后再建立一个从“用户权限类型编码”到“用户权限类型名称”的映射表，这样就可以尽可能的减少冗余，提高库结构设计的合理性。因此我们设计用户管理表以及用户权限类型表如下：

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
YHDM	Y	用户代码	Varchar	10	
YHMC		用户名称	Varchar	20	Y
YHZW		用户职务	Varchar	20	Y
YHKL		用户口令	Varchar	20	
YHQXLXBM		用户权限类型编码	Char	2	

表 4-1 用户管理表

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
YHQXLXBM	Y	用户权限类型编码	Char	2	
YHQXLXMC		用户权限类型名称	Varchar	20	
YHQXLXMS		用户权限类型描述	Varchar	100	Y

表 4-2 用户权限类型表

在这两个表中，用户管理表记录了有关用户的一些基本信息，其中还包括了用户口令。它以用户代码为关键字。当添加新用户以及修改用户信息时，将操作此表。用户也有权更改自己的口令，但无权更改自己的用户权限类型。用户权限类型表除了给出用户权限类型编码到用户权限类型名称的对应，还提供一个用户权限类型描述的字段，给出对该种用户权限类型的描述。这一字段是可空的。用户权限类型编码是内部编码，对外部用户是不可见的，在这里可以采用系统自动生成的方法，也可以预先给出内部定义。

在定义了用户的权限类型信息之后，就可以来具体定义用户的权限了。为此，我们先需要一个“菜单维护表”，刻画每个菜单名称的对应编码。这里的想法跟前面提到的用户权限类型编码一样。然后，我们就可以建立一个“菜单操作权限表”，给出每一种用户权限类型对于每一个菜单项的操作权限。这两个表的具体描述如下：

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
CDBM	Y	菜单编码	Int	4	
CDMC		菜单名称	Varchar	30	

表 4-3 菜单维护表

字段名称	关键字	字段描述	字段类型	字段长度	是否可空
YHQXLXBM	Y	用户权限类型编码	Char	2	
CDBM	Y	菜单编码	Int	4	
CZQX		操作权限	Char	1	

表 4-4　菜单操作权限表

一旦系统的主菜单确定下来，就可以将菜单维护表初始化，这部分工作是系统正常运作之前所必需的。我们可以手工在表中加入初始化信息，也可以编制程序让机器自动完成。菜单操作权限表也需要初始化，我们至少应该给出几个缺省用户类型对应的菜单操作权限。在这个表中，用户权限类型编码和菜单编码共同作为关键字，这样才能对菜单操作的权限做出完整的描述。“操作权限”字段只是长度为 1 的 Char 类型，内容为“Y”或者“N”，标志当前用户对当前的菜单项是否能够操作。实际上，“用户权限类型表”与“菜单维护表”是一个多对多的关系，描述它们之间的这种多对多关系的正是“菜单操作权限表”。

有了这些库表描述，我们就可以来设计一个完整的基于菜单的用户权限管理了。

4.3　程序结构设计

考虑到深圳天音通讯系统的需求以及库结构设计，我们采用 SQL Server 7.0 作为数据库平台，采用 Visual Basic 6.0 作为系统开发平台，并采用 Microsoft 最新推荐的 ADO（ActiveX Data Object）技术访问数据库。这些系统实现平台的选择一方面是考虑到平台之间的兼容性和完整性，另一方面也考虑到系统的通用性以及可扩展性。

根据前面的用户权限库表设计，当系统启动时，一般要按照下面的流程来操作：

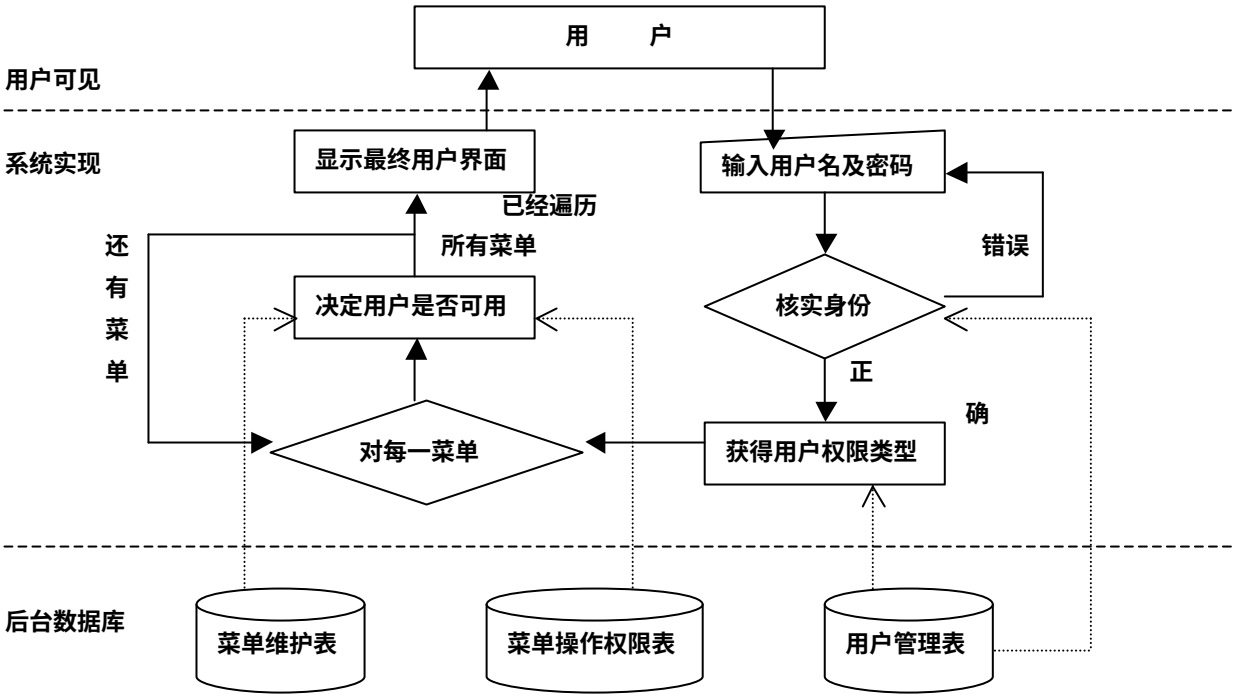

图 4-2　权限操作流程

系统启动时，要显示登录对话框，提示用户输入用户名以及密码。如果当前用户的记录在“用户管理表中”存在，而且输入的密码也准确无误，登录过程结束。这时系统读取“用户管理表”中当前用户的“用户权限类型编码”字段的内容，这个编码就标志了当前用户的权限。然后，系统根据这个编码查询“菜单操作权限表”，这样对于每一个菜单项，系统就可以得到当前用户的操作权限。在这以后，系统读取“菜单维护表”，将当前用户的所有操作权限从“菜单编码”对应到“菜单名称”中去。最后，根据这种对应关系以及系统实际建立的菜单条，调整每个菜单的可用性，然后将主界面显示给当前用户。

根据深圳天音公司对于用户权限管理的要求，系统管理员具有操作所有菜单的权限，并且可以对系统现有的用户以及用户类型进行管理，尤其是可以管理所有用户的权限。其他用户对于系统管理这一部分不允许操作“用户管理”这一命令，但可以进行“口令更改”等等其他操作。查询员可能只能操作所有主菜单中的各种查询功能。

系统实现后经反复测试，权限管理基本上令人满意。只要最初系统中有一个用户具有系统管理员的权限，所有的系统管理工作都可以由他来完成，包括对用户信息的添加、删除、修改，对用户口令的替换，对用户权限类型的修改，对菜单操作权限的修改等等。只要系统管理员级的用户不恶意操纵系统，整个系统就是安全的。

4.4 用户界面及其具体实现

在这一部分我主要介绍一下目前深圳天音购销存系统胖客户端有关权限管理以及用户管理的主要界面以及关键代码的实现细节。

用户最初登录时，显示登录界面。如果用户输入的用户名以及密码都正确无误，系统将提示正在登录，请用户稍等，如图 4-3 所示。

图 4-3　正确登录界面

由于每次用户登录时，系统都要检查一遍菜单操作权限表，然后再与实际菜单对应，因此会花费很长的时间。在目前的测试环境下（COMPAQ WorkStation 200M，RAM 64M），大

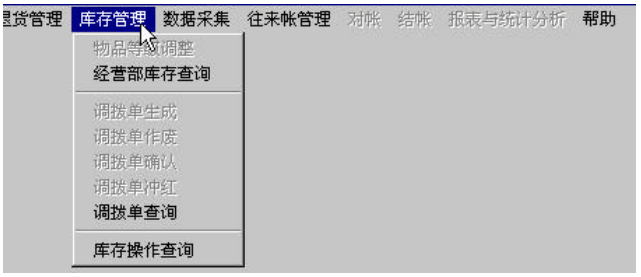

图 4-4　查询员登录后的部分菜单

约需要 5 秒钟。我争取在今后的工作中逐步缩短系统登录时间。

在这以后，系统就会进入主界面。所有菜单项的 Enabled 属性都是根据当前用户的权限来设定的，用户没有权限的菜单项就会变为灰色。如图 4-4 所示是查询员登录后系统显示的部分菜单。

所有用户都有权更改用户口令，但是出于系统安全考虑，用户都只能更改自己的用户口令。因此，更改口令时用户是不能选择用户名的。口令更改的对话框如图所示。虽然如此，系统管理员还是可以更改所有用户的口令，请参看下文。

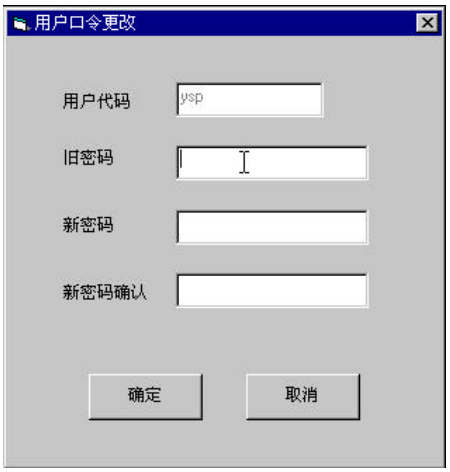

图 4-5　用户口令更改

下面我主要介绍一下“系统管理”主菜单中的“用户管理”一项。用户各种信息的增、删、改以及用户权限管理的所有操作都在这一菜单中完成。当系统管理员点击“用户管理”子菜单时，可以得到如图 4-6 所示的对话框。

图 4-6　用户管理

左边列表框中显示的就是当前系统的所有用户的用户代码。系统管理员只要单击列表框中的任一用户，该用户的所有信息就会显示在右边的文本框中，不过所有信息都是只读的。如果管理员想修改用户信息，请点击“修改”按钮，将得到如图 4-7 所示的对话框。在这里，

管理员可以更改用户的所有信息，甚至包括用户的密码。通过在“用户权限类型”的组合框中进行选择，管理员可以随意指定所有用户的权限类型。

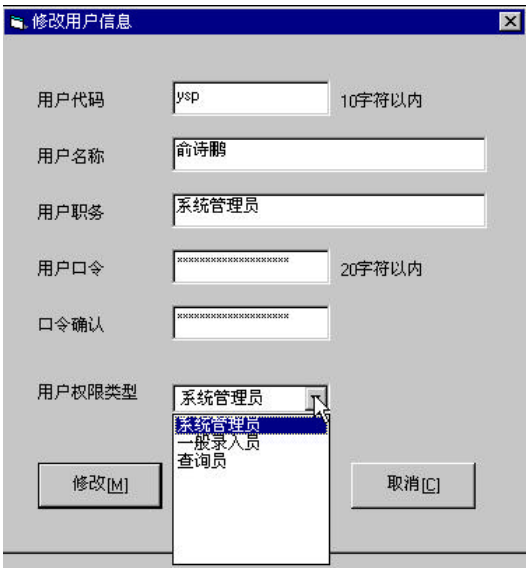

图 4-7　用户信息修改

在用户管理的对话框中点击“添加”按钮，可以添加一个新用户，弹出的界面与修改用户信息的界面类似。在这里管理员可以为新用户指定一个初始密码，也可以保持用户口令为空，让新用户在初次登录时自己修改密码。

管理员同样可以删除当前用户，只要点击“删除”按钮即可。这时系统会给出删除提示，确认后即可进行删除。

图 4-8　删除提示

如果系统管理员点击“菜单操作权限”按钮，就可以得到如图 4-9 所示的对话框，设置每一种用户类型的菜单操作权限。

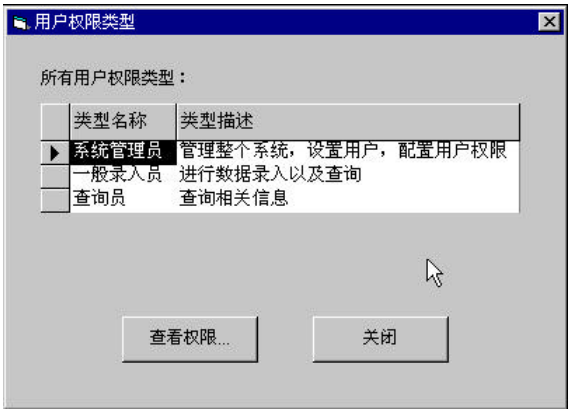

图 4-9　查看用户权限类型

这个对话框中显示了当前所有的用户权限类型名称及其类型描述。如果要查看每种类型的具体菜单权限，请选定网格中的一行（比如我们选择标志为“查询员”的那一行），然后点击“查看权限”按钮，系统会给出如图 4-10 所示的对话框。

图 4-10　查询员的菜单操作权限

对话框中上面的文本框标志当前查看的用户权限类型，由于已经选定，不能更改。下面两个列表框分别显示当前用户能进行以及不能进行的操作。这些都是根据当前的用户权限类型到“菜单操作权限表”中查找得来的数据。在这里管理员可以利用中间的四个按钮方便的进行权限调整，也可以直接用鼠标双击来更改权限。更改完毕后，点击“确定”按钮，可以将更改保存到菜单操作权限表中去。点击“取消”按钮可以随时取消所做的权限更改。管理员对用户类型的权限所做的修改将在下一次用户登录时生效。

4.5　不足之处与下一步研究方向

由于时间关系以及对权限管理的理解不够深入，我在这里设计的用户权限管理还存在许多不足之处，主要有以下几点：

- 登录时间过长。
- 暂时还没有考虑用户类型的添加以及删除。
- 表单过多，系统界面的设计不够合理。
- 没有考虑菜单的父子关系，使得定义菜单权限时操作复杂，效率低。
- 通用性低。整个权限管理强烈的依赖于原先菜单的设计，无法动态的根据菜单变化做出反应，更无法直接移植到类似的系统中。

用户权限管理是非常复杂的，它涉及到系统方方面面的问题。我在这里只是对这个具体的系统设计了一个很简单的权限管理，还有很多问题没有考虑到。而且整个购销存系统还没有经过公司的实际运作，可能还有许多权限管理问题没有在目前的测试中暴露出来。我争取在今后的研究生阶段继续完善该系统，尽量朝着下面几个目标努力：

- ✧ 对每一个菜单项建立索引，力争在库结构设计上更注重菜单项的维护问题，提高系统对菜单项的检索速度。
- ✧ 从通用性上考虑，尽量将菜单维护以及用户维护从个别系统中抽象出来，力争建立

一个通用性强，可移植性好的用户权限管理组件，使之能够用于任何具有类似功能的系统。

- ✧ 在菜单权限管理的基础上，综合考虑其他类型的权限管理，并将它们综合在一个大的权限管理范畴中。这样，对于一个大型企业，可以自己挑选所采用的权限管理类型，也可以综合运用多种权限管理方法，达到全面、完善的管理用户权限的目的。

致 谢

本文从最初的选题一直到最后的成文，都是在我的导师林作铨教授的精心指导和热情鼓励下完成的，在此我首先对林教授表示衷心的感谢。他以渊博的知识和深厚的学术造诣指导我进行相关课题的研究，使我明确了钻研与突破的方向。为了使我们进一步了解整个系统需求，林教授还亲自带我们到公司去，与公司的管理人员和销售人员进行讨论，研究系统的详细设计方案。在我整个毕业设计期间，我向林教授学到了很多东西，这其中不仅包括学术上的成就，还包括教学上严肃的态度以及生活上严谨的作风。衷心感谢林教授这一学期对我的指导。

这一学期在实验室里，我得到了许多老师和研究生同学的热情帮助，他们给我提供了良好的环境和氛围，使我能够充分发挥自己的水平，培养自己的能力。在这里我也向他们表示衷心的感谢。

在深圳的那段日子里，天音公司的管理人员和销售人员和我们一起探讨了系统的需求、详细设计以及其他方方面面，对于我们设计系统结构有很大的帮助，对他们表示衷心的感谢。

这一次我们一起做毕业设计的还有几名同学，我们经常在一起讨论，一起设计库结构，互相学习，以求共同进步。感谢他们这一学期以来对我的帮助。

参考文献

[1] 陆汝钤，人工智能（上、下册），科学出版社，1996

[2] Lyn Robison 著，黄惠菊等 译，轻松掌握用 Visual C++ 6.0 对数据库编程，电子工业出版社，1999.6。

[3] D.Solomon，R.Rankins 著，熊桂喜等 译，Microsoft SQL Server 6.5 开发指南，清华大学出版社，1998.4。

[4] Roger Jennings 著，前导工作室 译，Visual Basic 6数据库开发人员指南，机械工业出版社，1999.9。

[5] David Bennett 等 著，徐军等 译，Visual C++ 5 开发人员指南，机械工业出版社，1998.9。