独 创 性 声 明

秉承学校严谨的作风和优良的科学道德，本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，不包含本人或他人已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示致谢。

申请学位论文与资料若有不实之处，本人承担一切相关责任

论文作者签名：吴林涛 2009年9月15日

保护知识产权声明

本人完全了解西安理工大学有关保护知识产权的规定，即：研究生在校攻读学位期间，论文工作的知识产权单位属西安理工大学。本人保证毕业离校后，发表论文或使用论文成果时署名单位仍然为西安理工大学。学校有权保留送交论文的复印件，允许论文被查阅或借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的学位论文在解密后应遵守此规定）

论文作者签名：吴林涛 导师签名：王永康 2009年9月15日

多媒体餐饮管理系统及应用研究

学科领域：电气工程　　答辩日期：
作者姓名：吴彬琦　　作者签字：吴彬琦
导师姓名：王新房　　导师签字：王新房

摘　要

本文探讨了以关系型数据库为基础的多媒体管理信息系统的基本实现方法。在论文中，主要探讨了如何利用关系型数据库实现多媒体数据库，本文也讨论了多媒体数据的读取和更新、数据库的优化和安全性设计等等。论文工作中实现了一个基本的多媒体管理系统——多媒体餐饮管理系统，在系统开发过程中，介绍了在客户端的多媒体数据的存取和播放等技术。系统的开发是基于 Windows NT 4.0 操作系统、TCP/IP 协议、MS SQL Server 7.0 数据库管理系统和 Borland Delphi 5.0 开发工具的。

关键词：多媒体 管理信息系统 关系数据库 优化

THE APPLICATION RESEARCH OF MULTIMEDIA MANAGEMENT INFORMATION SYSTEM

ABSTRACT

A basic realization method of a multimedia management information system (MMIS) is discussed in this emphasis. How to realize a multimedia database and how to read and update multimedia data are the main topics. The optimization and the security designation of a database are also the topics. In the emphasis working, a basic MMIS- Multimedia Restaurant Management Information System is realized. The technologies of multimedia data's reading and saving on the client site and of multimedia data's playing are introduced in the emphasis. The developed system is based on Windows NT 4.0, TCP/IP, MS SQL Server 7.0 and Borland Delphi 5.0.

Keyword: Multimedia MIS RDBMS Optimization

1 概 论

1.1 引 言

多媒体技术在我们的生活、生产和国防中得到日益广泛的发展和应用，如视频点播、电视会议、远程教育、地理信息系统等等。尤其是近几年互联网络的飞速发展，也给多媒体技术的发展和应用提供了广阔的前景，但同时也对多媒体技术提出了空前的应用需求和挑战。如何高效率的组织和加工多媒体信息，以及基于内容的检索、视频信息的内容表示和多媒体信息的建模等技术，已成为目前研究的一些热点问题。

本文以多媒体餐饮管理系统的开发为背景，主要探讨了一个基本的多媒体管理系统的实现方法和途径。

该系统以 Microsoft SQL Server 7.0 为数据库平台，采用 TCP/IP 网络协议进行通信，利用 Borland Delphi 5.0 开发客户端应用程序。由于多媒体信息通常是海量的信息体，要它对进行管理和维护，需要较高的性能要求和技术要求。本文的工作集中在基于关系型数据库的多媒体数据库的实现、多媒体数据的读取和更新、视频信息的播放以及数据库系统的性能优化等方面。

1.2 多媒体管理系统发展现状

当前，在开发多媒体信息管理系统时主要采用的是针对数据密集型应用的关系数据库。数据密集型应用(Data Intensive Application) 是计算机应用中最大的一个领域，管理信息系统、决策支撑系统、证券交易系统、银行信息系统、民航订票系统等都属于这一类。其主要特点是管理和维护大量的、共享的持久数据。

数据密集型应用也经历了从低级向高级的发展过程。最初采用文件系统的形式，后来也逐步发展为数据库系统。早期的数据库支持大量的结构统一的数据项，数据项是基本的单位，不能再包含结构。70 年代出现层次

结构和网状结构数据库，其结构是面向记录的，每个域中定义一个基本的属性值，它使得应用与数据本身的存储结构相互独立，并且具有查询语言、查询优化、约束条件的检测和可对底层存储系统进行控制等特性。70 年代后期出现了关系数据库(RDBMS)，由于其结构简单、使用方便，很快取代了层次和网状数据库，成为数据库市场上的主流产品。

关系数据库(RDBMS)是以关系模型为基础的数据库，关系模型是用关系来描述现实世界的，是描述实体以实体之间的联系。关系数据模型是实体与实体间的联系通过表格数据来实现，是一个二维表格。定义关系及属性的关系模式集合组成了一个关系型数据库模式。关系型数据库具有数据结构简单、数据独立性高、面向集合的存取方式及使用简单、易于控制等特点，而且有利于分布式数据库的实现，易实现分布数据的管理、操作和重组。但由于关系数据库不支持复杂对象的内在缺陷，对于多媒体数据的应用具有一定的局限性。

MS SQL Server 7.0 是一客户/服务器关系数据库管理系统。它具有性价比高、开发简单、安全性能好、支持 BLOB 字段以及性价比高等特点，因此本系统采用了 MS SQL Server 7.0 作为数据库平台。

1.3 多媒体信息管理系统分析

近年来，随着多媒体技术的突飞猛进的发展，以及多媒体信息的空前需求，对数据的管理方法又开始酝酿新的变革。当图像、声音、动态视频等多媒体信息引入计算机之后，可以表达的信息范围大大扩展，但又带来许多新的问题。在这种情况下，如何使用数据库系统来描述这些数据呢？另一方面，传统数据库可以在用户给出查询条件之后迅速地检索到正确的信息，但那是针对使用字符数值型数据的。现在，我们面临着这样的问题：如果基本数据不再是字符数值型，而是图像、声音，甚至是视频数据，那我们将怎样检索？如何表达多媒体信息的内容？我们应该如何组织这些数据呢？查询应该如何进行？这些都是我们不得不考虑的。

可喜的是，随着技术的发展，产生了许多可以对多媒体数据进行管理和使用的技术，例如面向对象数据库、基于多媒体内容检索技术、超媒体技术等等。一般认为，多媒体数据库不应该是对现有的数据库系统进行界面上的包装，使之看起来象一个多媒体数据库，而应该是从多媒体数据与信息的本身特征出发，才能找到相应的解决方法。

1.4 多媒体餐饮管理系统开发思路

本文主要研究了基于关系型数据库的多媒体管理信息系统的基本实现方法。它包括多媒体数据在关系型数据库中的读取和更新、多媒体数据在客户应用程序中的播放等技术。对于多媒体信息系统，往往具有海量的信息存储，需要较高的性能，因此还讨论了数据库的性能优化。论文中也讨论了数据库的安全性设计和数据复制等技术。

论文工作中还实现了一个基本的多媒体管理信息系统——多媒体餐饮管理系统。它包括系统的需求分析、总体设计和各功能模块的结构化设计，并利用 Delphi 5.0 实现了系统。该系统使用的数据库平台为 MS SQL Server 7.0，操作系统为 Windows NT 4.0，网络传输协议为 TCP/IP。

2 多媒体数据库及其实现

2.1 多媒体及多媒体数据库

2.1.1 多媒体

在计算机和通信领域，我们所指的信息的正文、图形、声音、图像、动画，都可以称为媒体。从计算机和通信设备处理信息的角度来看，我们可以将自然界和人类社会原始信息存在的形式——数据、文字、有声的语言、音响、绘画、动画、图像（静态的照片和动态的电影、电视和录像）等，归结为三种最基本的媒体：声、图、文。传统的计算机只能够处理单媒体——“文”，电视能够传播声、图、文集成信息，但它不是多媒体系统。通过电视，我们只能单向被动地接受信息，不能双向地、主动地处理信息，没有所谓的交互性。可视电话虽然有交互性，但我们仅仅能够听到声音，见到谈话人的形象，也不是多媒体。所谓多媒体，是指能够同时采集、处理、编辑、存储和展示两个或以上不同类型信息媒体的技术，这些信息媒体包括文字、声音、图形、图像、动画和活动影像等。

2.1.2 多媒体数据库及其特点

多媒体数据包括数值、字符串、文本、图形、图像、声音、视频等。对这些信息进行管理和应用的数据库就是多媒体数据库。多媒体数据具有以下几个鲜明的特征：

a. 信息量大

b. 数据的非格式化

由于多媒体数据本身的复杂和多样性，媒体大都以大二进制对象BLOB(Binary Large Object)的形式而存在；

c. 声音、视频等媒体具有时间敏感性；

d. 由于多媒体信息的集成性引起不同媒体之间关系复杂；

e. 交互性强。

2.1.3 多媒体数据库所面临的问题

在传统的数据库中引入多媒体数据和操作，是一个极大的挑战。传统的字符数值型的数据虽然可以对很多的信息进行管理，但由于多媒体数据的抽象特性，应用范围十分有限。为了构造出符合应用需要的多媒体数据库，我们必须解决以下几个方面，这些问题也是目前多媒体数据库技术所面临的前沿问题：

a. 数据库的组织和存储。媒体数据的数据量大，而且媒体之间的差异也极大，从而影响数据库的组织和存储方法。如动态视频压缩后每秒仍达上百 K 的数据量，而字符数值等数据可能仅有几个字节。只有组织好多媒体数据库中的数据，选择设计好合适的物理结构和逻辑结构，才能保证磁盘的充分利用和应用的快速存取；

b. 媒体种类的增多增加了数据处理的困难。每一种多媒体数据类型都要有自己的一组最基本的操作和功能、适当的数据结构以及存取方式、高性能的实现。不同媒体类型对应不同数据处理方法，这就要求多媒体数据库管理系统能够不断扩充新的媒体类型及其相应的操作方法。新增加的媒体类型对用户应该是透明的；

c. 数据库的多解查询问题。传统的数据库查询只处理精确的概念和查询。但在多媒体数据库中非精确匹配和相似性查询将占相当大的比重。在对多媒体字段进行查询时通常是一种模糊的非精确的匹配方式。媒体的复合、分散及其形象化的特点，注定要使数据库不再是只通过字符进行查询，而应该是通过媒体的语义进行查询。然而，我们却很难了解并且正确处理许多媒体的语义信息。这些基于内容的语义在有些媒体中是易于确定的（如字符、数值等），但对另一些媒体却不容易确定，甚至会因为应用的不同和观察者的不同而产生不同；

d. 用户接口的支持。多媒体数据库的用户接口不能用一个表格来描述，对于媒体的公共性质和每一种媒体的特殊性质，都要在用户的接口上、在查询的过程中加以体现。例如对媒体内容的描述、对空间的描述、以及对时间的描述等等。多媒体要求开发浏览、查找和表现多媒体数据库内容的

新方法，使得用户很方便地描述他的查询需求，并得到相应的数据。多媒体数据库的查询结果应不仅仅是传统的表格，而将是丰富的多媒体信息的表现，甚至是由计算机组合出来的结果；

e．处理长事务增多。传统的事务一般是短小精悍的，在多媒体数据库管理系统中也应该尽可能采取短事务。但有些场合，短事务不能满足需要，如从动态视频库中提取并播放一部数字化影片，往往需要长达几个小时的时间，作为良好的数据库管理系统，应该保证播放过程中不会发生中断，因此不得不增加处理长事务的能力；

2.2 多媒体数据库的实现途径

2.2.1 多媒体数据库的字段类型

要对多媒体信息进行存储，一般需要使用到以下的字段形式：

a．字符数值型数据

字符数值型数据记录的是事物非常简单的属性（如人的性别），数值属性（如人数），或是高度抽象的属性（如事物的所属类别）。这种数据具有简单、规范的特点，因而易于管理。传统数据库主要是针对这种数据的，在多媒体数据库中仍然需要管理这一类数据；

b．文本数据

文本是最常见的媒体格式，各种书籍、文献、档案等无不是由文本媒体数据为主构成的。在计算机内文本数据是由一个具有特定意义的字符串表示。字符串长短不一，给数据的存储和再现带来不便。自然语言理解技术的不成熟也使查询文本数据的难度加大。因此，许多通用型数据库系统根本就没有管理和使用文本媒体的有效手段。检索文本数据主要采用关键字检索和全文检索两种方法。关键字检索是在存储文本的同时，自动或手工生成能够反映该文本数据主题的关键字的集合，并将其存储在数据库中。检索时通过某些关键字的匹配找到所需的文本数据。全文检索方法可以根据文本数据中任何单词或者词组进行检索，检索是进行全文扫描。此外，

大多数的实用系统使用文件直接存储文本系统，或把数据规范化成标准长度的字符串。在普通数据库中并不具备很强的文本数据管理能力；

c．声音数据

音乐数据在计算机里是由字符表示的，因而数据量小，对它的存储、查询可以当作文本处理。但计算机目前还无法模拟不同人的口音，以及人们讲话时的抑扬顿挫的语气。因而语音数据还是以数字化的波形数据为主，这样存储空间就比较大。语音识别技术还没有达到可以广泛应用的程度，这为语音数据的检索带来不利。目前，对语音数据的检索主要有两种方法，第一种是给语音数据人工附加属性描述或文字描述，例如我们可以给录音数据附上讲话人的姓名、讲话日期、讲话题目和主要内容等。之后，我们就可以用字符数据和文本数据的检索方法检索语音数据。第二种方法是浏览，把语音逐一播放出来，边听边判断所需查找的语音数据，这种方法最大的缺点是速度太慢。在具体应用中，一般是与第一种方法配合使用，由第一种方法缩小范围之后再进行浏览；

d．图形数据

图形数据的管理已经有一些成功的应用范例，例如地理信息系统、工业图纸管理系统、建筑 CAD 数据库等等。图形数据可以分解为点、线、弧等基本图形元素。描述图形数据的关键是要有可以描述层次结构的数据模型。对图形数据来说最大的问题是如何对数据进行表示。对图形数据的检索也是如此。一般来说，由于图形是用符号或特定的数据结构表示的，更接近于计算机的形式，还是易于管理的。但管理方法和检索使用需要有明确的应用背景；

e．图像数据

图像数据是指图式图像。图像数据在应用中出现的频率很高，也很有实用价值。图像数据库较早就有研究，已提出许多方法，包括属性描述法、特征提取、分割、纹理识别、颜色检索等等。特定于某一类应用的图像检索系统已经取得成功的经验，如指纹数据库、头像数据库等，但在多媒体

数据库中将更强调对通用图像数据的管理和查询；

f．视频数据

动态视频数据要比刚才介绍的信息类型复杂得多，在管理上也存在新的问题。特别是由于引入了时间属性，对视频的管理还要在时间空间上进行。检索和查询的内容可以包括镜头、场景、内容等许多方面，这在传统数据库中是从来没有过的。对于基于时间的媒体来说，为了真实地再现就必须做到实时，而且需要考虑视频和动画与其它媒体的合成和同步。例如给一段视频加上一段字幕，字幕必须在适当的时候叠加到视频的适当位置上。再如给一段视频配音，声音与图像必须配合的恰到好处，合成和同步不仅是多媒体数据库管理的问题，它还涉及到通信、媒体表现、数据压缩等诸多方面。

2.2.2 多媒体数据库的实现途径

目前，多媒体数据库的实现常见有三种方法：扩充关系数据库法、面向对象方法和超媒体方法。由于关系数据库具有成熟的理论基础，有广泛的应用和成熟的技术，扩充关系数据库法实现起来也比较容易，对于一般的应用来说具有一定的优势，因此在该论文中主要讨论该方法，并用该方法实现了多媒体餐饮管理系统。

a．扩充关系数据库

关系数据库具有成熟的理论基础，有广泛的应用和成熟的技术，但关系模型只能描述字符和数字这些常规数据，无法描述图像、视频、动画、声音等多媒体信息。鉴于 RDB 缺乏处理复杂数据的能力，很多人正研究如何对其进行扩展。为支持多媒体数据，RDB 的扩展有以下几个方面：

（1）在 RDBMS 中引入抽象数据类型，用来描述复杂数据类型的逻辑表示扩充的数据类型主要有：图像 (image)、图形 (picture)、声音 (sound)，并应用一些新技术如 Windows 的 OLE(对象链接与嵌套技术)，从而使一些传统的 MDBMS，如 MS SQL Server、 FoxPro、Paradox、Oracle 等具有一定的多媒体管理功能。

这种扩充 RDBMS 支持多媒体信息存取的方法具有实现代价小，RDBMS 内核可被 MDBMS 利用，程序设计语言丰富等优点。但是这种实现方法也有严重缺陷，主要是对多媒体信息的建模能力差，无法反映多媒体信息中各媒体间的空间关系、时间关系和语义关系，有关处理必须由应用程序实现。此外，在多媒体信息的同步继承和表现上，在基于内容的查询和检索方面更是难以实现。

（2）引入嵌套表，在记录与表之间建立层次关。系关系模型中一条很重要的限制是一个关系必须是 1NF，即关系模型要求每个属性均为原子数据类型，这样，用关系为应用建模时，一个属性可能不得不存储在若干关系中，对象内部的结构联系通过关系联结属性来表现，故对复杂对象的处理变得十分困难。为此，有人提出打破关系模型约束的 NF2 模型，NF2 即非第一范式，不再遵从关系范式中“表中不允许再有表”的规定。这样，NF2 模型就允许关系的属性是另一个关系，因而支持层次结构，使层次结构语义在一个关系中直接得到体现。虽然这种方法可以利用关系数据库特有的优势，继承许多市场上的成果，但是，NF2 方法建模能力不强，虽然 NF2 方法对比传统关系模型可以描述信息的复杂结构，但在定义抽象数据类型和反映多媒体数据各成份间的空间关系、时间关系以及媒体对象的处理方法等方面仍有困难。例如要增加一种媒体，NF2 数据库无法在 MDBMS 中增加新媒体的类型定义以及对该媒体对象的处理方法，在特殊媒体的基于内容查询方面、存储效率方面等都有很大困难。这与它的数据模型的特性是密切相关的。

b．基于超文本或超媒体的数据模型

超媒体是近年来兴起的一种表示和管理多媒体信息的强有力工具，它采用非线性的网状技术组织和表示块状信息，结点 (node)和链 (link)是超媒体的两个核心概念，结点是信息的单位 ,可以包括文本、图像、视频、声音、动画等各种媒体信息。链用来组织信息，表达信息间的关系 (包括媒体间的时空关系)，结点联结成网状结构。

利用超媒体技术实现 MDBMS 还有许多问题及相关技术有待解决：

（1）如何用结点和链来组织和表示多媒体信息及其相互关系（包括媒体间的时空关系）；

（2）如何实现媒体间尤其是时基类媒体的同步和协调等等；

（3）多媒体信息检索和查询问题。超媒体中的浏览不是严格意义上的查询，它无法替代检索和查询；

（4）版本控制问题，即要求系统具有随着信息不断变化而更新版本的功能；

（5）标准化问题：包括对用户需求模式的描述 ,系统的体系结构、标准用户接口、数据交换格式与协议等。

超媒体数据模型一般说来比数据库数据模型还要高一个层次，它承担着建立超媒体超链联系的任务。在多媒体数据库中使用超媒体数据模型是为了建立多媒体数据之间的联系，包括时间、空间、位置、内容的关联，支持信息结点网的开放性，支持对信息结构的建模，支持浏览和搜索等新的操作。

c．基于面向对象的数据模型

面向对象方法的基本出发点是把客观世界的复杂实体的信息形态的各方面抽象为一个有机的研究目标，即对象。面向对象的数据模型必须提供表示对象及其相互关系的机制。在面向对象数据模型中，对象描述及其上的操作被封装为一个整体，因此可将对象看成是一抽象数据类型 ADT。ADT 实际上是一个数据结构的封装，利用其封闭、继承、聚合和归纳等特性，使其具体实现细节对外部过程是不可见的。

一个 ADT 属性可以表示一个复杂对象。用户只需按格式要求输入某 ADT 属性的参数，系统即自动生成对应的复杂对象。目前，已有一些面向对象数据库 (OODB)系统投入使用，如 IBM Almanden 研究中心的 XSQL 系统和 Ontologic 公司的 Vbase 系统等。这些系统为设计 MDBMS 提供了宝贵经验。

面向对象模型能较好地满足多媒体应用提出的建模和功能需求，具体表现在以下几个方面：

（1）支持“聚集”与“概括”的概念，从而更好地处理多媒体数据与复杂对象的结构语义；

（2）封装允许多媒体类型通过一个公用界面进行访问与操纵；

（3）支持抽象数据类型和用户定义的方法，便于数据库系统支持新的数据类型和操作；

（4）继承能够有效地减少媒体数据的冗余存储和由此引起的一系列问题，还非常有利于版本控制，同时它也是聚集分层和特性传播的基本方法；

（5）对象类与实例的概念有效地维护了多媒体数据的语义信息，也为聚集抽象提供了一种可行的方案；

（6）复合对象 (composite object)根据复合引用的语义，对象间的引用只是被引用对象的标志符放在引用对象的属性中，从而实现共享引用、依赖引用和独立引用，为多媒体数据的关系表示提供了一种很好的机制；

（7）面向对象系统的查询语言通常沿着系统提供的内部固有联系进行，避免了大量查询优化工作。

由于面向对象模型的上述优点，它能较好地解决多媒体信息表示和管理所面临的问题，因而受到了人们的重视。目前，国内外对基于 OO 的 MDBMS 研究应用采取了两种途径：

1）关系数据库 (RDB)与 OODB 集成

RDB 已积累了相当成功的经验，并为工业界广泛接受。将成熟的 RDBMS 的功能特性和面向对象的建模能力结合起来，从而提供对复杂数据进行查询的支持，这种方法形成的模型称为关系 -对象模型。这类模型保留了关系模型的集合操作、视图定义及代数优化等优点，增加了面向对象的概念，扩充了表达复杂关系语义和数据抽象的机制，如：类型、类层次、规则等，使其不但具有支持复杂应用所需的语义表达能力，而且具有较强的数据操作能力和系统效率，在关系 -对象模型中，关系模型的概念与面向

对象模型的概念可建立起对应关系：如具有唯一标识的元组可与对象对应，关系对应于类，元组或关系定义中的有关约束和规则可对应于方法。

关系-对象模型具有许多面向对象的特征，它是从关系模型和查询语言SQL出发，在此基础上建立起来的。SQL语言关于对象查询的扩充包括路径表达式，类似于方法的函数调用语法，以及对于FROM子句中嵌套集合的支持等。

用RDB与OODB集成的方法开发MDBMS的方法，一个优点是既保留了传统数据库的优点，又扩展了数据库的面向对象风范；另一个优点是可减少研制工作量，缩短研制周期；缺点是有一些面向对象的语义仍不能支持，而且由于保留关系数据库的存储结构而牺牲了一些面向对象的特征。

2）研究完善OODB，使之适合多媒体数据处理

OODB的研究始于1980年代中，早期的OODB研究主要集中在建造复杂对象的模型方面。在形式化描述和语言标准、嵌套关系、复杂对象演算方面的研究取得了相应成果，为构造复杂对象模型建立了形式化的理论框架。在实现方面，OODB也已从原型走向产品。

2.3 利用MS SQL Server 7.0实现多媒体数据库

2.3.1 MS SQL Server 7.0介绍

MS SQL Server 7.0是Microsoft公司开发的一种多线程关系型数据库管理系统（RDBMS）它能够处理大量的数据和管理众多并发的用户，而且具有事务管理的功能，保证了数据的完整性。此外，它还提供了许多高级管理和数据分布的能力。在多媒体应用方面，它支持TEXT和IMAGE两种BLOB(Binary Large Object)字段类型，而且还有三条独立的TEXT/IMAGE操作语句：writetext，readtext和updatetext，满足一般的多媒体数据库开发的要求。下面列举了MS SQL Servrer7.0的几个特点：

a. 全面的数据完整性保护

无论是复杂的事物支持和高级安全性，还是以用户数据库隐式部分支

持用户的商业

规则的对象，数据完整性都适用；

b. 与 Windows NT 4.0（或 Windows 2000 Server）集成

允许在 SMP（对称多处理）系统中彻底多线程和对称多处理，并且可集成到分布式管理环境中，因而它提高了系统的效率；

c. 具有一流的管理工具

MS SQL Server7.0 具有非常友好图形化管理工具和查询、事务跟踪工具；

d. 在分布式应用方面

它与 MS DTC（Distributed Transact Coordinator）结合，还具有很强的并发协调处理能力；

e. 在同级产品中具有较强的价格优势

由上可以看出，MS SQL Server7.0 具有很高的性价比，满足一般的多媒体数据库系统开发的要求，因此，在多媒体餐饮管理系统的开发中，选取 MS SQL Server7.0 是一种比较合理的选择。

2.3.2 利用 MS SQL Server7.0 实现多媒体数据库

SQL Server 7.0 的 TEXT/IMAGE 数据类型可以存储每行最长为 2GB 的 BLOB 对象。此外 SQL Server 为了提高性能和简化对这两种数据类型的访问，它还提供了三条语句来完成这些字段的读写操作，它们是 readtext，writetext 和 updatetext，分别实现 TEXT/IMAGE 字段的读出、插入和更新。

TEXT 和 IMAGE 数据存储在分离的数据页链中，远离行中的其他数据。在行本身，SQL Server 存储了 TEXT 和 IMAGE 数据页链接起始页的指针。上诉三条语句就是用该指针来找到数据页链直接修改该链而无需修改该行基本部分。

a. TEXT/IMAGE 的存储结构

建立多媒体数据库需要存储很大的二进制对象（BLOB），存储的数据可能多达几十兆甚至几百、几千兆（视频文件）。但是，SQL Server 限制最

大行长为 1962 个字节。为存储这些二进制对象，SQL Server 提供了 TEXT 和 IMAGE 数据类型，它可以保证能存储多达 2GB 的二进制对象。具体实现是将这文本/图像数据存储在一组链接的页上，这些页是从本身包含了的数据行分离出来的。

该数据行包含了一个 16 字节的字段，定义成 varbinary(16)，它包含了一个指针指向第一页，第一页中包含了那一列中文本/图像数据。如果需要，文本/图像数据就会串起许多页来存储数据，最大可达 2GB。每个 TEXT/IMAGE 页又都包含了一个嵌入式指针，用于指向存储该列的下一个链接页。其结构如下图所示：

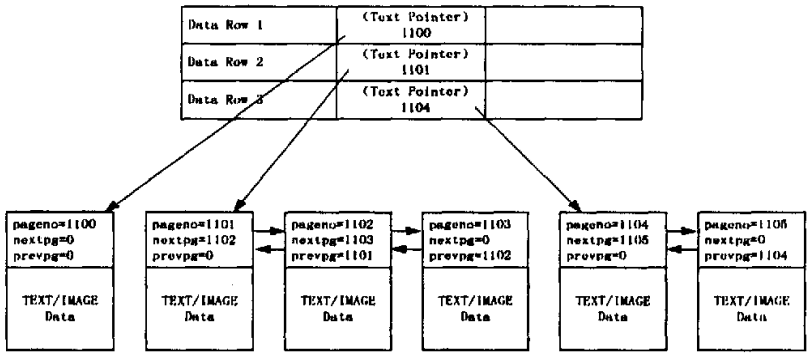

图 2.1 SQL Server 的文本/图像存储页

一个文本/图像页是一个标准的数据页，包括一个 32 字节的页头，每页还要加上一个附加 112 字节的冗余。一个文本/图像页可以存放 1800 字节的信息。文本/图像数据可以通过读取数据行中的指针，再通过文本/图像指针读取来自页中的数据，顺着文本/数据页中的嵌入式页指针一直读取，直到 nextpg=0 时为止。

b. TEXT/IMAGE 字段的约束

使用 TEXT/IMAGE 数据类型解决了多媒体数据的存储问题，但在使用这些数据类型时却并不象常规数据类型那样方便，它要受到一些限制：

（1） 能创建一个含有 TEXT/IMAGE 数据类型列的索引；

（2） where 查询子句中不能包含 IMAGE 类型的数据，在 TEXT 列中也只能使用 like

运算符，例如：

```
select * from dish_menu where
        dish_introduction='西安特色'
```

就会报错；

（3） 不能定义 TEXT/IMAGE 类型的局部变量。例如：

```
declare @@my_intrduction TEXT
```

就会报错；

（4） 在 TEXT/IMAGE 列上使用内建函数受到限制，但有时可以使用 CONVERT 函数作转换来避免其限制。例如：

```
select substring(dish_introduction,1,10) from dish_menu
```

就不允许，但可以这样使用：

```
select substring(convert(char,dish_introduction),1,10)
from dish_menu;
```

（5） 不能在 TEXT/IMAGE 列上执行标准的 SQL 合计与数值计算操作。例如：

```
select avg(sounddata) from musictable
```

就会非法；

（6） 在 GROUP BY 子句中不能包含 TEXT/IMAGE 列。

c. 多媒体数据表的创建

创建多媒体数据表与创建一般的数据表相同，只需要在相应的字段指明其字段类型为 TEXT 或 IMAGE 类型就可以了。例如：

```
create table dish_menu
    ( dish_id int identity,
    dish_name string notnull,
    dish_price string null,
```

```
dish_introduction text null,    /* 定义 TEXT 类型字段 */
dish_photo   image null)        /* 定义 IMAGE 类型字段 */
```

d. EXT/IMAGE 字段的维护

对 TEXT/IMAGE 的操作需要两个步骤，即首先必须找到行的页指针，然后在该页上执行相应的操作。SQL Server 提供了 textptr( )函数可以获得 TEXT/IMAGE 字段的页指针。例如：

```
select dish_id, textptr(dish_photo) dish_photo_textptr
from dish_menu
```

SQL 语句就可得到各行的 dish_photo 字段的页指针。其结果如下：

```
 dish_id          dish_photo_textptr
-------------------------------------------------------------------------------
1                 0x650100000000000000010000009a080000
2                 (null)
3                 0x6b010000000000000001000000a2080000
... ...
```

前面已讲到 SQL Server 提供 writetext,readtext 和 updatetext 三条语句可以对 TEXT/IMAGE 字段进行读写和更新。下面再分别阐述它们的使用：

（1）向 TEXT/IMAGE 字段中添加数据

使用 writetext 可以向数据表中添加一个新的 TEXT/IMAGE 值，例如：

```
writetext dish_menu.dish_photo
    0x6f010000000000000001000000f6080000 image_data
```

（2）更新 TEXT/IMAGE 值

writetext 语句还可以更新 TEXT/IMAGE 值。其方法是先获得页指针，然后再用变量的形式传递给 writetext 语句，其语法如下：

```
declare @pageptr varbinary(16)
select @pagerptr=textptr(dish_introduction)
```

```
        from dish_menu
        where dish_id=2
    writetext dish_menu.dish_introduction @pageptr
```

“老孙家羊肉泡馍具有悠久的历史，它采用优质内蒙嫩羊肉“

此外，SQL Server 还提供了 updatetext 专门用于 TEXT/IMAGE 值的更新操作，例如：

```
    declare @pageptr varbinary(16)
    select @pagerptr=textptr(dish_introduction)
        from dish_menu
        where dish_id=2
     updatetext dish_menu.dish_introduction @pageptr NULL 0
```

“和优质白吉馍经过数十道工序做成！“

（3）对 TEXT/IMAGE 字段的读去：

select 语句只能返回 TEXT/IMAGE 字段的前 255 个字节，而用 readtext 可以得到列中的任何字符的序列。下面的语句可以得到第 2000 字符为起点的 30 个字符：

```
    declare @pageptr varbinary(16)
    select @pagerptr=textptr(dish_introduction)
        from dish_menu
        where dish_id=2
    readtext dish_menu.dish_introduction @pageptr 2000 30
```

3 数据库的性能优化与安全性设计

3.1 媒体数据库在性能优化及安全性方面存在问题

多媒体管理信息系统（MMIS）通常要管理大量的多媒体数据，如声音、图像、图片和长文本等，这些数据通常需要很大的存储空间。例如一个视频图像文件通常要占用几十兆到数个吉字节的存储空间。

MS SQL Server 具有很强数据管理能力。它不仅能将数据库文件安装在几个大容量的硬盘上，而且还可以管理分布式的数据库，及可将数据库分布在多个数据库服务器上。在设计时应估算出数据库的容量大小并要充分的考虑到冗余。在采购服务器时应考虑到大容量硬盘驱动器或磁盘阵列。为保证访问速度，尽可能的使用高速 SCSI（Small Computer System Interface）硬盘和 RAID 卡以提供磁盘条带化（Tripping）等技术。

此外，由于多媒体数据库往往具有很复杂的关系模型，要搜索到需要的数据必须进行复杂的关系运算，如果数据库设计不当，会带来计算上的严重负担，影响访问速度。在设计数据库时要用到范式（NORM）。建立索引和合理的使用 SQL Server 优化器，也会对数据库的性能产生很大的影响。

在安全性方面，SQL Server 提供了比较全面的安全机制，例如访问控制、备份等。但要基本满足管理信息系统的安全性要求，还必须合理的使用这些安全机制，例如如何合理的划分用户和组等，如何对用户和组分配数据访问权限等。此外，对于一个安全、稳定的信息系统的要求来讲，还要考虑到数据备份、复制（Replication）等技术。

3.2 多媒体餐饮管理系统数据库的性能优化

3.2.1 范式（NORM）与数据库设计

在数据库系统工程中，必须满足的一个性质是数据完整性，即要求数据库中的信息完整齐备。有了完备的信息之后，需要按一定的原则对数据的结构进行规范化的设计，已达到在冗余性、灵活性和访问速度等方面性

能最优的效果。

数据组织的规范化形式是关系数据库的创始人之一——库德（E.F.Codd）首先提出的。早在1971年库德就提出了规范化理论，并在随后一系列的论文中逐步形成一整套数据规范化模式，这些模式已经成为建立关系数据库的基本范式。规范化表达中规定在每一个基本表中必须定义一个数据元素为关键字，它可以唯一地标识出该表中其它相关的数据元素。其中，数据元素就是基本表中的字段。在规范化理论中表是二维的，它的任意一列上，数据项应属于同一个属性，其行和列的顺序都无关紧要的，且不允许有相同的列和所有列都相同的行出现。

在对表的形式进行了规范化定义后，Codd还对数据结构进行了七种规范化定义，并定名为规范化模式，称为范式。在这七种范式中，一般只用前三种，对于常用系统就足够了。而且这七种范式是“向上兼容的”，即满足第七范式的数据结构必满足前六种范式，满足第四范式的数据结构必满足前三种范式，依此类推。

第一范式(first normal form,简称 1st NF)就是指在同一表中没有重复项出现,如果有则应将重复项去掉。这个去掉重复项的过程就称之为规范化处理。一般按规范化建立的表都满足 1st NF。

第二范式(second normal form，简称 2nd NF) 指表中其它数据元素都完全依赖于主关键字，或称该数据元素唯一地被主关键字所标识。化为该范式的规则是去掉功能上不完全依赖于主关键字的字段，它处理的是部分依赖关系。该规则只用于已经是第一范式的表，主要是处理关键字是两个或多个字段的表。

第三范式(third normal form，简称 3rd NF)就是指表中的所有数据元素不但要能够唯一被主关键字所标识,而且它们之间还必须相互独立,不存在其它的函数关系也就是说对于一个满足了 2nd NF 的数据结构来说,表中有可能存在某些数据元素依赖于其它非关键字数据元素的现象,必须加以消除。

下面是一个进行规范化的例子。该例子是对一个订单的表结构的规范化过程，如下图所示。图 3.1 中列出了订单中涉及的全部信息，没有规范化，所有信息作为一张表。图 3.2.把出现重复的字段提取出来，单独列为一张表：订购的产品，形成了第一范式。其中关联线上的数字表示一张销售订单可以有 1—7 种订购的产品，而一种订购的产品必须属于一张销售订单。图 3.3.运用第二范式的规则提取出一个新表：产品，把结构规范化为第二范式。之所以在产品和订购的产品两个表中都有产品单价这一字段，是因为单价可能随时间变化，使这两个表中的值不一致。图 3.4.从销售订单中提取出来客户表，而销售订单表中保留了客户编号来保持两个表之间的联系。

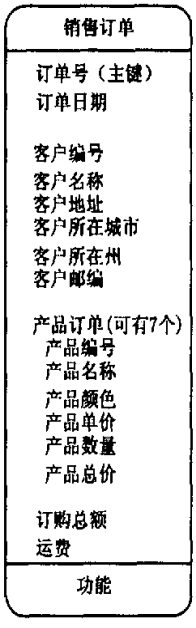

图 3.1. 非规范化结构

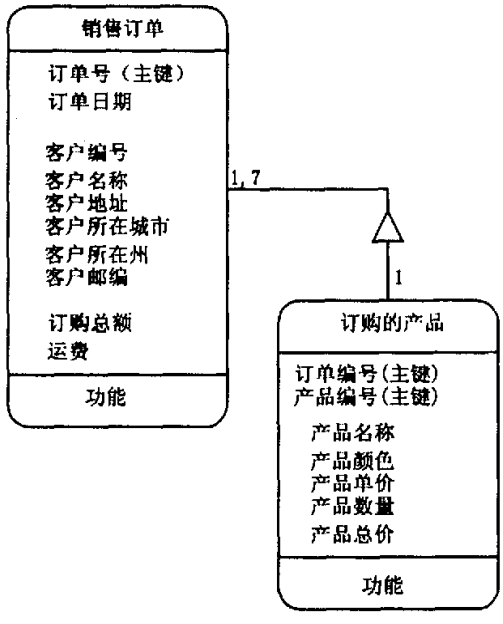

图 3.2. 第一范式结构

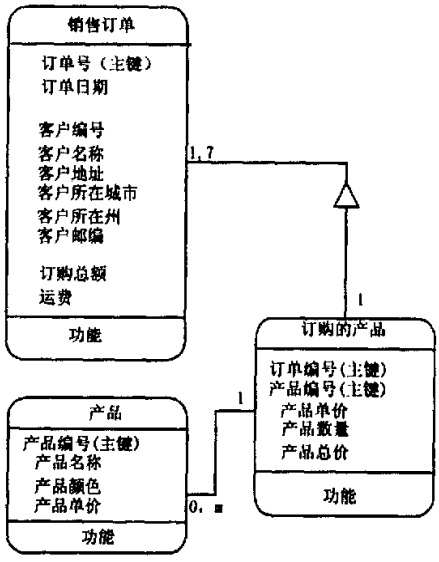

图 3.3. 第二范式结构

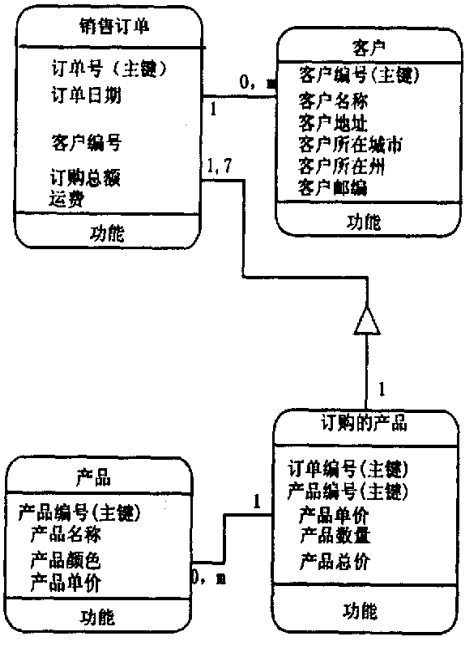

图 3.4. 第三范式结构

在利用扩展关系数据库实现多媒体数据库方法中，往往需要引入嵌套表，在记录与表之间建立层次关。有人提出打破关系模型约束的 NF2 模型，NF2 即非第一范式，不再遵从关系范式中“表中不允许再有表”的规定。这样，NF2 模型就允许关系的属性是另一个关系，因而支持层次结构，使层次结构语义在一个关系中直接得到体现。

3.2.2 检索策略

在关系数据库中首先需要正确的定义主关键值和外关键值。此外，为改善性能或加强唯一性，还得合理的设计检索策略。但数据库的检索方法没有对或错一说，不同的应用有不同的数据访问的描述，在一个应用下工作的很好的一个检索策略在另一个应用下可能无法工作。

SQL Server 在建立索引后，可将索引信息储存于磁盘上，在查找这些数据时，这些信息可帮助用户快速定位于相应的行。这些索引可以是簇式的，也可以是非簇式的。

SQL Server 提供两种索引类型：簇式（Clustered）和非簇式（Nonclustered）。而这都是 B-Tree 索引。对簇索引来说，数据是按照簇索引的顺序来保存的，因此，每一表只能有一个簇索引，因为数据只能按一种物理方式排序。但每个表中可有 249 个非簇索引，因为非簇索引保存的是行指针，而不是数据页。

一个索引可以包含 1 至 16 列，但索引总宽度不能超过 255 字节。服务器负责维护并使用索引，以改善性能或加强唯一性。

a. 簇式索引（Clustered Index）机制

对于簇索引，在中间索引级的末尾每个数据页都有一个入口。这表明数据页是簇索引的页级索引。簇索引的结构如图 3.6 所示：

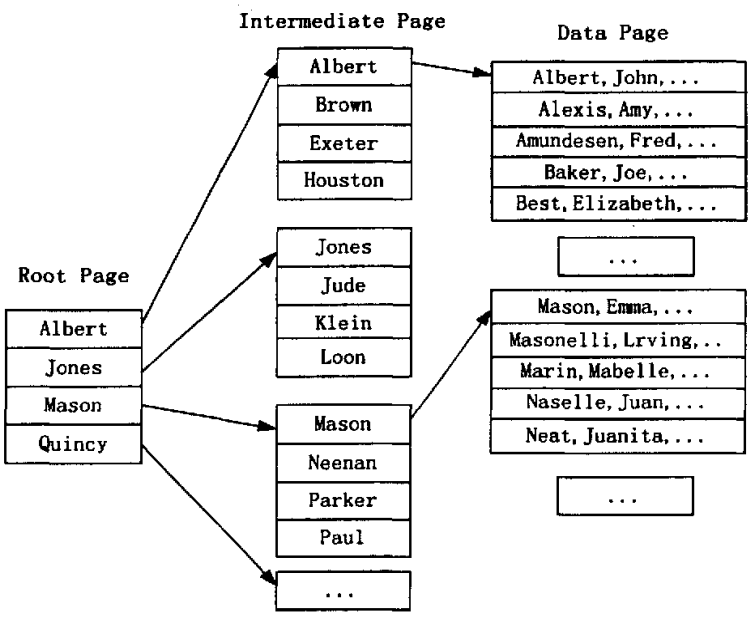

图 3.5 簇式索引机制

当定义一索引为簇式索引时，SQL Server 按索引的顺序排列基本表中的数据，当在表中定义一主关键字时，SQL Server 在该主关键字上创建唯一的簇式索引，除非指定创建的主关键字索引为非簇式的。

创建簇索引的语法如下：

```
create unique clustered index cindexofmenu
    on dish_menu(dish_id)
```

要创建簇索引，必须要保证数据库中有大约 1.2 倍表大小的自由空间，以确保表在排序时仍存在可用空间。

建立簇索引的基本思想是：

（1）大多数表都应该有簇式索引或使用分区来降低对表尾页的竞争，在一个高事务的环境中，对最后一页的封锁严重影响系统的吞吐量；

（2）在簇式索引下，数据在物理上按顺序排在数据页上，重复值也排在一起，因而在那些包含范围检查(between、<、<=、>、>=)或使用 group by 或 order by 的查询时，一旦找到具有范围中第一个键值的行，具有后续索引值的行保证物理上毗连在一起而不必进一步搜索，避免了大范围扫描，可以大大提高查询速度；

（3）在一个频繁发生插入操作的表上建立簇式索引时，不要建在具有单调上升值的列(如 IDENTITY)上，否则会经常引起封锁冲突；

（4）在簇式索引中不要包含经常修改的列，因为码值修改后，数据行必须移动到新的位置；

（5）选择簇式索引应基于 where 子句和连接操作的类型；簇式索引的侯选列是：

1）主键列,该列在 where 子句中使用并且插入是随机的;

2）按范围存取的列，如 dish_price >40 and dish_price < 100 ;

3）在 group by 或 order by 中使用的列;

4）不经常修改的列;

5）在连接操作中使用的列。

b. 非簇式索引（Nonclustered Index）

每一个表中只能有一个簇索引，但可以有多个非簇索引，每个非簇索引提供访问数据的不同排序顺序。当创建以非簇式索引时，SQL Server 仍旧读如所有的行，创建索引页，非簇式索引与簇索引的主要差异是非簇式索引的基础行不变化，物理上不对数据重新排序，这意味着当插入一新行到已有的行中时，不必考虑系统重新排序所有数据页的潜在可能性。非簇索引的机制如下图所示：

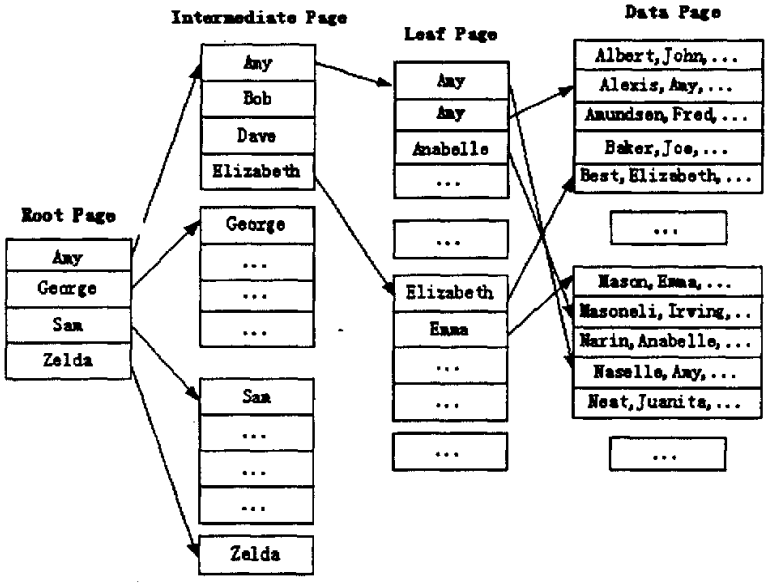

图 3.6 非簇式索引机制

创建非簇索引的语法如下：

```
create nonclustered index nindexofmenu
    on dish_menu(dish_name,dish_price)
```

若没有另行规定，在未指定索引类别时（缺省情况），创建的索引为非簇索引。

在建立非簇索引时，要权衡索引对查询速度的加快与降低修改速度之间的利弊。另外，还要考虑下列问题：

（1）索引需要使用多少空间；

（2）合适的列是否稳定；

（3）索引键是如何选择的，扫描效果是否更佳；

（4）是否有许多重复值。

对更新频繁的表来说，表上的非簇索引比簇索引和根本没有索引需要更多的额外开销。对移到新页的每一行而言，指向该数据的每个非簇索引的页级行也必须更新，有时可能还需要索引页的分理。从一个页面删除数据的进程也会有类似的开销，另外，删除进程还必须把数据移到页面上部，以保证数据的连续性。所以，建立非簇索引要非常慎重。非簇索引常被用在以下情况:

（1）某列常用于集合函数(如 Sum,....)；

（2）某列常用于 join,order by,group by；

（3）查寻出的数据不超过表中数据量的 20%。

c. 覆盖索引(Cover ing Indexes)

覆盖索引是指那些索引项中包含查寻所需要的全部信息的非簇索引，这种索引之所以比较快也正是因为索引页中包含了查寻所必须的数据,不需去访问数据页。如果非簇索引中包含结果数据，那么它的查询速度将快于簇索引。

但是由于覆盖索引的索引项比较多,要占用比较大的空间。而且 update 操作会引起索引值改变。所以如果潜在的覆盖查询并不常用或不太关键，则覆盖索引的增加反而会降低性能。

d. 索引设计准则：

良好的索引设计可以带来系统性能极大的改善，特别是对于面积很大的数据表。但对于不合理的设计，也可能造成效率降低，甚至无法工作。通常，索引的设计须遵循下列准则:

（1）对表中已依次排列的列集合只能定义一个索引；

（2）最好是为每个表均建立一个簇索引，否则删除行所得到的空间就不会被重新利用。如果表中没建立簇索引，那么新增行和修改后的行将被存储在表的末尾；

（3）索引的数量太多时，修改时用来保存索引的开销就会太大；在执

行信息系统（EIS）或决策支撑系统（DSS）中，由于没有实时更新，所以可以有较多的索引，索引数量只受磁盘空间的限制；

（4）机事务处理（OLTP）应用中，应使用尽可能少的索引，以加快更新、插入、删除的速度。

由于索引对查询有很大的益处，但它对需要频繁的插入、更新和删除记录的应用（如 OLTP）来说，就会带来很大的开销，导致速度降低。其中较好的办法就是建立两个独立的数据库，一个用于 EIS 或 DSS，一个用于 OLTP。但这种方案需要某种机制保证两个数据库的同步。MS SQL Server7.0 提供了数据复制（Replication）的功能可实现这种机制。关于数据复制，将在后面的章节中有所讨论。

e. 索引的设计思路

索引的有无，建立方式的不同将会导致不同的查询效果，选择什么样的索引基于用户对数据的查询条件，这些条件体现于 where 从句和 join 表达式中。一般来说建立索引的思路是：

（1）主键时常作为 where 子句的条件，应在表的主键列上建立簇式索引，尤其当经常用它作为连接的时候；

（2）有大量重复值且经常有范围查询和排序、分组发生的列，或者非常频繁地被访问的列，可考虑建立簇式索引；

（3）经常同时存取多列，且每列都含有重复值可考虑建立复合索引来覆盖一个或一组查询，并把查询引用最频繁的列作为前导列，如果可能尽量使关键查询形成覆盖查询；

（4）如果知道索引键的所有值都是唯一的，那么确保把索引定义成唯一索引；

（5）在一个经常做插入操作的表上建索引时，使用 fillfactor(填充因子)来减少页分裂，同时提高并发度降低死锁的发生。如果在只读表上建立索引，则可以把 fillfactor 置为 100；

（6）在选择索引键时，设法选择那些采用小数据类型的列作为键以使每个索引页能够容纳尽可能多的索引键和指针，通过这种方式，可使一个查询必须遍历的索引页面降到最小。此外，尽可能地使用整数为键值，因为它能够提供比任何数据类型都快的访问速度。

f. 索引的维护

不合适的索引影响到 SQL Server 的性能，随着应用系统的运行，数据不断的发生变化，当数据变化达到某一个程度时将会影响到索引的使用。这时需要用户自己来维护索引。索引的维护包括：

（1）重建索引：随着数据行的插入、删除和数据页的分裂，有些索引页可能只包含几页数据，另外应用在执行大块 I/O 的时候，重建非簇索引可以降低分片，维护大块 I/O 的效率。重建索引实际上是重新组织 B-树空间。在下面情况下需要重建索引：

1）数据和使用模式大幅度变化；

2）排序的顺序发生改变；

3）要进行大量插入操作或已经完成；

4）使用大块 I/O 的查询的磁盘读次数比预料的要多；

5）由于大量数据修改，使得数据页和索引页没有充分使用而导致空间的使用超出估算；

6）dbcc 检查出索引有问题。

当重建簇式索引时，这张表的所有非簇索引将被重建。

（2）索引统计信息的更新：当在一个包含数据的表上创建索引的时候，SQL Server 会创建分布数据页来存放有关索引的两种统计信息：分布表和密度表。优化器利用这个页来判断该索引对某个特定查询是否有用。但这个统计信息并不动态地重新计算。这意味着当表的数据改变之后，统计信息有可能是过时的，从而影响优化器追求最优工作的目标。因此，在下面情况下应该运行 update statistics 命令以更新统计信息：

1）数据行的插入和删除修改了数据的分布；

2）对用 truncate table 删除数据的表上增加数据行；

3）修改索引列的值。

3.2.3 查询优化

a. 优化器的基本概念

查询优化就是分析单个的查询从而确定一个最好的实现方法的过程，它包括了解从基本的存储结构到定义于这些查询之上的索引，以确定是否存在更有效的实现方法。SQL Server 采用了基于开销（cost-based）的优化器。查询优化器检测分析过的 SQL 查询并在基于涉及对象的有关信息下，输出一查询方案。查询方案是完成查询的一系列执行步骤。深刻的理解优化器的工作原理有利于我们设计出最有效的查询方案。

SQL Server 处理查询时，按以下步骤执行：

（1）规范化查询的合法语句和对象引用；

（2）优化查询并生成查询方案；

（3）编译查询方案；

（4）执行查询方案并将结果返回给用户。

其中优化步骤可分解为如下几个阶段：

（1）查询分析

1）寻找查询参数（SARG）

2）寻找 or 字句

3）寻找连接操作

（2）索引选择

1） 为 SARG 选择最好的索引

2）选择最好的 or 运算方法

3）为每个 join 字句选择最好的索引

4）为每个表选择最好的索引

（3）连接顺序的选择

1） 检查连接顺序

2）计算开销

3） 评估分解连接的其他服务器选项

（4）方案的选择

优化器会在执行查询时自动实现优化操作，而不需要程序员去干预。但不同的查询方案会带来不同的优化效果。下面将讨论如何根据优化器的工作原理设计最优的查询方案。

b. 查询方案设计

（1） 定搜索参数（SARG）：

搜索参数的存在使优化器能够限制搜索到的满足某查询的行数。优化的基本目标是用索引来匹配一个 SARG，从而避免表扫描。查询参数为一 where 字句，其格式如下：

columnname operator constant [and...]

其中合法的操作符（operator）可以为=、>、<、>=、<=，而!=、<>、like 和 between 等。如果使用了非法的操作符，优化器将忽略那一作为搜索参数的语句，因为它不能用来匹配索引中不存在的值，除非查询能够通过索引覆盖的方式解决。例如：

select * form dish_menu where dish_price != 0

就是非法的 SARG 语法；

下列情况也属于非法的 SARG：

1）中一列与另一列进行比较，并非与某一常数进行比较，如：

lname=fname；

2）上有函数运算，例如：

upper(name)='JOHN',dish_price*0.8>60;

由上可知一些 SQL 语句并不遵循 SARG 的 SRAG 的语法，这些 SQL 语句虽然在执行时不会报错，但它们会导致表扫描，从而导致性能下降，在设计时应该避免。

有时这些 SQL 语句也可以改写成合法的 SARG 语句，而不影响查询结果。例如：

dish_price !=0 改为：

dish_price >0

dish_price between 10 and 20 改为：

dish_price >=10 and dish_price<=20

au_name like 'Mi%' 改为：

au_name >='Mi' and au_name<='Mj'

……

（2）or 策略：

or 子句的格式如下：

SARG or SARG [or ...]

or 子句是一种选择判断。符合两个标准之一的所有行都会出现在结果集中。同时符合两个标准的行也只出现一次。

优化器在处理 or 子句的方式不同于标准的 SARG。or 子句所包含的所有列属于同一数据表。

SQL Server 优化器在处理 in 子句仍会当作 or 子句处理，例如：

dish_category in ('海鲜'，'凉菜'，...)会当作

dish_category='海鲜' or dish_category='凉菜' or ...

处理。

or 子句即可用表扫描处理，也可用 or 策略处理。使用表扫描时，SQL

Server 读取表中的每一行并与所有的搜索标准比较。Or 策略是将查询分成两个或更多的部分，用有效的索引运行每一部分并得到匹配行的 ID(行号)，在这些 ID 中执行 UNION 对行排序以删除重复行，最后，把行的 ID 作为动态索引从基本表中取得数据。

假设查询语句为：

```
select * from dish_menu
        where dish_name='鱼香肉丝'
                or dish_category='凉菜'
```

其工作过程如下：

1）用表扫描的时间；

2）查询分成多个部分，如将该例分解为：

```
select * from dish_menu where dish_name ='鱼香肉丝'
select * from dish_menu where dish_category='凉菜'
```

3）每一部分，取得各行的 ID，作为“动态索引”保存在临时表中；

1.行 ID 进行 UNION 操作以排序并删除重复行；

2.态索引从中求得符合条件的行。

如果索引在 or 子句的两列中都出现，SQL Server 会评估使用 or 策略。如果 or 子句中某一部分需要采用表扫描，或通过评估表扫描更有效，则 SQL Server 将简单的采用表扫描来完成整个查询而不采用 or 策略。

如果 or 子句只包含以列，如：

```
select * from dish_menu where dish_name='鱼香肉丝'
        or dish_name='清蒸桂鱼'
```

并且在这列有簇索引，优化器将寻找一 between 子句来代替而不使用 or 策略。因为数据是分段的，优化器知道满足 or 自句的所有值都将位于两个值之间。因此，它只是找到第一个匹配 or 子句的所有值都将位于两个

值之间。因此，它只是找到第一个匹配 or 子句的行并简单的扫描接下来的数据页，直到最后一个匹配行找到为止。在某些场合，这种方法可能会导致比用动态索引更少的 I/O 开销。

除非使用了 or 策略，否则将在每个表中使用一个索引以满足查询。

3.2.4 存储过程的优化

存储过程（Stored Procedure）是分析和编译后的 SQL 程序，它驻留在数据库中，可以被客户应用程序通过引用其名称调用。使用存储过程可以减少网络传输和加快 SQL 的执行效率。此外，它还具有如下优点：

（1）程序的模块化

（2）对表的严格的、基于函数的访问

（3）减少操作错误，提高一致性

（4）能自动处理复杂或敏感的事务

（5）提高了软件的可重用性

在这里主要讨论存储过程对系统性能的优化及如何优化存储过程。

存储过程能包含巨大而复杂的查询或 SQL 操作，它们被编译并存储在数据库内，当用户发出请求时，它们就在 SQL Server 上运行，只把结果返回给用户，而不必每次需要运行程序时将程序代码从客户端送给服务器，也不必将一些中间结果送给客户进行处理，从而降低了网络传输。

使用存储过程的最大优点是提高了运行效率。SQL Server 不仅保存了存储过程执行代码，而且它还保存了优化的查询方案，在需要执行时直接使用，而不需要每次运行时再进行优化，从而省去了大量的处理开销，提高了效率。查询方案是存储过程第一次执行时建立的，它存在于高速缓存之中。

下表是存储过在第一次执行和在后续执行时的比较：

表 3.1. 存储过程的执行过程比较：

第一次执行	后续执行
在磁盘上定位存储过程并装入缓存 代入参数 制定优化方案 编译优化方案 从高速缓存中执行	在高速缓存中定位存储过程 代入参数值 从高速缓存中执行

然而，由于后续执行时使用的是第一次运行存储过程时所得到的优化方案，而优化方案将得不到实时的更新，这样在后续执行时会带来一定的性能损失。这是因为最优查询方案会因数据表、统计数据和参数的变化而有所不同。如果对数据量很大表进行查询，这种损失就非常明显，有时这种损失要远远大于优化过程所带来的开销。

对这个问题，可以用重编译（Recompile）的方法加以解决，即在每次执行存储过程时进行重新优化和编译。

重编译有两种方案，一是选择用 with recompile 选项建立存储过程，如：

```
create proc print_orders(@lowdate datetime,@highdate datetime)
with recompile
as
select * from dish_debt
where repastdate between @lowdate and @highdate
return
```

这个选项迫使 SQL Server 优化器为每次运行都建立一个新查询方案。

另一个方法是按通常方法建立存储过程，在执行时使用 with recompile 选项，如：

```
exec print_orders @lowdate= " 04/20/2002 " , @highdate= "
    05/19/2002 " with recompile
```

3.2.5 TEXT/IMAGE 列与性能优化

在前面已论述过，对 TEXT/IMAGE 列的读取至少需要两次 I/O 操作，一次用于读取指针，还要至少读取一次 TEXT/IMAGE 数据，因此会带来很大的 I/O 开销。此外，由于每个数据无论多小都会占据至少 2K 字节的空间。如果要存取 40 字节的 TEXT 值，而且有 1 百万条这样的记录，就会占据 2GB 的空间，但若采用 VARCHAR 类型，则只需 41M 的空间。事实上，在很多情况下应避免使用这样的字段，特别是在记录数很多，数据大小差异很大的情况下。有两种方法可以避免使用这样的字段，而同样能够存取多媒体数据又可以提高速度而且还节省了大量的空间。

方法一：将多媒体数据存储在其他地方而不存放在数据表里，在数据库里只需存取它的存储位置或路径。例如可将 hsbc_introduce.rm 视频文件存放在共享目录 E:\dish\intrduce\下，而在 dish_menu 表中的存储它的路径。为存储其路径，可将 dish_menu 表中加入 dish_introduce 字段，将其属性设为 VARCHAR(30)。

要存入数据，可分两步：

(1)多媒体文件拷贝到本地硬盘的某个目录下(如：E:\dish\inroduce\)。通常由程序完成；

(2)将文件名(如 hsbc_introduce.rm)与存放路径(如：E:\dish\inroduce\)合并，得到多媒体数据的存放位置信息，将这个位置信息字符串插入数据表。

要读取数据，则只需先从数据表中读出多媒体数据所存放的位置信息，然后由操作系统根据得到的位置信息获取数据；

要修改数据，可将旧的数据拷贝到其他位置作备份用，并将新的数据直接覆盖原有的数据，而不需要对数据表进行操作。

方法二：将 TEXT/IMAGE 数据拆分成多个小的数据片(1~255 字节)，然后将这些小的数据片分别存放在不同的行的 varchar(255)或 varbinary(255)的字段里。读取时先将这些数据片读出然后再串到一块儿就可以了。这虽然在程序上实现起来比较麻烦，但有时它能带来性能的很大的改善。

3.3 饮管理系统数据库的安全性设计

针对本文说涉及的多媒体餐饮管理信息系统来说，在安全性方面必须满足下列要求：

对用户或组的管理：它要求具有对用户或组的添加与删除的能力，具有对用户或组权限的有效控制的能力，如允许哪些用户对某些数据表的访问，是否对这些表有更改的权限；

容错能力：系统应具有对某些非法的操作或具有容错能力，它不会因为输入了非法的数据造成整个系统的崩溃，或由于在网络环境中多个用户同时对某个表进行操作时造成修改数据的混乱；

对灾难的恢复能力：它应具有在某些灾难发生后对灾难所造成的数据丢失具有恢复能力。如在保存数据的某块硬盘怀掉后不会造成数据的永久丢失。

在这一节，主要针对数据库和数据库服务器的安全性进行讨论。事实上，MS SQL Server 7.0 提供了比较完善的安全特性，包括用户管理、锁定、备份、复制等。但要真真的实现一个满足安全要求的系统来说，还必须进行合理的规划和设计，包括从体系结构、软硬件平台的选择、用户管理到数据库、应用软件的设计等等。这一节主要讨论在 MS SQL Server 7.0 提供的安全特性的基础上如何构建安全的多媒体数据库。

3.3.1 MS SQL Server 7.0 的安全特性

SQL Server 在用户验证上有两个安全级别：登录鉴别；数据库用户帐号和角色的许可验证。

登录鉴别：用户可以用 SQL Server 帐号和口令登录，也可以用 windows

NT 分组或帐号登录。这需要对 SQL Server 的鉴别模式事先设置，要么windows NT 鉴别模式，要么混合鉴别模式。

数据库用户帐号和角色的许可验证：角色即 SQL Server 系统设计和定义好了的权限组合，不同的角色具有不同的权限。角色的分配是由 DBA（数据库系统管理员，database

administrator）针对不同的用户、密级等来决定的。如图 3.11 所示：

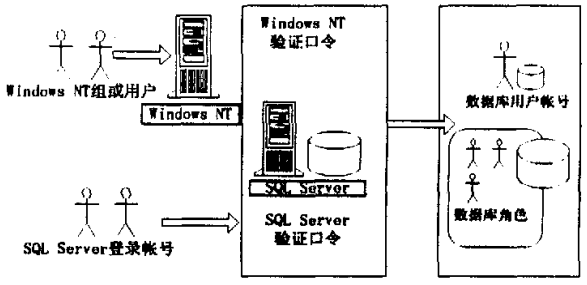

图 3.11 SQL Server 的登录安全模式

在硬件上，可以使用 RAID（冗余独立/廉价磁盘阵列 Redundant Array of Independent/Inexpensive Disks）技术来提高数据的安全性。它主要是使用校验码或硬盘镜像等技术提供了数据的安全性，这种安全性是指在存储数据的多块硬盘中其中一块或几块（不能是全部）损坏后数据仍能恢复。目前常用到的 RAID 包括 RAID0、RAID1 和 RAID5，其中 RIAD5 利用了磁盘条代化(Tripping)和 ECC 技术，不仅提高了 I/O 速度，而且提高了安全性。

然而，仅依靠存储机制还不能完全满足系统的安全性要求。比如对数据的非法篡改和误操作造成的数据丢失就无法对数据进行恢复。SQL Server 提供了日志、备份和复制等机制可以对这样的数据丢失或损坏进行恢复。下面主要讨论数据库的备份和复制。

3.3.2 数据库的备份

数据库的备份是数据库系统的管理中非常重要的一个部分。天有不测

风云，当任何可能引起数据丢失的灾难性事件发生时，如服务器崩溃或数据库损坏或硬件损坏等，唯一可做的事就是利用数据库备份进行恢复，否则只有从头再来，但是会耗费大量的人力物力和时间。甚至有些情况下，从头收集数据根本无法实现。

不管数据的损失是意外的还是人为的，都应认真考虑数据库的备份策略和方法。一般从以下几方面考虑：

a. 库系统的联机处理

库系统主要用于哪些处理？OLTP（联机事务处理），还是 OLAP（联机分析处理），由

此大致估计备份数据的频繁性；

b. 备份的过程

知道备份的过程是动态的，除了个别例外情况，备份可以在数据库正在被使用或数据正在被修改时进行；

c. 备份内容：

模式和文件结构、数据、部分事务日志文件；

d. 在 SQL Server 数据库中，系统信息存储在系统数据库中，主要的系统数据库包括：

（1）master-从整体上控制用户数据库和 SQL Server 操作，在创建了任何用户定义的对象后，都要备份它；

（2）model-为新数据库提供模版和原型，若有过修改，也要备份它；

（3）msdb-包含了有关作业、报警及操作员等信息，若有过修改，也要备份它。

对系统数据库 master 的备份非常重要。因为 master 数据库是用于跟踪用户帐号、远程用户帐号、远程服务器、环境变量、系统错误消息、其他系统数据库、数据库存储分配、设备和活动等的系统数据库。Master 数据库的损坏将导致 SQL Server 无法启动。此外，Master 数据库中的一些系统

表可以理解成数据字典，它是系统自动扩充的，而在以前数据字典都是在数据库设计中所要考虑的事情。

e. 备份方式：

（1）备份，备份整个数据库的数据、存储过程、用户帐号等参数的完整备份；

（2）差异备份，将自上一次全库备份以来的变化进行备份；

（3）事务日志备份；

（4）数据库文件或文件组的备份，这是对特大型数据库或由于时间原因而无法全库备份时的考虑。

根据以上考虑，本文制定了一个基本的数据库备份策略。由于多媒体餐饮管理系统中部分数据如菜单、配料等数据具有相对稳定性，可采用差异备份的方式，而如库存、账单、顾客等信息每日都有很多新的数据进入，数据更新较频繁，采用差异备份会带来很大的系统损耗，采用每日定期备份是一种比较合理的选择，比如可以选择每日凌晨 1:00 开始自动备份。此外，在对具有相对稳定性的数据进行大量的更改后可以人为干预的进行备份。

在数据遭受到破坏（非法篡改或物理破坏等）后需要将数据从备份数据库里恢复到工作数据库里。通常，数据还原需要人工干预。可以有多种方法进行数据还原：

（1） load database 命令进行还原，其语法如下：

load database databasename from devicename

（2） 以使用 load table 命令对单个表进行还原，其语法如下：

load table tablename from devicename

（3） 使用 SQL-EM(Enterprise Manager)还原数据库。使用 SQL-EM 的 TOOLS 菜单下面的 Restore 就可以恢复数据库。

3.3.3 复制技术在本系统中的运用

复制（Replication）是将一组数据或影响数据改变的事务从一个数据源只读的拷贝给多个潜在源的过程。如果拷贝的是影响数据的事务，那么这些记录源数据库数据改变的事务拷贝到目标数据库后也必须作用于该数据库，这样就会保证目标数据库的特定数据集与源数据库的一致性。

数据复制与备份虽然在物理上都是提供数据的拷贝。但备份是基于数据安全性考虑的，它主要是由 DBA（Database Administrator 数据库管理员）或由软件实施的大批量的数据的原始拷贝。而复制则主要是基于应用考虑的，虽然也能提供一定的安全性。它是为分布的应用环境提供多份的数据拷贝并保证它们的一致性而采用的这样一种机制。复制与备份相比前者具有很大灵活性和时实性，而且还具有备份所不具备的伸缩性、异类性和互操作性。

通常需要在下列情况下需要考虑复制技术：

a. 将数据提供给多种的应用环境

在前面讨论索引时讲到在 DSS 系统中利用索引技术可以大大的改善信息检索的性

能，而在需要频繁的更新数据的 OLTP 系统中，使用索引会导致系统性能的下降，若建立两个数据库服务器分别用于 DSS 和 OLTP 并用复制技术保证数据的一致性就可以解决这样的问题；

b. 整个系统跨越较大地理位置

例如分布于不同的城市，利用 VPN(Virtual Private Network 虚拟专网)建立连结，如果这个系统只利用中心数据库（Center Database）来维护数据，那么在处理数据时就会频繁的建立连结，而造成大量的 Roudtrip，特别是在多媒体数应用中。此外还会给中心数据库服务器带严重的处理负荷，这是不希望的。若在每个分公司或部门建立一个部门数据库，然后从中心数据库“定制”必要的企业数据，并将自己业务信息“推”给中心数据库，就可以解决这样的问题；

c. 关于负荷的考虑

如果数据库服务器的配置并不是很高，而用户量很多，数据访问的突发性较强，则会给服务器带来很大处理负荷，严重的情况下可能会造成服务器的瘫痪。

利用复制技术将数据分发给多个工作组数据库服务器，让各个用户分别访问它们的工作组服务器，就会将这些负荷分担。而数据可利用复制技术保持它们之间的互操作性和一致性；

d. 安全性考虑

有时企业数据库不仅为企业内部服务，还要提供互联网上的信息查询和数据搜集（网上订货）等功能。企业内部的数据往往是很敏感的，而又很难保证互联网不会被黑客攻破，被黑客攻破盗走企业数据是一件灾难性的事。如果建立两个服务器，一个用于企业内部，一个用于互联网，Web服务器可以从企业数据库服务器定制必要的数据，而将敏感数据隐藏起来，就可以解决这样的问题。

MS SQL Server 7.0 复制（Replication）是在 SQL Server 6.0 引入的“发布和预定的”模式的基础上发展起来的。SQL Server 将复制的源数据库服务器定义为出版服务器（Publication Server），将复制的目标数据库服务器定义为订阅服务器（Subscription Server）。复制具有两种方式：

a. 推式复制（Push Replication）

指发布者不需要订阅者的请求而向订阅者告知数据发生的变化。它一般用于在数据发生变化时告知别的数据库（订阅者），或订阅者用来设置调度集；

b. 拉式复制（Pull Replication）

是指订阅者询问发布者所有改变的阶段性更新。它对具有大量订阅者的出版物是很适用的；

此外，SQL Server 还在此基础上提供了三种复制类型：

a. 快照复制

是指在某一时刻及时的为数据库中的发布数据建立一个图像或快照。快照复制是最简单的复制类型，并且可以保证发布者和订阅者之间潜在的一致性。快照复制适合于只读的应用方案（如查找表或编码表），以及数据等待时间要求不苛刻并且数据量不大的 DSS 系统。快照复制可以调度执行（推式）也可请求执行（拉式）；

b. 事务复制

是指在事务一级对发布服务器的变化进行监视：及插入、删除或更新操作。发布者的变化持续的或在调度间隔时流向一个或多个订阅服务器。变化几乎是实时的传播，通常需要几秒的延迟，对于事务复制，变化必须在发布站点上进行以避免冲突保证事务一致性；

c. 合并复制

它提供了最高级别的自治性。发布者和预定者可以独立的工作并阶段性的重新连结以合并复制结果。如果多个站点对相同数据元素所做的改变发生了冲突，它会自动解决这些冲突。

在建立的多媒体餐饮管理系统中，由于用户往往比较多，而且有时还会有分店，加之多媒体数据的传输量较大，因此利用总店的中心数据库服务器处理所有访问者的请求会给服务器带来很大的负荷，而且会给网络带来极大的传输量，这对硬件的要求很高。为建立这样的系统，有时可能需要购置数十万元的小型机和铺设价格昂贵的光传输网络。这对一个餐饮企业来说是一个极大的开销。但是若利用复制技术将数据复制给多个部门数据库服务器，然后让用户访问部门数据库，只需铺设普通的五类线乙太网、购置多台价格要低廉得多的部门级服务器就可以了。复制技术保证了数据处理的互操作性和同步性。

在该系统中，可以对菜谱、菜单、原料、代码这些不需要经常改变、由技术员集中维护的数据可以采用推式事务复制方式，而对客户信息、点

菜单、账单等需要频繁更新、由各服务台维护的数据可用拉式合并复制方式，这样以便中心服务器有效的控制数据合理的分发和时实性，而不导致冲突。

4 多媒体餐饮管理系统的开发

4.1 多媒体餐饮管理系统需求分析和总体设计

4.1.1 多媒体餐饮管理系统简介及任务需求

针对需求的分析有助于我们建立明确的软件框架，该系统是为某餐饮企业建立的一个管理信息系统。该公司是一家大型的餐饮公司，主要经营餐饮业，它在市内拥有 5 家分店，并计划在其他的城市开设分店。要建立的系统必须满足下列其本要求：

1. 加强酒店管理，改善各部门各岗位的协作性，提高工作效率；

2. 分析经营状况，以便提高经营效益；

3. 介绍酒店特色和特色菜，使用多媒体菜单，方便顾客点菜，促进销售，刺激消费；

4. 有效库存管理。

具体任务需求：

1．酒店的介绍、特色菜、饮食文化、常识等介绍：

饮食文化：客人可以了解很多饮食文化方面的习俗、食俗、典故等；

2．体菜单的查询、为各户提供在线点菜服务：

作为多媒体形式的菜单，可以向客人展现菜、饭的外观、烹饪方法、原材料简介、营养成分、所属菜系、历史典故、民族习俗等；

3．客户提供消费情况（品名、数量、单价、总额等）的查询服务；

4．为顾客提供服务需求呼叫功能；

5．顾客意见薄功能；

6．为客户提供点播歌曲、影视节目的服务；

7．服务请求应答；

8．打印消费清单和账单；

9．对顾客消费账单的结算；

10．察看顾客意见，及时反映情况。

11．分别打印某台顾客的点菜请求分给相应的厨师；

12．库存状况查看；

13．库存自动更新；

14．对各种菜所点的频次进行统计，以便合理计算各种原料的储备；

15．入库管理；

16．库存状况查询；

17．库存短缺预警；

18．经营状况分析；

19．进行成本利润分析，宏观管理；

20．进行顾客流进行统计，改善服务质量；

21．人员的部门、岗位分配；

22．人员的薪酬管理；

23．总的劳力成本核算。

4.1.2 系统总体设计

a. 网络拓扑设计

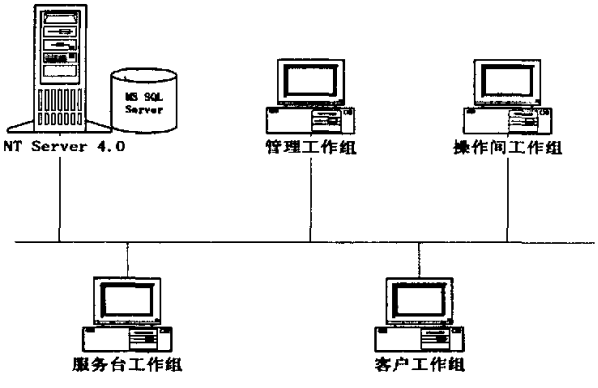

图4.1. 多媒体餐饮管理系统网络拓扑

b. 业务流程图

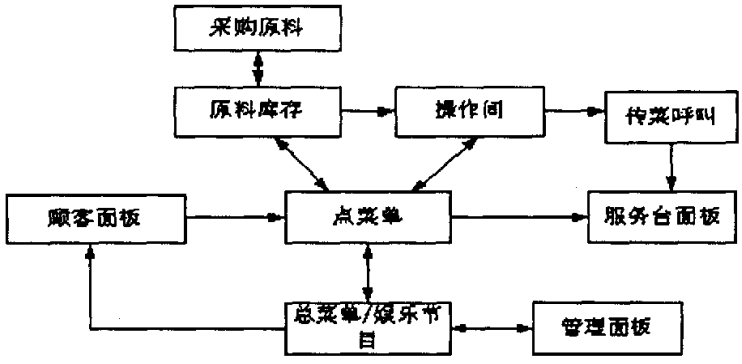

图 4.2 多媒体餐饮管理系统业务流程图

c. 软件模块设计

（1）统的模块划分

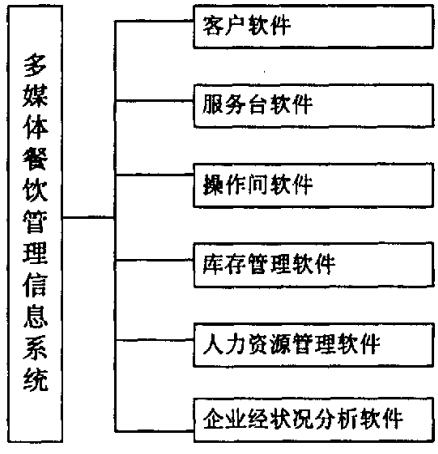

图 4 .3 餐饮管理系统软件模块划分

（2） 模块的主要功能

● 客户软件模块：酒店、饮食文化、特色菜等介绍、菜单查询、点菜、服务请求、意见薄、消费情况查询、歌曲、影视节目点播等；

● 服务台软件模块：响应客户服务请求、打印账单、结算账单，响应操作间传菜呼叫等；

● 操作间软件模块：接收顾客点菜单并发出消息通知工作人员、自动弹出所需原料的库存情况并作相应处理、更新原料库、呼叫服务台传菜等；

● 库存管理软件模块：对点菜频次统计以估计每日需要的原料库存量、入库管理、库存短缺预警等；

● 人力资源管理软件模块：人员岗位分配、自动生成执勤表、薪酬管理、员工请假处理等；

● 企业经营状况分析软件模块：统计经营的各项成本和利润，对顾客流进行分析，改善经营策略和服务质量。

4.2 多媒体餐饮管理数据库详细设计

系统所需的主要数据表及其属性：

1. 菜单数据表（Dish_Menu）

属性：代码（Dish_ID）、菜名（Dish_Name）、类别（Dish_Category）、价格(Dish_Price)、单位(Dish_Unit)、图片(Dish_Photo)、图像(Dish_Image)、简介（Dish_Introduction）；

2. 库存数据表（RawMaterial）

属性：代码（Material_ID）、原料名称（Material_Name）、库存数量(Storage_Amount)、单位(Material_Unit)、进库时间（Material_Input_Time）

3. 用料数据表（Dish_Material）

属性：菜代码(Dish_ID)、菜单位(Dish_Unit)、用料代码(Material_ID)、用量（Material_Amount）、用量单位（Material_Unit）

4. 点菜记录数据表（Dish_Order）

属性：编号（Order_Number）、台号（Table_Number）、点菜代码（Dish_ID）、点菜数量（Dish_Amount）

5. 账单数据表（Temp_Bill）

属性：编号（Bill_Number）、台号（Table_Number）、开始时间

（Start_Time）、结账时间（Checkout_Time）、消费类性（Consumption_Type）、消费金额（Consumption_Sum）、应付金额（Should_Payment）、实付金额（Actual_Payment）、支付方式（Payment_Manner）

7．歌曲数据表（Songs）

属性：歌曲代码（Song_ID）、歌曲名称（Song_Name）、歌曲类别（Song_Category）、歌曲语言（Song_Language）、歌手（Singer_Name）、歌曲数据（Song_Data）、歌手国别（Singger_Country）、歌手性别（Singer_Gender）

8．影片数据表（Movies）

属性：影片代码（Movie_ID）、影片类别（Movie_Category）、影片国别（Movie_Country）、影片语言（Movie_Language）、影片数据（Movie_DataSource）、播放时间（Play_Time）

9．员工数据表（Employee）

属性：员工编号（Employee_ID）、员工姓名（Employee_Name）、员工性别（Employee_Gender）、出生日期（Birth_Date）、教育程度（Employee_Education）、工作年限（Work_Year）、特长（Employee_Specialty）、工作岗位（Employee_Department）、上岗日期（Start_Work_Time）、月薪（Employ_Salary）、考核分数（Assess_Mark）

说明：对于图片类数据，用 IMAGE 数据类型，对于长文本，用 TEXT 数据类型，对于图像数据，由于一般所占空间较大，用 IMAGE 数据类型会带来较大 I/O 负荷，所以只用 CHAR 列保存其存放路径，在用文件系统对其读写。其他字段说明略。

4.3 各功能模块的系统结构设计

根据前面一节的需求分析及系统模块划分，我们进一步将软件功能细化。按着完成功能的不同分为：顾客软件模块、服务台软件模块、操作间软件模块、库存管理软件模块、人力资源管理软件模块和企业经营状况分

析模块等六个部分。县面分别给出各个部分的软件模块结构图，在每一部分的模块框图之后，附上了本餐饮系统的相应软件界面。

4.3.1 顾客软件模块

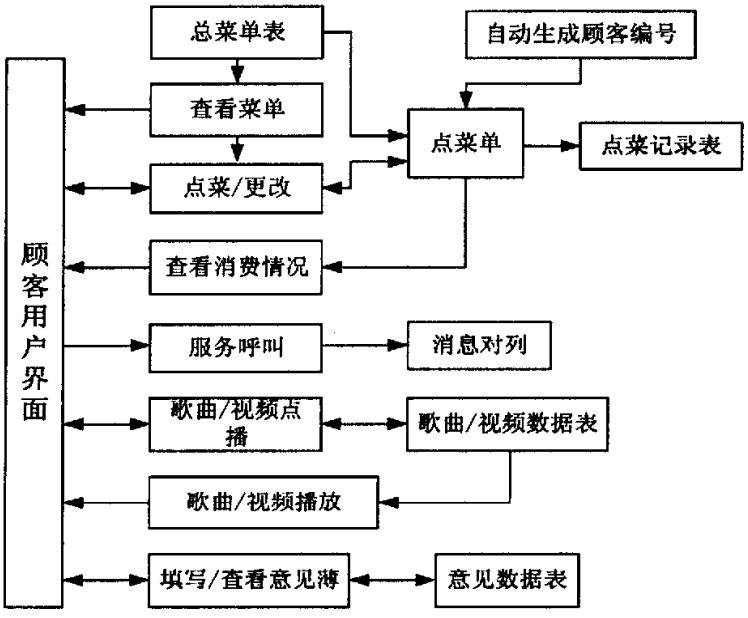

图4.4 顾客软件结构图

4.3.2 服务台软件

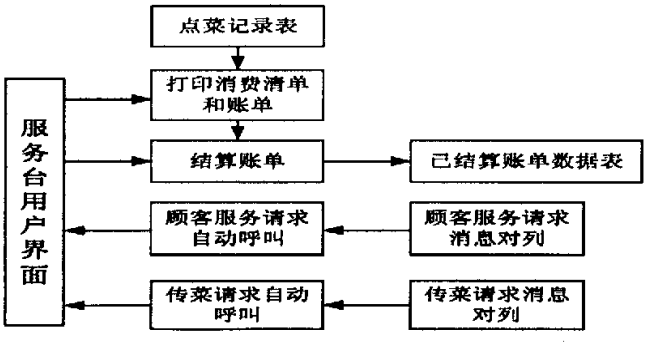

图4.5 服务台软件结构图

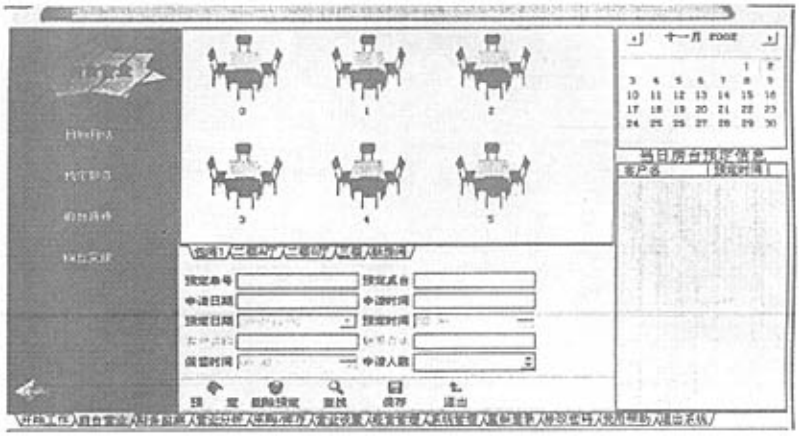

图4.6 服务台软件界面

如图4.6所示，在有关服务台的操作界面部分，有非常详细的当前餐厅作为预定状态查询、点菜纪录、结算情况（银台交接）等信息的查询和管理。

4.3.3 操作间软件

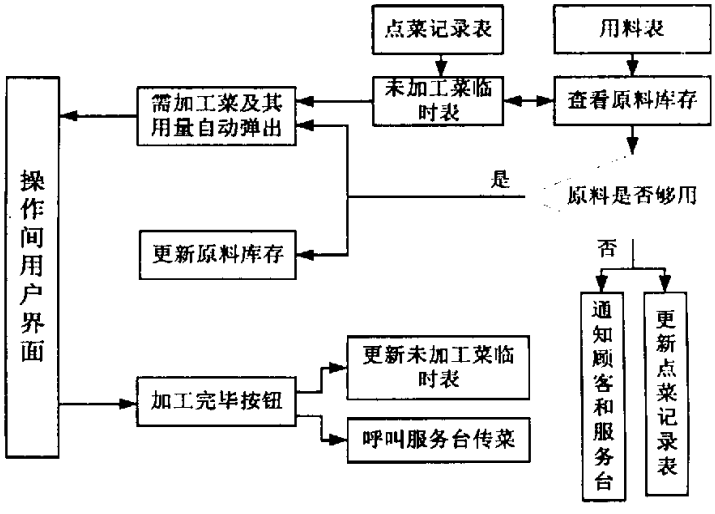

图4.7 操作间软件结构图

4.3.4 库存管理软件

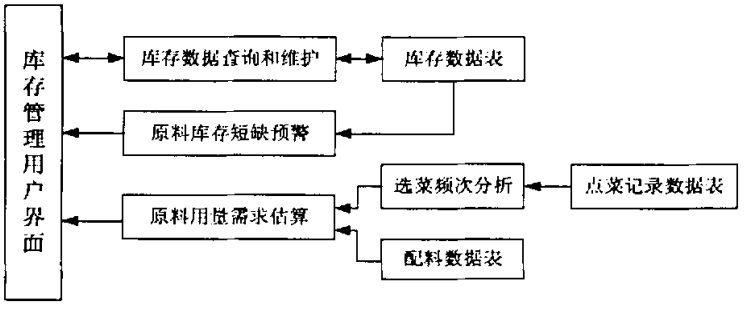

图4.8 库存管理软件结构图

4.3.5 人力资源管理软件模块

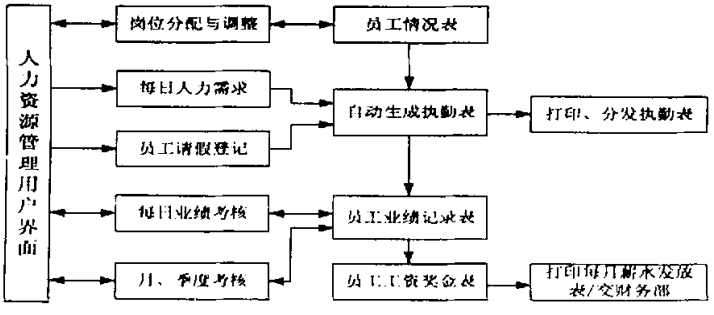

图4.9人力资源管理软件模块结构图

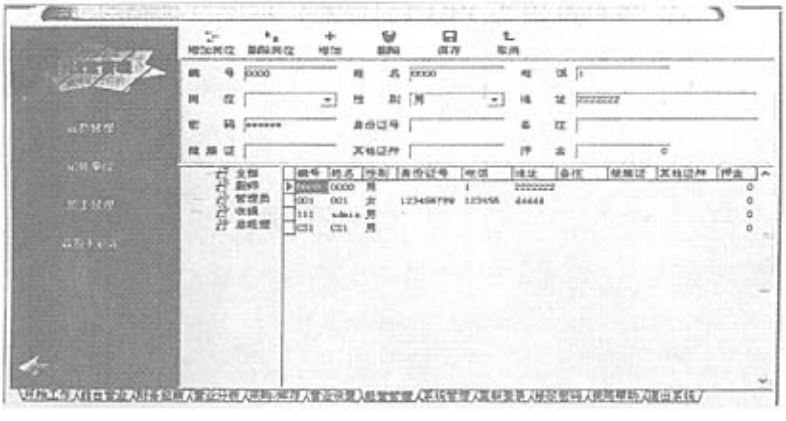

图4.10人力资源管理软件界面

在图4.10所示的经营管理界面中，可以对整个公司的人力资源进行统一管理，相关信息的查询以表格的方式给出，在显示上一目了然。

4.3.6 企业经营状况分析模块（略）

这一部分主要涉及对数据库数据进一步加工，通过各种数学分析工具，找出经营中存在的问题，明确企业经营状况，为进一步改善企业的经营管理提供依据。由于其内容较为庞杂，这里就不作详细说明。

4.4 数据库系统开发中的一些技术问题

4.4.1 本文工作开发工具的选用

目前数据库开发的可选工具很多，其中比较有优势的有微软的 Visual Basic、Visual C++等，Borland 公司的 Delphi、Borland C++等，PowerSoft 公司的 PowerBuilder 等等。其中 Delphi 作为快速应用开发工具（RAD），在数据库开发方面具有很强的优势，它具有丰富的数据库连结和数据集控制控件，特别是它的 BDE(Borland Database Engine)引擎，具有很快的连结速度。此外，在多媒体支持方面，它也具有很强的能力。

Delphi 是 Borland 公司于 1994 年底发布的用于开发数据库应用程序的工具，是一个完全导向的可视化系统开发工具，具有功能强大、运行速度快、易于使用以及开发迅速等特点，一经推出就受到广大用户的喜爱。Delphi 被称为是第四代编程语言，以它基于窗口和面向对象的编程方法，与 Windows 操作系统紧密结合。强大的数据库技术支持、迅捷的编程速度，同时兼备生动的界面和易学灵活等特点，一直为程序员们喜爱的编程工具。在 Delphi 的众多优势中，它在数据库方面的特长显得尤为突出：适应于多种数据库结构；从客户机/服务器模式到多层数据库结构模式；高效率的数据库管理系统和先进的数据库引擎；先进的数据分析手段和提供大量的企业组件。

目前具有两类版本：Delphi 的标准版本和企业版。标准版本包含一个 Borland Database Engine 的局部拷贝，它允许用户创建能访问 dBASE、

Paradox 和 Local InterBase 服务器的数据库应用，它还支持具有 ODBC 接口的数据库。Delphi 的 C/S 版本包括 Borland SQL Link，它能直接访问 ORACLE、SyBase 和 Microsoft SQL Server，Informix 以及 InterBase 数据库服务器。

Delphi 可以访问多种数据库管理系统的数据库，凭借窗体(Forms)和报表(Reports)，BDE(Borland Database Engine)可以访问诸如 Paradox、dBASE、本地 InterBase 服务器的数据库，也可以访问远程数据库服务器上的数据库(如 ORACLE、Sybase SQL Server、Informix 等客户/服务器数据库中的数据库)，或任何经 ODBC(开放式数据库联接，Open Database Connectivity) 可访问的数据库管理系统中的数据库。

下面是几种频繁使用的数据库控件：

1）数据库访问部件（Data Access）：Datasource、Table、Query、Storedproc 等。

2）数据控制部件（Data Control）：DBGrid、DBNavigator、DBText、DBEdit、DBImage 等各种数据显示控件。

3）QReport 部件：它包含了报表设计时要用到的多种控件，报表的设计和制作在该系统中使用也比较多。

4）还有 Servers 部件，如 Wordapplication、Worddocument 等。

由于是针对数据库编程的，所以编程的总思想就是，前台用于接收和发送参数，浏览数据记录、统计结果、打印等等。后台主要完成数据库的关联查询、统计运算、数据存储和更新、返还结果等。

在视频播放方面，Delphi 的组件板 System 页上有一个称为 MediaPlayer 的组件，它是多媒体软件制作的核心。

MediaPlayer 组件主要是用于控制 MCI（Media Control Interface）设备，该组件提供了一系列按钮用于控制诸如 CD－ROM、MIDI、VCR 等多媒体设备，这些多媒体设备可以是硬件也可以是软件。这些按钮在缺省情况下

从左到右依次为：Play（播放）、Pause（暂停）、Stop（停止）、Next（下一个曲目）、Prev（前一个曲目）、Step（步进）、Back（后退）、Record（录音）、Eject（弹出媒体）等 9 个功能按钮。在用户自己开发的程序中，多媒体设备可以有两种方法实现以上 9 个功能，一种方法是将 MediaPlayer 组件放在窗体上，在程序运行时单击相应按钮；另一种方法是用与按钮相应的"方法"（Method）来实现这些功能。

此外，Delphi 还完全支持 OLE(Object Linking & Embedding 对象连结与嵌入)和 ActiveX，可以很容易的安装其它的多媒体播放插件。

4.4.2 TEXT/IMAGE 字段的存取技术

Delphi 提供了数据访问（Data Access）和数据控制（Data Controls）的可视化控件，能够方便快捷地产生具有良好界面且功能强大的数据库应用程序。对于涉及文本/图像数据（含 TEXT/IMAGE 字段）的数据库应用程序，图像数据的存取技术是一个关键。

a. 图像数据的保存

（1）建立窗体，设置窗体中各控件的属性。

该窗体的主要功能是将信息进行编辑和保存。在窗体需要放置 TDatasuorce、Ttable 控件。需要注意的是图像保存所用的图像框必须用 TImage 而不能用 TDBImage，文本编辑框宜用 Tedit 而不宜用 TDBEdit，这一点与图像的读取恰好相反。

其中，各主要控件的属性设置如下：

Datasource1.Dataset:Table1；

Table1.Databasename:RGMISDatabase；

Table1.Tablename:Dish_Menu；

Table1.Active:True；

其他诸如 Caption 之类的属性设置不再叙述。

（2）数据处理程序的建立

1）图像数据打开的处理

```
ProcedureTForm1.PictopenbtnClick（Sender： TObject）;
Beginopendialog1.Execute;
Image1.Picture.Loadfromfile ;
End;
```

2） 图像保存的处理

图像保存的处理程序完成把在窗体中所编辑的信息包括图像保存到相应的数据库中，其关键是要定义一个 Graphic 类型的变量且该变量要用 assing（）函数传递到相应数据库中保存。具体程序如下：

```
ProcedureTform1.SavebtnClick（Sender:TObject）;
var Graphic1:TGraphic;
var Image1:TGraphic;
var Text1:TText;
Begingraphic1:Tgraphic.Create;
Graphic1.Loadfromfile（Opendialog1.Filename）;
Table1.Insert;
Table1.Fieldbyname('Dish_ID').Astring:Tedit1.Text;
Table1.Fieldbyname('Dish_Name').Astring:Tedit2.Text;
Table1.Fieldbyname('Dish_Category').Asfloat:Tedit3.Text;
Table1.Fieldbyname('Dish_Price').Asfloat:Tedit4.Text;
Table1.Fieldbyname('Dish_Unit').Asfloat:Tedit5.Text;
Table1.Fields[5].Assign(Graphic1);
Table1.Fields[6].Assign(Image1);
Table1.Fields[7].Assign(Text1);
Table1.Post;
```

```
Graphic1.Free;
End;
```

b. 图像数据的读取

建立窗体，放置 TTable、TDatasource、TDBEdit、TDBImage 控件，设置各控件的属性。

Table1 和 Datasource1 的属性的设置与数据的保存部分相同，所不同的是数据库数据的读取时用 TDBEdit 和 TDBImage 控件而不用 TEdit 和 TImage。控件 TDBEdit 和 TDBImage 只要将 DataField 属性设置为其相对应的域；TDBNavigator 的 Datasource 属性设置为 Datasource1 即可。

4.4.3 视频数据的播放

在建立的多媒体餐饮管理系统中，为减少存储空间和网络传输负荷，可将视频文件压缩为 real 形式（后缀名为.ra、.rm 或.ram），这种压缩形式具有体积很小、适合网络传输的特点。

在该系统中，利用数据表保存了视频文件的存放路径及问及文件名等信息。在客户端，可以用 Delphi 开发相应的播放窗口，先获取视屏文件信息，然后再选用相应的控件进行播放。

Delphi 支持 ActiveX，可以很方便地创建、注册、安装、发布和使用 ActiveX 控件、ActiveForm 和 OLE 自动化对象。利用 Delphi 开发 real 视频文件的播放程序的具体步骤为：

1．在开发工作站上安装 RealPlayer;

2．在 Delphi 的可视化编程环境中，选择 File 菜单下的"New Application"项，新建一个新的应用程序。然后选择 Component（组件）菜单下的"Import ActiveX Control..."（导入 ActiveX 控件）选项，选中其中的"Real Player ActiveX Control Library(Vision1.0)"项，并单击 Install，会出现一个 install 窗口。如下图：

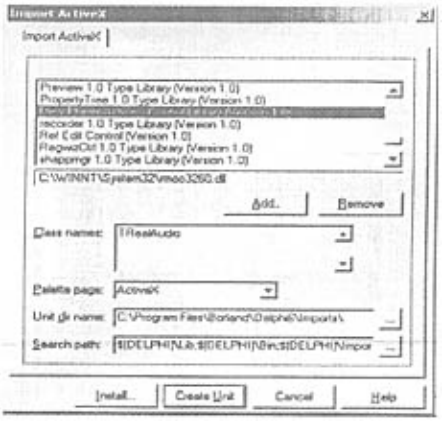

图4.11. RealAudio组建的安装

3．把它添加到一个新建的包中，在"Into new package"对话框中的"File name:"窗口中选择你想安装的路径，并新建一个包，假如命名为MoviePlay，并单击"OK"；它询问"Package test.bpk will be built. Continue?"，单击"Yes"。 在弹出的窗口中单击击"Install"，该控件就安装完毕；

4．此时，会在控件条的 ActiveX 下发现一个新的控件，名字为RealAudio， 单击它把它放在 Form 中，并在该 Form 中放一个 panel，在该 panel 上放六个 button，他们的 Caption 属性分别命名为 "播 放"、"全 屏"、"暂 停"、"停 止"、"静音"、"退出"，并依次设置这些按钮的 Name 属性"PlayButton"、"FullScreenButton"、"PauseButton"、"StopButton"、"MuteButton"和"QuitButton"；

5．视屏播放部分程序代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);//将从 Movies 表中获得的
begin                                          //视频文件名及路径赋给
    Realaudio1.SetSource
```

```
                (MoviesTable.FieldByName('Movie_DataSource').asString);
    end;                                                    //RealAudio1

procedure TForm1.PlayButtonClick(Sender: TObject);
begin
  if RealAudio1.CanPlay then
  begin
    RealAudio1.DoPlay;
    PlayButton.Enabled:=False;
    PauseButton.Enabled:=True;
    StopButton.Enabled:=True;
  end
end;

procedure TForm1.FullScreenButtonClick(Sender: TObject);
begin
  RealAudio1.SetFullScreen;
end;

procedure TForm1.PauseButtonClick(Sender: TObject);
begin
  if RealAudio1.CanPlayPause then
  begin
    RealAudio1.DoPlayPause;
```

```
    PauseButton.Enabled:=False;
    PlayButton.Enabled:=True;
  end
end;

procedure TForm1.StopButtonClick(Sender: TObject);
begin
  if RealAudio1.CanStop then
  begin
    RealAudio1.DoStop;
    StopButton.Enabled:=False;
    PauseButton.Enabled:=False;
    PlayButton.Enabled:=True;
  end
end;

procedure TForm1.MuteButtonClick(Sender: TObject);
var pbMute:wordbool;
begin
  pbMute:=RealAudio1.GetMute;
  RealAudio1.SetMute(not pbMute);
  if pbMute then
     MuteButton.Caption:='静      音'
  else
```

```
    MuteButton.Caption:='开　　音';
end;

procedure TForm1.QuitButtonClick(Sender: TObject);
begin
  RealAudio1.FreeOnRelease;
  Close;
end;
```

视频文件播放界面：

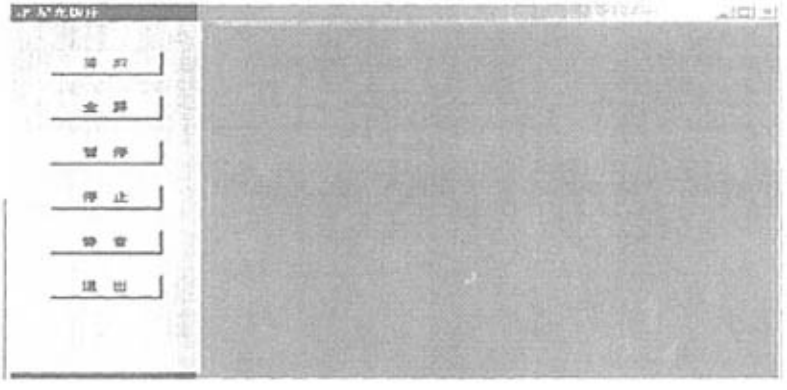

图4.12. 视频播放界面

4.4.4 本系统用户操作方式方面的考虑

本文所开发的多媒体餐饮信息管理系统由于其自身的特点在用户操作方式、用户界面方面应进一步的采用易操作、方便、直观的操作方式。使用户能够方便的进行相应的操作。因此除了为管理员设计较为专业、功能全面的管理界面以外，针对用户操作的终端部分，我们将操作进一步简化：设想用户利用触摸屏即可完成点菜等简单操作，并在界面中加入相应操作的向导及解释，使用户能够很快的了解相应的操作方法，这方面仍可做大量的工作。

5 结束语

多媒体信息系统涉及很多十分前沿、复杂的理论和技术，例如基于内容的检索、视频和图像内容的表示、超媒体、面向对象的数据建模等等都是目前比较前沿的课题。本文的工作重点是研究与实现一个基本的多媒体管理信息系统，对上述问题并没有进行详细的研究和讨论。在论文中，主要探讨了基于关系型数据库的多媒体管理系统的基本实现方法，其中涉及到多媒体数据在关系数据库中的存取和更新、数据库的性能优化和安全性设计、视频数据在客户端的播放等技术。本文也实现了一个基本的多媒体管理信息系统——多媒体餐饮管理系。在系统实现过程中，按照需求分析→总体设计→各功能模块的设计→编码→调试的基本设计步骤。在设计阶段采用了结构化的设计方法。对于数据库的设计，由于篇幅有限，只讨论了系统所需的主要数据表及其属性。系统的开发工具使用的是 Borland Delphi5.0，数据引擎使用的是 BDE，它具有速度快、方法丰富、使用方便的特点。系统通过功能、性能测试基本能满足要求。系统也经过用户的使用，反映界面友好，使用方便，提高了公司的经济效率。

对于像多媒体餐饮管理系统等这样的信息系统来讲，利用关系型数据库进行数据管理，将多媒体数据作为某个表的某个字段，利用 BDE、ODBC、ADO 等数据库引擎进行数据访问和维护等这些技术是够用的，而且还利用了关系型数据库的优点。这也是本文讨论的主题。然而对于更高级的应用—如地理信息系统、军事情报信息系统等这样一些需要图像、声音内容理解和基于内容的查询，还需要更高层次的技术需求，这些技术也正在研究和发展之中。

致 谢

我在西安理工大学学习期间，受到了西安理工大学和自动化学院各级领导的热情关心和支持，得到了所有授课老师的言传身教，使自己拓宽了知识视野，提升了专业水平。在此，我特别向理工大学潘永湘教授、张景教授、李守智教授以及所有授课老师致以最诚挚的感谢。

本文的研究工作是在我的导师王新房教授的亲切关怀和悉心指导下完成的。王新房老师为我提供了良好的科研条件，并给予论文理论工作以方向性的指导。在整个研究生期间，王新房老师始终给予我亲切的关怀和指导，从导师那里，我不仅学到了专业知识，而且学到了严谨的治学态度、认真的工作作风和不断进取的精神。为此，我谨向我的导师王新房教授表示最衷心的感谢。

吴 彬 琦

2003/03

参考文献

[1] [美]W.L.Crosky 等《多媒体信息管理技术手册》-1 版 -北京 科学出版社 1998.11

[2] 萨师煊 王珊 《数据库系统概论》 -2 版 -北京 高等教育出版社 1991.4

[3] [印度]Abraham Silberschatz、Henry F.Korth、S.Sudarshan 《Database System Concepts》(英文) -3 版 -北京 机械工业出版社 1999.3

[4] 郑人杰 殷人昆 《软件工程概论》 -1 版 -北京 清华大学出版社 1998.4

[5] Rondald J. Norman 《Object-Oriented Systems Analysis and Design》(英文) -1 版 -北京 清华大学出版社 1998.6

[6] 魏军 王威 郭冰冰 《管理信息系统》-1 版 -北京 国防工业出版社 1999.1

[7] [美]Microsoft 《网络数据库系统管理 Microsoft SQL Server6.0》 -1 版 -北京 科学出版社 1997.7

[8] [美]Robert D.Schneider《规划与建立高性能 SQL Server6.5 数据库》-1 版 -北京 机械工业出版社 1997.11

[9] [美]微软公司 《Microsoft SQL Server7.0 实现数据库设计》 -1 版 -北京 北京希望电子出版社 1999.5

[10] [美]微软公司 《Microsoft SQL Server7.0 系统管理》 -1 版 -北京 北京希望电子出版社 1999.5

[11] 刘韬 肖永顺 王宇 《Delphi4.0 数据库编程》 -1 版 -北京 人民邮电出版社 1999.9

[12] [美]Marco Cantu、Tim Gooch、John F.Lam 《Delphi 高级开发指南》-1 版 -北京 电子工业出版社 1998.8

[13] SQL SERVER 7 开发人员指南 希望图书创作室 1999-12-1 北京希望电子出版社

[13] SQL SERVER 数据库管理系统 刘杰 1999-4-1 中国水利电力出版社

[14] SYBASE SQL SERVER 11 性能及其优化技术 夏洪山 1998-7-1 北京希望电子出版社

[15] SQL Server 7.0 开发指南 张蓉 1999-10-1 电子工业出版社
[16] SQL Server 6.5 技术内幕 姜鸿英 1999-2-1 清华大学出版社
[17] SQL Server 7.0 数据库系统管理与应用开发 袁鹏飞 1999-5-1 人民邮电出版社
[18] SQL Server 7.0 程序设计超级管理篇 陈宗兴 2000-7-1 中国铁道出版社
[19] SQL Server 7.0 程序设计管理与应用篇 陈宗兴 1999-10-1 中国铁道出版社
[20] 规划与建立高性能 SQL Server 6.5 数据库 李小坚 1997-11-1 机械工业出版社数据库技术及开发教程 常明华 2000-3-1 电子工业出版社
[21] Informix 关系数据库 钟显红 1998-9-1 电子工业出版社关系数据库理论 马垣 1999-4-1 清华大学出版社
[22] 怎样浏览与查询数据库 李超 2000-11-1 人民邮出版社
[23] 分布式数据库管理系统实现技术 周龙骧 1998-7-1 科学出版社
[24] 并行关系数据库管理系统引论 李建中 1998-7-1 科学出版社
[25] Delphi 5 实战与精通（实战篇） 唐健 2000-5-1 清华大学出版社
[26] 关系数据库系统 Delphi 4.0 及其应用 张健沛 1999-4-1 中国水利电力出版社
[27] Delphi 5.0 数据库应用开发 曾令友 2000-4-1 中国水利电力出版社
[28] 关系数据库系统 Delphi 4.0 及其应用 张健沛 1999-4-1 中国水利电出版社
[29] 深入 Delphi 的多媒体编程 吴克河 2000-1-1 中国电力出版社
[30] Delphi 5 应用程序设计实例 王小华 2000-3-1 电子工业出版社
[31] 多媒体图像技术及应用 刘富强 2000-11-1 人民邮电出版社
[32] 视频软件插件技术与制作实例(附盘) 李瑞芳 2000-1-1 北京科海出版社
[33] 多媒体技术 黄孝建 2000-11-1 北京邮电出版社
[34] 多媒体网络通信技术与应用 朱秀昌 1998-2-1 电子工业出版社
[35] 多媒体图像技术及应用 杨俭 2000-12-1 人民邮电出版社

附录：数据列表

菜单数据表（Dish_Menu）	
Dish_ID	代码
Dish_Name	菜名
Dish_Category	类别
Dish_Price	价格
Dish_Unit	单位
Dish_Photo	图片
Dish_Image	图像
Dish_Introduction	简介

库存数据表（RawMaterial）	
Material_ID	代码
Material_Name	原料名称
Storage_Amount	库存数量
Material_Unit	单位
Material_Input_Time	进库时间

用料数据表（Dish_Material）	
Dish_ID	菜代码
Dish_Unit	菜单位
Material_ID	用料代码
Material_Amount	用量
Material_Unit	用量单位

点菜记录数据表（Dish_Order）	
Order_Number	编号
Table_Number	台号
Dish_ID	点菜代码
Dish_Amount	点菜数量

账单数据表（Temp_Bill）	
Bill_Number	编号
Table_Number	台号
Start_Time	开始时间
Checkout_Time	结账时间
Consumption_Type	消费类型
Consumption_Sum	消费金额
Should_Payment	应付金额
Actual_Payment	实付金额
Payment_Manner	支付方式

歌曲数据表（Songs）	
Song_ID	歌曲代码
Song_Name	歌曲名称
Song_Category	歌曲类别
Song_Language	歌曲语言
Singer_Name	歌手
Song_Data	歌曲数据
Singger_Country	歌手国别
Singer_Gender	歌手性别

影片数据表（Songs）	
Movie_ID	影片代码
Movie_Category	影片类别
Movie_Country	影片国别
Movie_Language	影片语言
Movie_DataSource	影片数据
Play_Time	播放时间

员工数据表（Employee）	
Employee_ID	员工编号
Employee_Name	员工姓名
Employee_Gender	员工性别
Birth_Date	出生日期
Employee_Education	教育程度
Work_Year	工作年限
Employee_Specialty	特长
Employee_Department	工作岗位
Start_Work_Time	上岗日期
Employ_Salary	月薪
Assess_Mark	考核分数