项目 3 网站数据库的建设与使用

数据库是一个网站的核心，对于网站而言，数据库有着提供数据支持和用户管理等非常重要的作用。因而，学习数据库知识在网站建设中是必不可少的。本节内容将向读者介绍数据库技术的基本知识，使读者了解数据库技术的常用概念，为本书后面知识的学习打下基础。

任务 1　数据库基础知识

子任务 1　什么是数据库

随着计算机技术、通信技术和网络技术的飞速发展，信息系统渗透到社会各个领域，作为其核心的数据库技术更是得到了广泛的应用。数据的建设规模、数据库的信息量大小及使用频度已成为衡量一个国家信息化程度的重要标志。

从性质上讲，数据库就是存储信息的工具，是依照某种数据模型组织起来的并存放在存储器中的数据集合。这种数据集合具有如下特点：

（1）尽可能不重复，以最优方式为某个特定组织的多种应用服务；

（2）对数据的增、删、改和检索由统一软件进行管理和控制。从发展的历史看，数据库是数据管理的高级阶段，它是由文件管理系统发展起来的。

数据库技术是随着数据管理的需要而产生的。数据管理指的是对数据的分类、组织、编码、存储、检索和维护，它是数据处理的核心。随着计算机硬件技术和软件技术的发展，数据管理经历了如下三个阶段：

①人工管理

②文件系统处理

③数据库管理

当人们收集了大量的数据后，应该把它们按照一定格式保存起来以便进一步处理。随着社会发展和数据量急剧增长，现在人们就借助计算机和数据库技术科学地保存大量的数据，以

便能更好地利用这些数据资源。自此，数据库便成为计算机领域的常用技术。

所以，从数据库的特点和应用上理解，数据库是指“长期储存在计算机内的、有组织的、可共享的数据集合”。

在 Web 应用程序只能中，需要显示各种各样的信息，而这些显示的信息是以数据库中存储的数据为基础的。数据库技术和动态站点开发技术相互结合，为广大用户奉献了丰富多彩的 Web 页面。

数据库包含关系数据库、面向对象数据库及新兴的 XML 数据库等多种，目前应用最广泛的是关系数据库。

子任务 2　数据库的常用概念

本部分内容将向读者介绍数据库技术几个常用的概念，以加深读者对数据库技术的理解。

1. 数据库管理

数据库管理是有关建立、存储、修改和存取数据库中信息的技术，是指为保证数据库系统的正常运行和服务质量，有关人员进行的技术管理工作。负责这些技术管理工作的个人或集体称为数据库管理员（DBA）。数据库管理的主要内容有数据库的建立、数据库的调整、数据库的重组、数据库的重构、数据库的安全控制、数据库的完整性控制和对用户提供技术支持。

2. 数据库

数据库是长期存储在计算机内有组织的大量共享的数据集合，它可以提供各种用户共享且具有最小冗余数和较高的数据与程序独立性。

3. 数据模型

数据模型是现实世界数据特征的抽象，是数据技术的核心和基础。它是数据库系统的数学形式框架，是用来描述数据的一组概念和定义，主要包括如下方面的内容：

（1）静态特征：对数据结构和关系的描述。

（2）动态特征：在数据库上的操作，例如添加、删除和修改。

（3）完整性约束：数据库中的数据必须满足规则。

4. 概念模型

概念模型用于信息世界的建模，人们常常先将信息世界抽象为信息世界，然后讲信息世界转换为机器世界，而概念模型是现实世界到机器世界的一个中间层次。

概念模型是对信息世界的建模，可以用 E-R 图来描述概念模型。E-R 图提供了表示实体型、属性和联系的方法。

实体型：用矩形表示，框内写实体名称。

属性：用椭圆表示，框内写属性名称。

联系：用菱形表示，框内写联系名称。

例如，如图 3-1 所示为实体－属性图的一个实例。

如图 3-2 所示为实体－联系图的一个实例。

5. 数据模型

不同的数据模型具有不同的数据结构。目前最为常用的数据模型有层次模型、网状模型、关系模型和面向对象数据库。其中层次模型和网状模型统称为非关系模型。

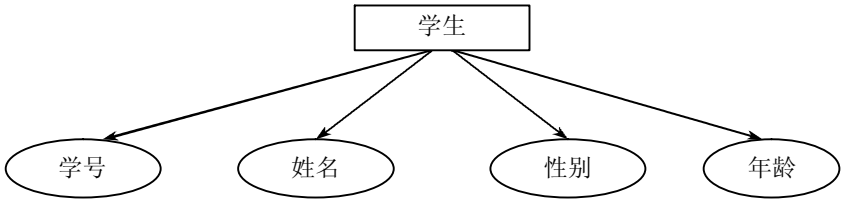


图 3-1　实体－属性图

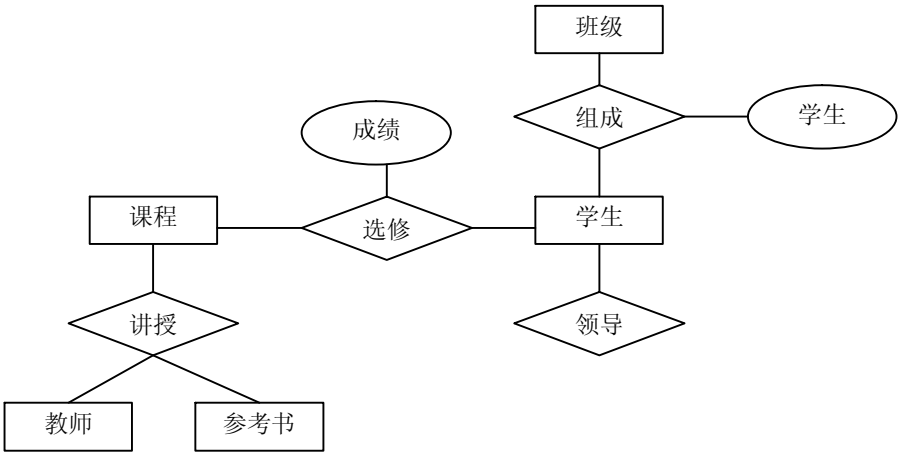


图 3-2　实体－联系图

概念模型是从用户的角度对数据和信息进行建模，而数据模型是从计算机系统的角度对数据进行建模。

6. 关系数据模型

关系数据模型是当前应用最广泛的一种模型。关系数据库都采用关系模型作为作为数据的组织方式。自从 20 世纪 80 年代以来，计算机厂商推出的数据库管理系统几乎都支持关系模型。

关系模型的基本要求是关系必须要规范，即要求关系模型必须满足一定的规范条件，关系的分量必须是一个不可再分的数据项。

7. 数据库系统的结构

设计数据库时，强调的是数据库结构；使用数据库时，关心的是数据库中的数据。从数据库系统角度看，数据库系统通常采用三级模式结构，这是数据库管理系统的内部系统结构。

数据库系统的三级模式结构是指数据库系统由外模式（物理模式）、模式（逻辑模式）和内模式三级抽象模式构成，这是数据库系统的体系结构或总结构。上述具体结构如图 3-3 所示。

8. 数据库管理系统

数据库管理系统即 DBMS，是为数据库的建立、使用和维护而配置的软件。它建立在操作系统的基础之上，是位于操作系统与用户之间的一层数据管理软件，负责对数据库进行统一的管理和控制。

数据库管理系统的功能主要包括 6 个方面，即数据定义、数据操作、数据库运行管理、数据组织、存储和管理、数据库的建立和维护、数据通信接口。

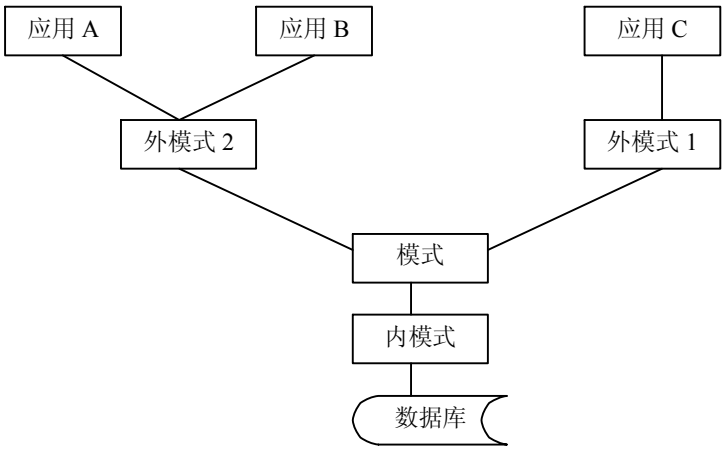


图 3-3　三级模式结构图

任务 2　创建 Access 数据库

本书后面讲解的实例都是基于 Access 数据库的，在本节的内容中，将对 Access 数据库的基本知识进行简要介绍。

子任务 1　Access 概述

Access 是微软 Office 工具中的一种数据库管理程序，可赋予更加的用户体验，并且具有导入、导出和处理 XML 数据文件等功能。

Microsoft Access 是一种关系式数据库，有一系列表组成。表由一系列行和列组成，每一行是一个记录，每一列是一个字段，每个字段有一个字段名，字段名在一个表中不能重复。

Access 数据库由如下 6 种对象组成：

（1）表（Table）：是数据库的基本对象，是创建其他 5 种对象的基础。表由记录组成，记录由字段组成，表用来存储数据库的数据，故又称数据表。

（2）查询（Query）：可以按索引快速查找需要的记录，按要求筛选记录并能连接若干个表的字段组成新表。

（3）窗体（Form）：也称之为表单，它提供了一种方便浏览、输入及更改数据的窗口；还可以创建子窗体显示相关联的表的内容。

（4）报表（Report）：功能是将数据库中的数据分类汇总，然后打印出来，以便分析。

（5）宏（Macro）：它相当于 DOS 中的批处理，用来自动执行一系列操作。Access 列出了一些常用的操作供用户选择，使用起来十分方便。

（6）模块（Module）：其功能与宏类似，但它定义的操作比宏更精细和复杂，用户可以根据自己的需要编写程序。

Access 适用于小型商务活动，用以存储和管理商务活动所需要的数据。Access 不仅是一个数据库，而且具有强大的数据管理功能，它可以方便地利用各种数据源，生成窗体（表单）、查询、报表和应用程序等。再利用 ASP 开发小型项目时，Access 往往是首先被考虑的数据库

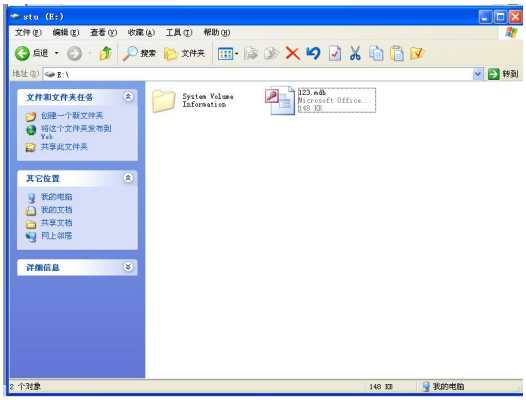
工具。它以操作简单、易学易用的特点而受到大多数用户的青睐。

子任务 2　启动和关闭 Access

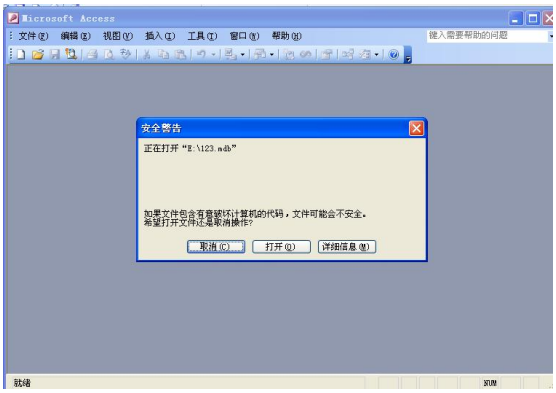
在个人机器上安装了 Office 后就可以使用 Access 数据库了。Access 数据库的启动和关闭十分简单，下面将对其进行简要介绍。

1. 启动 Access

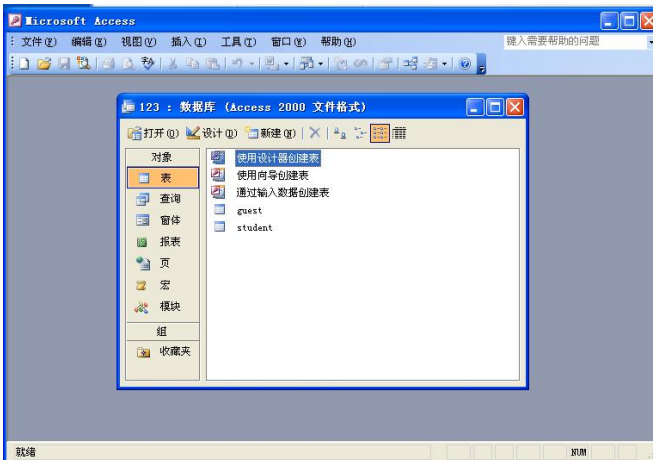
启动 Access 数据库的流程如图 3-4 所示。



鼠标双击要启动的数据库文件图标



在弹出的安全警告窗口中单击“打开”按钮



启动后的 Access 数据库界面

图 3-4　Access 启动流程图

2. 关闭 Access

关闭 Access 数据库的方法十分简单，只需要单击界面右上角的关闭图标即可。

注意

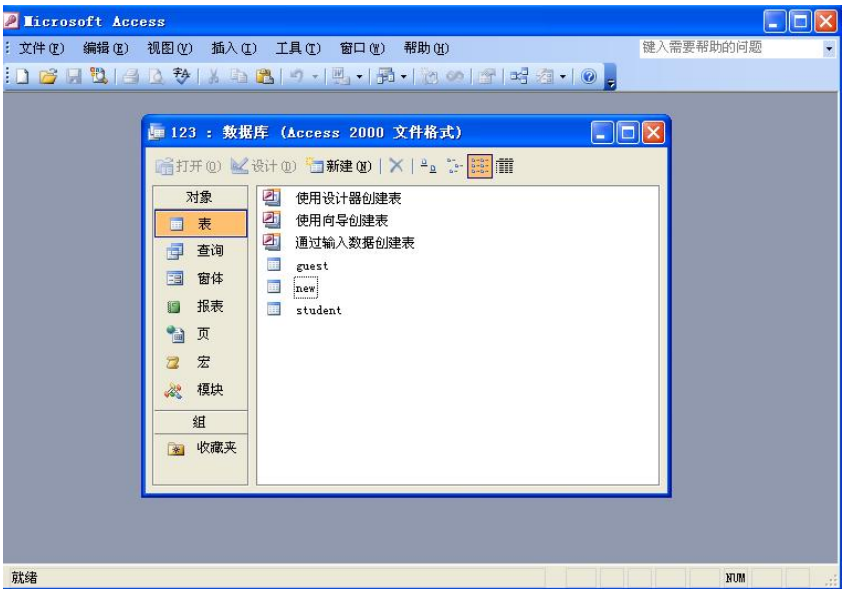
启动 Access 时的安全警告对话框时 Office 关于“不安全表达式“的安全警告。对普通用户而言，可以将其关闭，这样就不会遇到类似的提示问题了。具体操作方法如下：启动 Access，在“工具“菜单上依次选择“宏”、“安全性”，单击“安全级”选项卡，然后单击“低”，最后单击“确定”按钮，重新启动 Access 即可。

子任务 3　Access 的基本操作

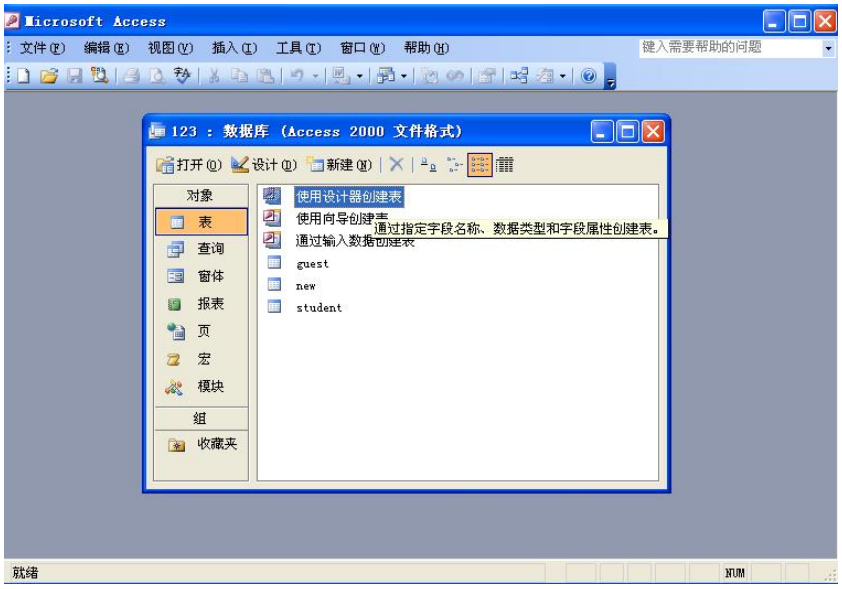
数据库最重要的功能是保存数据，Access 数据库中的数据保存在它的表面里。下面对 Access 中表的操作进行简要介绍。

1. 新建表

新建表即在库中创建一个新的表，具体操作流程如图 3-5 所示。

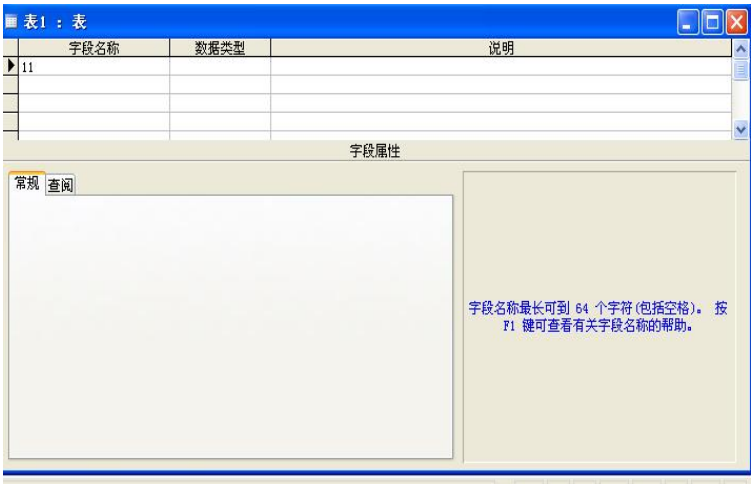


（一）启动操作数据库后单击“表”选项



（二）在弹出界面中单击“使用设计器创建表”选项

图 3-5　新建表流程图



（三）在弹出界面依次输入字段名、数据类型和属性条件，单击保存按钮

（四）在弹出的“另存为”对话框中输入表的名称，然后单击“确定”按钮

（五）在弹出的尚未定义主键界面中单击“是”按钮，完成此表的创建

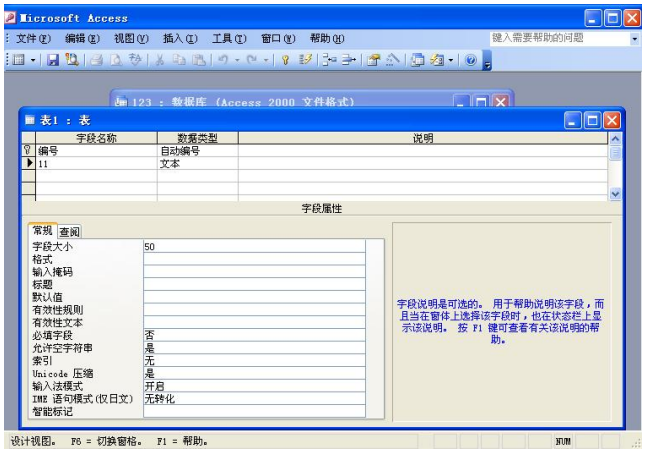
图 3-5　新建表流程图（续）

2. 修改表

可以对已创建的 Access 数据库表进行修改，其具体操作流程如图 3-6 所示。

（一）在修改的表上右击“设计视图”选项

（二）在弹出的表属性界面中对此表数据、数据类型和属性进行修改

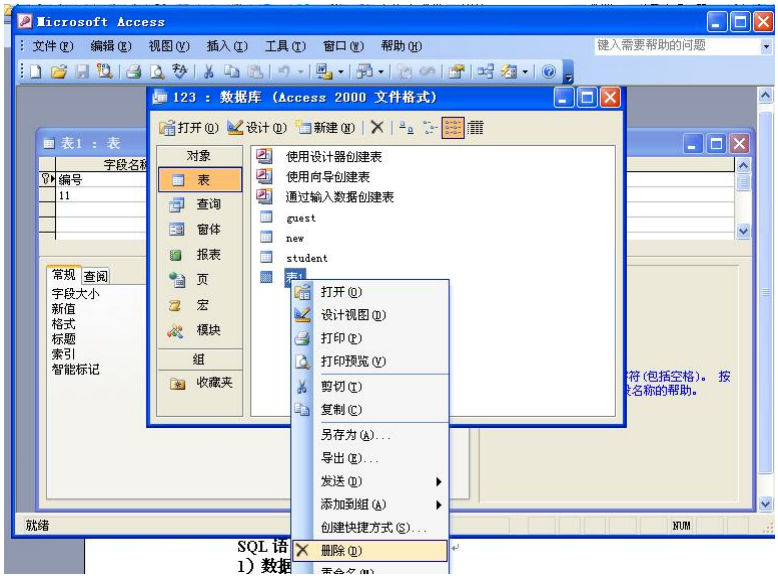


（三）单击“保存”按钮后即完成修改

图 3-6　修改表操作流程

3. 删除表

根据系统的具体需要，可以删除已创建的 Access 数据库表，其具体操作流程如图 3-7 所示。



（一）在要操作的表上右击“删除”选项

（二）在弹出的删除确认界面中单击“是”按钮将此表删除

图 3-7　删除表操作流程

任务 3　SQL 语言操作数据库

SQL 又称为结构化查询语言，1986 年 10 月美国国家标准局确立了 SQL 标准，1987 年，国际标准化组织也通过了这一标准。自此，SQL 成为了国际标准语音，所以各个数据库厂家

纷纷推出各自支持的 SQL 软件或接口软件。

SQL 成为国际标准，对数据库以外的领域也产生了很大的影响，有不少软件产品讲 SQL 语言的数据查询功能与图形功能、软件工程工具、软件开发工具、人工智能程序结合起来。SQL 已经成为了关系数据库领域中的一个主流语言。

SQL 语言主要具有如下功能：

1）数据定义

2）数据操作

3）视图

在下面的内容中，将对 SQL 的上述基本功能的实现进行简要介绍。

子任务 1　数据定义

关系数据库是由模式、外模式和内模式构成的，所以关系数据库的基本对象是表、视图和索引。因此 SQL 的数据定义功能包括定义表、定义视图和定义索引。

1. 数据库操作

数据库是一个包括了很多基本表的数据集，使用 SQL 创建数据库的语句格式如下：

Create database<数据库名>（其他参数）

其中，“数据库名”在系统中必须是唯一的，不能重复，不然将导致数据存取失误；“其他参数”因具体数据库实现系统不同而异。

例如，可以通过如下语句建立一个名为 manage 的数据库。

Create database manage;

将数据库及其全部内容从系统中删除的语法格式如下：

Drop database <数据库名>;

例如，可以通过如下语句删除前面创建的数据库 manage：

Drop database manage;

2. 表操作

表是数据库中的最重要构成部分，可以存储大量的数据。

（1）创建表。SQL 语言使用 Create table 语句定义基本表，其具体语法格式如下：

Create table <表名>;

例如，创建一个职工表 zhigong，它由职工编号 id、姓名 name、性别 sex、年龄 age 和部门 dept 五个属性组成。具体实现代码如下：

```
Create table zhigong
   (id char(5),
Name char (20)
Sex char (1)
Age int,
Dept char(15));
```

上述代码中的 char()和 int 是这些属性的数据类型。

（2）修改表。随着应用环境和应用需求的变化，有时需要修改已经建立好的表。其具体语法格式如下：

Alter table<表名>

(Add<新列名><数据类型>(完整性约束))

(Drop<完整性约束名>)

(Modify<列名><数据类型>);

其中，“表名”是指要修改的表，“Add”向表内添加新列和新的完整性约束条件，“Drop”子句用于删除指定的完整性约束条件，“Modify”子句用于修改原有的列定义。

例如，通过如下语句向表 zhigong 增加工作时间列“shijian”，数据类型为日期型。

Alter table zhigong ADD shijian Date;

（3）删除表。当删除某个不再需要的表时，可以使用 SQL 语句中的 Drop table 语句进行删除。其具体语法格式如下：

Drop table<表名>

例如，通过如下语句可以删除表 zhigong：

Drop table zhigong;

使用 Drop table 命令时一定要小心。一旦一个表被删除之后，将无法恢复。

在建立一个站点时，很可能需要向数据库中输入测试数据。而当将这个站点推出时，需要清空表中的这些测试信息。如果想清除表中的所有数据但不删除这个表，可以使用 Truncate table 语句。例如，如下代码从表 zhigong 中删除所有数据。

Truncate table mytable

3. 索引操作

建立索引是加快表的查询速度的有效手段。读者可以根据个人需求在基本表上建立一个或多个索引，从而提高系统的查询效率。建立和删除索引是由数据库管理员或表的属性负责完成。

（1）建立索引。在数据库中建立索引的语法格式如下：

Create(unique | fulltext | spatial)index index_name

(using index_type)

On tb1_name (index_col_name,…)

Index_col_name:

Col_name ((length))(asc | desc)

Create index 被映射到一个 Alter table 语句上，用于创建索引。通常，当使用 Create index 创建表时，也同时在表中创建了所有的索引，Create index 允许向已有的表中添加索引。

例如，通过如下语句为表 zhigong 建立索引，按照职工号升序和姓名降序建立唯一索引。

Create unique index no_index on zhigong(id asc,name desc);

（2）删除索引。通过 Drop 子句可以删除已经创建的索引，具体语法格式如下：

Drop index <索引名>

子任务 2　数据操纵

SQL 的数据操纵功能包括 Select、Insert、Delete 和 Update 四个语句，即检索查询和更新两部分功能，在下面内容中将分别介绍上述功能的实现。

1．SQL 查询语句

SQL 的意思是结构化查询语言，其主要功能是同各种数据库建立联系沟通。查询指的是对存储于SQL的数据的请求。查询要完成的任务是将SELECT语句的结果集提供给用户。Select语句从 SQL 中检索出数据，然后以一个或多个结果集的形式将其返回给用户。

Select 查询的基本语法结构如下：

```
Select (predicate){*｜table.*｜(table.))field(,(table.)field2(,…)}
(as alias1(,alias2(,…)))
(into new_table_name)
From tableexpression(,…)
(where…)
(group by…)
(order by…)(asc｜desc))
```

参数说明如下：

“predicate”：指定返回记录（行）的数量，可选 ALL、TOP。

“*”：指定表中所有字段（列）。

“table”：指定表名称。

“field”：指定表中字段（列）的名称。

“(AS alias)”：表中实际字段（列）名称的别名。

“(INTO new_table_name)”：创建新表及名称。

“tableexpression”：表的名称。

“(GROUP BY…)”：表示以该字段的值分组。

“(ORDER BY…)”：表示按升序排列，降序选 DESC。

例如，可以通过如下代码获取表 zhigong 内的所有职工信息。

```
Select*
From zhigong;
```

通过如下代码选择获取表 zhigong 内的部分表职工信息。

```
Select id ,name
Form zhigong;
```

上述代码只是获取了职工表中的职工编号和姓名信息。

```
Select*
From zhigong
Where name="红红";
```

上述代码获取职工表中姓名为“红红”的职工信息。

通过如下代码获取表 zhigong 内年龄大于 30 岁的职工信息。

```
Select*
From users
Where age>30
```

2．SQL 更新语句

SQL 的更新语句包括修改、删除和插入三类，下面将分别介绍。

（1）修改。SQL 语句的修改语法格式如下：

Update <表名>set<列名>=<新列名>

Where<表达式>

例如，如下代码将表 zhigong 内名为“红红”的职工年龄修改为 50 岁。

```
Update zhigong set age=50
Where name="红红"
```

同样，用 UPDATE 语句也可以同时更新多个字段，例如，如下代码将表 ZHIGONG 内名为“红红”的职工年龄修改为 50，所属部门修改为“化学”。

```
Update zhigong set age="50",dpt="化学"
Where name="红红"
```

（2）删除。SQL 语句的删除语法格式如下：

Delete

From<表名>

Where<表达式>

例如，如下代码将表 zhigong 内名为“红红”的职工信息删除。

```
Delete zhigong where name="红红"
```

（3）插入。SQL 语句的插入新表语法格式如下：

Insert into<表名>

Values (value1,value2….)

插入一行数据在指定的字段上的语法格式如下：

Insert into<表名>(column1,column2…)

Values (value1,value2….)

例如，通过如下代码向表 zhigong 插入名为“红红”、年龄为“20”的职工信息。

```
Insert into zhigong(tname,age)
Values("红红","20")
```

子任务 3　视图

视图是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制。视图是从一个或几个表导出的表，它与基本表不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据。

视图一经定义后，就可以和基本表一样被查询、删除，也可以在一个视图之上再定义新的视图。在下面内容中，将对视图的操作知识进行简要介绍。

（1）建立视图。SQL 语句建立视图的语法格式如下：

Create view<视图名>((列名))

As <子查询>

(where check option)

例如，通过如下代码建立不及格学生的视图。

```
Create view v_101 不及格(学号,姓名,课程号,成绩)
As select top 1000
Xs_kc.课程号,xs_kc.成绩,xsqk.学号,姓名
From xs_kc,xsqk
```

```
Where 课程号="101"and 成绩<60
Order by 学号
```

上面的子查询可以是任意复杂的 Select 语句，但是通常不允许含有 Order by 的子句和 disting 短句。

（2）删除视图。SQL 语句删除视图的语法格式如下：

```
Drop view<视图名>
```

例如，通过如下代码删除不及格学生的视图 v_101。

```
Drop view v_101;
```

也可以一次删除多个视图，示例如下：

```
Drop view view1,view_2
```

（3）修改视图。SQL 语句修改视图的语法格式如下：

```
Alter view(<datdbase_name>.)(<owner>.)view_name((column(,…n)))
As
Select_statement
(with check option)
```

子任务 4　SQL 高级操作

经过前面内容的学习，读者应该初步掌握了 SQL 语言的基本语法格式。在下面的内容里，向读者介绍 SQL 语言高级操作方面的知识。

1. 查询运算符

通过查询运算符，不但可以将库中的数据以更加精确的格式显示出来，而且可以指定操作数据。

（1）UNION 运算符。UNION 运算符通过组合其他两个结果表（例如 TABLE1 和 TABLE2）并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

（2）EXCEPT 运算符。EXCEPT 运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时（EXCEPT ALL），不消除重复行。

（3）INTERSECT 运算符。INTERSECT 运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 ALL 随 INTERSECT 一起使用时（INTERSECT ALL），不消除重复行。

使用运算符的几个查询结果行必须是一致的。

2. 连接查询

（1）左外连接。左外连接（左连接）：结果集既包括连接表的匹配行，也包括左连接表的所有行。语法如下：

```
Select a.a,a.b,a.c,a.d,b.c,b.d,b.f from a leet out join b on a.a=b.c
```

（2）右外连接。右外连接（右连接）：结果集既包括连接表的匹配连接行，也包括右连接的所有行。

（3）全外连接。全外连接：不仅包括符合连接表的匹配行，还包括两个连接表中的所有记录。

3．跨数据库操作

例如，通过如下语句实现跨数据库的表的复制功能。

```
Insert into b(a,b,c)select d,e,f from b in"data.mdb" where 条件
```

此处的数据库必须使用其绝对路径。

4．Between 的用法

Between 限制查询数据范围时包括了边界值，not between 表示不包括。示例如下：

```
Select*from table where time between time1 and time2
Select a,b,c from table1 where a not between 数值 1and 数值 2
```

5．关联表操作

例如，如下代码删除主表中已经在副表中没有的信息：

```
Delect*from table1 where not exists (select*from table2 where table1.field1=table2.field1)
```

如下代码实现了 4 表联查：

```
Select*from a left inner join b on a.a=b.b right inner join c on a.a=c.c inner join d on a.a=d.d where…..
```

至此，数据库技术的基础知识介绍完毕。由于篇幅有限，只对其基本的知识进行了简要介绍，至于更加详尽和完善的具体使用知识，读者可以参阅相关资料。本系统的数据库该如何设计呢？

子任务 5　购物系统数据库设计

1．数据库设计说明

结合前面进行的系统功能分析和需求分析，并应用数据库相关知识，设计如下面所示的数据项和数据结构。

A．商品信息：包括商品分类、商品名称、编号、价格等。

B．用户信息：包括用户名、密码、电子邮件、手机号码、QQ、真实姓名、密码提示问题及答案、详细地址等。

C．购物信息：包括购物用户、购物商品、购物总价、购物时间等。

D．留言信息：包括留言人姓名、留言内容、留言时间。

用户－商品关系图如图 3-8 所示。

2．系统数据库逻辑结构设计

逻辑结构设计的任务就是把概念结构设计阶段设计好的基本 E-R 图转换为与选用 DBMS 产品所支持的数据模型相符合的逻辑结构（本系统所使用的是 Microsoft 的 Access 2003 为关系型 DBMS）。下面运用关系数据库规范化理论，使 E-R 图向关系模型的转换。

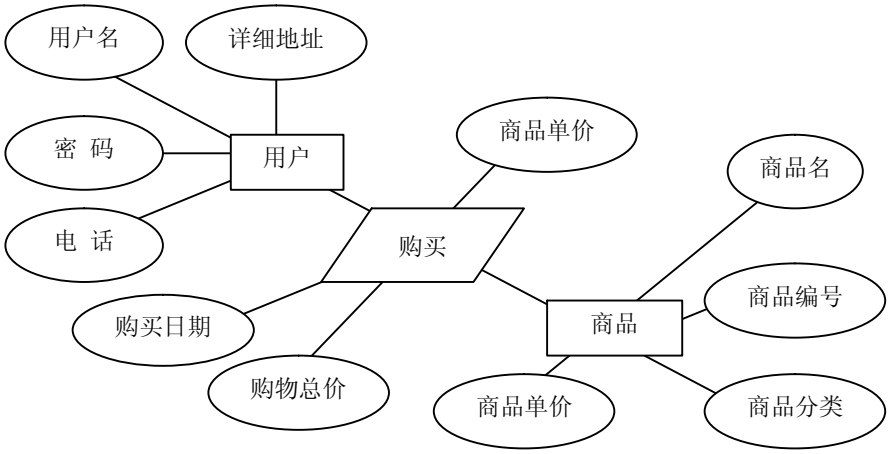


图 3-8　用户－商品关系图

数据库中的主要表结构如下：

用户信息表（User）及实体属性图如图 3-9 和图 3-10 所示。

	字段名称	数据类型
🔑	id	自动编号
	username	文本
	password	文本
	time	日期/时间
	useremail	文本
	UserQuestion	文本
	UserAnswer	文本
	realname	文本
	welcomeshengbuy	文本
	UserMobile	文本
	userqq	文本
	post	文本

图 3-9　用户信息表

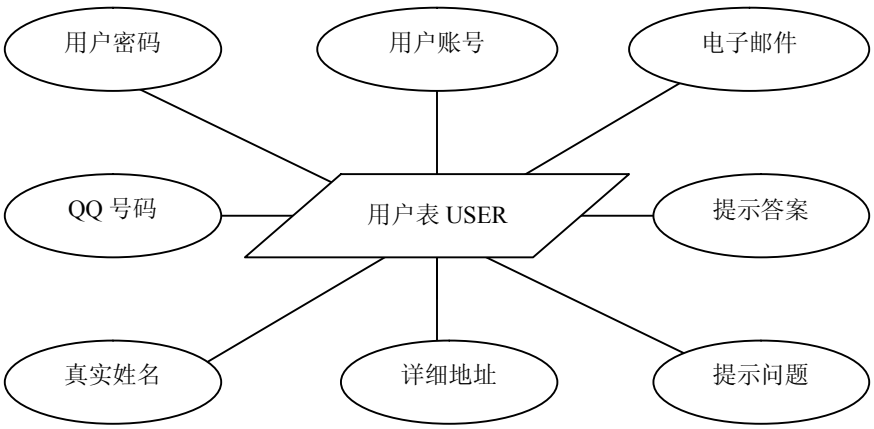


图 3-10　“用户”实体的属性图

商品信息表（wp）及实体属性图如图 3-11 和图 3-12 所示。

字段名称	数据类型
a_id	自动编号
a_name	文本
a_hb	文本
a_img	文本
a_type	文本
a_time	日期/时间
a_buy	是/否

图 3-11　商品信息表

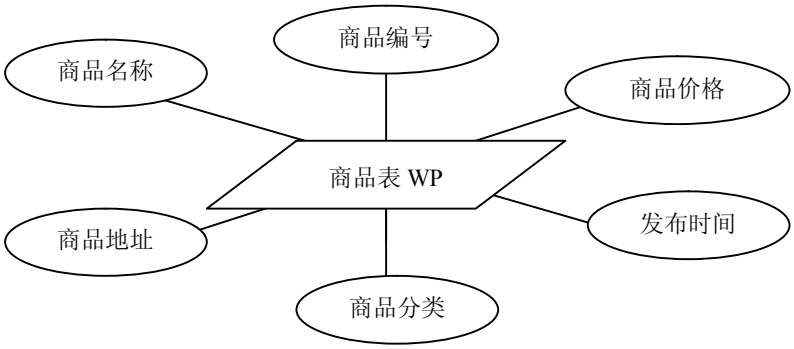


图 3-12　“商品”实体的属性图

购物表（p）及实体属性图如图 3-13 和图 3-14 所示。

字段名称	数据类型
id	自动编号
uname	文本
p_id	数字
state	是/否
p_img	文本

图 3-13　购物表

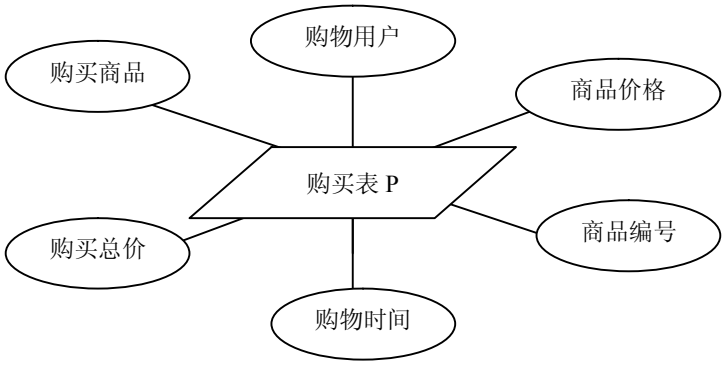


图 3-14　“购物”实体的属性图

留言表（liuyan）及实体属性图如图 3-15 和图 3-16 所示。

字段名称	数据类型
b_id	自动编号
b_name	文本
b_title	文本
b_neirong	文本
b_time	日期/时间

图 3-15　留言表

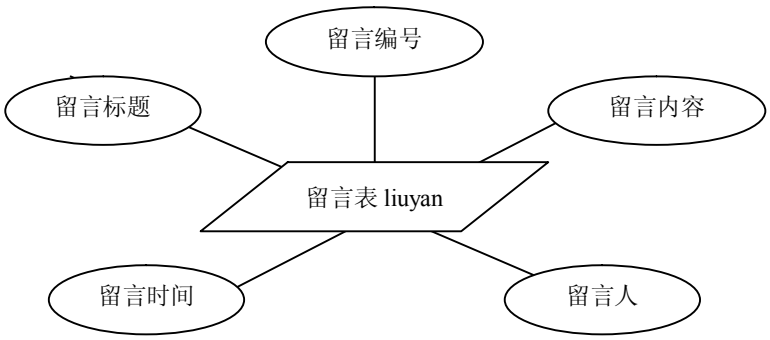


图 3-16　“留言”实体的属性图

任务 4　创建数据库连接

使用数据库时，必须首先建立和数据库的连接。只有建立连接后，系统程序才能调用和控制数据库中的数据。本节将对数据库连接方面的知识进行简要介绍。

子任务 1　代码连接

代码连接是指利用数据库的连接参数进行连接。主要包括如下参数：

（1）数据库位置：数据库的保存路径。

（2）数据库密码：数据库设置的密码。

上面介绍的数据库路径既可以是相对路径，也可以是绝对路径。例如，下面代码使用绝对路径建立数据库连接。

```
<%
Set conn = server.createobject(ADODB.connection)
Strconn=DEIVER=｛microsoft access driver(*.mdb)｝;
Strconn=strconn & DBQ= e: \ yanhang \ database.mdb
  Conn.open strconn
%>
```

下面代码使用相对路径建立数据库连接。

```
<%
Set conn = server.createobject(ADODB.connection)
```

```
Strconn=DEIVER=｛microsoft access driver(*.mdb)｝;
Strconn=strconn&DBQ=&server.mappath
("/database/yanhang.mdb")
Conn.open strconn
%>
```

在实际中的具体应用系统中，数据库连接文件往往是通用文件，是系统首先需要读取的代码通常使用 include 语句调用。在实际的系统开发过程中，为提高系统安全性，需要经常在数据库连接文件中，设置一些安全措施。例如，如下代码不仅能够建立数据库连接，而且能够过滤掉系统的单引号，实现字符相互转换。

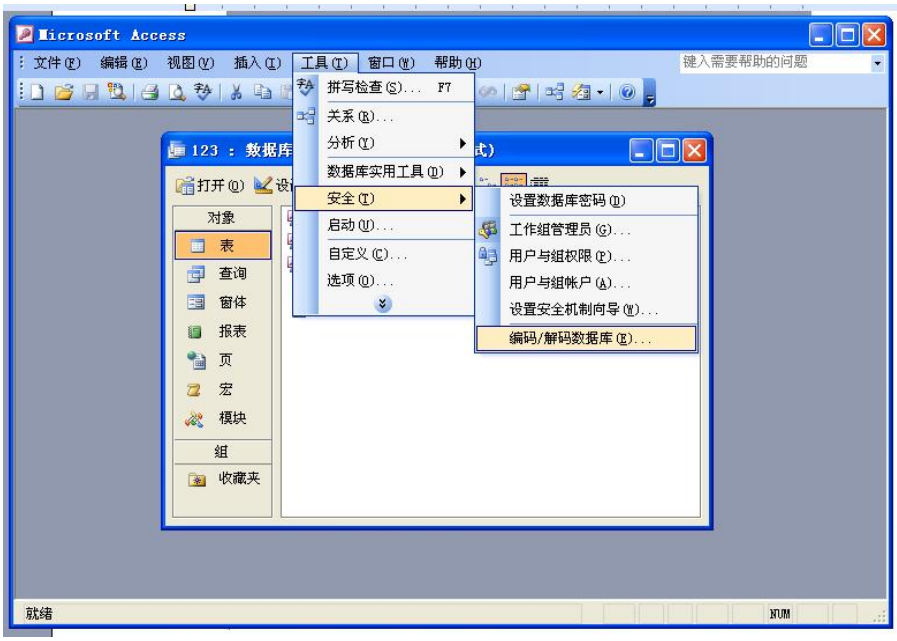
```
<%
Dim conn
Dim connstr
Dim db
Db=" database/trfsoft.mdb"
Connstr=provider=Microsoft,jet.OLEBD.4.0;datasource=& server.mappath
Set conn=server.createobject(ADODB.connection)
If err then
Err.clear
End if
Conn.cursorlocation=aduseclient
Conn.open connstr
Sub closeconn()
    Conn.close()
    Set conn=nothing
End sub
Function realstring(strscr)
Realstring=replace(trim(strscr),",")
End function
Function convert(strscr)
    Convert=server.HTMLENcode(replace(trim(strscr),"""")
    Convert=replace(convert,chr(13),"<br>")
End function
%>
```

如果连接字符串中的数据库路径一旦泄露，将会危及到整个系统的安全。例如，如果知道了 Access 数据库的存放路径，在浏览器地址栏中输入路径地址后，就可以轻易地把数据库文件下载到本地的机器中。

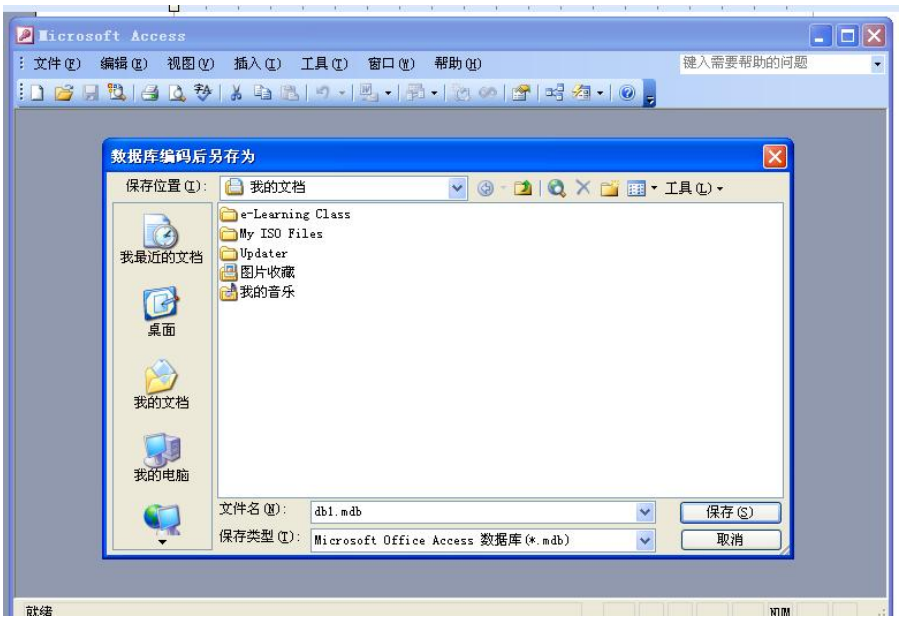
在建立和 Access 数据库的连接时，笔者提出如下建议：

1）非常规方法命名。防止数据库被找到的最简便方法是为 Access 数据库文件起一个复杂的非常规名字，并存放在多层目录下，另外加上特殊字符，并且以.asp 之类的格式作为后缀。例如，对于网上商城系统的数据库文件，不要简单地命名为“shop.mdb”等，而是要起个非常规的名字。例如，#faq19bal*.asp，再把它放在如./ffgfg6t/kjgf/zcd/av 之类的深层目录下。

2）数据库编码、加密处理。为了维护 Access 数据库的安全性，最好对数据库文件进行编码及加密。数据库编码处理的操作流程如图 3-17 所示。



（一）依次单击“工具”“安全”“编码/解码数据库”选项，选取要加密的数据库



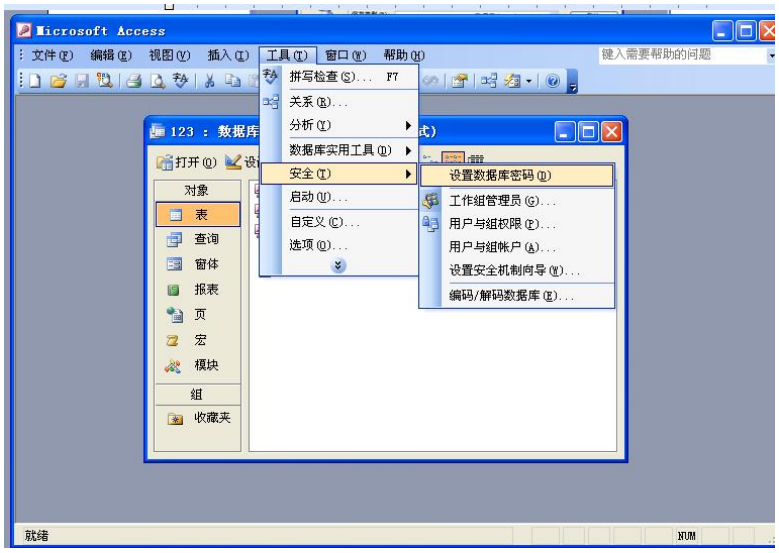
（二）在出现的“数据库编码后另存为”的窗口中，保存编码后的数据库。

单击“确定”按钮后数据库将会被编码

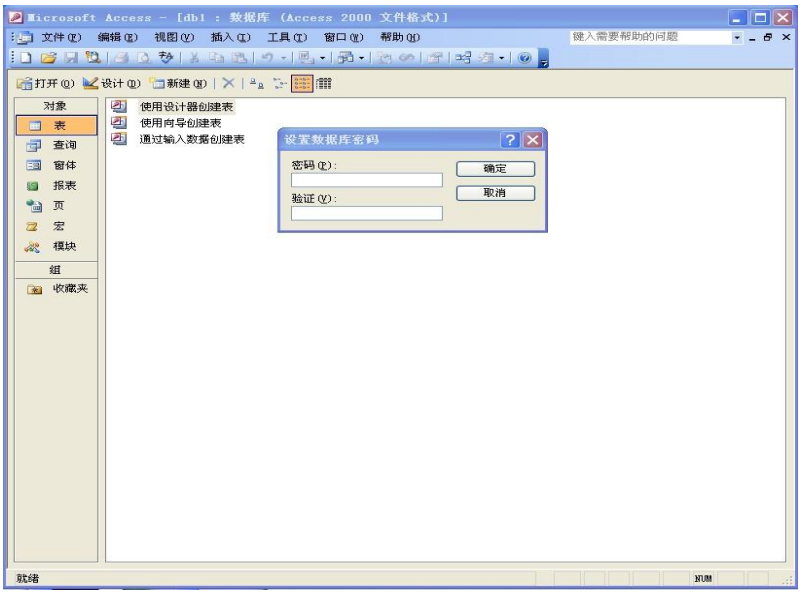
图 3-17　数据库编码流程图

以上的操作并不是对数据库设置密码，而只是对数据库文件加以编码，目的是为了防止他人使用别的工具来查看数据库文件的内容。

接下来要为数据库加密，具体操作流程如图 3-18 所示。



（一）依次单击“工具”“安全”“设置数据库密码”选项



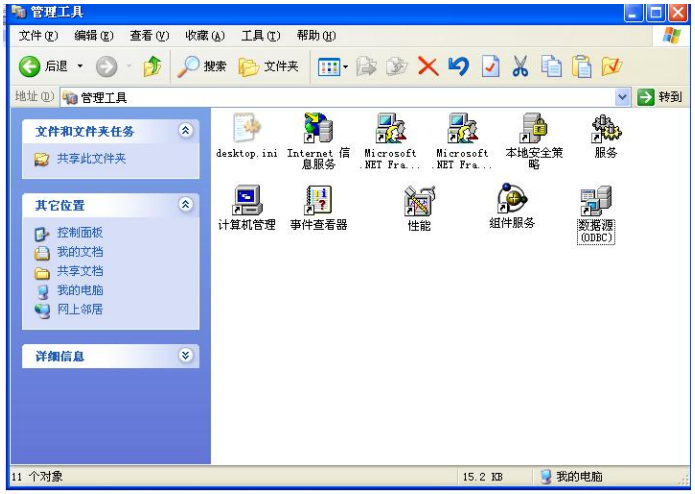
（二）在出现的“设置数据库密码”的窗口中，输入加密密码并单击“确定”按钮后，此数据库将会被加密

图 3-18　数据库加密流程图

子任务 2　数据源连接

利用数据源连接（DSN）连接，即利用 ODBC 数据源进行连接。在使用此方法前，需要首先给连接数据库起一个数据源名，然后利用数据源名进行连接。

要创建一个数据源连接，首先给服务器提供数据库的名字、通信所需要的 ODBC 驱动程序以及在网络中的地址。下面以 Access 数据库为例，介绍数据源的创建流程。具体操作流程如图 3-19 所示。



（一）在控制面板的管理工具中双击打开数据源（ODBC）

（二）在数据源管理器窗口中选择“系统 DSN”，然后单击“添加”按钮

（三）选择 Access（*.mdb）选项后单击“完成”按钮

图 3-19　数据设置流程图

（四）在弹出的安装对话框中单击“选择”按钮

（五）在弹出的界面中选择数据库，单击“确定”按钮

（六）输入数据源名后，单击“确定”按钮完成操作

图 3-19　数据源设置流程图（续）

经过上述流程的数据源设置后，就可以建立数据库的连接了。如果连接数据库没有设置访问密码，可以通过如下代码实现连接：

```
<%
Set conn=server.createobject(ADODB.connection)
Conn.open 111
%>
```

其中“111”是在如图 2-6 所示流程中建立的数据源名，username 和 password 是具有访问权限的用户名和对应得密码。

在 ASP 程序设计中，应尽量使用 ODBC 数据源连接数据库。不要把数据库名直接写在程序中，否则，数据库名将随 asp 源代码的失密而一同失密。

拓展知识：ODBC 技术和 Recordset 记录集

ActiveX Data Objects（ADO）是一项容易使用并且可扩展的将数据库访问添加到 Web 页的技术。可以使用 ADO 去编写紧凑简明的脚本以便连接到 Open Database Connectivity（ODBC）兼容的数据库和 OLE DB 兼容的数据源。

如果您是一个对数据库连接有一定了解的脚本编写人员，那么您将发现 ADO 命令语句并不复杂而且容易掌握。同样地，如果您是一个经验丰富的数据库编程人员，您将会正确认识 ADO 的先进的与语言无关性和查询处理功能。

1. 创建 ODBC DSN 文件

在创建数据库脚本之前，必须提供一条使 ADO 定位、标识和与数据库通讯的途径。数据库驱动程序使用 Data Source Name（DSN）定位和标识特定的 ODBC 兼容数据库，将信息从 Web 应用程序传递给数据库。典型情况下，DSN 包含数据库配置、用户安全性和定位信息，且可以获取 Windows NT 注册表项中或文本文件的表格。

通过 ODBC，您可以选择希望创建的 DSN 的类型：用户、系统或文件。用户和系统 DSN 存储在 Windows NT 注册表中。系统 DSN 允许所有的用户登录到特定的服务器上去访问数据库，而用户 DSN 使用适当的安全身份证明限制数据库到特定用户的连接。文件 DSN 用于从文本文件中获取表格，提供了对多用户的访问，并且通过复制 DSN 文件，可以轻易地从一个服务器转移到另一个服务器。

由于以上原因，本主题中的示例将使用文件 DSN。

通过在 Windows 的“开始”菜单打开“控制面板”，可以创建基于 DSN 的文件。双击“ODBC”图标，然后选择“文件 DSN”属性页，单击“添加”，选择数据库驱动程序，然后单击“下一步”按钮。按照后面的指示配置适用于您的数据库软件的 DSN。

2. 配置 Microsoft Access 数据库的文件 DSN

在“创建新数据源”对话框中，从列表框选择“Microsoft Access Driver”，然后单击“下一步”按钮。

键入您的 DSN 文件名，然后单击“下一步”按钮。

单击“完成”按钮创建数据源。

在“ODBC Microsoft Access 97 安装程序”对话框中，单击“选择”按钮。选择 Microsoft Access 数据库文件（*.mdb），然后单击“确定”按钮。

注意由于性能和可靠性的原因，我们极力推荐使用“客户－服务器数据库引擎”配置由

这样一种 Web 应用程序驱动的数据，这些 Web 应用程序必须满足 10 个以上的用户同时访问。尽管 ASP 可以使用任何 ODBC 兼容的数据库，但它是为使用客户－服务器数据库而设计的，而且经过了严格的测试，这些数据库包括 Microsoft SQL Server、Oracle 等。

ASP 支持共享文件数据库（如 Microsoft Access 或 Microsoft FoxPro）作为有效的数据源。尽管在 ASP 文档中的一些示例使用共享文件数据库，但我们建议只将此类数据库引擎用于开发或有限的配置方案。共享文件数据库可能无法很好地适用于可满足高需求、高质量的 Web 应用程序的客户－服务器数据库。

3．配置 SQL Server 数据库文件 DSN

注意如果数据库驻留在远程服务器上，请与服务器管理员联系，获取附加的配置信息；下面的过程使用 SQL Server 的 ODBC 默认的设置，它可能不适用于您的硬件配置。

在“创建新数据源”对话框中，从列表框中选择“SQL Server”，然后单击“下一步”按钮。

键入 DSN 文件的名称，然后单击“下一步”按钮。

单击“完成”按钮创建数据源。

键入运行 SQL 服务程序的服务器的名称、登录 ID 和密码。

在“创建 SQL Server 的新数据源”对话框中，在“服务器”列表框中键入包含 SQL Server 数据库的服务器的名称，然后单击“下一步”按钮。

选择验证登录 ID 的方式。

如果要选择 SQL 服务器验证，请输入一个登录 ID 和密码，然后单击“下一步”按钮。

在“创建 SQL Server 的新数据源”对话框中，设置默认数据库、存储过程设置的驱动程序和 ANSI 标识，然后单击“下一步”按钮（要获取详细信息，请单击“帮助”）。

在对话框（同样名为“创建 SQL Server 的新数据源”）中，选择一种字符转换方法，然后单击“下一步”按钮（详细信息，请单击“帮助”）。

在下一个对话框（同样名为“创建 SQL Server 的新数据源”）中，选择登录设置。

注意典型情况下，只能使用日志来调试数据库访问问题。

在“ODBCMicrosoft SQL Server 安装程序”对话框中，单击“测试数据源”。如果 DSN 正确创建，“测试结果”对话框将指出测试成功完成。

4．SQL Server 连接和安全信息

如果您正在开发用于连接远程 SQL Server 数据库的 ASP 数据库应用程序，应考虑以下问题：

连接方案：您可以选择 TCP/IP 套接字和命名管道的方法访问远程的 SQL Server 数据库。当使用命名管道时，因为在建立连接之前，数据库用户必须被 Windows NT 确认，所以对只有适当的 SQL Server 访问身份而在该计算机上没有 Windows NT 用户账号的用户可能会被拒绝访问命名管道。作为一种替代方案，使用 TCP/IP 套接字的连接可直接连接到数据库服务器，而不必通过使用命名管道的中间计算机。因为使用 TCP/IP 套接字连接可直接连接到数据库 Server，所以通过 SQL Server 的确认，用户就可以获得访问权，而不必通过 Windows NT 的确认。

注意在连接到远程数据库时使用 TCP/IP 套接字可提高性能。

安全性：如果您使用 SQL Server 的集成或混合安全特性，并且 SQL Server 数据库位于远程服务器上，则不能使用 Windows NT 请求/响应的确认。也就是说，不能将 Windows NT 请

求/响应身份证转发到远程计算机上，而只能使用基本身份验证，它根据用户提供用户名和口令信息进行。

有关这一主题的详细信息，请参阅 http://www.microsoft.com/sqlsupport/上的 Microsoft SQL Server 技术支持主页。

5. 配置 Oracle 数据库文件 DSN

首先要确保 Oracle 用户软件被正确地安装要创建 DSN 的计算机上。详细信息请与服务器管理员联系或参阅数据库软件文档。

在“创建新数据源”对话框中，从列表框中选择“MicrosoftODBCfor Oracle”，然后单击“下一步”按钮。

键入 DSN 文件的名称，然后单击“下一步”按钮。

单击“完成”按钮创建数据源。

输入用户名、密码和服务器名，然后单击“确定”按钮。

注意 DSN 文件用.dsn 扩展名，位于 \\Programs\\Common Files\\ODBC\\Data Sources 目录中。

有关创建 DSN 文件的详细信息，请访问 Microsoft ODBC Web 站点：http://microsoft.com/odbc/。

6. 连接数据库

访问数据库信息的第一步是和数据库源建立连接。ADO 提供 Connection 对象，可以使用该对象建立和管理应用程序和 ODBC 数据库之间的连接。Connection 对象具有各种属性和方法，可以使用它们打开和关闭数据库连接，并且发出查询请求来更新信息。要建立数据库连接，首先应创建 Connection 对象的实例。例如，下面的脚本创建 Connection 对象，接着打开数据库连接：

```
<%
\'Create a connection object
Set cn = Server.CreateObject(\"ADODB.Connection\")
\'Open a connection; the string refers to the DSN
cn.Open \"FILEDSN=MyDatabase.dsn\"
%>
```

注意无论在等号（=）之前还是之后，DSN 字符串都不能包含空格。

在这种情况下，Connection 对象的 Open 方法引用基于 DSN 的文件，其中包含关于数据库的位置和配置信息。也可以不引用 DSN，直接显式引用供应程序、数据源、用户 ID 和密码。有关建立连接的可选方法的详细信息，请参阅 Microsoft ActiveX Data Objects（ADO）。

7. 用 Connection 对象执行查询

用 Connection 对象的 Execute 方法，您可以发出结构化查询语言（SQL）查询数据库源并检索结果。SQL 是用于与数据库通信的工业标准语言，它有许多命令可用来检索和更新信息。

下面的脚本使用 Connection 对象的 Execute 方法在 SQL INSERT 命令的表格中发出查询，该命令将数据插入特定的数据库表格。在下面的示例中，脚本将名称 Jose Lugo 插入名为 Customers 的数据库表中。

```
<%
\'Define file based DSN
strDSN = \"FILEDSN=MyDatabase.dsn\"
\'Instantiate the Connection object and open a database connection
```

```
Set cn = Server.CreateObject(\"ADODB.Connection\")
cn.Open strDSN
\'Define SQL SELECT statement
strSQL = \"INSERT INTO Customers (FirstName, LastName) VALUES
(\'Jose\',\'Lugo\')\"
\'Use the Execute method to issue a SQL query to database
cn.Execute(strSQL)
%>
```

注意基于 DSN 路径字符串的文件在等号（=）前后不应包含空格。

除了 SQL INSERT 命令以外，您也可以使用 SQL UPDATE 和 DELETE 命令更改和删除数据库信息。

用 SQL UPDATE 命令，您可以改变数据库表中各项目值。下面的脚本使用 UPDATE 命令将 Customers 表中每个 LastName 字段包含姓 Smith 记录的 FirstName 字段更改为 Jeff。

```
<%
Set cn = Server.CreateObject(\"ADODB.Connection\")
cn.Open \"FILEDSN=MyDatabase.dsn\"
cn.Execute \"UPDATE Customers SET FirstName = \'Jeff\' WHERE LastName =
\'Smith\' \"
%>
```

要想从数据库表中删除特定的记录，可使用 SQL DELETE 命令。下面的脚本从 Customers 表中删除了所有姓 Smith 的行：

```
<%
Set cn = Server.CreateObject(\"ADODB.Connection\")
cn.Open \"FILEDSN=MyDatabase.dsn\"
cn.Execute \"DELETE FROM Customers WHERE LastName = \'Smith\'\"
%>
```

注意在使用 SQL DELETE 命令时，必须谨慎从事。当使用不带 WHERE 子句的 DELETE 命令时，它将删除表中的所有行。一定要包含 SQL WHERE 子句来指定要删除的确切行。

8. 使用 Recordset 对象处理结果

尽管 Connection 对象简化了连接数据库和查询任务，但 Connection 对象仍有许多不足。确切地说，检索和显示数据库信息的 Connection 对象不能用于创建脚本；您必须确切知道要对数据库作出的更改，然后才能使用查询实现更改。

对于检索数据、检查结果、更改数据库，ADO 提供了 Recordset 对象。正如它的名称所暗示的那样，Recordset 对象有许多可以使用的特性，根据您的查询限制，检索并且显示一组数据库行，即记录。Recordset 对象保持查询返回的记录的位置，允许一次一项逐步扫描结果。

根据 Recordset 对象的指针类型属性设置，可以滚动和更新记录。数据库指针可以在一组记录中定位到特定的项。指针还用于检索和检查记录，然后在这些记录的基础上执行操作。Recordset 对象有一些属性，可用于精确地控制指针的行为，提高检查和更新结果的能力。例如，可以使用 CursorType 和 CursorLocation 属性设置指针的类型，将结果返回给客户端应用程序（结果通常保留在数据库服务器上）并显示其他用户对数据库的最后一次更改。有关配置 Recordset 对象指针的信息，请参阅 Microsoft ActiveX Data Objects（ADO）。

9. 检索记录

一个成功的数据库应用程序都使用 Connection 对象建立链接并使用 Recordset 对象处理返回的数据。通过“协调”两个对象的特定功能，您可以开发出几乎可以执行任何数据处理任务的数据库应用程序。例如，下面的服务器端脚本使用 Recordset 对象执行 SQL SELECT 命令。SELECT 命令检索一组基于查询限制的信息。查询也包含 SQL WHERE 子句，用来缩小查询的范围。此例中，WHERE 子句将查询限制为所有的 Customers 数据库表中包含的姓 Smith 的记录。

```
<%
\'Establish a connection with data source
strDSN = \"FILEDSN=MyDatabase.dsn\"
Set cn = Server.CreateObject(\"ADODB.Connection\")
cn.Open strDSN
\'Instantiate a Recordset object
Set rsCustomers = Server.CreateObject(\"ADODB.Recordset\")
\'Open a recordset using the Open method
\' and use the connection established by the Connection object
strSQL = \"SELECT FirstName, LastName FROM Customers WHERE LastName =
\'Smith\' \"
rsCustomers.Open strSQL, cn
Cycle through record set and display the results
\' and increment record position with MoveNext method
Set objFirstName = rsCustomers(\"FirstName\")
Set objLastName = rsCustomers(\"LastName\")
Do Until rsCustomers.EOF
Response.Write objFirstName & \" \" & objLastName & \"<BR>\"
rsCustomers.MoveNext
Loop
%>
```

注意，在前面的例子中，用来建立数据库连接的 Connection 对象和 Recordset 对象使用该连接从数据库中检索结果。当您需要精确地设置和数据库建立链接所采用的方式时，这个方法是非常有用的。例如，如果需要在连接尝试失败之前指定等待的时间，则需要使用 Connection 对象去设置属性。但是，如果仅仅想使用 ADO 默认的连接属性建立连接，则应该使用 Recordset 对象的 Open 方法去建立链接：

```
<%
strDSN = \"FILEDSN=MyDatabase.dsn\"
strSQL = \"SELECT FirstName, LastName FROM Customers WHERE LastName =
\'Smith\' \"
Set rsCustomers = Server.CreateObject(\"ADODB.Recordset\")
\'Open a connection using the Open method
\'and use the connection established by the Connection object
rsCustomers.Open strSQL, strDSN
\'Cycle through the record set, display the results,
\' and increment record position with MoveNext method
Set objFirstName = rsCustomers(\"FirstName\")
```

```
Set objLastName = rsCustomers(\"LastName\")
Do Until rsCustomers.EOF
Response.Write objFirstName & \" \" & objLastName & \"<BR>\"
rsCustomers.MoveNext
Loop
%>
```

当使用 Recordset 对象的 Open 方法建立一个连接时，必须使用 Connection 对象去保证连接的安全。详细信息请参阅 Microsoft ActiveX Data Objects（ADO）。