第 3 章 数据库中表的设计

创建了物理数据库框架后，接下来即根据逻辑结构的分析结果将实体、联系、属性等所有相关内容完整地体现在数据库系统中。笼统地说数据库是存放大量相关数据的集合，而作为数据库中数据存储的主要载体，二维表的使用最为重要。

本章主要讲授表结构设计的相关内容，表数据的操作将在第 5 章详述。

3.1 数据表的创建

每一个完整的表都是由表结构（即平时所说的表头）和表数据两部分组成的，具体如图 3.1 所示。

学号	姓名	性别	出生时间	专业	家庭住址	联系电话	备注信息
0801101	张力	男	1988/03/11	软件	辽宁大连	0411-8784****	优等生
0801102	王强	男	1988/08/24	软件	辽宁大连	0411-8513****	
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

表结构

表数据

图 3.1 表的基本描述

相关知识

1. 表内容

一个数据库中的所有实体都能够建立起直接或间接的联系，因此在一个数据库中的所有数据也会因为实体与实体之间的联系而建立一定的联系。而每一个实体的所具有的各个方面的特性都将其属性统一整合在一个表中存储，置于数据库中，表与表之间建立的联系使数据库中的所有数据都产生一定的联系。诸如一个百货公司的商品销售管理系统中，若插入学生的考试成绩之类的内容就会显得格格不入。因此数据库中的表的设计必须具有相关性。

2. 表的分类

(1) 自由表：一经建立就永远储存于当前用户数据库中。

例如在用户数据库中创建的长期存储数据的表就是自由表。

(2) 临时表：为了测试在操作过程中的一些中间结果而临时创建表，不希望其占有内存，希望在数据库的连接关闭时随之不存在，即为临时表的用法。

数据库中的临时表包括全局临时表和局部临时表。其中以＃＃开头表示全局临时表，创建此表后对任何用户都是可见的，当所有引用该表的用户都断开链接时此表被删除；以＃开头的表示局部临时表，创建后只有在当前本地用户链接中是可见的，当用户与实例断开链接时，此表也同时被删除。

创建临时表与永久表的操作基本相同，也可以在临时表的定义上设置各种完整性约束来方便操作。

3. 表的组成

字段：二维表中的每一个列称为一个字段，字段的名字称为字段名(相当于属性的概念)。

记录：二维表中的每一行称为一条记录(相当于元组的概念)。

主键：能够唯一标识表中的每一条记录的字段或是字段的组合。

注意：数据表中的主键设置可以由两个字段或几个字段组成来满足记录的唯一性，但是必须注意的是无论是几个字段，只是它们的组合组成了主键。主键只有一个，每一个组成主键的字段只是主键的一部分。

4. 表的规范化设计

- 1NF：达到属性不可再分的程度。
- 2NF：去掉部分函数依赖。
- 3NF：去掉传递函数依赖。

5. SQL Server 2000 数据类型简介(如表 3.1 所示)

(1) 位型：bit 由 0 或 1 组成，表示真或假、开或关、是或否的关系，一般在设计字段的数据类型时，在“性别”字段只有“男”和“女”两个值，就可以设置为 bit 数据类型，省却空间。

表 3.1 SQL Server 中的数据类型

字段数据类型	范围及表示
bit	0 或 1 的整型数字
int	－2^31(－2 147 483 648)～2^31(2 147 483 647)的整型数字(32 位)
smallint	－2^15(－32 768)～2^15(32 767)的整型数字(16 位)
tinyint	0～255 的整型数字(8 位)
decimal	－10^38～10^38－1 的定精度与有效位数的数字
numeric	decimal 的同义词
money	－2^63(－922 337 203 685 477.580 8)～2^63－1(922 337 203 685 477.580 7)的货币数据，最小货币单位千分之十
smallmoney	－214 748.364 8～214 748.364 7 的货币数据，最小货币单位千分之十
float	－1.79E＋308～1.79E＋308 可变精度的数字

续表

字段数据类型	范围及表示
real	−3.04E+38～3.04E+38 可变精度的数字
datetime	从 1753 年 1 月 1 日到 9999 年 12 月 31 日的日期和时间数据，最小时间单位为 0.03 秒或 3.33 毫秒
smalldatetime	从 1900 年 1 月 1 日到 2079 年 6 月 6 日的日期和时间数据，最小时间单位为分钟
timestamp	时间戳，一个数据库宽度的唯一数字(8B)
uniqueidentifier	全球唯一标识符 GUID
char	定长非 unicode 的字符型数据，最大长度为 8 000
varchar	变长非 unicode 的字符型数据，最大长度为 8 000
text	变长非 unicode 的字符型数据，最大长度为 2^31−1(2GB)
nchar	定长 unicode 的字符型数据，最大长度为 8 000
nvarchar	变长 unicode 的字符型数据，最大长度为 8 000
ntext	变长 unicode 的字符型数据，最大长度为 2^31−1(2GB)
binary	定长二进制数据，最大长度为 8 000
varbinary	变长二进制数据，最大长度为 8 000
image	变长二进制数据，最大长度为 2^31−1(2GB)

(2) 数值型：包括 tinyint、smallint 和 int 等。

具体的表示范围可见表 3.1，需要注意 tinyint 类型的数据占用 1B 的存储空间；smallint 类型的数据占用 2B 的存储空间；int 类型的数据占用 4B 的存储空间。在实际操作中，可以根据表示范围的大小来设置合适的数据类型，表示分数值就可以使用 tinyint 类型，若表示范围大就可以选定 int 等类型。

(3) 精确数值型：即带有有限小数的精确数值，包括 decimal 和 numeric 两种数据类型，不存在本质的区别。具体使用方法见本节“知识扩展”内容。

注意：精确数值型与浮点型数据有一定区别，浮点型数据是无限小数，不能准确表示数据的精度；精确数值型表示的是带小数的精确数值，能够准确表示数据的精度。

(4) 货币型：包括 money 和 smallmoney 两种。其中 smallmoney 数据类型要求占用 4B 的存储空间；money 数据类型要求占用 8B 的存储空间。

(5) 浮点型：即含有近似小数的数据类型，包括 float 和 real 两种类型，表示范围见表 3.1。也可以用来存储科学记数法方式表示的数据。

(6) 日期时间型：包括 datetime 和 smalldatetime 两种。由有效的日期和时间组成，具体的表示范围见表 3.1。

(7) 时间戳型：timestamp 用于表示 SQL Server 各种执行内容的先后顺序，此数据类型设置字段的值由系统自动生成，形式为二进制数，确保这些数在数据库中是唯一的，与具体插入的数据内容或日期和时间等都没有关系。

使用时间戳型时需要注意，一个表中只能有一个 timestamp 列。

(8) 全局唯一标识符型：uniqueidentifier 由 16B 的十六进制数字组成，表示全局唯一性。当表中的记录行要求唯一时，使用 GUID 是较为方便的。

(9) 字符型：包括 char、varchar 和 text。表示固定长度的非 unicode 字符数据，最大

长度为 8 000 个字符。

① char：截长补短，表示定长字符型。在存储此类型数据时，若存储的字符串长度不足定义的字符串长度，则在存储字符串后面用空格补位；若存储的字符串长度超过定义的字符串长度，则将存储字符串后面多余的字符删除，保证满足定义的字符串长度要求。

例：定义为 char(5)，输入 abc 时，实际存储 abc＋两个空格；输入 abcdef 时，实际存储 abcde。

② varchar：截长不补短，表示变长字符型。在存储此类型数据时，若存储的字符串长度不足定义的字符串长度，直接存储即可；若存储的字符串长度超过定义的字符串长度，将存储字符串后面多余的字符删除，保证满足定义的字符串长度要求。

例：定义为 varchar(5)，输入 abc 时，实际存储 abc，输入 abcdef 时，实际存储 abcde。

(10) unicode 型：包括 nvarchar、nchar 和 ntext 等数据类型。

要求当字段中的值长度有变化时，则应该用 nvarchar 类型，最多可以存储 4 000 个字符；当字段中的值的长度固定时，则应使用 nchar 类型，最多可以存储 4 000 个字符；当使用 ntext 类型时，则可以存储 4 000 个字符，认为是长文本的存储方式。

注意：在 SQL Server 中，unicode 型的数据类型定义都以 n 做开头。

在 SQL Server 中，unicode 型的数据所占的存储空间比非 unicode 数据类型占据的空间要多一倍。

(11) 二进制数据类型：包括 binary、varbinary 和 image。

① binary：可以固定长度也可以改变长度。定义方法为 binary(n)，表示 n 位固定的二进制数据，n 的取值范围为 1～8 000。

② varbinary：表示 n 位变长度的二进制数据，n 的取值范围为 1～8 000。

③ image：数据以位字符串存储，由具体的应用程序解释而不是 SQL Server。可以存储的图片格式为 bmp、tief、gif 和 jpeg 等。

能力目标

以学生成绩管理数据库为例，按照特定的用户需求分析结果和逻辑设计结果，根据设计表的相关原则，能够自行创建符合一定规范化的二维表。

具体要求

(1) 掌握使用企业管理器创建学生信息表的方法。

(2) 掌握使用查询分析器创建学生信息表的语法结构。

实训任务

1. 分析学生实体后，可获得学生的若干属性，创建学生信息表(student)的相关属性和设定条件的表结构，如表 3.2 所示。

表 3.2　学生信息表(student)表结构

字段名	数据类型及长度	可否为空值	完整性设定	含义说明
stud_id	char(8)	否	唯一标识学生	学生学号
stud_name	nvarchar(8)	否		学生姓名
sex	bit	否	默认为 1	学生性别
birthday	datetime	否		出生时间
speciality	nvarchar(10)	是		所属专业
address	nvarchar(30)	是		家庭住址
telcode	nvarchar(14)	是		联系电话
meno	ntext	是		备注信息

1) 企业管理器创建学生信息表(student)

操作步骤：根据表 3.2 设置的表结构，选择数据库 student，在下拉菜单中选择“表”选项，右击选择“新建表”命令，打开对话框，如图 3.2 所示。

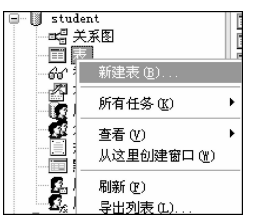

图 3.2　新建表的方式

之后在打开的表格中，按照具体的要求，输入各个字段的字段名称及相关的数据类型、宽度定义内容即可。一些完整性约束条件如主键、默认值等都是可以在建表伊始直接进行设置的，具体操作过程如图 3.3 所示。

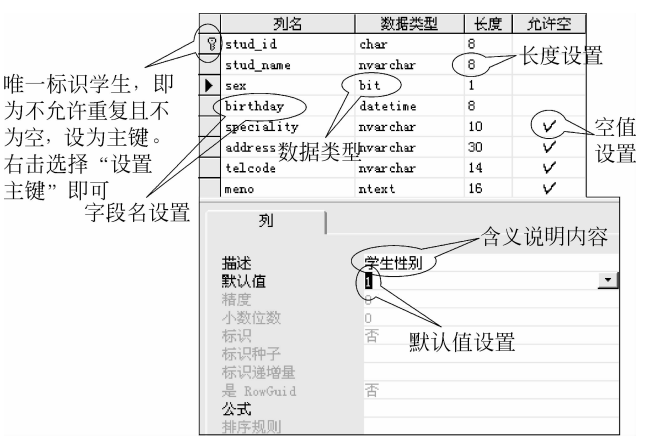

图 3.3　企业管理器创建学生信息表结构

保存时，单击工具栏中的按钮，在打开的对话框中输入 student，单击“确定”按钮即可，如图 3.4 所示。

除此之外，也可以不用数据库的下拉功能列表来选择“新建表”命令，而只需选中创建数据表的数据库，右击选择“新建”列表中的“表”命令创建亦可，具体操作如图 3.5 所示。

图 3.4 表的命名方式

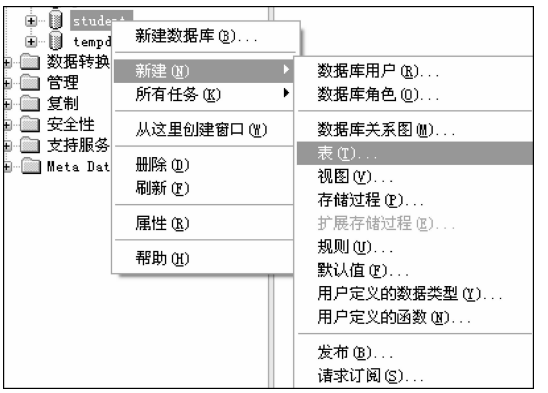

图 3.5 用数据库创建表的方式

2) 查询分析器创建学生信息表(student)

创建表的语法格式如下：

```
create table 表名
(字段名 1 数据类型(长度) 是否可为 null 值 其他完整性约束,
 字段名 2 数据类型(长度)……
 ……
 字段名 n 数据类型(长度) 是否可为 null 值 其他完整性约束)
```

注意：在编写程序时，都会使用一些注释语句，提高程序的可读性。现在介绍一下 SQL Server 2000 中的注释方法。

-- 单行注释，用来解释说明当前所在行的命令。

/*...*/ 多行注释，一般可放在程序开头，用来介绍整个程序的功能等。

具体在 student 数据库中创建学生信息表(student)的语法操作格式如图 3.6 所示。

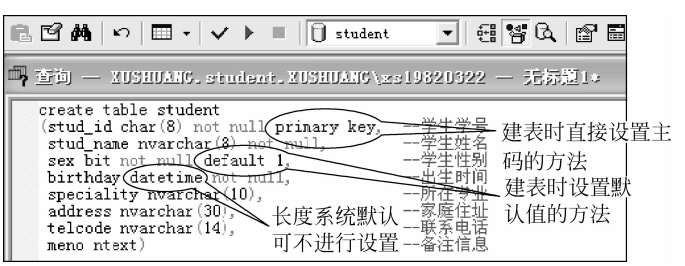

图 3.6 查询分析器创建学生信息表代码

注意：若要创建 student 数据表，首先应该打开用户数据库 student。使用命令 use student 即可。

命令代码写完后，单击工具栏上的 ▶ 按钮，运行即可，结果显示在消息框中，如图 3.7 所示。

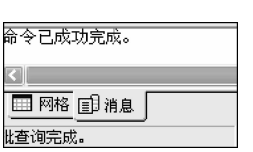

图 3.7 运行结果

2. 根据逻辑数据库分析结果，创建学生成绩管理数据库课

程信息表(lesson_info)的表结构,具体要求如表 3.3 所示。

表 3.3　课程信息表(lesson_info)表结构

字段名	数据类型及长度	可否为空值	完整性设定	含义说明
course_id	char(6)	否	唯一标识课程	课程编号
course_name	nvarchar(16)	否		课程名称
course_cap	nvarchar(10)	否		课程性质
course_type	nvarchar(10)	否	默认为必修	课程类型
course_mark	int	是		课程学分
course_time	int	是		课程学时

1) 企业管理器创建表(略)

2) 查询分析器创建表

```
use student
create table lesson_info
(course_id char(6) not null,
 course_name nvarchar(16) not null,
 course_cap nvarchar(10) not null,
 course_type nvarchar(10) not null default '必修',
 course_mark int,
 course_time int)
```

3. 根据逻辑结构分析结果,创建学生成绩管理数据库选课成绩表(stud_grade)的表结构,具体要求如表 3.4 所示。

表 3.4　选课成绩表(stud_grade)表结构

字段名	数据类型及长度	可否为空值	完整性设定	含义说明
stud_id	char(8)	否		学生学号
course_id	char(6)	否		课程编号
grade	int	否		选课成绩

1) 企业管理器方式创建表(略)

2) 查询分析器方式创建表

```
use student
create table stud_grade
(stud_id char(8) not null,
 course_id char(6)not null,
 grade int not null)
```

能力测试

1. 测试题

(1) SQL Server 的字符型系统数据类型主要包括(　　)。

A. int、money、char　　B. char、varchar、text

C. datetime、binary、int　　D. char、varchar、int

(2) 当向表中某一定义为 char 型、长度为 6 的字段上输入 china 时，则实际上存储的数据为_____，若该字段设置的数据类型为 varchar 型，长度也为 6，则实际上存储的数据是_____。

2. 操作题

在 student 数据库中再创建一张教师信息表(teacher_info)的表结构，分别使用企业管理器和查询分析器方式，主要的内容如表 3.5 所示。

表 3.5　教师信息表表结构

字段名	数据类型及长度	可否为空值	完整性设定	含义说明
teacher_id	char(6)	否		教师编号
teacher_name	nvarchar(10)	否		教师姓名
sex	bit	否	默认值为 0	性别
birthday	datetime	否		出生时间
department	nvarchar(16)	是		所属系别
graduation	nvarchar(10)	否		学历
school	nvarchar(40)	是		毕业院校

知识扩展

1. decimal 和 numeric 数据类型详细介绍

两者均为精确数值型，既不是整型，也不是浮点型，是带有有限小数的数值类型的表示方法。此数据类型种类最多可以存储 38 位数字，而且所有的数字都能够放在小数点的右边。两种精确数值型之间用法基本相同。下面以 decimal 为例进行讲解。

语法格式：`decimal(p,s)`
　　`numeric(p,s)`

注释：p 表示精度(即有效数位，不包括小数点在内)，s 表示小数位数。

规则：0<=s<=p<=38

注意：若设置 decimal 或 numeric 数据类型的字段中的数据值精度超过给定的精度长度，SQL Server 2000 自动按照四舍五入进行数值处理。精确数值型数据所占的存储空间大小是根据数据值本身的位数来确定的。

例如：创建一个名为 a 的表，其中只有一个名为 b、数据类型为 decimal(7,3)字段。

代码为：
```
create table a values(b decimal(7,3))
```

当向表 a 插入数据时，必须满足整数位<=4，小数位任意长度可由系统自行处理到指定长度。如向表中插入数据 456.567 8，结果将四舍五入。

代码为：
```
insert into a values(456.567 8)
```

插入后，查看表中的数据如图 3.8 所示。

例：如下面命令，创建表 a1。

```
create table a1(num decimal(7,3))
```

之后向表中插入数据值 9 999.999 4，则会出现什么结果？若插入的数据值为 9 999.999 6，则又会出现什么样的结果？

解：由于 decimal 类型的数据具有的特点，按照条件给定设置，当插入表的数据是 9 999.999 4 时，取三位小数四舍五入，则实际存储的数据为 9 999.999；而当插入的数据为 9 999.999 6 时，四舍五入后，会产生如图 3.9 所示的错误。

	b
1	456.568

图 3.8　数据插入结果

```
服务器：消息 8115，级别 16，状态 8，行 1
将 numeric 转换为数据类型 numeric 时发生算术溢出错误。
语句已终止。
```

图 3.9　溢出错误

因此，在操作 decimal 和 numeric 类型的数据时，必须注意不要让输入的数据值超出设定范围，否则会导致各种错误。

2．timestamp（时间戳）数据类型

一种自动记录时间的数据类型。当在一个数据表上创建一个数据类型为 timestamp 的列时，这个列的值会被自动赋值。

时间戳型是二进制数据类型的一种，长度为 8B。当对定义了 timestamp 数据类型的表进行插入、更新和删除操作时，系统就会自动更新时间戳型列上的数值。

例如，在 student 数据库中创建一个名为 table1 的数据表，包含 a、b 两个字段，a 字段为字符型，b 字段设置为 timestamp 数据类型，具体的表结构如图 3.10 所示。

图 3.10　创建 timestamp 列的数据表

之后，向表中插入两条记录，按照插入的顺序，添加数据时只需给出 a 字段的值即可，系统会自动为此两条记录的 b 字段赋值，结果如图 3.11 所示。

可以看到，插入的第一条记录的 b 字段的值比第二条记录的 b 字段的值小，是按照顺

序进行赋值的。此时可以再做测试，修改第一条记录的 a 字段值为 dog，查看对应的 b 字段的值是否随之被自动修改，结果如图 3.12 所示。

	a	b
1	hello	0x0000000000000134
2	china	0x0000000000000135

图 3.11　插入数据的结果

	a	b
1	dog	0x0000000000000137
2	china	0x0000000000000135

图 3.12　修改数据后的结果

如此可以看出，修改后的第一条记录的 b 字段值随之更改，timestamp 数据类型的列记录了数据表的最后更改数据。因此可以在数据表上定义时间戳型的列来记录用户对数据库中数据表的修改情况。

同样的原理也适用于增加和删除数据的操作。

3. 用户自定义数据类型

当操作数据库中的数据表时，表之间都会存在直接或间接的联系。而这种联系的形成都是用相同的字段做中介的(即主外键约束)。在此情况下，相同的字段多次被使用在不同的表中，每一次都要进行定义是烦琐的(如学生成绩管理数据库中的学生信息表和选课成绩表中相同的字段 stud_id)。如何简化这样的操作过程，SQL Server 中引入了用户自定义数据类型的概念。

使用用户自定义数据类型，即将相同字段的相同定义放在一起，做出一个构造的数据类型储存，在具体的语义环境中使用此数据类型时，直接调用用户自定义数据类型即可。

① 创建用户自定义数据类型的方法为使用系统存储过程 sp_addtype。

语法格式：exec sp_addtype 用户自定义数据类型名,'系统数据类型及宽度','是否允许为空'

② 删除用户自定义数据类型的方法为使用系统存储过程 sp_droptype。

语法格式：exec sp_droptype 用户自定义数据类型名

实例：为学生成绩管理数据库(student)创建一个用户自定义数据类型 x，其系统数据类型为定长字符型，长度为 8，不允许为空。并将其绑定到一个用户表 a 的唯一一个字段 b 上使用。

(1) 企业管理器方式：在企业管理器中打开 student 数据库，在打开的下拉列表中选择“用户定义的数据类型”选项，右击选择“新建用户定义数据类型”命令，在打开的对话框中进行设置，具体操作如图 3.13 所示。

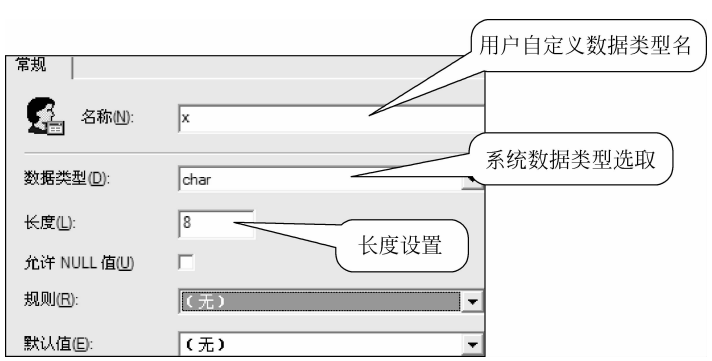

图 3.13　“用户自定义数据类型设置”对话框