目　录

目 录

contents

目 录

contents

CHAPTER

1

概念架构

这是本不一样的 Excel 宏书，我假设你已经具备宏程序设计的基础。其实，一直看下去你会发现，要读懂本书，需要的宏基础也不必太深。如果你完全不会宏，市面上、网络上都有非常多的书籍参考资料；如果你刚学会基本的宏程序设计，我相信本书所介绍的整套思路及其产生的技巧会让你以后的日子轻松愉快；如果你已经“误入歧途”（说笑的啦！），是个已经有多年老练经验的超级用户（power user），把 Excel 拿来当数据库用，而且已经用它设计了许许多多好用的系统，相信你更可以从本书获得新的启发。

语法、语义与本书

在许多领域，人们都试图以最少的投入做最多的事。我们总希望“事半功倍”，不想做重复的事。所以，在企业的经营上，经理人想找出 80-20 法则；在科学的探索上，科学家想发现各种一劳永逸、完美无瑕的公式；在管理上，政治家希望建立制度；对于国际纷争，各国要建立机制或惯例；在股市里，分析师想找出各种技术线型；对于景气预测，经济学家则建立各种指标；软件的开发就更不用说了，尽管“改变”已经跃居软件开发的头号敌人，“改变是惟一的不变”已成为软件开发的无上真理，但软件工程专家仍孜孜不倦地希望在这个善变的人类活动中勾勒出一些可以倚赖的基本元素。我们所熟知的操作系统（OS）、数据库管理系统（DBMS），以及可能不怎么熟悉的面向对象（OO）、SQL、XML、CMM、RUP/XP/Agile 等，正是我们开发各种信息应用系统，在项目管理或程序技术上所倚赖的“大型架构”。

语法与语义的关系

小技巧发挥小功能，大架构则发挥大功能。在学习本书之前所学的那些 Excel 宏命令，与其说是“一行行的命令”，倒不如把它视为“一种语言”或“一种语义模型”更为贴切。但这语义不是你的，是 Excel 它自己的。

你大概觉得纳闷，命令跟语言或语义的区别在哪里？不都是一行行的“程序代码”吗？

你可以这么想，命令如同孤伶伶的字词，语义则像是一篇文章，有你的思想在里面。命令只有正确与否，语义却有丰富的生命，命令大家都可以用，语义则可以表现出差异性，命令是软件厂商对该软件的定位，它提供了一个最基本的语法平台，他只做到那里，剩下的事，要靠你对自己信息需求的想法，以你自己的语义，用它所提供的命令加以实现。同一个程序的需求，每个人用的都是一样的 Excel 语法，但语义（程序逻辑）却很难完全相同。甚至一个月前的你跟现在的你，对同一个程序的想法都会有所不同。

Excel 只能检查宏的语法，检查不了宏语义。如果写程序永远停留在语法的层次，那程序代码就成了一本本命令堆砌出来的流水账，不但难以维护、难以理解、也难以适应改变，提升到语义的层次，程序设计这件事才有乐趣跟艺术可言。

语法、语义与编程

这些东西好像不是写给你看的，你可能听一些程序设计师说过什么面向对象，报上登的程序设计课程也有一种叫“xxx 面向对象程序设计”的名词，我学的是 Excel，最多被人称作是“power user”（超级用户），懂这些做什么？

你当然不需要也可以不必懂面向对象（虽然你懂了之后可能会上瘾），就可以写出能运行的 Excel 程序，我在这本书谈的也不是面向对象（它只是行文至此的一个例子）。但程序设计这种唯心的纯脑力活动，能了解一下更深入的程序技巧，不也是一件好玩且很有意义的事吗？事实上，Excel 可以接受的宏写法，远超出绝

大多数用户的想象，买了一部保时捷不开，却把它当作是野狼在骑，不是也挺可惜？

一种语法，多种表述

自然语言的语法规则

自然语言尽管有所谓的语法在规范着它，但仍具有极大的自由度，所以作家或诗人可以用有限且枯燥的字词，构筑出无限而流传千古的文章和诗集，语言是我们跟其他动物最根本的不同。而这里所说的语言，当然不是猴群的示警声、鲸鱼的求偶声或迅恐龙召唤同伴的信号（尽管电影声称它们已经很聪明）这样单纯，它所指的当然是语义。

程序语言的语法规则

程序语言的规则就是语法，这很简单，不难学会，除了 on-line help，Excel 甚至可以把你的一举一动都录制下来（少数特殊的动作仍无法以宏录制，例如在线数据查询、设置单元格的批注字体、发送邮件等），但即使这样，也仍旧制造不出“聪明”的宏程序代码，原因还是那一个：计算机无法了解你的语义。这一能力，有时动物都比计算机强，我的宠物有时似乎真听得懂我在跟它讲什么呢！即使 Excel 能把你的一举一动全都记录下来，但无法知道你真正想做的事情是什么，所以它录制不出一个两层循环的九九表程序。

语言的自由度与语义

由于语言的这种自由度，所以一种语法可以产生出千千万万种不同的文章，同一个词汇可以孕育出好几种不同的意义，字有限，而人类想要表达的意义无穷，所以词汇的重载（overload）——一个词汇代表多重意义就成了一个必然现象。

程序语言也是如此，每个程序语言的关键词——所谓的“保留字”或“命令”，例如 if then、for loop、procedure 等，数量上都少得惊人，学会任何一个程序语言的这些保留字，绝不会像是我高中时英语老师要我背英文字典那样没完没了，而即使是号称最新式的所谓面向对象语言，保留字的增长幅度也绝不可能像自然语言那样，每年都会出现一大堆的新字。

既然程序语言的保留字这么少，又几乎都是中学程度的英文单词，那程序语言到底难在哪里？或者说，让很多人觉得它无趣在哪里？我为何这么问？因为如果不难又有趣，加上待遇也不算太差，为什么我们的程序设计师密度不能像美国、日本或以色列那样？程序设计师虽然不是当今职场上的显赫行业，但到底还算得上“热门行业”之列，虽然菜鸟的待遇有时还比不过卖茶鸡蛋的收入，但也总算得上“知识经济”一族，况且大家可别忘了，（曾经是、直到去年都还是）举世最有钱的人比尔·盖兹就是个写程序起家的。

原因就在于程序设计并非把语法或命令背熟就好，那大概花三炷香的时间就够了，程序设计的精髓乃是有内涵的语义，而非硬邦邦的语法。

语义、哲学与架构

觉得我好像在讲哲学？

语义与哲学的关系

也许你想得没错。而且，很多学问讲到最后往往都会自动升级成哲学。连张三丰教张无忌太极拳，都要他“只重其意，不重其招”，不要死背招式（甚至还要忘掉它们），而要把太极心法了然于胸。

我曾经上过一种叫面向对象（这名词我讲了好多遍了）的“计算机”课程，课时为 50 小时，要价 25 000 台币，都还有不少人自费参加，某些人（不包含我）上完之后的感觉大概跟你现在有点像：他们觉得好像上了一堂哲学课。那门课程全都是没有任何计算机的。这让我想起了美国投资大师巴菲特的“营运总部”，是在一个没有计算机的乡间，他刻意不让底下的分析师过份依赖计算机所提供的信息，以免影响他们的对某些事物的基本判断。

第五项修炼的作者，管理学大师——彼得圣吉，他开的一些管理课程，也在乡间举行，让经理人放松心情，有些经理人因为没有听到“实务的管理技巧”而气愤地认为，圣吉教的不是管理，而是哲学。

再举一个例子，形象颇受争议的国际金融基金经理人兼慈善家索罗斯在他的“全球资本主义危机”一书中曾经提及，他的生财之道及人生观“是建立在一些抽象的哲学概念之上”的。

为何我要提及哲学？因为它跟程序设计的关系非常密切，甚至可以这么说，程序设计的本质并非数学，而是哲学（或者也可以这么说，它是好多领域的综合体，包括工程、艺术、管理、数学、心理学、法律、甚至政治），套用一位我所钦佩的软件工程师的话：“我们是在扮演虚拟世界的上帝角色”。

程序设计与数学的关系

常听到有人说，写程序要数学好，但写了这么多年的程序，我发现其实用到数学的时候并不多，用到的也不难。当然，这得看你对“数学好”的定义是什么。我的经验是，程序设计，尤其是商用的“一般”数据处理程序设计，大概只要高中数学程度就够了。具体来说，就是代数的概念，也可以说，它是一种将数字抽象化的能力。我记得有一次教一群小学生写程序（应补习班要求），我要他们计算长方形的面积，并在实际测试程序时，输入任意两个数字“代表”长方形的长与宽，那是 BASIC 吧！输入命令是 INPUT，照理说，程序是应该写成：

```
INPUT X, Y
```

而一位小学四年级的男生好不容易终于想出了程序的目的，但写出来的却是：

```
INPUT 2, 3
```

这时，一位六年级的男生告诉他说，要写成 INPUT X, Y，四年级的那个小男生不解，那个年长他两年的小朋友说出了一句

让我恍然大悟且至今仍记得的话：“喔！那个 X 跟 Y 是‘代词’啦！”

我这才告诉补习班的小学生，代数要中学才会教呢！缺乏将数字抽象化的最基本能力，连带也会缺乏将真实世界问题抽象化的能力，而这种能力正是进入程序设计世界的第一步。

语义是程序设计的核心精髓

先把你心中对哲学的刻板印象抛开（忘了人生哲学这等字眼），把它视作是一种“自圆其说的能力”。程序设计并不是要你去“证明”什么，而是要你去“创造”什么。所以数学不够好也没关系，大多数的时候，程序设计只需要中学程度的英语，数学呢？除非你要写的是反飞弹防御系统或 F16 战斗机上的程序，不然，前面已经提及，一般的商用数据处理最多高中程度的数学应该就可以搞定了。即使像保险系统中的精算功能，也有精算师（所谓的“领域专家”）会帮你搞定公式，你只需（在别人的帮助下）理解那些数学公式，将它翻译成数学公式的近亲——程序语言就行了。

答案已经很清楚，前面说了半天的“语义”，就是这里说的“哲学”。你对一个程序的想法，就是你日后向别人也向自己解释该程序的“语义模型”（程序的世界，一个月后的自己可能会完全不认识一个月前的自己，所以相信我，日前的你会有机会向日后的你解释的）。注意想法的“法”字，它在这里，等同于“制度”、“机制”、“模型”、“模式”、“原则”，以及我接下来要谈的“架构”。

程序的架构

简单程序中架构的作用

程序如果没有架构当然还是可以执行无误，一个九九表，不用两层循环，用 print 指令打印 81 次，除了多占一些硬盘空间跟 CPU 时间（反正因此而多花的空间跟时间大概比新鲜空气还便宜），结果并不会错误，如果还可以因此交出作业，虽然不会得到 A，但大概也不至于不及格，所以又有何不可？

当然可以，如果这样，我不必再说下去，你也不必再看下去，我们彼此谢谢，也不必再联络了。

但事情没这么单纯，之后如果老师要你交一份能产生八八乘法表或七七乘法表，或看准了你用 print 指令而要你交一份 100*100 乘法表的程序，甚至更狠，说任意输入两个数 m 跟 n，就输出 m*n 乘法表，那你的程序可就写不完了。由于缺乏一个最根本的架构（或“心法”），你的 print 招式禁不起一点点外在条件的变动。

复杂程序的架构

如果说这么简单而无聊、纯属学生交作业层次、对企业界毫无实际效益可言的程序，都存在着有架构没架构就有差别的状况，那稍微复杂一点的程序不就差更多了？

这一切之所以会如此，是因为软件的天敌是“改变”，程序代码不像电饭锅或电扇，可以一直用 30 年，它要一直变动、一直创

新才有用，DOS 并没有坏掉，它是可以用的，但大部分时候它只能供凭吊和怀念之用（不过我这么说也不尽正确，我常看到要求成本与稳定的百货商店的 POS 系统，仍在快乐地使用 DOS）。

软件做为一种形式化、结构化的语言，注定无法精确地描述瞬息万变的真实世界，但如果因此放弃精确描述的（部分）可能，显然也会把程序设计这回事带进完全混沌的随机选择之中，既然如此，则程序也不成为程序，更别说重复使用（reuse）的可能，因为每个程序都会是独一无二的“艺术品”。

架构与程序设计的关系

所以，你的程序还是要有架构。不论是何种架构，只要你觉得前后一致、易于理解、易于维护、易于扩充，不管是 Java 还是.NET、也不管是 Excel 还是 Word，都是好架构。有架构胜过没架构，架构可以说是你在程序语言与真实世界之间建立的一座思想桥梁，这座桥梁也会一直跟随着你，日后学习不同程序语言时，它会是一个基础，也就是，你总是可以从这个架构切入新的程序语言。你不会只是从单纯的 if then else、for loop 等语法层次切入新语言，因为每个一般性程序语言的语法部分都大同小异，只要几个小时就可以切入。你需要的是从你自己的语义角度切入，现代的开发工具或程序语言普遍支持面向对象、XML、SQL 等机制，这些“大架构”（有别于“设计模式“或“重整”等面向对象设计里的“微架构”），就是你可以倚赖的“心法”或是“底层结构”。《理性之梦》这本优秀的科普书中，作者曾提及他认识一位外交官，可以熟练掌握十几种语言，对那位语言天才而言，他心中似

乎有一种“心智结构”，每一种语言对他来说都像是同一种语言。

为何有些科学家可以如此迅速地从一个领域跳到另一个领域？他们难道可以记住如此多的东西？有些人的记性看似好极了，早上出门前到晚上回家后，他家里的东西如果被动过，他一眼就能察觉，但这真的是记性好吗？能迅速跨越领域的科学家与能记住东西摆设记性佳的人，他们会不会是用了“以简御繁”这个古老而基本的技巧？也就是，他们记住的东西不是表面的，是里面的；不是具体的，是抽象的；不是某个特定领域的知识，而是众多领域的共同架构；不是东西的确切位置，而是东西的摆放原则（物归定位、方向一致等）。

果真如此，我们是不是也可以不要只专注在语法，而要琢磨出语义，然后用这套语义学遍天下的程序语言？

说实话，我不知道，但我愿意相信这种可能，也一直在探求这种可能。每个程序语言诚然有其惟一性，但绝大多数的程序语言，尤其是关于数据处理的，彼此都长得极为相似。

在 Excel 中寻找程序架构

我们把 Excel 称为“电子表格”，对个人计算机软件史有一些认识的人应该对 VisiCalc 不陌生，VisiCalc 的原始构想正是有人在墙壁上画满了一个个的单元格，并且用来做这样的运算：“如果这一单元格的值改变，另一个单元格的值会变成什么？”

如果说电子表格就是单元格（Cell）的大集合，不失为一种直观的模拟、想法、隐喻、架构或模型。用 Cell 来切入电子表格，

以“泛 Cell 化”的观点来看待电子表格中的一切活动是符合电子表格本质的。我们在 Excel 中做的绝大多数操作，包括图形的产生（不包括产生之后的各种操作）都跟 Cell 有关。

随便举一些例子：

- 计算一堆 Cell 的总和、求出一堆 Cell 的总个数、“最大/最小/平均”的“文本/数字/日期”、标准差、方差、“最大/最小”的“行号/列标/地址”等。
- 在一大堆 Cell 中寻找某些符合条件的一小堆 Cell，然后对它做一些变颜色、改字体之类的显示方面的格式化操作。
- 按照一堆 Cell 的内容进行排序，甚至按照颜色进行排序。
- 从一个 Cell 的值带入或带出另一个 Cell 的值（这就是所谓“串联数据”）。
- 拆分、合并 Cell 的内容；Cell 中每隔多少字符自动换行。
- 取得 Cell 内容的某个部分，例如日期的年、月、周、日、时、分、秒等部分。
- 对一堆 Cell 中某个内容的出现次数统计、出现区间统计。
- 把不同工作表（sheet）中相同地址的 Cell 做重迭、相加、相减、相乘、相除等运算。
- 把一堆 Cell 中重复的内容消掉，或按照上一个 Cell 填入相同内容（在某种缩进量的一对多结构中）。
- 在一堆 Cell 中按照某条件删除或修改内容。

- 在一堆 Cell 中按照某个条件设置显示格式（条件可以不至 Excel 内定的 3 个）。
- 按照 Cell 本身的属性搜索 Cell，例如只搜索字号为 9 的 Cell 或颜色为红色的 Cell。
- ……

我们还可以举出更多例子。然而重要的是，这些看似不相干的操作，除了通通跟 Cell 有关，可否用同一个概念去理解它？如果可以，写宏时就可以用同一种架构，或说同一种写法去实现这些功能。

以“单元格”为出发点的 Excel 程序架构

如果把 Excel 当成是单元格 Cell，这将使得我们“见林不见树”、“看大不见小”，只看到一堆的 Cell，却没看到“一 Cell 一世界”的微观事实。如果把 Cell 看成是 Excel 的核心，我们不妨像物理学家那样，把物质理解到纳米层次，深入探究 Cell 里究竟有什么。

单元格的常规属性

单元格的属性很多，每一个单元格至少有以下“属性”：

- **内容**：可能是各种数据类型，如文本、数字、日期、时间等。当然数字还可再细分为是否有小数。
- **数据类型**：因内容的性质不同而产生的数据类型。

- **文本**：因为任何类型都可以看作是一种文本，所以文本也算 Cell 的一个属性。也就是说，如果某个 Cell 的值是“数字 100”，则文本值就是“字符串 100”。同理，一个 Cell 既然有可能是数字、日期、时间等，则我们也可以保存这些值。如果一个 Cell 的实际值是“时间 2004-01-01 10:30:30”的话，则其数字化之后的值就是 2004（按照 Excel 的 val 函数），当然，也可以根据需要换用其他值，例如零来表示。
- 总结一下，如果一个 Cell 的值是：
  - “数字 100”，则我们一共为这个 Cell 保存了另外的 3 个值：文本 100、空白的日期、空白的时间。
  - “文本 ABC”，则我们一共为这个 Cell 保存了另外的 3 个值：文本 ABC、空白的日期、空白的时间。
  - “日期 2004-01-01”，则我们一共为这个 Cell 保存了另外的 3 个值：数字 2004、时间 2004-01-01 12:00:00 PM、文本 2004-1-1（按照你设置的显示格式而定，也有可能是 2004-01-01）。
- **文本长度**：因内容而产生文本长度。对“文本断行”、“按照字数排序”这类需求，文本长度的信息是必要的。
- **地址**：包括行号、列标、行加列构成的地址。
- **格式**：包含背景色、内容的颜色（前景色）、文字（字体、字号、样式）、边框、对齐方式、缩进量、行高列宽、图片等。更细一点，还可以判断是否该 Cell 为合并后的 Cell 等。

单元格的其他属性

还有一些看不到的信息，比如 Cell 是依附于整个工作表的，

而工作表又依附于工作簿，工作簿最终有个包含完整路径的惟一文件名，这些都可以当作是 Cell 本身的信息内容。或许你现在不觉得这些内容有用，然而程序设计就是这样，需求总在不经意之间就从各处冒出来，所以，在硬盘成本跟 CPU 时间可接受的状况下（这点我没办法给你一个量化的指标，你得自己衡量），信息能存得多一点就多一点吧！

最后，还有一个所谓的“自定义内容”。就好像地球实际上并没有经纬线，是人们把它假想出来以便于理解某些事物一样。这类自定义内容是我们自己对 Cell 假想的一种秩序，最常见的就是给每个搜索到的 Cell 一个流水号。这样一来，当你只想处理符合条件的前 10 个 Cell 时，就有个基础了。

“一 Cell 一世界”可以保存的信息远不止于此，你当然还可以依自身需求保存一些别的东西。

上述那些信息在一般状况下已经足够，本书后面探讨的许多功能都将以那些信息为基础，你会发现，这些 Cell 的 meta data（元数据）可以轻易、直观地实现很多传统宏技巧必须大费周章才能实现的功能（而实现的方式却又是那么难以理解）。

把“单元格元数据”整理成数据库表

要让事物变得易于理解，将之“秩序化”、“形式化”是好方法。通俗地讲就是“把事情整理得井井有条”。这不但是 20 世纪伟大物理学家费曼的爸爸在他小时候教他的事，也是一般父母要求已懂事小孩的第一件事。不同的是，费曼老爸教他如何从磁铁

排列中看出秩序，一般的爸妈则要求小朋友乖乖别吵，守秩序，或把玩具归位。

单元格的元数据

考虑图 1-1 中这些单元格：

	A	B	C
1	100		
2	2004-1-1		
3	ABC		
4	2004-1-1 0:00		
5			
6			

图 1-1

把它们的元数据（meta data）找出来之后，整理成这种表格后形式如图 1-2 所示：请顺着箭头，以顺序的方式看这张图，它是一个表格，不是 4 个。

序号	地址	数据类型	文本	文本长度	数字
1	$A$1	5	'100	3	100
2	$A$2	7	'2004-1-1	8	
3	$A$3	8	'ABC	3	
4	$A$4	7	'2004-1-1 0:00	13	

日期	时间	列号	行号
		1	1
2004-1-1	2004-1-10:00:00	1	2
		1	3
2004-1-1	2004-1-10:00:00	1	4

字体	字号	字体颜色	单元格颜色	工作表
Tahoma	8	-4105	-4142	Sheet1
Tahoma	8	-4105	-4142	Sheet1
Tahoma	8	-4105	-4142	Sheet1
Tahoma	8	-4105	-4142	Sheet1

工作簿	文件名称
找出meta_data.xls	D:\document\excel魔法书\成稿\example\找出meta_data.xls
找出meta_data.xls	D:\document\excel魔法书\成稿\example\找出meta_data.xls
找出meta_data.xls	D:\document\excel魔法书\成稿\example\找出meta_data.xls
找出meta_data.xls	D:\document\excel魔法书\成稿\example\找出meta_data.xls

图 1-2

至此，用过 Access 或熟悉数据库开发工具的读者，可能已经知道接下来我们要做什么。

单元格元数据与数据库的关系

在数据库的世界里，这正是一个数据表，一个关于单元格的数据表。图中“序号”、“地址”、“数据类型”等这些纵向的信息，称为“字段”（Field or Column），而“序号为 1、2、3、4”的行称为“记录”（Record or Row）。数据库的表（Table）就是字段跟记录加起来的一种数据结构。

这张表有一些需要解释的东西：

- 数据类型的值（5、7、8）代表 Excel 的 VBA 是以这些常数代表数字、文本、日期时间的数据类型。
- 字符串可以包含数字或日期，从某种程度而言，任何东西都可视之为文本，我们生成的这张表格，字段的数据类型必须一致（这是数据表的基本原则），所以图 1-2 中会出现“’100”、“’2004-1-1”这种内容分别为数字及日期的文本。如果不这样处理，Excel 会把 100 当作“数字 100”，这样在之后的搜索上会有问题。
- 颜色-4105 是接近米色的一个颜色（我的计算机使用的颜色），-4142 则代表“不设置颜色”（但不代表“没有颜色”），此时 Excel 将使用你在 Windows 控制面板中的“显示”中的设置值。

元数据与程序架构

同一件事物，往往可以用不同的角度去观察及理解。一粒稻

米，可以用一粒稻米的肉眼角度去看，也可以把它的基因序列展开，用更细层次的模型去观察。警察在拍犯人的照片时都是正面一张、左右侧面各一张，甚至后面也来一张，我想，那是为了方便从不同角度辨认犯人的缘故吧！同样的，只有 4 个单元格的一份 Excel 工作表，你看到的是 4 个单元格，经过上述元数据的展开后，又是另一番风貌，而 Excel 本身在存储时，看到的恐怕比这些元数据更多。

我们的架构是以单元格为中心。Excel 做的很多工作抽象化来看，都不外乎是“找到一堆符合条件的单元格”，然后“对那些找到的单元格进行操作”。通常前者比较麻烦，因为条件可能很复杂，但有了上述这个数据表，这个查找的操作会变得非常容易。找到之后，再通过一个循环，就可以完成我们的任务了。

提取工作表元数据

事实上，工作表的 meta data 通常都可以从该工作表本身“推导”出来，不需要其他的东西。当然，图 1-2 中的表“序号”是“自定义内容”，这是外加的，无法自行推导，但仍可以通过程序设计用某种“有规则”的方式产生出来。

范例 1-1　从活动 Excel 工作簿提取元数据

从科海网站（www.khp.com.cn）下载本书程序代码，打开“找出 meta_data.xls”，确认打开宏（Alt+E），然后按 Alt+F11 键进入宏编辑环境，看看位于模块 Module1 内的这个宏：

范例程序 1

```
Public Sub ActiveWorkSheet_To_Table()
    Application.ScreenUpdating = False

    Dim loSheet As Worksheet
    Dim lnSerialNumber As Long
    Dim lcText As String
    Dim lcTextValue As String
    Dim loCell As Range
    Dim ldDate, ltTime, luValue As Variant
    Dim lcValue As String

    lnSerialNumber = 0

    Open "c:\cell_meta_data.csv" For Output As #1

    ' 写入字段标题
    Write #1, _
    "序号", _
    "地址", _
    "数据类型", _
    "文本", _
    "文本长度", _
    "数字", _
    "日期", _
    "时间", _
    "列标", _
    "行号", _
    "字体名称", _
    "字号", _
    "字体颜色", _
    "单元格颜色", _
    "工作表", _
    "工作簿", _
    "文件名称", _
    ""
```

```
    Set loSheet = Application.ActiveWorkbook.ActiveSheet
    If loSheet.Type = -4167 Then
        loSheet.Activate
        For Each loCell In loSheet.UsedRange.Cells
            With loCell
                lcTextValue = Trim(.Text)
                lcText = "'" + lcTextValue
                luValue = .Value

                lnSerialNumber = lnSerialNumber + 1

                ldDate = ""
                ltTime = ""
                lcValue = ""

                If IsDate(lcTextValue) And Not IsNumeric
(lcTextValue) Then
                    ldDate = CStr(DateValue(lcTextValue))
                    ltTime = Replace(Replace(DateValue
(lcTextValue) & TimeValue(lcTextValue), "上午", ""), "下午", "")
                End If

               If IsNumeric(.Value) And Len(lcTextValue) > 0 Then
                    lcValue = Trim(Str(Val(.Value)))
                End If

                Write #1, _
                lnSerialNumber, _
                .Address, _
                VarType(loCell), _
                lcText, _
                Len(lcTextValue), _
                lcValue, _
                ldDate, _
                ltTime, _
                .Column, _
```

```
                .Row, _
                .Font.Name, _
                .Font.Size, _
                .Font.ColorIndex, _
                .Interior.ColorIndex, _
                loSheet.Name, _
                loSheet.Parent.Name, _
                loSheet.Parent.FullName
            End With
        Next
    End If

    Close #1

    If Dir("c:\cell_meta_data.xls") <> "" Then
        Kill "c:\cell_meta_data.xls"
    End If

    Workbooks.Open ("c:\cell_meta_data.csv")
    ActiveWorkbook.SaveAs "c:\cell_meta_data.xls", xlExcel9795
    ActiveWorkbook.Close
    Kill "c:\cell_meta_data.csv"

    Application.ScreenUpdating = True
End Sub
```

范例程序关键代码点评

条件判断语句：

```
If loSheet.Type = -4167 Then
```

指的是，如果该工作表（sheet）是一般的工作表而非“图表

式的工作表”的话。因为“图表式的工作表”内没有任何 Cell，所以不在我们的读取之列。

```
    Val(Replace(Replace(Replace(lcTextValue, "D", ""), "E",
""), ",", ""))
```

这是因为有些数字是以科学记数法表示，有些则有加上千分位分隔符（3 位数字一撇）。

上述这个宏产生了一些关于单元格的元数据，正如你所见，这些元数据都可以从 Cell 这个对象（object）中的属性（property）直接获得。如果你有兴趣的话，还可以获得更多的属性，例如该 Cell 是否已经被设置为“合并单元格”、或该 Cell 的边框、是否被锁定、宽度、高度、数字格式...等属性。在一般状况下，上述这些属性应该够用了。

不知你是否注意到，我们是先以 CSV 格式写入，然后再“多此一举”地把 CSV 重新调出来，再存成 XLS 格式。之所以这么做，是为了写入 meta data 时的效率考虑。由于在应用上，你的 UsedRange 所含的 Cell 数目可能很多（甚至跨工作表及工作簿），而如你所知，Cell 内容的变动（当然也包括把数据写入 Cell 这种操作）在 Excel 内部会牵涉到许多事件，即使你把 ScreenUpdating 跟自动重新计算等属性关闭也是这样，所以把 Cell 的 meta data 写入 Cell 可能不是个好主意，我试过这么做，结果与“写入文本文件、调出来、再存成 XLS 文件”的执行速度相比差很多！

另外，要避免 UsedRange 过大的问题，除了不要对不必要

的 Cell 做格式设置之外（例如要设置 A1:E1 的颜色，请选 5 个 Cell，不要一次就选一列），最好也先把一些空白 Cell 整行或整列删除，以免导致 UsedRange 因为这些“肉眼看不见的 Cell”偷偷的非必要膨胀（但我发现，新版的 Excel 对这些无实质内容 Cell 的判断已经变得聪明多了，也就是不会把它列入 UsedRange 里，可我还是尽可能做上述操作，大概是对 Excel 的这点聪明还不太信任吧）。

除了 UsedRange，也可以“只针对选取区域”获得 meta data，只要把：

```
For Each loCell In loSheet.UsedRange.Cells
```

改成：

```
For Each loCell In Selection.Cells
```

就行了。这样就不会有 UsedRange 过大的问题，但必须多个选取的操作。

也可以直接针对某一块区域：

```
For Each loCell In Range("A20:Z100").Cells
```

如果你想试试计算机的性能——处理全部的 Cell（循环将进行 65536 * 256 = 16,777,216 次，一千六百多万次——危险动作，请勿模仿……）：

```
For Each loCell In loSheet.Cells
```

最后，如果不会利用“文件名称”这个属性，就不用把它写进院数据中，因为这部分也要占据相当大的空间（文件名称包含完整路径，所以可能会很长），从而消耗更长的时间。

Excel 工作簿与数据库的关系

单元格元数据表与传统数据库的数据表非常像，对数据库稍有概念的读者看到这个表就知道接下来该怎么做了，没错，就是利用 SQL（Structured Query Language，结构化查询语言）把符合条件的 Cell(s)找出来，然后对这些找出来的 Cell(s)进行操作，就是这么简单！

现在的问题是，单元格元数据是保存在 Excel 里，并不是保存在 Access 或其他数据库管理软件里，怎么用 SQL 呢？

上一小段历史课，虽没见诸于相关的官方文件，但我记得以前听过一种说法，还记得 Excel 的前辈 Lotus 1-2-3 吗？为何叫 123 呢？该种说法是，1 代表电子表格、2 代表图表（电子表格可以画图表）、3 代表的就是“数据库”。

电子表格本身看上去，就是长得一副表格的样子，对于一些了解应用程序开发工具的程序员来说，有时也像是一个超大型的 DataGrid 组件，而 DataGrid 这种组件，几乎可以说在任何数据库应用程序开发工具中，代表的都是数据表格（table）的一种可视化组件。更别说 Excel 的字段数目限制在 256，这个数字正是大多数数据库管理软件对于表格的字段数目上限（一般都是介于 250~256 之间）。

所以，Excel 本身可以被当作数据库，其单元格中的数据可以用数据库的方式取得，也就是用 Data Model 的方式取得；另一种获取数据的方式是通过 OLE Automation，也就是 Object Model 的方式取得。

安装完 Excel 后，到 Windows 的控制面板（以 Windows 2000 为例）选取“管理工具/数据源（ODBC）”，在“用户 DSN”选项卡中可以找到 Excel Files，单击“配置”按钮，就可以选取一个 Excel 文件当作数据库了。

不过，这边只是举例，告诉你安装完 Excel 后，Excel 的 ODBC Driver 也一并装起来了。我们日后真正存取 Excel 文件时，用的不是这里谈的方法，因为一个 ODBC DSN 只涉及到一个 Excel 文件，我们需要的是直接在程序代码中涉及到任何一个 Excel 的文件。否则，万一你有 50 个 Excel 文件，岂不是要设置 50 个 ODBC 数据名称了？

也许你会说，可以把所有数据表（也就是 sheet——工作表，后面很快就会谈到）通通放在同一个 xls 文件中。当然可以这么做，不过这样一来，该文件也许会太庞大而导致存取缓慢，还有一张表毁就全毁的风险，再者，有时我们会动态、临时产生一个 xls 文件，而且也希望用数据库的方式存取该文件。这时如果也要动态新建 ODBC DSN 指向那个临时产生的 xls 文件，就实在太麻烦了。

Excel 工作表与数据库表的关系

一般人常常把数据库挂在嘴边，这个数据库那个数据库，而

大部分的时候他们谈到的其实只是“一堆数据”，好像只要有一堆数据就自然变成了数据“库”（就像一大池子的水就被称为“水库”一样）。其实，即使用最简单的定义（就是不管视图、存储过程等），数据库应该都是由一堆数据表所构成。这里讲的数据表就是 Excel 里的工作表（sheet）。如果一个名为 MyData.xls 的文件有 5 个（数据结构井然的）工作表，分别是“客户”、“供货商”、“订单”、“订单明细”、“产品”，那我们可以说 MyData 这个数据库有上述这 5 个数据表。

注意，Excel 的工作表虽然看似是整整齐齐的一副表格样，但由于有“合并单元格”、同一栏允许不同数据类型等明显违反数据表的特性（例如 A1=数字 100，A2=文本“Hello, World”），所以，如果要用“数据库的方式”来存取某个 Excel 文件，这两项 Excel 的“特异功能”最好不要用，我说最好不要用意思是，用了虽然还是可以存取得到，但不保证存取到的内容百分之百是你要的，就是之前曾说过的会有数据类型转换的问题。当然，如果你不介意数据转换后的问题，或可以利用宏过滤掉“转换不正确”的数据，那当然没问题。这里所谓的“以数据库的方式存取 Excel 文件”就是“以 SQL 语法存取 Excel”的意思（不是以 SQL Server 存取，别搞混噢）。

包含 5 个工作表的这个 MyData.xls 工作簿可以用 SQL 存取，而且就在 Excel 自己的宏里用，不需要 Access 这类数据库软件（当然，Access 也可以通过 SQL 存取到 Excel 的数据，反之亦然）。

Excel 中的 SQL 数据库操作技术与应用范例

要在 Excel 中使用 SQL，要先小谈一下 ADO。

技术点：数据库存取组件技术 ADO

ADO 是 ActiveX Data Object 的缩写，它是微软开发的数据库存取组件，可以在诸如 Visual Basic 、Delphi、FoxPro、Office VBA 等中调用，直接用以下的程序代码来说比较清楚。

关于 ADO（及其新版的 ADO.NET），可以谈上一本书，市面上的确也有专讲 ADO 的书，但这并非本书的重点。我们虽然以 SQL 的方式利用 ADO 这个现成的组件来操作 Excel，但重点在于之后的种种 SQL 指令运用。

范例 1-2　用 ADO 技术读出 Excel 元数据表的内容

假设你已经找 meta data 的内容，并把它保存在文件 c:\cell_meta_data.xls 里，那么下面这段程序代码。

范例程序 2

```
Public Sub get_data_from_table()
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As Variant
    Dim loADODBRecordset As Variant

    lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                         "DBQ=c:\cell_meta_data.xls;" & _
                         "ReadOnly=True"
```

```
    lcCommandText = "select * from [cell_meta_data$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add

    Dim r, f As Integer
    r = 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
        Sheets(1).Cells(r, f + 1).Value =
loADODBRecordset.Fields(f).Name
    Next

    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
            Sheets(1).Cells(r, f + 1).Value =
loADODBRecordset.Fields(f).Value
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    Sheets(1).Activate
    Cells.Rows.AutoFit
    Cells.Columns.AutoFit
    Cells(1, 1).Select
End Sub
```

就可以把 c:\cell_meta_data.xls 文件中的内容以 SQL 的方式逐

个取出来。

注意：在本书以后的范例程序中，我们会多次用到 ADO，调用方式与本例几乎完全相同，惟一的差别是 lcCommandText 的内容会依你本身的需求有所改变。

范例程序关键代码点评

对于数据库应用不是那么熟悉的人，简单解释一下这几行指令的意义：

```
Dim lcConnectString, lcCommandText As String
Dim loADODBConnection As Variant
Dim loADODBRecordset As Variant
```

这是变量声明（如果你在菜单的“引用”中引用 ADO 的话，也可以把这两个 ADO 对象分别声明为 ADODB.Connection 及 ADODB.Recordset 的数据类型）。

```
lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
                     "DBQ=c:\cell_meta_data.xls;" & _
                     "ReadOnly=True"
```

虽然 c:\cell_meta_data.xls 是个 Excel 文件，但此时扮演一个数据库的身份，用数据库的方式去存取它时，要先打开数据库连接，这个字符串就是所谓的连接字符串，届时 ADO 就用这个字符串存取 c:\cell_meta_data.xls。

```
lcCommandText = "select * from [cell_meta_data$]"
```

这是 SQL 命令，cell_meta_data 是一个数据表，也就是

c:\cell_meta_ data.xls 这个文件里的（惟一的那个）工作表名称。如果存取的 xls 文件里不只一个工作表也没关系，反正 from 后面那个名称就是 xls 文件里的某一个工作表名称就对了。Excel 式的数据库比较特殊一些，要用类似于[cell_meta_data$]这种字符串，也就是，工作表名称前后加上中括号，还有一个“$”符号。

“select *”的意思是选择所有字段。还记得字段吗？对此例中的 c:\cell_meta_data.xls 而言，就是“序号”、“地址”、“数据类型”、“文本”等信息。

```
Set loADODBConnection = CreateObject("ADODB.Connection")
Set loADODBRecordset = CreateObject("ADODB.Recordset")
```

直接调用微软的 ADO 对象，这是一个现成的软件组件，你不必知道它安装在哪儿，只需调用就行了。

```
loADODBConnection.Open lcConnectionString
```

用于打开数据库连接。

```
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1
```

以上述的数据库连接执行数据库查询。后面的 3,1,1 这几个 Cursor 及 Lock 参数就不必理会了，我们的应用范例暂时还不必知道这些细节。其实，大多数的时候，这 3 个参数也可以省略，此处写出来是因为之后会有不是 3,1,1 的状况（那时会再做说明）。

```
r = 1
For f = 0 To loADODBRecordset.Fields.Count - 1
    Sheets(1).Cells(r, f + 1).Value = loADODBRecordset. Fields(f).Name
Next
```

执行完数据库查询后，loADODBRecordset 这个对象中已经装有查询出来的结果。loADODBRecordset.Fields.Count 可以知道（查询结果）有几个字段；loADODBRecordset. Fields(f).Name 则可以知道第 f 个字段的名称（从零算起，也就是 loADODBRecordset. Fields(3).Name 是第 4 个字段的名称）。

```
While Not loADODBRecordset.EOF
  r = r + 1
  For f = 0 To loADODBRecordset.Fields.Count - 1
      Sheets(1).Cells(r,f+1).Value=loADODBRecordset. Fields(f).Value
  Next

  loADODBRecordset.MoveNext
Wend
```

这是一个把查询结果逐个找出来的循环，loADODBRecordset.EOF 可以告诉我们是否已经到最后一个记录了（End Of File）。loADODBRecordset. Fields(f).Value 跟 loADODBRecordset.Fields(f). Name 很像，它的意思是取得第 f 个字段的内容（从零算起，也就是 loADODBRecordset. Fields(3). Value 是第 4 个字段的内容）。loADODBRecordset. MoveNext 则是移到下一个记录，在循环内一直 MoveNext，loADODBRecordset.EOF 这件事才会发生，不然就会变成无穷循环。

```
loADODBConnection.Close
```

关闭数据库连接。

以此例而言，由于我们是以只读模式打开 c:\cell_meta_data.xls（那个 3,1,1 参数，中间的那个 1 所代表的意义），所以其实（以此例而言）没关闭连接也无妨。但一般的习惯是最好关闭，c:\cell_meta_data.xls 才可以被其他程序存取。

用 ADO 技术存取数据库的编程步骤

好了，基本的数据库查询已经讲完了。简单整理一下之前讲的整个步骤：

（1）先把你的 Excel 数据的 meta data 写入 c:\cell_meta_data.xls 里。

（2）使用现成的 ADO 对象把该 c:\cell_meta_data.xls 所在的 meta data 按照你的查询条件取出来。

整个语义就是，（1）先把你的“所有”数据写入数据表；（2）再从该数据表取“想要”的数据。

其实步骤（2）可以达到的功能当然不止找数据这么简单而已，还可以计算、串联数据等，而这些操作都是通过简单而强大的 SQL 来实现。

先别担心 SQL，真的很简单，中学程度的英语+高中程度的数学就可以搞定。我听过一种说法是，有一般程序设计基础的人，一个下午就可以学会了（我以前教过我的女朋友 SQL，她花了两个钟头）。没有一般程序设计基础也别紧张，在大多数状况下，你会发现 SQL 指令像是一些很直观的英语。而即使 SQL 有很多功能，

也没规定你一定要全用上或用得很多才可以，实际上，一点点的 SQL 就可以发挥很大的效果了。

本书不会详细介绍 SQL，只会在用到时简单扼要解释用法，如果你想熟悉 SQL，网络上跟市面上都有一些很好的入门和高级参考资料，SQL 已经是个非常成熟的技术，基本上到处都找得到参考资料。

Excel 单元格地址与 SQL 查询范例

取出元数据的目的是什么？在 Excel 里做的很多操作说穿了都跟单元格直接相关，所以，只要能找出你想操作的单元格就能完成工作。那么单元格要怎么找呢？

技术点：Excel 单元格地址

像人的身份证号码一样，每一个 Excel 单元格都有一个、独一无二的地址，我们可以通过单元格地址来寻找单元格。单元格地址的具体形式可以是像“A1”、“DV100”这样的文本格式地址，也可以是像 R1C1 这种用行号列标表示的数字格式地址，示例如“R1C1”、“R100C126”等。在数字表示法中，R 代表 Row（行），C 代表 Column（列），这种地址表示法的重点在数字，不是 RC 这两个固定的字母。

找一群单元格，就是找到一群单元格所代表的地址。

这样明白了吗？我们的元数据不就存有每个单元格的文本格式地址跟数字格式地址吗？而找到元数据中的地址后，将之当

作 Range(<文本格式地址>)或 Cell(<数字格式地址的行号部分>,<数字格式地址的列标部分>)函数的参数，就可以找到需要的单元格了。

范例 1-3　列出 Excel 元数据表中的单元格地址

“范例程序 1-2”做的工作，是把所有的元数据不分青红皂白地全列出来，但我们现在要做的不是列出所有的元数据，而是要把单元格找出来，所以，让我们修改一下程序，变成“范例程序 1-3”的样子。

范例程序 1-3

```
Public Sub get_data_from_table()
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As Variant
    Dim loADODBRecordset As Variant

   lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                        "DBQ=c:\cell_meta_data.xls;" & _
                        "ReadOnly=True"

    lcCommandText = "select * from [cell_meta_data$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    While Not loADODBRecordset.EOF
        Debug.Print loADODBRecordset.Fields(“地址”).Value

        loADODBRecordset.MoveNext
```


```
    Wend

    loADODBConnection.Close
End Sub
```

请注意 Debug.Print 那一行，在真正应用这段程序时不会这样写，此处只是把它列出来，让你看到我们的确找到了想要的地址。真正应用时，我们会把 loADODBRecordset.Fields (“地址”).Value 的值传给 Range 对象当参数。

OK，接下来我们通过一个又一个的范例，来解释这种元数据的宏程序设计方式如何可以应用到各式各样的场合，而这些范例彼此看似都不相干，但却可以用同一种思维来看待，这就是抽象化的力量。

范例 1-4　找出数值小于 60 的单元格，并将其底色设置成红色

考虑图 1-3 所示的这些数据。

A	B	C
30	60	30
20	76	10
22	89	90
100	90	100
98	12	99
12	78	100
99	65	86
100	34	0
80	100	100
74	99	88

图 1-3

我们要在这些数据中找一群数值小于 60 的单元格，并将其单元格背景色设置成红色。

这个简单的范例用传统的宏写法是这样：

```
Public Sub RunMacro()
    Dim loCell As Range
    For Each loCell In ActiveSheet.UsedRange.Cells
        If loCell.Value < 60 Then
            loCell.Interior.ColorIndex = 3
        End If
    Next
End Sub
```

如果用元数据的方法，则是这样：

找出小于 60 的数字并标成红色.xls

```
Public Sub RunMacro()
    run ("'" + ActiveWorkbook.Path + "\找出 meta_data.xls'!
Module1.ActiveWorkSheet_To_Table")

    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As Variant
    Dim loADODBRecordset As Variant

    lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                         "DBQ=c:\cell_meta_data.xls;" & _
                         "ReadOnly=True"

    lcCommandText = "select * from [cell_meta_data$] where 数字 < 60"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Dim loRange As Range
    While Not loADODBRecordset.EOF
```


```
        Set loRange = ActiveSheet.Range(loADODBRecordset.
Fields("地址").Value)
        loRange.Interior.ColorIndex = 3

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    Workbooks("找出 meta_data.xls").Close
End Sub
```

其实，上述程序代码不过是把先前提过的程序代码兜起来，再稍稍改一下罢了。请注意粗体字的部分。

我知道你大概要抗议了，花了这么多时间，结果这比传统方法写起来更费劲、执行（有时可能）也更费时，为什么要用呢？

原因就在于“架构”、“语义”、“隐喻”这些概念。用元数据的方法，可以让我们处理许多“看似不相干”的问题时都用“同一种方法”。况且，你自己真正写的程序代码其实只有粗体字那几行而已，其他都是很固定的、可以不必写而直接引用的现成东西，所以并不比传统的方法更复杂。

再者，这是一个很基本的小范例，如果复杂一些，当更可以显示出元数据方法的威力。请继续往下瞧瞧啰！

范例程序关键代码点评

虽然前面已经讲了很多数据库跟数据表的概念，这里还是稍微解释一下这几行粗体字的关键程序代码的意义。

```
lcCommandText = "select * from [cell_meta_data$] where 数字 < 60"
```

“select*”先前已解释过，是找出所有字段之意，where 则是“条件为……”的意思。之所以可以直接下达“数字 < 60”这样的条件，是由于我们已经把所有的数据都写入元数据表，而 cell_meta_data 这个数据表（Table）中有“数字”这个字段（field），所以我们可直接利用 SQL 把“数字 < 60”的记录都找出来。找出来之后再利用：

```
loADODBRecordset.Fields("地址").Value
```

就可以在

```
    While Not loADODBRecordset.EOF
    Wend
```

这个循环中找到一个一个记录里“地址”这个字段的值，然后利用这行指令：

```
    Set loRange = ActiveSheet.Range(loADODBRecordset.Fields
("地址").Value)
```

把这个值传给 Excel VBA 中的 Range 对象当参数，就可以找到所有符合“数字 < 60”条件的 Cell(s)。找到之后，就可以直接指定 Cell 的颜色了：

```
loRange.Interior.ColorIndex = 3
```

当然，别忘了一个记录（定位到一个 Cell）做完后，要把记

录指针移到下一个（利用 loADODBRecordset.MoveNext 指令），不然会变成死循环的。

如前所提，在 Excel 里的活动既然绝大部分都跟 Cell 有关，所以我们大可重复不断地利用这种方法，顶多就是在 SQL 条件那边做做变化，就可以轻易地定位到（找到）我们想要的（不同条件下的）单元格。而找到单元格之后，简单一点的，可以像此例直接设置单元格的属性，复杂一点的，还可以把单元格这个对象丢给我们自定义的函数，像拆分中文英语数字、读取字符、转换等（这些自定义函数在本书第 3 章“单元格自定义函数”中有些范例可供参考）。

把单元格对象传给函数的技术

那要怎么把单元格对象传给一个函数呢？很简单，就跟传数字或传字符串一样，传数字的程序代码（pnNumber 是自己取的参数名称）：

```
Public Sub 我的自定义函数(ByVal pnNumber As Integer)
    pnNumber = pnNumber + 1
    .
    .
    .
End Sub
```

传整单个单元格对象（Object）的程序代码：

```
Public Sub 我的自定义函数(ByVal poCell As Range)
    poCell.Interior.ColorIndex = 3
    .
```

```
        .
        .
    End Sub
```

或是这样，传整个 Range 进去，之后就可以找到该 Range 里的每一个单元格：

```
    Public Sub 我的自定义函数(ByVal poRange As Range)
        Dim loCell As Range
        For Each loCell In poRange.Cells
            loCell.Interior.ColorIndex = 3
        Next
        .
        .
        .
    End Sub
```

就传 Range 而言，你可以发现，其实后面那一种做法比较通用（虽然程序代码多了寥寥几行），因为整个 Range 也可以只有一个 Cell，所以循环也只会进行一次而已，本书第 3 章将采用此方式。

除了 Cell，Range 也是另一个有力的隐喻，Range 是大区域的 Cell，更精确一点说应该是可大可小，Range 可以小到一个 Cell，可以大到整个工作表，可以指向一列、一行、几列、几行、一块选取的区域、几块不连续选取的区域等。当你要以上述这些 Range 为单位处理数据时，Range 就是一个极有力的隐喻，稍后要介绍的“删除空白列”范例应该可以让你体会到这一点。

OK，马上来看看第 2 个例子。

范例 1-5 找出数值小于 60 的单元格，将其底色设置成红色，且添加标题文本

细心的读者可能已经看出来，上例可以实现“找出哪些学生的哪些科目不及格”功能，但还需要添加标题，如图 1-4 所示。

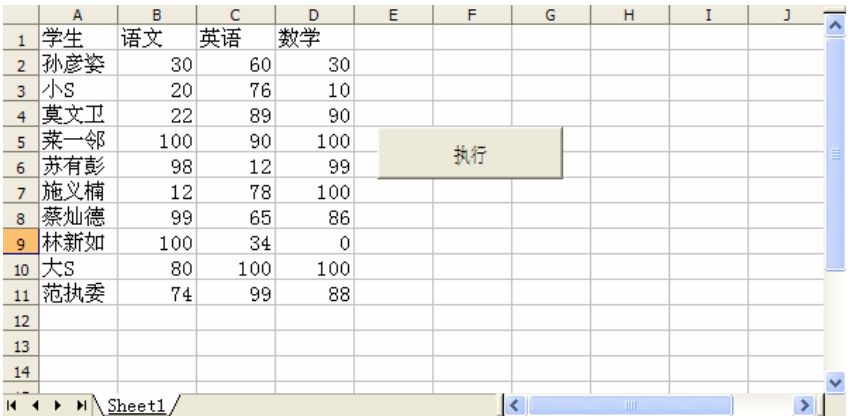


图 1-4

用传统的方法，程序要改成这样：

```
Public Sub RunMacro()
    ActiveWorkbook.Save
    Dim loCell As Range

    For Each loCell In ActiveSheet.UsedRange.Cells
        If loCell.Row > 1 And loCell.Column > 1 Then
            If loCell.Value < 60 Then
                loCell.Interior.ColorIndex = 3
            End If
        End If
    Next
End Sub
```

请注意粗体字的部分，这是因为分数所在的区域变成是行号

与列标都必须大于 1 了。

如果用元数据的写法呢？只需把 SQL 条件再加上一些，变成：

```
    lcCommandText = "select * from [cell_meta_data$] where 数字 < 60 and 行号 > 1 and 列标 > 1"
```

就 OK 了。请参照范例文件<找出小于 60 的数字并标成红色_有标题文本.xls>。

小贴士：查询条件举一反三

到目前为止，SQL 的使用都很简单，下面来举一反 N 一番：

- 找数字为 90：数字 = 90。
- 找数字为 90 至 100：数字 >= 90 and 数字 <= 100。

（注意，不是这种写法：数字 >= 90 and <= 100，条件是各自独立的，这是初学者常犯的错误之一）。

- 找奇数：数字 mod 2 = 1（也就是数字除以 2 的余数等于 1）。
- 找偶数：数字 mod 2 = 0（也就是数字除以 2 的余数等于 0，整除啦！）。
- 找 3 的倍数：数字 mod 3 = 0。
- ……

以上都是数字方面的搜索，如果是文本，则可以：

- 找左边 3 个字符为 THE 的文本：left(文本, 4) = '"THE'。你没看错，左边有 3 个单引号，而且第 2 个参数是 4 而不是 3。这是因为我们在元数据中存的是“'THE”，而 SQL 中对于文本类型的条件式必须加上单引号，在文本中找的内容含有单引号时，就必须用两个单引号来表示，而“THE”加上一个单引号后，长度是 4 而不是 3。
- 找第 2 个字符~第 4 个字符为 THE 的文本：mid(文本, 3, 3) = 'THE'。因为不是从第一个字符开始找，所以没有两个单引号的问题。第 2 个字符~第 4 个字符的意思是：从第 3 个字符开始（因为文本内容的第一个字符是单引号）长度为 3 的字符串值。长度为 3 就是“第 2 个字符、第 3 个字符、第 4 个字符”。
- 找右边 3 个字符为 THE 的文本：right(文本, 3) = 'THE'。同上，因为不是从第一个字符开始找，所以没有两个单引号的问题。
- 找文本长度为 3：len(文本) = 3。
- ……

由于 meta data 存有颜色，所以还可以按照颜色找：

- 找单元格颜色为红色的：单元格颜色 = 3（红色在 Excel VBA 的代码）。
- 找文本颜色为红色的：字体颜色 = 3。
- ……

找出了数值小于 60 的单元格，难道就只能设置其底色为红色吗？当然不只如此，既然已经定位出（找出）了单元格，单元格能做的事，诸如设置字体属性、内容的删除、移动、复制、修改、剪切、加减乘除、加上批注、设置显示格式等等，自然

也都可以做。

再举一反 N 一番，难道只能以数值来进行单元格的搜索吗？当然也不止如此，可以找出字体颜色为红色的单元格、列标为 2 的单元格、列标为 2 且（and）行号介于 1~100 的单元格、文本长度不为 0 的单元格、数据类型为数字的单元格等。

跨工作表、跨工作簿的查询

Excel 自带的查询功能无法跨越多个工作表进行，而用元数据的方法不但可以跨越工作表查找、替换，还可以跨越工作簿呢！

技术点：工作簿、工作表的通用表示方法与跨表查询

先来看一下，我们在前面程序的循环中以 Range 对象定位单元格的方式：

```
    Set loRange = ActiveSheet.Range(loADODBRecordset.Fields
("地址").Value)
```

其中，ActiveSheet 代表当前打开的工作表。之所以这么做，是因为我们在读取元数据的过程中，只针对目前工作表。

若想跨表、跨工作簿查询，需要关注的是工作表的通用表示方法，也就是如何替换 ActiveSheet 的问题，下面的程序对此作了更改，添加了部分代码，就实现了跨表查询。

范例 1-6　从所有 Excel 工作簿提取元数据

把程序稍稍改一下（请注意粗体字部分）。

范例程序 1-4

```
Public Sub AllWorkBook_To_Table()
    Application.ScreenUpdating = False

    Dim loWorkBook As Workbook
    Dim loSheet As Worksheet
    Dim lnSerialNumber As Long
    Dim lcText As String
    Dim lcTextValue As String
    Dim loCell As Range
    Dim ldDate, ltTime, luValue As Variant
    Dim lcValue As String

    lnSerialNumber = 0

    Open "c:\cell_meta_data.csv" For Output As #1

    ' 写入字段标题
    Write #1, _
    "序号", _
    "地址", _
    "数据类型", _
    "文本", _
    "文本长度", _
    "数字", _
    "日期", _
    "时间", _
    "列标", _
    "行号", _
    "字体名称", _
    "字号", _
```

```
    "字体颜色", _
    "单元格颜色", _
    "工作表", _
    "工作簿", _
    "文件名称", _
    ""

    For Each loWorkBook In Application.Workbooks
        If loWorkBook.Name <> "找出 meta_data.xls" Then
            For Each loSheet In loWorkBook.Sheets
                If loSheet.Type = -4167 Then
                    loSheet.Activate
                   For Each loCell In loSheet.UsedRange.Cells
                        With loCell
                            lcTextValue = Trim(.Text)
                            lcText = "'" + lcTextValue
                            luValue = .Value

                          lnSerialNumber = lnSerialNumber + 1

                            ldDate = ""
                            ltTime = ""
                            lcValue = ""

                            If IsDate(lcTextValue) And Not
IsNumeric(lcTextValue) Then
                           ldDate = CStr(DateValue(lcTextValue))
                           ltTime = Replace(Replace(DateValue
(lcTextValue) & TimeValue(lcTextValue), "上午", ""), "下午", "")
                            End If

                        If IsNumeric(.Value) And Len(lcTextValue)
> 0 Then
                            lcValue = Trim(Str(Val(.Value)))
                            End If
```


```
                        Write #1, _
                        lnSerialNumber, _
                        .Address, _
                        VarType(loCell), _
                        lcText, _
                        Len(lcTextValue), _
                        lcValue, _
                        ldDate, _
                        ltTime, _
                        .Column, _
                        .Row, _
                        .Font.Name, _
                        .Font.Size, _
                        .Font.ColorIndex, _
                        .Interior.ColorIndex, _
                        loSheet.Name, _
                        loSheet.Parent.Name, _
                        loSheet.Parent.FullName
                    End With
                Next
            End If
        Next
    End If
Next

Close #1

If Dir("c:\cell_meta_data.xls") <> "" Then
    Kill "c:\cell_meta_data.xls"
End If

Workbooks.Open ("c:\cell_meta_data.csv")
ActiveWorkbook.SaveAs "c:\cell_meta_data.xls", xlExcel9795
ActiveWorkbook.Close
Kill "c:\cell_meta_data.csv"
```

```
    Application.ScreenUpdating = True
End Sub
```

注意，不要把范例 1-1 中生成的 meta_data.xls 也加到元数据中。

增加以粗体显示的几行代码，在查询元数据时就可以针对“工作表”、“工作簿”甚至“文件名称”进行搜索了。

范例 1-7　跨工作表搜索小于 60 的数字并标成红色

我们把这段程序也放进<找出 meta_data.xls>里。马上来修改先前的例子：范例 1-4，请注意粗体字部分。

找出小于 60 的数字并标成红色_跨工作表.xls

```
Public Sub RunMacro()
    Run ("'" + ActiveWorkbook.Path + "\找出
meta_data.xls'!Module1.AllWorkBook_To_Table")

    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As Variant
    Dim loADODBRecordset As Variant

   lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                        "DBQ=c:\cell_meta_data.xls;" & _
                         "ReadOnly=True"

    lcCommandText = "select * from [cell_meta_data$] where 数
字 < 60 and 行号 > 1 and 列标 > 1"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")
```


```
    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Dim loWorkBook As Workbook
    Dim loSheet As Worksheet
    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loWorkBook = Workbooks(loADODBRecordset.Fields("
工作簿").Value)
        Set loSheet = loWorkBook.Sheets(loADODBRecordset.Fields
("工作表").Value)
        Set loRange = loSheet.Range(loADODBRecordset.Fields
("地址").Value)
        loRange.Interior.ColorIndex = 3

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    Workbooks("找出 meta_data.xls").Close
End Sub
```

有了工作表及工作簿的元数据，要找出两个工作表同一单元格地址的内容是否有所不同就容易得多了：

```
    select 地址, 文本, count(*) from [cell_meta_data$]
    group by 地址, 文本 having count(*) <> 2
```

group by 的意思马上就会谈到。如果两个工作表的同一地址的内容相同的话，那么，该“地址+内容”这个值组应该会有两个，不是两个就表示彼此不同。

知识类：SQL 各件查询技术

我们提过，SQL 不只可以用来搜索数据，还可以计算。所谓的“空白行”，就是按照行号统计其内 Cell(s)的文本总长度，总长度为 0 的就代表该行无数据。空白列的道理也是一样。

按照行号统计其内单元格的文本总长度的 SQL 语法是：

```
select 行号, sum(文本长度) from [cell_meta_data$] group by 行号
```

group by 就是“按照行号统计出……，或搜索出……”，统计方面，可以统计总和、平均、（有些数据库引擎的 SQL 还支持标准差）、笔数；搜索方面，则可以搜索出最大值、最小值。最大值就是该行号在“xx 方面”的最大值，例如最大文本长度、最大日期等。

上面会计算并搜索出每一行的文本总长度，如果要只挑那些文本总长度为 0 的行，则要加个条件：

```
select sum(文本长度) from [cell_meta_data$] group by 行号
having sum(文本长度) = 0 order by 行号 desc
```

having 跟 where 不一样，它是针对 group by 而来的“两次条件设置”，像 sum、avg、count、max、min 这些因 group by 而出现的“聚合函数”的条件设置要用 having（虽然有些数据库引擎也可以把原先用在 where 条件、不属于聚合函数的字段放在 having 子句中）。having 可以想成是，找出数据并统计之后，那些统计结果还有着……的条件。

最后用了一个 order by，排序之意，它是有作用的。因为找出来的行号要删除时必须从大的行号开始删（所以用 desc 排序，desc

为 descending，递减之意），这样才不会因为第 3 行被删后，原来要删的第 10 行已经变成第 9 行而导致删错行。

最后，请注意一个地方，由于我们定位的 Range（不论是一个 Cell、一个数据区域、一行、一列）必须在前面加上工作簿及工作表名称，所以，必须把工作簿及工作表这两个字段也一并找出来，如下：

```
    select 工作簿, 工作表, 行号, sum(文本长度) from [cell_meta_
data$] group by 工作簿, 工作表, 行号 having sum(文本长度) = 0 order
by 行号 desc
```

当进行 group by 操作时，必须把聚合函数以外的字段都纳入 group by 的清单中，此处的聚合函数是 sum(文本长度)，因此要 group by 工作簿，工作表，行号。

范例 1-8　删除空白行（或空白列）、空白工作表

本范例介绍一个看似跟数据处理无关的范例，但从本质上讲，还是可以“泛数据化”一番，把元数据用上。而且前面介绍的 SQL 查询技术都能用上。

找到要删除的行号后，程序稍稍有点变动，因为现在要找的是行号而非地址（请注意粗体字部分）：

删除空白行.xls

```
    While Not loADODBRecordset.EOF
        Set loWorkBook = Workbooks(loADODBRecordset.Fields("
工作簿").Value)
        Set loSheet = loWorkBook.Sheets(loADODBRecordset.Fields
("工作表").Value)
```

```
        Set loRange = loSheet.Rows(loADODBRecordset.Fields("
行号").Value)
        loRange.Delete

        loADODBRecordset.MoveNext
    Wend
```

同理，删除空白列就是把上述的行号统统改成列标、还有 Rows 改成 Columns 就 OK 了。而要标示行或列的话，只需把 Delete 方法改成：

```
    Application.Union(Application.Selection,
    Rows(loADODBRecordset.Fields("行号").Value)).Select
```

即可。

那空白的工作表呢？一张工作表如果没有任何数据（但有可能有格式设置），表示该工作表所属的每个 Cell(s)，其文本长度都是 0，也就是文本总长度为 0。以 SQL 表示，就是找出 group by 该工作表、sum(文本长度) = 0 的（工作表）：

删除空白工作表.xls

```
    lcCommandText = "select 工作表, sum(文本长度) from
[cell_meta_data$] group by 工作表 having sum(文本长度) = 0"

    Dim loSheet As Worksheet

    Application.DisplayAlerts = False
    While Not loADODBRecordset.EOF
        Set loSheet = Sheets(loADODBRecordset.Fields("工作表
").Value)
        loSheet.Delete

        loADODBRecordset.MoveNext
```


```
    Wend
    Application.DisplayAlerts = True
```

由于删除工作表时，Excel 会提示确认，因此先把这种提示关掉：

```
    Application.DisplayAlerts = False
```

之后再恢复（Application.DisplayAlerts = True）。

group by 的用处极多，随便举一个，假如要找出某行最后一个 Cell：

```
    lcCommandText = "select 行号, max(列标) as 最后一列 from
[cell_meta_ data$] group by 行号"
```

一行的最后一个 Cell 不就是该行最大列标的 Cell 吗？找出某列最后一个 Cell 把上述行列对调即可。

范例 1-9　从 Excel 工作簿提取需要的数据（或 Cells）

有时候，拿到某些工作表，我们会想从中找些数据，而不去动工作表本身。这种动作可粗分为两类，第一种是找一些 Cells 的内容，另一种则是把两张（或更多张）工作表彼此之间的数据按照某个字段关联起来。

先说第一种。还记得范例程序 1-2 吗？从元数据中逐个读取数据。那时是这样写的：

```
    select * from [cell_meta_data$]
```

没有任何 where 条件，所以就是逐个找。如果想找奇数行的数据，就是：

```
select distinct 行号 from [cell_meta_data$] where 行号 mod 2 = 1
```

然后在循环内把找到的行号当作 Rows 的参数，把原工作表中的那些行号的数据 copy 到一张新的工作表，这样就不会动原先的工作表了。

distinct 是“不重复”之意，因为行号为奇数的 cells 有很多，彼此会有重复的现象，例如 A1、B1、C1、D1 同属奇数列，但行号都是 1，如果不加上 distinct，就会有 4 个 1 传给 Rows，那 Rows(1) 就会被 copy 4 次到新的工作表。

关键程序代码如下（请注意粗体字部分）：

取奇数行的数据.xls

```
    lcCommandText = "select distinct 行号 from
[cell_meta_data$] where 行号 mod 2 = 1"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add
    Dim r As Integer
    r = 0

    Dim loRange As Range
    While Not loADODBRecordset.EOF
```

```
        Set loRange = Sheets(2).Rows(loADODBRecordset.Fields
("行号").Value)
        r = r + 1
        loRange.Copy (Sheets(1).Rows(r))

        loADODBRecordset.MoveNext
    Wend
```

Sheets.Add 之后，被 Add 的 Sheet 会跃升为第 1 张工作表，而原先的工作表就成了 Sheets(2)了。

所以，取偶数行/列、或取行/行号为 N 的倍数的行/列，应该难不倒你了吧！

distinct 很常用，计算一张工作表内共有几行，就是找出不重复的：

```
    lcCommandText = "select distinct 行号 from [cell_meta_ data$])"
```

如果工作表有 10 行（每行的 Cell 数不同），这样会找出 10 个记录。我们必须在之后的那个 While 循环中以一个变量来计算才能得知共有 10 行。

SQL 的威力是以“泛 Table 化”来描述真实世界，SQL 的工作环境就是“面向 Table”。SQL 取 Table 中的数据，取出来的结果也是一个 Table。这意味着我们可以对取出来的数据进行两次（甚至 N 次）读取：

```
    lcCommandText = "select count(*) as 行数 from (select
distinct 行号 from [cell_meta_data$])"
```

count(*)是读取个数之意，如此就可以一次搞定了，之后即可直接取值：

```
loADODBRecordset.Fields("行数").Value
```

范例 1-10　拆分单元格的内容

Cell 的数据如下：

```
F123456789
H256102411
```

如果想拆分一下，变成：

```
F   123456789
H   256102411
```

那 SQL 可以写成这样：

```
    select 地址, mid(文本, 2, 1) as 第 1 部分的文本, mid(文本, 3)
as 第 2 部分的文本 from [cell_meta_data$]
```

mid 函数是从第 a 个字符读取某段文本，长度为 b，例如：

```
    mid('A123', 1, 2) = 'A1'      从第 1 个字符开始，读 2 个字符
    mid('A123', 2, 1) = '1'       从第 2 个字符开始，读 1 个字符
    mid('A123456', 2) = '123456' 从第 2 个字符开始，读取到最后（不
    管有几个字符）
```

那为什么第 1 部分的文本要从第 2 个字符开始读取呢？因为我们的元数据中，文本存的都是“'”加上原内容。

在 select 时，“文本”字段加上 mid 函数后，字段名称就变成了一种“运算结果”而非原先的“文本”字段，这时，可以加上

as <自己取的字段名称>，否则系统会自己给一个类似 Expr1、Expr2、Expr3 这样的列名。用自己取的名字，之后在提取时不会忘记，也比较有“语义”。

关键程序代码如下（请注意粗体字部分）：

拆分 Cell 的内容.xls

```
    lcCommandText = "select 地址, mid(文本, 2, 1) as 第 1 部分的文本, mid(文本, 3) as 第 2 部分的文本 from [cell_meta_data$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loRange = Sheets(1).Range(loADODBRecordset.Fields ("地址").Value)
        loRange.Value = "'" + loADODBRecordset.Fields("第一部分的文本").Value
        loRange.Offset(0, 1).Value = "'" + loADODBRecordset.Fields("第二部分的文本").Value

        loADODBRecordset.MoveNext
    Wend
```

loRange.Offset(0,1)是 loRange 地址的右边，所谓右边就是行号不变，列标加 1。当然，此范例中假设你的原始数据只在第一列有。

这几个范例（乃至于本书的许多范例）或许都不具备“实际”效益，它们是给你熟悉 SQL 用的。然而，熟悉这些基本功，有助于你解决其他更复杂的问题。这些范例也说明了，用一种模式就可以解决许多看似不相干的问题，而这正是本书的最大目的与特色。

范例 1-11　文本转换成日期

由其他格式的文件（特别是文本文件）导入数据至 Excel 时，日期格式常会因为 Excel“认不得”而变成文本或数字。

假设日期是“年月日，年为 4 个字符的公元年格式”，可以用 mid 函数分别取得年月日，然后将字符串相加就可以变成日期了。

关键程序代码如下（请注意粗体字部分）：

文本转换成日期.xls

```
    LcCommandText = "select 地址, mid(文本, 2, 4) + '/' + mid(文本, 6, 2) + '/' + mid(文本, 8, 2) as 转换过的日期 from [cell_meta_data$]"

    Sheets.Add

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loRange = Sheets(1).Range(loADODBRecordset.Fields("地址").Value)
        loRange.Value = loADODBRecordset.Fields("转换过的日期").Value

        loADODBRecordset.MoveNext
    Wend
```

如果你只想找 12 月份的日期，则是：

```
    select 地址, mid(文本, 2, 4) + '/' + mid(文本, 6, 2) + '/'
+ mid(文本, 8, 2) as 转换过的日期 from [cell_meta_data$] where
val(mid(文本, 6, 2)) = 12
```

val(mid(文本, 6, 2)) = 12 的意思是，月份转成数值后等于 12。

当然，上例用这样的方式找出来后，12 月份的单元格不会放在一起，而是会按照它们在原工作表中的地址复制到 Sheets(1)，如果要连着放，就自定义一个变量累加，不要用：

```
    Set loRange = Sheets(1).Range(loADODBRecordset.Fields("
地址"). Value)
```

就 OK 了，关键程序代码如下（请注意粗体字部分）：

文本转换成日期_2.xls

```
    lcCommandText = "select 地址, mid(文本, 2, 4) + '/' + mid(文
本, 6, 2) + '/' + mid(文本, 8, 2) as 转换过的日期 from
[cell_meta_data$] where val(mid(文本, 6, 2)) = 12"

    Sheets.Add
    Dim r As Integer
    r = 0

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        r = r + 1
        Set loRange = Sheets(1).Cells(r, 1)
       loRange.Value = loADODBRecordset.Fields("转换过的日期").Value

        loADODBRecordset.MoveNext
    Wend
```

范例 1-12　文本转换成数字

如果你收到一份从别的软件导出到 Excel 的数据，而数字部分类似：

12,345,678

如果你是以手动方式转换的话，正常状况下 Excel 会把这样的文本当作数字看待，但如果不是如此，这种文本可以用这样的方式转换：

文本转换成数字.xls

```
    lcCommandText = "select 文本 from [cell_meta_data$]"

    Sheets.Add
    Dim r As Integer
    r = 0

    While Not loADODBRecordset.EOF
        r = r + 1
       Sheets(1).Cells(r, 1) = Val(Replace(Mid(loADODBRecordset.
Fields("文本").Value, 2), ",", ""))

        loADODBRecordset.MoveNext
    Wend
```

范例 1-13　随心所欲的自动填充

Excel 的自动填充只能做到一个方向（上下左右）。

如果想随心所欲地填入某个值到某些单元格，不限定一个方向，也不限定一块区域，该如何做？

由于元数据使用 UsedRange 当作数据源，所以先把你想填充的区域的左上角单元格跟右下角的单元格填入随便一个值，例如“C9”跟“D26”，这样 UsedRange 可以找到“C9:D26”的所有单元格。找到之后，就可以在循环内按你的想法填入内容了。

关键程序代码如下（请注意粗体字部分）：

自动填充.xls

```
    lcCommandText = "select 地址 from [cell_meta_data$]"

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Range(loADODBRecordset.
Fields("地址").Value)
        loRange.Value = 100

        loADODBRecordset.MoveNext
    Wend
```

也可以跟 Excel 的填充数列一样，按等差数列填充：

自动填充数列.xls

```
    lcCommandText = "select 地址 from [cell_meta_data$]"

    Dim s As Integer
    s = 0

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Range(loADODBRecordset.
Fields("地址").Value)
        loRange.Value = s
        s = s + 2

        loADODBRecordset.MoveNext
    Wend
```

等比数列也是同理，就不赘述了。

还可以变化出其他花样，例如：

填充不连续区域：

自动填充数列_不连续区域.xls

```
lcCommandText = "select 地址 from [cell_meta_data$] where 行号 mod 2 = 0"
```

有东西就跳过：

自动填充数列_有东西就跳过.xls

```
lcCommandText = "select 地址 from [cell_meta_data$] where 文本长度 = 0"
```

超过某数就重来：

自动填充数列_超过某数就重来.xls

```
    lcCommandText = "select 地址 from [cell_meta_data$]"

    Dim s As Integer
    s = 1

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Range(loADODBRecordset.
Fields("地址").Value)
        loRange.Value = s
        If s = 20 Then
            s = 1
        Else
            s = s + 1
        End If

        loADODBRecordset.MoveNext
    Wend
```

从上到下，然后再从左到右的“直式填充”方式：

自动填充数列_直式填充.xls

```
    lcCommandText = "select 地址 from [cell_meta_data$] order by 列标, 行号"

    Dim s As Integer
    s = 1

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Range(loADODBRecordset.Fields("地址").Value)
        loRange.Value = s
        s = s + 1

        loADODBRecordset.MoveNext
    Wend
```

order by 是“排序”之意，不同的排序方式可以改变“访问 Cell 的顺序”。

跨工作表填充也 OK，跟先前的范例结合一下，找出 meta data 时指定要跨工作表：

自动填充_跨工作表.xls

```
        Run ("'" + ActiveWorkbook.Path + "\找出meta_data.xls'!Module1.AllWorkBook_To_Table")

    lcCommandText = "select * from [cell_meta_data$]"

    Dim s As Integer
    s = 1

    Dim loWorkBook As Workbook
```

```
    Dim loSheet As Worksheet
    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loWorkBook = Workbooks(loADODBRecordset.Fields("
工作簿").Value)
        Set loSheet = loWorkBook.Sheets(loADODBRecordset.
Fields("工作表").Value)
        Set loRange = loSheet.Range(loADODBRecordset.
Fields("地址").Value)

        loRange.Value = s
        s = s + 1

        loADODBRecordset.MoveNext
    Wend
```

执行此范例时，多打开几个工作簿（记得先指定 UsedRange，然后存盘），就不只是跨工作表，而是跨工作簿及工作表的自动填充了。

自动填充也不只是这样而已，我们可以把这个功能抽象化，“找出一堆符合你指定条件的 Cell(s)，然后设置成某个值”。这不也符合之前所提到的，在 Excel 里，很多事情都不外乎是“找到一堆符合条件的 Cell”，然后“对那些找到的 Cell 做一些事”吗？

感觉到 SQL 的威力了吧？把数据整整齐齐、有秩序的放在一起，就能以一种有秩序的方式来操作它。这也是“理性”的力量。

范例 1-14　随心所欲的运算

数字的运算，有求数字总和、求平均值、求最大值、求最小值、计数等，但 Excel 的运算限于连续区域，下面的程序可以在任何选定的单元格上进行各种数字运算。

数字运算.xls

```
    lcCommandText = "select sum(数字) as 数字总和, avg(数字) as
数字平均, max(数字) as 最大数字, min(数字) as 最小数字, count(*) as
单元格数目 from [cell_meta_data$]"


    While Not loADODBRecordset.EOF
        MsgBox "数字总和: " + Str(loADODBRecordset.Fields("数
字总和").Value) + Chr(13) + _
                "数字平均: " + Str(loADODBRecordset.Fields("
数字平均").Value) + Chr(13) + _
                "最大数字: " + Str(loADODBRecordset.Fields("
最大数字").Value) + Chr(13) + _
                "最小数字: " + Str(loADODBRecordset.Fields("
最小数字").Value) + Chr(13) + _
                "单元格数目: " + Str(loADODBRecordset.Fields("
单元格数目").Value) + Chr(13) _
                , vbOKOnly, "数字信息"

        loADODBRecordset.MoveNext
    Wend
```

由于没有 group by，上述的 SQL 执行结果只会有一个记录。这就是说，还可以按照行号（或列标，或你想 group by 的字段）统计上述加总、平均等。这样就可能不止一个记录了：

数字运算_按行号统计.xls

```
    lcCommandText = "select 行号, sum(数字) as 数字总和, avg(数
字) as 数字平均, max(数字) as 最大数字, min(数字) as 最小数字,
count(*) as 有数字的单元格数目 from [cell_meta_data$] where 文
本长度 > 0 group by 行号"


    Sheets.Add
    Dim r As Integer
    r = 1
```

```
    Sheets(1).Cells(r, 1) = "行号"
    Sheets(1).Cells(r, 2) = "数字总和"
    Sheets(1).Cells(r, 3) = "数字平均"
    Sheets(1).Cells(r, 4) = "最大数字"
    Sheets(1).Cells(r, 5) = "最小数字"
    Sheets(1).Cells(r, 6) = "有数字的单元格数目"

    While Not loADODBRecordset.EOF
        r = r + 1

        Sheets(1).Cells(r, 1) = loADODBRecordset.Fields("行
号").Value
        Sheets(1).Cells(r, 2) = loADODBRecordset.Fields("数
字总和").Value
        Sheets(1).Cells(r, 3) = loADODBRecordset.Fields("数
字平均").Value
        Sheets(1).Cells(r, 4) = loADODBRecordset.Fields("最
大数字").Value
        Sheets(1).Cells(r, 5) = loADODBRecordset.Fields("最
小数字").Value
        Sheets(1).Cells(r, 6) = loADODBRecordset.Fields("有
数字的单元格数目").Value

        loADODBRecordset.MoveNext
    Wend
```

加上“文本长度 > 0”条件，可以确保不会找到空白的单元格，而“数字总和”除以“单元格数目”也才会等于“数字平均”。

举一反三：数字运算的几种变化条件

加些 where 条件变化一下，就可以只运算偶数、奇数、介于某个区域的数值、正数，或负数……

数字运算_只算正数.xls

```
lcCommandText = "select sum(数字) as 数字总和, avg(数字) as 数字平均, max(数字) as 最大数字, min(数字) as 最小数字, count(*) as 单元格数目 from [cell_meta_data$] where 数字 > 0"
```

- 偶数：where 数字 mod 2 = 0。
- 奇数：where 数字 mod 2 = 1。
- 介于某个区域的数值：where 数字 >= 10 and 数字 <= 100。
- 负数：where 数字 < 0。
- 10 这个数跳过：where 数字 <> 10。
- 10 和 20 都跳过：where 数字 <> 10 and 数字 <> 20。
- 加总时，负数当作是正数（绝对值）：select sum(abs(数字)) as 数字总和……
- ……

SQL 中有一种 top 的功能，也就是找出“前几个”记录。这个“前几个”，可以是明白地指定“前几个”，也可以，以百分比的方式”指定，例如找出符合条件的前 20%的记录。

所以，如果想找出“前 3 大数字”（前 500 大企业……），就是把数字做降序排序，然后找 top 3 个的意思：

找出前 3 大数字.xls

```
    lcCommandText = "select top 3 数字 from [cell_meta_data$] order by 数字 desc"

    Sheets.Add
    Dim r As Integer
    r = 0
```

```
    While Not loADODBRecordset.EOF
        r = r + 1
        Sheets(1).Cells(r, 1) = loADODBRecordset.fields("数
字").Value

        loADODBRecordset.MoveNext
    Wend
```

还有一种常见的需求是“分组统计”，也就是各个数字（或文本、日期……）出现过几次：

数字运算_分组统计.xls

```
    lcCommandText = "select 数字, count(*) as 出现几次 from
[cell_meta_data$] group by 数字"

    Sheets.Add

    Dim r As Integer
    r = 1

    Sheets(1).Cells(r, 1) = "数字"
    Sheets(1).Cells(r, 2) = "出现几次"


    While Not loADODBRecordset.EOF
        r = r + 1
        Sheets(1).Cells(r, 1) = loADODBRecordset.fields("数字").Value
        Sheets(1).Cells(r, 2) = loADODBRecordset.fields("出
现几次").Value

        loADODBRecordset.MoveNext
    Wend
```

如果要计算日期（文本、时间……）出现过几次，把上述 SQL 的数字改成日期即可。

分组统计的另一个意思就是“找出重复数据”，所谓重复，就是分组统计时出现次数超过 1 次。反之，找出不重复的数据，就是分组统计时，出现次数为 1 次的那些数据。不过找出不重复的数据不必要还用 group by 这么麻烦，因为 SQL 有个 distinct 关键字，可以直接去掉重复的数据：

去掉重复的数据.xls

```
lcCommandText = "select distinct 数字 from [cell_meta_data$]
where 文本长度 > 0 order by 数字"
```

范例 1-15　原单元格的值加减乘除某个数

原 Cell 的值加减乘除某个数.xls

```
    lcCommandText = "select 地址, 数字 + 10 as 新数字 from [cell_meta_data$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loRange =
Sheets(1).Range(loADODBRecordset.Fields("地址").Value)
        loRange.Value = loADODBRecordset.Fields("新数字").Value

        loADODBRecordset.MoveNext
    Wend
```

减乘除也是同理。日期加上（减去）某个数也是同理。如果是文本加上某个字符串：

原 Cell 的值加上某字符串.xls

```
lcCommandText = "select 地址, '前段' + mid(文本,2) + '后段' as 新文本 from [cell_meta_data$]"
```

当然也可以这样加：

原 Cell 的值加上某字符串_2.xls

```
lcCommandText = "select 地址, '前段' + mid(文本, 2, 1) + '中段' + mid(文本, 3, 1) + '后段' as 新文本 from [cell_meta_data$]"
```

文本也有“减”的时候，就是删除前后空白：

```
    lcCommandText = "select trim(文本) as 新文本 from [cell_meta_data$]"
```

替换函数 replace 也是一样的用法。SQL 里是可以用这些函数的。

有时把“10-20”这样的“文本”数据从别的地方导入或粘贴时，Excel 常会太过聪明地自动把它转换成“日期”。这时就可以利用这个技巧，把所有这些数据的最前面都加上一个单引号：

```
    lcCommandText = "select ’’’’ + mid(文本, 2) as 新文本 from [cell_meta_ data$]"
```

在 SQL 命令里，文本要加上单引号，如果你想选择单引号本身就用“两个”单引号即可。

范例 1-16 照单元格颜色排序

由于元数据有记录颜色（包括字体颜色、单元格颜色），所以按照颜色排序就变得非常容易，还可以按照颜色排序后，再按照数字大小排序。

按照颜色排序.xls

```
    lcCommandText = "select 数字, 单元格颜色 from
[cell_meta_data$] order by 单元格颜色, 数字"

    Sheets.Add
    Dim r As Integer
    r = 0

    Dim loRange As Range
    While Not loADODBRecordset.EOF
        r = r + 1
        Set loRange = Sheets(1).Cells(r, 1)
        loRange.Value = loADODBRecordset.Fields("数字").Value
        loRange.Interior.ColorIndex = loADODBRecordset.
Fields("单元格颜色").Value

        loADODBRecordset.MoveNext
    Wend
```

order by 可以不止 by 一个字段（group by 亦同）。order by 数字是因为这样才会将相同颜色放在一块，且按照数字大小排序，如果想从大排到小，就在数字之后加上 desc。select 出单元格颜色则是为了在 show 结果时保留原色。

自定义 Excel 元数据

记得之前提过的自定义元数据吗？我们还可以自定义更多内容。

但要自定义什么呢？按你当时的需求！但这真是一句废话。好吧，既然不知道自定义什么，索性就自定义一个叫做“自定义数据”的元数据吧（请注意粗体字部分）。

范例 1-17　提取自定义的 Excel 元数据

范例程序 1-5：由“范例程序 1-1”修改而来

```
Public Sub ActiveWorkSheet_To_Table()
    Application.ScreenUpdating = False

    Dim loSheet As Worksheet
    Dim lnSerialNumber As Long
    Dim lcText As String
    Dim lcTextValue As String
    Dim loCell As Range
    Dim ldDate, ltTime, luValue As Variant
    Dim lcValue As String

    lnSerialNumber = 0

    Open "c:\cell_meta_data.csv" For Output As #1

    ' 写入字段标题
    Write #1, _
    "序号", _
    "地址", _
    "数据类型", _
    "文本", _
```

```
    "文本长度", _
    "数字", _
    "日期", _
    "时间", _
    "行号", _
    "列标", _
    "字体名称", _
    "字号", _
    "字体颜色", _
    "单元格颜色", _
    "工作表", _
    "工作簿", _
    "文件名称", _
    "自定义数据", _
    ""

    Set loSheet = Application.ActiveWorkbook.ActiveSheet
    If loSheet.Type = -4167 Then
        loSheet.Activate
        For Each loCell In loSheet.UsedRange.Cells
            With loCell
                lcTextValue = Trim(.Text)
                lcText = "'" + lcTextValue
                luValue = .Value

                lnSerialNumber = lnSerialNumber + 1

                ldDate = ""
                ltTime = ""
                lcValue = ""

             If IsDate(lcTextValue) And Not IsNumeric(lcTextValue) Then
                    ldDate = CStr(DateValue(lcTextValue))
                 ltTime = Replace(Replace(DateValue(lcTextValue) &
TimeValue(lcTextValue), "上午", ""), "下午", "")
                End If
```

```
                If IsNumeric(.Value) And Len(lcTextValue) > 0 Then
                     lcValue = Trim(Str(Val(.Value)))
                 End If

                 Write #1, _
                 lnSerialNumber, _
                 .Address, _
                 VarType(loCell), _
                 lcText, _
                 Len(lcTextValue), _
                 lcValue, _
                 ldDate, _
                 ltTime, _
                 .Column, _
                 .Row, _
                 .Font.Name, _
                 .Font.Size, _
                 .Font.ColorIndex, _
                 .Interior.ColorIndex, _
                 loSheet.Name, _
                 loSheet.Parent.Name, _
                 loSheet.Parent.FullName, _
                 ""
             End With
        Next
    End If

    Close #1

    If Dir("c:\cell_meta_data.xls") <> "" Then
        Kill "c:\cell_meta_data.xls"
    End If

    Workbooks.Open ("c:\cell_meta_data.csv")
    ActiveWorkbook.SaveAs "c:\cell_meta_data.xls",
xlExcel9795
```

```
    ActiveWorkbook.Close
    Kill "c:\cell_meta_data.csv"

    Application.ScreenUpdating = True
End Sub
```

就这样，只是加一个字段而已？没错。加个字段，然后可以按照其他字段的值，设置这个“自定义数据”字段的值。这就好像自己为单元格加了一个属性一样，单元格可没有一个叫做“数字区间”的属性吧！微软也不可能定义这种属性吧！没关系，那就自己加一个。

目前为止所讲的 SQL 都只是找数据而已，SQL 这般英明神武的技术，当然不只是这样么简单。它还可以改数据。

范例 1-18　用“数字区间设置”修改元数据

数字区间设置.xls

```
Public Sub RunMacro()
    Run ("'" + ActiveWorkbook.Path + "\找出
meta_data.xls'!Module1.ActiveWorkSheet_To_Table")
    Workbooks("找出 meta_data.xls").Close
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As Variant
    Dim loADODBCommand As Variant

    lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                         "DBQ=c:\cell_meta_data.xls;" & _
                         "ReadOnly=False"

    Dim lcaCommandText(5) As String
    lcaCommandText(0) = "update [cell_meta_data$] set 自定义
数据 = '100~120' where 数字 >=100 and 数字 <= 120"
```

```
    lcaCommandText(1) = "update [cell_meta_data$] set 自定义
数据 = '121~200' where 数字 > 120 and 数字 <= 200"
    lcaCommandText(2) = "update [cell_meta_data$] set 自定义
数据 = '201~350' where 数字 > 200 and 数字 <= 350"
    lcaCommandText(3) = "update [cell_meta_data$] set 自定义
数据 = '351~ 999' where 数字 > 350 and 数字 <= 999"
    lcaCommandText(4) = "update [cell_meta_data$] set 自定义
数据 = '其他' where isNull(自定义数据)"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBCommand = CreateObject("ADODB.Command")

    loADODBConnection.Open lcConnectionString
    Set loADODBCommand.ActiveConnection = loADODBConnection

    Dim i As Integer
    For i = 0 To UBound(lcaCommandText) - 1
        loADODBCommand.CommandText = lcaCommandText(i)
        loADODBCommand.Execute
    Next
End Sub
```

事情变得稍微有点不同了，首先，因为要更新 meta data 的数据，所以连接数据库的字符串的 ReadOnly=False，也就是“不是 ReadOnly 的”，那就是可以更改了。

接着，要声明一个 ADODB 中的 Command 对象，有别于 RecordSet，这个对象是用来执行一些“读数据以外的动作”，通常是写入的操作，例如新增、修改等（Excel ODBC Driver 版本不支持删除操作，不过没关系，删除操作可以用“消去法”模拟。也就是，本来想从 100 个数据中删掉 10 个，那就反过来，选出不想删的 90 个，然后存成另一个新文件，再删掉原文件，最后把新文件重命名为原文件）。

同 RecordSet 一样，Command 对象也必须指定如何连接数据库，只不过它用的是 ActiveConnection 这个对象类型来指定，而不像 RecordSet 在 Open 方法中直接指定，但道理一样。要执行的指令也一样，Command 对象用 CommandText 文本类型来指定，RecordSet 是在 Open 方法中直接指定。

Command 对象的 Execute 方法就如同 RecordSet 的 Open 方法一样，当各指令设置妥当之后，接着就是去执行了。

上述操作执行完之后，打开 c:\cell_meta_data.xls，“自定义数据”字段应该已经变成你想要的样子了。这可以看作是对 meta data 的“2 次处理”，当然，也可以“N 次处理”。

改完 meta data 之后呢？就把它读出来呀！单元格依然通过定位得到，不同的是，现在又多了一个自定义数据，如果你想新建一个工作表，并把单元格的原先内容替换成该自定义数据：

数字区间设置.xls

```
Dim loADODBRecordset As Variant

lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                     "DBQ=c:\cell_meta_data.xls;" & _
                     "ReadOnly=True"

lcCommandText = "select 地址, 自定义数据 from [cell_meta_data$]"

Set loADODBRecordset = CreateObject("ADODB.Recordset")
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

Sheets.Add
```

```
    Dim loRange As Range
    While Not loADODBRecordset.EOF
        Set loRange = Sheets(1).Range(loADODBRecordset.
Fields("地址").Value)
        loRange.Value = loADODBRecordset.Fields("自定义数据").Value

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close
```

可以打开个新工作表进行替换，也可以在原工作表中设置颜色，把 Sheets.Add 拿掉，并让 loRange 等于 ActiveSheet……然后设置 loRange.Interior.ColorIndex 即可。加批注、删除、字体修改……所用方法类似。

小小的一个自定义数据字段，可以发挥的功能很大吧！举一反 N，又没规定该自定义数据只能存文本，也没规定只能存数字区间这种语义的信息、更没规定只能有一个自定义数据字段！总之，Excel 以单元格为中心，把单元各的元数据按照你的需求任意扩充是很直观且简单的一种做法。当设置多个自定义数据后，你还可以按照多重 where 条件更新那些自定义字段呢！

用 SQL 直接操纵排列整齐的 Excel 文件

到目前为止，我们介绍的都是把 Excel 的数据整理成元数据的 Table 形式，然后以威力无穷的 SQL 一网打尽，找出符合我们条件的单元格，然后操作它们。

但如此的 SQL 也只是入门而已。

其实，如果你的数据长得就像 Table 时，也可直接以 SQL 操作它。Excel 的一张张工作表就是 Table，一个工作簿就是 DataBase，还记得吗？DataBase 是 Table 的集合体，就好像工作簿是工作表的集合体一样。

技术点：Excel 文件排列整齐的概念

工作表要能成功地转化为 Table，还有个小前提，那就是 Excel 文件一定要排列整齐，具体概念是：每一列（就是 A、B、C…）要有个字段标题，同一列的数据其数据类型也必须相同，换句话说，就是数据要摆得整整齐齐的啦！比方说，A1 是“姓名”，姓名的数据类型是“文本”，所以从 A2~A65536 的内容都必须是文本。字段标题不能以数字开头，不能有特殊的符号如“+*/$@!…”、可以有“-”号但不建议使用（怕你以后学别的 DataBase 时不支持，但现在养成了坏习惯。

直接看图 1-5 所示的数据比较清楚：

	A	B	C
1	编号	姓名	生日
2	1	Chris	1970-1-10
3	2	Rita	1981-11-12
4	3	Irene	1970-9-1
5	4	Calvin	1980-12-12
6	5	Kevin	1971-12-17
7	6	Bruce	1970-11-1
8	7	Maggie	1973-6-5
9	8	Amy	1978-10-10
10	9	Urania	1972-6-16
11	10	Vincent	1961-10-1

图 1-5

这些数据所在的工作表是“Sheet1”，代表 Table 名称是[Sheet1$]（注意要加个“$”符号）；它有 3 个字段：编号、姓名、生日，数据类型分别是数字、文本、日期。有字段标题、每列数据的数据类型一致，字段名称也没问题，数据摆得整整齐齐，没有任何合并或拆分单元格之类的情况。OK，这就算是一份格式正确的 Table 了。当你的 Excel 数据长得如此模样，恭喜你，不必申请，SQL 仆人已经随侍在侧了。

范例 1-19　直接在 Excel 工作表中找数据

从最简单的 case 开始，找所有数据：

找数据_001.xls

```
Dim lcConnectString, lcCommandText As String
Dim loADODBConnection As ADODB.Connection
Dim loADODBRecordset As ADODB.Recordset

ActiveWorkbook.Save

lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                     "DBQ=" + ActiveWorkbook.FullName + ";" & _
                     "ReadOnly=True"

lcCommandText = "select * from [sheet1$]"

Set loADODBConnection = CreateObject("ADODB.Connection")
Set loADODBRecordset = CreateObject("ADODB.Recordset")

loADODBConnection.Open lcConnectionString
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

Sheets.Add
```


```
Dim r, f As Integer
r = 1
For f = 0 To loADODBRecordset.Fields.Count - 1
   Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
Next

While Not loADODBRecordset.EOF
    r = r + 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
      Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Value
    Next

    loADODBRecordset.MoveNext
Wend
loADODBConnection.Close
```

将loADODBRecordSet等变量声明成它应有的对象数据类型，而非通用的 Variant 类型的原因是，这样就可以享有 IntelliSense 的方便性（不必记住属性、方法等）了。

连接数据库的字符串中的 DBQ 不再是 c:\cell_meta_data.xls 了，而是当前文件自己，所以用 ActiveWorkBook.Fullname。在存取这个 Fullname 之前，先保存文件，以确保 Fullname 有值。

接下来的 SQL 命令之前已经出现过 N 次，应该很熟悉了。

Fields.Count 之类的东西跟“范例程序 1-2”几乎一样，不赘述了。

以 SQL 的方式直接操作 Excel 文件时也不能避免密码保护的问题，也就是，如果某文件有密码保护，则连接数据库时就会出错。

范例 1-20　按照自定义的字段顺序找数据

像图 1-5 这样的数据，如果要把编号跟姓名对调，也就是按照“姓名、编号、生日”这样顺序来找的话，只要把“*”（所有字段）改成“姓名、编号、生日”就 OK 了.

找数据_002.xls

```
lcCommandText = "select 姓名, 编号, 生日 from [sheet1$]"
```

找部分字段：

```
lcCommandText = "select 姓名, 生日 from [sheet1$]"
```

指定条件以筛选出“编号＜5”的记录：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where 编号 < 5"
```

姓名为大写 C 开头的（也就是姓名左边第一个字符等于大写 C）：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where left(姓名, 1) = 'C'"
```

姓名为大写 C 或大写 A 开头的：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where left(姓名, 1) = 'C' or left(姓名, 1) = 'A'"
```

姓名中含有 C 的（不分大小写），使用 like 运算加上通配符“%”：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where 姓名 like '%C%' or 姓名 like '%c%'"
```

利用函数的话，就不用以 or 下达两个条件了：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where
ucase(姓名) like '%C%'"
```

1970 年以前出生的人（日期类型的字段，前后用“#”符号括起来）：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where 生
日 < #1970/01/01#"
```

1970 年生（也就是生日取年份等于 1970）且姓名中含有 C 的（不分大小写）：

```
    lcCommandText = "select 姓名, 生日 from [sheet1$] where
year(生日) = 1970 and ucase(姓名) like '%C%'"
```

还可以合并单元格的内容（因为生日是日期类型，所以用 cstr() 函数转换成文本后再与姓名相加）：

```
    lcCommandText = "select 姓名 + ':' + cstr(生日) as 个人信
息 from [sheet1$]"
```

范例不够多？语法不熟？这里只能一点一滴地教你，无法穷举你会碰到的每一个问题，可以求助 Google，也可以发邮件给作者：excel-excel@hotmail.com。

范例 1-21　给满足条件的记录加底色

当 Excel 数据长得就像一个 Table 时，同样可以自定义字段。

SQL 除了可以选择（select）已有的字段外，也可以选择一个表达式（expression），这个表达式可以是常数、经过运算后的字段等。其实之前用过多次的 left(<某字段>,N)、sum（<某字段>）、count(*)等，都是一种表达式。

自定义字段配合之前说过的 SQL 更新指令（利用 ADODB.Command 对象），可以对一张工作表做到“两次（或 N 次）处理”，轻松达到传统宏写法意想不到或很难达到的效果。

下面的例子是将图 1-5 中姓名含有 C 的（不分大小写）所在行整行标上红色。

找数据_003.xls

```
Dim lcConnectString, lcCommandText As String
Dim loADODBConnection As ADODB.Connection
Dim loADODBRecordset As ADODB.Recordset

ActiveWorkbook.Save

lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                     "DBQ=" + ActiveWorkbook.FullName + ";" & _
                      "ReadOnly=False"

' 自定义 1 个字段：颜色
lcCommandText = "select 姓名, 生日, 0 as 颜色 from [sheet1$]"

Set loADODBConnection = CreateObject("ADODB.Connection")
Set loADODBRecordset = CreateObject("ADODB.Recordset")

loADODBConnection.Open lcConnectionString
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

Sheets.Add
```

```
    Sheets(1).Name = "Sheet2"
    Dim r, f As Integer
    r = 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
       Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
    Next

    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
          Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Value
        Next

        loADODBRecordset.MoveNext
    Wend

    ' 因为数据已经更改过，所以重新连接数据库一次
    loADODBConnection.Close
    loADODBConnection.Open lcConnectionString

    ' 有了名为 Sheet2 的工作表，可以对它 SQL 一番了
    lcCommandText = "update [Sheet2$] set 颜色 = 3 where
ucase(姓名) like '%C%'"

    Dim loADODBCommand As ADODB.Command
    Set loADODBCommand = CreateObject("ADODB.Command")
    Set loADODBCommand.ActiveConnection = loADODBConnection
    loADODBCommand.CommandText = lcCommandText
    loADODBCommand.Execute

    ' 找 Sheet2 更改后的结果，并以该结果设置最原先的 Sheet 中的颜色
    lcCommandText = "select * from [sheet2$]"
   loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1
    Dim loRange As Range
    r = 1
    While Not loADODBRecordset.EOF
        r = r + 1
```

```
        Set loRange = Sheets(2).Rows(r)
        loRange.Interior.ColorIndex = loADODBRecordset.
Fields("颜色").Value

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    ' sheets(1) 功成身退，删掉它
    Application.DisplayAlerts = False
    Sheets(1).Delete
    Application.DisplayAlerts = True
```

希望你别看晕了。这种写法有其概念上的整体性（concept integrity）与一致性，更重要的是，许许多多的其他工具都支持类似的写法（就是 SQL 啦！），如果用传统的宏语法，每个人写的都很有可能极不一样。而这种方式，则更易于在不同的程序员之间沟通。

范例 1-22　统计处理员工请假记录

本范例用到的 Excel 数据表如图 1-6 所示。

	A	B
1	姓名	请假时间
2	Rita	2004-02-05
3	Irene	2004-04-11
4	Chris	2004-03-01
5	Chris	2004-03-19
6	Irene	2004-01-05
7	Rita	2004-03-01
8	Chris	2004-04-09
9	Rita	2004-04-10
10	Chris	2004-02-03

图 1-6

找出最后（近）一个请假记录：

找出最后一个请假记录.xls

```
lcCommandText = "select 姓名, max(请假时间) as 最后请假时间 from [sheet1$] group by 姓名"
```

找出最早一个请假记录，则用 min(请假时间) as 最早请假时间。如果要算出两个时间差，就用 max(请假时间) - min(请假时间) as 时间差。

统计每个人共请几次假：

```
    lcCommandText = "select 姓名, count(*) as 请假次数 from [sheet1$] group by 姓名"
```

按照月份统计每个人共请几次假：

```
    lcCommandText = "select 姓名, month(请假时间) as 月份, count(*) as 请假次数 from [sheet1$] group by 姓名, month(请假时间)"
```

按照年份、部门、性别、职等、年龄等，或按照这些字段相加统计也一样。按照什么统计就 group by 什么，这个“什么”可以不止一个字段，也可以加上表达式。

还可以用 iif 函数自定义字段：

```
    lcCommandText = "select 姓名, 请假时间, iif(month(请假时间) mod 2 = 0, '双数月', '单数月') as 月别 from [sheet1$]"
```

有个很常见的问题（及其他类似的问题）也都可以用 iif 函数，考虑如图 1-7 所示的工作表。

	A	B	C
1	年	月	日
2	2004	1	1
3	2004	1	2
4	2004	3	4
5	2004	2	2
6	2004	12	11

图 1-7

如果要在 D 列显示 ABC 3 栏连接出的完整日期：

连接完整日期.xls

```
lcCommandText = "select 年, 月, 日,
trim(str(年))+'/'+trim(str(月))+'/'+trim(str(日)) as 日期 from
[sheet1$]"
```

因为年月日都是数字，所以把它们都转成文本后（str 函数）再截去前后多余的空格（trim 函数），中间以“/”相连接，就可以连接出完整日期了。

范例 1-23　计算工龄工时

计算工龄：

计算工龄.xls

```
lcCommandText = "select 姓名, 到职日,
iif(Month(Now)*100+Day(Now) < Month(到职日)*100+Day(到职日),
Year(Now)-Year(到职日)-1, Year(Now)-Year(到职日)) as 工龄_满几
年 from [sheet1$]"
```

计算工龄、年龄、（某网站）注册日至今的年数、升经理至今的年数……凡牵涉到日期跟此时此刻的差值的都可以用这种算法。这里采用的是“满几年”的计算方式，例如 2001-10-01 到职，2004-10-01 满 3 年，2004-9-30 则仍算 2 年。

计算两个时间的时间差，也可以用 DateDiff 函数。例如计算工时：

计算工时.xls

```
select 姓名, DateDiff('h', 上班时间, 下班时间) as 工时 from [sheet1$]
```

DateDiff 中的参数“h”表示要以小时为单位来表示这两个时间的时间差。也可以用年、季、月、日、分、秒等来表示。注意，“h”只会计算单位比小时大的部分，其他则会被忽略。也就是：

- “2004-10-01 08:00”至“2004-10-01 17:59”是 17-8=9 小时。
- “2004-10-01 08:00”至“2004-10-02 17:59”是 24+9=33 小时。
- “2004-10-01 08:00”至“2004-10-01 18:00”是 18-8=10 小时。
- “2004-10-01 08:00”至“2004-10-01 18:59”还是 18-8=10 小时，并不会因为很接近 11 个小时就自动进位成 11。
- 如果你要算两个日期之间相差“几年几月几天”的话，可以利用函数 DateDif（是的，跟本范例的 DateDiff 只相差一个字母），例如要计算 1996-04-15（置于单元格 A1）至 2002-12-20（置于单元格 B1）之间差几年几个月几天的话，则输入：

```
Range("C1").FormulaR1C1 = _
"=DATEDIF(RC[-2],RC[-1],""Y"") & "" 年 "" & DATEDIF(RC[-2],
RC[-1],""YM"") & "" 个月 "" & DATEDIF(RC[-2],RC[-1],""MD"") &
"" 天"""
```

如此，单元格 C1 的值就会是“6 年 8 个月 5 天”了。

- DateDiff 的其余用法请参照范例文件“两个时间之间的差.xls”。

范例 1-24　算成绩

图 1-8 的成绩表很常见，学生考试成绩、业务员销售成绩都是这种表格的翻版。

	A	B	C	D
1	学生	语文	英语	数学
2	孙彦姿	30	60	30
3	小S	20	76	10
4	莫文卫	22	89	90
5	菜一邻	100	90	100
6	苏有彭	98	12	99
7	施义楠	12	78	100
8	蔡灿德	99	65	86
9	林新如	100	34	0
10	大S	80	100	100
11	范执委	74	99	88

图 1-8

算成绩，用表达式：

算成绩.xls

```
lcCommandText = "select 学生, 语文, 英语, 数学, (语文+英语+数学) as 总分, format((语文+英语+数学)/3, '###.##') as 平均 from [sheet1$]"
```

“平均”计算出来的结果小数点后可能有很多位，因为最多就是“100.00”，所以可利用 format 函数传入“###.##”字符串。

这个范例的原始 Table 有 4 个字段，“学生、语文、英语、数学”，如果我不想写这么多，只想写“*”代表所有字段（这样写的好处是，如果后来字段增加或减少了几个，还是可以用“*”），然后再写总分跟平均分那两个表达式，结果出来的字段顺序居然是表达式在前，即使把“*”放在表达式之后亦同。

对这种结果我感觉很奇怪，但至今仍无法找到原因。所以，只好不能偷懒，一个一个字段都要写。

范例 1-25　排名次

成绩算完只是第一步，老师（或老板）还想知道名次。在 show 出关键程序代码之前，先回到这个用过多次的指令：

```
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1
```

之前说过，大部分的时候不必管这些参数，反正我们用的都是“3,1,1”。现在稍稍改一下，改成“3,4,1”。这是因为我们查询出来后还要对那个查询结果做修改，所以第 2 个参数改成 4。

来看看关键的粗体字程序代码：

排名次.xls

```
lcCommandText = "select 学生, 语文, 英语, 数学, (语文+英语+数学) as 总分, format((语文+英语+数学)/3, '###.##') as 平均, 0 as 名次 from [sheet1$] order by (语文+英语+数学) desc"

Set loADODBConnection = CreateObject("ADODB.Connection")
Set loADODBRecordset = CreateObject("ADODB.Recordset")

loADODBConnection.Open lcConnectionString
loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 4, 1

Sheets.Add
Dim r, f As Integer
r = 1
For f = 0 To loADODBRecordset.Fields.Count - 1
   Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
Next

Dim lnRank As Integer
```

```
    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
            If loADODBRecordset.Fields(f).Name = "名次" Then
                lnRank = lnRank + 1
                loADODBRecordset.Fields(f).Value = lnRank
            End If
            Sheets(1).Cells(r, f + 1) = loADODBRecordset.
Fields(f).Value
        Next

        loADODBRecordset.MoveNext
    Wend
```

加了一个表达式字段“名次”。名次的意义就是按照总分（或平均分）降序（desc，由大到小）排序。暂且都先给 0。然后在查询出来的循环结果中依次排名。这个操作需要涉及到该查询结果中“名次”字段的值，所以才要把“3,1,1”改成“3,4,1”。

排序后，名次就是按照“1、2、3…”这样排下来，所以把一个简单的累加 lnRank 变量填入“名次”字段就 OK 了。

范例 1-26　对多个科目分组排序

如果有“同分”的状况呢？算成绩的话，通常再找一科比较，假设是语文（插个话，语文是我认为跟英文一样重要、甚至更为重要的科目，它牵涉到基本的表达能力与沟通能力，程序设计生涯用到的语文并不比数学少，当然，或许英语还是用得最多的），那 SQL 的 order by 就再加一个字段：

```
    lcCommandText = "select 学生, 语文, 英语, 数学, (语文+英
语+数学) as 总分, format((语文+英语+数学)/3, '###.##') as 平均,
```

```
0 as 名次 from [sheet1$] order by (语文+英语+数学) desc, 语文 desc"
```

还记得吗，order by 可以不止一个字段。所以，网络上经常被问到的、因为 Excel 最多只能针对 3 个 key 值排序的这种问题，应该不必再用一些绕口令的咒语，如 VLookUp 来 CountIF 去了吧！

另外，order by 的各个字段也不一定都要用同一种顺序（同为升序或同为降序），可以字段 A 用升序、字段 B 用降序。order by 的字段也不一定要被 select 出来，也就是可以 select 学生，但却 order by 数学成绩。

即使在图 1-8 最左边加上“班级”字段也是一样，就再多加一个 order by 班级：

排名次_加上班级.xls

```
lcCommandText = "select 班级, 学生, 语文, 英语, 数学, (语文+英语+数学) as 总分, format((语文+英语+数学)/3, '###.##') as 平均, 0 as 名次 from [sheet1$] order by 班级, (语文+英语+数学) desc, 语文 desc"
```

还需要加一个变量，在循环中遇到不同班级的话就让名次重新开始：

排名次_加上班级.xls

```
    Dim lnRank As Integer
    Dim lcLastClass As String
    lcLastClass = "!@#$%^&*()"

    While Not loADODBRecordset.EOF
        If loADODBRecordset.Fields("班级").Value <>
lcLastClass Then
            lnRank = 0
        End If
```

```
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
            If loADODBRecordset.Fields(f).Name = "名次" Then
                lnRank = lnRank + 1
                loADODBRecordset.Fields(f).Value = lnRank
            End If
            Sheets(1).Cells(r,f+1)=loADODBRecordset.Fields(f).Value
        Next
        lcLastClass = loADODBRecordset.Fields("班级").Value
        loADODBRecordset.MoveNext
    Wend
```

范例 1-27　找出不重复的值（或说“去掉重复值”）

图 1-9 这个看似毫无意义的 Table，其实比较像是某个更大 Table 的一部分，但却是个典型的问题。这条 SQL 实在太简单了：

	A
1	学生
2	孙彦姿
3	小S
4	莫文卫
5	莱一邻
6	苏有彭
7	施义楠
8	蔡灿德
9	林新如
10	大S
11	范执委
12	孙彦姿
13	施义楠
14	大S
15	林新如
16	苏有彭
17	林新如
18	施义楠
19	大S
20	大S
21	范执委
22	施义楠
23	莱一邻
24	小S
25	小S
26	莱一邻

图 1-9

找出不重复的值.xls

```
lcCommandText = "select distinct 学生 from [sheet1$]"
```

如果是图 1-10 这样，想找出“谁曾到过哪些国家”当然也不能重复：

	A	B	C
1	学生	到过的国家	哪一年去的
2	孙彦姿	美国	1998
3	小S	瑞士	1998
4	莫文卫	朝鲜	1997
5	菜一邻	马来西亚	1999
6	苏有彭	马来西亚	1999
7	施义楠	美国	2000
8	蔡灿德	瑞士	1996
9	林新如	瑞士	2003
10	大S	美国	2004
11	范执委	马来西亚	2001
12	孙彦姿	新加坡	2003
13	施义楠	美国	2002
14	大S	新加坡	1998
15	林新如	瑞士	1995
16	苏有彭	美国	2001
17	林新如	美国	2004
18	施义楠	日本	2003
19	大S	日本	2002
20	大S	新加坡	1998
21	范执委	美国	1999
22	施义楠	日本	2002
23	菜一邻	日本	2001
24	小S	美国	2004
25	小S	日本	2003
26	菜一邻	马来西亚	2001

图 1-10

SQL 里的很多参数，像是 order by、group by、字段列表、表格列表（是的，还没讲到）、where 条件、having 条件等都可以不止一个。distinct 之后接的是字段，当然也可以不止一个：

找出不重复的值_2.xls

```
lcCommandText = "select distinct 学生, 到过的国家 from [sheet1$]"
```

这时候的 distinct 是把“学生”跟“到过的国家”加起来看的，也就是孙彦姿到过美国，苏有彭到过美国，虽然到过的国家都是美国，但因学生不同，会算成是两个。

用户自定义函数（User Defined Function，UDF）

范例 1-28　在 Excel 中编写用户自定义函数

按 Alt+F11 键进入 VBA 宏编辑环境，写一小段简单的、分辨数字区间的程序：

用户自定义函数.xls

```
Public Function MyGroup(pnNumber As Integer) As Double
    Select Case pnNumber
        Case 100 To 199
            MyGroup = 1
        Case 200 To 229
            MyGroup = 2
        Case 230 To 299
            MyGroup = 2.5
        Case 300 To 399
            MyGroup = 3
        Case 400 To 449
            MyGroup = 4
        Case 450 To 499
            MyGroup = 4.5
        Case Else
            MyGroup = 5
    End Select
End Function
```

然后在 A1 输入 100，B1 输入“=MyGroup(A1) ”，结果 B1 的值会等于 1。当然也可以在 B1 直接输入“=MyGroup(100) ”。OK，先记住这点。

范例 1-29　在 Excel 中使用用户自定义函数编程

在比较正式的数据库产品中，从羽量级的 Access、轻量级的 FoxPro、到重量级的 SQL Server、DB2、Oracle 等，都支持用户自定义函数。有了这种功能，在使用 SQL Select 命令时就可以不限于只用 left(<某字段>,1)、mid(<某字段>,1)…这种自带的函数，而可以用上自定义的函数。上面的 MyGroup 就是一种自定义函数，可惜 Excel 做为一种数据库时并不支持这种功能。所以我们不能直接写：

```
select MyGroup(数字) as 该数字所在区间…
```

这样的命令（即使已经自行定义了 MyGroup 这个函数）。

为何举这个例子？还记得<数字区间设置.xls>那个范例吗？我们用的是“两次处理”方法更改查询出来的结果，该方法很棒，没错，但现在介绍另一种作法，就是利用自定义函数。

虽然不能直接 select MyGroup，但可以间接：

```
select 数字, '=MyGroup(数字)' as 该数字所在区间…
```

注意到在 MyGroup 前面加了一个“=”吗？而且 MyGroup 是以一对单引号括起来？没错！这是在“制造公式”。但有些不对，因为 select“'=MyGroup(数字)'”出来的结果将全部是常数，不会随着数字变化。所以要这样：

```
    select 数字, '=MyGroup(' + trim(str(数字)) + ')' as 该数字所在区间…
```

这样出来就会是：

```
100, =MyGroup(100)
200, =MyGroup(200)
230, =MyGroup(230)
……
```

也达到了“间接”使用自定义函数 MyGroup 的目的。

而由于数字所在区间都已经由 MyGroup 计算得出，之后就可以直接 group by 该数字所在区间来做分组统计：

用户自定义函数.xls

```
    lcCommandText = "select 数字, '=MyGroup(' + trim(str(数字)) + ')' as 该数字所在区间 from [sheet1$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add
    Sheets(1).Name = "Sheet2"
    Dim r, f As Integer
    r = 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
       Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
    Next

    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
          Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Value
        Next

        loADODBRecordset.MoveNext
    Wend
```

```
    ' 数据已经改过，再重新联机一次
    loADODBConnection.Close
    loADODBConnection.Open lcConnectionString

    lcCommandText = "select 该数字所在区间, count(*) as 数量 from
[sheet2$] group by 该数字所在区间"
    loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add
    Sheets(1).Name = "统计结果"
    r = 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
       Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
    Next

    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
          Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Value
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    ' Sheet2 功成身退
    Application.DisplayAlerts = False
    Sheets("Sheet2").Delete
    Application.DisplayAlerts = True
```

由这个例子也可以得知，用数据库的方式存取 Excel 工作表时，其 Cells 的值并不会因为是公式（此例是 MyGroup(100)等）就有不同，还是可以找得到正确内容。

Excel 多表联合数据操作

到目前为止，我们谈的都是一个 Table 的数据读取，如果 SQL 只能做到这样，那今天的企业信息化大概会累得没人要做，因为随便就会遇到一个 Table 以上的数据比较、交叉读取等需求。

范例 1-30　多表联合数据查询（一）

还是直接看图 1-11 所示的数据。

	A	B	C
1	编号	姓名	生日
2	1	Chris	1970-1-10
3	2	Rita	1981-11-12
4	3	Irene	1970-9-1
5	4	Calvin	1980-12-12
6	5	Kevin	1971-12-17
7	6	Bruce	1970-11-1
8	7	Maggie	1973-6-5
9	8	Amy	1978-10-10
10	9	Urania	1972-6-16
11	10	Vincent	1961-10-1

Sheet1

	A	B	C
1	编号	请假日期	请假原因
2	1	2004-3-1	病假
3	1	2004-3-2	事假
4	2	2004-3-3	事假
5	2	2004-3-20	病假
6	2	2004-4-1	病假
7	1	2004-4-1	事假
8	1	2004-5-1	事假
9	3	2004-6-1	休假
10	3	2004-7-1	休假
11	4	2004-1-6	休假
12	5	2004-6-1	事假
13	6	2004-2-2	休假
14	7	2004-3-1	公假
15	1	2004-6-1	休假
16	2	2004-6-1	病假
17	2	2004-1-10	休假
18	1	2004-10-1	休假
19	3	2004-8-20	病假
20	3	2004-9-1	病假
21	9	2004-6-1	公假

Sheet2

图 1-11

两个 Table，Sheet1 及 Sheet2，把“编号”当作是相互连接的键值（key），有了这个键值，Sheet1 的“姓名”就无须在 Sheet2 中重复再存一次了。

直接这样写：

找数据_2 个 Table.xls

```
lcCommandText = "select sheet1$.*, sheet2$.* from [sheet1$],
[sheet2$] where sheet2$.编号 = sheet1$.编号"
```

因为现在有两个 Table 了，所以 from 后面就会有两个 Table，而“*”也要加上 Table 名称，SQL 才知道这是谁的“*”。如果两个字段有相同名称（此例中的“编号”），也要加上 Table 名称以作区别：

```
    lcCommandText = "select sheet1$.编号, sheet2$.请假日期 from
[sheet1$], [sheet2$] where sheet2$.编号 = sheet1$.编号"
```

当然，如果两个 Table 的所有字段都要找出来的话，可以直接用一个“*”不冠 Table 名称。此时的这个“*”代表这两个 Table 的所有字段集合。

就是这样，很简单吧！两个 Table 之间的连接条件可直接看作是 where 条件的一部分。

查询结果（见图 1-12）：

	A	B	C	D	E	F
1	编号	姓名	生日	编号	请假日期	请假原因
2	1	Chris	1970-1-10	1	2004-10-1	休假
3	1	Chris	1970-1-10	1	2004-6-1	休假
4	1	Chris	1970-1-10	1	2004-3-2	事假
5	1	Chris	1970-1-10	1	2004-3-1	病假
6	1	Chris	1970-1-10	1	2004-5-1	事假
7	1	Chris	1970-1-10	1	2004-4-1	事假
8	2	Rita	1981-11-12	2	2004-1-10	休假
9	2	Rita	1981-11-12	2	2004-6-1	病假
10	2	Rita	1981-11-12	2	2004-3-3	事假
11	2	Rita	1981-11-12	2	2004-3-20	病假
12	2	Rita	1981-11-12	2	2004-4-1	病假
13	3	Irene	1970-9-1	3	2004-7-1	休假
14	3	Irene	1970-9-1	3	2004-9-1	病假
15	3	Irene	1970-9-1	3	2004-8-20	病假
16	3	Irene	1970-9-1	3	2004-6-1	休假
17	4	Calvin	1980-12-12	4	2004-1-6	休假
18	5	Kevin	1971-12-17	5	2004-6-1	事假
19	6	Bruce	1970-11-1	6	2004-2-2	休假
20	7	Maggie	1973-6-5	7	2004-3-1	公假
21	9	Urania	1972-6-16	9	2004-6-1	公假

图 1-12

不知你有没有注意到，查询结果并没有涵盖所有人，这是正常的，因为请假记录中就已经不是涵盖所有人了（有人得全勤奖）。

像这种两个 Table 之间的连结，有个正式的名称叫“join”。两个 Table join 的结果可看作是这两个 Table 之间的“交集”。

范例 1-31　多表联合数据查询（二）

如何把那些没请假的人也要列出来呢？这也难不倒 SQL，直接使用 join 命令就可以了：

```
    lcCommandText = "select * from [sheet1$] left outer join
[sheet2$] on sheet1$.编号 = sheet2$.编号 order by sheet1$.编号,
sheet2$.请假日期"
```

只是把 where 改成 on，然后在前面加个 left outer join 而已。也就是左边 Table 有的，即使右边 Table 没有相对应的记录，也要以空白内容（精确的说法应该是 Null 值）把它呈现出来。

查询结果（见图 1-13）：

	A	B	C	D	E	F
1	编号	姓名	生日	编号	请假日期	请假原因
2	1	Chris	1970-1-10	1	2004-3-1	病假
3	1	Chris	1970-1-10	1	2004-3-2	事假
4	1	Chris	1970-1-10	1	2004-4-1	事假
5	1	Chris	1970-1-10	1	2004-5-1	事假
6	1	Chris	1970-1-10	1	2004-6-1	休假
7	1	Chris	1970-1-10	1	2004-10-1	休假
8	2	Rita	1981-11-12	2	2004-1-10	休假
9	2	Rita	1981-11-12	2	2004-3-3	事假
10	2	Rita	1981-11-12	2	2004-3-20	病假
11	2	Rita	1981-11-12	2	2004-4-1	病假
12	2	Rita	1981-11-12	2	2004-6-1	病假
13	3	Irene	1970-9-1	3	2004-6-1	休假
14	3	Irene	1970-9-1	3	2004-7-1	休假
15	3	Irene	1970-9-1	3	2004-8-20	病假
16	3	Irene	1970-9-1	3	2004-9-1	病假
17	4	Calvin	1980-12-12	4	2004-1-6	休假
18	5	Kevin	1971-12-17	5	2004-6-1	事假
19	6	Bruce	1970-11-1	6	2004-2-2	休假
20	7	Maggie	1973-6-5	7	2004-3-1	公假
21	8	Amy	1978-10-10			
22	9	Urania	1972-6-16	9	2004-6-1	公假
23	10	Vincent	1961-10-1			

图 1-13

这种 outer join 很常用，但凡出现在 Table A 但不出现在 Table B 的相同 key 值（如此例中的编号），都可以用 outer join。

范例 1-32　对多表查询结果进行统计

SQL 的威力就是凡事以 Table 的角度看待（就如同 Excel 凡事以 Cell 看待一样），所以，不要被两个 Table 搞混了，两个 Table 查询出来的结果其最终形式仍旧是一个 Table。既然是一个 Table，就可以用原先查询一个 Table 的种种语法来对其操作，所以当然也可以直接统计每个人的请假次数了：

统计请假次数.xls

```
lcCommandText = "select sheet1$.编号, sheet1$.姓名,
sum(iif(isnull(sheet2$.编号), 0, 1)) as 请假次数 from [sheet1$]
left outer join [sheet2$] on sheet1$.编号 = sheet2$.编号 group
by sheet1$.编号, sheet1$.姓名"
```

为什么不直接用 count(*)呢？因为图 1-13 的编号 8 跟 10 虽然无内容，但是还是算一个空的数据，所以 count(*)时，一样会被算成一个。

但可以用 sum 加上 iif 函数，这样一来单元格中虽有数据但内容空白（也就是 Null 值）的话，就不 sum 进去，否则就 sum 进去。

这个问题还有其他解法（SQL 的 union 加上 not in），此处说的应该算是最简单且直观的方法了。

执行结果（见图 1-14）：

	A	B	C	D
1	编号	姓名	请假次数	
2	1	Chris	6	
3	2	Rita	5	
4	3	Irene	4	
5	4	Calvin	1	
6	5	Kevin	1	
7	6	Bruce	1	
8	7	Maggie	1	
9	8	Amy	0	
10	9	Urania	1	
11	10	Vincent	0	
12				

图 1-14

举一反 N，两个 Table 可以，那更多 Table 呢？当然可以一路连接下去：

找数据_3 个 Table.xls

```
lcCommandText = "select sheet1$.*, sheet3$.部门名称, sheet2$.请假日期, sheet2$.请假原因 from [sheet1$], [sheet2$], [sheet3$] where sheet2$.编号 = sheet1$.编号 and sheet3$.部门代号 = sheet1$.部门代号"
```

请注意，连接的方式并不限定只有 A 连接 B、B 连接 C、C 连接 D…，只要是表之间彼此有相同的字段或字段表达式，都可以当作是它们之间的交集拿来连接。连接的条件就写在 where 里，也不表示就不能写其他条件了，你可以先过滤（或后过滤）Table A 的记录，再拿来跟别的 Table 连接。

对 Excel 工作表的区域进行数据操作

到目前为止，我们谈的 Excel 的表都是以工作表为单位，在 select 时，都是 select from sheet。诚然，工作表长得就像

表没错，但不知你有没有注意到，其实工作表里的一个区域（Range），例如 A10:F200，不也长得像表吗？只是小一点罢了。

知识点：Excel 工作表区域的表示方式

Excel 工作表中的区域，是用左上角单元格的地址和右下角单元格的地址中间加冒号来表示的，比如 A10:F200。

如果要指定特定工作表的某个区域，则需要区域地址的前面，添加工作表的名称，比如：

```
sheet1$C10:E20
```

就表示工作表 sheet1 中的一个区域，这个区域左上角的单元格为 C10，右下角的单元格为 E20。

范例 1-33　单工作表单区域的数据查询

既然可以：

```
lcCommandText = "select * from [sheet1$]"
```

也就可以用：

找数据_从某个区域.xls

```
lcCommandText = "select * from [sheet1$C10:E20]"
```

对图 1-15 所示的工作表区域进行查询。

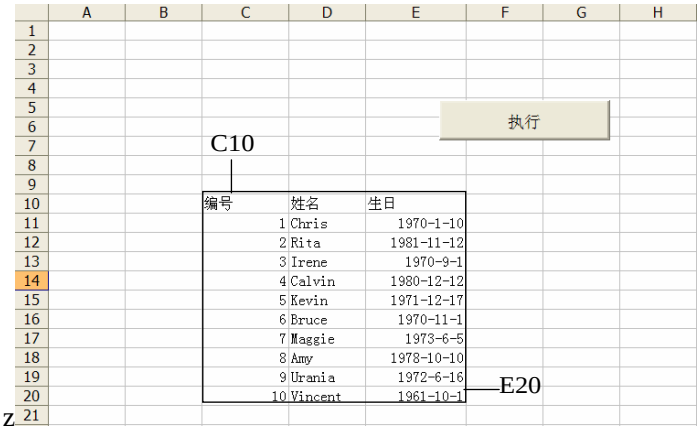
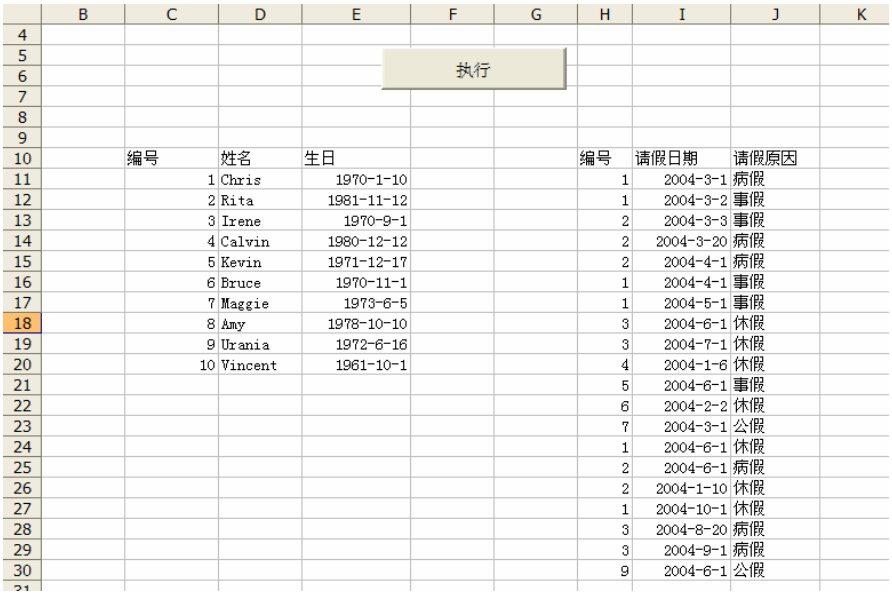


图 1-15

范例 1-34　单工作表多区域的数据查询

如果你想节省空间，把两个 Table 放在同一个工作表里的话（虽然我极不提倡这种摆放方式，见图 1-16）：



编号	姓名	生日	编号	请假日期	请假原因
1	Chris	1970-1-10	1	2004-3-1	病假
2	Rita	1981-11-12	1	2004-3-2	事假
3	Irene	1970-9-1	2	2004-3-3	事假
4	Calvin	1980-12-12	2	2004-3-20	病假
5	Kevin	1971-12-17	2	2004-4-1	病假
6	Bruce	1970-11-1	1	2004-4-1	事假
7	Maggie	1973-6-5	1	2004-5-1	事假
8	Amy	1978-10-10	3	2004-6-1	休假
9	Urania	1972-6-16	3	2004-7-1	休假
10	Vincent	1961-10-1	4	2004-1-6	休假
			5	2004-6-1	事假
			6	2004-2-2	休假
			7	2004-3-1	公假
			1	2004-6-1	休假
			2	2004-6-1	病假
			2	2004-1-10	休假
			1	2004-10-1	休假
			3	2004-8-20	病假
			3	2004-9-1	病假
			9	2004-6-1	公假

图 1-16

那么 join 这两个 Table 就可以这样下：

找数据_从同一工作表的两个区域.xls

```
lcCommandText = "select * from [sheet1$C10:E20] a,
[sheet1$H10:J30] b where b.编号 = a.编号"
```

字段可以用 as 关键词给出一个新的别名（还记得吗？在 select 字段表达式的时候），Table 当然也可以，但不必加 as。加了别名之后，（此处）的 where 条件就可以不必写得这么长了。

同理，3 个或更多的 Table 也可以类似处理，放在同一个工作表。

如你所知，在 Excel 中可以给某个区域命名（见图 1-17）：

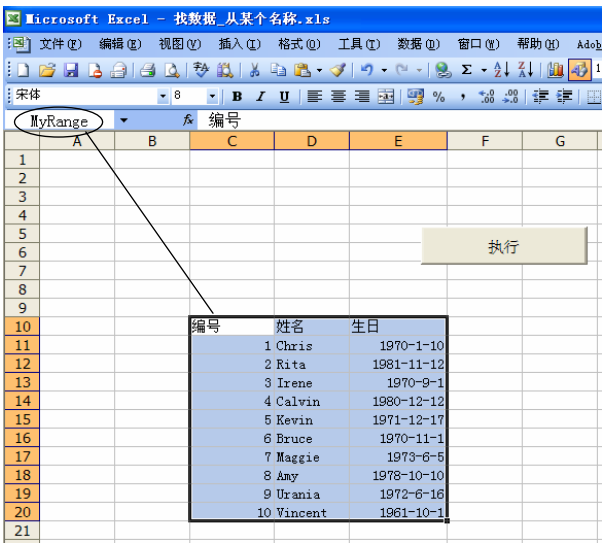


图 1-17

命名之后，就可以直接把该名称就当作 Table 名称，而不必记住这个区域的地址了：

找数据_从某个名称.xls

```
lcCommandText = "select * from [sheet1$][MyRange]"
```

指定查询结果的存储位置

由于 SQL 查询结果也是一个表，所以我们都会先新添加一个工作表，然后写入字段名称，再逐个把查询结果塞到该工作表中。但当查询结果很多时，这么做会比较慢。

范例 1-35　用 CopyFromRecordSet 方法指定存储位置

有个指令可以一次就把查询结果放到指定的位置：CopyFromRecordSet 方法。

找数据_使用 CopyFromRecordSet 方法.xls

```
Public Sub RunMacro()
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As ADODB.Connection
    Dim loADODBRecordset As ADODB.Recordset

    ActiveWorkbook.Save

   lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                        "DBQ=" + ActiveWorkbook.FullName + ";" & _
                           "ReadOnly=True"

    lcCommandText = "select * from [sheet1$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
   loADODBRecordset.Open lcCommandText, loADODBConnection, 3, 1, 1

    Sheets.Add
    Sheets(1).Cells(1, 1).CopyFromRecordset loADODBRecordset
```

```
    loADODBConnection.Close
End Sub
```

为何现在才说这个简洁又快速的方法？嗯，我想还是先让你了解多一点的字段名称跟字段结构的 Object Model 吧！反正也不算复杂嘛！

而且，其实 CopyFromRecordSet 方法有几个缺点：首先，遇到日期类型的字段时，它显示出来的却是数字形式（虽然骨子里还是日期没错），必须自己手动把它的格式调整过来。再者，如果不是 Excel 的 Database，遇到一些图形类型的那种二进制数据，CopyFromRecordSet 方法也会失灵（如果用自定义的方法就可以避免不显示）。最后，它不会显示出字段名称这个标题行（或许也算是缺点，因为应该要有参数来选择是否要显示）。

范例 1-36　用 Select_Into 方法指定存储位置

最后介绍一种更直接了当的方法，但这只在你的表来源是 Excel 且必须把查询结果放到另一个 Excel 文件时才适用。

找数据_使用 Select_Into 方法.xls>

```
Public Sub RunMacro()
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As ADODB.Connection
    Dim loADODBCommand As ADODB.Command

    ActiveWorkbook.Save

    lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                         "DBQ=" + ActiveWorkbook.FullName + ";" & _
```

```
                                "ReadOnly=False"

    lcCommandText = "select * into [Excel 8.0;Database=
c:\result.xls].[sheet_result] from [sheet1$]"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    loADODBConnection.Open lcConnectionString

    Set loADODBCommand = CreateObject("ADODB.Command")
    Set loADODBCommand.ActiveConnection = loADODBConnection
    loADODBCommand.CommandText = lcCommandText
    loADODBCommand.Execute

    loADODBConnection.Close
End Sub
```

由于这是一种“查询后再写入”的动作，而不是“纯查询”，所以要用 ADODB.Command 而非 ADODB.Recordset 对象。另外，此例中的 c:\result.xls 不必事先存在（如果事先存在也没关系），但 c:\result.xls 中则不可以存在 sheet_result。就是文件存不存在没关系，但工作表不能是已经存在的工作表。

这种存数据的方式非常简洁，本书第 2 章的 SQL 范例集会采用此种方式。

范例 1-37 用 Insert_Into 方法为多个查询结果指定同一个存储位置

如果查询了两次（或更多次），而想把这两次的结果都放在同一个工作表的话，就改用 insert 指令（当然也可以在 select 里用 union，不过 union 太多则会显得指令有些冗长，因为 union 要通通写在一道指令里）：

找数据_使用 Insert_Into 方法.xls

```
Public Sub RunMacro()
    Application.DisplayAlerts = False
    Workbooks.Add
    ActiveWorkbook.ActiveSheet.Cells(1, 1).Value = "编号"
    ActiveWorkbook.ActiveSheet.Cells(1, 2).Value = "姓名"
    ActiveWorkbook.ActiveSheet.Cells(1, 3).Value = "生日"
    ActiveWorkbook.SaveAs ("c:\result.xls")
    ActiveWorkbook.Close
    Application.DisplayAlerts = True

    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As ADODB.Connection
    Dim loADODBCommand As ADODB.Command

    ActiveWorkbook.Save

   lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)}; " & _
                        "DBQ=" + ActiveWorkbook.FullName + ";" & _
                          "ReadOnly=False"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    loADODBConnection.Open lcConnectionString

    Set loADODBCommand = CreateObject("ADODB.Command")
    Set loADODBCommand.ActiveConnection = loADODBConnection

    lcCommandText = "insert into [sheet1$] in '' [Excel 8.0;
Database=c:\result.xls] select * from [sheet1$] where 编号 < 10"
    loADODBCommand.CommandText = lcCommandText
    loADODBCommand.Execute

    lcCommandText = "insert into [sheet1$] in '' [Excel 8.0;
Database=c:\result.xls] select * from [sheet1$] where 编号 > 10"
    loADODBCommand.CommandText = lcCommandText
    loADODBCommand.Execute
```

```
    loADODBConnection.Close
End Sub
```

跟 select into 方法不同的是，用 insert into 方法目的工作簿中的工作表必须事先存在，且字段标题要跟 select 出的结果一致（也就是说，要在该工作表的第一行有字段标题，而且数据类型跟名称都要一致）。

把其他数据库的数据导入 Excel

ADO 除了可以找 Excel 的数据之外，当然也可以找别的 Database（甚至连结构化的文本文件都可以找），只要有安装该 Database 的 ODBC Driver 即可，如果是 Windows 2000 的话，一般都已经自带 Access、Excel、Visual FoxPro、SQL Server、Oracle 等 ODBC Driver。

知识点：SQL 访问其他数据库的技术要点

方法再简单不过：只要把之前讲过的例子（范例 1-2）中数据库连接字符串（lcConnectionString 那个变量）的值改变即可：

```
lcConnectionString = "Driver={Microsoft Excel Driver (*.xls)};
```

对 ADO 而言，不管是 Excel、Access、FoxPro、结构化文本文件、SQL Server、Informix、Oracle、DB2 等，都是数据库。不同数据库的运作机理大都相同，同一个 SQL 指令也都可以畅行无阻，除非你要用到某种数据库特有的功能（例如 Oracle 强大的 PL/SQL）。

例如，对于 SQL Server 数据库，这个字符串的值为：

```
lcConnectionString = "Provider=SQLOLEDB
```

范例 1-38　把 SQL Server 数据库数据存成 Excel 文件

下面是个读取 SQL Server 数据库自带文件 NorthWind 的范例程序。

找数据_从 SQL_Server.xls

```
Public Sub RunMacro()
    Dim lcConnectString, lcCommandText As String
    Dim loADODBConnection As ADODB.Connection
    Dim loADODBRecordset As ADODB.Recordset

    ActiveWorkbook.Save

    lcConnectionString = "Provider=SQLOLEDB;Data Source=(local)\
SS; Initial Catalog=Northwind;User Id=sa;Password=sa"

    lcCommandText = "select title, birthdate from employees"

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open lcConnectionString
    loADODBRecordset.Open lcCommandText, loADODBConnection

    Sheets.Add
    Dim r, f As Integer
    r = 1
    For f = 0 To loADODBRecordset.Fields.Count - 1
       Sheets(1).Cells(r, f + 1) = loADODBRecordset.Fields(f).Name
    Next
```

```
    While Not loADODBRecordset.EOF
        r = r + 1
        For f = 0 To loADODBRecordset.Fields.Count - 1
          If loADODBRecordset.Fields(f).Type <> adLongVarBinary Then
                Sheets(1).Cells(r, f + 1) = loADODBRecordset.
Fields(f).Value
            End If
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close

    Sheets(1).Cells.EntireColumn.AutoFit
End Sub
```

连接数据库的字符串中的“(local)\SS”是我的 SQL Server 连接数据库字符串，要在你的计算机执行，请把它改成你的连接数据库字符串即可（当然，账号及密码也要改过来）。

如果不知道数据库连接字符串，可以连到这个网站：

http://www.connectionstrings.com

已经有人帮我们整理好了。

如果你的需求只是单纯把 Excel 当作数据查询、绘制图表的前端工具，那么以 ADO 连接到后端数据库、再在 Excel 中呈现，不失为一种快速省事的方法。

把动态网页上的数据抓取成 Excel 文件

用 Web 查询、用 TelePro 之类的软件、甚至用 Copy&Paste……

没错，这些方法都可以用，但大都仅限于所谓的静态网页，如果你要读取的网页是动态查询出来的（通常有个 asp、aspx、php、pl 等结尾的网页名称），而且包含了许多你不想要的信息（例如横幅广告），那么大概很难有任何一个软件可以在不写任何程序代码的状况下完全满足你的需求。你还是必须自己多少写一点，或是通过复杂的操作流程获得你真正想要的信息。

知识点：动态网页信息读取的概念

或许有些软件号称可以读取动态网页，但那也是只能读取“某一页”而已，我们这里说的“动态网页读取”是要能“连续读取”、“只读取我们想要的”才算。最常见的就是下载整个讨论区的数据，由于这种性质的数据读取通常都不会只有一页，所以不大可能有那么聪明的软件，可以分析下一页要怎么读取，并且只读取我们想要的。如果说还要通过一堆参数告诉该软件怎么找，还不如自己写一个比较快。

我说“静态”，有些人可能不太同意：“明明可以输入 asp 这种动态网址呀！怎么说是静态数据呢？”

我所谓的“静态数据”不是“静态动态网页”那种“静态动态”的定义，而是说，你只能找某个网页上“看得到的数据”，如果那些数据本身只是“超级链接的集合”——几乎所有的动态网页都长得这个样子，像是讨论区、电子读物、搜索引擎、图文列表等，因为计算机屏幕就那么大，信息提供者要在一页中塞进尽可能多的信息就必须用超级链接。

此时，你真正想找的其实是那些超级链接所指向的内容，而非

超级链接本身，Excel 的上述功能对此种信息需求是无法满足的。

而且，如果 A 网站跟 B 网站的网页信息结构不同（绝大多数的情况是不会相同的），你针对 A 网站所设计的查询程序也只能用来查询 A 网站，而不能用来查询 B 网站。必须针对每个网站特有的信息结构（或说查询参数，就是 asp“问号”后面接的那些参数）来设计程序。

知识点：动态网页信息读取技术

读取网页的流程跟程序技巧是相同的，都是利用微软所提供的 IE 对象。

而且幸运的是，这类程序都不算很复杂，一个网站设计一个读取程序也还好。但是，即使有了这样的程序，要非常精准的读取到你真正要的信息（例如在方格中的股价信息）还是得靠一些运气——如果该网站的查询参数改变了，你的程序也要随之改变。而前面提到的横幅广告这种东西，更是极难完全过滤掉，因为它有可能是随机出现的。

所以，除非该网页非常结构化，否则我们也只能读取到整个网页的信息，然后尽可能地分析该网页的信息结构，砍掉不要的东西，然后把所要的塞进数据库或 Excel 里。

哪里有方法可以真正“无杂质”地找到网页上的信息吗？也是有的，如果该网站有提供公开的 Web Service 可以让别人以此读取信息，或把该网页以结构化的 XML 呈现，那就可以完全过滤掉不想要的东西了。只是这种状况相当少见，大多是相关集团内的

企业才会这样相互分享信息，就是 B2B 啦。近来有些“网格”（Blog）式的网站也会提供 XML 的显示功能，这种就可以“无杂质”地找到数据。

范例 1-39　抓取动态网页信息的范例程序

因为动态网页非常复杂，所以这里给的不是“通用”的范例，它仅能读取某个网站内的某一版的数据：

上网找数据.xls

```
Public Sub GetDataFromInternet()
    Cells.Clear

    Dim lcaLink()
    Dim ie As Object
    Dim i, n, k, c As Integer
    Dim lnNextPage As Integer
    Dim addr As String

    addr = "http://www.bdfx.net.cn/"
    Set ie = CreateObject("InternetExplorer.Application")
    ie.Visible = True
    ie.Navigate (addr)



    c = 0
    Do
        While Left(ie.StatusText, 2) <> "完毕" Or ie.Busy
        Wend

        lnNextPage = 0
        n = 0
```

```
        For k = 0 To ie.document.body.all.Length - 1
            If Not ie.document.body.all(k) Is Nothing Then
                If ie.document.body.all(k).NodeName = "A" Then
                    If Left(ie.document.body.all(k).href,
Len(addr)) = addr Then
                        n = n + 1
                        ReDim Preserve lcaLink(n)
                        lcaLink(n) = ie.document.body.all(k).href
                    End If

                   If Left(ie.document.body.all(k).href, 41) = _
                    "javascript:__doPostBack('Grid$_ctl1$_
ctl3" Then
                        If ie.document.getElementByID ("Grid:_
ctl1:_ctl0").Value = _ie.document.getElementByID("Grid:_ctl1:_
ctl0"). Length Then
                            lnNextPage = 0
                        Else
                            lnNextPage = ie.document.
getElementByID("Grid:_ctl1:_ctl0").Value + 1
                        End If
                    End If
                End If
            End If
        Next

        For i = 1 To n
            ie.Navigate (lcaLink(i))
            While Left(ie.StatusText, 2) <> "完毕" Or ie.Busy
            Wend
            c = c + 1
            Cells(c, 1) = ie.document.body.innertext
        Next

        If lnNextPage <> 0 Then
            ie.Navigate (addr = "http://162.105.6.229/sg/")
```

```
            While Left(ie.StatusText, 2) <> "完毕" Or ie.Busy
            Wend
            ie.document.getElementByID("Grid:_ctl1:_ctl0").
Value = lnNextPage
            ie.document.form1.submit
        Else
            Exit Do
        End If
    Loop

    ie.Quit
    Set ie = Nothing
End Sub
```

范例程序注意事项

使用上述范例程序时，应注意以下问题：

（1）此范例仅限用在微软的 IE。我是在 Windows 2000 中文专业版用 IE 6 中文版执行这个范例的，不过基本上 IE 5 应该也行得通，如果不行，就请下载 IE 6 啰！

（2）网页的超级链接一层一层下去极可能无穷无尽没完没了，所以本范例只深入一层做搜索，绝大部分的需求应该都是这个样子，因为绝大部分的网站不会不友善到一直要用户不断深入地点击超级链接才能浏览内容。

（3）再次强调，本范例不是一个“通用”的范例，要读取其他网站内容必须针对你的需求稍微修改一下：在 GetDataFromInternet()函数中，用你要求的网页地址对 Addr 变量重新赋值即可。因为超级链接字符串的内容不尽相同，我们有很

多方法去判定哪些超级链接的字符串是我们想要的，最简单的就是利用 left 函数去读取网页。例如 156 个超级链接中，我们想要 20 个。通常那 20 个超级链接字符串的前（就是“左”，所以说利用 left 函数）几个字符都是一样的。你可以先用鼠标右键将那 20 个超级链接的内容找出来瞧瞧就知道了。

（4）我的经验是，以微软最新的 ASP.NET 技术写的网页比较难找（如本范例），因为分页信息常常不是一个单纯的超级链接（信息一多时，一定会有分页的），而是 Form Submit 之后的结果（postback）。此时必须分析该网站设计者对“下一页”的处理方式，本范例可以通过设置下拉式菜单的分页顺序来设置。其他很多网页对分页的作法多半很简单，就是一个超级链接，然后有着“下一页”之类的文字，通过像是 ie.document.body. all(k).innertext 属性就可以判断，然后就可纳入“待会要 Navigate 的 Page 列表”里。

（5）如果你想知道更多关于 IE 的属性，可以执行如下步骤：

- 步骤 1　打开 Excel。
- 步骤 2　打开“控件工具箱”工具栏。
- 步骤 3　单击最后一个有“...”的那个“其他控件”图标。
- 步骤 4　从弹出的列表中选择“Microsoft Web 浏览器”选项，这时你的鼠标光标会变成十字状，把它在工作表上拉出一块，可以看到一个 Windows 的 logo，此时就好像你的 Excel 上有个 IE 了。
- 步骤 5　用鼠标双击该 IE 控件，并找到其 WebBrowser1_Document Complete 事件，输入 Stop 命令。
- 步骤 6　按 Alt+F11 键切换到 VBA 环境。

- 步骤 7　启动“视图/监视窗口”，在监视窗口处按鼠标右键，添加一个监视，在“表达式”输入框中输入 WebBrowser1。
- 步骤 8　按 Ctrl+G 键切换到立即窗口。
- 步骤 9　输入这样的网址（或一个你喜欢的网址）：

```
Sheet1.WebBrowser1.Navigate("http://www.pku.edu.cn")
```

- 步骤 10　等到网页下载完成，WebBrowser1_ DocumentComplete 事件中的 Stop 命令也随即生效。此时，可以去瞧瞧监视窗口中 WebBrowser1 的各项属性了，跟资源管理器一样，它是以树状结构显示这些信息的。

（6）如果要找的网页本身的信息非常结构化，也就是，它就是一个 HTML Table 的话，那么可以直接用：

```
Set loTable = Workbooks.Open( _
"http://你的账号:你的密码@要去的网址/要找的 html 文件")
```

然后再将它另存为一个新的 Excel 文件，就可以用 SQL 去操作它了。请参阅下载程序代码中的“上网找 HTML_Table.xls”文件（账号、密码以及网址请自行修改）。

本书第 2 章还有一个上网找数据的范例，利用 IE 找到网页上的所有 HTML Table(s)，然后分别放到 Excel 的不同工作表中，该范例就比较通用些（但也是不一定“完全”符合你的要求，我说过了，这真的很难）。

有时会找到“要求你登录”的页面，这时，请先按照该网站的登录程序进行登录之后再执行此程序，不然每页都会找到同样

的信息（也就是“要求你登录”）。

关于 SQL……

好了，练就了“以 SQL 的方式操作 Excel”这一招，在 Excel 做为一个数据处理（说数据处理，因为我没说其他诸如图表、VBE、函数、一般操作等领域也可以用数据库的方式，事实上，那些领域并不适合用此方式处理）的工具上，相信应该没什么可以难倒你的问题。用 Excel 的人大概可以分为两种，一种是一般用户（end user），用鼠标画画表格；另一种是超级用户（power user），自己写 VBA 来节省力气。如果你属于后者，但不属于信息部门的员工，我可以告诉你，他们写数据库程序也是利用类似的技巧，当然会更复杂一些（例如利用面向对象的方法），不过基本上就是利用 SQL。

不用 SQL 当然也可以解决问题，但利用 SQL 却能在你和团队之间、以及现在的你和过去的你之间搭起一道沟通的桥梁：因为这种写程序的方式好比大家都用国语交谈一般，不会鸡同鸭讲。

Excel 虽然可以做为数据库，但由于一个 Table（工作表）只能容纳 65535 个（扣掉 1 个字段标题），而且没有 Primary Key 的概念，所以当你的数据量开始膨胀时，还是认真考虑把数据交给“正宗”的 DBMS，如 SQL Server 来管理比较恰当。

把 Excel 本身当数据库或利用 ADO 从别的数据库找数据都不算是新鲜玩意，但把这个观念活用后，配合 meta data 的概念，我

相信全世界都找不到这样一本书，我在 Google 上也没见过任何关于这种技巧的讨论。而利用 IE DOM 从网络上找数据的那个方法，更可以解决许多长久以来用 Web Query 无法解决的头痛问题。

这是一本不一样的 Excel 书，但愿你能从中获得动力以及发现的乐趣。

欢迎你来信讨论“以 SQL 的方式操作 Excel”的相关问题，但务必把问题描述清楚，最好有原始数据的图片，而且尽量不要有错别字，以免语义含混。我的 E-mail：

excel_excel@hotmail.com

心得栏

CHAPTER

2

SQL实例集

这部分的范例是给您熟悉 SQL 技巧用的（就是不必管范例的实用性或真实性，范例都比较简单，不要关心范例本身，多多体会范例内在的抽象意义——关键的程序代码）。

范例 2-1　找出请病假超过（含）两次者

原始数据（见图 2-1）：

	A	B	C
1	姓名	请假时间	请假原因
2	John	2004-02-05	病假
3	Calvin	2004-04-11	休假
4	Chris	2004-03-01	病假
5	Chris	2004-03-19	休假
6	Irene	2004-01-05	休假
7	Rita	2004-03-01	病假
8	Irene	2004-04-09	休假
9	Rita	2004-04-10	事假
10	Chris	2004-02-03	病假
11	Urania	2004-5-1	休假
12	John	2003-9-30	病假
13	John	2003-10-1	病假

图 2-1

程序代码：

SQL 范例_找出病假超过两次者.xls

```
Public Sub RunMacro()
    For Each loWindow In Application.Windows
        If LCase(loWindow.Caption) = "result.xls" Then
            loWindow.Close (False)
        End If
    Next

    If Dir("c:\result.xls") <> "" Then
        Kill "c:\result.xls"
    End If

    ActiveWorkbook.Save

    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")
```


```
    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=" + ActiveWorkbook.FullName + ";" & _
                      "ReadOnly=True"

    loRecordset.Open _
    "select * into [Excel 8.0;Database=c:\result.xls].[sheet1]
from [sheet1$] where 姓名+请假原因 in " & _
    "(select 姓名+请假原因 from [sheet1$] where 请假原因 = '病
假' group by 姓名+请假原因 having count(*) > 1) order by 姓名,
请假时间", _
    loConnection

    loConnection.Close

    Workbooks.Open ("c:\result.xls")
End Sub
```

执行结果（见图 2-2）：

	A	B	C
1	姓名	请假时间	请假原因
2	Chris	2004-2-3	病假
3	Chris	2004-3-1	病假
4	John	2003-9-30	病假
5	John	2003-10-1	病假
6	John	2004-2-5	病假
7			

图 2-2

关键代码评析：

（1）“姓名+请假原因”可看作一个字段表达式。字段表达式等同字段，字段可以做的事字段表达式也都可以。

（2）where 条件中的 in 是一个特殊的运算子，用来筛选该字段的可能值，这样除了可以不必用很多 or 之外，像这种根本不会

事先知道有多少可能值的状况下，也必须用 in（当然还有其他解法，但 in 最为直观）。你可以把 in 后面的东西当作是一个数组来看待。更精确来说，它是仅具备一个字段的 Table。不过 in 后面也不是非接一个 Table 不可，不是说可以不必用很多 or 吗？所以 in 还可以直接这样用：

```
    select * into [Excel 8.0;Database=c:\result.xls]. [sheet1]
from [sheet1$] where 姓名+请假原因 in ('Chris 病假', 'Rita 病假
', 'Irene 病假') order by 姓名, 请假时间
```

（3）写 group by、having count(*)时，不一定要把 count(*)也一起 select 出来。此例中，不把 count(*) select 出来的原因是，这样 in 后面的字段就有两个了：“姓名+请假原因”以及“count(*)”。但 in 前面的字段却只有一个：“姓名+请假原因”。这样会出现 SQL 语法错误。

（4）本例中，in 后面接的是所谓的“子查询”（sub-query），可以这样理解：先把请的是病假、而且请假超过（含）两次者找出来，然后再按照这结果从原 Table 筛选出对应的记录。

（5）如果想找出请假超过（含）两次者（不限于病假）：

```
    select * into [Excel 8.0;Database=c:\result. xls].[sheet1]
from [sheet1$] where 姓名+请假原因 in (select 姓名+请假原因 from
[sheet1$] group by 姓名+请假原因 having count(*) > 1) order by
姓名, 请假时间
```

把 where 拿掉即可。

范例 2-2 找出请假超过（含）两天者

原始数据（见图 2-3）：

	A	B	C	D
1	姓名	请假开始时间	请假结束时间	请假原因
2	John	2004-02-05	2004-02-05	病假
3	Calvin	2004-04-11	2004-04-11	休假
4	Chris	2004-03-01	2004-03-01	病假
5	Chris	2004-03-19	2004-03-19	休假
6	Irene	2004-01-05	2004-01-10	休假
7	Rita	2004-03-01	2004-03-01	病假
8	Irene	2004-04-09	2004-04-09	休假
9	Rita	2004-04-10	2004-04-13	事假
10	Chris	2004-02-03	2004-02-03	病假
11	Urania	2004-5-1	2004-5-1	休假
12	John	2003-8-31	2003-9-1	病假
13	John	2003-10-1	2003-10-1	病假
14				

图 2-3

关键程序代码：

SQL 范例_找出请假超过两天者.xls>

```
select 姓名, 请假开始时间, 请假结束时间, 请假结束时间-请假开始时间 as 请了几天, 请假原因 from [sheet1$] where 请假结束时间-请假开始时间+1 >= 2 order by 姓名, 请假开始时间
```

执行结果（见图 2-4）：

	A	B	C	D	E
1	姓名	请假开始时间	请假结束时间	请了几天	请假原因
2	Irene	2004-1-5	2004-1-10	6	休假
3	John	2003-8-31	2003-9-1	2	病假
4	Rita	2004-4-10	2004-4-13	4	事假

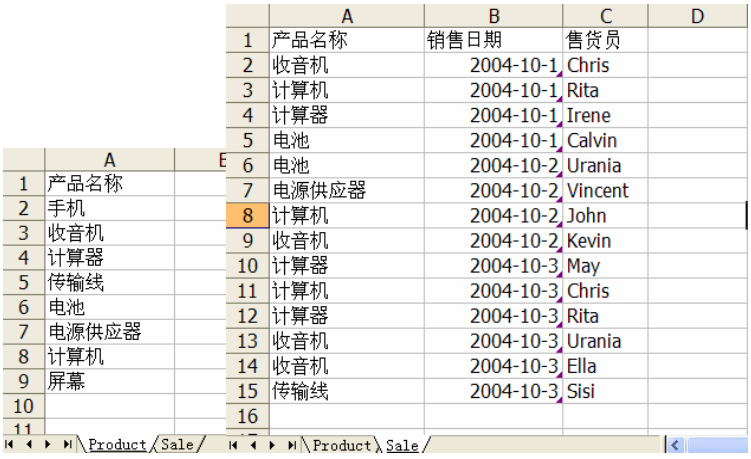
图 2-4

注意，这里找出的是一次请假天数超过（含）两天的请假记录，不是总请假天数或总请假次数超过（含）两的请假记录。

范例 2-3 找出“出现在第 1 张工作表”但“未出现在第 2 张工作表”的记录

这是 SQL 中典型的“not in”应用。“不在……之列”的“不在”，不就是 not in 吗？

原始数据（见图 2-5）：



	A
1	产品名称
2	手机
3	收音机
4	计算器
5	传输线
6	电池
7	电源供应器
8	计算机
9	屏幕
10	
11	

Product / Sale

	A	B	C	D
1	产品名称	销售日期	售货员	
2	收音机	2004-10-1	Chris	
3	计算机	2004-10-1	Rita	
4	计算器	2004-10-1	Irene	
5	电池	2004-10-1	Calvin	
6	电池	2004-10-2	Urania	
7	电源供应器	2004-10-2	Vincent	
8	计算机	2004-10-2	John	
9	收音机	2004-10-2	Kevin	
10	计算器	2004-10-3	May	
11	计算机	2004-10-3	Chris	
12	计算器	2004-10-3	Rita	
13	收音机	2004-10-3	Urania	
14	收音机	2004-10-3	Ella	
15	传输线	2004-10-3	Sisi	
16				

Product / Sale

图 2-5

我们要找出“出现在第 1 张工作表”但“未出现在第 2 张工作表的记录”，就是“还没卖掉的产品”。

关键程序代码：

SQL 范例_找出出现在第 1 张工作表但未出现在第 2 张工作表的记录.xls

```
select 产品名称 as 还没卖掉的产品 into [Excel 8.0;Database=c:\result.xls].[sheet1] from [product$] where 产品名称 not in (select 产品名称 from [sale$])
```

执行结果（见图 2-6）：

	A
1	还没卖掉的产品
2	手机
3	屏幕

图 2-6

跟 in 一样，not in 左边跟右边（或说前面跟后面）的字段数目跟数据类型必须一致。not in 后面也可以接一个数据清单，不是一定要 SQL 子查询不可。

范例 2-4　合并两张工作表的记录

原始数据：不同月份的销售记录放在不同的工作表中，如图 2-7 所示。

	A	B	C
1	产品名称	销售日期	售货员
2	收音机	2004-10-1	Chris
3	计算机	2004-10-1	Rita
4	计算器	2004-10-1	Irene
5	电池	2004-10-1	Calvin
6	电池	2004-10-2	Urania
7	电源供应器	2004-10-2	Vincent
8	计算机	2004-10-2	John
9	收音机	2004-10-2	Kevin
10	计算器	2004-10-3	May
11	计算机	2004-10-3	Chris
12	计算器	2004-10-3	Rita
13	收音机	2004-10-3	Urania
14	收音机	2004-10-3	Ella
15	传输线	2004-10-3	Sisi
16			

	A	B	C
1	产品名称	销售日期	售货员
2	收音机	2004-11-1	Chris
3	计算机	2004-11-2	Rita
4	计算器	2004-11-3	Irene
5	电池	2004-11-4	Calvin
6	电池	2004-11-5	Urania
7	电源供应器	2004-11-6	Vincent
8	计算机	2004-11-7	John
9	收音机	2004-11-8	Kevin
10	计算器	2004-11-9	May
11	计算机	2004-11-10	Chris
12	计算器	2004-11-11	Rita
13	收音机	2004-11-12	Urania
14	收音机	2004-11-13	Ella
15	传输线	2004-11-14	Sisi
16			

图 2-7

关键程序代码：

SQL 范例_合并两张工作表的记录.xls

```
select * into [Excel 8.0;Database=c:\result.xls].[sheet1]
from (select * from [sale_2004_10$] union select * from
[sale_2004_11$]) order by 销售日期, 售货员
```

重点在 union 关键词，union 可以连结多个包含相同数据结构的 Table，所谓相同的数据结构指的是 union 的两端（union 的“左右”或说“前后”，如果连结两个 Table 的话。如果连结多个 Table 则应该说“每一端”，字段数目、字段数据类型必须一致。

union 出来的结果还是 Table，所以可以直接 select from 该 Table。

执行结果（见图 2-8）：

	A	B	C
1	产品名称	销售日期	售货员
2	电池	2004-10-1	Calvin
3	收音机	2004-10-1	Chris
4	计算器	2004-10-1	Irene
5	计算机	2004-10-1	Rita
6	计算机	2004-10-2	John
7	收音机	2004-10-2	Kevin
8	电池	2004-10-2	Urania
9	电源供应器	2004-10-2	Vincent
10	计算机	2004-10-3	Chris
11	收音机	2004-10-3	Ella
12	计算器	2004-10-3	May
13	计算器	2004-10-3	Rita
14	传输线	2004-10-3	Sisi
15	收音机	2004-10-3	Urania
16	收音机	2004-11-1	Chris
17	计算机	2004-11-2	Rita
18	计算器	2004-11-3	Irene
19	电池	2004-11-4	Calvin
20	电池	2004-11-5	Urania
21	电源供应器	2004-11-6	Vincent
22	计算机	2004-11-7	John
23	收音机	2004-11-8	Kevin
24	计算器	2004-11-9	May
25	计算机	2004-11-10	Chris
26	计算器	2004-11-11	Rita
27	收音机	2004-11-12	Urania
28	收音机	2004-11-13	Ella
29	传输线	2004-11-14	Sisi

图 2-8

举一反三，如果想合并 3 个（相同数据结构的）工作表：

```
    select * into [Excel 8.0;Database=c:\result.xls].[sheet1]
from (select * from [sale_2004_10$] union select * from
[sale_2004_11$] union select * from [sale_2004_12$]) order by
销售日期, 售货员
```


union 还有一个功用，由于 Excel 的工作表最多仅允许 65535

列（扣掉字段标题那一列），所以如果你的数据量大于这个上限时，又不想用 Access 等数据库管理软件时，可以考虑把数据分开存放在不同的工作表中，之后再以 union 的方式存取它们（这会造成程序较繁复，但的确也是一种方法）。

范例 2-5　合并两个工作簿的记录

利用与上例一样的 union 技巧，也可以合并两个工作簿的记录。

原始数据（见图 2-9）：

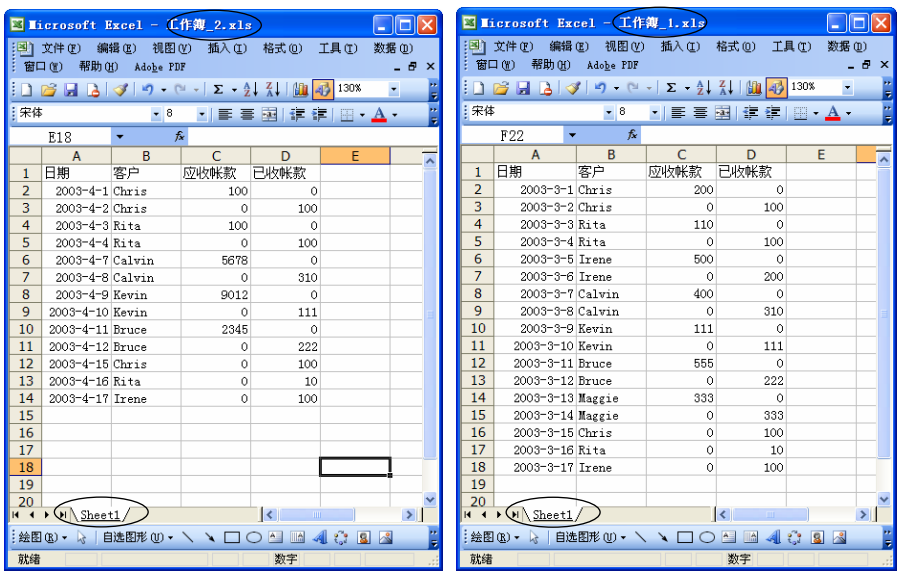


图 2-9

完整程序代码：

SQL 范例_合并两个工作簿的记录.xls

```
Public Sub RunMacro()
    For Each loWindow In Application.Windows
```

```
        If LCase(loWindow.Caption) = "result.xls" Then
            loWindow.Close (False)
        End If
    Next

    If Dir("c:\result.xls") <> "" Then
        Kill "c:\result.xls"
    End If

    ActiveWorkbook.Save

    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")

    ' 第 1 个工作簿（假设数据都在 sheet1)
    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ="+ActiveWorkbook.Path+"\工作簿_1.xls"+";"&_
                      "ReadOnly=True"
    loRecordset.Open "select * into [Excel 8.0;Database=
c:\result_temp.xls].[工作簿_1] from [sheet1$]", loConnection
    loConnection.Close

    ' 第 2 个工作簿（假设数据都在 sheet1）
    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ="+ActiveWorkbook.Path+"\工作簿_2.xls" +";"&_
                      "ReadOnly=True"

    loRecordset.Open "select * into [Excel 8.0;Database=
c:\result_temp.xls].[工作簿_2] from [sheet1$]", loConnection
    loConnection.Close

    ' 合并 2 个工作簿找出来的记录
    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=c:\result_temp.xls;" & _
                      "ReadOnly=True"
```

```
    loRecordset.Open "select * into [Excel 8.0;Database=
c:\result.xls].[sheet1] from ( " & _"select * from [工作簿_1]
union select * from [工作簿_2])", loConnection
    loConnection.Close

    Kill "c:\result_temp.xls"

    Workbooks.Open ("c:\result.xls")
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Name =
"Tahoma"
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Size = 8
    ActiveWorkbook.ActiveSheet.Cells.EntireRow.AutoFit
    ActiveWorkbook.ActiveSheet.Cells.EntireColumn.AutoFit
End Sub
```

范例 2-6 改成绩

SQL 不只是可以查询数据而已，更改数据也很直观。

原始数据：因出题争议，所以每个人的语文成绩无条件加 3 分，超过 100 分的话则以 100 分计，原始成绩表如图 2-10 所示。

	学生	性别	语文	英语	数学
2	孙彦姿	女	48	60	30
3	小S	女	38	76	10
4	莫文卫	女	40	89	90
5	菜一邻	女	100	90	100
6	苏有彭	男	100	12	99
7	施义楠	男	100	78	100
8	蔡灿德	女	25	65	86
9	林新如	女	35	34	0
10	大S	女	98	100	100
11	范执委	男	92	99	88

图 2-10

关键程序代码：

SQL 范例_改成绩.xls

```
Public Sub RunMacro()
    ActiveWorkbook.Save

    Set loConnection = CreateObject("ADODB.Connection")
    Set loCommand = CreateObject("ADODB.Command")

    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=" + ActiveWorkbook.FullName + ";" & _
                       "ReadOnly=True"

    Set loCommand.ActiveConnection = loConnection
    loCommand.CommandText = "update [score$] set 语文 = iif(语
文 + 3 > 100, 100, 语文 + 3)"
    loCommand.Execute

    loConnection.Close
End Sub
```

因为要执行的动作不是查询而是更新，所以用 ADODB.Command 对象而非 ADODB.Recordset 对象。

也可以不用 iif 而直接用 where 条件：

```
update [score$] set 语文 = 语文 + 3 where 语文 + 3 < 100
```

也就是说，加 3 分之后不会超过 100 分的才去加分。

之前曾提过，Excel 当作数据库时不支持删除操作（真正标准的 SQL 里是有这种操作的），但可以用“消去法”。就是所谓“删除不要的”，就等同于“找出想要的”。所以，想要剔除数学低于 90 分者，就是查询出数学高于（含）90 分者（into [Excel 8.0；

Database=c:\result.xls].[sheet1]这段语法都一样，之后除非必要，不然就不再列出了）：

SQL 范例_剔除数学低于 90 分者.xls

```
select * from [score$] where 数学 >= 90
```

消去法的概念不只可以用于消去不想要的记录，还可用于消去某字段中不想要的信息：

```
select left(学生, 1) as 姓 from [score$]
```

注意，left 函数遇到中文就会以中文汉字为单位，所以尽管一个中文汉字占 2 bytes，但还是用 left(学生，1) 而非 left(学生，2)。

范例 2-7　上网找某一网页所有的 HTML Table

假设有个网页长得像这样：结构化的表格 1（表格名称为“销货记录”）如图 2-11 所示。

有 3 个很结构化的表格跟一些不怎么结构的其他信息，这是很普遍的网页形式。如果只想读取那 3 个表格，并把它们分别放到 Excel 的不同工作表中，可以利用 IE 对象的 DOM（文档对象模型）先找到所有的 HTML Table，再找到每个 Table 内的每一个 Row，再往下找到每个 Row 的单元格。

产品名称	销售日期	售货员
收音机	2004/11/01	Chris
计算机	2004/11/02	Rita
计算器	2004/11/03	Irene
电池	2004/11/04	Calvin
电池	2004/11/05	Urania
电源供应器	2004/11/06	Vincent

再来一个表格（无表格名称）：

姓名	请假时间
Rita	2004/02/05
Irene	2004/04/11
Chris	2004/03/01
Chris	2004/03/19
Irene	2004/01/05
Rita	2004/03/01
Chris	2004/04/09
Rita	2004/04/10
Chris	2004/02/03

产品名称	销售日期	售货员
收音机	2004/10/01	Chris
计算机	2004/10/01	Rita
计算器	2004/10/01	Irene
电池	2004/10/01	Calvin
电池	2004/10/02	Urania
电源供应器	2004/10/02	Vincent
计算机	2004/10/02	John
收音机	2004/10/02	Kevin
计算器	2004/10/03	May
计算机	2004/10/03	Chris
计算器	2004/10/03	Rita
收音机	2004/10/03	Urania
收音机	2004/10/03	Ella
传输线	2004/10/03	Sisi

不结构化的数据 1...

不结构化的数据 2...

结构化的表格 2（无表格名称）：

图 2-11

上网找某一网页的所有 HTML_Table.xls

```
Public Sub GetTableFromInternet()
    Dim loIE As Variant

    Set loIE = CreateObject("InternetExplorer.Application")
    loIE.Navigate ("file://" & ActiveWorkbook.Path &
"\web_data.html")
    While Left(loIE.StatusText, 2) <> "完毕"
    Wend

    Dim loTable, loRow, loCell As Variant
    Dim r, c As Integer

    For Each loTable In loIE.Document.body.getElementsByTagName("Table")
        Sheets.Add
        If loTable.ID <> "" Then
            Sheets(1).Name = loTable.ID
        End If

        r = 0
```

```
        For Each loRow In loTable.Rows
            r = r + 1
            c = 0

            For Each loCell In loRow.Cells
                c = c + 1
                Sheets(1).Cells(r, c) = loCell.InnerText
            Next
        Next

        Sheets(1).Cells.EntireColumn.AutoFit
        Sheets(1).Cells.EntireRow.AutoFit
        Sheets(1).Cells(1, 1).Activate
    Next

    Sheets(1).Activate

    loIE.Quit
End Sub
```

“"file://" & ActiveWorkbook.Path & "\web_data.html"”是一个范例文件，可以把它改成任一个像图 2-11 那样形式的网页的网址。

这个指令：

```
loIE.Document.body.getElementsByTagName("Table")
```

会找到某一网页内中所有的 HTML Table。这是 IE DOM 的一部分，介绍 IE DOM 已经大大超出本书的范围，请连接：

```
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/dhtml_node_entry.asp
```

查询它的对象模型。

HTML Table 可以给它一个名字（设定 ID 属性），如果你读取某个网页，设计该网页的程序设计师有设定这个名字的话，这个范例可以把它自动转成工作表的名称。

范例 2-8　outer Join

连接数据时如果连接不到也要显示出空白。

这是个很常见的需求，本书第 1 章也曾提及。SQL 里的 outer join 就是为了解决此问题而设。

原始数据（见图 2-12）：

	A	B	C
1	编号	姓名	生日
2	1	Chris	1970-1-10
3	2	Rita	1981-11-12
4	3	Irene	1970-9-1
5	4	Calvin	1980-12-12
6	5	Kevin	1971-12-17
7	6	Bruce	1970-11-1
8	7	Maggie	1973-6-5
9	8	Amy	1978-10-10
10	9	Urania	1972-6-16
11	10	Vincent	1961-10-1
12			

	A	B
1	编号	出生地
2	1	Beijing
3	2	USA
4	3	Beijing
5	6	Chicago
6	8	Tokyo
7	10	London
8		

图 2-12

关键程序代码：

SQL 范例_Outer_Join.xls

```
select s1.*, IIF(IsNull(s2.出生地), '(无登记)', s2.出生地) as 出生地 from [sheet1$] s1 left outer join [sheet2$] s2 on s2.编号 = s1.编号
```

执行结果（见图 2-13）：

	A	B	C	D
1	编号	姓名	生日	出生地
2	1	Chris	1970-1-10	Beijing
3	2	Rita	1981-11-12	USA
4	3	Irene	1970-9-1	Beijing
5	4	Calvin	1980-12-12	(无登记)
6	5	Kevin	1971-12-17	(无登记)
7	6	Bruce	1970-11-1	Chicago
8	7	Maggie	1973-6-5	(无登记)
9	8	Amy	1978-10-10	Tokyo
10	9	Urania	1972-6-16	(无登记)
11	10	Vincent	1961-10-1	London

图 2-13

请注意，这个范例跟“找出出现在第 1 张工作表但未出现在第 2 张工作表的记录”不一样，这个范例是，如果某记录出现在第 1 张工作表但未出现在第 2 张工作表，则第 1 张的记录“还是要”列出来，只是在找不到时补个自定义信息罢了（本范例中的“未登记”字样）。

范例 2-9　比较汇总后的结果

计算应收账款总和、已收账款总和后，比较这两者是否一致。

原始数据（见图 2-14）：

	A	B	C	D
1	客户	应收帐款	已收帐款	
2	Chris	200	0	
3	Chris	0	100	
4	Rita	110	0	
5	Rita	0	100	
6	Irene	500	0	
7	Irene	0	200	
8	Calvin	400	0	
9	Calvin	0	310	
10	Kevin	111	0	
11	Kevin	0	111	
12	Bruce	555	0	
13	Bruce	0	222	
14	Maggie	333	0	
15	Maggie	0	333	
16	Chris	0	100	
17	Rita	0	10	
18	Irene	0	100	
19				

图 2-14

关键程序代码：

SQL 范例_比较加总后结果.xls

```
Select 客户, sum(应收账款) as 应收账款总和, sum(已收账款) as 已收账款总和 from [sheet1$] group by 客户 having sum(应收账款) <> sum(已收账款)
```

执行结果（见图 2-15）：

	A	B	C
1	客户	应收帐款总和	已收帐款总和
2	Bruce	555	222
3	Calvin	400	310
4	Irene	500	300
5			

图 2-15

范例 2-10　按照年或月统计销售额

原始数据：常见的销售记录，如图 2-16 所示。

	A	B	C
1	售货员	销售日期	销售员
2	Calvin	2004-10-01	300
3	Chris	2004-01-01	100
4	Chris	2004-11-03	900
5	Ella	2004-11-07	200
6	Irene	2004-10-01	100
7	John	2004-10-02	200
8	Kevin	2004-11-01	400
9	May	2004-11-02	500
10	Rita	2004-02-01	200
11	Rita	2004-11-05	800
12	Sisi	2004-11-07	300
13	Urania	2004-10-02	100
14	Urania	2004-11-05	100
15	Vincent	2004-10-02	100
16	Vincent	2004-10-02	100

图 2-16

关键程序代码：

SQL 范例_按照年或月统计销售额.xls

```
select month(销售日期) as 月份, sum(销售额) as 销售额总和 from [sheet1$] group by month(销售日期)
```

如果是按照年份统计，方法类似：

```
    select year(销售日期) as 年份, sum(销售额) as 销售额总和 from [sheet1$] group by year(销售日期)
```

按照年月也是一样，用字符串相加（请注意，日期要先转换成文字）：

```
    select trim(str(year(销售日期)))+'/'+trim(str(month(销售日期))) as 年月, sum(销售额) as 销售额总和 from [sheet1$] group by trim(str(year(销售日期)))+'/'+trim(str(month(销售日期))) order by trim(str(year(销售日期)))+'/'+trim(str(month(销售日期)))
```

你可以注意到，用 SQL 的方式处理这种加总统计的问题，原始数据并不需要先排序。当然，处理过的结果数据是可以排序的。

按照年月统计会出现如图 2-17 所示的这种结果。

	A	B	
1	年月	销售额总和	
2	2004/1	100	
3	2004/10	900	
4	2004/11	3200	
5	2004/2	200	
6			

图 2-17

没有按照年月排序。这是因为把年月转换成文字后，“2”这个数字本来就会在“1”的后面，所以上述的结果是必然的。

解决之道是把小于 10 的月份前面补零，把“2”变成“02”即可：

```
    select trim(str(year(销售日期))) + '/' +
    iif(month(销售日期) < 10, '0', '')
    trim(str(month(销售日期))) as 年月, sum(销售额) as 销售额总和
    from [sheet1$]
    group by trim(str(year(销售日期))) + '/' +
    iif(month(销售日期) < 10, '0', '') +
    trim(str(month(销售日期)))"
    order by trim(str(year(销售日期))) + '/' +
    iif(month(销售日期) < 10, '0', '') +
    trim(str(month(销售日期)))
```

别忘了 where 条件，SQL 并不是用了 group by 就不能用 where，所以，如果只想统计 Chris 这位售货员的业绩：

```
    select month(销售日期) as 月份, sum(销售额) as 销售额总和 from
[sheet1$] group by month(销售日期) where 售货员 = 'Chris'
```

统计某两位：

```
    select month(销售日期) as 月份, sum(销售额) as 销售额总和 from
[sheet1$] group by month(销售日期) where 售货员 = 'Chris' or 售
货员 = 'Rita'
```

如果想统计多位，而且不想写很长的 or，别忘了还有 in：

```
    select month(销售日期) as 月份, sum(销售额) as 销售额总和 from
[sheet1$] group by month(销售日期) where 售货员 in ('Chris',
'Rita', 'Irene', 'Calvin')
```

想找出 Chris 这位售货员销售额超过 1000 的月份（where 跟 having 并用，可达到对统计数据再做一次条件过滤的效果）：

```
    select 售货员, month(销售日期) as 月份, sum(销售额) as 销售额总和 from [sheet1$] where 售货员 = 'Chris' group by 售货员, month(销售日期) having sum(销售额) > 1000
```

想找出所有售货员销售额超过 1000 的月份：

```
    select 售货员, month(销售日期) as 月份, sum(销售额) as 销售额总和 from [sheet1$] group by 售货员, month(销售日期) having sum(销售额) > 1000
```

看哪位售货员的业绩最好（把销售额总和按照降序排序）：

```
    select 售货员, month(销售日期) as 月份, sum(销售额) as 销售额总和 from [sheet1$] group by 售货员, month(销售日期) having sum(销售额) > 1000 order by sum(销售额) desc
```

还记得 top 关键词可以找出前几个记录吗？所以，找出第一名的业务员：

```
    select top 1 售货员 from (select 售货员, month(销售日期) as 月份, sum(销售额) as 销售额总和 from [sheet1$] group by 售货员, month(销售日期) having sum(销售额) > 1000 order by sum(销售额) desc)
```

就是把第一次的查询结果再当成来源，然后从这个来源中找出第一个记录。这种两次查询当然可以不止两次（至于最多可以到几次，我没试过，在 Excel 或 ADO 的说明文件也没看到限制，但我想一般而言，3 或 5 次应该就够用了。而且，写太多次还不如

分开写，否则只会把自己和看你程序的人都搞晕）。

像 sum、avg、count 这种聚合函数可以同时使用，所以，同时统计出各个售货员的销售次数与销售总额：

```
select 售货员, month(销售日期) as 月份, count(*) as 销售笔数, sum(销售额) as 销售额总和 from [sheet1$] group by 售货员, month(销售日期)
```

范例 2-11　找出数学考最高分的同学

原始数据：普通的一张成绩表，如图 2-18 所示。

	A	B	C	D	E
1	学生	性别	语文	英语	数学
2	孙彦姿	女	39	60	30
3	小S	女	29	76	10
4	莫文卫	女	31	89	90
5	菜一邻	女	100	90	100
6	苏有彭	男	99	12	99
7	施义楠	男	98	78	100
8	蔡灿德	女	16	65	86
9	林新如	女	26	34	0
10	大S	女	89	100	100
11	范执委	男	83	99	88
12					

图 2-18

关键程序代码：

SQL 范例_找出数学考最高分的同学.xls

```
select * from [score$] where 数学 = (select max(数学) from [score$])
```

子查询不只可以用 in，也可以用“=”。

当然，此例用 in 也会得到一样的结果，但那是对 SQL 而言，对“我们人类”而言，此例用“=”显然较符合语义，因为最高分

只会是一个分数（但不一定是一位同学），所以“=”比“in”更贴近我们想表达的真正意义。

找出销售业绩最好的业务员、最好的销售业绩（不分业务员）、最……的东西，都可以利用此技巧。

执行结果（见图 2-19）：

	A	B	C	D	E	F
1	学生	性别	语文	英语	数学	
2	菜一邻	女	100	90	100	
3	施义楠	男	98	78	100	
4	大S	女	89	100	100	
5						

图 2-19

范例 2-12　找出到过的不重复的国家

看到“不重复”几个字，想到 distinct。

原始数据（见图 2-20）：这是某位糊涂的数据整理员整理的数据结构，一般状况下，除非你的数据量非常大，大到要做数据挖掘（data mining），否则数据结构的设计应该纵向增加，而非横向展开。

	A	B	C	D	E
1	姓名	出差到过的国家	旅游到过的国家	参赛到过的国家	参加会议到过的国家
2	Chris	美国	美国	日本	日本
3	Rita	新加坡	泰国	德国	新加坡
4	Irene	日本	纽西兰	德国	韩国
5	Calvin	朝鲜	英国	法国	加拿大
6	Clemen	印度	日本	日本	马来西亚
7	May	中国	纽西兰	中国	澳洲
8	John	日本	美国	韩国	美国
9	Kenny	美国	希腊	韩国	马来西亚

图 2-20

找出每个人到过的不重复的国家，关键程序代码：

SQL 范例_找出到过的不重复国家.xls

```
select distinct 姓名, 到过的国家
from
(
select 姓名, 出差到过的国家 as 到过的国家 from [sheet1$] union
select 姓名, 旅游到过的国家 as 到过的国家 from [sheet1$] union
select 姓名, 参赛到过的国家 as 到过的国家 from [sheet1$] union
select 姓名, 参加会议到过的国家 as 到过的国家 from [sheet1$]
)
order by 姓名, 到过的国家
```

没有规定一个 Table 只能被 select 一次（待会儿还会谈到 Table 自己跟自己 join 呢）。所以，先用 union“分别”把所有到过国家给找出来，然后再“一次”找出不重复的国家。

执行结果（见图 2-21）：

	A	B
1	姓名	到过的国家
2	Calvin	朝鲜
3	Calvin	加拿大
4	Calvin	法国
5	Calvin	英国
6	Chris	美国
7	Chris	日本
8	Clemen	印度
9	Clemen	日本
10	Clemen	马来西亚
11	Irene	德国
12	Irene	纽西兰
13	Irene	日本
14	Irene	韩国
15	John	美国
16	John	日本
17	John	韩国
18	Kenny	希腊
19	Kenny	美国
20	Kenny	韩国
21	Kenny	马来西亚
22	May	中国
23	May	澳洲
24	Rita	纽西兰
25	Rita	新加坡
26	Rita	德国
27	Rita	泰国

图 2-21

如果把图 2-21 这个执行结果当作原始数据，要找出不重复的国家（即忽略姓名）：

```
select distinct 到过的国家 from [sheet1$]
```

很简单，用 distinct 就好，但如果不重复的国家有好几个呢（不是要列出国家，而是要统计有几个）？

由于 Excel 做为数据库不支持这样的语法（直接在 count 内接一个字段名称），即：

```
select count(distinct 到过的国家) from [sheet1$]
```

所以要转个弯，把 distinct 与子查询并用：

```
    select count(*) from (select distinct 到过的国家 from
[sheet1$])
```

范例 2-13　找出主管所管辖的部属

原始数据：人事基本数据表的一部分，如图 2-22 所示。

关键程序代码：

SQL 范例_找出主管所管辖的部属.xls

```
select a.员工 as 大大老板, b.员工 as 大老板, c.员工 as 主管, d.
员工 as 所管辖部属     from [sheet1$] a, [sheet1$] b, [sheet1$]
c, [sheet1$] d
where b.直属主管 = a.员工
and c.直属主管 = b.员工
and d.直属主管 = c.员工
order by a.员工, b.员工, c.员工, d.员工
```

	A	B
1	员工	直属主管
2	Alice	Ken
3	Amy	Kevin
4	Ben	Maggie
5	Bill	Bill
6	Blinda	Vincent
7	Bruce	Kevin
8	Calvin	Rita
9	Chris	Rita
10	Cooper	Ken
11	Ella	Vincent
12	Eric	Ken
13	Ginger	Vincent
14	Irene	Rita
15	Jacky	Ken
16	John	Maggie
17	Jonna	Ken
18	Ken	Bill
19	Kevin	Bill
20	Maggie	Bill
21	Mary	Ken
22	Pony	Vincent
23	Rita	Bill
24	Snoopy	Maggie
25	Steve	Rita
26	Urania	Maggie
27	Vicky	Maggie
28	Vincent	Bill
29	Cindy	Vicky
30	Clemen	Vicky

图 2-22

这就是同一个 Table 自己跟自己 join 了 3 次，你可以想象有 4 个 Table，只是长得一模一样罢了。如果图 2-22 的人事组织超过 3 层，那就一直 join 下去。自己跟自己做 join 有个专有名词：self-join。self-join 的场合非常多，算是 SQL 的常用技巧。

执行结果（见图 2-23）：

	A	B	C	D
1	大大老板	大老板	主管	所管辖部属
2	Bill	Bill	Bill	Bill
3	Bill	Bill	Bill	Ken
4	Bill	Bill	Bill	Kevin
5	Bill	Bill	Bill	Maggie
6	Bill	Bill	Bill	Rita
7	Bill	Bill	Bill	Vincent
8	Bill	Bill	Ken	Alice
9	Bill	Bill	Ken	Cooper
10	Bill	Bill	Ken	Eric
11	Bill	Bill	Ken	Jacky
12	Bill	Bill	Ken	Jonna
13	Bill	Bill	Ken	Mary
14	Bill	Bill	Kevin	Amy
15	Bill	Bill	Kevin	Bruce
16	Bill	Bill	Maggie	Ben
17	Bill	Bill	Maggie	John
18	Bill	Bill	Maggie	Snoopy
19	Bill	Bill	Maggie	Urania
20	Bill	Bill	Maggie	Vicky
21	Bill	Bill	Rita	Calvin
22	Bill	Bill	Rita	Chris
23	Bill	Bill	Rita	Irene
24	Bill	Bill	Rita	Steve
25	Bill	Bill	Vincent	Blinda
26	Bill	Bill	Vincent	Ella
27	Bill	Bill	Vincent	Ginger
28	Bill	Bill	Vincent	Pony
29	Bill	Maggie	Vicky	Cindy
30	Bill	Maggie	Vicky	Clemen

图 2-23

范例 2-14　找出学生成绩是退步还是进步

原始数据：看看下面这个如图 2-24 所示的数据，很多家长、学生跟老师都会关心。

	A	B	C	D
1	学生	考试日期	数学成绩	
2	孙彦姿	2004-10-1	100	
3	小S	2004-10-1	10	
4	孙彦姿	2004-10-2	90	
5	孙彦姿	2004-10-3	99	
6	小S	2004-10-2	20	
7	小S	2004-10-3	30	
8	蔡灿德	2004-10-1	86	
9	林新如	2004-10-1	100	
10	蔡灿德	2004-10-2	55	
11	蔡灿德	2004-10-3	88	
12	林新如	2004-10-2	90	
13	林新如	2004-10-3	80	

图 2-24

如果想知道每个学生在最接近的连续两次考试中，数学成绩是进步还是退步：

```
select a.学生,
a.考试日期 as 某次考试日期, b.考试日期 as 下次考试日期,
a.数学成绩 as 某次数学成绩, b.数学成绩 as 下次数学成绩
from [score$] a, [score$] b
where b.学生 = a.学生
and b.考试日期 = a.考试日期 + 1
order by a.学生, a.考试日期, b.考试日期
```

执行结果（见图 2-25）：

	A	B	C	D	E
1	学生	某次考试日期	下次考试日期	某次数学成绩	下次数学成绩
2	小S	2004-10-1	2004-10-2	10	20
3	小S	2004-10-2	2004-10-3	20	30
4	蔡灿德	2004-10-1	2004-10-2	86	55
5	蔡灿德	2004-10-2	2004-10-3	55	88
6	孙彦姿	2004-10-1	2004-10-2	100	90
7	孙彦姿	2004-10-2	2004-10-3	90	99
8	林新如	2004-10-1	2004-10-2	100	90
9	林新如	2004-10-2	2004-10-3	90	80

图 2-25

要判断是进步还是退步加个 iif 判断就可以了：

SQL 范例_判断数学成绩进步或退步.xls

```
select a.学生,
a.考试日期 as 某次考试日期, b.考试日期 as 下次考试日期,
a.数学成绩 as 某次数学成绩, b.数学成绩 as 下次数学成绩,
iif(b.数学成绩 > a.数学成绩, '进步', '退步') as 进步或退步
from [score$] a, [score$] b
where b.学生 = a.学生
and b.考试日期 = a.考试日期 + 1
order by a.学生, a.考试日期, b.考试日期
```

请注意，join，不论 self-join 与否，彼此 join 的字段间不一定都是“完全等于”的关系，以此例而言，因为要判断连续日期，所以“b.考试日期”跟“a.考试日期”之间就是相差 1 的关系。其他诸如“>、>=、<、<=、<>…”等关系当然也都可以，视你的需求而定。

范例 2-15　找出进步和退步最大的学生

在两次的数学成绩中（考试日期不一定是连续两天），找出谁进步最多、谁退步最多，就不是“完全等于”的 join 关系，而要利用“大于”：

```
select a.学生,
a.考试日期 as 第 1 次考试日期, b.考试日期 as 第 2 次考试日期,
a.数学成绩 as 第 1 次数学成绩, b.数学成绩 as 第 2 次数学成绩,
(b.数学成绩-a.数学成绩) as 相差分数
from [score$] a, [score$] b
where b.学生 = a.学生
and b.考试日期 > a.考试日期
order by (b.数学成绩-a.数学成绩) desc
```

执行结果（见图 2-26）：

	A	B	C	D	E	F
1	学生	第一次考试日期	第二次考试日期	第一次数学成绩	第二次数学成绩	相差分数
2	蔡灿德	2004-10-2	2004-10-3	55	88	33
3	小S	2004-10-1	2004-10-3	10	30	20
4	小S	2004-10-2	2004-10-3	20	30	10
5	小S	2004-10-1	2004-10-2	10	20	10
6	孙彦姿	2004-10-2	2004-10-3	90	99	9
7	蔡灿德	2004-10-1	2004-10-3	86	88	2
8	孙彦姿	2004-10-1	2004-10-3	100	99	-1
9	林新如	2004-10-2	2004-10-3	90	80	-10
10	林新如	2004-10-1	2004-10-2	100	90	-10
11	孙彦姿	2004-10-1	2004-10-2	100	90	-10
12	林新如	2004-10-1	2004-10-3	100	80	-20
13	蔡灿德	2004-10-1	2004-10-2	86	55	-31
14						

图 2-26

相差分数最大者就是进步最多者，相差分数最小者就是退步最多者。如果只想找出进步最多者和退步最多者这两位则可以利用子查询，将此结果再做一次查询：

SQL 范例_找出数学成绩进步及退步最多者.xls

```
    Dim lcBaseQuery As String
    lcBaseQuery = "(select a.学生, a.考试日期 as 第1次考试日期,
b.考试日期 as 第2次考试日期, " & _
    "a.数学成绩 as 第1次数学成绩, b.数学成绩 as 第2次数学成绩, " & _
    "(b.数学成绩-a.数学成绩) as 相差分数, " & _
    "iif(b.数学成绩-a.数学成绩 > 0, '进步 ', '退步
')+trim(str(abs(b.数学成绩-a.数学成绩)))+' 分' as 批注 " & _
    "from [score$] a, [score$] b " & _
    "where b.学生 = a.学生 " & _
    "and b.考试日期 > a.考试日期 " & _
    "order by (b.数学成绩-a.数学成绩) desc)"

    loRecordset.Open _
    "select * " & _
    "into [Excel 8.0;Database=c:\result.xls].[sheet1] " & _
    "from " & _
    lcBaseQuery & _
    "where 相差分数 = (select max(相差分数) from " & lcBaseQuery
& ") " & _
    "or 相差分数 = (select min(相差分数) from " & lcBaseQuery & ")", _
    loConnection
```

因为 lcBaseQuery 所代表的查询结果（见图 2-26）会一再被使用到，所以先把它存在一个字符串内。

或许你会认为只要把图 2-26 的结果取第一个和最后一个就好了，但会有一个问题，就是进步最多和退步最多者可能都不止一位。所以必须先把进步最多和退步最多的分数找出来（此例中 max

及 min 函数做的事），然后再找出相差分数等于这两个分数的所有记录。

此处对于“进步”的定义是“不进则退”，所以相差分数是“>0”而非“>=0”。

范例 2-16　找出工资高于某人者

self-join 的应用实在是不胜枚举，图 2-27 所示是员工工资表的一部分。

	A	B
1	员工	工资
2	Chris	30000
3	Rita	35000
4	Irene	38000
5	Calvin	37000
6	Steve	20000
7	Kevin	200000
8	Amy	15000
9	Bruce	55000
10	Maggie	65000
11	Urania	12000
12	Ben	23000
13	Vicky	33000
14	John	11000
15	Snoopy	33000
16	Vincent	150000
17	Pony	30000
18	Ginger	10000
19	Ella	13000
20	Blinda	10000
21	Ken	55000
22	Alice	24000
23	Eric	33000
24	Jacky	34000
25	Mary	10000
26	Cooper	10000
27	Jonna	30000
28	Bill	1000000

图 2-27

如果想找出工资比 Chris 高的员工，可以用如下代码：

SQL 范例_找出工资比某人高的所有员工.xls

```
select b.*
from [sheet1$] a, [sheet1$] b
where a.员工 = 'Chris'
and   b.工资 > a.工资 "
order by b.工资 desc, b.员工
```

有时候，self-join 也可以改用子查询来做：

SQL 范例_找出工资比某人高的所有员工_以子查询做.xls

```
select *
from [sheet1$]
where 工资 > (select 工资 from [sheet1$] where 员工 = 'Chris')
order by 工资 desc, 员工
```

范例 2-17　计算百分比排位

成绩排名的方式除了按照总分高低之外，还有一种常见的方式：百分比排位。就是看自己“赢了多少人”，自己的成绩是在百分之多少的排名（图 2-28 中，左图是原始数据，右图是计算后的百分比排位）：

	A	B	C	D	E	F
1	学生	性别	语文	英语	数学	总分
2	菜一邻	女	100	90	100	290
3	大S	女	89	100	100	289
4	范执委	男	83	99	88	270
5	施义楠	男	98	78	100	276
6	莫文卫	女	31	89	90	210
7	苏有彭	男	99	21	99	219
8	小S	女	29	76	10	115
9	孙彦姿	女	39	60	30	129
10	蔡灿德	女	34	65	86	185
11	林新如	女	26	34	0	60

	A	B
1	学生	百分比排位
2	菜一邻	100.00
3	大S	88.89
4	施义楠	77.78
5	范执委	66.67
6	苏有彭	55.56
7	莫文卫	44.44
8	蔡灿德	33.33
9	孙彦姿	22.22
10	小S	11.11
11	林新如	0.00
12		

图 2-28

这种问题同样是利用子查询，而且是字段列表中的子查询：

SQL 范例_计算百分比排位.xls

```
select *
from (
select a.学生,
(select count(*) from [score$] where 总分 < a.总分)/(select
count(*)-1 from [score$])*100 as 百分比排位
from [score$] a)
order by 百分比排位 desc
```

赢了多少人，就是找出总分低于自己的人数。百分比排位就是“赢了多少人/(总人数-1) ”，减 1 是因为要扣掉自己。

请注意，要考虑同分的状况，所以用总分来计算百分比排位。

范例 2-18　计算成绩排名

本书第 1 章曾提到过计算成绩排名的写法，用 self-join 也可以做，其语义跟计算百分比排位很类似，所有人数减去总分比自己低的人数，再减去跟自己同分的人数就是自己的名次：

SQL 范例_计算排名.xls

```
select *
from (select
(select count(*) from [score$])-
(select count(*) from [score$] where 总分 < a.总分)-
(select count(*) from [score$] where 总分 = a.总分 and 学生 <>
a.学生) as 名次, *
from [score$] a)
order by 1"
```

同样要考虑同分的情况，所以要扣掉跟自己同分者。

范例 2-19　寻找每次都比自己考得好的人

原始数据：如图 2-29 所示的数据成绩单。一位尽职敬业的老师，想帮每位同学都找出“每次”都考得比他高的人，做为超越的对象。

	A	B	C
1	学生	考试日期	数学成绩
2	孙彦姿	2004-10-1	100
3	小S	2004-10-1	10
4	孙彦姿	2004-10-2	90
5	孙彦姿	2004-10-3	99
6	小S	2004-10-2	20
7	小S	2004-10-3	30
8	蔡灿德	2004-10-1	86
9	林新如	2004-10-1	70
10	蔡灿德	2004-10-2	55
11	蔡灿德	2004-10-3	88
12	林新如	2004-10-2	90
13	林新如	2004-10-3	80
14	蔡康涌	2004-10-1	100
15	苏有彭	2004-10-1	79
16	蔡康涌	2004-10-2	99
17	苏有彭	2004-10-2	80
18	蔡康涌	2004-10-3	98
19	苏有彭	2004-10-3	81

图 2-29

程序范例代码：

SQL 范例_找出每次都考得分数比自己高的人.xls

```
select 学生_1 as 这位同学, 学生_2 as 想超越的对象
from

(select
a.学生 as 学生_1, a.考试日期 as 考试日期_1, a.数学成绩 as 数学成绩_1,
b.学生 as 学生_2, b.考试日期 as 考试日期_2, b.数学成绩 as 数学成绩_2
from [score$] a, [score$] b
where b.考试日期 = a.考试日期
and   b.学生 <> a.学生
and   b.数学成绩 - a.数学成绩 > 0)
```


```
group by 学生_1, 学生_2
having count(*) =
(select count(*) from (select distinct 考试日期 from [score$]))
```

第一段粗体字所代表的子查询，用 self-join 找出除自己以外、在同一次考试中成绩比自己高的人。找出来的结果，再作为源数据做二次查询（图 2-30 中仅节录苏有彭的记录）：

	A	B	C	D	E	F
1	学生_1	考试日期_1	数学成绩_1	学生_2	考试日期_2	数学成绩_2
2	苏有彭	2004-10-2	80	蔡康涌	2004-10-2	99
3	苏有彭	2004-10-1	79	蔡康涌	2004-10-1	100
4	苏有彭	2004-10-3	81	蔡康涌	2004-10-3	98
5	苏有彭	2004-10-1	79	蔡灿德	2004-10-1	86
6	苏有彭	2004-10-3	81	蔡灿德	2004-10-3	88
7	苏有彭	2004-10-3	81	孙彦姿	2004-10-3	99
8	苏有彭	2004-10-2	80	孙彦姿	2004-10-2	90
9	苏有彭	2004-10-1	79	孙彦姿	2004-10-1	100
10	苏有彭	2004-10-2	80	林新如	2004-10-2	90
11						
12						

图 2-30

所谓的“每次都考得比自己高”的人，以此例而言，就是将考试日期 join 后，分数高过自己 3 次的人。这个“3 次”，就是第 2 段粗体字的查询结果，因为我们不能把这个“3 次”写死，必须要用计算的。

答案已经呼之欲出，找出 3 次都考得比自己高的人，就是以“自己跟每次考得比自己高的同学”做 group by，次数为 3 就对了，例如图 2-30 的“苏有彭 vs 蔡康涌”以及“苏有彭 vs 孙彦姿”。

执行结果如图 2-31 所示：林新如 2004/10/2 跟孙彦姿打平，而我们的条件是要每次都要高，所以孙彦姿还不算是林新如要超越的对象：

	A	B
1	这位同学	想超越的对象
2	小S	蔡灿德
3	小S	蔡康涌
4	小S	孙彦姿
5	小S	林新如
6	小S	苏有彭
7	蔡灿德	蔡康涌
8	蔡灿德	孙彦姿
9	林新如	蔡康涌
10	苏有彭	蔡康涌
11	苏有彭	孙彦姿

图 2-31

也可以写成这样：

SQL 范例_找出每次都考得分数比自己高分的人_另一种写法.xls

```
select a.学生 as 这位同学, b.学生 as 想超越的对象
from [score$] a, [score$] b
where a.学生 <> b.学生
and a.考试日期 = b.考试日期
group by a.学生, b.学生
having sum(IIf(b.数学成绩 - a.数学成绩 > 0, 1, 0)) = sum(1)
order by a.学生, b.学生
```

由此可知，很多问题都可以 SQL 化，把查询结果做两次或 N 次处理。当然，这种多次处理的写法比较复杂，如果你不希望这样，那就先把查询结果放到某个工作表中，然后以该工作表来查询，而不要用到此例中这种子查询。

最后介绍的 self-join 例子是用它来做排列组合（此例只有组合，没有排列，因为不用考虑出现顺序），这是我突然想出来的技巧。

范例 2-20 挑选彩票号码

举个会让你眼前一亮的例子：彩票。这里的彩票是 1~42 找 6 个数字做不重复的组合且不考虑出现顺序。为举例方便，先简化一下题目，假设是从 1~4 找 3 个数字做组合，则共有 C(4,3)=4*3*2/1*2*3=4 种组合：

```
123、124、134、234
```

可以把这个动作想成是图 2-32 所示的这样。

	A	B	C
1	数字	数字	数字
2	1	1	1
3	2	2	2
4	3	3	3
5	4	4	4
6			

图 2-32

因为每个数字都不能重复，所以一开始“A 列取 1，B 列取 2，C 列取 3”，然后分别取“124、134、234”。再进一步想，图 2-32 就是一模一样的 3 个 Table，进行着一个字段（即“数字”）比同名的前一个字段大（不一定是大 1，只要大就可以）的 self-join（还记得吗？self-join 可以不止 join 两个 Table）：

SQL 范例_彩票号码排列组合.xls

```
Select a.数字 as 第一码, b.数字 as 第二码, c.数字 as 第三码
from [sheet1$] a, [sheet1$] b, [sheet1$] c
where b.数字 > a.数字
and   c.数字 > b.数字
order by a.数字, b.数字, c.数字
```

执行结果（见图 2-33）：

	A	B	C	D
1	第一码	第二码	第三码	
2	1	2	3	
3	1	2	4	
4	1	3	4	
5	2	3	4	

图 2-33

提醒你一下，如果你要从 42 个数字中选 6 个，或从 49 个数字中选 6 个，请记得 Excel 只有 65535（扣掉一列当做字段标题）列，而我们的彩票可有五百多万组号码呢！如果不考虑用多个工作表存放的话，看来最多只能从 21 个数字中实现了（C(21,6) 也有高达 54,264 注呢）。

总之，记住一个关键概念：把 self-join 看成是两个（或 N 个）一模一样的表就对了，不要想成是一个表出现多次，那样比较容易搞混。

在实际的使用上，SQL 主要用于查询（观其名就知道了：Structured Query Language），比较关键的部分就是如何写 where 条件，复杂一点的，就像是之前介绍过的那些子查询（包括字段层次、记录层次的子查询）、self-join、outer join…等。熟悉 where 条件的运用，不只可以用于查询上，也可用于添加、删除、修改上。

除了比较不常用的 all、any、exists 之外，本书的 SQL 范例已经涵盖所有查询会用到的指令。但 SQL 的应用千变万化，不可能一一列举这些变化，还是那句话，觉得有疑惑的话，请通过 E-mail 与笔者讨论！讨论时，最好如同本书这样，先把原始数据以图的方式附上，再清楚描述要做出什么样的执行结果。然后尽量不要错字连篇，以免影响语义。

心得栏

CHAPTER

3

单元格自定义函数

本书第 1 章曾经提到，使用元数据把所有的单元格整理成表后，即可以在ADODB.RecordSet的循环内把单元格一个个地传给我们所设计的自定义函数，本章收录了许多实用的自定义函数，你可以单独调用（处理少量数据时），也可以在 ADODB.RecordSet 的循环内调用（处理大量数据时）。

范例 3-1　拆分中英文数字

原始数据（见图 3-1）：

	A	B	C	D
1	Alpha and 12			
2	aa123bbb中文45			
3	Only Alpha			
4	123			
5	只有中文			
6	中文 and Alpha			
7	中文95			
8				

图 3-1

完整的程序代码：

单元格自定义函数.xls

```
Public Sub 拆分中文英文数字(ByVal poRange As Range)
    Dim i, j, lnASCII, lnSkipPart, lnTargetCount As Integer
    Dim lcList, lcOneChar, lcaPart(3), lcaTargets() As String
    Dim loCell As Range

    For Each loCell In poRange.Cells
        lcList = Trim(loCell.Text)
        lnTargetCount = 0
        For i = 1 To UBound(lcaPart)
            lcaPart(i) = ""
        Next

        For i = 1 To Len(lcList)
            lcOneChar = Mid(lcList, i, 1)
            lnASCII = Asc(lcOneChar)

            Select Case lnASCII
                Case 48 To 57
```


```
                lcaPart(1) = lcaPart(1) + lcOneChar
                lnSkipPart = 1
            Case 0 To 256
                lcaPart(2) = lcaPart(2) + lcOneChar
                lnSkipPart = 2
            Case Else
                lcaPart(3) = lcaPart(3) + lcOneChar
                lnSkipPart = 3
        End Select

        For j = 1 To UBound(lcaPart)
            If j <> lnSkipPart Then
                If Len(lcaPart(j)) <> 0 Then
                    lnTargetCount = lnTargetCount + 1
                    ReDim Preserve lcaTargets(lnTargetCount)
                    lcaTargets(lnTargetCount) = lcaPart(j)
                    lcaPart(j) = ""
                End If
            End If
        Next
    Next

    For j = 1 To UBound(lcaPart)
        If Len(lcaPart(j)) <> 0 Then
            lnTargetCount = lnTargetCount + 1
            ReDim Preserve lcaTargets(lnTargetCount)
            lcaTargets(lnTargetCount) = lcaPart(j)
        End If
    Next

    For i = 1 To lnTargetCount
        loCell.Offset(0, i).Value = lcaTargets(i)
    Next
Next
End Sub
```

在其他程序调用：

单元格自定义函数范例_拆分中文英文数字.xls

```
Public Sub RunMacro()
    Run "'" + ActiveWorkbook.Path + "\单元格自定义函数.xls'!拆分中文英文数字", ActiveSheet.UsedRange
    Workbooks("单元格自定义函数.xls").Close
End Sub
```

这样一来，位于“单元格自定义函数.xls”文件内的“拆分中文英文数字”宏，就可以被同目录下的其他程序重复使用了。如果你要把“单元格自定义函数.xls”放到别的目录，把上述的ActiveWorkbook.Path改成该目录即可。

“其他程序调用”的程序代码在之后的范例中就不列出了，因为除了宏名称不同之外，其他都完全一样。

执行结果（见图 3-2）：

	A	B	C	D	E	F	G
1	Alpha and 12	Alpha and	12				
2	aa123bbb中文45	aa	123	bbb	中文	45	
3	Only Alpha	Only Alpha					
4	123	123					
5	只有中文	只有中文					
6	中文 and Alpha	中文	and Alpha				
7	中文95	中文	95				

图 3-2

范例 3-2　拆分年月日

原始数据：有些是不正确的日期（见图 3-3）。

	A	B	C
1	2004/10/13		
2	2004/13/10		
3	10/13/2004		
4	13/10/2004		
5	10/13		
6	13/10		
7	2004-10-13		
8	2004-13-10		
9	10-13-2004		
10	13-10-2004		
11	10-13		
12	13-10		
13			

图 3-3

完整程序代码：

单元格自定义函数.xls

```
Public Sub 拆分年月日(ByVal poRange As Range)
    Dim ldDate As Date
    Dim loCell As Range

    For Each loCell In poRange.Cells
        If IsDate(loCell.Text) Then
            ldDate = CDate(loCell.Text)
            With loCell
                ' 先清除内容，以免其格式还是先前的格式
                .Offset(0, 1).Clear
                .Offset(0, 2).Clear
                .Offset(0, 3).Clear

                .Offset(0, 1).Value = Year(ldDate)
                .Offset(0, 2).Value = Month(ldDate)
                .Offset(0, 3).Value = Day(ldDate)
            End With
        End If
    Next
End Sub
```

执行结果（见图 3-4）：

	A	B	C	D	E
1	2004/10/13	2004	10	13	
2	2004/13/10				
3	10/13/2004	2004	10	13	
4	13/10/2004	2004	10	13	
5	10/13	2005	10	13	
6	13/10	2005	10	13	
7	2004-10-13	2004	10	13	
8	2004-13-10				
9	10-13-2004	2004	10	13	
10	13-10-2004	2004	10	13	
11	10-13	2005	10	13	
12	13-10	2005	10	13	
13					

图 3-4

范例 3-3　字符串反转

单元格自定义函数.xls

```
Public Sub 字符串反转(ByVal poRange As Range)
    Dim i As Integer
    Dim lcReverseResult As String
    Dim loCell As Range

    For Each loCell In poRange.Cells
        lcReverseResult = ""
        For i = Len(loCell.Text) To 1 Step -1
            lcReverseResult = lcReverseResult + Mid(loCell, i, 1)
        Next
        loCell.Value = lcReverseResult
    Next
End Sub
```

范例 3-4　闰年判断

完整程序代码（直接用该年的 2/29 是否为正确日期做判断即

可，不必 100、400 地除来除去）：

单元格自定义函数.xls

```
Public Sub 闰年判断(ByVal poRange As Range)
    Dim ldDate As Date
    Dim lcText As String
    Dim loCell As Range

    For Each loCell In poRange.Cells
        lcText = Trim(loCell.Text)
        If IsDate(lcText) Then
            ldDate = CDate(lcText)
            If IsLeapYear(Year(ldDate)) Then
                loCell.Interior.ColorIndex = 3
            End If
        End If
    Next
End Sub

Public Function IsLeapYear(ByVal pnYear As Integer) As
Boolean
    IsLeapYear = IsDate(Trim(Str(pnYear)) + "/02/29")
End Function
```

范例 3-5　各种数制间的互相转换

单元格自定义函数.xls

```
Public Sub 二进制转成十进制(ByVal poRange As Range)
    Dim lcValue As String
    Dim i, t, d, c As Integer

    Dim loCell As Range
```

```
    For Each loCell In poRange.Cells
        lcValue = loCell.Text
        If IsNumeric(lcValue) Then
            t = Len(lcValue)
            d = 0
            For i = 1 To t
                c = Val(Mid(lcValue, i, 1))
                d = d + c * 2 ^ (t - i)
            Next
            loCell.Value = d
        End If
    Next
End Sub

Public Sub 二进制转成十六进制(ByVal poRange As Range)
    Dim loCell As Range

    Call 二进制转成十进制(poRange)
    For Each loCell In poRange.Cells
        If IsNumeric(loCell.Text) Then
            loCell.Value = Hex(loCell.Value)
        End If
    Next
End Sub

Public Function AScan(poArray As Variant, ByVal puSearchFor)
As Integer
    If IsArray(poArray) Then
        Dim i As Integer
        For i = 0 To UBound(poArray) - 1
            If poArray(i) = puSearchFor Then
                AScan = i
                Exit Function
            End If
        Next
    End If
    AScan = -1
```


```
End Function

Public Function padl(ByVal pcValue As String, ByVal pnFillLen
As Integer, ByVal pcFillChar As String) As String
    Dim lnLen As Integer
    pcValue = Trim(pcValue)
    lnLen = Len(pcValue)
    If lnLen < pnFillLen Then
        Dim i As Integer
        For i = lnLen + 1 To pnFillLen
            pcValue = pcFillChar + pcValue
        Next
    End If
    padl = pcValue
End Function

Public Sub 十进制转成二进制(ByVal poRange As Range)
    Dim lcValue, lcHex, lcBinary, b As String
    Dim t, i, c As Integer
    Dim loCell As Range

    For Each loCell In poRange.Cells
        If IsNumeric(loCell.Text) Then
            loCell.Value = Hex(loCell.Value)

            lcValue = loCell.Text

            lcHex = "0123456789ABCDEF"
            lcBinary =
"0000000100100011010001010110011111000100110101011110011010111101111"

            b = ""
            t = Len(lcValue)
            For i = 1 To t
                c = InStr(lcHex, UCase(Mid(lcValue, i, 1)))
                If c <> 0 Then
                    b = b + Mid(lcBinary, (c - 1) * 4 + 1, 4)
```

```
                Else
                    Exit For
                End If
            Next
            loCell.Value = "'" + b
        End If
    Next
End Sub

Public Sub 十进制转成十六进制(ByVal poRange As Range)
    Dim loCell As Range

    For Each loCell In poRange.Cells
        If IsNumeric(loCell.Text) Then
            loCell.Value = "'" + Hex(loCell.Value)
        End If
    Next
End Sub

Public Sub 十六进制转成二进制(ByVal poRange As Range)
    Dim i, t As Integer
    Dim b, lcHex, lcBinary, lcValue As String
    Dim loCell As Range

    For Each loCell In poRange.Cells
        lcValue = loCell.Text

        lcHex = "0123456789ABCDEF"
        lcBinary =
"00000001001000110100010101100111100010011010101111001101111011111"

        b = ""
        t = Len(lcValue)
        For i = 1 To t
            c = InStr(lcHex, UCase(Mid(lcValue, i, 1)))
            If c <> 0 Then
```


```
                b = b + Mid(lcBinary, (c - 1) * 4 + 1, 4)
            Else
                Exit For
            End If
        Next
        loCell.Value = "'" + b
    Next
End Sub

Public Sub 十六进制转成十进制(ByVal poRange As Range)
    Dim lcaHex(16) As String
    Dim lcValue, lcOneChar As String
    Dim i, t, d, c As Integer
    Dim loCell As Range

    For i = 0 To 9
        lcaHex(i + 1) = Trim(Str(i))
    Next
    For i = 1 To 6
        lcaHex(i + 9) = Chr(64 + i)
    Next

    For Each loCell In poRange.Cells
        lcValue = loCell.Text

        d = 0
        t = Len(lcValue)
        For i = 1 To t
            lcOneChar = UCase(Mid(lcValue, i, 1))
            If AScan(lcaHex, lcOneChar) = -1 Then
                Exit For
            End If

            If Asc(lcOneChar) - 64 >= 1 And Asc(lcOneChar) -
64 <= 6 Then   ' A-F
                c = Asc(lcOneChar) - 55
```

```
            Else
                c = Val(lcOneChar)
            End If
            d = d + c * 16 ^ (t - i)
        Next
        loCell.Value = d
    Next
End Sub
```

心得栏

CHAPTER

4

综合技巧

本章中，综合前面所介绍的元数据概念、SQL 技巧，来解决一些学习 Excel 宏程序设计时会遇到的问题。此外，也会介绍一些有趣的小程序。

范例 4-1　在空白单元格填充同列上一个单元格的数值

原始数据如图 4-1 所示，依照列标将空白部分填成上一个值后，如图 4-2 所示。

	A	B	C	D
1	1	5	A	
2				
3				
4				
5				
6		6		
7			B	
8	2			
9				
10			C	
11				
12				
13		7		
14				
15				
16				
17	3			
18			D	
19				
20		8		
21				
22				
23				
24	4			
25				
26				
27			E	
28				
29				

图 4-1

	A	B	C	D
1	1	5	A	
2	1	5	A	
3	1	5	A	
4	1	5	A	
5	1	5	A	
6	1	6	A	
7	1	6	B	
8	2	6	B	
9	2	6	B	
10	2	6	C	
11	2	6	C	
12	2	6	C	
13	2	7	C	
14	2	7	C	
15	2	7	C	
16	2	7	C	
17	3	7	C	
18	3	7	D	
19	3	7	D	
20	3	8	D	
21	3		D	
22	3		D	
23	3		D	
24	4		D	
25			D	
26			D	
27			E	
28				
29				
30				

图 4-2

完整程序代码：

综合技巧范例_填充上一个值.xls

```
Public Sub RunMacro()
    ' 先读取 meta data
    Run ("'" + ActiveWorkbook.Path + "\找出 meta_data.xls'!
Module1.ActiveWorkSheet_To_Table_No_Empty_Cell")
    Workbooks("找出 meta_data.xls").Close

    Set loADODBConnection = CreateObject("ADODB.Connection")
```

```
    Set loADODBRecordset = CreateObject("ADODB.Recordset")
    loADODBConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                           "DBQ=c:\cell_meta_data.xls;" & _
                           "ReadOnly=False"

    ' 抓出所有列标
    loADODBRecordset.Open "select 列标, max(行号) as 最后行号
from [cell_meta_data$] group by 列标", loADODBConnection

    Dim loRange, loCell As Range
    Dim luLastValue As Variant
    Dim lnTheLastRowNumber As Integer

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Columns(loADODBRecordset.
Fields("列标").Value)
        lnTheLastRowNumber = loADODBRecordset.Fields("最后行号").Value

        luLastValue = Chr(0)
        For Each loCell In loRange.Cells
            If Len(loCell.Text) = 0 Then
                loCell.Value = luLastValue
            End If

            If loCell.Row = lnTheLastRowNumber Then
                Exit For
            Else
                luLastValue = loCell.Value
            End If
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close
End Sub
```

请注意，此处的“找出 meta data”变成了：

```
ActiveWorkSheet_To_Table_No_Empty_Cell
```

而非原先的：

```
ActiveWorkSheet_To_Table
```

ActiveWorkSheet_To_Table_No_Empty_Cell 加入了一个选项：不读取空白单元格的元数据。

以此例而言，不读取空白单元格的元数据可以让第 1、2、3 列的数据分别填到 4、8、E 就停止，而不会通通对齐到 E 所在的行号。如果你想对齐的话，就使用原先的 ActiveWorkSheet_To_Table 那段程序代码，因为它读取的是 UsedRange，而图 4-1 的 UsedRange，是会把空白单元格的元数据也算进去的！这样一来，所有列标的最大行号就都会是 E 所在的那一列，而非最后一个不为空白的行号了。

图 4-1 是这个问题的抽象化描述，这种填充上一个值的问题其数据通常是源于图 4-3 这种一对多的缩进报表。

	A	B	C
1	姓名	请假时间	请假类型
2	Chris	2004-3-1	病假
3		2004-3-19	病假
4		2004-4-9	休假
5		2004-5-3	事假
6	Irene	2004-4-11	事假
7		2004-11-5	产假
8	Rita	2004-2-5	休假
9		2004-3-1	休假
10		2004-4-10	休假
11			
12			

图 4-3

但这种报表的格式是给人看的，其实它并不适合计算机处理，把它填充上一个值才方便程序处理。

当然，有时我们就是要给人看，所以，有填充的需求，反过来就有清空的需求。

范例 4-2　清空同列相同数值（仅保留首次出现）

综合技巧范例_清空上一个值.xls

```
Public Sub RunMacro()
    ' 先读取 meta data
    Run ("'" + ActiveWorkbook.Path + "\找出 meta_data.xls'!
Module1.ActiveWorkSheet_To_Table_No_Empty_Cell")
    Workbooks("找出 meta_data.xls").Close

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                           "DBQ=c:\cell_meta_data.xls;" & _
                           "ReadOnly=False"

    ' 抓出所有列标
    loADODBRecordset.Open "select 列标, max(行号) as 最后行号
from [cell_meta_data$] group by 列标", loADODBConnection

    Dim loRange, loCell As Range
    Dim luLastValue As String
    Dim lnTheLastRowNumber As Integer

    luLastValue = ""

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Columns(loADODBRecordset.
Fields("列标").Value)
```

```
        lnTheLastRowNumber = loADODBRecordset.Fields("最后行号").Value

        For Each loCell In loRange.Cells
            If Len(loCell.Text) <> 0 Then
                If luLastValue = "" Then
                    luLastValue = loCell.Text
                Else
                    If loCell.Value = luLastValue Then
                        loCell.ClearContents
                    Else
                        luLastValue = loCell.Text
                    End If
                End If
            End If

            If loCell.Row = lnTheLastRowNumber Then
                Exit For
            End If
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close
End Sub
```

由于单元格的 Text 属性是只读的，所以在填充（上一例）时，只能设置 Value 属性，但这一例是用来比较跟上一个值是否一样，所以可以用 Text 属性。

范例 4-3　清空同列相同数值后合并单元格

如果想再进一步，清空上一个值之后，合并单元格（见图 4-4）。

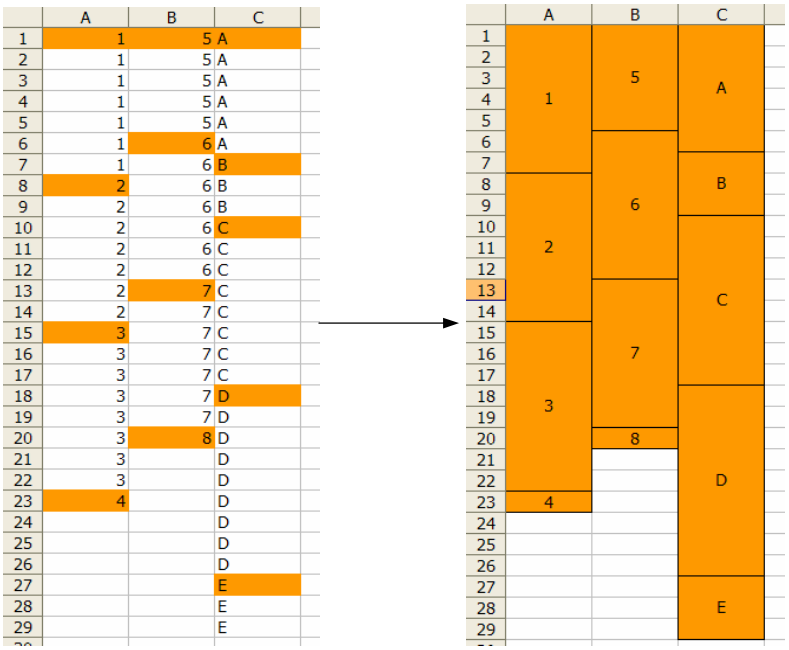


图 4-4

那就修改一下程序代码：

综合技巧范例_清空上一个值后合并单元格.xls

```
Public Sub RunMacro()
    ' 先读取 meta data
    Run ("'" + ActiveWorkbook.Path + "\找出 meta_data.xls'!
Module1.ActiveWorkSheet_To_Table_No_Empty_Cell")
    Workbooks("找出 meta_data.xls").Close

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open "Driver={Microsoft Excel Driver
(*.xls)}; " & _
                           "DBQ=c:\cell_meta_data.xls;" & _
                           "ReadOnly=False"

    ' 抓出所有列标
```

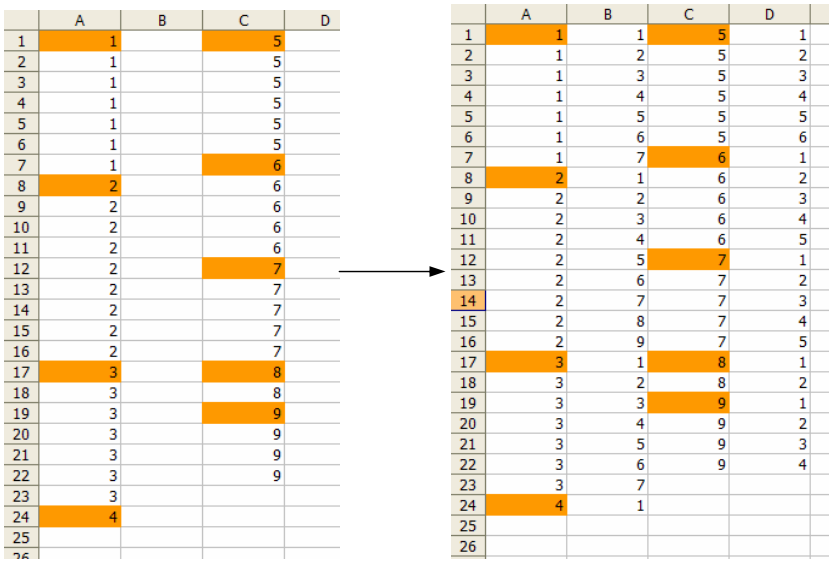
```
    loADODBRecordset.Open "select 列标, max(行号) as 最后行号
from [cell_meta_data$] group by 列标", loADODBConnection

    Dim loRange As Range
    Dim loCell As Range
    Dim loCellToMergeBegin As Range
    Dim loCellToMergeEnd As Range
    Dim loCellToMergeEndLast As Range
    Dim luLastValue As String
    Dim lnTheLastRowNumber As Integer

    luLastValue = ""

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Columns(loADODBRecordset.
Fields("列标").Value)
        lnTheLastRowNumber = loADODBRecordset.Fields("最后行号").Value

        For Each loCell In loRange.Cells
            If Len(loCell.Text) <> 0 Then
              If luLastValue = "" Then
                  luLastValue = loCell.Text
                  Set loCellToMergeBegin = loCell
              Else
                  If loCell.Value = luLastValue Then
                      loCell.ClearContents
                  Else
                      If loCell.Row > 1 Then
                          Set loCellToMergeEnd = loCell. Offset (-1, 0)
                      Else
                          Set loCellToMergeEnd = loCellTo MergeBegin
                        End If

                        With Range(loCellToMergeBegin,
loCellToMergeEnd)
                        .HorizontalAlignment = xlCenter
```


```
                    .VerticalAlignment = xlCenter
                    .MergeCells = True
                    .Borders(xlEdgeLeft).LineStyle=xlContinuous
                    .Borders(xlEdgeTop).LineStyle=xlContinuous
                    .Borders(xlEdgeRight).LineStyle=xlContinuous
                    .Borders(xlEdgeBottom).LineStyle=xlContinuous
                End With

                luLastValue = loCell.Text
                Set loCellToMergeBegin = loCell
            End If
        End If
    End If

    Set loCellToMergeEndLast = loCell

    If loCell.Row = lnTheLastRowNumber Then
        Exit For
    End If
Next

loADODBRecordset.MoveNext
Wend

With Range(loCellToMergeBegin, loCellToMergeEndLast)
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlCenter
    .MergeCells = True
    .Borders(xlEdgeLeft).LineStyle = xlContinuous
    .Borders(xlEdgeTop).LineStyle = xlContinuous
    .Borders(xlEdgeRight).LineStyle = xlContinuous
    .Borders(xlEdgeBottom).LineStyle = xlContinuous
End With

loADODBConnection.Close
End Sub
```

在清空的过程中，时时记住有着相同值的第一个单元格跟最后一个单元格。遇到一个新的单元格时，就把之前记住的第一个单元格跟最后一个单元格的地址所构成的区域，设置其合并单元格以及边框属性即可。还有，别忘了考虑到“最后一个区域”，因为，最后一个区域没有下一个“新的”单元格以资辨别它是“最后”，所以用 loCellToMergeEndLast 来记住每一个单元格，再跟“最近一次的”第一个单元格所构成的区域就是这“最后一个区域”了。

范例 4-4　为单元格中相同值添加序号

类似的应用还有这种为上一个相同值加上序号的问题（见图 4-5）：



	A	B	C	D
1	1		5	
2	1		5	
3	1		5	
4	1		5	
5	1		5	
6	1		5	
7	1		6	
8	2		6	
9	2		6	
10	2		6	
11	2		6	
12	2		7	
13	2		7	
14	2		7	
15	2		7	
16	2		7	
17	3		8	
18	3		8	
19	3		9	
20	3		9	
21	3		9	
22	3		9	
23	3			
24	4			
25				

	A	B	C	D
1	1	1	5	1
2	1	2	5	2
3	1	3	5	3
4	1	4	5	4
5	1	5	5	5
6	1	6	5	6
7	1	7	6	1
8	2	1	6	2
9	2	2	6	3
10	2	3	6	4
11	2	4	6	5
12	2	5	7	1
13	2	6	7	2
14	2	7	7	3
15	2	8	7	4
16	2	9	7	5
17	3	1	8	1
18	3	2	8	2
19	3	3	9	1
20	3	4	9	2
21	3	5	9	3
22	3	6	9	4
23	3	7		
24	4	1		
25				
26				

图 4-5

完整程序代码：

综合技巧范例_为相同值加上序号.xls

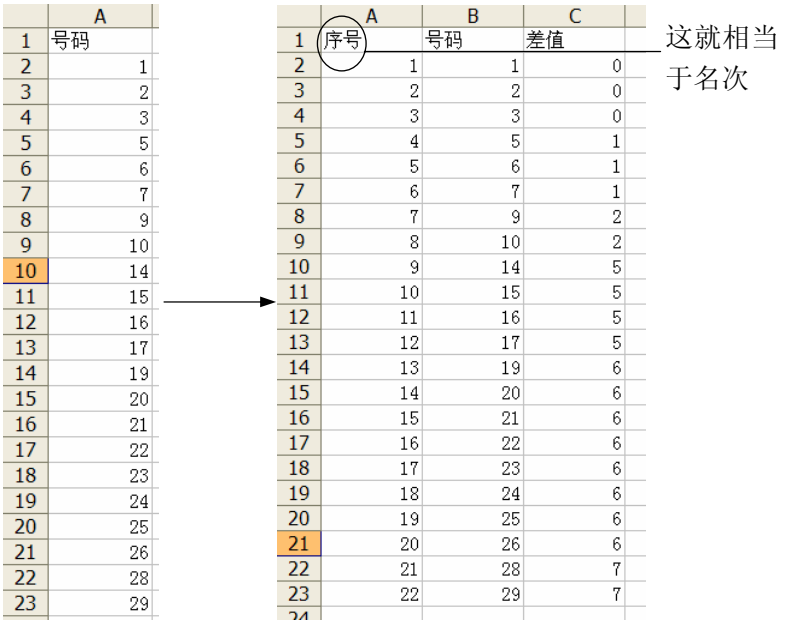
```
Public Sub RunMacro()
    ' 先读取 meta data
    Run ("'" + ActiveWorkbook.Path + "\找出 meta_data.xls'!
Module1.ActiveWorkSheet_To_Table_No_Empty_Cell")
    Workbooks("找出 meta_data.xls").Close

    Set loADODBConnection = CreateObject("ADODB.Connection")
    Set loADODBRecordset = CreateObject("ADODB.Recordset")

    loADODBConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                           "DBQ=c:\cell_meta_data.xls;" & _
                           "ReadOnly=False"

    ' 抓出所有列标
    loADODBRecordset.Open "select 列标, max(行号) as 最后行号
from [cell_meta_data$] group by 列标", loADODBConnection

    Dim loRange, loCell As Range
    Dim luLastValue As String
    Dim lnTheLastRowNumber As Integer
    Dim lnSerialNumber As Integer

    luLastValue = ""
    lnSerialNumber = 1

    While Not loADODBRecordset.EOF
        Set loRange = ActiveSheet.Columns(loADODBRecordset.
Fields("列标").Value)
        lnTheLastRowNumber = loADODBRecordset.Fields("最后行
号").Value

        For Each loCell In loRange.Cells
            If Len(loCell.Text) <> 0 Then
```

```
                If luLastValue = "" Then
                    luLastValue = loCell.Text
                    loCell.Offset(0, 1).Value = lnSerialNumber
                Else
                    If loCell.Value = luLastValue Then
                        lnSerialNumber = lnSerialNumber + 1
                        loCell.Offset(0, 1).Value=lnSerialNumber
                    Else
                        luLastValue = loCell.Text
                        lnSerialNumber = 1
                        loCell.Offset(0, 1).Value=lnSerialNumber
                    End If
                End If
            End If

            If loCell.Row = lnTheLastRowNumber Then
                Exit For
            End If
        Next

        loADODBRecordset.MoveNext
    Wend

    loADODBConnection.Close
End Sub
```

范例 4-5　寻找缺号

还记得前面提到过两次的范例——排名次吗？

排名次是个典型的 SQL 应用，不只如此，它还可以衍生出“寻找缺号”的功能，请看这个原始数据（见图 4-6）：



	A
1	号码
2	1
3	2
4	3
5	5
6	6
7	7
8	9
9	10
10	14
11	15
12	16
13	17
14	19
15	20
16	21
17	22
18	23
19	24
20	25
21	26
22	28
23	29

	A	B	C
1	序号	号码	差值
2	1	1	0
3	2	2	0
4	3	3	0
5	4	5	1
6	5	6	1
7	6	7	1
8	7	9	2
9	8	10	2
10	9	14	5
11	10	15	5
12	11	16	5
13	12	17	5
14	13	19	6
15	14	20	6
16	15	21	6
17	16	22	6
18	17	23	6
19	18	24	6
20	19	25	6
21	20	26	6
22	21	28	7
23	22	29	7

图 4-6

经过排名次的 self-join 后，图中的“序号”就类似于“名次”的意思。取出序号与原号码的“差值”，如果不是 0，就表示有缺号（但无法得知缺几号，号码也必须是数字类型，而不能是文本）。

看着差值，如何从这差值的规律性得知从哪个号码开始缺号呢？以差值为 group，然后取最小的号码（min(号码)）就可以了。

综合技巧范例_寻找缺号.xls

```
Public Sub RunMacro()
    For Each loWindow In Application.Windows
        If LCase(loWindow.Caption) = "result.xls" Then
            loWindow.Close (False)
        End If
    Next
```

```
    If Dir("c:\result.xls") <> "" Then
        Kill "c:\result.xls"
    End If

    ActiveWorkbook.Save

    Dim lcCurrentWorkBookName As String
    lcCurrentWorkBookName = ActiveWorkbook.Name

    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")

    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=" + ActiveWorkbook.FullName + ";" & _
                      "ReadOnly=True"

    loRecordset.Open _
    "select min(号码) as 是谁缺号 " & _
    "into [Excel 8.0;Database=c:\result.xls].[sheet1] " & _
    "from ( " & _
    "select * " & _
    "from ( " & _
    "select (select count(*) from [score$])-(select count(*)
from [score$] where 号码 > a.号码) as 序号, 号码, " & _
    "号码-((select count(*) from [score$])-(select count(*)
from [score$] where 号码 > a.号码)) as 差值 " & _
    "from [score$] a) " & _
    "order by 1) " & _
    "where 差值 > 0 group by 差值", _
    loConnection

    loConnection.Close

    Workbooks.Open ("c:\result.xls")

    Dim loCell As Range
    Dim loTargetRange As Range
```

```
    Dim loTargetCell As Range
    For Each loCell In
Workbooks("result.xls").Sheets(1).UsedRange.Cells
        If loCell.Value <> "号码" Then
            Set loTargetRange =
Workbooks(lcCurrentWorkBookName).Sheets(1).UsedRange
            Set loTargetCell =
loTargetRange.Find(loCell.Value)
            If Not loTargetCell Is Nothing Then
                loTargetCell.Interior.ColorIndex = 3
            End If
        End If
    Next

    Workbooks("result.xls").Close
End Sub
```

由于这个范例还使用到了 Range 对象的 Find 方法，而且是在原工作表标示出缺号所在位置，所以我把它列入综合技巧——尽管它用的是 SQL 的方式。

范例 4-6　导入文本文件

Excel 的导入功能已经可以从一定程度上解决结构化文本文件与 Excel 之间的信息交换问题，但仍有一些限制，最常见的就是数据过滤的问题，比如要忽略空白行、某字段必须符合某些条件才能导入等。

像如图 4-7 这样一个文本文件，以逗号分隔，如果要导入 Excel，需忽略第二行的分隔线。

图 4-7

综合技巧范例_导入文本文件.xls

```
Public Sub RunMacro()
    Cells.Clear

    ActiveWorkbook.Save

    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")

    ' 读取 c:\myTxtDatabase 目录下的 asc,csv,tab,txt 等文件，
    ' 每个文件看作一个 Table
    loConnection.Open _
    "Driver={Microsoft Text Driver (*.txt; *.csv)};" & _
    "Dbq=" + ActiveWorkbook.Path & "\;" & _
    "Extensions=asc,csv,tab,txt;"

    loRecordset.Open "select * from friends.txt where id <> 0",
loConnection

    Dim f As Integer
```


```
    For f = 0 To loRecordset.Fields.Count - 1
        Cells(1, f + 1).Value = loRecordset.Fields(f).Name
    Next

    Range("A2").CopyFromRecordset loRecordset

    loConnection.Close
End Sub
```

因为 id 字段是数字类型，所以分隔线那一行所在的 id 字段值会等于 0（因为没有值），如此，条件“`where id <> 0`”就可以过滤掉分隔线。

另外，还有一些文本文件是没有标题的，这时，除了可以利用 Excel 内嵌的数据分析功能，以手工的方式导入文本文件之外，也可以利用 Open、Close、Line Input、Dir 和 Split 等命令及函数，自己做个实用的文本文件导入程序，特别是你有大量的文本文件要导入时，这种自动化的方式会更好，以下几个范例展示了几种常见的导入方式。

综合技巧范例_导入文本文件_2.xls

```
Public Sub Reset()
    Dim s As Integer
    Application.DisplayAlerts = False
    For s = Sheets.Count To 2 Step -1
        Sheets(s).Delete
    Next
    Application.DisplayAlerts = True

    Sheets(1).Cells.Delete Shift:=xlUp
    Sheets(1).Cells(1, 1).Select

    Dim loShape As Shape
```

```
    Dim i As Integer
    i = 0
    For Each loShape In ActiveSheet.Shapes
        i = i + 1
        With loShape
            .Left = 300
            .Width = 200
            .Top = (i - 1) * (.Height + 5) + 20
        End With
    Next
End Sub

Public Function GetDir() As String
    Dim lcDir As String
    lcDir = Trim(ThisWorkbook.Path)
    lcDir = IIf(Right(lcDir, 1) <> "\", lcDir & "\", lcDir)
    GetDir = lcDir
End Function

Public Sub 一般状况_导入文本文件()
    Call Reset

    Dim lcOneLine As String
    Dim r As Integer

    Cells.Clear
    Open GetDir & "某一文本文件.txt" For Input As #1
    r = 0
    Do While Not EOF(1)
        Line Input #1, lcOneLine
        r = r + 1
        Cells(r, 1).Value = lcOneLine
    Loop
    Close #1
    ActiveSheet.UsedRange.Columns.AutoFit
    Range("A1").Activate
```


```
End Sub

Public Sub 两行当作一个_导入文本文件()
    Call Reset

    Dim lcOneLine As String
    Dim i, r As Integer

    Cells.Clear
    Open GetDir & "某一文本文件.txt" For Input As #1
    i = 0
    r = 0
    Do While Not EOF(1)
        Line Input #1, lcOneLine
        i = i + 1
        If i = 1 Then
            r = r + 1
            Cells(r, 1).Value = lcOneLine
        Else
            Cells(r, 1).Value = Cells(r, 1).Value & lcOneLine
            i = 0
        End If
    Loop
    Close #1
    ActiveSheet.UsedRange.Columns.AutoFit
    Range("A1").Activate
End Sub

Public Sub 多个文件_固定分隔符_导入文本文件()
    Call Reset

    Dim lcFileName As String
    Dim lcOneLine As String
    Dim s, r, c As Integer
    Dim lc_data As Variant
```

```
    s = 0
    lcFileName = Dir(GetDir & "固定分隔符_?.txt")
    Do While Len(lcFileName) > 0
        s = s + 1
        If Sheets.Count < s Then
            Sheets.Add after:=Sheets(Sheets.Count)
            Sheets(Sheets.Count).Name = lcFileName
        End If

        Sheets(Sheets.Count).Activate
        Cells.Clear

        Open GetDir & lcFileName For Input As #1
        r = 0
        Do While Not EOF(1)
            Line Input #1, lcOneLine
            r = r + 1
            c = 0
            For Each lc_data In Split(lcOneLine, ",")
                c = c + 1
                Cells(r, c).Value = lc_data
            Next
        Loop
        Close #1
        ActiveSheet.UsedRange.Columns.AutoFit
        Range("A1").Activate
        lcFileName = Dir()
    Loop

    Sheets(1).Activate
End Sub

Public Sub 自定义日期格式_导入文本文件()
    Call Reset

    Dim lcOneLine As String
```


```
    Dim r As Integer

    Cells.Clear
    Open GetDir & "日期文件.txt" For Input As #1
    r = 0
    Do While Not EOF(1)
        Line Input #1, lcOneLine
        r = r + 1
        '假设日期格式是“月日年”且年份为公元格式，例如：12/31/2002
        Cells(r, 1).Value = Mid(lcOneLine, 7, 4) & "/" &
Mid(lcOneLine, 1, 2) & "/" & Mid(lcOneLine, 4, 2)
    Loop
    Close #1
    ActiveSheet.UsedRange.Columns.AutoFit
    Range("A1").Activate
End Sub
```

范例 4-7　把 Excel 工作表导出为文本文件

如果要把 Excel 的工作表导出成为文本文件，可以用内置的 SaveAs 方法，不过该方法只支持以 Tab 字符及逗号为分隔符的文本文件格式（逗号分隔的话，即为 CSV 文件），如果你想自定义格式，例如以“|”为分隔符，或以每个单元格而非整行转成一行文本，则可以用找出元数据所用的命令：

```
    Open "c:\导出文件名称.asc" For Output As #1
      .
      .
      .
    Write #1, ...
    Close #1
```

把 Write #1 放在循环中：

综合技巧范例_转成自定义文本文件.xls

```
Public Sub RunMacro()
    Dim loCell As Range
    Dim lnLastRow As Integer
    Dim lcOneLine As String

    lnLastRow = ActiveSheet.UsedRange.Cells(1).Row
    lcOneLine = ""

    Open ThisWorkbook.Path + "\导出文件名称.asc" For Output As #1

    For Each loCell In ActiveSheet.UsedRange.Cells
        If loCell.Row <> lnLastRow Then
            Write #1, Mid(lcOneLine, 2)
            lcOneLine = ""
        End If
        lcOneLine = lcOneLine & "|" & loCell.Text
        lnLastRow = loCell.Row
    Next
    Close #1

    Shell "notepad.exe " & ThisWorkbook.Path + "\导出文件名
称.asc", vbNormalFocus
End Sub
```

如果不想在输出结果的每行前后加上双引号，就用 Print 而不要用 Write。

这个简单的自定义转换程序还可以很弹性地加上自定义的条件，例如不转出地址（地址在第 3 列，也就是 C 列）：

```
    Public Sub RunMacro()
    Dim loCell As Range
```


```
    Dim lnLastRow As Integer
    Dim lcOneLine As String

    lnLastRow = ActiveSheet.UsedRange.Cells(1).Row
    lcOneLine = ""

    Open ThisWorkbook.Path + "\导出文件名称.asc" For Output As #1

    For Each loCell In ActiveSheet.UsedRange.Cells
        If loCell.Row <> lnLastRow Then
            Write #1, Mid(lcOneLine, 2)
            lcOneLine = ""
        End If
        If loCell.Column <> 3 Then
            lcOneLine = lcOneLine & "|" & loCell.Text
        End If
        lnLastRow = loCell.Row
    Next

    Close #1

    Shell "notepad.exe " & ThisWorkbook.Path + "\导出文件名
称.asc", vbNormalFocus
End Sub
```

此外，如果你的 Excel 数据本身就长得跟 Table 一样，也可以把它当数据表来看，然后直接以 ODBC 的方式将其 select 出来，然后再写入某个文本文件：

综合技巧范例_转成自定义文本文件_使用 ODBC 方式.xls

```
Public Sub RunMacro()
    ' 先把目前的 Excel 文件 select 出来
    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")
```

```
    loConnection.Open "Driver={Microsoft Excel Driver
(*.xls)}; " & _
                      "DBQ=" + ActiveWorkbook.FullName + ";" & _
                      "ReadOnly=True"

    loRecordset.Open "select * from [sheet1$]", loConnection

    ' 先建立一个文本文件，并写入一个记录以设置其数据类型
    Open ActiveWorkbook.Path & "\综合技巧范例_转成自定义文本文件
_使用 ODBC 方式.txt" For Output As #1
    Print #1, "id , name, addr"
    Print #1, "0, '', ''"
    Close #1

    ' 再以 ODBC 方式写入文本文件
    Set loConnection2 = CreateObject("ADODB.Connection")
    loConnection2.Open _
    "Driver={Microsoft Text Driver (*.txt; *.csv)};" & _
    "Dbq=" + ActiveWorkbook.Path & "\;" & _
    "Extensions=asc,csv,tab,txt;"

    Set loCommand = CreateObject("ADODB.Command")
    loCommand.ActiveConnection = loConnection2

    Dim f As Integer
    Dim fv As String
    While Not loRecordset.EOF
        loCommand.CommandText = "insert into 综合技巧范例_转成
自定义文本文件_使用 ODBC 方式.txt values"
        fv = ""
        For f = 0 To loRecordset.Fields.Count - 1
            If IsNumeric(loRecordset.Fields(f).Value) Then
                fv = fv & "," & loRecordset.Fields(f).Value
            Else
                fv = fv & ",'" & loRecordset.Fields(f).Value & "'"
            End If
```


```
        Next

        loCommand.CommandText = loCommand.CommandText & "(" &
Mid(fv, 2) & ")"
        loCommand.Execute

        loRecordset.MoveNext
    Wend

    loConnection.Close

    ' 再从该文本文件写入另一文本文件，并删掉第一个记录
    If Dir(ActiveWorkbook.Path + "\综合技巧范例_转成自定义文本文
件_使用 ODBC 方式_最后结果.txt") <> "" Then
        Kill (ActiveWorkbook.Path + "\综合技巧范例_转成自定义文
本文件_使用 ODBC 方式_最后结果.txt")
    End If
    loRecordset.Open "select * into 综合技巧范例_转成自定义文本
文件_使用 ODBC 方式_最后结果.txt from " & _"综合技巧范例_转成自定义
文本文件_使用 ODBC 方式.txt where id <> 0", loConnection2
    loConnection2.Close

    Kill (ActiveWorkbook.Path + "\综合技巧范例_转成自定义文本文
件_使用 ODBC 方式.txt")
    Kill (ActiveWorkbook.Path + "\schema.ini")

    Shell "notepad.exe " & ThisWorkbook.Path + "\综合技巧范
例_转成自定义文本文件_使用 ODBC 方式_最后结果.txt", vbNormalFocus
End Sub
```

与 Excel 相同的是，当文本文件作为一个表时，也可以从该文件 select 出数据来，然后再直接把结果存入另一个文本文件。

范例 4-8　产生随机数

产生随机数的程序很容易让人联想到彩票，没错，接下来要谈的就是彩票。

假设我们的彩票是在 1~42 之间取不重复的 6 个数字，如果需要，还可以再加以排序，最后显示出来。

综合技巧范例_产生随机数.xls

```
Public Sub RunMacro()
    Randomize

    Dim lnRandomValue, t, i, s, nd, x, y, lnTemp As Integer
    Dim lnaLotto(6) As Integer
    Dim llDuplicate As Boolean

    For i = 1 To 6
        Cells(1, i).Value = "号码" + Trim(Str(i))
    Next

    ' 产生 30 组
    For t = 1 To 30
        ' 初始化
        For i = 1 To 6
            lnaLotto(i) = 0
        Next

        ' 产生不重复的 6 个号码
        nd = 0
        While nd < 6
            ' 产生 1~42 之间的随机数
            lnRandomValue = Int((42 * Rnd) + 1)

            ' 检查是否重复?
            llDuplicate = False
```


```
            For s = 1 To 6
                If lnaLotto(s) = lnRandomValue Then
                    llDuplicate = True
                    Exit For
                End If
            Next

            ' 不重复才计入
            If Not llDuplicate Then
                nd = nd + 1
                lnaLotto(nd) = lnRandomValue
            End If
        Wend

        ' 每组产生完后，都去做排序的操作
        For x = 1 To 5
            For y = x + 1 To 6
                If lnaLotto(x) > lnaLotto(y) Then
                    lnTemp = lnaLotto(x)
                    lnaLotto(x) = lnaLotto(y)
                    lnaLotto(y) = lnTemp
                End If
            Next
        Next

        ' 显示结果
        For i = 1 To 6
            Cells(t + 1, i).Value = "'" &
IIf(Len(Trim(Str(lnaLotto(i)))) = 1, "0" &
Trim(Str(lnaLotto(i))), Trim(Str(lnaLotto(i))))
        Next
    Next
End Sub
```

执行结果（见图 4-8）：

	A	B	C	D	E	F
1	号码1	号码2	号码3	号码4	号码5	号码6
2	3	9	11	15	20	33
3	6	19	21	34	36	38
4	2	8	16	23	28	33
5	2	5	8	18	23	31
6	5	6	7	13	14	28
7	16	21	23	28	37	41
8	4	20	25	26	28	36
9	4	11	20	30	33	35
10	1	5	8	22	26	41
11	3	7	14	15	36	38
12	1	3	18	21	22	34
13	2	5	22	24	29	37
14	9	12	23	28	33	41
15	3	12	13	19	26	33
16	5	10	15	17	32	35
17	13	14	20	21	30	33
18	12	13	25	28	31	42
19	3	21	30	32	39	40
20	20	22	27	30	31	34
21	2	29	34	35	39	41
22	4	9	10	11	18	19
23	2	5	12	17	40	41
24	1	2	7	16	29	42
25	8	14	17	25	33	42
26	6	9	10	16	28	31
27	10	12	17	20	30	34
28	2	19	22	25	36	38
29	5	13	16	21	35	42
30	1	8	19	28	31	33
31	8	9	18	19	30	31
32						

图 4-8

随机数产生程序的作法都要先调用一个所谓的“随机数初始化”函数，Excel VBA 也不例外，它用的是 Randomize 命令。产生某数到某数之间随机数的方法，也不外是 Int((最大要产生的数-最小要产生的数+1) * Rnd) +最小要产生的数)，所以要产生 1~42 之间的随机数就是 Int((42-1+1) *Rnd+1)= Int(42*Rnd+1)。

Excel 的 Array 没有内置的搜索及排序功能，本范例分别使用最简单的循环搜索法及冒泡排序法来达到此需求。

本范例还有另一种做法，还记得之前介绍过的“排列程序”吗？那个程序会产生 N 组彩票号码，你可以产生 1~N 之间的随机数（要产生几次随便你），比如产生出一个 M，M 介于 1~N 之间，

然后直接从那 N 组中取第 M 个号码即可，这样就不必一个数字一个数字地产生了，当然也不必检查重复以及进行排序。

范例 4-9　彩票信息统计

假设你利用了本书的“上网抓某一网页所有的 HTML Table”技巧，从彩票开奖网站上读取了所有的中奖号码，经过简单的整理后，变成如图 4-9 所示的表格。

	A	B	C	D	E	F	G
1	期号	号码1	号码2	号码3	号码4	号码5	号码6
2	93001	6	16	21	22	26	41
3	93002	9	10	16	34	39	40
4	93003	2	9	28	34	40	41
5	93004	6	14	23	25	33	35
6	93005	1	2	11	27	28	38
7	93006	3	4	10	20	26	29
8	93007	4	20	25	29	35	41
9	93008	5	6	8	29	30	37
10	93009	3	4	12	18	19	26
11	93010	14	21	22	23	26	35
12	93011	1	12	18	23	30	35
13	93012	1	4	20	26	32	35
14	93013	7	14	24	28	33	37
15	93014	3	15	19	36	39	40
16	93015	3	15	16	17	23	24
17	93016	2	6	12	19	23	42
18	93017	4	16	17	19	23	27
19	93018	1	12	16	19	20	27
20	93019	9	12	18	19	27	36
21	93020	3	13	14	20	24	32
22	93021	16	26	30	34	37	38
23	93022	2	18	23	25	31	41
24	93023	9	19	23	26	28	32
25	93024	3	21	24	25	35	40
26	93025	2	6	21	23	28	39
27	93026	3	14	22	29	32	36
28	93027	1	2	19	25	38	39
29	93028	2	11	13	19	32	39
30	93029	6	13	21	28	35	39
31	93030	5	6	10	20	34	37
32							

图 4-9

现在，你想统计一些数字：

- 哪个号码最常出现？

- ⊙ 哪个号码最不常出现？
- ⊙ “39”出现过几次？
- ⊙ 有几次是有连号的？
- ⊙ 各个号码各自有几期没出现过了？
- ⊙ 哪个号码最久没出现？
- ⊙ “08”最近一次出现是在第几期？

在统计之前，先定义一下上述某些统计量的意义：所谓“最常出现”指的是“出现最多次”而且“出现的期号最连续”，那何谓“最连续”呢？就是出现期号所构成的数列拥有最低的标准差，也就是最密集。所以，假设 22 跟 33 在 30 期的开出号码中都同样出现过 12 次，但 22 分别出现在“1、2、3、4、5、6、7、8、10、11、12”（纯举例）等 12 期，33 则出现在“1、2、3、4、5、6、7、8、10、11、29”等 12 期，那我们就定义 22 比 33 更常出现。

同理，“最不常出现”就是出现次数最少，出现时，又“期号离得最远”，也就是标准差最大。

“连号”指的是一期的号码中有 1 个以上的连号，例如“10,13,14,22,33,41”、“10,11,21,23,24,40”或“10,11,12,23,35,37”等。

“某个号码几期没出现过”指的是距离最新一期而言。例如目前总共开出 30 期，“10”上一次出现是在第 12 期，那就是 30-12=18 期没出现过了。而最久没出现的就是这个差值最大者。

“某个号码最近一次出现的期号”就是该号码出现期号最大者。例如“08”分别出现在“10、12、20、22”等期号，则最近

一次出现就是在第 22 期。

OK，讲完了统计量的定义，从图 4-9 中，你是否能利用 SQL 技巧很轻易地算出这些统计量呢？

我不知道你是否可以（如果可以请来信告知），至少我是不行。

咦，你不是说 SQL 很强？对，SQL 很强，但数据的结构也一样重要，如果结构定义得不够好，那 SQL 也无用武之地。

以图 4-9 为例，要统计出上述那些统计量，最好把数据结构定义成图 4-10 所示的这样。

	A	B
1	期号	开奖号码
2	93001	6
3	93001	16
4	93001	21
5	93001	22
6	93001	26
7	93001	41
8	93002	9
9	93002	10
10	93002	16
11	93002	34
12	93002	39
13	93002	40
14	93003	2
15	93003	9
16	93003	28
17	93003	34
18	93003	40
19	93003	41
20	93004	6
21	93004	14
22	93004	23
23	93004	25
24	93004	33
25	93004	35

图 4-10

有没有觉得有点“反数据透视表”的味道？没错，就是如此。Excel 已经有数据透视表的功能，但反过来的功能，从图 4-9 到图 4-10，可以用 SQL 做到：

综合技巧范例_彩票统计.xls

```
select 期号, 号码
from (
select 期号, 号码1 as 号码 from [sheet1$] union
select 期号, 号码2 as 号码 from [sheet1$] union
select 期号, 号码3 as 号码 from [sheet1$] union
select 期号, 号码4 as 号码 from [sheet1$] union
select 期号, 号码5 as 号码 from [sheet1$] union
select 期号, 号码6 as 号码 from [sheet1$])
order by 期号, 号码
```

事实上，数据透视表是给人看的，它代表统计出来的结果，但反数据透视表这样的原始样式，像图 4-10 这种，才是适合给计算机看的（至少适合给 SQL 看）。

范例 4-10　统计最常出现或最少出现的彩票号码

接下来就轻松了，统计最常出现的号码的程序代码如下。

综合技巧范例_彩票统计_最常出现的号码.xls

```
Public Sub RunMacro()
    ' 先把数据结构定好
    For Each loWindow In Application.Windows
        If LCase(loWindow.Caption) = "result.xls" Then
            loWindow.Close (False)
        End If
    Next

    If Dir("c:\result.xls") <> "" Then
        Kill "c:\result.xls"
    End If

    ActiveWorkbook.Save
```


```
    Set loConnection = CreateObject("ADODB.Connection")
    Set loRecordset = CreateObject("ADODB.Recordset")

    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=" + ActiveWorkbook.FullName + ";" & _
                      "ReadOnly=True"

    loRecordset.Open _
    "select 期号, 号码 " & _
    "into [Excel 8.0;Database=c:\result.xls].[sheet1] " & _
    "from ( " & _
    "select 期号, 号码1 as 号码 from [sheet1$] union " & _
    "select 期号, 号码2 as 号码 from [sheet1$] union " & _
    "select 期号, 号码3 as 号码 from [sheet1$] union " & _
    "select 期号, 号码4 as 号码 from [sheet1$] union " & _
    "select 期号, 号码5 as 号码 from [sheet1$] union " & _
    "select 期号, 号码6 as 号码 from [sheet1$]) " & _
    "order by 期号, 号码", _
    loConnection

    loConnection.Close

    Workbooks.Open ("c:\result.xls")
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Name = "Tahoma"
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Size = 8
    ActiveWorkbook.ActiveSheet.Cells.EntireRow.AutoFit

ActiveWorkbook.ActiveSheet.Cells.EntireColumn.AutoFit
    ActiveWorkbook.Close (True)

    ' 接下来，计算统计量时就轻松多了
    loConnection.Open "Driver={Microsoft Excel Driver (*.xls)}; " & _
                      "DBQ=c:\result.xls;" & _
                      "ReadOnly=True"

    ' Excel SQL 没有内建的 StdDev 聚合函数，我们可以自己依照标准差
的公式来模拟它
```

```
    loRecordset.Open _
    "select top 1 号码 as 最常出现的号码 " & _
    "into [Excel 8.0;Database=c:\result.xls].[最常出现的号码] " & _
    "from ( " & _
    "select 号码, 出现次数, 出现的密集程度 " & _
    "from ( " & _
    "select 号码, count(*) as 出现次数, " & _
    "sqr(sum((abs(期号-(select avg(期号) from [sheet1$] where
号码 = a.号码)))^2)/(select iif(count(*)=1, 1, count(*)-1) from
[sheet1$] where 号码 = a.号码)) as 出现的密集程度 " & _
    "from [sheet1$] a " & _
    "group by 号码) " & _
    "order by 出现次数 desc, 出现的密集程度)", _
    loConnection

    loConnection.Close

    ' 看最后的结果
    Workbooks.Open ("c:\result.xls")
    Application.DisplayAlerts = False
    ActiveWorkbook.Sheets(1).Delete
    Application.DisplayAlerts = True
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Name = "Tahoma"
    ActiveWorkbook.ActiveSheet.UsedRange.Font.Size = 8
    ActiveWorkbook.ActiveSheet.Cells.EntireRow.AutoFit
    ActiveWorkbook.ActiveSheet.Cells.EntireColumn.AutoFit
End Sub
```

统计最不常出现的号码，反过来就行，程序代码如下：

综合技巧范例_彩票统计_最不常出现的号码.xls

```
select top 1 号码 as 最不常出现的号码
from (
select 号码, 出现次数, 出现的密集程度
from (
select 号码, count(*) as 出现次数,
```

```
sqr(sum((abs(期号-(select avg(期号) from [sheet1$] where
号码 = a.号码)))^2)/(select iif(count(*)=1, 1, count(*)-1)
from [sheet1$] where 号码 = a.号码)) as 出现的密集程度
from [sheet1$] a
group by 号码)
order by 出现次数, 出现的密集程度 desc)
```

还是取第 1 个记录，但因为顺序倒过来了，变成是“出现最少而且出现的期号最离散”，会被排列到最前面，所以会取到最不常出现的号码。

请注意这两个范例都没有考虑到“同分同等”的状况，但我之前谈过（在计算排名那个范例），所以，为了不把这个范例搞得更复杂，这个问题就交给你举一反 N 了（所谓“同分同等”，就是“出现次数）跟“出现的密集程度”都一样的号码们，所以，先找出第 1 个的“出现次数”和“出现的密集程度”后，再在外面加一层 select 即可）。

另外，由于 Excel 的 SQL 没有内置 StdDev（标准差）这个函数，所以我们自己写个公式来实现。就是样本中的每一个数减去样本平均数的平方和，再除以（样本数-1），这是方差，方差再取平方根就是标准差了。

“某号码出现过几次”，这太简单了，以号码为分组，统计 count(*)即可。在“统计最常出现的号码”时就已经列过程序代码，因此不再重复列出。

范例 4-11　列出彩票连号的期号和连号号码

“在所有开出号码的期号中，有几期是有连号的”，回答这个

问题之前还是先想想看，在图 4-10 这样的数据结构中，是否可以很快让你想出这个问题的 SQL 要如何写？如果不能很快想出，或许不是你的 SQL 不够厉害，而是数据结构本身蕴含的语义不够丰富。

连号的概念，牵涉到 6 个号码中每一个号码跟下一个号码的“差值”，差值为 1 就表示有连号，而 6 个号码一共会产生 5 个差值。

所以，不妨把数据结构定义成图 4-11 所示这样。

	A	B	C
1	期号	第几个	号码
2	93001	1	6
3	93001	2	16
4	93001	3	21
5	93001	4	22
6	93001	5	26
7	93001	6	41
8	93002	1	9
9	93002	2	10
10	93002	3	16
11	93002	4	34
12	93002	5	39
13	93002	6	40
14			

图 4-11

而这可以由下列的程序代码产生：

综合技巧范例_彩票统计_列出连号的期号及其号码.xls

```
select 期号, 第几个, 号码
from (
select 期号, 1 as 第几个, 号码1 as 号码 from [sheet1$] union
select 期号, 2 as 第几个, 号码2 as 号码 from [sheet1$] union
select 期号, 3 as 第几个, 号码3 as 号码 from [sheet1$] union
select 期号, 4 as 第几个, 号码4 as 号码 from [sheet1$] union
select 期号, 5 as 第几个, 号码5 as 号码 from [sheet1$] union
select 期号, 6 as 第几个, 号码6 as 号码 from [sheet1$])
order by 期号, 第几个, 号码
```

跟之前那个很像，只是多加了一个常数式的字段表达式。

有了这样的数据结构，我们的目标是要取出图 4-12 说示的差值：

	A	B	C	D
1	期号	上一个	下一个	差值
2	93001	6	16	10
3	93001	16	21	5
4	93001	21	22	1
5	93001	22	26	4
6	93001	26	41	15
7	93002	9	10	1
8	93002	10	16	6
9	93002	16	34	18
10	93002	34	39	5
11	93002	39	40	1
12				

图 4-12

取出差值后，答案已经呼之欲出，就是找出“差值=1”的期号。而在一期中，“差值=1”的可能不止 1 个，例如“11,12,22,33,34,41”就有两个“差值=1”的状况，但没关系，我们的目的只是要找出有连号的期号，并没有规定要几组连号才算，所以只要找出不重复的期号即可，还记得吗？利用 SQL 的 distinct 关键词：

```
    select distinct 期号 from 图4-12的结果 where 差值 = 1 order by 期号
```

那要如何产生图 4-12 呢？利用 self-join：

综合技巧范例_彩票统计_列出连号的期号及其号码.xls

```
select a.期号, a.号码 as 上一号, b.号码 as 下一号, b.号码-a.号码 as 差值
from [sheet1$] a, [sheet1$] b
where b.期号 = a.期号
and   b.第几个 = a.第几个+1
```

结合上述两者，最后就是：

综合技巧范例_彩票统计_列出连号的期号及其号码.xls

```
select distinct 期号
from (
select a.期号, a.号码 as 上一号, b.号码 as 下一号, b.号码-a.号码
as 差值
from [sheet1$] a, [sheet1$] b
where b.期号 = a.期号
and   b.第几个 = a.第几个+1)
where 差值 = 1
order by 期号
```

“各个号码各自有几期没出现过了”，也很容易，先计算出总共有几期：

综合技巧范例_彩票统计_列出某号几期没出现过了.xls

```
select count(*) from (select distinct 期号 from [sheet1$])
```

然后以此结果减去各个号码最近一次（也就是期号最大者）的出现期号：

综合技巧范例_彩票统计_列出某号几期没出现过了.xls

```
(select count(*) from (select distinct 期号 from [sheet1$]))
- max(val(mid(期号,5,2))) as 几期没出现过
```

因为期号是“93001”的文本格式，所以要再转成数值。

结合起来就是：

综合技巧范例_彩票统计_列出某号几期没出现过了.xls

```
select 号码, (select count(*) from (select distinct 期号 from
[sheet1$])) - max(val(mid(期号,5,2))) as 几期没出现过
from [sheet1$]
group by 号码
```

接着是“哪个号码最久没出现”，可直接以上述“几期没出现过”的结果为基础进行信息再处理：

综合技巧范例_彩票统计_列出最久没出现过的号码.xls

```
select 号码
from [某号几期没出现$]
where 几期没出现过 =
(select max(几期没出现过) from [某号几期没出现$])
```

最后，“某号码最近一次的出现期号”就不赘述，因为已经含在上面这几个例子中了。

范例 4-12　复制目前选取区域至其他工作表

当你在 Excel 中以 Ctrl 键连续选取了不相邻的多个区域后，是无法以手动的方式进行剪贴的。

看看录制宏是怎么做的：

一边按住 Ctrl 键不放，一边利用鼠标选取多个范围，选愈多愈好（选它个 100 个 Range），录制完成之后去检查宏程序代码，你会发现 Excel 是以 Union 配合 Range 录制的，而且它会很聪明地自动“断 Range”，也就是 Range 的字符串太长时它会分成数个 Range，一起放在 Union 里：

```
Union( _
Range("D26,B23,A16,O30,N20:N21,K20:K21,K13,K7,J2:J3,M10,N10
,N4,O5,O11,P17,P30,O37:O38,N35:N36,M32:M33,J28:J29,H21:H23,
G18:G19,D7:D11,F11:G15,D17:D21,F22:F23,C21:C22,B10,B4,B14,J
19,J9"), _
Range("I21,I30:I31,E31:E32,B29,F28,L26:L27,L15,M7,N16,N27,L
35:L36,P21:P22,P9,H5:H6,F4:F5,H14:H15,H26:H27,K26,J36,G36,C
```

```
35:C36,D35:D36")).Select
```

但我们自己在写这种程序时有个小麻烦，就是要自己做这种“断 Range”的操作，而且程序还会写得又臭又长。

我们可以利用 Union 传回两个 Range 之后的结果，然后再不断（在循环中）传回给自己。

例如，你已经知道了众多不相邻、不规则的单元格（或区域）地址，并已经将它们都存在一个数组之中，假设数组内容如下：

```
MyArray(0) = "A1"
MyArray(1) = "A12"
MyArray(2) = "B21"
MyArray(3) = "C4"
MyArray(4) = "E10:F15"
```

那大可不必大费周章地自己“组合 Range 字符串”或“断 Range”（如果 Range 太长时），只需：

```
Dim MyRange As Range
Set MyRange = Range(MyArray(0))
For i = 1 to UBound(MyArray)-1
Set MyRange = Union(MyRange, Range(MyArray(i)))
Next
```

即可。

以此为出发点，我们可以把不相邻的选取区域中的所有单元格内容从工作表 A 复制（或转移）到工作表 B，并且在工作表 B 也做相同的选取：

综合技巧范例_复制当前选取区域至其他工作表.xls

```
Public Sub CopyMultipleSelectAreas()
    Union(Range("A5:A7"), Range("C3:C5"),
Range("F4:F8")).Select

    ' 最后要 mark 这个多重区域
    Dim loMarkSelected As Range

    Dim loCell As Range
    For Each loCell In Selection.Cells
        Sheets(2).Range(loCell.Address).Value = loCell.Value
        If loMarkSelected Is Nothing Then
            ' 如果是第一次……
            Set loMarkSelected = Sheets(2).Range(loCell.Address)
        Else
            ' 之后就一直累加下去
            Set loMarkSelected = Union(loMarkSelected,
Sheets(2).Range(loCell.Address))
        End If
    Next

    ' 要 mark 这个多重区域位于 Sheets(2)
    Sheets(2).Activate
    loMarkSelected.Select
End Sub
```

这种方法只确保了程序的简洁性，却无法保证它的效率。如果你觉得执行速度太慢（事实上，我不知道会变得多慢），可在宏执行前后加上 ScreenUpdating 之类的设置。

范例 4-13　Word 与 Excel 之间的数据交换

Excel 的表格直接贴到 Word 没问题，通常数据都不会跑掉，

但如果你把一个单元格中有多行文本的 Word 表格贴到 Excel 时，却会因为多行的关系，让贴上去之后的结果在 Excel 里变成多行（见图 4-13）。

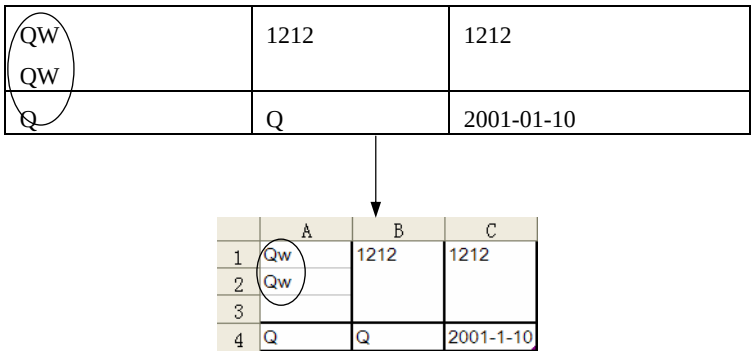


图 4-13

这乃是 Word 与 Excel 对表格内单元格有多行时的解释不同所致，要彻底从 Word 把数据不失真地转到 Excel，可利用这一段宏程序逐个 Cell 地抛数据：

综合技巧范例_Word 与 Excel 间的数据交换.doc

```
Public Sub WordTableToExcel()
    Dim loCell As Cell
    Dim t As Integer
    Dim p As Integer

    Dim loExcel As Variant
    Dim loExcelApp As Excel.Application
    Dim loExcelRange As Excel.Range

    Set loExcel = CreateObject("Excel.Sheet")
    Set loExcelApp = loExcel.Application
    loExcelApp.Visible = True
    loExcelApp.Workbooks.Add
```


```
    For t = 1 To ThisDocument.Tables.Count
        For Each loCell In ThisDocument.Tables(t).Range.Cells
            Set loExcelRange = loExcelApp.ActiveWorkbook.
ActiveSheet.Cells(loCell.RowIndex, loCell.ColumnIndex)
            For p = 1 To loCell.Range.Paragraphs.Count
                loExcelRange.Value = loExcelRange.Value +
IIf(p = 1, "", Chr(10)) + loCell.Range.Paragraphs(p).Range
            Next
        Next
    Next


loExcelApp.ActiveWorkbook.ActiveSheet.Cells.EntireColumn.AutoFit

loExcelApp.ActiveWorkbook.ActiveSheet.Cells.EntireRow.AutoFit
    loExcelApp.ActiveWorkbook.ActiveSheet.Cells(1, 1).Select
End Sub
```

这是从 Word 执行的宏，要改成从 Excel 执行也不是什么难事，程序代码甚至更简洁呢。

综合技巧范例_Word 与 Excel 间的数据交换.xls

```
Public Sub WordTableToExcel()
    Cells.Clear
    Dim loCell As Cell
    Dim t As Integer
    Dim p As Integer

    Dim loWord As Variant
    Dim loWordApp As Word.Application
    Set loWord = CreateObject("Word.Document")
    Set loWordApp = loWord.Application
    loWordApp.Documents.Open (ThisWorkbook.Path + "\综合技巧
范例_Word 与 Excel 间的数据交换.doc")

    For t = 1 To loWordApp.Documents(1).Tables.Count
```

```
        For Each loCell In loWordApp.Documents(1).Tables(t).
Range.Cells
            Set loExcelRange = ActiveWorkbook.ActiveSheet.Cells
(loCell.RowIndex, loCell.ColumnIndex)
            For p = 1 To loCell.Range.Paragraphs.Count
                loExcelRange.Value = loExcelRange.Value +
IIf(p = 1, "", Chr(10)) + loCell.Range.Paragraphs(p).Range
            Next
        Next
    Next

    ActiveWorkbook.ActiveSheet.Cells.EntireColumn.AutoFit
    ActiveWorkbook.ActiveSheet.Cells.EntireRow.AutoFit
    ActiveWorkbook.ActiveSheet.Cells(1, 1).Select

    loWordApp.Quit
End Sub
```

范例 4-14　关闭应用程序

其实这个范例不只可以关闭 Excel，只要不是那些“不可关闭的”DLL 库、特殊的 EXE、或 Windows 服务等，知道了某个应用程序的执行文件名称，也都可以关闭它们，以下范例是打开 5 个记事本程序后再逐个关闭：

综合技巧范例_关闭应用程序.xls

```
Public Sub CloseApp()
    Dim loProcess As Object
    Dim c As Integer

    For c = 1 To 5
        Shell "notepad.exe"
```

```
    Next

    Cells.Clear
    c = 0
    For Each loProcess In
GetObject("winmgmts://").InstancesOf("Win32_Process")
        If LCase(loProcess.Name) = "notepad.exe" Then
            If lo_process.Terminate() = 0 Then
                c = c + 1
                Cells(c, 1) = "第" & c & " 个" & loProcess.Name & " 关闭了"
            End If
        End If
    Next
    Range("A:A").Columns.AutoFit
    Range("A1").Activate
End Sub
```

注意，此技巧仅适用于操作系统为 Windows 2000 以后的版本（是 Windows 的版本，不是 Excel 版本）。

用这个方法来关闭应用程序不会出现任何询问窗口，例如“文件已经修改，是否存盘？”之类的画面。另外，也比用 FindWindow API 这种通过由窗口的标题来关闭应用程序的方法更准确，因为有些应用程序的窗口标题是会一直变动的，例如打开不同的文件，但执行文件名却是固定的。而且此方法还可关闭已经死掉但仍在内存中的应用程序，例如死掉的 IE（IE 的执行文件名是 IEXPLORE.EXE）。

以本范例而言，如果你想只关闭“一个”notepad.exe 的话，就在关闭第一个之后输入 Exit For 命令。至于这样会关掉哪一个就不得而知了，可能是最先执行的那一个，也可能是刚刚被存取的那一个（也就是最近一次被鼠标点到的那一个）。

除了关闭应用程序，winmgmts 另一个常见的应用是读取出网卡的地址，如果你想让自己开发的应用程序只能在某些计算机上执行，可以在应用程序执行前检查那些计算机的网卡是否在原先的列表中。当然，如果网卡因为坏掉而更换，该列表也必须随之更新。不过一般说来这并不常发生，所以此技巧不失为保护应用程序的一个简易方法。

综合技巧范例_读取网卡地址.xls

```
Public Sub GetNetworkAdapterAddress()
    Dim loAdapter As Object
    Dim c As Integer

    Cells.Clear
    c = 0
    For Each loAdapter In GetObject("winmgmts:").ExecQuery
("select * from Win32_NetworkAdapter")
        If Not IsNull(loAdapter.MACAddress) Then
            c = c + 1
            Cells(c, 1) = loAdapter.Name
            Cells(c, 2) = loAdapter.MACAddress
        End If
    Next
    Range("A:B").Columns.AutoFit
    Range("A1").Activate
End Sub
```

范例 4-15　Excel 工作表行列互换

把一块区域 Excel 内置的数据转置是将行列互换，用角度来看的话，是把原数据先顺时针转 270 度、再进行上下翻转。以下范例可以分别将数据做如下的转置：顺时针 90 度、顺时针 180 度、顺

时针 270 度、左右翻转、上下翻转，可以补 Excel 内置功能之不足。

综合技巧范例_数据转置.xls

```
Public Sub DataTransposeFor(ByVal pcDirection As String)
    Sheets.Add
    ActiveSheet.Name = pcDirection

    Dim loTargetSheet As Worksheet
    Set loTargetSheet = Sheets(pcDirection)

    Dim loCell As Range
    Dim i, r, c, j As Integer

    Sheets("Sheet1").Activate
    ActiveSheet.Range("B4:D7").Select

    Select Case pcDirection
        Case "顺时针 90 度"
            For i = Selection.Rows.Count To 1 Step -1
                r = 0
                c = c + 1
                For Each loCell In Selection.Rows(i).Cells
                    r = r + 1
                    loTargetSheet.Cells(r, c).Value = loCell.Value
                Next
            Next
        Case "顺时针 180 度"
            For i = Selection.Columns.Count To 1 Step -1
                r = 0
                c = c + 1
                For j = Selection.Columns(i).Cells.Count To 1 Step -1
                    r = r + 1
                    loTargetSheet.Cells(r, c).Value = Selection.
Columns(i).Cells(j).Value
                Next
            Next
```

```
        Case "顺时针 270 度"
            For i = 1 To Selection.Rows.Count
                r = 0
                c = c + 1
                For j = Selection.Rows(i).Cells.Count To 1 Step -1
                    r = r + 1
                    loTargetSheet.Cells(r, c).Value = Selection.
Rows(i).Cells(j).Value
                Next
            Next
        Case "左右翻转"
            For i = 1 To Selection.Rows.Count
                c = 0
                r = r + 1
                For j = Selection.Rows(i).Cells.Count To 1 Step -1
                    c = c + 1
                    loTargetSheet.Cells(r, c).Value = Selection.
Rows(i).Cells(j).Value
                Next
            Next
        Case "上下翻转"
            For i = Selection.Rows.Count To 1 Step -1
                c = 0
                r = r + 1
                For j = 1 To Selection.Rows(i).Cells.Count
                    c = c + 1
                    loTargetSheet.Cells(r, c).Value = Selection.
Rows(i).Cells(j).Value
                Next
            Next
    End Select
End Sub

Public Sub DataTranspose()
    Application.DisplayAlerts = False
    Dim loSheet As Worksheet
```


```
    For Each loSheet In ActiveWorkbook.Sheets
        If loSheet.Name <> "Sheet1" Then
            loSheet.Delete
        End If
    Next
    Application.DisplayAlerts = True

    Call DataTransposeFor("顺时针 90 度")
    Call DataTransposeFor("顺时针 180 度")
    Call DataTransposeFor("顺时针 270 度")
    Call DataTransposeFor("左右翻转")
    Call DataTransposeFor("上下翻转")
End Sub
```

范例 4-16　工作表排序

以下范例会依照工作表的名称由左至右将工作表排序，也就是改变其 Index 属性，但因为 Sheet 对象的 Index 属性是只读的，所以必须通过 Move 方法。

综合技巧范例_工作表排序.xls

```
Public Sub WorkSheetSort()
    Dim i, j, m, n As Integer
    Dim lcaSheetName() As String
    Dim lcTempSheetName As String

    n = 0
    Dim loSheet As Worksheet
    For Each loSheet In ActiveWorkbook.Sheets
        n = n + 1
        ReDim Preserve lcaSheetName(n)
        lcaSheetName(n) = loSheet.Name
    Next
```

```
    For i = 1 To Sheets.Count - 1
        For j = i + 1 To Sheets.Count
            If lcaSheetName(i) > lcaSheetName(j) Then
                lcTempSheetName = lcaSheetName(i)
                lcaSheetName(i) = lcaSheetName(j)
                lcaSheetName(j) = lcTempSheetName
            End If
        Next
    Next

    For m = n To 1 Step -1
        Sheets(lcaSheetName(m)).Select
        Sheets(lcaSheetName(m)).Move Sheets(1)
    Next

    Sheets("Sheet1").Activate
End Sub
```

作法是将所有的 Sheet 名称先行保存并排序，然后再逐个移到第 1 个 Sheet 的后面，这样一来，名称内容大的 Sheet 就会一直被“挤到后面”，挤完所有 Sheet 后也就完成排序了。

执行结果（见图 4-14）：

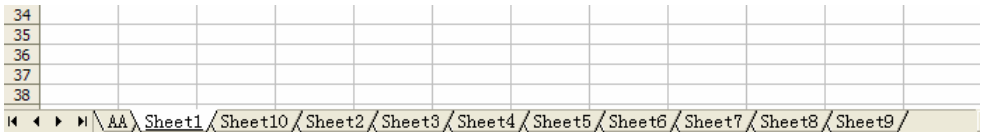


图 4-14

注意名称为“Sheet10”的将会排在“Sheet2”的前面，因为“1”比“2”小。如果在图 4-14 想依照“Sheet+数字”来排序的话，方法是将“Sheet2”改名成“Sheet02”。

范例 4-17　改变单元格及批注的字体

单元格与批注内的文本都可以部分改变其字体（见图 4-15）。

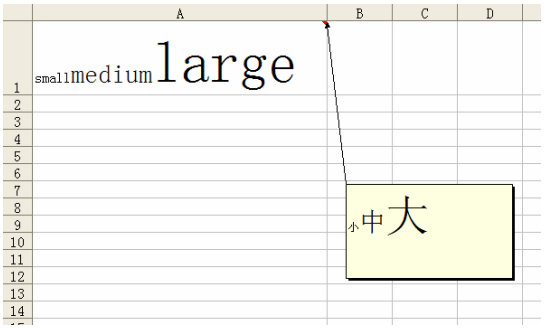


图 4-15

当然也可以改变字体颜色：

综合技巧范例_改变单元格及批注的字体.xls

```
Public Sub Reset()
    Cells.Clear
    Cells(1, 1).Value = "This is a test."
    Cells(2, 1).Value = "中文测试."

    Range("A1").AddComment
    Range("A1").Comment.Visible = True
    Range("A1").Comment.Text Text:="Comment on A1."
    Range("A1").Comment.Shape.Left = 150
    Range("A1").Comment.Shape.Top = 10
    Range("A1").Comment.Shape.Width = 200
    Range("A1").Comment.Shape.Height = 100

    Range("A2").AddComment
    Range("A2").Comment.Visible = True
    Range("A2").Comment.Text Text:="A2 的批注."
    Range("A2").Comment.Shape.Left = 150
    Range("A2").Comment.Shape.Top = 150
```

```
    Range("A2").Comment.Shape.Width = 200
    Range("A2").Comment.Shape.Height = 100

    Dim loComment As Comment
    For Each loComment In ActiveSheet.Comments
        loComment.Shape.AutoShapeType = msoShapeRectangle
    Next

    Cells.EntireColumn.AutoFit
    Cells.EntireRow.AutoFit
End Sub

Public Sub ChangeCellFontPartly()
    With Range("A1").Characters(Start:=6, Length:=2).Font
        .Name = "Microsoft Sans Serif"
        .Size = 20
        .ColorIndex = 3
    End With

    With Range("A2").Characters(Start:=3, Length:=1).Font
        .Name = "宋体"
        .Size = 20
        .ColorIndex = 5
    End With

    Cells.EntireColumn.AutoFit
    Cells.EntireRow.AutoFit
End Sub

Public Sub ChangeCommentFont()
    MsgBox ("改变批注的字体（全部文本）...")
    Range("A1").Comment.Shape.TextFrame.Characters.Font.Size = 24
    Range("A2").Comment.Shape.TextFrame.Characters.Font.Size = 24

    MsgBox ("改变批注的字体（部分文本）...")
    Range("A1").Comment.Shape.TextFrame.Characters(1, 2).Font.Size=12
```


```
    MsgBox ("改变所有批注的字体（全部文本）...")
    Dim loComment As Comment
    For Each loComment In ActiveSheet.Comments
        loComment.Shape.TextFrame.Characters.Font.Size = 12
    Next
End Sub

Public Sub ChangeShape()
    Dim loComment As Comment
    For Each loComment In ActiveSheet.Comments
        loComment.Shape.AutoShapeType =
msoShapeRoundedRectangle
    Next
End Sub
```

此外，批注也可以改变其形状（见图 4-16），不一定都是方方正正的：

```
Range("A1").Comment.Shape.AutoShapeType = soShapeRoundedRectangle
```

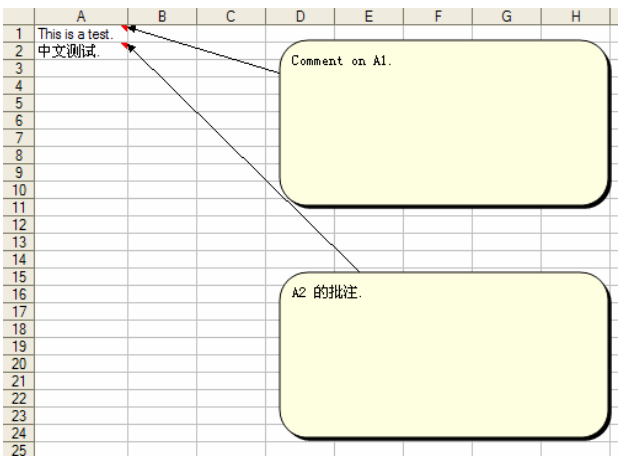


图 4-16

这个技巧只能针对文本，如果单元格内容是数字或日期都不适用。

范例 4-18　重新命名工作表

Excel 单一工作簿内可以允许的工作表数目没有限制，有时工作表太多且名称又混乱，找起来颇不方便，我们可以依自定义的名称重新命名一番（见图 4-17）。

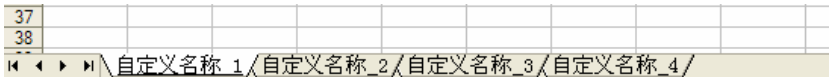


图 4-17

完整程序代码：

综合技巧范例_重新命名工作表.xls

```
Public Sub RenameWorkSheetsRandom()
    Dim loSheet As Worksheet

    Randomize
    For Each loSheet In ThisWorkbook.Worksheets
        loSheet.Name = "~!@#$%^" & Trim(Str(Int(Rnd * 100000) + 1))
    Next

    Sheets(1).Activate
End Sub

Public Sub RenameWorkSheets()
    Dim loSheet As Worksheet

    Call RenameWorkSheetsRandom

    Dim lnSheetSerialNumber As Integer
    lnSheetSerialNumber = 0
    For Each loSheet In ThisWorkbook.Worksheets
        lnSheetSerialNumber = lnSheetSerialNumber + 1
        loSheet.Name = "自定义名称_" &
```


```
Trim(Str(lnSheetSerialNumber))
    Next

    Sheets(1).Activate
End Sub
```

重新命名之前先把名称设置成一个随机数所构成的（字符串）名称，以免跟现有的名称冲突。

范例 4-19　变工作表为独立的工作簿

如果你想“做个了结”，把工作簿内的所有工作表都独立存成一个 xls 文件，也不难办到，就是逐个将工作表复制到一个新建的工作簿。

综合技巧范例_每个工作表独立存成一个文件.xls

```
Public Sub RenameWorkSheetsRandom()
    Dim loSheet As Worksheet

    Randomize
    For Each loSheet In ThisWorkbook.Worksheets
        loSheet.Name = "~!@#$%^" & Trim(Str(Int(Rnd * 100000) + 1))
    Next

    Sheets(1).Activate
End Sub

Public Sub SaveEachSheetAsXLS()
    Dim loSheet As Worksheet
    Call RenameWorkSheetsRandom

    Dim lnSheetSerialNumber As Integer
    lnSheetSerialNumber = 0
```

```
    Dim lcCurrentPath As String
    lcCurrentPath = ActiveWorkbook.Path & "\" &
ActiveWorkbook.Name

    Dim loNewWorkBook As Workbook
    Dim d As Integer

    Application.DisplayAlerts = False
    For Each loSheet In ThisWorkbook.Worksheets
        lnSheetSerialNumber = lnSheetSerialNumber + 1
        loSheet.Name = "自定义名称_" &
Trim(Str(lnSheetSerialNumber))

        Workbooks.Add
        Set loNewWorkBook = Workbooks(Workbooks.Count)
        loSheet.Copy Before:=loNewWorkBook.Sheets(1)
        loNewWorkBook.Sheets(1).Name = loSheet.Name

        ' 删到剩下一个工作表为止，因为每删一个工作表，
        ' 之后的那个工作表就成了 Sheets(2)，
        ' 所以把 Sheets(2) 删 loNewWorkBook.Sheets.Count - 1 次即可
        For d = 1 To loNewWorkBook.Sheets.Count - 1
            loNewWorkBook.Sheets(2).Delete
        Next

        loNewWorkBook.Sheets(1).Name = loSheet.Name
        loNewWorkBook.SaveAs (lcCurrentPath & "_" &
loSheet.Name)
        loNewWorkBook.Close
    Next
    Application.DisplayAlerts = True

    Sheets(1).Activate
End Sub
```


能独立存成一个 xls 文件，当然也就能够存成其他 Excel 内置

的文件类型，例如网页，只要把保存那一行改成这样：

综合技巧范例_每个工作表独立存成一个网页文件.xls

```
loNewWorkBook.PublishObjects.Add(xlSourceSheet, _
lcCurrentPath & "_" & loSheet.Name, loSheet.Name, "", _
xlHtmlStatic, "Book1_" & Str(Int(Rnd * 1000000 + 1)),
"").Publish (True)
```

有一个参数，我们传进了一个随机数产生的字符串，其实可以省略，但这么做的原因是 Excel 把工作表存成网页文件时，会把工作表的内容以 HTML DIV 元素表示，而该随机数字符串即为该 DIV 区块的“对象变量名称”。我们用这种产生变量名称的方式并不见得可以绝对保证它的“惟一性”，但应该极少机会重复就是了。

范例 4-20　每隔几行插入一行（或每隔几列插入一列）

图 4-18 是很规律的隔行插入效果。

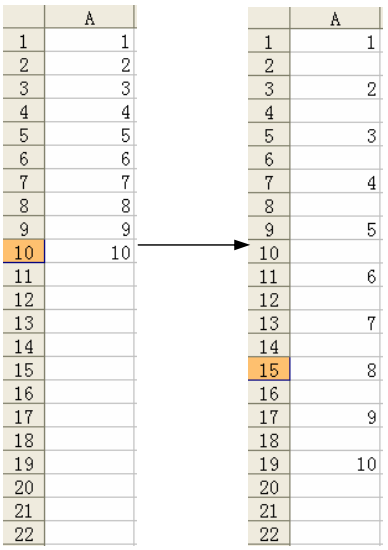


图 4-18

这类插入行（或列）的问题不外乎是插入一行后原有的行会向下移，所以注意是要在“哪一行”进行插入的动作，本范例是很规律的每 N 行插入一行，所以这个“哪一行”是用以下方式表示的：

```
Rows(i * (pnEveryNRow + 1)).Insert Shift:=xlDown
```

插入列与此类似，完整程序代码如下：

综合技巧范例_插入行或列.xls

```
Public Const gnCount As Integer = 10

' 每隔 N 行插入 1 行
Public Sub InsertRow(ByVal pnEveryNRow As Integer)
    Dim i As Integer
    For i = 1 To gnCount
        Rows(i * (pnEveryNRow + 1)).Insert Shift:=xlDown
    Next
End Sub

' 每隔 N 列插入 1 列
Public Sub InsertColumn(ByVal pnEveryNColumn As Integer)
    Dim i As Integer
    For i = 1 To gnCount
        Columns(i * (pnEveryNColumn + 1)).Insert Shift:=xlRight
    Next
End Sub

' 原始数据：行
Public Sub ResetRow()
    Cells.Clear
    For i = 1 To gnCount
        Cells(i, 1).Value = i
    Next
```


```
End Sub

' 原始数据：列
Public Sub ResetColumn()
    Cells.Clear
    For i = 1 To gnCount
        Cells(1, i).Value = i
    Next
End Sub

' 每隔 1 行插入 1 行
Public Sub InsertOneRowEveryOneRow()
    Call ResetRow
    MsgBox ("请按任意键继续...")
    Call InsertRow(1)
End Sub

' 每隔 1 列插入 1 列
Public Sub InsertOneColumnEveryOneColumn()
    Call ResetColumn
    MsgBox ("请按任意键继续...")
    Call InsertColumn(1)
End Sub

Public Sub ClearAllContent()
    Cells.Clear
End Sub
```

范例 4-21　插行并作分类汇总

这类技巧还可衍生出另一种应用，就是每隔几行进行同一组数据的汇总（即 Excel 的“数据/分类汇总”功能，只是没有缩进效果），如图 4-19 所示。

	A	B
1	A	1
2	A	2
3	A	3
4	A Sum:	6
5	B	4
6	B	5
7	B Sum:	9
8	C	6
9	C	7
10	C	8
11	C Sum:	21
12	D	9
13	D Sum:	9
14	E	10
15	E Sum:	10
16	Total:	55
17		

图 4-19

这个就没有那么规律了，因为每隔几行并不固定，所以要不断记住每行第一个单元格的值，遇到不同的，就插入一行并加总。可是不要真的先插入行，因为问题大都是以 UsedRange.Rows 来判断总行数，如果在统计时插入新行，UsedRange 的内容变会不断膨胀，如此将导致程序进入无穷循环。

所以先记住到时候要插在哪一行，还有插在那一行的总和，等通通统计完后，再一次进行插入新行的动作，完整代码如下：

综合技巧范例_插入行或列_总和小计.xls

```
Public Sub Reset()
    Sheets("Sheet1").Activate
    Cells.Clear
    Range("A1").Select
    Sheets("Sample").UsedRange.Copy
    Sheets("Sheet1").Paste
    Range("A1").Select
End Sub

Public Sub InsertRow()
```


```
Call Reset

Dim lvLastValue As Variant
lvLastValue = ActiveSheet.UsedRange.Rows(1).Cells(1).Value

Dim lnRowNumber As Integer, lnSum As Integer, lnSumTotal As Integer
Dim lnRowNumberToAdd As Integer
lnRowNumber = 0
lnSum = 0
lnSumTotal = 0
lnRowNumberToAdd = 0

Dim lvaGroupByValue() As Variant
Dim lnaRowToAdd() As Integer
Dim lnaRowSum() As Integer

Dim loRange As Range
For Each loRange In ActiveSheet.UsedRange.Rows
    lnRowNumber = lnRowNumber + 1
    If loRange.Cells(1).Value <> lvLastValue Then
        lnRowNumberToAdd = lnRowNumberToAdd + 1
        ReDim Preserve lvaGroupByValue(lnRowNumberToAdd)
        ReDim Preserve lnaRowToAdd(lnRowNumberToAdd)
        ReDim Preserve lnaRowSum(lnRowNumberToAdd)
        lvaGroupByValue(lnRowNumberToAdd) = lvLastValue
        lnaRowToAdd(lnRowNumberToAdd) = lnRowNumber
        lnaRowSum(lnRowNumberToAdd) = lnSum

        lnSum = 0
    End If
    lnSum = lnSum + loRange.Cells(2).Value
    lnSumTotal = lnSumTotal + loRange.Cells(2).Value
    lvLastValue = loRange.Cells(1).Value
Next

' 最后一个也要算在内，因为最后一个没有更后面的一个可供比较
```

```
    lnRowNumberToAdd = lnRowNumberToAdd + 1
    ReDim Preserve lvaGroupByValue(lnRowNumberToAdd)
    ReDim Preserve lnaRowToAdd(lnRowNumberToAdd)
    ReDim Preserve lnaRowSum(lnRowNumberToAdd)
    lvaGroupByValue(lnRowNumberToAdd) = lvLastValue
    lnaRowToAdd(lnRowNumberToAdd) = ActiveSheet.UsedRange.
Rows.Count + 1
    lnaRowSum(lnRowNumberToAdd) = lnSum

    ' 通通统计完后 , 再一次进行插入新列的操作
    Dim i As Integer
    For i = 1 To UBound(lnaRowToAdd)
         Rows(lnaRowToAdd(i) + (i - 1)).Insert Shift:=xlDown
        Rows(lnaRowToAdd(i) + (i - 1)).Cells(1).Interior.ColorIndex = 6
        Rows(lnaRowToAdd(i) + (i - 1)).Cells(2).Interior.ColorIndex=6
        Rows(lnaRowToAdd(i) + (i - 1)).Cells(1).Value =
lvaGroupByValue(i) & " Sum:"
        Rows(lnaRowToAdd(i) + (i - 1)).Cells(2).Value = lnaRowSum(i)
    Next

    Cells(lnaRowToAdd(lnRowNumberToAdd) + lnRowNumberToAdd,
1).Interior.ColorIndex = 4
    Cells(lnaRowToAdd(lnRowNumberToAdd) + lnRowNumberToAdd,
2).Interior.ColorIndex = 4
    Cells(lnaRowToAdd(lnRowNumberToAdd) + lnRowNumberToAdd,
1).Value = "Total:"
    Cells(lnaRowToAdd(lnRowNumberToAdd) + lnRowNumberToAdd,
2).Value = lnSumTotal

    Range("A1").Select
End Sub
```

这种问题也会遇到“最后一个值没有下一个值可供比较”的状况，所以这点也要考虑进去。

范例 4-22　对齐所有统计图表

统计图表一多，要逐个去调整其大小及位置很麻烦，Excel 可以帮我们做水平或垂直方向的对齐，但大小得用鼠标慢慢调整，要调到每个工作表内的图表都一般大，往往耗费时间，类似这种工作表内的图表、超级链接、批注、OLE 对象、垂直分页线（VPageBreaks）等都存放在 Sheet 对象旗下的各种 Collection 中，很方便就可以——取出并操作。

综合技巧范例_对齐所有统计图表.xls

```
Public Sub ShapeAlign()
    Dim loSheet As Worksheet
    Dim lnLastLeft As Integer, lnLastTop As Integer
    Dim i As Integer, c As Integer
    Dim loShape As Shape
    Dim lnShapeWidth As Integer, lnShapeHeight As Integer

    lnLastLeft = 100         ' 所有图表的左方位置
    lnLastTop = 10           ' 第一张图表的上方位置
    lnShapeWidth = 200       ' 所有图表的宽度
    lnShapeHeight = 150      ' 所有图表的高度
    lnVerticalSpace = 10     ' 图表与图表之间的垂直间距

    For Each loSheet In ActiveWorkbook.Sheets
        i = 0
        c = 0
        For Each loShape In loSheet.Shapes
            ' 统计图表才对齐，其他图表则忽略
            If loShape.Type = msoChart Then
                c = c + 1
                i = i + 1

                loShape.Top = lnLastTop + Int(c - 1) *
(lnVerticalSpace + lnShapeHeight)
```

```
                loShape.Left = lnLastLeft
                loShape.Width = lnShapeWidth
                loShape.Height = lnShapeHeight

                lnLastLeft = loShape.Left
            End If
        Next
    Next
End Sub
```

Shapes Collection 存放的不只统计图表而已，连单元格批注、按钮这种对象都算 Shape，所以要把它们区分开来，Shape 的 Type 属性就可以判断是哪种图表。所以，举一反 N，如果想砍掉所有按钮、设置所有按钮的外观、设置所有统计图表的标题、统一所有批注的大小、设置所有文本输入框（TextBox）的外观等，也都可以使用此 Type 过滤取出并操作。

同理，其他的 Collection 也是这样 For Each 取出并操作（方法名称也大都顾名思义，例如 Delete、Move、Copy 之类的），所以，像是删除所有批注、列出所有超级链接、删除所有图表等都可视作一样的技巧。

不过有些 Collection 却有例外，像是要删除所有的名称：

```
Dim loName As Name
For Each loName In ActiveWorkbook.Names
    loName.Delete
Next
```

照理说，名称应该是在工作表之下，但如果用：

```
    Dim loName As Name
    For Each loName In ActiveSheet.Names
        loName.Delete
    Next
```

却删不了任何名称。但 Sheet 对象却又有 Names Collection。

对于 Names Collection，如果其命名是以“工作表名称! ”开头的，Excel 就会把它归类到该工作表的旗下，但也同时属于整个工作簿的旗下。所以，如果有个名称叫做“Sheet1!Orders”，那么：

```
    For Each loName In ActiveSheet.Names
    Next
```

就可以抓得到了。

范例 4-23　删除所有分页线

虽说垂直分页线是 Sheet 旗下的一个 Collection，但要删掉所有分页线却与它无关：

```
ActiveSheet.Cells.PageBreak = xlPageBreakNone
```

是的，只要这一行就可以搞定了。

范例 4-24　删除空的工作表

之前用过元数据的方法删除空的工作表，这里有个传统方法的范例也提供给您做参考。

所谓“空的工作表”，就是存盘后只有一个单元格，且该单元

格的内容为空白的工作表。为何要先保存呢？因为这样 Excel 才会更新 UsedRange 对象的内容。

所以删除空的工作表就变得很简单了。

综合技巧范例_删除空白工作表.xls

```
Public Sub Reset()
    Application.DisplayAlerts = False
    Dim loSheet As Worksheet
    For Each loSheet In ActiveWorkbook.Worksheets
        If loSheet.Name <> "Sheet1" Then
            loSheet.Delete
        End If
    Next
    Application.DisplayAlerts = False
    Sheets(1).Cells(1, 1).Value = 1

    Dim s As Integer
    For s = 2 To 7
        Sheets.Add After:=Sheets(s - 1)
        Sheets(s).Name = "Sheet" & Trim(Str(s))
        If s = 2 Or s = 3 Or s = 7 Then
            Sheets(s).Cells(1, 1).Value = s
        End If
    Next

    Sheets(1).Activate
End Sub

Public Sub DeleteEmptySheet()
    Call Reset

    Application.DisplayAlerts = False
    Dim loSheet As Worksheet
    For Each loSheet In ActiveWorkbook.Worksheets
        If loSheet.Name <> "Sheet1" Then
```

```
            If loSheet.UsedRange.Cells.Count = 1 Then
                If IsEmpty(loSheet.UsedRange.Cells(1).Value) Then
                    loSheet.Delete
                End If
            End If
        End If
    Next

    Application.DisplayAlerts = False
    Sheets(1).Activate
End Sub
```

另一种判断空白工作表的方式更简洁：

```
IsEmpty(ActiveSheet.UsedRange)
```

范例 4-25　列出某目录下含子目录的所有目录及文件

这个常用功能可直接利用 Windows Script Host（WSH）里的现成组件，再利用简单的递归——自己调用自己即可轻松完成此功能。

你可以把这个搜索结果与在 Windows 资源管理器中按 F3 键的搜索结果对照一番，它们是完全吻合的。

综合技巧范例_列出某目录下含子目录的所有目录及文件.xls

```
Public lnSearchCount As Long

Public Sub SearchFilesFor(ByVal pcDirectory As String)
    Dim loFile As File
    Dim loFolder As Folder

    Set loShell = CreateObject("WScript.Shell")
    Set loFileObject = CreateObject("Scripting.FileSystemObject")
```

```
    If Not loFileObject.FolderExists(pcDirectory) Then
        Exit Sub
    End If

    Set loCurrentDirectory =
loFileObject.GetFolder(pcDirectory)
    For Each loFile In loCurrentDirectory.Files
        lnSearchCount = lnSearchCount + 1
        Cells(lnSearchCount, 1).Value = "'" & loFile.Path
        Cells(lnSearchCount, 2).Value = "'" &
loFile.ShortPath
        Cells(lnSearchCount, 3).Value = "'" &
loFile.ShortName
        Cells(lnSearchCount, 4).Value = "文件"
    Next

    ' search sub directory
    For Each loFolder In loCurrentDirectory.SubFolders
        lnSearchCount = lnSearchCount + 1
         Cells(lnSearchCount, 1).Value = "'" &
loFolder.Path
        Cells(lnSearchCount, 2).Value = "'" &
loFolder.ShortPath
        Cells(lnSearchCount, 3).Value = "'" &
loFolder.ShortName
        Cells(lnSearchCount, 4).Value = "目录"

        SearchFilesFor (loFolder.Path)
    Next
End Sub

Public Sub SearchFiles()
    Cells.Clear
    lnSearchCount = 1

    Cells(lnSearchCount, 1).Value = "Path"
```


```
    Cells(lnSearchCount, 2).Value = "ShortPath"
    Cells(lnSearchCount, 3).Value = "ShortName"
    Cells(lnSearchCount, 4).Value = "Type"

    Call SearchFilesFor("C:\Program Files\Microsoft Office")

    Cells(1, 1).Select
    MsgBox ("共找到 " & Trim(Str(lnSearchCount - 1)) & " 个目录及文件。")
End Sub
```

有了这段程序，你就可以搜索任何想要的文件，这里只是把它列出来，你当然可以调用 File 对象的 Copy、Move、Delete 等方法达到复制、重命名、转移的功能，等于在你的程序中无缝（seamless）加入了资源管理器的部分功能（就是非外挂，直接与你的程序结合）。

如果要将找到的文件打开，可以利用 Shell 对象：

```
    Set loShell = CreateObject("Shell.Application")
    loShell.Open loFile.Path
    Set loShell = Nothing
```

当然，也可以利用 Excel 内置的 FileSearch 对象来搜索，程序代码如下。

综合技巧范例_列出某目录下含子目录的所有目录及文件_另一种写法.xls

```
Public Sub SearchFilesFor(ByVal pcDirectory As String)
    Dim i As Integer

    With Application.FileSearch
        .FileType = msoFileTypeAllFiles
```

```
        .SearchSubFolders = True
        .LookIn = pcDirectory

        If .Execute() > 0 Then
            For i = 1 To .FoundFiles.Count
                Sheets(1).Cells(i + 1, 1).Value = .FoundFiles(i)
            Next
        End If

        MsgBox ("共找到 " & Trim(Str(.FoundFiles.Count)) & " 个文件。")
    End With
End Sub

Public Sub SearchFiles()
    Cells.Clear

    Cells(1, 1).Value = "Path"

    SearchFilesFor ("C:\Program Files\Microsoft Office")

    Cells(1, 1).Select
End Sub
```

FileSearch 的搜索速度比 WSH 对象快得多，不过，FileSearch 对象无法找出目录，而且，若找不到文件，FoundFiles 还是会维持上一次的搜索结果，这点让人疑惑。

这个范例配合之前所讲的一切关于 Excel 的技巧可以做到大量的文件处理，例如把整个目录下的 Excel 文件都打开，然后把其中的空白工作表都删掉。

范例 4-26　自定义菜单

在 Excel 的任一单元格按下鼠标右键，会出现一些菜单选项让

我们选择，如果我们写了一个跟单元格相关功能的宏，想把这个功能也加入快捷菜单之中，这时可以利用 Application.CommandBars 对象。

跟单元格相关的菜单都放在 Application.CommandBars("cell").Controls 之中，我们只需在工作表的 Worksheet_BeforeRight- Click 事件中加入自定义的项目即可：

综合技巧范例_自定义菜单.xls

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range,
Cancel As Boolean)
    Dim loControl As CommandBarControl

    ' 先砍掉已定义过的、重复名称的菜单项目
    DeleteCustomMenuItem

    With CommandBars("cell").Controls.Add
        .Caption = "当前时间"
        .OnAction = "InsertNow"
        .Parameter = Target.Address
    End With
End Sub

Public Sub DeleteCustomMenuItem()
    For Each loControl In CommandBars("cell").Controls
        If loControl.Caption = "当前时间" And Not
loControl.BuiltIn Then
            loControl.Delete
            Exit For
        End If
    Next
End Sub
```

OnAction 设置为 InsertNow，因此要真的写一个 InsertNow（不

然会出错），代码如下。

综合技巧范例_自定义菜单.xls

```
Public Sub InsertNow()
    With Range(CommandBars("cell").Controls("当前时间
").Parameter)
        .Value = Now
        .Columns.AutoFit
    End With
End Sub
```

最后要记得在 Workbook_BeforeClose 事件中删除自定义的菜单项目，不然等这个范例关掉后，别的 Excel 文件打开时，也会看到这个自定义的项目（当然，如果你就是要留着它的话就不要删除）：

综合技巧范例_自定义菜单.xls

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    DeleteCustomMenuItem
End Sub
```

从 Worksheet_BeforeRightClick 事件中接收到的 Target 就是当时按下鼠标右键时的单元格地址，把该地址当作是添加菜单项目的参数，最后再把它传给 InsertNow 的 Range 对象就可以对当时的单元格进行操作了。

可以对一个单元格进行操作，当然也就可以对一堆单元格操作，因为 Worksheet_BeforeRightClick 事件中的 Target，在你选的是一堆单元格而非一个单元格时，代表的即是一个 Range 而非一个 Cell。所以在按下鼠标右键时要计算标准差：

综合技巧范例_自定义菜单_计算标准差.xls

```
Public Sub CalculateStdDev()
```

```
    Dim loRange As Range
    Set loRange = Range(CommandBars("cell").Controls("计算标
准差").Parameter)
    Range("IV65536").Formula = "=STDEV(" & loRange.Address & ")"
    If VarType(Range("IV65536").Value) <> vbError Then
        MsgBox (Range("IV65536").Value)
    End If
    Range("IV65536").ClearContents
End Sub
```

在加入自定义的项目之前记得先将之前加的删掉，或在 OnAction 启动的宏中将之删掉。

OnAction 属性无法“直接”传参数给我们的自定义宏，也就是无法写成“OnAction="InsertNow('$A$1')" ”，但可以通过项目的另一属性 Parameter 来传参数。但经我测试结果，不能传入对象类型的参数，所以上例传的是 Target.Address 而非 Target。

自定义如上例的菜单后，我们就可以按鼠标右键插入“当前时间”，而且可以插入不止一个，如果你选取一块区域（或多块区域），然后按鼠标右键执行“当前时间”，将会发现被选取区域的单元格都填充了“当前时间”。这使我想到，那能不能写个简单的填充数列功能呢？例如，选一块区域，此功能就会填入 1～20（或 2002/11/01 12:01～2002/11/01 12:20），但我发现选取多个单元格之后执行“当前时间”，基本上 Excel 做的是“重复”执行自定义宏（如上例的 InsertNow），即使在该段程序代码中使用全局（Public）变量，例如.Value = Now + lnNowAdd，然后 lnNowAdd 一直累加，还是不能做到填充数列——只会填充一堆的 Now + lnNowAdd。

如果你想传入不止一个参数给自定义宏，那除了 Parameter 还有 Tag 可以用，如果还想再多，大概就只能使用全局变量了。

如果单就上例而言，也可以在 InsertNow 中直接用 ActiveCell，而不必传任何参数（但传参数是个很有用的技巧）。

单就上例而言，如果选定的范围很大，那计算起标准差时速度就不可避免地会慢下来，这是设计此类功能时要考虑的一点。

如果要操作其他菜单也是一样，例如在“数据”菜单中加上与上例一样的“计算标准差”项目，代码如下。

综合技巧范例_自定义菜单_计算标准差_加在数据菜单.xls

```
Public Sub AddCustomMenuItem()
    DeleteCustomMenuItem
    With CommandBars("Data").Controls.Add
        .Caption = "计算标准差"
        .OnAction = "CalculateStdDev"
    End With
End Sub

Public Sub CalculateStdDev()
    Range("IV65536").Formula = "=STDEV(" & Selection.Address & ")"
    If VarType(Range("IV65536").Value) <> vbError Then
        MsgBox (Range("IV65536").Value)
    End If
    Range("IV65536").ClearContents
End Sub

Public Sub DeleteCustomMenuItem()
    For Each loControl In CommandBars("Data").Controls
        If loControl.Caption = "计算标准差" And Not loControl.
BuiltIn Then
            loControl.Delete
            Exit For
        End If
    Next
End Sub
```

不只可以加上自定义的项目，原先的项目也可以操作，例如修改文本：

```
    Commandbars("Data").Controls("小计(B)...").Caption = "小计及总计(B)..."
```

或删除：

```
    Commandbars("Data").Controls("小计(B)...").Delete
```

CommandBars 是以文本当作索引，所以连“...”都要写才抓得到该项目。如果删除后想恢复，可以到“工具/自定义/命令”处，把原先的项目用鼠标拉回来。

如果不想“寄人篱下”，而是要自己独立出来，变成一个像图 4-20 所示的新菜单。

图 4-20

尽管这也算是一种“寄人篱下”，但它依附的却是 Excel 的主菜单。

综合技巧范例_自定义菜单_新的菜单.xls

```
Public Sub CreateCustomMenu()
    Dim loMyCustomBar As CommandBarPopup
    Dim loMyCustomPopup As CommandBarPopup
    Dim loMyCustomButton As CommandBarButton
```

```
    DeleteCustomMenu

    Set loMyCustomBar = CommandBars("Worksheet Menu Bar").
Controls.Add(Type:=msoControlPopup)

    With loMyCustomBar
        .Caption = "自定义菜单(&X)"

        Set loMyCustomPopup = .Controls.Add(Type:=msoControlPopup)
        With loMyCustomPopup
            .Caption = "子菜单(&S)"
            Set loMyCustomButton=.Controls.Add(Type: =
msoControlButton)
            With loMyCustomButton
                .Caption = "按钮 1(&A)"
                .OnAction = "Button1_Click"
                .Style = msoButtonCaption
            End With
        End With

        Set loMyCustomButton =.Controls.Add(Type:=msoControlButton)
        With loMyCustomButton
            .Caption = "按钮 2(&B)"
            .OnAction = "Button2_Click"
            .Style = msoButtonCaption
        End With

        .Visible = True
    End With
End Sub

Public Sub DeleteCustomMenu()
    Dim loControl As CommandBarControl
    For Each loControl In CommandBars("Worksheet Menu Bar").Controls
        If loControl.Caption = "自定义菜单(&X)" And Not
loControl.BuiltIn Then
```


```
            loControl.Delete
            Exit For
        End If
    Next
End Sub

Public Sub Button1_Click()
    MsgBox ("按钮 1")
End Sub

Public Sub Button2_Click()
    MsgBox ("按钮 2")
End Sub
```

你可以看到，新菜单是加在名为“Worksheet Menu Bar”的CommandBar之下的，但我们怎么知道有“Worksheet Menu Bar”这个名字呢？

就把 CommandBars Collection 通通都列出来即可：

```
Public Sub ListAllCommandBarName()
    Dim c As Integer
    Dim loCommandBar As CommandBar
    For Each loCommandBar In CommandBars
        c = c + 1
        Cells(c, 1).Value = loCommandBar.Name
        Cells(c, 2).Value = loCommandBar.NameLocal
    Next
End Sub
```

结果如下表所示。

名称	说明
Worksheet Menu Bar	工作表菜单栏
Chart Menu Bar	图表菜单栏

（续表）

名称	说明
Standard	常用
Formatting	格式
PivotTable	数据透视表
Chart	图表
Reviewing	审阅
Forms	窗体
Stop Recording	停止录制
External Data	外部数据
Auditing	公式审核
Full Screen	全屏显示
Circular Reference	循环应用
Visual Basic	Visual Basic
Web	Web
Control Toolbox	控件工具箱
Exit Design Mode	退出设计模式
Refresh	刷新
Drawing	绘图
WordArt	艺术字
Picture	图片
Query and Pivot	查询及数据透视表
PivotChart Menu	数据透视表菜单
Workbook tabs	工作簿标签
Cell	单元格
Column	列
Row	行
Ply	工作表
XLM Cell	XLM 单元格
Document	文件
Desktop	桌面
Nondefault Drag and Drop	非默认拖放功能

（续表）

名称	说明
AutoFill	自动填充
Button	按钮
Dialog	对话框
Series	系列
Plot Area	绘图画布
Floor and Walls	底纹及背景
Chart	图表
Format Data Series	数据系列格式
Format Axis	坐标轴格式
Format Legend Entry	图例格式
Formula Bar	编辑栏
PivotTable Context Menu	数据透视表菜单
Query	查找
Query Layout	查找版面配置
AutoCalculate	自动求和
Object/Plot	对象/绘图区
Title Bar (Charting)	标题栏（图表）
Layout	版面配置
Pivot Chart Popup	数据透视表弹出菜单
Phonetic Information	拼音指南
Shadow Settings	阴影设置
3-D Settings	三维设置
Borders	边框
Chart Type	图表类型
Pattern	样式
Font Color	字体颜色
Fill Color	填充颜色
Line Color	线条颜色
Order	排序
Nudge	微移

（续表）

名称	说明
Align or Distribute	对齐或分布
Rotate or Flip	旋转或翻转
Lines	线条
Connectors	连接符
AutoShapes	自选图形
Callouts	标注
Flowchart	流程图
Block Arrows	箭头总汇
Stars & Banners	星与旗帜
Basic Shapes	基本形状
Shapes	图示
Inactive Chart	非活动状态图表
Excel Control	Excel 控件
Curve	曲线
Curve Node	曲线节点
Curve Segment	曲线片段
Pictures Context Menu	图片菜单
OLE Object	OLE 对象
ActiveX Control	ActiveX 控件
WordArt Context Menu	艺术字菜单
Rotate Mode	旋转模式
Connector	连接符
Script Anchor Popup	Script 定位点弹出菜单
符号栏	符号栏
WavEditer I	WavEditer I
WavEditer II	WavEditer II
Add Command	添加命令
Built-in Menus	内置菜单
System	系统
Clipboard	剪贴板

我没有刻意依照英文字母排列（当中居然还有重复的呢）。看到这张表，应该可以找出哪个名称代表哪个菜单，如果还是找不到，就让每个菜单“露出原形”：

```
loCommandBar.Position = msoBarFloating
```

就可以知道谁是谁了。

OK，如果真的要完全自成一个菜单，那就把“Worksheet Menu Bar”字样拿掉，代码如下。

综合技巧范例_自定义菜单_新的菜单_2.xls

```
Public Sub CreateCustomMenu()
    Dim loMyCustomBar As CommandBar
    Dim loMyCustomPopup As CommandBarPopup
    Dim loMyCustomButton As CommandBarButton

    DeleteCustomMenu

    Set loMyCustomBar = CommandBars.Add
        With loMyCustomBar
        .Name = "自定义菜单"
        .Top = 200
        .Left = 100

        Set loMyCustomPopup = .Controls.Add(Type:=msoControlPopup)
        With loMyCustomPopup
            .Caption = "子菜单(&S)"
            Set loMyCustomButton = .Controls.Add(Type: =
msoControlButton)
            With loMyCustomButton
                .Caption = "按钮 1(&A)"
                .OnAction = "Button1_Click"
                .Style = msoButtonCaption
            End With
```

```
        End With

        Set loMyCustomButton = .Controls.Add(Type:=msoControlButton)
        With loMyCustomButton
            .Caption = "按钮 2(&B)"
            .OnAction = "Button2_Click"
            .Style = msoButtonCaption
        End With

        .Visible = True
    End With
End Sub

Public Sub DeleteCustomMenu()
    Dim loControl As CommandBar
    For Each loControl In CommandBars
        If loControl.Name = "自定义菜单" And Not loControl. BuiltIn Then
            loControl.Delete
            Exit For
        End If
    Next
End Sub

Public Sub Button1_Click()
    MsgBox ("按钮 1")
End Sub

Public Sub Button2_Click()
    MsgBox ("按钮 2")
End Sub
```

请注意，CommandBar 对象是没有 Caption 属性的，对等概念是 Name 属性，我们是以 Name 来找到 CommandBar 的。

执行此范例，你会看到有一个浮动的菜单，事实上这不奇怪，因为每个 Excel 的菜单其实都是浮动的，每个菜单左边的那根线就

是用来拖曳菜单的，或是利用刚刚提到的“loCommandBar.Position = msoBarFloating”方法也能了解这个浮动的菜单。

除了建立有按钮的菜单，还能建立有下拉列表的菜单。图 4-21 是工作表清单菜单，在单一工作簿内的工作表很多时这个菜单很有用。

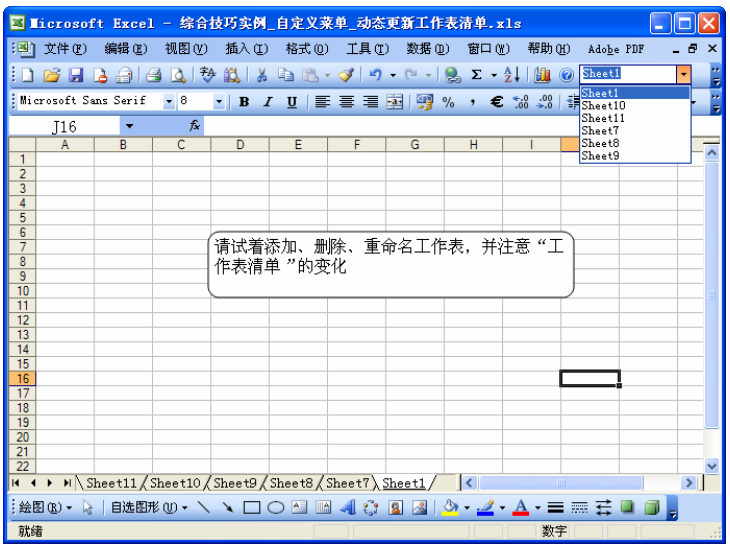


图 4-21

以下是完整的程序代码，基本上是利用与前一例同样的技巧。

综合技巧范例_自定义菜单_动态更新工作表清单.xls

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)
    Module1.建立工作表清单
End Sub

Private Sub Workbook_Open()
    Module1.建立工作表清单
End Sub

Private Sub Workbook_SheetActivate(ByVal Sh As Object)
```

```
    ' Excel 没有删除工作表事件，但工作表被删除后，必然会有某一工作表
被 Activate
    Module1.建立工作表清单
End Sub

Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object,
ByVal Target As Range)
    ' Excel 没有工作表重命名事件，但工作表被重命名后，必然会 Activate
某一单元格
    Module1.建立工作表清单
End Sub

Public Const gcSheetListName = "工作表清单"

Public Sub 到某张工作表()
    Dim lcSheetName As String
    Dim llFound As Boolean
    Dim loSheet As Worksheet

    lcSheetName = Application.CommandBars("Standard").Controls
(gcSheetListName).Text
    llFound = False
    For Each loSheet In ActiveWorkbook.Sheets
        If loSheet.Name = lcSheetName Then
            Sheets(loSheet.Name).Activate
            llFound = True
            Exit For
        End If
    Next

    If Not llFound Then
        MsgBox lcSheetName & ": 找不到这张工作表"
    End If
End Sub

Public Sub 建立工作表清单()
```


```
    Dim loSheetList As CommandBarComboBox
    Dim loControl As CommandBarControl
    Dim loSheet As Worksheet
    Dim llFound As Boolean

    ' 此清单是否已经建立?
    llFound = False
    For Each loControl In
Application.CommandBars("Standard").Controls
        If loControl.Caption = gcSheetListName Then
            llFound = True
            Exit For
        End If
    Next

    ' 如果已经建立则直接指到它，不然就建立它
    If Not llFound Then
        Set loSheetList = Application.CommandBars("Standard").
Controls.Add(msoControlComboBox)
    Else
        Set loSheetList = Application.CommandBars("Standard").
Controls(gcSheetListName)
    End If

    ' 把目前工作簿所有的工作表名称放进数组中并排序
    loSheetList.Caption = gcSheetListName
    loSheetList.OnAction = "到某张工作表"

    Dim lcaSheetName() As String
    Dim s As Integer
    s = 0
    For Each loSheet In ActiveWorkbook.Sheets
        s = s + 1
        ReDim Preserve lcaSheetName(s)
        lcaSheetName(s) = loSheet.Name
    Next
```

```
    Dim i As Integer, j As Integer
    Dim lcSheetName As String
    For i = 1 To UBound(lcaSheetName) - 1
        For j = i + 1 To UBound(lcaSheetName)
            If lcaSheetName(i) > lcaSheetName(j) Then
                lcSheetName = lcaSheetName(i)
                lcaSheetName(i) = lcaSheetName(j)
                lcaSheetName(j) = lcSheetName
            End If
        Next
    Next

    ' 然后将排序后的数组放到工作表清单中
    With loSheetList
        .Clear
        For i = 1 To UBound(lcaSheetName)
            .AddItem (lcaSheetName(i))
        Next
        .Text = ActiveWorkbook.ActiveSheet.Name
    End With
End Sub
```

“到某张工作表”子程序之所以要逐个检查工作表的名称，而不直接以 Sheets(CommandBars("Standard").Controls(gcSheetListName). Text 来进入该张工作表，是因为我们的下拉式菜单可以让用户输入，这样才可以“随机搜索”，输入哪个工作表名称就到该张工作表，而不必一直用鼠标拖动下拉式菜单的滚动条，而使用者可能会输入错误，所以必须检查，如果输入错误则给他一个提示信息。

此工作表清单既然名为“动态”，就表示当工作表发生变动时都必须给出最新的清单，变动可能发生在工作表添加、删除、重命名之时，然而，除了添加之外，删除与重命名都没有事件发生，所以必须利用一点小技巧。

工作表被删除后必然会有某一工作表被 Activate；而工作表被重命名后必然（马上）会 Activate 某一单元格（虽然重命名后按 Enter 键并不会 Activate 某一单元格，但不可能永远不 Activate），所以可分别在 Workbook_SheetActivate 与 Workbook_SheetSelectionChange 事件中进行“更新工作表清单”的操作。

范例 4-27　判断某区域是否被选

Excel 可以让我们选取不连续的多个区域，不论是用鼠标或程序的方式皆可，选取之后，Application.Selection 就会记载着目前被选到的单元格，如果选了多个区域，Application. Selection. Areas.Count 也可以告诉我们共选了几块区域。但如果反过来想判断某个单元格或某块区域有没有被选到的话，该怎么办呢？

当然，可以利用一个循环，看看想判断的单元格或区域在不在 Selection 所记载的 Cells 之中，一经判断确实被选到就立即跳出循环，但假设你选到了 A1:A1000，而想判断 A999 是否被选到，那这个循环得跑上 999 次才能知道的确被选到，此种循环判断的方式的确是慢了一点。

我们可以利用 Application.Intersect 函数，它会传回我们指定的多个区域的“集合”区域，其传回值为 Range 类型。

如果你在实时运算窗口输入：

```
? Application.Intersect(Range("A1:A10"), _
Range("A3:A20"), Range("A4:A30")).Address
```

那会传回“$A$4:$A$10”。

所以，判断一个单元格或区域是否被选，就等于判断该区域跟 Selection 相交的区域是否就是该区域本身，如果是，那就代表该区域整块被选取了。除此之外，Intersect 也可以判断某个区域是否位于另一个区域之内，道理是相同的，如果区域 A 在区域 B 之内，两个区域相交的结果就会是区域 A。

综合技巧范例_判断某区域是否被选.xls

```
Public Function 是否被选(Target As Range) As Boolean
    是否被选 = False

    Dim loIntersectRange As Range

    Set loIntersectRange = Application.Intersect(Target,
Selection)
    If Not loIntersectRange Is Nothing Then
        If loIntersectRange.Address = Target.Address Then
            是否被选 = True
        End If
    End If
End Function

Public Function 是否在范围之内(ByVal Target_1 As Range, ByVal
Target_2 As Range) As Boolean
    是否在范围之内 = False

    Dim loRange As Range
    Set loRange = Intersect(Target_1, Target_2)
    If Not loRange Is Nothing Then
        是否在范围之内 = (loRange.Address = Target_1.Address)
    End If
End Function

Public Sub 测试是否被选()
    Range("A1:A3").Select
```

```
    MsgBox "A1:A3 是否被选: " + IIf(是否被选(Range("A1:A3")), "是", "否")
    MsgBox "A2:A3 是否被选: " + IIf(是否被选(Range("A2:A3")), "是", "否")
    MsgBox "A2:C3 是否被选: " + IIf(是否被选(Range("A2:C3")), "是", "否")
    Range("A1").Select
End Sub

Public Sub 测试是否在范围之内()
    MsgBox "Range(A2:B3) 是否在 Range(C2:D3) 之内: " + IIf(是否在范围之内(Range("A2:B3"), Range("C2:D3")), "是", "否")
    MsgBox "Range(A2:B3) 是否在 Range(A1:D3) 之内: " + IIf(是否在范围之内(Range("A2:B3"), Range("A1:D3")), "是", "否")
End Sub
```

如果两区域不相交，Intersect 会传回 Nothing，所以必须先行判断是否为 Nothing 才可以进行 Address 的比较。

范例 4-28　导入宏并执行

想象一个情节，有个宏很好用，于是想让每个工作簿都具备该宏，但问题是，宏是跟着 Excel 文件在走的，每个工作簿都具备该宏的结果就造成大量重复，最后，如果该宏面临改版（别忘了，软件一定要变才有用），那所有工作簿内的该宏“分身”不是通通都得同步改版？

要是能够用“实时加载，实时执行，然后实时删除”的方式把宏在要执行之时才导入工作簿中，执行完后“立刻走人”，该宏就能够达到最大程度的重复使用性了。

有办法的，Excel 是个高度自动化的软件，不但前台的工作簿如此，后台的 VBE 程序整合环境也都可以实现一定程度的自动化。也就是，我们可以用程序来操作控制 VBE。

综合技巧范例_导入宏并执行.xls

```
Public Sub ImportMacroAndRun()
    ActiveWorkbook.Save

    With Application
        For i = 1 To .VBE.VBProjects.Count
            Set loProject = .VBE.VBProjects(i)
            If loProject.Filename = ActiveWorkbook.FullName Then
                loProject.VBComponents.Import (ActiveWorkbook.
Path & "\导入的宏.bas")
                Set loModule = loProject.VBComponents(loProject.
VBComponents.Count)
            End If
        Next

        .Run "'" & ActiveWorkbook.FullName & "'!" &
loModule.Name & ".SomeMacro"
        .Run "'" & ActiveWorkbook.FullName & "'!" &
loModule.Name & ".SomeMacroWithPara", "有参数的状况"

        loProject.VBComponents.Remove (loModule)
    End With
End Sub
```

注意第一行的 ActiveWorkbook.Save，如果在写这个宏时还未存过盘，那么底下的 ActiveWorkbook.FullName 将会出问题。

这个看似不起眼的功能极其强大且富有弹性，它意味着我们可以动态产生一个文本文件（使用 WSH 或 Excel 的 Open+Write 命令），内容为 Excel VBA 命令，然后把某个（未打开的）工作簿打开，动态加载也是动态产生的该宏，然后执行宏，最后删除宏，“不留一点痕迹”。

这种执行动态程序代码的特色，除了当今 Web 程序设计的方

式外（HTML+JavaScript 多为后台动态产生），据我所知，也只有像早期的 dBase、近期的 FoxPro、深具传统且仍热门的 Perl 以及 JavaScript 等开发工具才具备这种“特异功能”。新式的面向对象语言如 Java、.NET 等，可得大费周章的通过一种称为“Reflection”的机制才能做到一样的事呢！

范例 4-29　导出并删除宏

可以导入宏，就可以导出宏、删除宏，如果某个目录下有很多工作簿里都有宏，想一次把这些宏都删除，可使用以下程序代码。

综合技巧范例_导出宏并删除.xls

```
Public Sub ExportMacroAndDelete()
    Dim i As Integer, e As Integer
    Dim lcPath As String, lcFile As String
    Dim loProject As Variant
    Dim loModule As Variant

    lcPath = ActiveWorkbook.Path & "\这些工作簿的宏要被导出\"
    lcFile = Dir(lcPath & "*.xls")

    Do Until lcFile = ""
        Set loWorkBook = Workbooks.Open(lcPath & lcFile)

        For i = 1 To Application.VBE.VBProjects.Count
            Set loProject = Application.VBE.VBProjects(i)
            If loProject.Filename = loWorkBook.FullName Then
                For e = 1 To loProject.VBComponents.Count
                    Set loModule = loProject.VBComponents(e)
                    loModule.Export (lcPath & "\" &
loWorkBook.Name & "_" & loModule.Name & ".bas")
```

```
                    If loModule.Type <= 3 Then

loWorkBook.VBProject.VBComponents.Remove (loModule)
                    Else
                        loModule.CodeModule.DeleteLines 1,
loModule.CodeModule.CountOfLines
                    End If
                Next
            End If
        Next

        loWorkBook.Close (True)
        lcFile = Dir
    Loop
End Sub
```

配合之前的“列出某目录下含子目录的所有目录及文件”技巧，就可以把整个硬盘中不想要的宏删掉了。

范例 4-30　批量修改多个工作簿中的宏

另一种可能的应用是把很多工作簿内的某个宏改版，这就要先把该宏删除，然后“写入”改版后的宏。

综合技巧范例_改版宏.xls

```
' 请在“工具/引用”中，引用
'“Microsoft Visual Basic for Applications Extensibility *.*”
'（*.* 指的是版本编号，引用最新的版本即可）
Public Sub ReplaceSub(ByVal pcSubName As String)
    Dim loVBComponent As VBComponent

    lcPath = ActiveWorkbook.Path & "\这些工作簿的宏要被改版\"
    lcFile = Dir(lcPath & "*.xls")
```


```
    Do Until lcFile = ""
        Set loWorkBook = Workbooks.Open(lcPath & lcFile)
        For i = 1 To Application.VBE.VBProjects.Count
            Set loProject = Application.VBE.VBProjects(i)
            If loProject.Filename = loWorkBook.FullName Then
                For Each loVBComponent In loWorkBook.VBProject.
VBComponents
                    If loVBComponent.Type = vbext_ct_StdModule Then
                        If ModuleHasSub(loVBComponent,
pcSubName) Then
                            With loVBComponent.CodeModule
                                lnLineStart = .ProcBodyLine
(pcSubName, vbext_pk_Proc)
                                lnLineCount = _.ProcCountLines
(pcSubName, vbext_pk_Proc) .DeleteLines lnLineStart, lnLineCount

                                lnLine = 0
                                Open lcPath & "\改版过后的Sub.bas" _
                                For Input As #1
                                Do While Not EOF(1)
                                    Line Input #1, lcCode
                                    .InsertLines lnLineStart +
lnLine, lcCode
                                    lnLine = lnLine + 1
                                Loop
                                Close #1
                            End With
                        End If
                    End If
                Next
            End If
        Next

        loWorkBook.Close (True)
        lcFile = Dir
    Loop
```

```
End Sub

Public Function ModuleHasSub(poVBComponent As VBComponent,
ByVal pcSubName As String) As Boolean
    Dim c As Integer
    With poVBComponent.CodeModule
        For c = 1 To .CountOfLines
            If .ProcOfLine(c, vbext_pk_Proc) = pcSubName Then
                ModuleHasSub = True
                Exit Function
            End If
        Next
        ModuleHasSub = False
    End With
End Function

Public Sub ReplaceMacro()
    ReplaceSub ("RunMacro")
End Sub
```

注意，此处说的是“写入”而非“导入”，用导入的话，会自成一个模块，但我们的需求是写在原先的模块内，如果用导入而自成一个模块的话，原先调用该模块的程序就会因为找不到而出错（因为跑到新模块去了）。

执行此程序之前，请在“工具/引用”中引用“Microsoft Visual Basic for Applications Extensibility *.*”，“*.*”指的是版本编号，引用最新的版本即可，不然 vbext_ct_StdModule 的值将会是未定义的，如此，程序不会出错，但执行结果也不会正确。此外，上例“综合技巧范例_导出宏并删除.xls”则是要为所有“要被导出的宏的 xls 文件”都加上这个引用的动作，不然也会发生程序执行无误但却不会删除宏的现象。

此程序只会去寻找“模块”内的宏并进行改版，工作簿或工作表内的宏则不在寻找之列。如果也想把它们纳入进来，就再加上一个过滤条件，把：

```
    If loVBComponent.Type = vbext_ct_StdModule Then
```

改成：

```
    If loVBComponent.Type = vbext_ct_StdModule Or loVBComponent.
Type = vbext_ct_Document Then
```

不过，经笔者反复测试结果，纳入工作簿或工作表内的宏后，再以 InsertLines 方法写入程序代码时会有死机的情形发生。

由于 ProcBodyLine 与 ProcCountLines 这两个方法在遇到不存在的 SubName 时会发生执行错误，而用 On Error 把这种错误规避掉并不是个好习惯（除非真的不得已，尽量少用 On Error Resume Next），所以我们设计了一个简单的测试函数 ModuleHasSub，利用 ProcOfLine 方法逐一检查每行所在的 SubName 是否为传进来的 pcSubName 参数，如果模块内真有名为 pcSubName 的 Sub，那么迟早会被遇到而传回 True。

在被改版前的模块内，将 SubName 所在的开始行号与所占行数之间的程序代码删掉后，必须要在原来的地方写入（插入，所以叫 InsertLines）新版的宏，所以用了 lnLineStart + lnLine 的变量记载目前要写入的行号，这是因为在第 N 行插入程序代码造成的效果是“往下插”而非“往上插”，所以 lnLine 必须一直累加 1，不然就会一直在第 lnLineStart 这行插入，结果会把新版的宏程序

代码整个倒过来写了：

```
    End Sub
    MsgBox ("改版过后的 RunMacro")
    Public Sub RunMacro()
```

程序执行的结果，会把范例“这些工作簿的宏要被改版”目录底下的那 4 个 xls 文件里的“RunMacro”宏，改版成同目录下的“改版过后的 Sub.bas”文件。

范例 4-31　发送 E-mail

由于录制宏时 Email 这个动作无法录下来，所以我们也不知道以程序的方式该如何写。工作簿的 SendMail 方法固然可以用，但你的计算机上必须安装了某个“邮件管理系统”，也就是，在实时运算窗口中（在 Excel 按 Alt+F11 键后再按 Ctrl+G 键），？ xlNoMailSystem 必须传回一个不是 0 的数值才可以。

Windows 2000（含）以后的版本有个内置的 CDO 对象，用它发送 E-mail 的代码如下。

综合技巧范例_寄封 E-mail.xls

```
Public Sub SendEmail()
    Dim loCDO As CDO.Message
    Dim lcUserID, lcPassword, lcMailServer As String
    Dim loFile As Object
    Dim lcAttachmentFilename As String

    lcUserID = "myUserID"
    lcPassword = "myPassword"
```


```
    lcMailServer = "myMailServer"

    Set loCDO = New CDO.Message
    Set loFile = CreateObject("Scripting. FileSystemObject")

    loCDO.Configuration.Fields.Item(cdoSendUsingMethod) =
cdoSendUsingPort
    loCDO.Configuration.Fields.Item(cdoSendUserName) = lcUserID
    loCDO.Configuration.Fields.Item(cdoSendPassword) = lcPassword
    loCDO.Configuration.Fields.Item(cdoSMTPServer) = lcMailServer
    loCDO.Configuration.Fields.Update

    ThisWorkbook.Save
    lcAttachmentFilename = "c:\" & ThisWorkbook.Name
    loFile.CopyFile ThisWorkbook.FullName, "c:\"

    loCDO.AddAttachment lcAttachmentFilename, lcUserID,
lcPassword
    loCDO.From = lcUserID & "@" & lcMailServer
    loCDO.To = lcUserID & "@" & lcMailServer
    loCDO.To = "mywww.idv@msa.hinet.net"
    loCDO.Subject = "Test Subject"
    loCDO.TextBody = "内容" & vbCrLf &
"http://www.microsoft.com"
    loCDO.Send

    Set loCDO = Nothing
    Set loFile = Nothing

    Kill lcAttachmentFilename
End Sub
```

以上程序中的账号、密码、邮件服务器均为虚设的，请自行设置成你的账号、密码、邮件服务器（例如 msa.hinet.net）。

范例 4-32　抓股市收盘资料

这是另一个常见的应用，我常收到 E-mail 询问此问题。所以在本书再版时，特别新增此小节。

由于本书作者对台湾的股市比较熟悉，本节就以台湾为例，先分析网址结构。台湾的股市分为上市与上柜，归不同机关管理，网址如下：

上市：
```
http://www.tse.com.tw/ch/trading/exchange/MI_INDEX/genpage/
Report200603/A112*20060329*ALL_1.php?select2=ALL&chk_date=95/03/29
```
上柜：
```
http://otcstk.otc.org.tw/info/RSTA3104/RSTA3104_950329.HTML
```

以斜体字呈现的，就是网址结构所在，它显然与日期相关，这很合理，因为股市是以日期为运作单位。在设计自动抓股市收盘价的程序时，需要考虑以下因素：

星期假日不抓，不用说，一定是未开盘。

未开盘日不抓，例如春节、端午、中秋、以及其他特殊假日等。但我们无法知道每年的这些日期是在何时，所以一个简单的办法是，抓到的网页没有收盘信息，就是未开盘日了。如何知道抓到的网页没有收盘信息呢？这就回到了之前所讲的一个重要原则：自动抓网页数据的程序，必须针对该网页的结构而设计。

台湾股市收盘价的网页，其格式都非常固定，但每个人所看到的这种“固定性”会有所不同。笔者的原则是，找个最简单的“固定性”，反正到时网页有改时，再改写程序就是了，一个网页不论怎么改，固定性都很容易看出，尤其是这种跟数据库相关的

网页，大都是以程序产生，既然是以程序产生，也往往很容易以程序抓取。

上市收盘价的固定性是什么？如果所抓取网页有个表格（table）的行数超过 300，那个表格就是上市收盘信息了。

上柜收盘价的固定性又是什么呢？如果所抓取网页有个表格（table）的行数超过 100，那个表格就是上柜收盘信息了。

分析完网页结构后，利用 IE 对象，设计程序如下：

综合技巧范例_删除空白工作表.xls

```
Public Sub 抓股市收盘价_上市()
    Cells.Clear
    Application.ScreenUpdating = False

    If Weekday(Date) = 1 Or Weekday(Date) = 7 Then
        Exit Sub
    End If

    lc_date = Format(Date, "yyyy/mm/dd")
    year_month = Mid(lc_date, 1, 4) + Mid(lc_date, 6, 2)
    year_month_day = Mid(lc_date, 1, 4) + Mid(lc_date, 6, 2)
+ Mid(lc_date, 9, 2)
    year_month_day_taiwan = Trim(Str(Year(lc_date) - 1911)) +
"/" + Mid(lc_date, 6, 2) + "/" + Mid(lc_date, 9, 2)
    lc_url = "http://www.tse.com.tw/ch/trading/exchange/MI_INDEX/
genpage/Report" + year_month + "/A112" + year_month_day +
"ALL_1.php?select2=ALL&chk_date=" + year_month_day_taiwan

    Set ie = CreateObject("internetexplorer.application")
    ie.Navigate (lc_url)
    While ie.readystate <> 4 Or ie.busy
    Wend
    rn = 0
```

```
    Set all_tables = ie.document.getElementsByTagName("table")
    For i = 0 To all_tables.Length - 1
        Set one_table = all_tables.Item(i)
        If one_table.className = "board_trad" And one_table.
Rows.Length > 300 Then
            Set Row = one_table.Rows(1)
            content = Trim(Row.Cells(0).innerText)
            rn = rn + 1
            Set Cellss = Row.Cells
            For c = 0 To Cellss.Length - 6
                content = Replace(Replace(Trim(Cellss.Item(c).
innerText), ",", ""), " ", "")
                Cells(rn, c + 1) = content
            Next

            For r = 2 To one_table.Rows.Length - 1
                Set Row = one_table.Rows(r)
                content = Trim(Row.Cells(0).innerText)
                If Len(content) = 4 And Val(Left(content, 1)) > 0 Then
                    rn = rn + 1
                    Set Cellss = Row.Cells
                    For c = 0 To Cellss.Length - 6
                        content = Replace(Replace(Trim(Cellss.
Item(c).innerText), ",", ""), " ", "")
                        Cells(rn, c + 1) = content
                    Next
                End If
            Next
            Exit For
        End If
    Next

    Range("A1") = "证券代号"

    Range("C2").Select
    Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
    Selection.NumberFormatLocal = "#,##0_ "
```


```
    Range("F2").Select
    Range(Selection,
ActiveCell.SpecialCells(xlLastCell)).Select
    Selection.NumberFormatLocal = "0.00_ "

    Cells.Select
    Cells.EntireColumn.AutoFit
    Range("A1").Select

    Application.ScreenUpdating = True
    ie.Quit
End Sub

Public Sub 抓股市收盘价_上柜()
    Cells.Clear
    Application.ScreenUpdating = False

    If Weekday(Date) = 1 Or Weekday(Date) = 7 Then
        Exit Sub
    End If

    lc_date = Format(Date, "yyyy/mm/dd")
    year_month_day_taiwan = Trim(Str(Year(lc_date) - 1911)) +
Mid(lc_date, 6, 2) + Mid(lc_date, 9, 2)
    lc_url = "http://otcstk.otc.org.tw/info/RSTA3104/
RSTA3104_" + year_month_day_taiwan + ".HTML"

    Set ie = CreateObject("internetexplorer.application")
    ie.Navigate (lc_url)
    While ie.readystate <> 4 Or ie.busy
    Wend

    rn = 0
    Set all_tables = ie.document.getElementsByTagName("table")
    For i = 0 To all_tables.Length - 1
        Set one_table = all_tables.Item(i)
```

```
        If one_table.Rows.Length > 100 Then
            Set Row = one_table.Rows(0)
            content = Trim(Row.Cells(0).innerText)
            rn = rn + 1
            Set Cellss = Row.Cells
            For c = 0 To Cellss.Length - 7
                content = Replace(Replace(Trim(Cellss.Item(c).
innerText), ",", ""), " ", "")
                Cells(rn, c + 1) = content
            Next

            For r = 1 To one_table.Rows.Length - 1
                Set Row = one_table.Rows(r)
                content = Trim(Row.Cells(0).innerText)
                If Len(content) = 4 And Val(Left(content, 1)) > 0 Then
                    rn = rn + 1
                    Set Cellss = Row.Cells
                    If Cellss.Length = 19 Then
                        For c = 0 To Cellss.Length - 7
                            content = Replace(Replace(Replace
(Replace(Replace(Trim(Cellss.Item(c).innerText), ",", ""), "
", ""), "⊕", ""), "-", ""), "⊙", "")
                            Cells(rn, c + 1) = content
                        Next
                    End If
                End If
            Next
            Exit For
        End If
    Next

    Range("A1") = "股票代号"
    Range("B1") = "证券名称"
    Range("C1") = ""
    Range("D1") = "收盘价"
    Range("E1") = ""
    Range("F1") = "涨跌"
```

```
    Range("G1") = "开盘价"
    Range("H1") = "最高价"
    Range("I1") = "最低价"
    Range("J1") = "均价"
    Range("K1") = "成交股数"
    Range("L1") = "成交金额(元)"
    Range("M1") = "成交笔数"

    Columns("A:A").ColumnWidth = 100
    Columns("C:C").Delete Shift:=xlToLeft

    Range("C2").Select
    Range(Selection,
ActiveCell.SpecialCells(xlLastCell)).Select
    Selection.NumberFormatLocal = "0.00_ "

    Range("J2").Select
    Range(Selection, ActiveCell.SpecialCells(xlLastCell)).Select
    Selection.NumberFormatLocal = "#,##0_ "

    Cells.Select
    Cells.EntireColumn.AutoFit
    Range("A1").Select

    Application.ScreenUpdating = True
    ie.Quit
End Sub
```

本程序预设抓取当日收盘价，台湾股市交易时间到下午 1:30，所以本程序要在下午 1:30 之后执行才有东西跑出来。如果要改成抓一个自定义的日期，修改上述日期变量 lc_date 即可。例如：

```
    lc_date = Format(#2006/03/22#, "yyyy/mm/dd")
```

[General Information]
书名=Excel宏魔法书
作者=李潜瑞
页数=281
SS号=90114869
出版日期=2007年7月第1版