

# **EpiData3.0 使用手册**

## 声 明

本手册主要内容译自：Lauritsen JM & Bruus M. EpiData (version 3). A comprehensive tool for validated entry and documentation of data. The EpiData Association, Odense Denmark, 2003. 在此基础上，译者对部分内容的顺序作了调整，并增加了一些示范性图片，以及对某些操作的个人理解。

书稿中有不尽人意的地方或错误之处，诚恳地希望得到大家的谅解并不吝赐教。任何单位和个人，未经允许，不得摘录、复制本手册内容进行公开出版，违者必究。

编译：吕筠

北京大学公共卫生学院  
流行病学与卫生统计学系

2004.6

# 目 录

<b>1. 简介 (Introduction)</b> .....	<b>1</b>
1.1 EpiData 的历史.....	2
1.2 EpiData 与 Epi Info 的兼容性 .....	2
1.2.1 数据库.....	2
1.2.2 CHECK 语言 .....	3
<b>2. 建立调查表文件 (Create Questionnaire File)</b> .....	<b>3</b>
2.1 编辑器 (Editor) .....	3
2.1.1 自动缩进 (Auto Indent) .....	3
2.1.2 对齐变量 (Align Fields) .....	4
2.1.3 变量类型选择列表 (Field Pick List) .....	7
2.1.4 编码书写器 (Code Writer) .....	13
2.1.5 制表符@ (Tabulator Code @) .....	13
2.1.6 预览数据录入表格 (Preview Data Form) .....	16
2.2 定义变量名 (Field Names) .....	16
2.2.1 将第一个单词作为变量名 (First Word in Question is Field Name) .....	17
2.2.2 自动定义变量名 (Automatic Field Names) .....	19
2.2.3 变量标签 (Variable Labels) .....	20
<b>3. 数据库的建立和修改 (Create and Revise Data File)</b> .....	<b>20</b>
3.1 创建数据库 (Create Data File) .....	20
3.2 修改数据库 (Revise Data File) .....	21
3.3 重新定义变量名 (Rename Fields) .....	23
<b>4. 建立核查文件 (Check File)</b> .....	<b>24</b>
4.1 添加/修改核查命令 (Add/Revise Checks) .....	24
4.1.1 基本操作.....	24
4.1.2 基本 CHECK 命令的设置.....	27
4.1.3 编辑当前变量的所有 CHECK 语句 .....	32
4.2 CHECK 的文件结构 .....	32
4.3 实例.....	34
4.4 CHECK 命令列表.....	38
*.....	38
AFTER ENTRY .....	38
AFTER FILE.....	39
AFTER RECORD.....	39
AUTOJUMP .....	39
AUTOSAVE.....	40
AUTOSEARCH.....	40
BACKUP .....	41
BEEP.....	41
BEFORE ENTRY .....	41
BEFORE FILE.....	42
BEFORE RECORD .....	42
CLEAR .....	42

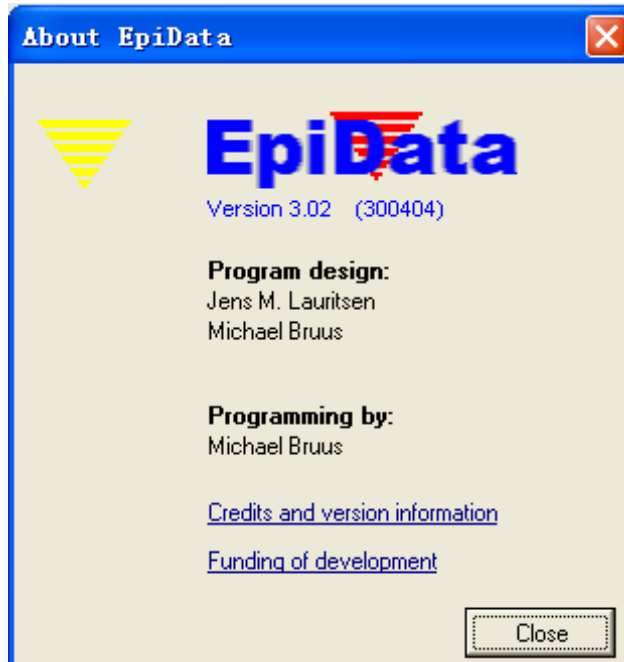
CLEAR COMMENT LEGAL .....	42
COLOR .....	42
COMMENT (*).....	44
COMMENT LEGAL .....	44
CONFIRM .....	47
CONFIRMFIELD .....	47
COPYTOCLIPBOARD .....	47
DEFINE .....	47
EXECUTE .....	48
EXIT .....	49
GOTO.....	50
HELP.....	51
HIDE, UNHIDE.....	52
IF... THEN .....	52
INCLUDE .....	54
JUMPS .....	54
KEY .....	55
LABEL.....	56
LABELBLOCK .....	56
LEGAL .....	56
LET .....	57
MISSINGVALUE .....	58
MUSTENTER.....	58
NOENTER .....	59
QUIT .....	59
RANGE.....	59
RELATE.....	59
REPEAT .....	60
TOPOFSCREEN.....	60
TYPE.....	60
TYPE COMMENT .....	61
TYPE STATUSBAR .....	62
UNHIDE .....	63
WRITENOTE .....	63
4.5 运算符和函数.....	64
4.5.1 运算符 (Operators) .....	64
4.5.2 函数 (Functions) .....	65
<b>5. 录入数据 (Enter Data) .....</b>	<b>71</b>
5.1 在变量间转换.....	72
5.2 在记录间转换.....	72
5.3 在关联数据库间转换.....	72
5.4 查找记录.....	73
5.5 查找变量和关联变量.....	74
5.6 滤过 (Filter) .....	75

5.6.1 设置滤过规则.....	75
5.6.2 解除滤过.....	75
<b>6. 数据库的管理和维护 .....</b>	<b>75</b>
6.1 数据库的追加与合并 (Append/Merge Data Files) .....	75
6.1.1 数据库的追加 (Append) .....	75
6.1.2 数据库的合并 (Merge) .....	77
6.2 双录入核查 (Double Entry and Validation) .....	79
6.2.1 选择关键变量.....	80
6.2.2 选项设置.....	81
6.3 逻辑一致性核查 (Logical Consistency Check) .....	81
REPORT.....	82
CHECKRANGE .....	82
CHECKLEGAL .....	82
CHECKRANGELEGAL .....	82
CHECKMUSTENTER .....	82
6.4 根据数据库创建 QES 文件 (Make QES File from Data File) .....	82
6.5 重新编码数据库 (Recode Data File) .....	82
6.6 将年的表示方式从 2 位数转换为 4 位数.....	84
6.7 打包数据库 (Pack Data File) .....	84
6.8 压缩数据库 (Compress Data File) .....	84
6.9 打印数据录入表格 (Print Data Entry Form) .....	85
6.10 将数据库存档 (Archive) .....	85
6.11 数据库相关信息的管理.....	86
6.11.1 数据库结构 (File Structure) .....	86
6.11.2 数据录入备忘录 (Data Entry Notes) .....	87
6.11.3 数据库标签 (Data File Label) .....	87
6.11.4 浏览数据 (View Data) .....	87
6.11.5 数据列表 (List Data) .....	88
6.11.6 基本的统计表格 (Codebook) .....	90
6.11.7 分变量统计记录数 (Count Records by Field) .....	91
<b>7. 数据库的输出和输入 .....</b>	<b>93</b>
7.1 数据库的输出 (Export Data) .....	93
7.1.1 备份数据库 (Backup of Data) .....	93
7.1.2 输出到文本文件 (Export to Text File) .....	93
7.1.3 输出到 dBaseIII (Export to dBaseIII format) .....	95
7.1.4 输出到 Excel (Export to Excel) .....	95
7.1.5 输出到 SPSS (Export to SPSS) .....	95
7.1.6 输出到 SAS (Export to SAS) .....	96
7.1.7 输出到 Stata (Export to Stata) .....	96
7.1.8 输出到新的 EpiData 数据库 (Export to New EpiData Data File) .....	96
7.2 数据库的输入 (Import Data) .....	96
7.2.1 从文本文件输入 (Import Text Files) .....	96
7.2.2 从 dBase 文件输入.....	98
7.2.3 从 Stata 文件输入.....	98

<b>8. 其它</b> .....	<b>99</b>
8.1 选项设置 (Options) .....	99
8.1.1 编辑器选项 (Editor Options) .....	99
8.1.2 数据录入表格选项 (Show Data Form Options) .....	100
8.1.3 创建数据库选项 (Create Data File Options) .....	101
8.1.4 输出报表选项 (Documentation Options) .....	101
8.1.5 高级选项 (Advanced Options) .....	102
8.1.6 文件关联 (File Associations) .....	103
8.2 其它.....	104
8.2.1 .INI 文件 (The .INI File) .....	104
8.2.2 程序参数 (Program Parameters) .....	104
8.2.3 语言包.....	105
8.2.4 数据库结构 (Datafile Structure) .....	105
8.2.5 快捷方式/鼠标 (Short-Cut Keys/Mouse) .....	109
8.2.6 主要菜单功能索引.....	111

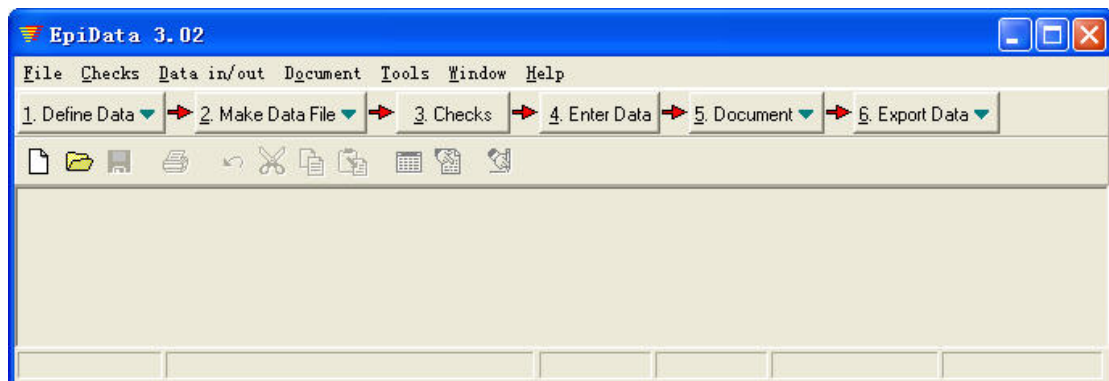
## 1. 简介 (Introduction)

EpiData 是一个免费的数据录入和数据管理软件。开发者是丹麦欧登塞 (Odense, Denmark) 的一个非盈利组织, 即 The EpiData Association (<http://www.epidata.dk/>)。程序设计者为 Jens M. Lauritsen, Michael Bruus 和 Mark Myatt。



该软件目前有多种语言版本, 如丹麦语、挪威语、荷兰语、意大利语、中文、法语、西班牙语、俄语、斯洛文尼亚语、塞尔维亚语、波兰语、葡萄牙语、阿拉伯语、德语、罗马尼亚语、英语等。

EpiData 的工作原理源自 DOS 版本的 Epi Info 6, 但是工作界面为 Windows 版。EpiData 的安装、运行不会依赖系统文件夹中的任何文件, 也不会你的系统文件夹中安装或替代任何 DLL 文件。程序设置等参数被保存在 EpiData.ini 的文件中。你可以通过 `setup.exe` 在计算机中安装这个程序; 也可以直接拷贝 EpiData.exe 文件到计算机中, 同样可以运行。



理论上, 该程序对录入的记录数没有限制。而实际应用中, 记录数最好不要

超过 200,000~300,000（曾经用 250,000 测试过）。整个录入界面不能超过 999 行。对数值或字符串编码进行解释的文字长度最多 80 个字符，编码长度最多为 30 个字符。

## 1.1 EpiData 的历史

EpiData 的研发工作最早由丹麦的 Jens M. Lauritsen 发起。最初是作为 Funen 县开展的“预防意外伤害行动”（Initiative for Accident Prevention）中的一部分。但是为什么会着手研发一种新的数据录入程序呢？

Epi Info 6 具备所有我们需要的数据录入和管理功能。但是，随着视窗（Windows）程序的发展，很多使用者发现很难应付 1990~1995 年开发出的 Epi Info 的 DOS 模块。而商业化的程序一般不针对数据的管理，使用起来也没有那么简单，也不具备双录入核查的功能。

美国 CDC 的 Epi Info 工作组将 Epi Info 6 更新到 Epi Info 2000。更新的 Epi Info 2000 采用了一种全新的工作策略，数据库不再采用过去简单的 ASCII 格式，而是启用 Access 数据库格式。

1999 年末，Jens M. Lauritsen、Mark Myatt 和 Michael Bruus 组成研发小组。Michael Bruus 是一位专业的 Pascal（帕其卡语言，一种高水平的计算机编程语言）程序设计师，编程工作主要由他完成。工作小组希望将 EpiData 开发成为一个简单、易用、独立的应用程序。这个程序不需要任何专门的数据库系统驱动（基于 dll）。同时，他们希望从社会团体、个人、以及其他捐助者那里得到资金支持，这样，可以免费发布这个软件。

## 1.2 EpiData 与 Epi Info 的兼容性

EpiData 的操作原理是基于 MSDOS 版本的 Epi Info v6.xx。在开发 EpiData 时，基本的原则就是要保证 EpiData 创建的文件能够与 Epi Info 兼容，反之亦然。不过，两个程序在一些变量的类型上还是存在一些差异。基本上，EpiData 和 Epi Info v6 非常近似。很多 Epi Info v6.xx 创建的数据库基本不需要什么调整就可以用于 EpiData，特别是当 Epi Info v6.xx 中只设置了简单的 CHECK 命令（例如，允许值、重复、必须录入、跳转等）时。下面简单介绍一下 EpiData 与 Epi Info 的差别。

### 1.2.1 数据库

#### 1. 在 EpiData 中运行 Epi Info 数据库：

EpiData 不支持下面几种 Epi Info 的变量类型：

- 电话号码
- 电话分机号码
- EpiData 忽略背景和单一录入变量的颜色编码

#### 2. 在 Epi Info 中运行 EpiData 数据库：

Epi Info 不支持下面几种 EpiData 的变量类型：

- 欧式当天日期<Today-dmy>
- 日期格式<yyyy/mm/dd>和<Today-dmy>
- 声索引（soundex）变量
- 制表符（@）
- 背景和录入变量等的颜色编码，EpiData 不保存这些设置



你可以从“[变量类型](#)”一节了解 EpiData 支持的所有变量类型。

## 1.2.2 CHECK 语言


■ IF...THEN 中，必须用圆括号将每个条件语句括起来。例如：

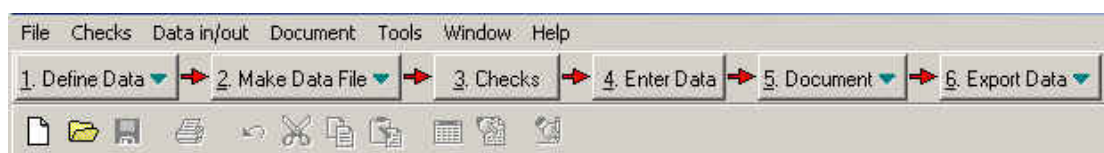
```
IF(a=2) AND (b>3) THEN...
```

另外，EpiData 在一些计算和表达式中的语句略有不同。

- EpiData 的 CHECK 语言有了进一步的扩展，含有许多 Epi Info v6 不支持的 CHECK 功能。
- HELP 命令的语句有所不同。
- EpiData 忽略了一些命令（例如，TYPE、HELP）中对颜色编码和屏幕坐标的设置
- EpiData 中的日期变量必须是 10 位数的欧式日期格式。例如：10/02/2001。
- EpiData 中不支持 Codefield 和 codes；取而代之的是 COMMENT LEGAL 和 TYPE COMMENT，具有相同的功能。
- Epi Info 不支持 QUIT 和 COPYTOCLIPBOARD 命令。

## 2. 建立调查表文件（Create Questionnaire File）

建立调查表文件是建立数据库、实现数据录入和管理的第一步。你可以点击菜单中的 **File**→**New**，或者在工作流程栏（Work Process Toolbar）上点击“**1. Define Data**”，或者点击编辑器工作栏（Editor Toolbar）上的 ，这时窗口中会自动显示一个空白的文档，你可以在这里键入你的调查表，实际上也就是数据录入表格的框架。编辑完成后，将此调查表文件保存，文件的扩展名统一为.QES。

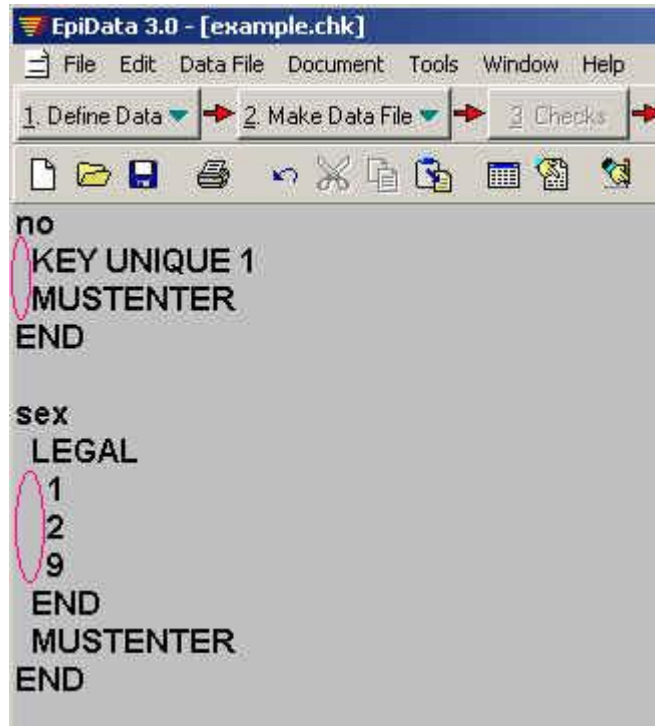


### 2.1 编辑器（Editor）

EpiData 编辑器的主要功能是创建和编辑调查表（.QES 文件）。当然，我们也可以使用编辑器处理程序输出的报表，以及编辑 CHECK 文件。编辑器的使用和其它字处理软件基本一致。下面，我们会对编辑器中一些特殊的功能选项进行介绍。

#### 2.1.1 自动缩进（Auto Indent）

如果勾选上这个功能（**Edit**→**Auto Indent**），在键入调查表、回车换行时，文档会自动缩进，像前一行一样空出相同的空格。这个功能特别适用于在编辑器中编辑 CHECK 文件。

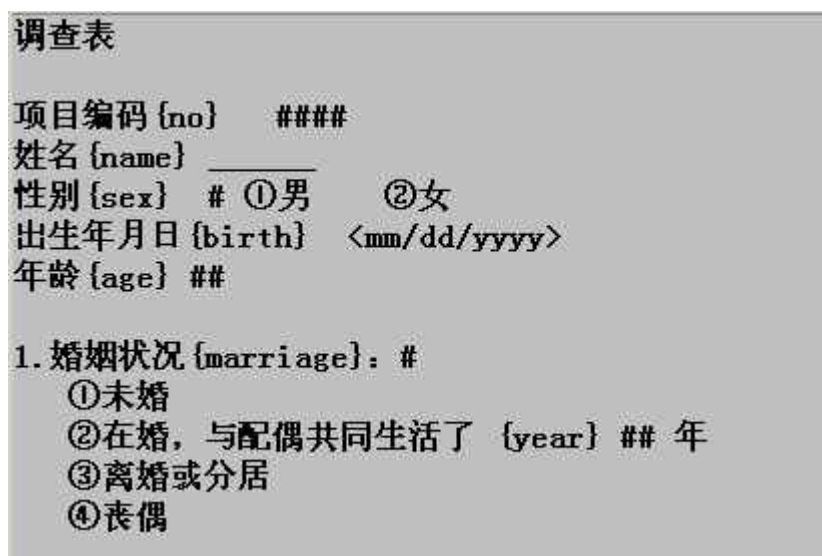


### 2.1.2 对齐变量 (Align Fields)

如果希望其它变量都向某个变量对齐, 则将光标放在该变量行上, 点击 **Edit** → **Align Fields**。因 **变量命名** 的设置 (见 **File** → **Options** → **Create data file: How to generate field names**) 不同, 运行该功能会产生不同的效果。

1. 选择“将第一个单词作为变量名”, 即: ① [First word in question is field name](#)

下面这个调查表, 在未使用“对齐变量”功能前, 如图所示。



让我们先把光标放在“项目编号”一行, 点击 **Edit** → **Align Fields**, 调查表如下

显示:

```
调查表
项目编码 {n o} #####
姓名 {name}
性别 {sex} # ①男 ②女
出生年月日 {birth} <mm/dd/yyyy>
年龄 {age} ##

1. 婚姻状况 {marriage}: #
  ①未婚
  ②在婚, 与 配偶共同生活了 {year} ## 年
  ③离婚或分居
  ④丧偶
```

可见, 座标位置在“项目编码”变量“#####”左边的变量, 其变量编码均向“项目编码”的“#####”对齐。而变量编码左侧的注释性文字仍然左对齐。下面再试一试, 把光标放在“婚姻状况”一行, 点击 **Edit**→**Align Fields**, 调查表如下显示:

```
调查表
项目编码 {n o} #####
姓名 {name}
性别 {sex} # ①男 ②女
出生年月日 {birth} <mm/dd/yyyy>
年龄 {age} ##

1. 婚姻状况 {marriage}: #
  ①未婚
  ②在婚, 与 配偶共同生活了 {year} ## 年
  ③离婚或分居
  ④丧偶
```

这回则都是以“婚姻状况”的变量编码“#”对齐。再把光标放在“与配偶共同生活##年”一行, 操作后显示如下:

```

调查表

项目编码 {no} #####
姓名 {name} _____
性别 {sex} # ①男    ②女
出生年月日 {birth} <mm/dd/yyyy>
年龄 {age} ##

1. 婚姻状况 {marriage}: #
  ①未婚
  ②在婚, 与配偶共同生活了 {year} ## 年
  ③离婚或分居
  ④丧偶

```

2. 选择“自动定义变量名”，即②[Automatic field names](#)  
 依然是上面那个例子，如果把光标放在“项目编码”一行，点击 **Edit** → **Align Fields**，调查表如下显示：

```

调查表

项目编码 {no} #####
  姓名 {name} _____
  性别 {sex} # ①男    ②女
  出生年月日 {birth} <mm/dd/yyyy>
  年龄 {age} ##

1. 婚姻状况 {marriage}: #
  ①未婚
  ②在婚, 与配偶共同生活了 {year} ## 年
  ③离婚或分居
  ④丧偶

```

可见，座标位置在“项目编码”变量“#####”左边的变量，其变量编码均向“项目编码”的“#####”对齐。而变量编码左侧的注释性文字随之缩进。下面再试一试，把光标放在“婚姻状况”一行，运行后如下显示：

```

调查表

    项目编号 {no} #####
    姓名 {name} _____
    性别 {sex} # ①男    ②女
    出生年月日 {birth} <mm/dd/yyyy>
    年龄 {age} ##

1. 婚姻状况 {marriage}: #
①未婚
②在婚, 与配偶共同生活了 {year} ## 年
③离婚或分居
④丧偶

```

再把光标放在“与配偶共同生活##年”一行，操作后显示如下：

```


调查表

    项目编号 {no} #####
    姓名 {name} _____
    性别 {sex} # ①男    ②女
    出生年月日 {birth} <mm/dd/yyyy>
    年龄 {age} ##

    1. 婚姻状况 {marriage}: #
①未婚
②在婚, 与配偶共同生活了 {year} ## 年
③离婚或分居
④丧偶

```

### 2.1.3 变量类型选择列表 (Field Pick List)

点击菜单 **Edit** → **Field Pick List**，或者同时按 **Ctrl+Q** 键，或者在编辑器工作栏 (Editor Toolbar) (在菜单空白处点击右键可选，或菜单 **Window** → **Toolbars** 中可选) 上点击  图标，可以打开变量类型选择列表，从中选择适当的变量类型插入即可。下面介绍一下，EpiData 中允许的变量类型。

变量类型	变量编码
ID 号	<IDNUM>
数值型	### ###.##
字符型	_____
	<E >
大写字母字符型	<A>, <A >
布尔逻辑变量	<Y>

变量类型	变量编码
日期	<dd/mm/yyyy> <mm/dd/yyyy> <yyyy/mm/dd>
当天日期	<today-dmy> <today-mdy> <today-ymd>
声索引变量	<S> <S >
制表符	@

## 数值型变量 (Numeric Fields)

###  
###.###  
#####  
##.####

数值型变量允许录入数字、减号和小数点。在 QES 文件中和数据录入过程中，你可以用圆点 (.) 或逗号 (,) 来表示小数点。一个变量中只允许输入一个小数点，这意味着，你不能用逗号作为千位的分隔符（例如：1,000,000）。字符“#”的数目表示变量的长度，小数点占一位字符。变量最长允许 14 个字符。如果人为将变量长度加到最多 18 位字符也是允许的。但是，字符长度为 17（或 18 位）时，录入的最后 1 位（或 2 位）会自动更改，无法满足录入要求。



## 字符型变量 (Text Fields)

—  
\_\_\_\_\_  
\_\_\_\_\_

下划线字符的数目表示变量的长度。字符型变量允许输入所有字符。变量最长允许 80 个字符。如果输入中文，请注意，一个中文字需占用 2 个字符。



## 大写字母的字符型变量（Upper-case Text）

<A>  
<A >

大写字母的字符型变量中可以录入任意字符，但程序会自动将录入的字母转换为大写。变量的长度即“<”和“>”间的字符数，其中包括大写字母“A”所占的1个字符。上面例子中，第一个变量的长度为1，第二个变量的长度为5。

## 加密变量（Encrypted Fields）

<E >

加密变量是一种特殊的字符型变量。加密变量的内容以可读的形式显示在屏幕上，但以密码形式保存在磁盘中。使用的运算法则是被称作 Rijndael AES 的超强加密。如果你想了解更多的这方面的内容，可查阅网站：

<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>和  
<http://csrc.nist.gov/CryptoToolkit/aes/rijndael/>。

千万别忘了你设置的密码，因为根据文件内容，你根本无法破坏或猜测密码。如果真的忘记了，你会丢失掉这些信息。利用加密变量，你可以实现对某些数据（如私人信息）的保护。

如果你建立的 QES 文件中含有一个或多个加密变量，当你在此基础上开始创建新的数据库时，程序会自动弹出对话框，询问你想设置的密码，该密码将被保存在数据库中。当对该数据库进行有关操作（例如，创建或编辑 CHECK 文件、打开数据库、输出数据库等）时，程序都会首先弹出一个对话框，只有输入了正确的密码，下一步操作才会继续。在 CHECK 文件中，加密变量可以按字符型变量处理。例如：

```
LET encrypt1="Howdy"
```

## 日期变量（Date Fields）

<dd/mm/yyyy>

<mm/dd/yyyy>

<yyyy/mm/dd>

EpiData v3中有三种类型的日期变量：欧式日期（日/月/年）、美式日期（月/日/年）和我们习惯的日期格式（年/月/日）。日期变量的长度通常是10个字符。在数据录入过程中，允许录入的字符包括数字和斜线（/）。如果你可以把日期数字按完整的格式输入，中间可以不加“/”。例如，1999年5月4日，对于欧式日期格式的变量，你可以键入04051999，当下一个变量被激活时，这个日期变量会自动变为标准的格式（04/05/1999）。如果键入不全，则年月日间要加“/”，例如上述日期可以这样键入：4/5/1999。

EpiData只支持用4位数表示年。但录入的时候，不一定要键入10个数字。如果在欧式日期格式下，录入040599，程序会自动将其转换为04/05/1999。用2位数表示“年”时，程序会把50~99默认为20世纪，即1950~1999；而把00~49默认为21世纪，即2000~2049。如果在欧式日期格式下，仅输入0405，则当前的年份会被自动加入。如当前是2003年，则该变量自动显示为04/05/2003。



### 当天日期变量（Today's Date Fields）

<today-dmy>

<today-mdy>

<today-ymd>

程序会用当天日期（即计算机的系统日期）自动填充这个变量。该型变量不允许使用者录入，不会被激活。如果现在编辑一个以前保存过的记录，该记录中含有一个“当天日期”变量，当修改了的记录再次被保存时，修改当天的日期会自动更新“当天日期”变量。需要注意的是，该变量中dmy和ymd两种格式是在EpiData中才出现的，Epi Info中没有，这一点是不兼容的。

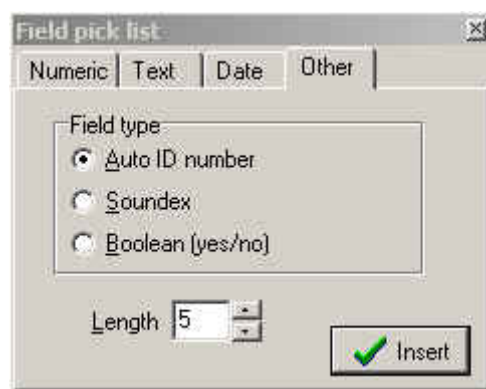
### ID 号（ID Number）

<IDNUM>

<IDNUM >



IDNUM 是一个能够自动生成 ID 号的变量，每录入一条新记录，ID 号会自动加 1。在数据录入过程中，ID 号不能修改。在一个新数据库中，ID 号默认从 1 开始，不过你可以在 **File**→**Options**→**Advanced**→**ID number fields**→**First ID number in new data file:1** 中修改起始编码。此变量长度为 5-14 个字符。



### 布尔逻辑变量（是/否变量）（Boolean Fields, Yes/No Fields）

<Y>

布尔逻辑变量只允许录入 Y、N、1、0。录入“1”，程序会自动将其转换为“Y”，录入“0”，则自动转换为“N”。布尔逻辑变量的长度仅为 1。因此，如果你在 QES 文件中输入 <Y >，会出现错误。

### 声索引变量（Soundex Fields）

<S >

<S >

该变量用于对被调查者的“姓”进行重新编码。该功能主要适用于英语语系国家。程序自动将输入的被调查者的“姓”转换为一个索引字母和 3 位数字的编码。索引字母就是“姓”的第一个字母。3 位数字是根据“姓”中余下的字母计算出来的。这项功能的优点是将被调查者的“姓”按发音归类，而不是按照确切的拼写进行归类。这样做可以保护被调查者的隐私，研究者无法从发音编码中确定被调查者的姓名，因此不会暴露被调查者。而对于研究者来说，根据这个声索引变量，加上人口学数据（如出生日期、性别等），仍然可以判断出哪些记录是重复报告的，不会影响分析。

录入过程中，我们可以在这个变量上输入任何字符，但是如果字符间有空格的话，程序只会用录入的最后一个单词去创建声索引变量。例如，输入“My surname is Wang”，而程序只会用最后一个词“Wang”去创建声索引变量。

声索引变量的格式为一个大写字母、一个连字符、加上 3 个数字，例如：A-999。创建声索引变量的原则如下：

1. 保留“姓”中的第一个字母，余下的字母会被转换为 3 位数字，编码的原则为：

A E I O U Y H W	不编码
B F P V	编码为1
C G J K Q S X Z	编码为2
D T	编码为3
L	编码为4
M N	编码为5
R	编码为6

2. 第一个字母后面的辅音字母被依次编码:

H O L M E S = H-452  
- 4 5 - 2

A D O M O M I = A-355  
3 - 5 - 5 -

3. 编码中只含第一个字母, 加上3位数字, 之后多出来的字母将被忽略:

V O N D E R L E H R = V-536  
- 5 3 - 6 - - -

4. 如果可转换的字母数不够, 可以用0填充:

B A L L = B-400  
- 4 -

S H A W = S-000  
- - -

5. 连续重复的字母, 只转换一个:

B A L L = B-400  
- 4 -

6. 相邻的、同属一个编码组的字母, 只转换前一个:

J A C K S O N = J-250  
- 2 - - - 5

7. 如果第二个字母与第一个字母同属一个编码组, 则忽略第二个字母:

S C A N L O N = S-545  
- - 5 4 - 5

8. 忽略单引号 (') 和连字符 (-):


K I N G - S M I T H = KINGSMITH = K-525  
- 5 2 - - 5 - - -

O ' N E I L = O-540

9. 来自相同编码组、中间以W或H相隔的字母，只编码前一个：

B O O T H - D A V I S = B-312  
 - - 3 - - - - 1 - 2

### 2.1.4 编码书写器 (Code Writer)

在输入变量时，借助于编码书写器可以很方便的实现对变量类型和长度的定义。点击菜单 **Edit**→激活 **Code Writer**，或者同时按 **Ctrl+W** 键，或者在编辑器工作栏 (Editor Toolbar) 中点击  图标。这时，按某些键，就可以开始进行变量的定义，程序会自动完成编码，或者主动询问你有关变量长度的信息。

例如，当你键入字符#，程序会认为你要键入数值型变量，随后弹出一个对话框，询问数值型变量的长度。当你键入需要的长度后，在光标当前所在位置上会自动插入相应长度的数值型变量的编码。下表中列出了编码书写器识别的一些字符串组合。

#	数值型变量 程序会询问你变量的长度 键入 5，会得到一个有 5 位数字的整数变量 (#####) 键入 5.2 或 5,2，得到的变量为：小数点前 5 位数，小数点后 2 位数 (#####.##)
_ (下划线)	字符型变量 程序会询问你变量的长度
<A	大写英文字母的字符型变量 程序会询问你变量的长度
<d	插入欧式日期<dd/mm/yyyy>
<m	插入美式日期<mm/dd/yyyy>
<y	插入布尔逻辑变量<Y>
<i	插入自动编码的 ID 号 程序会询问你变量的长度 默认的长度是 5 个字符 (即允许的最小长度)
<s	声索引变量 程序会询问你变量的长度

### 2.1.5 制表符@ (Tabulator Code @)

在数据录入的表格中，各个变量录入框的位置取决于变量前解释性文字的长度。由于各行解释性文字长度不一，使得前后变量很难对齐，观感较差。手动对齐又浪费时间。这时，我们可以在编辑 QES 文件时利用制表符@ (Epi Info 中不存在这种使用方法)，实现前后变量的对齐。

使用制表符@，只会改变变量录入框在表格中的位置，而不会对变量或 REC 数据库产生其它不良影响。在变量编码前插入@，可以将这个变量对齐到下一个制表位置。每个制表位置间的间隔可以在 **File**→**Options**→**Show data form**→**Tabs/Indents**→Tab stop in data form every  pixels 中进行设置。默认值为

40。注意，插入的@与变量编码间不能有空格。另外，请确认 **File**→**Options**→**Create data file**→**How to generate field names**→选择 Automatic field names。只有在这种设置情况下，才能实现制表符@的这项功能。例如：

```

调查表

项目编码 {no}@####
姓名 {name}@_____
性别 {sex}@# ①男 ②女
出生年月日 {birth}@<mm/dd/yyyy>
年龄 {age}@##

1. 婚姻状况 {marriage}: @#
  ①未婚
  ②在婚，与配偶共同生活了 {year}@## 年
  ③离婚或分居
  ④丧偶

2. 是否在业 {occup}: @#
  (1) 在业，职业是 {occup1}: @#
    ①工人 ②专业技术 ③行政管理
    ④商业服务业 ⑤农民 ⑥学生 ⑦其它 {other1}@_____

  (2) 不在业 {occup2}: @#
    1) 离退休，离退休前最长职业 {occup3}: @#
      ①工人 ②专业技术 ③行政管理 ④商业服务业
      ⑤农民 ⑥其它 {other2}@_____
    2) 下岗
    3) 家务
    4) 无业
  
```

如果我们设置 Tab stop in data form every **200** pixels，则数据录入表格显示如下：

调查表

项目编号no

姓名name

性别sex  ①男  ②女

出生年月日birth

年龄 age

1.婚姻状况marriage:

①未婚

②在婚, 与配偶共同生活了 year  年

③离婚或分居

④丧偶

2.是否在业occup:

(1) 在业, 职业是 occup1:

①工人 ②专业技术 ③行政管理

④商业服务业 ⑤农民 ⑥学生 ⑦其它 other1

(2) 不在业occup2:

如果我们设置Tab stop in data form every  pixels, 则数据录入表格显示如下:

调查表

项目编码no

姓名name

性别sex  ①男 ②女

出生年月日birth

年龄 age

1.婚姻状况marriage:

①未婚

②在婚, 与配偶共同生活了 year  年

③离婚或分居

④丧偶

2.是否在业occup:


(1) 在业, 职业是 occup1:

①工人 ②专业技术 ③行政管理

④商业服务业 ⑤农民 ⑥学生 ⑦其它 other1

(2) 不在业occup2:

### 2.1.6 预览数据录入表格 (Preview Data Form)


在 **Data File** 菜单中点击 **Preview Data Form**, 或者按 **Ctrl+T**, 或者在编辑器工作栏 (Editor Toolbar) 中点击  图标, 或者在工作流程栏 (Work process toolbar) 上点击 **2. Make Data File** → **Preview Data Form**, 我们可以在尚未建立数据库 (\*.rec) 的情况下, 先预览数据录入时的调查表布局。预览时, 因为尚未建立数据库, 因此 **CHECK** 功能还无法发挥作用。当修改了调查表文件 (\*.qes) 后, 预览的数据表格不会自动更新, 必须再按预览键来看更新的调查表格式。不过, 要实现新的预览, 无需关闭上一次的预览窗口, 程序会自动更新。关闭预览窗口, 可以点击菜单 **File** → **Close Form**, 或按 **Ctrl+F4**。

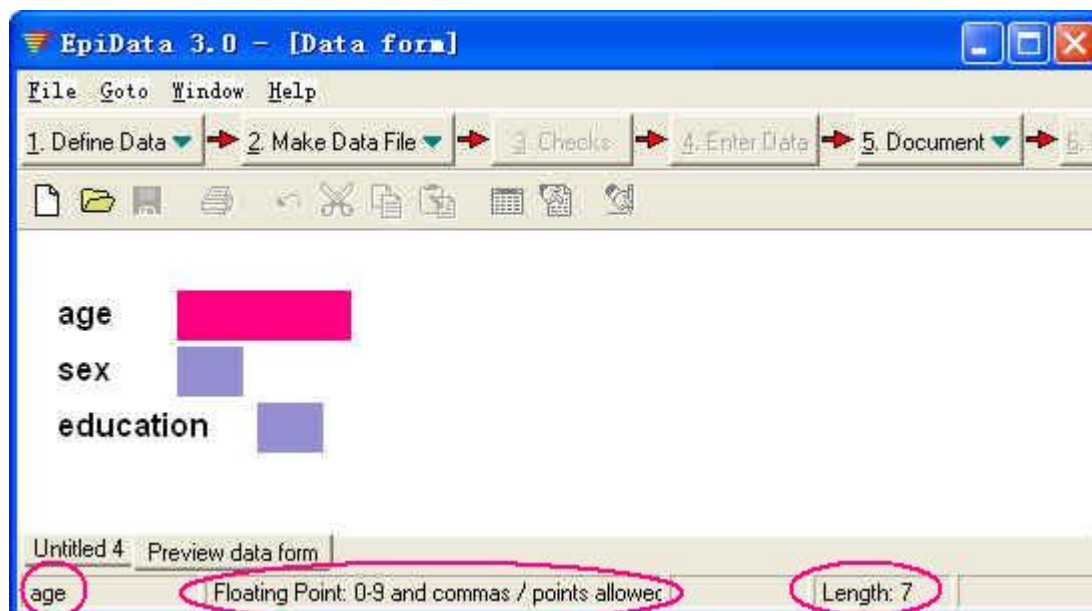
### 2.2 定义变量名 (Field Names)

一个数据库中录入变量的名称可以根据 QES 文件的内容自动创建。EpiData 中命名变量的方式有两种:

- 1) 将第一个单词作为变量名 (First word in question is field name)
- 2) 根据 Epi Info 的规则自动定义变量名 (Automatic field names)

你可以进入 **File** → **Options** → **Create data file** → **How to generate field**

names，选择定义变量名的方法。另外，程序到底使用什么变量名，你可以点击  图标，通过预览数据录入表格来查看，即将光标放在你想了解的变量上，窗口下方状态栏左侧显示的就是程序采用的变量名。另外，该变量的类型及长度也同时显示在状态栏上。



变量名的英文字母是大写还是小写，可以在 **File**→**Options**→**Create data file**→**Letter case of field names**中选择，①Upper-case: 大写；②Lower-case: 小写；③Leave as is: 维持输入时的大小写状态。变量名称的字体及其大小可以在 **File**→**Options**→**Show data form**中设置。

### 2.2.1 将第一个单词作为变量名（First Word in Question is Field Name）

如果你选择这一项，程序会自动将变量编码左侧解释性文字中的第一个单词认作是变量名。如果第一个单词的长度超过 10 个字符，程序只保留该单词的前 10 个字符作为变量名。例如：

1) 如果输入：

```
v1 Enter age of patient ###
```

则程序会创建一个变量名为“v1”的 3 位整数变量。

2) 如果输入：

```
Enter age of patient ###
```

则程序会创建一个变量名为“Enter”的 3 位整数变量。不过，在这种情况下，选择自动定义变量名（Automatic field names）的方式可能会更好。

- 3) 当一个变量名已经使用过，再次出现时，程序会自动添加一个数字，以保证变量名的唯一性。

```
v1 Enter age of patient ###  
v1 Height of patient ###
```

程序会将第一个变量命名为“v1”，而第二个变量命名为“v2”。在这个例子中，像这样输入解释性文字时，其实最好不要重复，而是能够直接写出真正的变量名。你可以在 **File**→**Options**→**Create data file** 中，勾选上“**Update question to actual field name**”，这样，即使你在创建调查表文件时有重复的变量名，在创建的数据库中，程序会自动将其更新为其实际的变量名。

另外，很多使用者习惯输入中文，下面让我们来看看，在选择“将第一个单词作为变量名”时，输入中文，程序会如何定义变量名。首先，如果我们按如下方式创建调查表文件：

```
姓名@_____  
年龄@###  
性别@#
```

结果，创建的数据库显示如下：

```
field1  
field2  
field3
```




上述 3 个变量的变量名被改为 field1、field2 和 field3，同时中文文字也不再显示。如果我们按如下方式创建调查表文件：

```
姓名name@_____  
年龄sex@###  
性别age@#
```

创建的数据库显示如下：

```
name  
sex  
age
```





程序以我们输入的英文来定义变量名，同时，输入的中文文字没有显示。另外，如果将调查表文件中的中英文输入顺序颠倒过来，也会得到相同的结果。可见，至少到当前的版本为止，EpiData 不支持用中文定义变量名。如果同时出现中文和英文，程序会自动选择第一个出现的英文单词作为变量名。

### 2.2.2 自动定义变量名 (Automatic Field Names)

如果你选择了这一项，EpiData 会根据变量编码前的解释性文字，自动生成变量名。变量名第一个字符一定为字母 (A-Z)，之后可以包含字母 (A-Z) 和数字 (0-9)，最多 10 个字符。程序不识别国际字母，例如丹麦字母 æ、Ø 和 å 会被自动转换成 ae、oe 和 aa。定义变量名时，程序会遵循以下几个原则：

1.	程序优先采用大括号中的文字作为变量名。例如，如果变量编码前的解释性文字为 “{my} first {field}”，则自动定义的变量名为 MYFIELD。所以， <b>强烈建议大家使用大括号来定义有意义的变量名。</b>
2.	一些常用词会被忽略（即，类似 “what”、“the”、“of”、“and” 等的词）。例如，“What did you do?”，程序采用的变量名称将会是 YOUUDO。
3.	如果变量编码前没有任何解释性的文字，那么程序会延续上一个变量的名字，同时在末尾加上一个数字。例如，前一个变量的名称为 MYFIELD，那么接下来的变量（如果没有任何解释性文字的话）名会是 MYFIELD1。如果前面一个变量名为 V31，那么下一个变量名会是 V32。如果前面没有变量，则默认的变量名为 FIELD1。
4.	如果可生成变量名的第一个单词是一个数字，程序会自动在数字前插入字母 N。例如，“3 little mice” 的变量名会是 N3LITTLEMI。

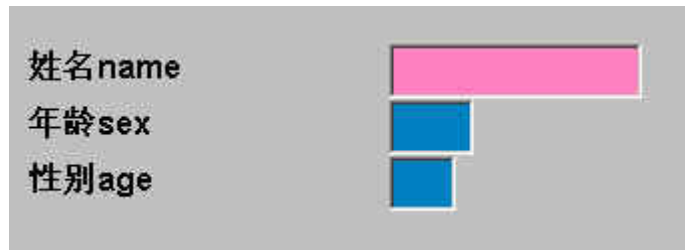
例如：

问题	产生的变量名称	应用的规则
State your {nation}ality	NATION	规则 1
Al{l} you l{i}ke is i{ce}cream	LICE	规则 1
What is your name	ISYOURNAME	规则 2
3. question:	N3QUESTION	规则 4

依然是上面那个中文例子，现在我们选择的是自动定义变量名。如果在调查表文件 (\*.qes) 中，变量编码前只输入中文，而没有英文字符，则和前面一样，3 个变量的变量名会被改为 field1、field2 和 field3，但是中文文字被保留。如果输入如下：

```
姓名name#
年龄sex###
性别age##
```

结果，创建的数据库显示如下：



变量名延续了使用者的定义，name、sex 和 age，但是不同于选择“将第一个词作为变量名”，中文的解释性文字也同样保留。

虽然程序优先采用大括号“{}”中的文字作为变量名，但是即使你把中文放在大括号中，程序还是不会识别。总之，到目前为止，我们不能用中文作为变量名。

### 2.2.3 变量标签 (Variable Labels)

变量标签是对一个变量所含数据内容的描述。在 EpiData 中，程序会根据 QES 文件中，变量编码左侧的解释性文字自动生成变量标签。如果选择了“First word in question is field name”，则扣除作为变量名的第一个单词后，自左向右的文字会被作为变量标签。例如：“v1 Age of patient ###”，变量名为“v1”，变量标签为“Age of patient”。如果选择了“Automatic field names”，则变量名为“v1ageofp”，变量标签为“v1 Age of patient”。

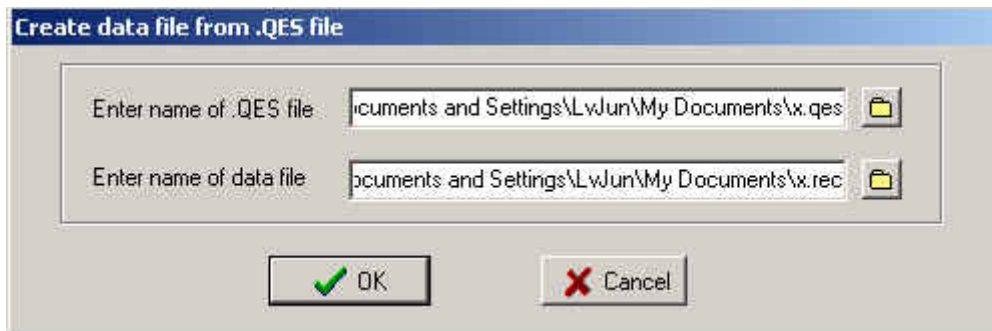
## 3. 数据库的建立和修改 (Create and Revise Data File)

### 3.1 创建数据库 (Create Data File)

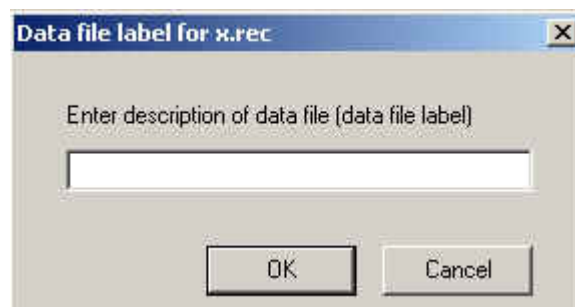
创建完调查表文件 (.QES)，第二步是在此基础上创建数据库，可以有几种操作方式，例如：

- 1) 在工作流程栏 (Work process toolbar) 上点击 **2. Make Data File** → **Make Data File**;
- 2) 在有激活窗口的情况下，从 **Data File** 菜单中点击 **Make Data File**;
- 3) 在没有任何激活窗口的情况下，从 **Data in/out** 菜单中点击 **New Data File**。

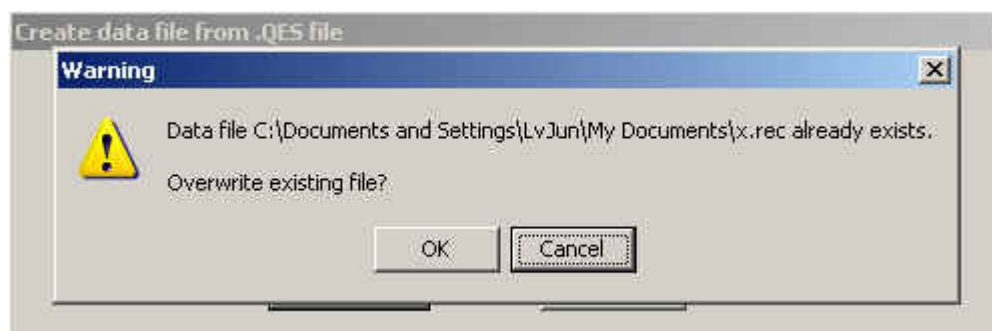
在创建数据库前，已经建立好的调查表文件 (\*.qes) 可以打开，也可以不打开。程序会自动弹出对话框，允许你选择用来创建数据库的调查表文件。数据库文件的扩展名统一为 .REC。程序默认数据库 (\*.rec) 的名称与调查表文件 (\*.qes) 的名称相同，只是扩展名不同。尽管，并不强求文件名一定相同，但我们还是建议数据库名与调查表文件名保持一致。



使用者可以给数据库输入一段简短的描述性文字（最长不超过 50 个字符），我们将其称为数据库标签（**data file label**）。数据库标签会作为数据库的一部分被保存起来，输出报表时也会一同显示。你可能会发现，一些分析 Epi Info 文件格式的数据分析软件可能无法读取有这种标签的文件。我们建议你可以用这类数据库去实验一下，会不会因为这个标签引起问题。如果是的话，你可以不必输入数据库标签，直接点击 **OK**。



如果新建的数据库与已有的数据库重名，那么程序会把已有的那个数据库删除，替换为新建的数据库，而旧的数据库中的全部信息将丢失。如果你只是想在原有数据库的基础上作一些调整，但又不想丢失已有的数据，例如，增加一个变量或者改变已有变量的类型，请你选择 **Tools** 菜单中选择 **Revise Data File**。



### 3.2 修改数据库（Revise Data File）

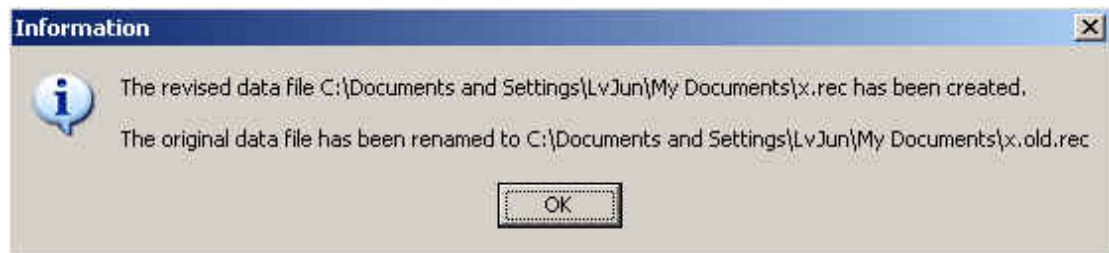
你可以在不丢失数据的前提下，修改一个已经录入了的数据库的结构。已经录入的数据会被拷贝到新数据库中相同名字的变量下，这个变量的格式有可能是修改了的。你也可以增加变量、修改变量的定义、或者删除变量。然后，先关闭所有的文件，点击菜单 **Tools** → **Revise Data File**，完成数据库的修改。主要的工

作步骤如下：

- 1) 打开调查表文件 (\*.qes) 进行修改。如果手头上没有现成的调查表文件，你也可以用已有的数据库 (\*.rec) 反过来创建一个新的调查表文件，即选择菜单 **Tools** → **QES File from REC File**;
- 2) 编辑、修改调查表文件 (\*.qes)，例如，增加新变量、删除变量、改变变量类型等；
- 3) 保存修改后的调查表文件，然后关闭该文件；
- 4) 选择 **Tools** → **Revise Data File**;
- 5) 选择修改好的调查表文件 (\*.qes) 和准备修改的数据库 (\*.rec)。



如果你删除了某些变量，或者修改了变量名，你会丢失一些数据。请认真检查修改了的数据库，避免错误操作。不过，即使操作失误，原来的数据库还是可以恢复的，它现在已被存为另外一个数据库（仍在相同的文件夹里），文件名为 FILENAME.OLD.REC。



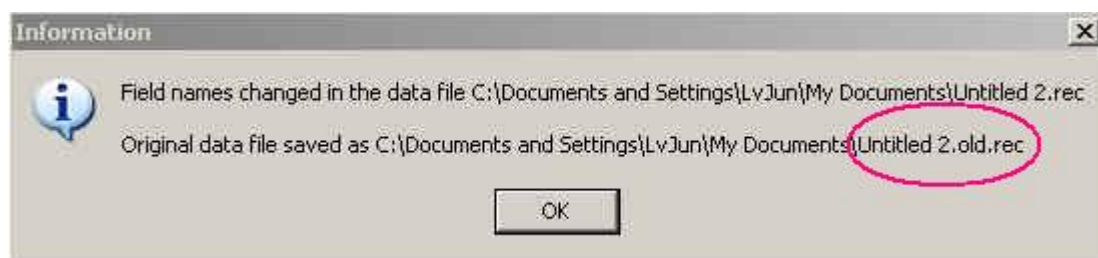
EpiData 中有两种方式生成变量名（可参见“[定义变量名](#)”一节）。改变命名方式，有可能会修改变量名，进而丢失数据。在修改数据库时，程序会先检查原始数据库创建时使用的是什么变量名定义方法，如果不同于现在的 **Options** 中的设置，程序会自动弹出一个警告框，建议你使用相同的变量名定义方法。

你可以修改原始数据库中的变量类型。所有变量类型都可以修改为字符型变量或大写字母的字符型变量。数值型变量可以修改为有相同或更多小数位数的数值型变量。如果修改后的小数位数变少了，程序会弹出警告框，提醒你，这样修改可能会丢失数据。

原始数据库的变量类型	可以修改为
整数	浮点型、字符型、大写字母字符型、加密变量
浮点型	浮点型、字符型、大写字母字符型、加密变量
IDNUM	整数、浮点型、字符型、大写字母字符型、加密变量
字符型	大写字母字符型、加密变量
大写字母字符型	字符型、加密变量
声索引变量	字符型、大写字母字符型、加密变量
所有数据类型	字符型、大写字母字符型、加密变量

### 3.3 重新定义变量名 (Rename Fields)

你可以对已有数据库中的变量名进行重新定义。进入 **Tools** → **Rename Fields** → 选择需要修改变量名的数据库。在弹出的窗口中，第一列是原有的变量名，第二列是标签。如果要修改变量名，请将光标放在第三列相应的变量行上，键入新的变量名。不需要修改变量名的变量，无需在对应的第三列上输入。修改完毕后，点击 **Save and close**，保存并关闭窗口。这时，旧的数据库会被另存为 **Filename.old.rec**，以备需要的时候恢复。



如果该数据库已经建立了配套的 CHECK 文件，尽管对应的变量块（field block）的名称会随之改变，但是，CHECK 语句中某些引用其它变量名的命令（例如，GOTO 变量名；或者 COMMENT LEGAL USE 变量名）不会改变，你必须手动修改。

#### 4. 建立核查文件（Check File）

EpiData 最简单的使用流程是：

- 1) 创建调查表文件（\*.qes）；
- 2) 在调查表文件的基础上建立数据库（\*.rec）；
- 3) 在数据库（\*.rec）中录入数据。

通过上述简单的工作流程，你可以得到需要的数据库。但是，所有数据录入完毕后，你可能需要花费一定的功夫去检查数据录入得是否合理、正确。但是，如果在录入数据前设置了 CHECK 文件，在数据的录入过程中，程序会自动根据你的设置的条件，实时检查录入数据的合理性、正确性，这是保障数据录入质量的一个重要的措施。同时，通过 CHECK 文件，你还可以控制数据录入的流程（例如，根据录入的数值，自动从一个变量跳转到另一个变量）。**CHECK 的文件名必须与数据库的文件名相同，唯一不同的就是扩展名**，前者为\*.chk，后者为\*.rec。

通常，我们是在创建完数据库（\*.rec）后再创建 CHECK 文件，创建方式主要有两种：

- 1) 点击菜单中的 **Checks** → **Add/Revise**，或者在工作流程栏（Work process toolbar）上点击 **3. Checks**。这种方式可以用来指定或修改变量的 CHECK 命令，但是变量块以外的命令（例如，BEFORE FILE 等命令）只能在编辑器中修改。
- 2) 使用编辑器手动编写或修改所有 CHECK 命令。切记，CHECK 的文件名必须与数据库的文件名相同，唯一不同的就是扩展名。

一般情况下，这两种方法我们都会用到。用第一种方法可以进行最基本的 CHECK 设置，然后用编辑器添加更复杂的 CHECK 命令或文件水平（而不是变量水平）上的 CHECK 命令。

只要同名的 CHECK 文件存在，在录入数据时，对应的 CHECK 文件中的命令会自动装载，对数据的录入过程发挥作用。

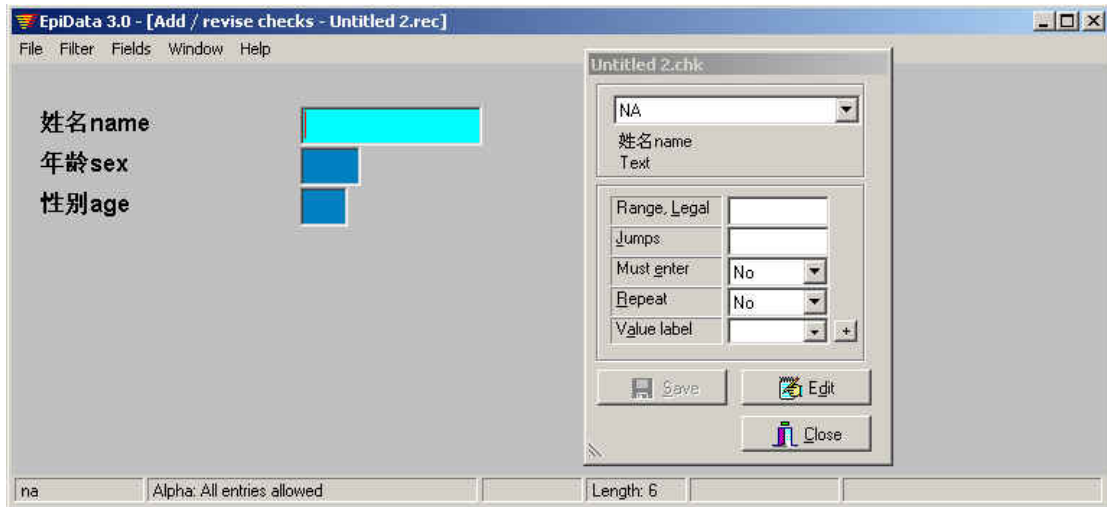
#### 4.1 添加/修改核查命令（Add/Revise Checks）

##### 4.1.1 基本操作

点击菜单中的 **Checks** → **Add/Revise**，选择需要建立或修改 CHECK 的数据库。这时，程序会弹出两个窗口，即数据录入表格窗口和 CHECK 设置窗口。按 **F6** 功能键可以实现数据录入表格窗口和 CHECK 设置窗口之间的转换。如果当前激活的是数据录入表格窗口，我们也可以按 **Ctrl**+**→** 键，激活 CHECK 设置窗

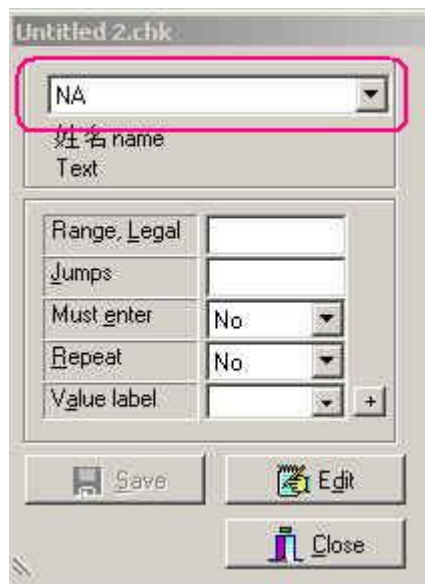
口；相反，如果当前激活的是 CHECK 设置窗口，我们可以按 **Ctrl**+**←** 键，激活

数据表格窗口。



选择变量，添加 CHECK 命令，可以有以下几种方式：

- 1) 在数据录入表格窗口中，通过鼠标点击，或者按 **Tab** 键、或 **Enter** 键可以激活目标变量；
- 2) 在 CHECK 设置窗口中，最上方有一个可以选择变量名的下拉列表，你可以从这个列表中选择目标变量，变量的顺序与数据表格中的顺序一致；



- 3) 当 CHECK 设置窗口激活时，按 **Ctrl+↑** 或者 **Ctrl+↓** 键可以选择目标变量。

当 CHECK 设置窗口激活时，你可以使用 **↑**、**↓** 键、或 **Tab** 键、或 **Enter** 键选择 5 个基本的 CHECK 设置，即 **Range, Legal**、**Jumps**、**Must enter**、**Repeat**、

**Value label**。

如果数据表格窗口处于激活状态，你可以使用下列组合键，激活上述 5 种基本 CHECK 设置。

- 1) **Ctrl+L**: 激活 **Range,Legal**, 设置当前变量的允许范围、允许值;
- 2) **Ctrl+J**: 激活 **Jumps**, 设置跳转功能;
- 3) **Ctrl+E**: 激活 **Must enter**, 将当前变量设置为“**Yes**”, 即必须录入; 如果再按一次 **Ctrl+E**, 则恢复默认状态“**No**”, 即“**必须录入**”状态取消;
- 4) **Ctrl+R**: 激活 **Repeat**, 将当前变量设置为“**Yes**”, 即重复输入; 如果再按一次 **Ctrl+R**, 则恢复默认状态“**No**”, 即“**重复**”状态取消;
- 5) **Ctrl+A**: 激活 **Value label**, 添加当前变量的数值标签。

另外, 还有一些组合键, 不论是哪一个窗口激活, 都可以使用:

- 1) **Alt+S**: 相当于按 CHECK 设置窗口中的 **save** 按钮, 保存所有的 CHECK 设置, 同时, 程序不会退出 **Add/Revise checks**;
- 2) **Alt+D** 或者 **F9**: 相当于按 CHECK 设置窗口中的 **Edit** 按钮, 弹出 CHECK 命令的编辑窗口;



- 3) **Alt+C**: 相当于按 CHECK 设置窗口中的 **Close**, 即退出 **Add/Revise checks**。如果修改后尚未保存, 程序会提示你, 是否要保存所做的修改。

你可以将一个变量上设置的 CHECK 命令拷贝或移到另一个变量上。激活设置好 CHECK 命令的变量, 按 **Ctrl+C** 拷贝或 **Ctrl+X** 剪切所有的 CHECK 命令, 再激活另一个变量, 按 **Ctrl+V** 把拷贝的 CHECK 命令粘贴到这个新的变量上。拷贝/剪切/粘贴功能可以拷贝基本的 CHECK 功能(如 RANGE, LEGAL, JUMPS, MUSTENTER, REPEAT) 和数值标签, 以及其它在 BEFORE/AFTER 录入块以



外的命令。

选择菜单 **Tools** → **Clear Checks**，可以清除某个数据库设置的所有 CHECK 命令。删除的 CHECK 将无法恢复，所以使用这个功能前一定要慎重考虑清楚。

#### 4.1.2 基本 CHECK 命令的设置

##### 数值允许范围及允许值 (Range,Legal)

在 **Range,Legal** 的定义框中键入允许录入的最小值和最大值，并用连字符“-”连接。例如，键入“2-5”，表示当前变量只允许录入 2、3、4、5 四个数值。如果只对最大值有限制，则用“-INF”（负无穷大）表示最小值。如果只对最小值有限制，则可以用“INF”（无穷大）表示最大值。例如，键入“-INF-5”表示当前变量录入的数值必须  $\leq 5$ 。键入“0-INF”表示录入的数值必须  $\geq 0$ 。

允许值 (legal) 是在 **Range,Legal** 的定义框中键入所有允许输入的数值，数值之间以逗号或空格间隔。如果使用空格作间隔，程序也会自动将其转换为逗号间隔。例如，键入“4,6,8,10”表示当前变量只允许录入 4、6、8、10 四个数值。

如果你的设置中既有允许的范围，又有允许值，那么范围值必须放在前面，允许值放在后面。例如，键入“2-6,8”表示允许录入的数值包括 2、3、4、5、6 和 8。键入“8,2-6”，程序会提示错误。

如果你想用逗号作为小数点分隔符，而不是常用的圆点，请用双引号将这个定义括起来。

##### 忽略缺失值 (IGNOREMISSING)

如果在数据录入表格中，某个变量是通过其它变量的计算得到的，而当参与这个计算的某个变量是缺失值时，EpiData 默认给出的计算结果也会是一个缺失值。如果在 CHECK 文件中的 BEFORE FILE、BEFORE ENTER 或其它地方中使用 IgnoreMissing 命令，即使计算中含有缺失值，依然可以得到一个有效的结果。这时，缺失的数值会被当做 0 处理。只有当所有的变量都是缺失值时，得到的结果才会是一个缺失值。

例如，一个数据库中有 4 个变量：V1, V2, V3 和 V4，这些都是整数变量。V1=2, V2 缺失, V3=5, V4 的 CHECK 命令为：

```
V4
  BEFORE ENTRY
    V4=V1+V2+V3
  END
END
```

这个计算过程的默认结果是缺失值，因为 V2 是缺失值。但是，如果修改 CHECK 命令为：

```
V4
  BEFORE ENTRY
    IGNOREMISSING
    V4=V1+V2+V3
```

END  
END

这时，计算结果为  $V4=2+0+5=7$ 。

### 跳转 (Jumps)

如果当前变量设置了跳转功能，则表示在输入某个指定的数值后，程序会自动跳到某个对应的变量上。例如，如果当前变量是一个性别变量（1=男性，2=女性），我们可以设置跳转功能为：当性别变量录入为 1（表示男性）时，直接跳转到变量 V23，当性别变量录入为 2（表示女性）时，直接跳转到变量 V40。

设置跳转功能时，请依次键入跳转值、大于号 (>)、跳转的目标变量名。跳转语句间用逗号分隔。例如，1>V23, 2>V40 表示输入 1 时，跳转到变量 V23；输入 2 时，跳转到变量 V40。如果定义中包含空格或逗号，请用双引号把整个定义括起来。例如，"2.5>V30", "3,5>V35"

除了可以指定跳转的目标变量外，还可以使用另外两种跳转方式，即 END 和 WRITE。END 表示“跳转到数据录入表格的最后一个变量”，WRITE 表示“将当前记录存盘”。例如，设置跳转为“1>V30,2>END,3>WRITE”表示：当输入 1 时，跳转到变量 V30；如果输入 2，直接跳转到最后一个变量；如果输入 3，将当前记录存盘。如果键入跳转命令：

```
AUTOJUMP V30
```

这表示不管当前变量录入什么数值，都直接跳转到变量 V30。如果使用 AUTOJUMP 命令，跳转的设置框中就只能输入这一个命令。AUTOJUMP 这个命令特别适用于那种不按正常的从左到右、从上到下顺序设计的表格。

为了使跳转设置更加便捷，你可以采用下面的快捷设置方式。当键入完跳转值和“>”后，用鼠标直接点击跳转的目标变量。这时，点击变量的变量名会自动插入到“>”后面。

### 必须录入 (Must Enter)

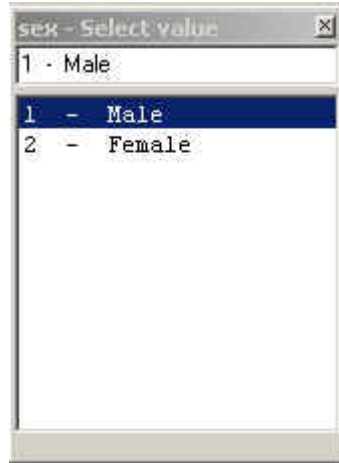
这个命令要求必须给当前变量输入数值，否则无法进入下一个变量。

### 重复 (Repeat)


如果选择“**Yes**”（是），则前一条记录在当前变量上录入的数据将在接下来的新记录上重复显示。在数据录入过程中，重复显示的数据可以修改。如果某些录入内容在不同记录间改动很少（例如，调查员姓名、调查者所属地区等），这项重复功能可以省去很多重复键入的工作。

### 数值标签 (Value Labels)

数值标签是一组数值加上对应的文字注释，可以解释每个数值代表的涵义。例如，创建一个性别变量，用 1 表示男性，2 表示女性。如果设置了这个数值标签，在数据录入过程中，当你按 **F9**，或数字键盘上的 **+** 时，程序会自动弹出一张表，告诉你 1 和 2 各自对应的涵义。



## 1. 定义新标签

点击 CHECK 设置窗口中 **Value label** 旁边的  按钮，会弹出一个“Edit value labels”的窗口。



你可以如下图所示键入数值标签。LABEL 后面的 *Label\_sex* 是根据变量名起的，如果你愿意，也可以修改。



数值“1”、“2”前面可以不输入空格，但是加上空格可以使阅读起来层次更加清楚。另外，如果标签的注释性文字中带有空格，请用引号括起来。例如：



编辑完毕后，可以按菜单上的 **Accept and Close** 或者 **Alt+A** 键关闭编辑窗口。这时，新标签的名字会显示在 **Value label** 的下拉列表中。

## 2. 编辑已有的标签

在 **Value label** 的下拉列表中选择你要编辑的数值标签的名字，然后点击右侧的 **+** 钮，编辑窗口弹出。修改后按菜单上的 **Accept and Close** 或者 **Alt+A** 键，保存并关闭编辑窗口。

## 3. 给变量指定已有的标签

在对应的变量上，从 **Value label** 旁的下拉列表中选择相关的标签。不同的变量可以共用相同的数值标签，你只需要定义一次。

## 4. 清除变量的数值标签

在对应的变量上，从 **Value label** 的下拉列表中选择[none]，即可清除在该变量上设置的数值标签。

## 5. 使用预设的标签

在安装 EpiData 时，同时会有一个数值标签库被保存在 EpiData 的程序文件夹下。该库的文件名叫 EpiData.lbl。在设置不同的数据库时，可能会用到相同的数值标签。这时，利用这个库会省去重复设置的麻烦。


```
Label Sex
  1 Male
  2 Female
  9 Unknown
End

Label Agegroups
  1 "0-15 years"
  2 "16-19 years"
  3 "20-29 years"
  4 "30-39 years"
  5 "40-49 years"
  6 "50-59 years"
  7 "60-69 years"
  8 ">69 years"
End

Label Continent
  1 Africa
  2 South America
  3 Asia
  4 Europe
  5 North America
  6 Australia
  9 other
end

For Help, press F1
```

点击 **Value label** 旁的下拉列表，你可以看到该库中保存的数值标签的名字，然后选择你想采用的数值标签。默认安装的数值标签库（EpiData.lbl）中有 3 个标签，性别（sex）、年龄组（agegroups）、洲（continent）。注意，这一版本的软件提供的数值标签库中有个小错误，洲（continent）标签中，2 对应的 South America 和 5 对应的 North America，标签中存在空格，而这里又没有使用双引号将其括起来，所以属语法错误。而 EpiData.lbl 中如果出现语法错误，程序是不会提示你的，只会忽略它。结果就是，我们在 **Value label** 旁的下拉列表中只能看到性别（sex）和年龄组（agegroups）的标签，而看不到洲（continent）的标签。

另外，由于未知的原因，下拉列表中显示的标签名内含乱码。如果你想使用这些标签，建议你在选择该标签名后，点击旁边的 ，进入编辑窗口，修改标签名，即 LABEL 后面的 *Label\_field*，保存后退出。然后选择这些改了名的数值标签就可以了。不过，经你修改了的这些新标签，只会保存在当前的 CHECK 文件中，而不会对 EpiData.lbl 文件有任何影响。如果你确实想在 EpiData.lbl 库中添加更多的数值标签，可以在编辑器中或其它字处理软件（如 Windows 自带的写字板、WORD 等）中添加、修改（例如，修改前面提出来的那个错误）。

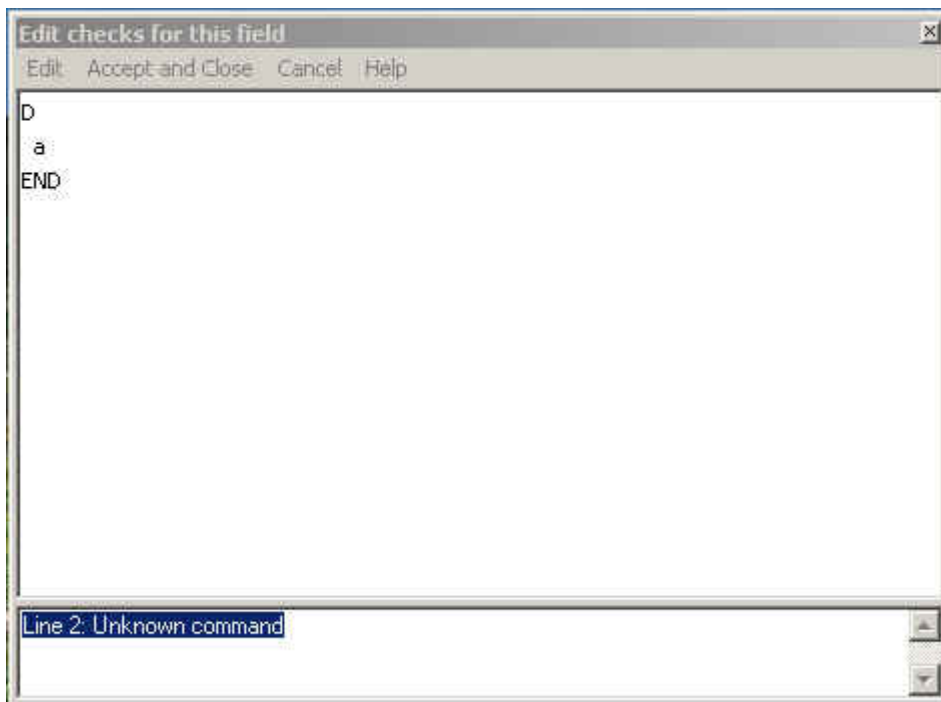
### 4.1.3 编辑当前变量的所有 CHECK 语句

点击 CHECK 设置窗口中的 **Edit** 按钮，或按 **Alt+D**，或在数据录入表格窗口激活的情况下按 **F9**，程序会弹出一个编辑窗口，你可以直接编辑当前变量的所有 CHECK 语句，就像用编辑器编辑整个 CHECK 文件一样。

如果当前变量没有设置 CHECK 命令，这时的编辑窗口中只会显示变量名（表示变量块开始）和 END（表示变量块结束）。如果设置了 CHECK 命令，这些命令会在编辑窗口中显示出来。同时，你也可以编辑或添加新的命令。按 **Esc** 或点击 **Cancel** 可以放弃修改。点击 **Accept and close** 或按 **Alt+A** 表示接受和保存修改。参见“[CHECK 的文件结构](#)”和“[CHECK 命令列表](#)”，可以了解如何编辑 CHECK 命令。

使用 **Checks** → **Add/Revise** 不能编辑除变量块以外的 CHECK 文件块（file block），你必须通过编辑器或其它字处理软件来编辑整个 CHECK 文件（\*.chk）。

当选择 **Accept and close** 时，程序会自动检查你编辑的 CHECK 命令。如果没有发现错误，编辑窗口会自动关闭。如果发现错误，编辑窗口会被分成两个窗口。上面的窗口显示 CHECK 命令，下面的窗口显示发现的错误，以及错误所在的行号。在下面的窗口中，双击显示错误的行，光标会自动跳到含有错误的 CHECK 命令行上。修改错误后，请选择 **Accept and close**。需要注意的是，程序对表达式和计算式不进行检错。



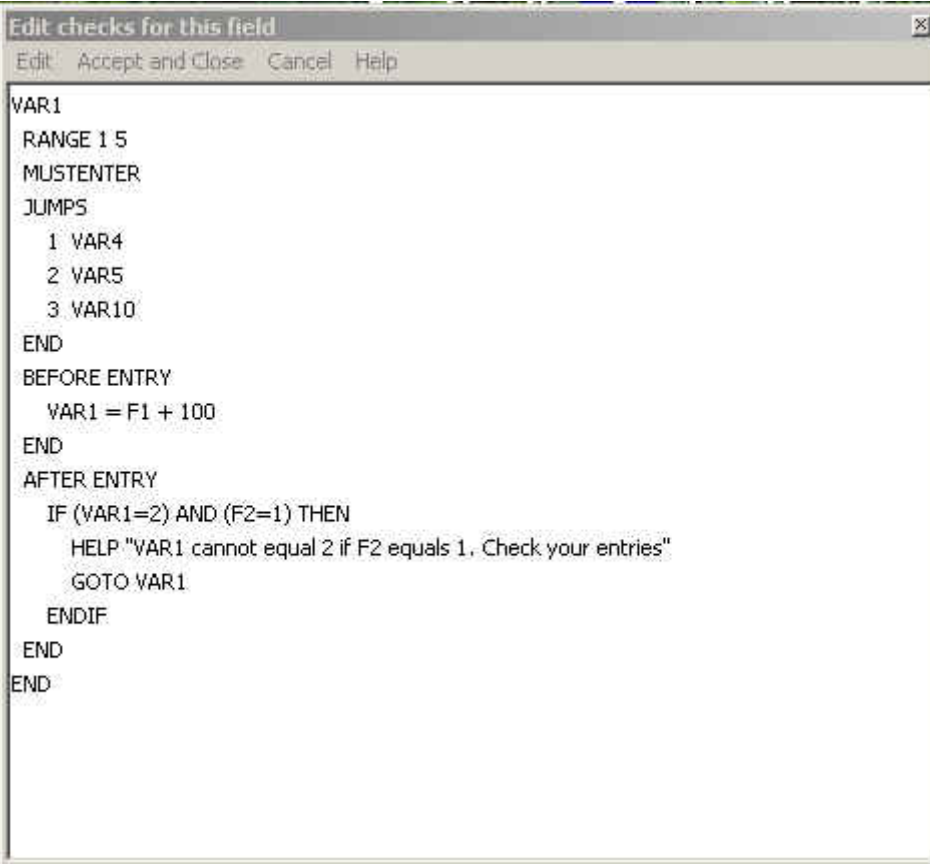
## 4.2 CHECK 的文件结构

CHECK 文件中的所有命令都保存在块（blocks）中。EpiData 支持两种基本的块：标注块（label block）和变量块（field block）。标注块的情况请参见 [CHECK 命令列表](#) 一节。

所有与某个变量相关的命令都保存在一个变量块中。变量块以变量名开始，以命令 END 结束。END 命令后面的内容通通会被忽略。小写“end”与大写“END”

表达的意义相同。

一些命令本身就是块（例如，**LEGAL...END**, **JUMPS...END**），而其它一些命令只占用一行（例如，**RANGE**, **GOTO**）。所有的块都以命令 **END** 结束。例如：



```
VAR1
RANGE 1 5
MUSTENTER
JUMPS
  1 VAR4
  2 VAR5
  3 VAR10
END
BEFORE ENTRY
  VAR1 = F1 + 100
END
AFTER ENTRY
  IF (VAR1=2) AND (F2=1) THEN
    HELP "VAR1 cannot equal 2 if F2 equals 1. Check your entries"
    GOTO VAR1
  ENDIF
END
END
```

第一行表示一个变量名为 **VAR1** 的变量块开始。最后一行为 **END**，表示该变量块结束。第 2 和 3 行只含简单的命令。第 2 行限制该变量只能录入从 1 至 5 的数值。第 3 行要求该变量必须录入。

第 4 行表示 **JUMPS** 命令的开始。**JUMPS** 命令本身就是块，必须以 **END** 结束（如第 8 行）。本例中，录入 1，程序自动跳转到变量 **VAR4**；录入 2，跳转到变量 **VAR5**；录入 3，跳转到变量 **VAR10**。

本例中，变量块的行表现为不同的缩进。**JUMPS** 块的行缩进更多。缩进不是必需的，但是这样可以使 **CHECK** 文件阅读起来更直观，使用者可以很方便的识别命令块的开始和结束。**EpiData** 的编辑器有[自动缩进](#)功能，这使 **CHECK** 文件的编辑过程中缩进功能很容易实现。

上面这个例子中，有两个重要的块命令：**BEFORE ENTRY...END** 和 **AFTER ENTRY...END**。当变量被激活，但尚未开始录入时，程序会执行 **BEFORE ENTRY...END** 块中的所有命令。我们可以利用这个功能给变量自动填充一个默认值，使用者可以修改这个默认值。当使用者将光标移到另一个变量时，程序会自动执行 **AFTER ENTRY...END** 块中的所有命令。如果在变量块中有命令，而未见 **AFTER** 或 **BEFORE ENTRY** 块，这些命令会被视做 **AFTER ENTRY...END** 块的一部分来处理。

在数据库中，不一定所有的变量都要设置变量块。

### 4.3 实例

在 EpiData 的安装文件夹内有个 `samples` 文件夹。下面我们将以其中的一个儿童生长研究为例进行讲解，请查看对应的文件 `SAMPLE.QES`、`SAMPLE.REC` 和 `SAMPLE.CHK`。

我们建议在编辑器中编辑 `CHECK` 文件时，请选用固定宽度的字体（例如，`Courier New`）。这样，使用缩进功能时较为方便，也便于文件的阅读。

`CHECK` 文件中的第一个命令块（第 2-21 行<sup>1</sup>）是一个标注块（`LABELBLOCK`）。在这个命令块里，我们定义了两组数值标签，一个是“国籍”（`nationality`），另一个是“年龄”（`years`）。

```
LABELBLOCK
  LABEL nationality
    1   Danish
    2   British
    3   American
    4   "Other Scandinavian countries"
    5   "Other European countries"
    6   Asian
    7   "South American"
    8   Other
    9   N.a.
  END
  LABEL years
    1   "0 years"
    2   "1-2 years"
    3   "3-5 years"
    4   "6-10 years"
    5   ">10 years"
  END
END
```

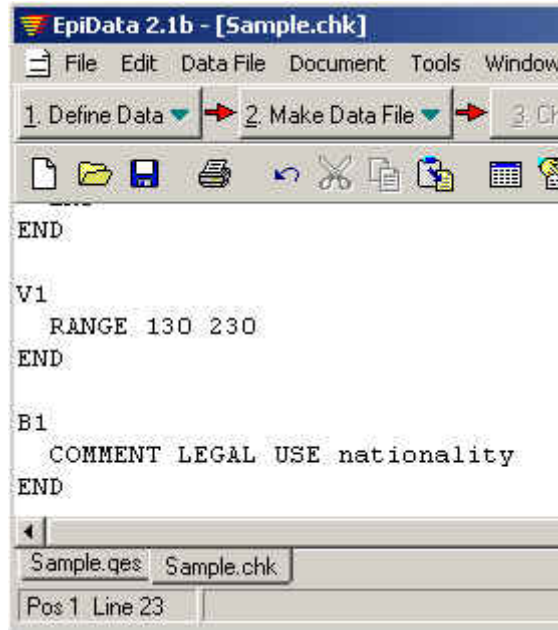
数值标签解释了每个数字对应的涵义（例如，2 表示英国公民）。两组数值标签本身就是一个块。国籍标签从第 3 行开始，到第 13 行结束。

第一个变量块从第 23 行开始，到第 25 行结束。这个变量块设置的是变量 `V1`，是关于被调查者身高的信息。这里只设置了一个命令，即“`RANGE 130 230`”，限制该变量的允许录入值范围在 130~230。超出这个范围的不合理录入将被拒绝。例如，输入被调查者的身高为 3cm，这显然不合理，程序将拒绝录入。

---

<sup>1</sup> 窗口下方的状态栏上可以显示光标所在位置的行、列数。





下一个变量块（第 27-29 行）设置的是变量 **B1**，有关被调查者的国籍。从 QES 文件中，我们可以看到这个变量是一位整数。第 28 行的命令：

```
COMMENT LEGAL USE nationality
```

表示该变量使用标注块中定义的“nationality”的数值标签。录入过程中，当该变量被激活时，使用者可以按 **F9** 或数字键盘中的 **+ 键**，查看允许值列表和对应的数值涵义。

**D1** 变量的变量块为第 31-45 行。录入的信息是被调查者的出生日期。第 32-44 行是一个 **AFTER ENTRY** 命令块。其中的命令要求在录完出生日期后：  
1) 检查录入的出生日期是否合理；2) 计算被调查者的年龄，并输入到 **D2** 变量中。对日期的检查是由两个嵌套的 **IF...THEN** 块实现的。第一个 **IF** 块（第 33 行）是：

```
IF (YEAR (d1)<1900) OR (d1>TODAY) THEN
```

这个语句的意思是，**如果** **D1** 变量录入的年份小于 1900，**或者**如果 **D1** 变量录入的日期大于今天的日期（即，录入的日期是未来的某一天），**那么**需要执行 **THEN**（第 33 行）和 **ELSE**（第 36 行）之间的命令。在这个例子中，程序会自动弹出一个信息框，要求录入员核对录入的出生日期。这时，光标仍然会停留在 **D1** 变量上，而不是继续跳到下一个变量。

如果，**IF** 语句（第 33 行）中的两种情况都不符合，那么程序会执行 **ELSE**（第 36 行）和 **ENDIF**（第 43 行）之间的命令。如果没有 **ELSE** 命令出现，那么下一个要执行的命令就是跟在 **ENDIF** 命令后面的一个命令。

本例中，**ELSE...ENDIF**（第 36-43 行）之间又含有一个新的 **IF...THEN-END** 块。可见，**IF...THEN...ELSE...ENDIF** 块是可以嵌套设置的。请留意 **ELSE** 和 **ENDIF** 的配套使用。在这个例子中，最里面的 **ENDIF**（第 42 行）与最里面的

IF（第 37 行）配套，而最外面的 ENDIF（第 43 行）与最外面的 IF（第 33 行）配套。

新的 IF...THEN...ELSE...ENDIF 块从第 37 行开始：

```
IF (ROUND (INT ((TODAY-D1) / 365.25))<15) THEN
```

这段语句表示，**如果**用当天的日期减去 D1 变量录入的日期，算得的被调查者年龄小于 15 岁，**那么**请执行第 38-39 行的命令，即弹出一个警告框，“被调查者的年龄太小，所以不会是病人”。如果第 37 行的情况不符，那么程序会执行 ELSE 块（第 41 行）中的命令，计算被调查者的年龄，然后将结果输入 D2 变量。

第 42 行结束最里面的 IF...THEN 块。

第 43 行结束最外面的 IF...THEN 块。

第 44 行结束 AFTER ENTRY 块。

第 45 行结束 D1 变量块。

嵌套的 IF...THEN 命令块不一定限于两个，你可以根据需要去嵌套更多数量的 IF...THEN 命令块。不过，嵌套的 IF...THEN 块越多，使用者就越容易遗漏 ELSE 和 ENDIF，所以建议使用缩进功能，使命令块阅读起来更加直观。

```
D1
AFTER ENTRY
  IF (Year(d1)<1900) OR (d1>Today) THEN
    HELP "Please check the date of birth.\n\nIt does not appear to be right." TYPE=CONFIRMATION
    GOTO D1
  ELSE
    IF (ROUND(INT((TODAY-D1)/365.25))<15) THEN
      HELP "Age too small for a parent." TYPE=ERROR
      GOTO D1
    ELSE
      D2=ROUND (INT({TODAY-D1}/365.25))
    ENDIF
  ENDIF
END
END

D2
NOENTER
END

V1A
JUMPS
  N v6

```

Sample.qes Sample.chk  
Pos 4 Line 31

D2 变量的变量块（第 47-49 行）只有一个 CHECK 命令，NOENTER，表示使用者不能在这个变量上录入数据。


V1A 变量是询问被调查者是否有孩子。如果没有，则不需要填写从 V1B 至 V2 变量的内容，可以直接进入 V6 变量。第 52-54 行的命令（JUMPS）表示，如果录入的是“N”（否），必须跳转到变量 V6。第 55 行（MUSTENTER）表示该变量必须录入。

第 56-74 行为一个 AFTER ENTRY...END 命令块。其中含有一个 IF...THEN...ELSE...ENDIF 块。如果 V1A 填“否”，即被调查者没有孩子，将执行第 58-67 行的命令，即清除变量 V1B、V1C、V1D、V1E 和 V2 中的内容，

并隐藏这些变量，使录入员无法录入这些变量，因为如果被调查者没有孩子的话，这些变量也就跟他没有什么关系。

如果 V1A 填“是”，那么将执行第 69-72 行的命令。UNHIDE（显示）同上的 4 个变量，即允许录入数据。

```
V1A
JUMPS
  N v6
END
MUSTENTER
AFTER ENTRY
  IF NOT v1a THEN
    CLEAR V1B
    CLEAR V1C
    CLEAR V1D
    CLEAR V1E
    CLEAR V2
    HIDE V1B
    HIDE V1C
    HIDE V1D
    HIDE V1E
    HIDE V2
  ELSE
    UNHIDE V1B
    UNHIDE V1C
    UNHIDE V1D
    UNHIDE V1E
  ENDIF
END
END
```



V1C 变量块（第一个孩子的身高，见第 89-103 行）给我们展示了如何使用 BEFORE ENTRY。BEFORE ENTRY...END 块（第 90-96 行）是在程序激活该变量、但尚未录入时执行的。在这个例子中，BEFORE ENTRY 这个命令是要给 V1C 变量事先赋值。如果已经录入了被调查者的身高，这里要赋的默认值就是被调查者身高的一半；否则，默认值设为 50cm。AFTER ENTRY 命令块中为一个有条件的 GOTO 命令。如果被调查者只有一个孩子（即，V1B=1），那么被调查者孩子的平均身高（V2）就等于这一个孩子的身高（V2=V1C），而接下来需要录入的变量就是 V6（GOTO V6）。在变量 V1D 和 V1E 同样应用了这种设置。

```

V1C
  BEFORE ENTRY
    IF v1>0 THEN
      V1C=V1/2
    ELSE
      V1C=50.0
    ENDIF
  END
  AFTER ENTRY
    IF v1b=1 THEN
      V2=V1C
      GOTO V6
    ENDIF
  END
END

```

Sample.qes    Sample.chk

Pos 6 Line 89

最后，我们想提一下的就是使用标注块的好处。变量 V12 和 V13 都需要设置年龄范围的数值标签。分别在两个变量的变量块中重复设置相同的数值标签，实在没有必要。这里，在每个变量块中，我们只用了一行命令就实现了这个目的。如果数值标签需要修改，你只需要在标注块中修改即可，无需到每个变量块那里都进行修改，这样做也很容易遗漏。

```

V12
  COMMENT LEGAL USE years
END

V13
  COMMENT LEGAL USE years
END

```

#### 4.4 CHECK 命令列表

\*

参见 [COMMENTS](#)

#### AFTER ENTRY

设置一个命令块，其中的命令是在当前变量录入完毕后和/或光标移到另一个变量时执行。AFTER ENTRY 是一个块命令，必须以 END 结束。如果命令出现在变量块中，即使没有放在 AFTER ENTRY 块中，这些命令仍被视作 AFTER ENTRY 块命令。例如：

```

AFTER ENTRY
  <command>
  <command>
  ...

```

END

## AFTER FILE

设置一个命令块，其中的命令是在数据库关闭时执行。参见 [BEFORE FILE](#)。  
例如：

```
AFTER FILE
  HELP "Remember to make a backup of the data file!"
TYPE=WARNING
END
```

## AFTER RECORD

设置一个命令块，其中的命令是在保存一条新的或者修改了的记录前执行。你可以用 **AFTER RECORD** 命令检查数据是否录入正确。如果在 **AFTER RECORD** 命令块中执行 **GOTO** 命令，那么当前记录将不被保存。

在下面这个例子中，数据表格的第一个变量为 ID 号 (ID1)，而最后一个变量还是要求录入 ID 号 (ID2)，以此进行核对。如果两个 ID 号录入的不一样，或者任何一个没有录入，程序会自动弹出一个警告框，光标会移到 ID1 变量上，该记录暂不被保存。

```
AFTER RECORD
  IF (ID1<>ID2) THEN
    HELP "ID1=@ID1 and ID2=@ID2\n\nPlease check the data"
TYPE=WARNING
    GOTO ID1
    EXIT
  ENDIF
  IF (ID1=.) OR (ID2=.) THEN
    HELP "ID-number must be entered in ID1 and ID2" TYPE=ERROR
    IF ID1=. THEN
      GOTO ID1
    ELSE
      GOTO ID2
    ENDIF
  ENDIF
END
```

## AUTOJUMP

无条件跳转到另一个变量。跳转会在使用者离开当前变量时执行。参见 [JUMPS](#)。

当然，你也可以不指定变量名，而是在 **AUTOJUMP** 后面使用 **END** 或 **WRITE**。**AUTOJUMP END** 使光标直接跳转到该条记录的最后一个变量。**AUTOJUMP WRITE** 使“Write record to disk?”（是否存盘？）的对话框弹出。点击  后，程序进入下一个或一条新的记录。

使用 **AUTOJUMP SKIPNEXTFIELD** 命令，当完成当前变量的录入后，程序会越过下一个变量，直接跳转到其后的变量。例如：

```
AUTOJUMP [name of field to jump to]
AUTOJUMP END
AUTOJUMP WRITE
AUTOJUMP SKIPNEXTFIELD
```

## **AUTOSAVE**

当记录被修改，欲离开时，程序就会询问“Save record to disk?”（是否存盘？）。我们可以在 **CHECK** 文件中加入命令 **AUTOSAVE**，程序就不会再提示这个问题。当然，这个命令也可以作为程序参数进行设置。不过，使用 **AUTOSAVE** 时也要小心。少了这种提示，即使覆盖了当前记录，也没有任何警告出现。例如：

```
BEFORE FILE
  AUTOSAVE
END
```

## **AUTOSEARCH**

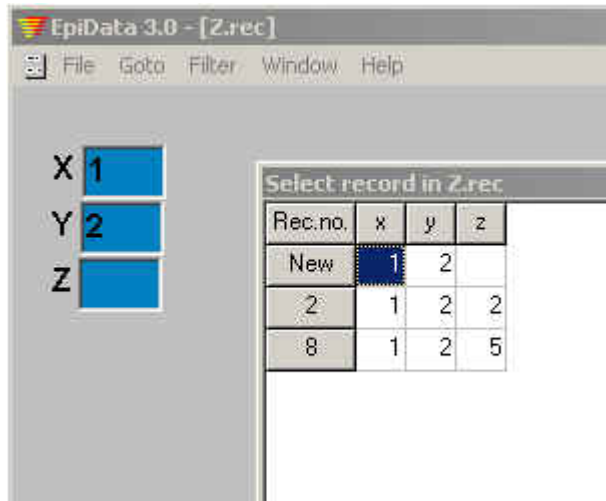
**AUTOSEARCH** 具有查找功能，可以在已有的数据库中查找与当前记录录入数值相同的记录。查找功能可以只在一个变量上进行，也可以在几个变量上同时进行。不过，我们这里建议在设置了 **AUTOSEARCH** 的变量上同时设置 [KEY](#) 命令。

```
V1
  KEY
  AUTOSEARCH
END
```

如果没有发现匹配的记录，数据录入将会继续。当发现匹配记录时，程序会弹出一个信息框，询问你：是希望打开并编辑这条已有的记录，还是继续在新记录中录入？选择“**Yes**”，程序会跳转到匹配记录上；选择“**Cancel**”，你可以继续在新记录中录入；选择“**No**”，你将无法继续进行任何操作。

如果同时设置多个变量，（例如，ID 和 v10 前后两个连续变量），在 v10 变量上设置“**AUTOSEARCH ID v10**”（记住在 ID 和 v10 两个变量上都要设置 **KEY** 命令）。在录入完 v10 后，EpiData 会在已有的记录中查找 ID 和 v10 两个变量上数值都相同的记录。

如果设置命令“**AUTOSEARCH LIST v10**”，程序会弹出所有匹配记录的列表。在列表中，你可以使用鼠标点击或用箭头键移动到你想继续录入的记录上。下面一张图中显示的是，当输入 X=1，Y=2 时，程序弹出一张表，其中列出了所有 X=1 且 Y=2 的记录。利用 **AUTOSEARCH** 功能，可以防止相同记录重复录入。



## BACKUP

该命令会在 REC 文件关闭时，将 REC 文件（及其它相关文件）拷贝到一个指定的文件夹中。程序会自动覆盖该文件夹中已有的任何文件。任何与 REC 文件重名的 CHK、QES、NOT 文件也会被拷贝。BACKUP 命令必须放在 AFTER FILE 命令块中。如果你的备份路径中含有空格，请用引号将路径前后括起来。BACKUP 只会在数据被修改后启动。

如果你用 RELATE 命令关联了一些数据库，那么所有打开的 REC 文件加上相关的文件都会被拷贝。例如：

```
AFTER FILE
```

```
    BACKUP F:\backup\dataentryprojects
END
```

## BEEP

调用该命令，系统会发出声音。BEEP 有三种类型的声音可选。调用该命令时需要进行一些调试，因为计算机间的声音设置都不太相同。在没有限定的情况下，程序会采用标准的嘟嘟声。在声音设置系统中，会用惊叹号(!)表示 CONFIRMATION, 用问号(?)表示 WARNING。BEEP 可以和 IF...THEN...ENDIF 命令块结合起来用。例如：

```
BEEP
BEEP CONFIRMATION
BEEP WARNING
```

当然，你也可以不设置 BEEP 命令，在菜单 **File** → **Options** → **Advanced** → **Sounds** 中，勾选 Warnings during data entry。

## BEFORE ENTRY

设置一个命令块，其中的命令是在当前变量激活，但尚未录入时执行。BEFORE ENTRY 是一个块命令，必须以 END 结束。如果命令出现在变量块中，

既没有放在 **AFTER ENTRY** 命令块中,也没有放在 **BEFORE ENTRY** 命令块中,这些命令将被视作 **AFTER ENTRY** 块命令。例如:

```
BEFORE ENTRY
  <command>
  <command>
  ...
END
```

### **BEFORE FILE**

设置一个命令块,其中的命令是在数据库被打开但尚未开始录入时执行。参见 [AFTER FILE](#)。在 **BEFORE FILE** 块中,你可以定义数据库中使用的一些临时变量。例如:

```
BEFORE FILE
  HELP "Welcome to my data file"
  DEFINE varAge ###
  DEFINE varRefDate <dd/mm/yyyy>
END
```

### **BEFORE RECORD**

设置一个命令块,其中的命令是在一条新记录开始,但尚未录入任何数据时执行。参见 [AFTER RECORD](#)。例如:

```
BEFORE RECORD
  VarAge=33
END
```

### **CLEAR**

清除指定变量的内容。如果在 **CLEAR** 后面没有指定变量名,那么清除的就是含有这条命令的变量的内容。例如:

```
CLEAR
CLEAR field5
```

### **CLEAR COMMENT LEGAL**

清除变量上的 **COMMENT LEGAL** 的定义。这个命令可以和条件 **COMMENT LEGAL** (即,在 **IF...THEN** 结构中定义的 **COMMENT LEGAL**) 联合使用。

### **COLOR**

这条命令是用以改变数据录入表格的背景颜色、表格中的文本颜色、或数据录入变量的颜色。**EpiData** 可以处理 **Epi Info 6** 的颜色编码,所以可以读取 **Epi Info** 的 **CHECK** 文件。同时, **EpiData** 也能处理表示颜色的文字。



选择菜单上的 **Tools** → **Color table**, 让我们看一看 EpiData 可以选择的颜色。

■ 使用表示颜色的文字:

```
COLOR DATA textcolor [backgroundcolor [highlightcolor]]
COLOR QUESTION textcolor [backgroundcolor of question]
COLOR BACKGROUND form_backgroundcolor
```

■ 使用 Epi Info 的颜色编码:

```
COLOR DATA code
COLOR QUESTION code
COLOR BACKGROUND code
```

例如:

COLOR DATA BLUE WHITE LIME	白色背景、蓝色文本、当变量激活时为石灰色背景
COLOR DATA BLACK YELLOW	黄色背景、黑色文本。激活后突出显示的颜色在 <b>Options</b> 中有定义。
COLOR DATA 31	藏青色背景、白色文本（可查看 <b>tools</b> → <b>color table</b> → <b>Number codes</b> ）

我们可以把颜色设置的命令放在 **BEFORE FILE** 命令块中，语法示范:

```
BEFORE FILE
  COLOR BACKGROUND SILVER
  COLOR QUESTION RED
  COLOR DATA BLACK YELLOW AQUA
END
```

根据这个颜色设置，数据表格的颜色如下显示。整个表格的背景是银灰色，注释性文字（变量名称）的颜色是红色，变量录入框的背景颜色是黄色，录入的数据为黑色，激活窗口的颜色为浅蓝绿色。



## COMMENT (\*)

注释行必须以\*字符开始。以\*字符开始的行在执行 CHECK 文件命令时将被忽略。

## COMMENT LEGAL

该命令与 [LEGAL](#) 都是对变量允许录入的内容进行限定，但是使用 COMMENT LEGAL 后，在录入数据的过程中，按 **F9** 或数字键盘上的 **+** 键，程序会自动弹出一个允许值及其涵义列表。



COMMENT LEGAL 有四种不同的形式:

1) 块命令	COMMENT LEGAL 1 Denmark 2 Somalia 3 Other END
2) 引用另一个变量中设置的 COMMENT LEGAL 命令	COMMENT LEGAL USE <i>fieldname</i>
3) 引用标注块中定义的一组数值标签	COMMENT LEGAL USE <i>labelname</i>
4) 引用含有数值及其标签的数据库	COMMENT LEGAL <i>filename</i> [ <i>.rec</i> ]

注意，当引用另一个变量中或在标注块中设置的 COMMENT LEGAL 时，必须加上 USE 这个词。当引用一个数据库时，文件扩展名 (.rec) 可以不写。引用的数据库中必须有两个变量，我们必须将这两个变量定义为 [KEY](#) 或 KEY UNIQUE 变量。设置为 KEY1 (或 KEY UNIQUE1) 的变量是数值型变量；设置为 KEY2 的变量是标签变量。

COMMENT LEGAL 也可以在 IF...THEN 结构中使用。例如，当一个变量使用何种数值标签取决于另一个变量的取值时 (分层编码)，我们就可以使用这个命令。可参见下面的例子，或安装文件夹内、sample 文件夹中的实例 HIERARTEST.REC 和 HIERARTEST.CHK，或下面的 [TYPE COMMENT](#) 命令。

如果在 COMMENT LEGAL 命令后加上 SHOW 这个单词，在录入过程中，当前变量一旦被激活，数值标签列表会自动弹出来。

值得一提的是，在设置 **COMMENT LEGAL** 项时，还要考虑与变量类型的搭配。例如，如果你对一个<A >型变量（大写英文字母的字符型变量）设置 **CHECK** 命令为：**mus** “Mouse was the animal”，这个 **CHECK** 文件肯定会被拒绝，因为该变量只能输入大写的 **MUS**，输入小写的 **mus** 是非法的。例如：

```
COMMENT LEGAL
  1 "Male gender"
  2 Female
END
```

```
COMMENT LEGAL
  1 One
  * 2 Two
  3 Three
END
```

（注：2 前是\*开头，因此该行注释将不在列表中显示）

```
COMMENT LEGAL USE [field name] SHOW
```

```
COMMENT LEGAL USE [labelname]
```

下面是一个在 **IF...THEN** 结构中嵌套 **COMMENT LEGAL** 的例子：

{首先选择一个国家}

```
V1
  COMMENT LEGAL
    1 USA
    2 CANADA
  END
END
```

{然后选择州}

```
V2
  IF V1=1 THEN
    COMMENT LEGAL
      1 Alabama
      2 "New York"
      3 Nevada
      4 Oklahoma
      .....
    END
  ENDIF
  IF V1=2 THEN
    COMMENT LEGAL
```

```

1 "Nova Scotia"
2 Quebec
.....
END
ENDIF
END

```

下面是一个 **COMMENT LEGAL** 引用数据库的例子。NAMELOOKUP.REC 是一个数据库（也可以看做是查询表），含有两个变量：ID（整数）和 NAME（字符型变量）。在 CHECK 文件中，我们将 ID 定义为 KEY UNIQUE1，NAME 定义为 KEY2。PATIENTDATA.REC 是另一个数据库，其 QES 文件格式如下：

```

ID      ENTER ID-code of patient #####
HEIGHT Patient's height in kg      ###
WEIGHT Patient's weight in kg      ###

```

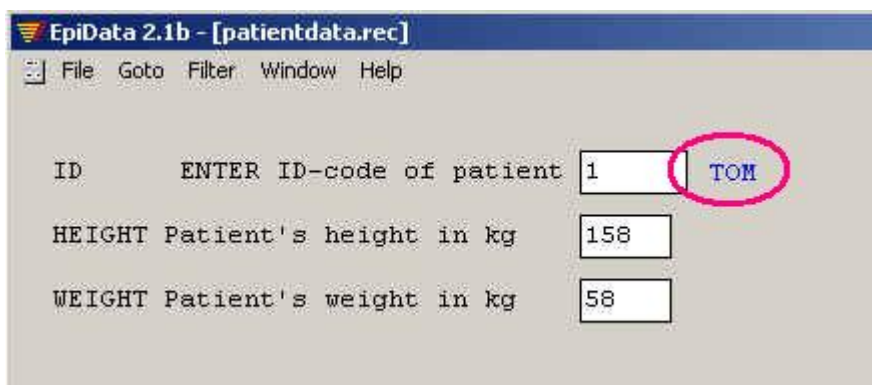
数据库 PATIENTDATA.REC 的 CHECK 文件（PATIENTDATA.CHK）中有以下设置：

```

ID
COMMENT LEGAL NameLookup
TYPE COMMENT
END

```

当在 PATIENTDATA.REC 中录入数据时，只有那些在 NAMELOOKUP.REC 库中有 ID 号的病人才被允许录入。当输完 ID 变量，激活下一个变量时，病人的姓名会自动显示在 ID 变量旁边。



需要指出的是，在你编辑 PATIENTDATA.CHK 中的 **COMMENT LEGAL** 命令前，你必须先在 NAMELOOKUP.REC 数据库中至少录入几个名字；否则会出现错误。因为 NAMELOOKUP.REC 这个数据库作为一个索引表，必须在进行 **COMMENT LEGAL** 设置时已经准备好被“引用”。

另外，你还可以参考 <http://www.epidata.dk/examples.php> 网页中提供的

“bacteria list”实例。

## CONFIRM

通常，当一个变量的允许字符数全部填满后，光标会自动移到下一个变量。如果在 **CHECK** 文件中使用 **CONFIRM** 这个命令，那么当一个变量的允许字符数填满后，只有在录入员键入 **ENTER** 键后，下一个变量才会被激活。参见 [CONFIRMFIELD](#)。例如：

```
BEFORE FILE
  CONFIRM
END
```

## CONFIRMFIELD

与 **CONFIRM** 的功能相同，但是 **CONFIRM** 设置的是数据库中的所有变量，而 **CONFIRMFIELD** 只设置一个变量。这个命令可以只在一个变量块中出现。例如：

```
V1
  CONFIRMFIELD
  MUSTENTER
END
```

## COPYTOCLIPBOARD

你可以在 **BEFORE/AFTER ENTRY** 命令块中使用这个命令，将一个字符串和/或一个或更多变量上的内容拷贝到 **Windows** 的剪贴板上。拷贝的内容可以粘贴到其它 **Windows** 应用程序中。

拷贝的文本需要用双引号括起来。在一个变量名前或一个用 **DEFINE** 定义的变量的名字前使用 **@**，可以将变量中的内容拷贝到剪贴板上。例如：

```
V10
  AFTER ENTRY
    V20=String(V5)+"-"+String(V10)
    COPYTOCLIPBOARD "Codenummer=@V20"
  END
END
```

```
COPYTOCLIPBOARD "Field V1 equals @V1 and V2 equals @V2"
```

## DEFINE

你可以利用这个命令定义新的、临时变量。这些临时变量可以用来保留计算过程中的中间值，然后将这些中间值从一个记录传到下一个记录，或者将一个文件中的一个变量的内容传到另一个相关文件中。

**DEFINE** 命令有两个选择项，**CUMULATIVE** 或 **GLOBAL**。设置了 **GLOBAL** 或 **CUMULATIVE** 的变量，前一条记录的内容会被保留到下一条记录。另外，设

置了 GLOBAL 的变量的内容，还可以保留到一个关联数据库的相关记录上。如果这两个选项都不写，则开始新的记录后，临时变量会被设回缺失值（参见 [RELATE](#)）。

用 DEFINE 定义的变量名最多 16 个字符。临时变量不会被保存在数据库中。在关联的数据库中，相同的 DEFINE 可以用在几个 CHECK 文件中。重复的 DEFINE 命令会被忽略。例如：

```
DEFINE MyTempVar #####
DEFINE varSurname <A    > CUMULATIVE
DEFINE tempDate <dd/mm/yyyy>
DEFINE varCity GLOBAL
```

（注：第一句定义了一个有 4 位整数的变量，变量名为 MyTempVar）

另外，你还可以参考 <http://www.epidata.dk/examples.php> 网页中提供的“Household, Person, Visit”实例。

## EXECUTE

在 CHECK 文件中，你可以使用 EXECUTE 运行外部程序。它的语句可以写作：

```
EXECUTE programname [parameters] WAIT | NOWAIT [HIDE]
```

WAIT 是让 EpiData 程序暂停，直到被调用的程序关闭后再继续运行。NOWAIT 则是在运行指定的程序的同时，EpiData 程序可以继续运行（例如，运行 AFTER ENTRY 命令块中余下的命令）。HIDE 命令是可选的，如果写上它，表示被调用的程序运行时，不必在屏幕上显示运行过程。注意，如果被调用的程序可以自动关闭，而不需要使用者干预，则可以使用这个选项。

被调用的程序可以是 EXE 文件（即应用程序），也可以是文档、图像文件，等等。如果程序名或参数中含有空格，必须用双引号括起来。例如：

1. 打开记事本，EpiData 暂停，待记事本关闭后再继续运行。

```
EXECUTE c:\windows\notepad.exe WAIT
```

2. 打开文档“ICD10 list.doc”。

```
EXECUTE "c:\my documents\ICD10 list.doc" NOWAIT
```

3. PICTNUM 是一个整数变量。如果使用者输入 3，系统会使用默认的图像浏览程序打开“d:\pics\picture3.jpg”，EpiData 暂停，待图像浏览程序关闭后再继续运行。@tmpName 表示用变量 tmpName 的内容替代。

```
PICTNUM
```

```
DEFINE tmpName _____
```

```

AFTER ENTRY
  LET tmpName="d:\pics\picture"+pictnum+".jpg"
  EXECUTE @tmpName WAIT
END
END

```

4. 下面这个命令块首先将数据备份到指定的文件夹 (c:\temp\), 然后程序会询问是否创建一个 zip 压缩文件, 文件名为当日的日期。(Pkzip 必须在寻找路径中)。

```

AFTER FILE
  DEFINE tmpz _____
  BACKUP c:\temp\
  HELP "Create zip file? 1:yes 2:no" keys="12"
type=CONFIRMATION
  if resultvalue=1 then
    let tmpz=string(day(today))
    if month(today)<10 then
      tmpz=tmpz+"0"+string(month(today))+string(year(today))+".zip"
    else
      tmpz=tmpz+string(month(today))+string(year(today))+
        ".zip c:\temp\*.*"
    endif
  endif
  execute pkzip.exe "@tempzip -r -u" wait hide
END

```

## EXIT

停止执行并离开一个命令块 (例如, AFTER ENTRY 命令块)。使用 EXIT, 可以使程序不必再继续执行 IF...THEN-ELSE 中 EXIT 后面长长的命令。例如:

```

D1
AFTER ENTRY
IF D1=. THEN
  HELP "A date must be entered" TYPE=ERROR
  GOTO D1
  EXIT
ENDIF
IF D1>TODAY THEN
  HELP "A future date is not valid" TYPE=WARNING
  GOTO D1
  EXIT
ENDIF
IF D1<TODAY-365 THEN
  HELP "@D1 is more than one year ago. Please re-enter" TYPE=ERROR
  GOTO D1
  EXIT
ENDIF
* The following commands are only executed
* if all above checks indicates a valid date
D2=D1+14
V4=Year(D1)
END
END

```

如果使用 EXIT 命令，当我们在 D1 变量处输入 10/31/1999（假定当前日期为 09/09/2003），显然这个日期早于当前日期 1 年以上。程序会立即终止，不再继续运行 AFTER ENTRY 命令块中后面的计算命令：D2=D1+14，V4=Year(D1)。相反，如果去掉 EXIT 这个命令，当输入非法日期后，尽管程序会自动弹出警告框，而光标也仍然停留在 D1 变量处，但后面的计算命令也会被执行，D2 和 V4 变量处都显示出对应的计算结果。

## GOTO

令光标移动到指定的变量上去。

```
GOTO field10
GOTO WRITE
```

GOTO WRITE 是 GOTO 命令的一种特殊形式，它会终止进一步的数据录入，程序会自动弹出对话框，询问 **Save record to disk?**（是否存盘？）。

如果在 GOTO 命令后没有指定变量名，则跳转功能会一直在含有该命令的变量上发挥作用，也就是始终在当前变量上死循环，除非用鼠标激活其它变量。例如，我们在 y 变量上设置：

y



```
GOTO
END
```

则程序实际执行的命令是：

```
Y
  AFTER ENTRY
    GOTO y
  END
END
```

## HELP

使用该命令可以使程序自动弹出一个消息框，消息框中的内容可以由使用者自行指定。如果想继续数据录入，使用者必须按 **OK** 键。有 4 种不同类型的消息框：信息（**information**）框、警告（**warnings**）框、确认（**confirmation**）框和错误（**error**）框。如果没有指定消息框的类型，程序会按信息框来显示。指定消息框的类型，无需写类型的全称，只用第一个字母表示即可（例如，“**C**”表示确认框）。

插入“**\n**”，可以使其后面的文字另起一行显示。当前变量的数值或用 **DEFINE** 定义的变量可以在帮助消息中显示，以 **@ 变量名** 表示。如果要显示 **@** 这个字符（例如，在 **email** 地址中），你可以输入 **@@**。

**HELP** 命令的一种特殊用法是要求使用者按一组键中的一个来继续录入。它的语句是：

```
HELP "Do you want to continue (y/n)?" KEYS="YN"
```

在这个例子中，程序会弹出一个消息框。当使用者按 **Y** 或 **N** 键时，消息框消失，可以继续录入。操作情况被保存在预先定义好的变量 **RESULTLETTER** 和 **RESULTVALUE** 中。继续上面的例子，如果使用者按 **N** 键，则 **RESULTLETTER** 为字符“**N**”，**RESULTVALUE** 为数字“**2**”（因为在 **KEYS** 系列中，**N** 排在后面）。例如：

```
HELP "This is the information text"
HELP "This is a \n two-line warning box" TYPE=WARNING
HELP "This is also a warning box" TYPE=W
HELP "Please confirm" TYPE=CONFIRMATION
HELP "You made an error!" TYPE=ERROR
HELP "The field V1 is equal to @V1"
HELP "EpiData's e-mail address is Info@@EpiData.dk"
```

下面是一个使用 **KEYS** 的例子：

```
V1
  AFTER ENTRY
```

```

HELP "Select option:\n A. Enter personal information\n B.
Enter occupational information\n C. Enter health history"
KEYS="123" TYPE=CONFIRMATION
  IF RESULTLETTER="A" THEN
* The user pressed the key "A"
    GOTO V100
  ELSE IF RESULTVALUE=2 THEN
* The user pressed the key "B"
    GOTO V200
  ENDIF
  IF RESULTVALUE=3 THEN
    GOTO V300
  ENDIF
END
END

```

## HIDE, UNHIDE

隐藏或显示一个变量。当设置一个变量为隐藏时，变量在屏幕上显示的颜色会发生改变，同时使用者无法在该变量上录入数据。例如：

```

HIDE field5
UNHIDE field5

```

如果 **HIDE** 或 **UNHIDE** 命令后面没有指定变量名，程序会隐藏或显示含有这个命令的变量。这种情况下，**HIDE** 或 **UNHIDE** 命令会被当做 **AFTER ENTRY** 命令块的一部分来处理。换句话说，只有在该变量录入工作完成后，才会执行隐藏或显示功能。例如，我们在 *y* 变量上设置：

```

Y
  HIDE
END

```

则程序实际执行的命令是：

```

Y
  AFTER ENTRY
  HIDE y
  END
END

```

## IF...THEN

**IF...THEN** 命令的结构是：

```

IF <expression> THEN

```

```
<commands to execute if expression is true>
ENDIF
```

或者是:

```
IF <condition expression> THEN
  <commands to execute if condition is true>
ELSE
  <commands to execute if condition is true>
ENDIF
```

记住 IF...THEN 命令块必须以 ENDIF 结束。执行的命令可以写成几行，其中也可以包含其它的 IF...THEN 命令（即嵌套的 IF 语句）。

条件表达式中必须有布尔逻辑（Boolean）结果（即“真”或“假”）。条件表达式中可使用的运算符和函数可参见[运算符和函数](#)一节。条件表示式可以包含几个部分，之间用 AND 或者 OR 连接。每部分必须用圆括号括起来（这一点不同于 Epi Info）。例如，将语句写成：

```
IF Field2>5 AND Field3<10 THEN
```

是不允许的。正确的写法是：

```
IF (Field2>5) AND (Field3<10) THEN
```

IF 语句的设置中如果有错误，该语句内容在数据录入过程中会被忽略。为了能即时地察觉到 IF 语句中的错误，你可以在菜单 **File** → **Options** → **Advanced** → **Error messages** → 勾选 Show errors that occur in calculations in the CHECK file during data entry。例如：

```
IF field1>10 THEN
  GOTO field10
ENDIF
```

```
IF (Cos(field1)*Sin(field1)<0.3) AND (field2<>0) THEN
  IF field2<field3 THEN
    HELP "Something is wrong."
    GOTO
  ENDIF
ELSE
  Field4=Tan(field1)
  GOTO field23
ENDIF
```

```
IF field10=. THEN
```

```
field11=.
field12=0
date1="12/03/2001"
ENDIF
```

## INCLUDE

在 CHECK 文件的当前位置上，打开一个文件。而这个文件中含有部分 CHECK 语句，可以用于当前位置。该命令要打开的文件不能是当前的这个 CHECK 文件。含有 INCLUDE 命令的 CHECK 文件不能用 **Checks** → **Add/Revise** 编辑，只能用编辑器编辑。例如：

```
LABELBLOCK
  LABEL numbers
  Include test6.inc
  4 four
  5 five
  6 six
END
END
```

其中，文件 test6.inc 的内容为：

```
1 one
2 two
3 three
```

INCLUDE 后面除了文件名以外，可以加上文件所在的位置，如果不指定文件的路径，程序默认这个文件就在当前文件夹下。文件的扩展名必须指定。

## JUMPS

有条件地跳转到其它变量上。JUMPS 是一个块命令，必须以 END 结束。在 JUMPS 和 END 之间，你需要指定：1) 当前变量某个可能录入的数值；2) 与指定数值对应的、跳转的目标变量名。参见 [AUTOJUMP](#)。

你也可以不指定变量名，而是写 END 或 WRITE。使用 END，光标会直接跳到该条记录的最后一个变量。使用 WRITE，程序会自动弹出 **Write record to disk?**（是否存盘？）对话框。另外，还可以使用 SKIPNEXTFIELD 这个单词。光标会从当前变量越过下一个变量，直接激活之后的一个变量。

一般来说，使用 JUMPS 时，起跳变量和跳转的目标变量间的其它变量中的内容应该被清除或设置为缺失值。为了保证这些变量不会被填上无关的数据，我们可以用 JUMPS RESET 命令，将这些变量中的内容统统清除。如果在 RESET 后面加上一个字符，则这些无关的变量就会被这个字符填充。例如，JUMPS RESET 9，程序会自动将这些无关变量填充为 9。AUTOJUMP 命令中不能使用 RESET。另外，如果用的是 WRITE 选项，RESET 的功能也会被忽略。例如：

```
JUMPS
  1 V5
  2 V10
  3 END
  4 WRITE
END
```

```
JUMPS RESET
  1 V5
  2 SKIPNEXTFIELD
END
```

```
JUMPS RESET 9
  1 V5
  2 V30
END
```

## KEY

该命令的语句是：

```
KEY {UNIQUE} {keynumber}
```

该命令是为其所设置的变量创建了一个索引。索引实际上是另外一个文件 (\*.eix)，它可以加速寻找含有某个特点变量值的记录的过程。如果在 **KEY** 后面加上 **UNIQUE**，索引就是一种唯一标识。例如，如果设置姓名为 **KEY UNIQUE**，则诸如 **JONES** 这样的姓名在所有记录中只能出现一次。这个要求似乎不太现实，所以姓名不适合作为 **KEY UNIQUE** 变量。**KEY UNIQUE** 变量应该含有唯一标识的号码，每个号码只对应一条记录。任何变量都可以加上 **KEY**（但没有 **UNIQUE**）命令，允许一条以上的记录有相同的数值。另外，像姓名、县或年龄等变量也可以设置 **KEY**。使用 **KEY** 主要有 4 种用途：

- 1) 如果数据库中有成百上千的记录，在录入 (**ENTER**) 窗口中对设置了 **KEY** 的变量进行查询，要比查询不设置 **KEY** 的变量快上好几倍；
- 2) 在 **LIST DATA** 功能中可以依此变量指定一种排序次序；
- 3) 保证每个 **ID** 号只能录入一次（只有 **KEY UNIQUE** 有这项功能）；
- 4) 在数据录入过程中，允许数据库间建立关联 (**RELATE**)。

你可以指定 **KEY** 后的号码 (*keynumber*)，也可以不指定。这个号码规定了索引文件中 **KEY** 的顺序。如果在 **KEY** 后，你没有指定号码，那么 **KEY** 在索引文件中的顺序就按照你设置命令的先后顺序依次排列。当你再次打开 **CHECK** 文件时，你会发现，程序已经自动在 **KEY** 命令后给出了号码。

如果设置了 **KEY** 的变量超过 30 个字符，则在 **KEY** 中只会使用头 30 个字符。在同一个数据库中，你最多可以定义 10 个 **KEY** 变量。当打开数据库准备录入时，索引文件 (\*.eix) 会自动生成。如果记录数或设置了 **KEY** 的变量数与

索引文件不匹配,索引会自动重新建立。如果你主动想让 EpiData 重新建立索引,可以使用 **Tools**→**Rebuild Indexes**。例如:

```
IDNUMBER
  KEY UNIQUE 1
END
```

```
NAME
  KEY
END
```

### **LABEL**

参见 [LABELBLOCK](#)。

### **LABELBLOCK**

标注块中可以包含多组数值标签的定义。该命令块必须以 **END** 结束,且不能作为变量块的一部分。每组数值标签必须以命令 **LABEL** 开始,接下来是标签的名字,最后是命令 **END**。在标注块中定义的这些数值标签可以用下述命令调用:

**COMMENT LEGAL USE** [*labelname*].

例如:

```
LABELBLOCK
  LABEL yesno
    1 Yes
    2 No
  END
  LABEL sex
    1 Male
    2 Female
    9 "Unknown sex"
  END
END
```

### **LEGAL**

设置变量允许录入的数值。**LEGAL** 是一个块命令,必须以 **END** 结束。如果多个变量使用相同的允许值设定,你不必在每个变量中都重复设定,可以引用其它变量上的设置。使用的命令是:

**LEGAL USE** field name

例如:

```
V1
  LEGAL
    2
    4
    6
    8
  END
END
```

```
V2
  LEGAL USE V1
END
```

## LET

令某个变量等于某个数值或某个计算的结果，或者等于用 **DEFINE** 定义的变量。在计算式或其它表达式中可以使用的运算符和函数可参见本文的[运算符和函数](#)一节。

**LET** 表达式中如果存在错误，数据录入过程中会忽略该语句。为了能即时地察觉到 **LET** 语句中的错误，你可以在菜单 **File** → **Options** → **Advanced** → **Error messages** → 勾选 **Show errors that occur in calculations in the CHECK file during data entry**。

注意，如果 **LET** 表达式中涉及多个变量，而其中任何一个变量为缺失值，**LET** 表达式都得出结果。**LET** 这个单词可以写，也可以不写，下面两种表达方式的意思都是一样的：

```
LET field5=(field2/field3)+INT(field4)
field5=(field2/field3)+INT(field4)
```

其它例子：

LET date1="14/09/2000"	令日期变量 <b>date1</b> 等于 14/09/2000
LET v1=.	将变量 <b>v1</b> 设置为缺失值
LET b1=((15/2)>4)	设置布尔逻辑变量为 “Y”
LET b2="Y"	设置布尔逻辑变量为 “Y”
LET b3=False	设置布尔逻辑变量为 “N”
LET v3=integer(copy(s2,1,2))	提取该字符变量中的头两个字符，将这个结果应用于一个整数变量
LET Text1="Q"+String(Number)	如果数字等于 14，则令字符变量 <b>Text1</b> 等于 Q14

注意，当你在 **BEFORE FILE** 或 **BEFORE RECORD** 中设置 **LET** 命令时，当前记录的状态不会发生改变；而当 **LET** 在第一个变量的 **BEFORE ENTRY** 中出现时，当前记录的状态会发生改变。当在第一个变量的 **AFTER ENTRY** 命令块中出现时，只当光标移到下一个变量时，记录的状态才会改变。

## MISSINGVALUE

该命令有三种语句：

1. 在变量块中可以使用：`MISSINGVALUE x[y[z]]`
2. 在 `BEFORE FILE` 命令块中可以使用：`MISSINGVALUE ALL x[y[z]]`
3. 在 `BEFORE FILE` 命令块中可以使用：`MISSINGVALUE field1-field5, field6... x[y[z]]`

你可以用 `MISSINGVALUE` 命令定义 1 个、2 个、或 3 个不同的数值，各自有特殊的涵义。例如：`9`=问卷中未填，`8`=无关变量。设置的数字可以是 `0-9` 任意数字。`MISSINGVALUE ALL` 是给数据库中所有的数值型变量都定义 1 个、2 个、或 3 个不同的缺失值。`MISSINGVALUE field1...` 可以用在 `BEFORE FILE` 命令块中，给某几个变量定义缺失值。`MISSINGVALUE x y z` 可以用在变量块中，只对当前变量进行设置。如果同时设置了 `MISSINGVALUE ALL` 和变量块中的 `MISSINGVALUE`，则程序优先考虑变量块中的定义。

在数据录入过程中，设置了 `MISSINGVALUE` 的变量中可以输入减号“-”。当离开该变量时，程序会自动将“-”转换为 `MISSINGVALUE` 定义的头一个缺失值。`MISSINGVALUE` 可以扩展变量中设置的 `RANGE`、`LEGAL` 或 `COMMENT LEGAL` 的定义。如果在变量块中定义了“`MISSINGVALUE 9`”，则除了已经定义的“`RANGE 1-5`”外，还允许录入 `9`。例如：

{在 `BEFORE FILE` 命令块中}

```
MISSINGVALUE ALL 9 8
```

(注：定义所有数值型变量的第一个缺失值为 `9`，第二个缺失值为 `8`)

```
MISSINGVALUE V1-V4,V10,V20 9 8 7
```

(注：定义变量 `V1`、`V2`、`V3`、`V4`、`V10`、`V20` 的缺失值)

{在变量块中}

```
MISSINGVALUE 9 8 7
```

(注：给当前变量定义 3 个不同的缺失值)

在输出到 `Stata 8` 数据库时，`MISSING VALUE` 会按 `Stata` 的格式输出，即 `.a`、`.b` 或 `.c`。同样，输入 `Stata 8` 数据库时，如果发现 `.a`、`.b` 或 `.c` 格式，将按 `MISSING VALUE` 处理。

## MUSTENTER

保证当前变量必须录入。如果数据缺失，程序会一直停滞在该变量上。所以，建议大家对设置了 `MUSTENTER` 的变量定义一个缺失值的编码，这样在遇到特殊情况时，可以较灵活的应对。例如：

```
MUSTENTER
```



## NOENTER

不允许在当前变量上录入。如果某个变量是通过其它变量计算得来的，不必录入，可以使用这个命令。例如：

```
NOENTER
```

## QUIT

终止数据录入，关闭数据录入表格。

## RANGE

为变量定义一组允许录入的数值。例如：

RANGE -5 5	允许录入-5 到 5 间的数值，包括-5 和 5
RANGE -INFINITY 99	允许值为<100
RANGE 100 INFINITY	允许值为≥100
RANGE 1/3/2001 31/3/2001	允许录入 2001 年 3 月份中的日期

如果你在同一个变量中同时也设置了 **COMMENT LEGAL** 或 **LEGAL**，那么一定要留心，有可能设置的规则是矛盾的。例如，如果允许范围 (**RANGE**) 指定的是 1-5，而允许值 (**LEGAL**) 指定的是 7 和 8，这时 1-5，及 7、8 都可以录入。

## RELATE

在数据录入过程中，你可以用 **RELATE** 命令将两个不同的数据库关联起来。在录入主（母）数据库时，当遇到 **RELATE** 命令，程序会自动进入另一个关联（子）数据库继续数据录入。**RELATE** 语法可以写作：

```
RELATE identifier_field filename [1]
```

上面语句中的“**identifier\_field**”是在主（母）数据库和关联（子）数据库中都存在的标识变量的变量名。含 **RELATE** 命令的变量与标识变量不一定相同。在主（母）数据库中，必须设置标识变量为 **KEY UNIQUE**；而在关联（子）数据库中，应设置标识变量为 **KEY**。

如果主数据库中的每条记录分别只对应关联数据库中的一条记录，那么 **RELATE** 命令后面应该加上数字“1”。当然，主数据库中的每条记录也可以对应关联数据库中的若干条记录，记录之间都有相同的标识变量。在这种情况下，使用者可以按 **F10**、**Ctrl+R** 或按关联数据库窗口的关闭按钮，即可返回主数据库。

如果关联数据库中有 **BEFORE FILE** 命令块，那么每次进入该关联数据库时，都会执行该命令块中的命令。如果一个数据库的 **CHECK** 文件中含有一个或多个 **RELATE** 命令，当打开该数据库时，所有关联数据库也会同时被打开。通过屏幕下方的小标签，我们可以看到或点击转换到各个关联的数据库中。你可以随时浏览关联数据库，但是这时它们只是处于只读状态；只有主数据库中的 **RELATE** 命令才能调用激活它们。当主数据库被关闭后，所有的关联数据库即被

关闭。直接关闭关联数据库，实际上并不能真正的关闭它，而只是激活其主数据库。你可以参考 <http://www.epidata.dk/examples.php> 网页中提供的“Household, Person, Visit”实例。

## REPEAT

设置了 REPEAT 的变量，在新的记录中，程序会自动复制前一条记录的内容到当前记录。例如：

```
REPEAT
```

## TOPOFSCREEN

这个命令可以用来模拟翻页，即当光标移动到该变量时，不管该变量现在位于屏幕的哪个位置，程序会自动将录入表格上移，使该变量位于屏幕的最顶端。在命令后面增加一个数字，表示当该变量移到屏幕顶端时，距离页面顶端的行数。例如：

```
TOPOFSCREEN  
TOPOFSCREEN 2
```

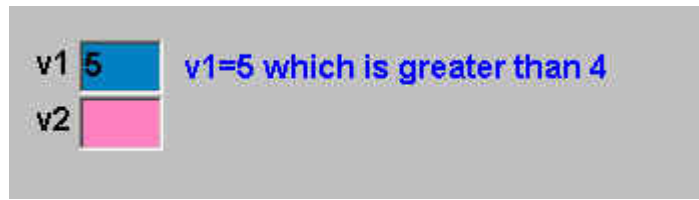
## TYPE

利用 TYPE 命令，在数据录入过程中，可以在录入变量框旁边显示指定的文字。例如：

```
TYPE "text"[color]
```

如果没有指定颜色，默认的颜色为蓝色。你也可以指定其它颜色，参见 [Tools](#) → [Color table](#)。用 DEFINE 命令定义的临时变量当前的数值也可以在 TYPE 信息中显示，表示方法是“@变量名”。如果想显示@字符（例如，在 e-mail 地址中），你可以键入@@。例如：

```
V1  
  AFTER ENTRY  
    IF V1<5 THEN  
      TYPE "Smaller than 5" RED  
    ELSE  
      TYPE "V1=@V1 which is greater than 4"  
    ENDIF  
  END  
END
```



## TYPE COMMENT

这个命令可以用于设置了 **COMMENT LEGAL** 的变量。当完成当前变量的录入，光标移到下一个变量时，与输入值对应的数值标签可以显示在变量录入框的右侧，或者根据定义，显示在一个指定的字符变量上。这个功能可以帮助录入员确定是否输入了正确的数值。“**TYPE COMMENT 变量名**”可以用来替代 Epi Info 中的 **CODES/CODEFIELD** 命令。注意，**TYPE COMMENT** 不能放在 **BEFORE / AFTER ENTRY** 命令块中。

默认的颜色为蓝色。你也可以指定其它颜色，参见 [Tools](#) → [Color table](#)。例如：

```
TYPE COMMENT [color]
TYPE COMMENT field name
TYPE COMMENT ALLFIELDS [color]
```

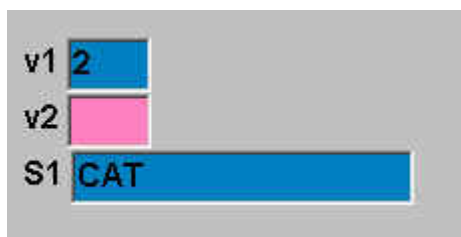
例如：

```
V1
COMMENT LEGAL
1 Dog
2 Cat
3 Lion
4 Rat
END
TYPE COMMENT YELLOW
END
```

如果在 V1 变量上输入 2，当光标离开该变量时，“Cat”这个词会显示在变量的右边。



如果使用语句“**TYPE COMMENT S1**”，则“Cat”这个词会出现在录入变量 S1（字符型变量）上。



我们建议大家在使用分层编码时，使用 **TYPE COMMENT** 这个命令（参见 [COMMENT LEGAL](#)）。

### TYPE STATUSBAR

在录入有很多变量的调查表时，程序需要不断的翻页。这时，最好是能在每页屏幕上都显示当前录入的记录号。在某一个变量中设置 **TYPE STATUSBAR**（只能设置一个变量），则该变量的当前值会显示在窗口下方的状态条上。

另外，我们可以在命令中加上一段解释性的文字（例如，对应的变量名）。默认的颜色为蓝色，你也可以指定其它颜色，参见 [Tools](#) → [Color table](#)。例如：

```
IDCODE
  TYPE STATUSBAR "ID number=" RED
END
```

状态条会显示：



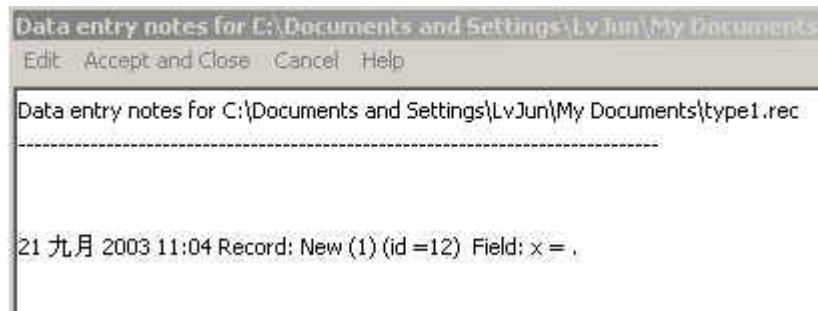
或者：

```
IDCODE
  TYPE STATUSBAR "" RED
END
```

状态条会显示：



如果在数据录入过程中，你打开了数据录入备忘录（[File](#) → [Data Entry Notes](#) 或按 [F5](#) 键），这时，当前显示在状态条上的文字也会记录在备忘录中。这样可以帮助我们把握录入的进度。



## UNHIDE

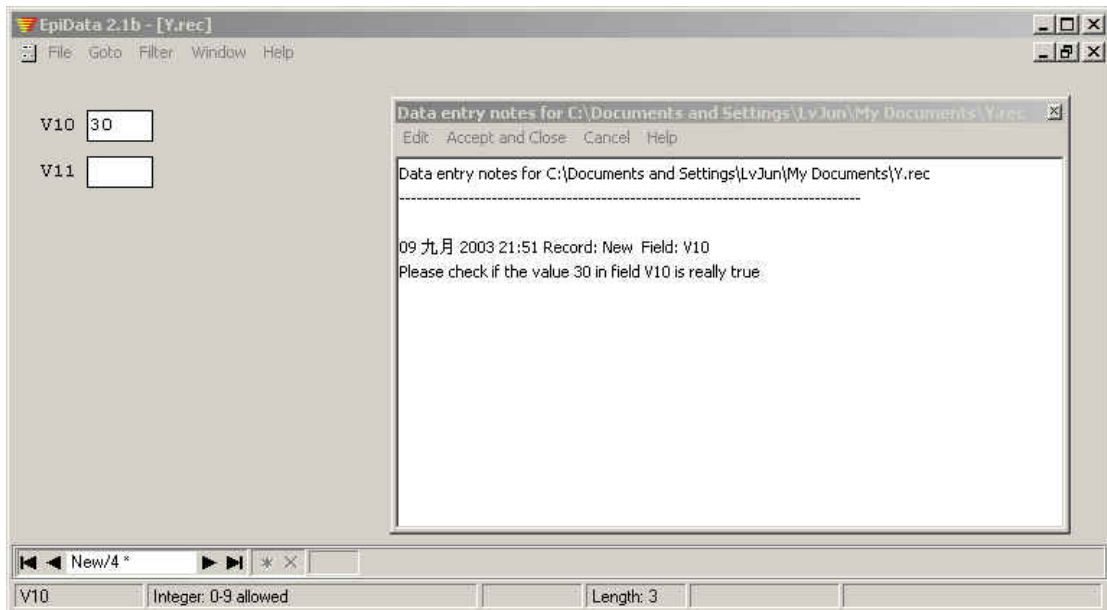
请参见 [HIDE](#)。

## WRITENOTE

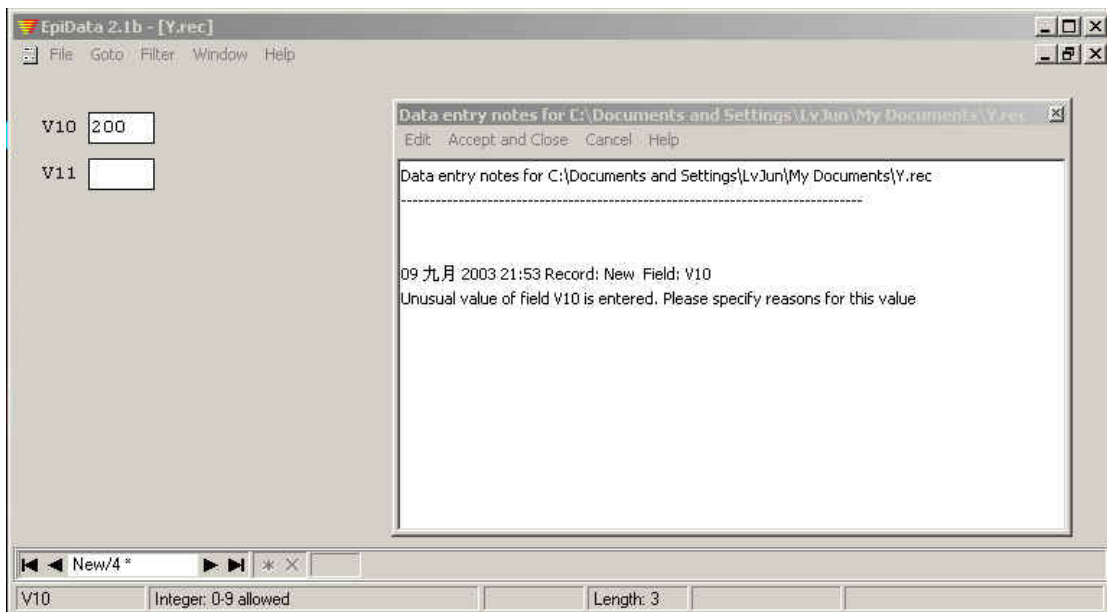
该命令可以在数据录入备忘录文件 ([File](#) → [Data Entry Notes](#) 或按 [F5](#) 键) 中加上注释。在变量名前加上前缀 @, 录入变量当前的内容可以被写进备忘录。加上选项 **SHOW**, 在数据录入过程中, 数据录入备忘录会自动弹出。例如:

```
V10
  AFTER ENTRY
    IF V10>100 THEN
      WRITENOTE "Please check if the value @V10 in field V10 is
really true"
    ENDIF
    IF V10>110 THEN
      WRITENOTE "Unusual value of field V10 is entered. Please
specify reasons for this value:" SHOW
    ENDIF
  END
END
```

当在 V10 变量上输入 30 时:



当在变量 V10 上输入 200 时:



## 4.5 运算符号和函数

这一节中，我们列出了在 IF 命令中的布尔逻辑表达式、以及 LET 命令中的计算式和表达式中可以使用的运算符号和函数。

### 4.5.1 运算符（Operators）

#### 4.5.1.1 算术运算符（Arithmetic Operators）

运算符	运算	数据类型	结果类型
^	指数	整数 浮点	浮点 浮点

运算符	运算	数据类型	结果类型
+	加法	整数	整数
		浮点	浮点
		字符串	字符串
-	减法	整数	整数
		浮点	浮点
		字符串	字符串
*	乘法	整数	整数
		浮点	浮点
		字符串	字符串
/	除法	整数	浮点
		浮点	浮点
		字符串	字符串
div	整数除法	整数	整数
mod	余数	整数	整数

注：浮点指的是小数，与之相对的是整数。

#### 4.5.1.2 逻辑运算符 (Logical Operators)

运算符	运算	表达类型	结果类型
not	否	布尔逻辑	布尔逻辑
and	和	布尔逻辑	布尔逻辑
or	或	布尔逻辑	布尔逻辑
xor	异或	布尔逻辑	布尔逻辑

注：XOR: Exclusive OR<sup>2</sup>

#### 4.5.1.3 关系运算符 (Relational Operators)

运算符	运算	比较值	结果类型
=	等于	可比	布尔逻辑
<>	不等于	可比	布尔逻辑
<	小于	可比	布尔逻辑
>	大于	可比	布尔逻辑
<=	小于等于	可比	布尔逻辑
>=	大于等于	可比	布尔逻辑

关系运算符也可以用于字符串。布尔逻辑结果类型等同于对“真”或“假”的回答，即，真：是或否。

### 4.5.2 函数 (Functions)

#### 4.5.2.1 算术函数 (Arithmetic Functions)

##### ABS(X): FLOAT

计算 x 的绝对值。x 可以是整数或小数。例如：

<sup>2</sup>异或运算符用于判断两个表达式的结果是否不同，在执行逻辑异或运算的过程中，如果两个操作数同为 True 或同为 False，返回结果为 False，否则返回结果为 True。如果有操作数为 Null，那么返回的结果为 Null。

$ABS(4)=4, ABS(-4)=4$

### **ARCTAN(X: FLOAT): FLOAT**

计算  $x$  的反正切。在下面的例子中，同样，用 Sin、Cos 和 Tan 计算其它三角函数。例如：

$Tan(x)=Sin(x)/Cos(x)$   
 $ArcSin(x)=ArcTan(x/sqrt(1-sqr(x)))$   
 $ArcCos(x)=ArcTan(sqrt(1-sqr(x))/x)$

### **COS(X: FLOAT): FLOAT**

计算角  $x$ （弧度）的余弦。

### **EXP(X: FLOAT): FLOAT**

计算  $e$  的  $x$  次幂， $e$  是自然对数的底。

### **COUNTMISSING(依次列出变量名或列出变量范围): INTEGER**

计算指定的变量范围内有缺失值的变量的数目。如果一个数据库中含有 5 个变量：V1、V2、V3、V4 和 V5，其中 V2 和 V4 变量含有缺失值，那么，下面几种表达方式都可以得到整数 2 的结果（记住用双引号把范围值括起来）：

$CountMissing(V1,V2,V4)$   
 $CountMissing("V1-V5")$   
 $CountMissing(V1,V2,"V3-V5")$

### **FLOAT (X): FLOAT**

将  $x$  转换为小数。如果 FIELD1 等于“Q34.3”，则下列函数的结果为 34.3。

$Float(copy(field1,2,4))$

### **FRAC(X: FLOAT): FLOAT**

$x$  是一个浮点型变量，该函数将得到  $x$  的小数部分，即：

$Frac(x)=x-Int(x)$

### **INT(X: FLOAT): FLOAT**

$x$  是一个浮点型变量，该函数将得到  $x$  的整数部分。但是，即使它只含  $x$  的整数部分，该结果仍然属于浮点型数值。

### **INTEGER(X): INTEGER**

该函数将一个字符串（其中含有数字）转换成一个整数。如果 FIELD1 等于“b410”，那么下列函数的结果为 41。

$Integer(copy(FIELD1,2,2))$



**LN(X: FLOAT): FLOAT**

计算浮点型变量  $x$  的自然对数。

**PI: FLOAT**

表示圆周率  $\pi$ ，即圆周长度与圆的直径长度之比，近似等于 3.1415926535897932385。

**POWER(BASE, EXPONENT: FLOAT): FLOAT**

计算某值 (base) 的任意次幂 (exponent)。该值 (base) 必须大于 0。

**RANGE(A,B,C: FLOAT): BOOLEAN**

如果  $a \geq b$  且  $a \leq c$ ，则返回“Y” (是)。b 和 c 可以是数字或数值型变量的变量名。如果结果变量被设置为整数，RANGE() 会返回 0 或 1，例如：

```
LET Teenager=Range(age, 13, 19)
```

**ROUND(X: FLOAT): INTEGER**

将浮点型数值四舍五入转换为整数型数值。例如，下列函数的结果为 3。

```
v2=round(2.5)
```

**SIN(X: FLOAT): FLOAT**

计算角  $x$  (弧度) 的正弦。

**SQR(X: FLOAT): FLOAT**

计算  $x$  的平方。

**SQRT(X: FLOAT): FLOAT**

计算  $x$  的平方根。

**SUM(依次列出变量名或列出变量范围): FLOAT**

将指定的这些变量的数值加起来。时间变量、IDNUM 变量和其它非字符串型变量都可以计算。例如：Sum(V1,V2,"V10-V20")。记住用双引号把范围值括起来。

**TRUNC(X: FLOAT): INTEGER**

该函数截去浮点数值的小数部分，保留整数部分。例如，下列函数的结果为 2。参见 [Round\(\)](#)。

```
v2=trunc(2.5)
```

#### 4.5.2.2 字符串函数 (String Functions)

##### UPPER(S: STRING): STRING

将字符串 S 中的所有字符都转换为大写字母。该转换会影响 ANSI 字符集中的所有字符。

##### LOWER(S: SRING): STRING

将字符串 S 中的所有字符都转换为小写字母。该转换会影响 ANSI 字符集中的所有字符。

##### COPY(S: STRING, INDEX, COUNT: INTEGER): STRING

返回字符串中的某个子串。S 是一个字符型表达式。Index 和 Count 是整数。Copy(S: string, Index, Count: Integer)可以返回一个字符串，其中含有 Count 个字符，从字符串 S 的 Index 位置开始。如果 Index 比 S 的长度大，则该函数只能返回一个空的字符串。如果 Count 指定的字符数超过了实际有的，则该函数只能返回从 Index 开始到结尾的字符。例如，下列函数的结果为“short”。

```
SHORT=copy("My short sentence", 4, 5)
```

##### POS(SUBSTR: STRING; S: STRING): INTEGER

该函数将在一个字符串 (S) 中查找某个子串 (Substr)。S 和 Substr 都是字符型。Pos(Substr: string; S: string)是在 S 中找 Substr，然后返回一个整数值，这个值就是 Substr 的第一个字符在 S 中的位置。Pos(Substr: string; S: string)。如果没有找到 Substr，函数返回数值 0。例如：下列函数的结果为 4。

```
position=pos("short", "My short sentence")
```

##### LENGTH(S: STRING): INTEGER

得出字符串 S 占用的字符数。例如，下列函数的结果为 11。

```
length("This string")
```

##### STRING(X): STRING

将 x 转换为字符串。如果 FIELD1 是一个整数变量，等于 41，则下列函数的结果为“sb41”。参见 [INTEGER](#) 函数，可将字符串转换为数字。

```
"sb"+String(FIELD1)
```

##### SOUNDEX(S: STRING): STRING

函数的结果是字符串 S 的声索引编码。参见[声索引变量](#)一节。

#### 4.5.2.3 日期和时间函数 (Date and Time Functions)

EpiData 将日期作为浮点型数值处理，将其换算为自 1899 年 12 月 31 日以来的天数。这么处理日期使与日期有关的计算非常方便。例如，计算两个日期之间相差的天数，用简单的减法即可实现。

**DATE(D: INTEGER, M:INTEGER,Y:INTEGER): DATE**

参数有 3 个：日、月和年，返回的日期就是由这三个参数组成。该函数得到的结果是日期格式还是整数，取决于指定的变量的类型。

**DAY(D: DATE): INTEGER**

得到日期 D 的日（即，1~31 中的一个数）。

**DAYOFWEEK(D: DATE): INTEGER**

得到一个数字，代表指定日期是星期几。例如，下列函数的结果为 4，表示该日为周四。注意，这里对应的编码是：周一=1，……，周日=7。

```
DayOfWeek("22/02/2001")
```

**MONTH(D: DATE): INTEGER**

得到日期 D 的月份（即，1~12 中的一个数）。

**NOW: DATE**

得到当前日期（整数部分）和当前时间（小数部分）。如果数据录入表格中有两个变量，D1 定义为<dd/mm/yyyy>，T1 定义为##.##，CHECK 语句定义如下：

```
LET D1=Now
```

```
LET T1=Num2Time(Now)。
```

如果当前为 2003 年 9 月 21 日 12:39 分，则上述两个变量显示如下：

**NUM2TIME(D: DATE): FLOAT**

将一个 0~1 的数转换为浮点数值##.##，其中，整数部分表示小时，从 0~24；小数部分表示分钟，从.00~.59。

**TIME2NUM(F: FLOAT): DATE**

如果 F 是一个浮点数值，表示时间，从 0.00~23.59，那么 Time2Num 函数会将这个时间转换为一个 0~1 的数。例如，Time2Num(12.00)=0.50，Time2Num(0.00)=0，Time2Num(24.00)=1.00，Time2Num(9.30)=0.40。

## TODAY: DATE

给出当天的日期。该函数得到的是日期格式还是整数，取决于指定的变量的类型。

## WEEKNUM(D: DATE): INTEGER

得到指定日期距离当年 1 月 1 日的周数。例如：下述函数的结果为 8。

```
WeekNum("22/02/2001")
```

## YEAR(D: DATE): INTEGER

得到日期 D 的年数（4 位数）。

## 关于日期

EpiData 内部将日期换算为自 1899 年 12 月 31 日以来的天数。例如，1899 年 12 月 31 日被定义为 1，则 2000 年 10 月 15 日的日期天数为 36814。这样处理使涉及日期的计算非常方便。

当我们想将某个变量作为日期变量时，如果指定的变量类型为日期格式（如，<dd/mm/yyyy>），则日期会按这种日期格式显示；如果指定的变量类型为整数型（如，#####），则日期会按计算出的日期天数显示。

日期常数可以有两种定义方式：“14/09/2000”或 Date(14,9,2000)，这两个表示的都是 2000 年 9 月 14 日。如果使用的是“dd/mm/yyyy”，当字符长度为 10 个字符，且其中含有有效的欧式日期格式，则该字符串被看作是日期。例如，在一个数据库中含有两个变量：D1 被定义为<dd/mm/yyyy>，INT1 是一个 5 位整数变量（#####）。

LET INT1=D1	如果使用者在 D1 处录入了日期 15/10/2000，则 INT1 会自动计算得 36814。
LET D1=INT1	如果使用者在 INT1 处录入了 36814，则该表达式会使程序将 D1 变量填充为 15/10/2000。
LET D1=D1+7	在 D1 变量录入日期的基础上再加一周，结果的格式为 (dd/mm/yyyy)。
LET INT1=D1+7	在 D1 变量录入日期的基础上再加一周，结果的格式为日期天数（例如，36821）。
LET INT1=Today-Date(1,10,2000)	计算从 2000 年 10 月 1 日起至今的天数。
LET INT1=ROUND((TODAY-D1)/365.25)	计算个体年龄，其出生日期为 D1。
LET D1=Date(1,INT1,2000)	如果在变量 INT1 处输入 4，则 D1 为 01/04/2000。
LET D1="01/04/2000"+5	令日期 D1 为 06/04/2000。

## 如何计算年龄？

假定出生日期变量为 DOB(date of birth)，计算个体在指定日期 2001.06.01 时的年龄。计算公式如下显示：

```
LET AGE=ROUND(("01/06/2001"-DOB)/365.25)
```

分解这个公式可见：

1. 计算出出生日期与指定日期之差：“01/06/2001”-DOB
2. 将结果转换为年数：“01/06/2001”-DOB)/365.25
3. 因为 AGE 变量是一个整数型变量“###”，而 EpiData 不能将实数存入整数变量中，所以需要 round 进行四舍五入，得到整数。

## 关于时间

EpiData 不支持时间变量，不过有两个函数（Time2Num 和 Num2Time）可以令浮点变量在计算过程中充当时间变量。Time2Num 是将浮点数值##.##（必须 $\geq 0.00$ ，且 $< 24.00$ ）转换为 0~1 之间的一个数值，表示一天的各个时点。例如，中午的 12.00 可转换为 0.5，下午的 18.00 转换为 0.75。注意，这里使用的是 24 小时制。

反映日期和时间的数值可以加在一起，其中整数部分代表日期，小数部分代表时间。Num2Time 函数是将一个 0~1 间的数值转换为 0.00~23.59 之间的浮点数值。前面提到的那种将日期和时间放在一起表示的数一般大于 1，不过，这并不会影响 Num2Time 函数，只有其中的小数部分会被转换。

你可以参考安装文件夹中 samples 内的实例 DateTime。

### 4.5.2.4 其它函数

#### ISBLANK(FIELD NAME): BOOLEAN

如果某个变量没有录入，则认为是“真”；如果变量中含有数据，则认为是“假”。另外，也可以不用“ISBLANK(变量名)”函数，直接用一个圆点表示缺失值。下面两种语句得到的结果是相同的。

```
IF ISBLANK(V1) THEN ...  
IF V1=. THEN ...
```

#### RECORDCOUNT: INTEGER

给出目前已经保存了的记录数。如果当前的记录是一个新记录，尚未录入、保存。则该条记录不会被计算在内。

#### RECORDNUMBER: INTEGER

给出当前记录的记录号。如果当前记录为一条新记录，则得到的值为“-1”。

## 5. 录入数据 (Enter Data)

调查表文件 (.QES)、数据库 (.REC) 和核查文件 (.CHK) 都已经创建完毕，现在你可以开始录入数据了。有几种方式可以打开已经建立好的数据库：(1) 菜单 **File** → **Open** → 选择要打开的数据库 (\*.rec)；(2) 菜单 **Data in/out** → **Enter Data** → 选择要打开的数据库 (\*.rec)；(3) 在 workflow 栏上点击 **4.Enter Data** → 选择要打开的数据库 (\*.rec)。如果程序发现在相同的文件夹内有同名的 CHECK 文件，其中设置的录入规则会自动应用于该数据库的录入过程。

一条记录中所有数据录入完毕，程序会提示录入员，是否保存该条记录。如果要终止录入、关闭数据库，请选择菜单 **File** → **Close Form** 或按 **Ctrl+F4** 键，或点击窗口右上角的

## 5.1 在变量间转换

在数据录入过程中，如果你使用鼠标实现在变量间的跳转，则 CHECK 文件中设置的录入规则通常无效。激活下一个变量，你可以使用 **Enter**、**Tab**、**↓** 键、或用鼠标直接点击目标变量。如果变量允许录入的字符数全部录满，则光标会自动移到下一个变量，除非你在 CHECK 文件中设置了 **CONFIRM** 命令。


如果想回到上一个变量，可以按 **Shift+Tab** 键，或 **↑** 键。按 **Ctrl+Home** 键可以直接回到数据录入表格的第一个变量。选择 **Ctrl+End** 则可以直接跳转到最后一个变量。


## 5.2 在记录间转换


数据录入表格的窗口下部有一个导航条。这些按钮的功能与 Goto 菜单是一致的。



上图显示当前数据库有 3 条记录，现在位于第 2 条记录。红色 DEL 显示为激活，表示当前记录已经被标记删除。


 移到第一条记录

 移到前一条记录 (或 **Ctrl+PgUp**, 或 **F7**)



 移到下一条记录 (或 **Ctrl+PgDn**, 或 **F8**)

 移到最后一条记录

 开始录入新记录 (或 **Ctrl+N**)

 删除记录或恢复一条删除的记录 (或 **Shift+Delete**)，这里需要说明的是，记录只是被标记为删除，实际上，这条记录仍然存在于数据库中，是可恢复的。在菜单 **Tools** 中选择 **Pack File**，可以永远的删除所有标记为删除的记录。


## 5.3 在关联数据库间转换

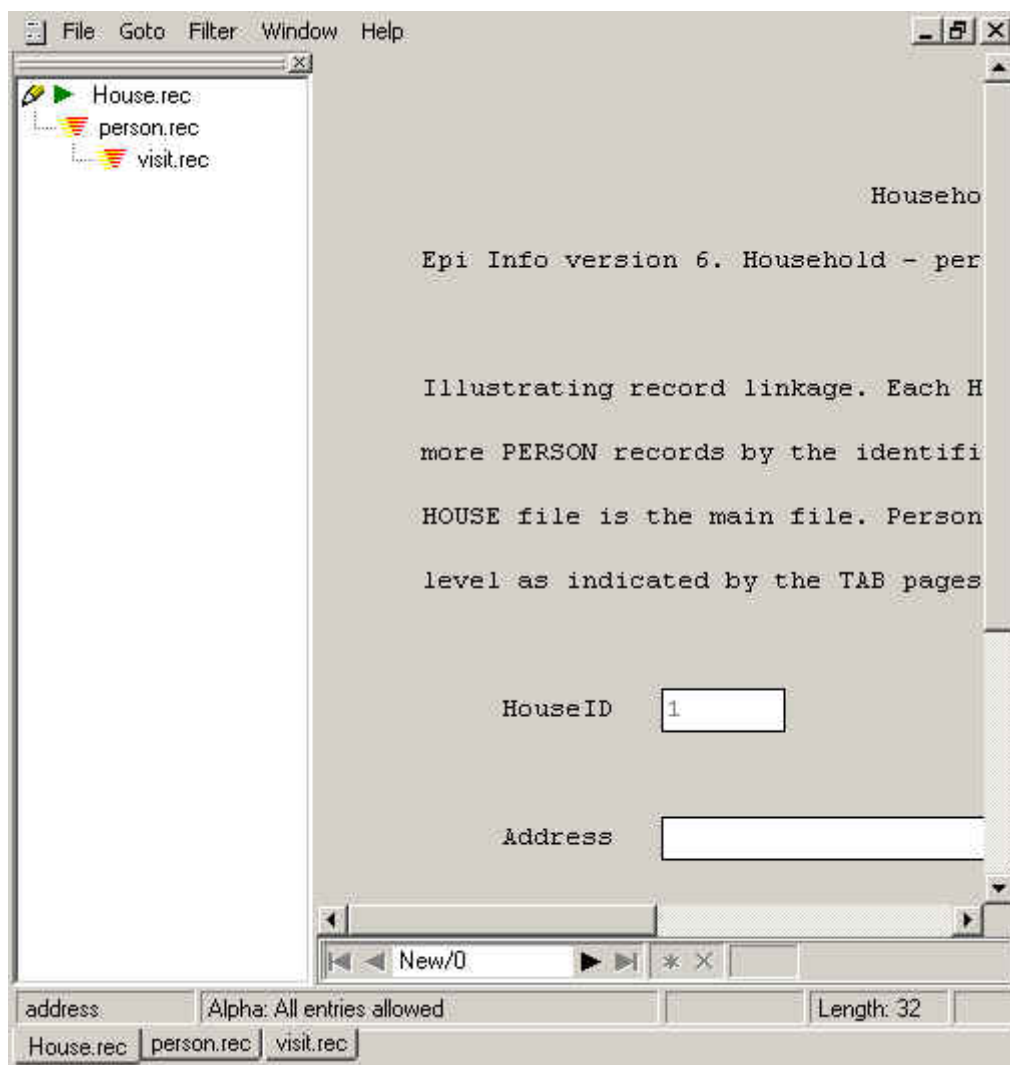
当打开一组关联的数据库 (即，有 **RELATE** 命令的数据库) 时，数据表格的左侧会显示“关联树” (relatetree)。根据关联树，我们可以清楚地了解数据库间的关系，并实现在数据库间的转换。 标记的文件为当前激活的数据库，即为了录入点击打开的数据库； 标记的数据库为当前屏幕上显示的数据库。

点击关联数据库的名字，可以浏览数据库的内容，但是仍处于只读模式，不能修改、录入数据。你只能从激活的数据库开始，经含 **RELATE** 命令的变量激

活、进入关联数据库，进行录入或修改。

在数据库名字上点击右键，会出现一个小标签，上面给出了有关关联关系的进一步信息。

点击关联树窗口右上角的  可以关闭该窗口，按 **F11** 或从菜单 **Window** → **Show relate tree** / **Hide relate tree** 也可以实现打开/关闭该窗口。拖拽关联树窗口顶端的边框条，可以移动该窗口到屏幕的任何位置。

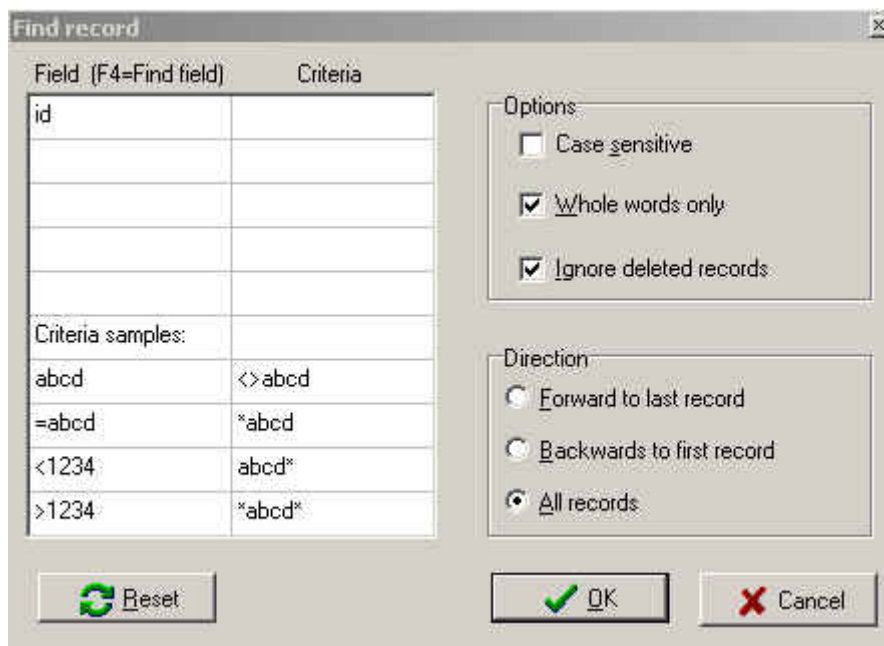


#### 5.4 查找记录

如果已知要寻找的记录号，可以从菜单 **Goto** → **Goto Record** 或按 **Ctrl+G**，在弹出的对话框中输入要寻找的记录号，即可转换到该条记录上。



如果不知道记录号，也可以使用 **Goto** 菜单中的 **Find Record** 或按 **Ctrl+F**，程序会自动弹出一个对话框。程序默认在当前变量（即选择 **Find Record** 时正处于激活状态的那个变量）上进行搜索，不过你也选择对其它变量进行搜索。其中也包括那些不能激活的变量，如 **IDNUM** 变量。



搜索可以同时最多在 10 个不同的变量上进行。可以设置的参数包括：等于（“=”可写、可不写）、不等于（<>）、大于（>）、小于（<）、以此开始（abcd\*）、以此结束（\*abcd）、或包含（\*abcd\*）。另外，在设置“等于”条件时，建议条件的写法与设置好的变量类型保持一致。例如，一个数值型变量设置为 4 位整数、2 位小数。则如果欲搜寻该变量等于 11 的记录，则应该写条件为“=11.00”，而不能简单的写为“=11”。如果勾选 **Options** 中第一个选项“case sensitive”，表示搜索字母时，大写和小写字母是不一样的。

按 **F3** 或选择 **Find Again** 可以用相同的搜索条件继续搜索。在搜索过程中，按 **Esc** 或按 **Cancel** 可以终止搜索。

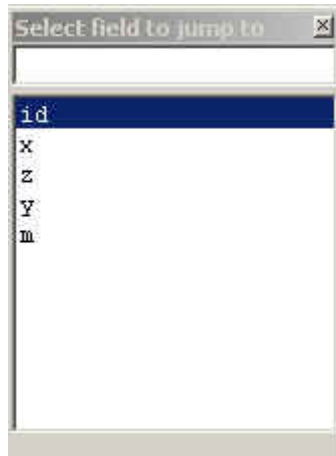
在设置了 **KEY** 的变量上搜索，速度更快。另外，还可以设置一些选项和搜索的方向。

## 5.5 查找变量和关联变量

如果想找某个变量，可以按 **F4** 或菜单 **Goto** → **Find field**，程序会弹出一个



当前数据库的所有变量列表。输入要寻找的变量名或在列表中选择，点击 **Enter**，即可跳到该变量。



按 **Shift+F4** 或选择菜单 **Goto** → **Find relate field**，或者按两次 **F4** 键，程序会弹出一个变量列表，其中只含有设置了 **RELATE** 命令的变量。

## 5.6 滤过 (Filter)

在数据录入过程中，我们可以限制哪些记录被显示出来。

### 5.6.1 设置滤过规则

把光标放在想要设置滤过功能的变量上。该变量必须是 **KEY** 或 **KEY UNIQUE** 变量。选择菜单上的 **Filter** → **Define Filter** → 键入滤过值。这时，只有符合条件的记录才被显示。如果选择进入下一条记录，依然是符合条件的记录才会被显示。

### 5.6.2 解除滤过

选择 **Filter** → **Deactivate Filter**，可以解除前面设置的滤过功能。

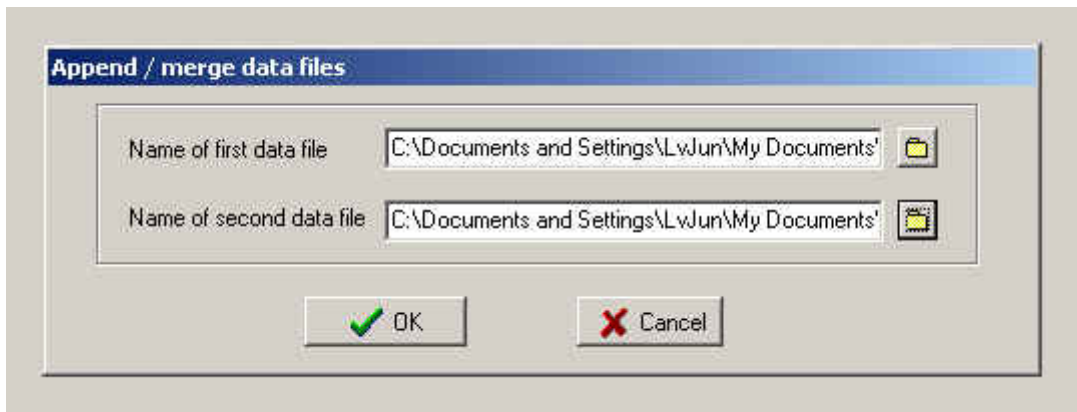
## 6. 数据库的管理和维护

### 6.1 数据库的追加与合并 (Append/Merge Data Files)

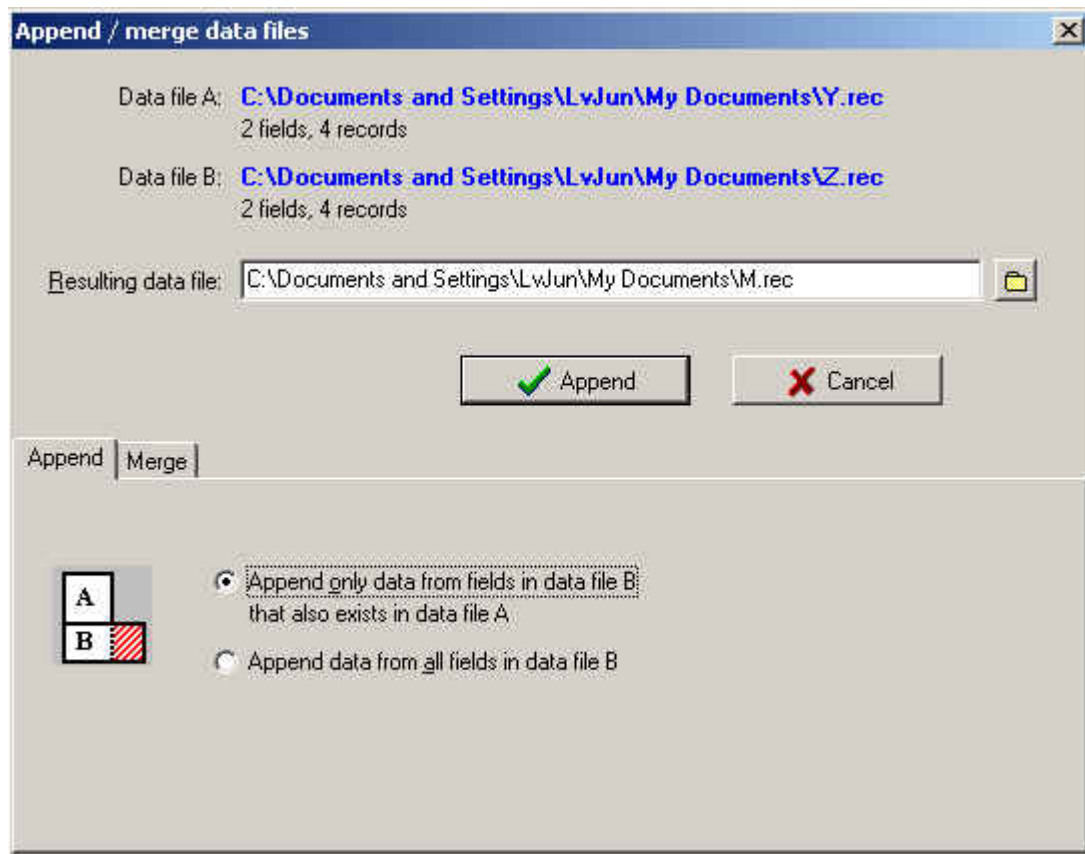
该项功能可以将两个数据库合并建成另一个新的数据库。追加 (**append**) 是将两个数据结构完全一样或基本上一样的数据库连起来。两个数据库是端对端 (**end-to-end**)，又称串联。而合并 (**merge**) 是将两个结构不同、但是有 1-3 个相同变量 (如，**ID** 变量或 **key** 变量) 的数据库合并。例如，一个数据库中录入的是问卷调查结果，而另一个数据库中录入的是同一批患者的临床检查结果。两个数据库都含有一个可以确定患者的 **ID** 号。这样的两个数据库合并是边对边 (**side-to-side**)，又称并联。

#### 6.1.1 数据库的追加 (Append)

从 **Data In/Out** 菜单中选择 **Append/Merge**，输入准备合并的两个数据库的文件名，点击 **OK**。

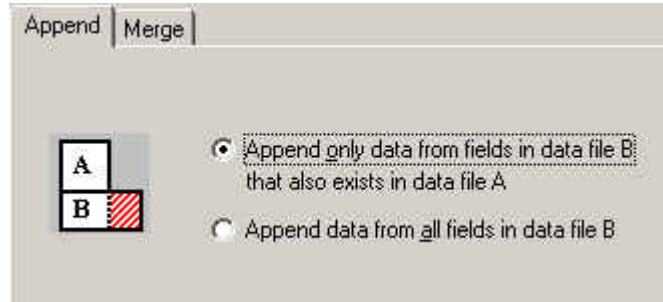


弹出的对话框中会显示两个数据库的情况。键入准备建立的新的数据库（包含两个数据库的内容）的文件名。这个操作不会修改两个准备合并的原始数据库。

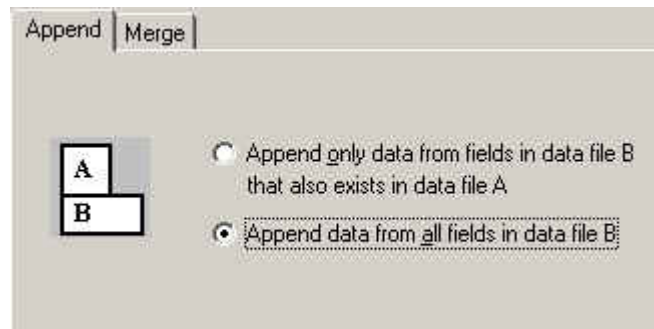


追加的方式有两种：

- (1) 追加后新建的数据库的结构与数据库 A 相同，即有相同的变量。至于数据库 B 中的数据，只有与数据库 A 相同的变量才会被追加到新的数据库中，数据库 A 中没有的变量会被忽略。

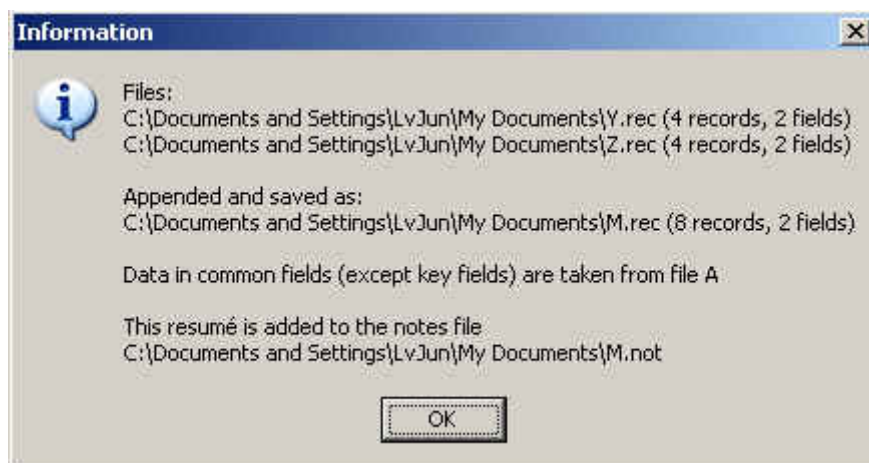


(2) 新的数据库中包含所有数据库 A 中的变量和数据库 B 中的变量。



注意，这里数据库 A 被看作是“主”数据文件。如果数据库 A 和数据库 B 中含有相同名称的变量，则追加、新创建的数据库中对应的变量类型将以数据库 A 中的为准。如果数据库 A 或数据库 B 有 CHECK 文件，追加/合并功能会将其引入新的合并后的数据库。使用者应该仔细检查和确认合并后的数据库的 CHECK 设置是否合适，特别留意 JUMPS、GOTO 和 IF...THEN...ENDIF 命令。

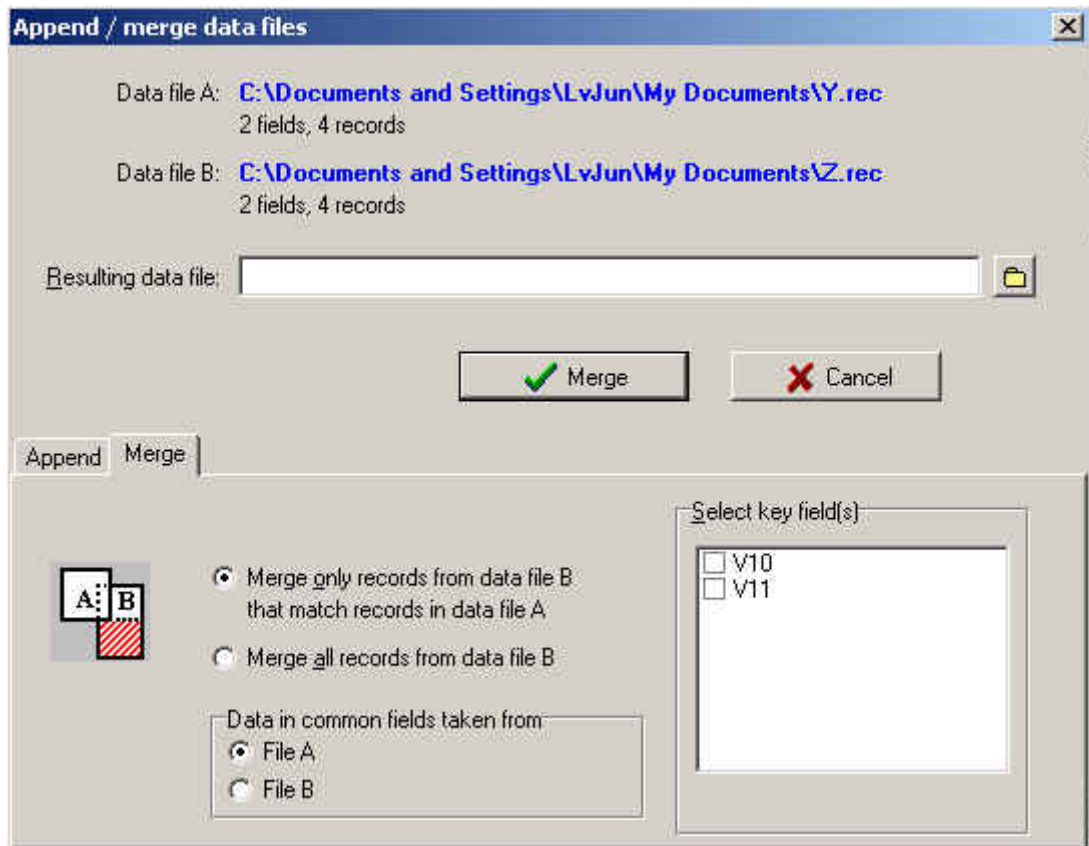
追加程序执行完毕后，程序会显示新建的这个合并数据库的简要情况。这些内容同时会被添加到新建的合并数据库的数据录入备忘录文件(data entry notes file)中。



### 6.1.2 数据库的合并 (Merge)

从 **Data In/Out** 菜单中选择 **Append/Merge**，输入准备合并的两个数据库的

文件名，点击 **OK**。弹出的对话框中会显示两个数据库的情况。键入准备建立的新的数据库（包含两个数据库的内容）的文件名。这个操作不会修改两个准备合并的数据库。

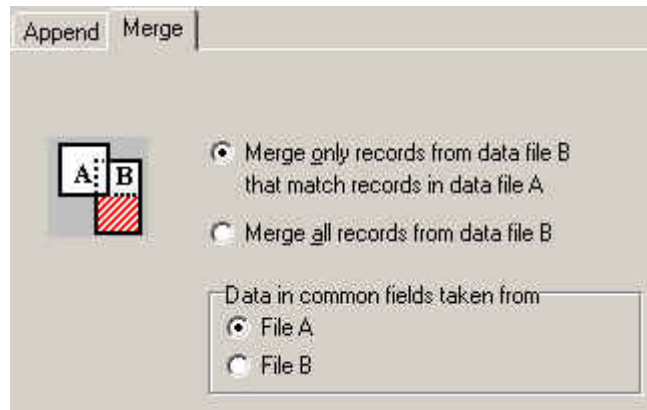


合并功能要求两个数据库都必须有一个或多个标识变量，以便匹配数据库 A 和数据库 B 中对应的记录。你最多可以选择 3 个标识变量。标识变量不一定要设置为 **KEY** 或 **KEY UNIQUE**，但是必须是在两个数据文件中都存在。

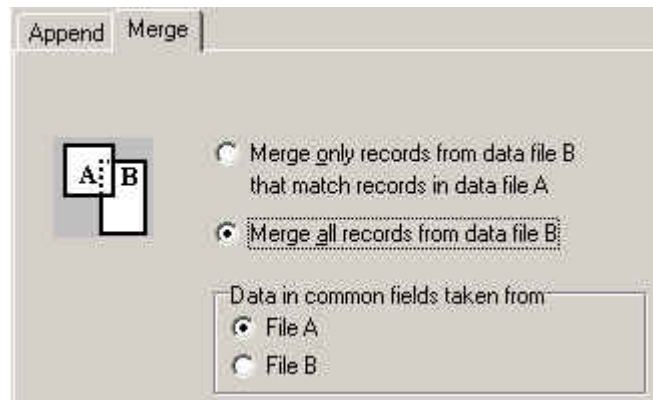
选择 **Merge** 页，右侧显示了数据库 A 和数据库 B 共有的变量列表。如果没有共同的变量，合并将无法继续。从共有的变量列表中选择 1-3 个变量。这些标识变量在两个数据库中都必须唯一的。

合并的方式有两种：

(1) 只合并那些标识变量在数据库 A 和数据库 B 完全匹配的记录。



(2) 合并两个数据库中的所有记录。这个操作可能会使很多变量出现缺失值，因为来自数据库 B 的一些记录，在数据库 A 中没有匹配的记录。

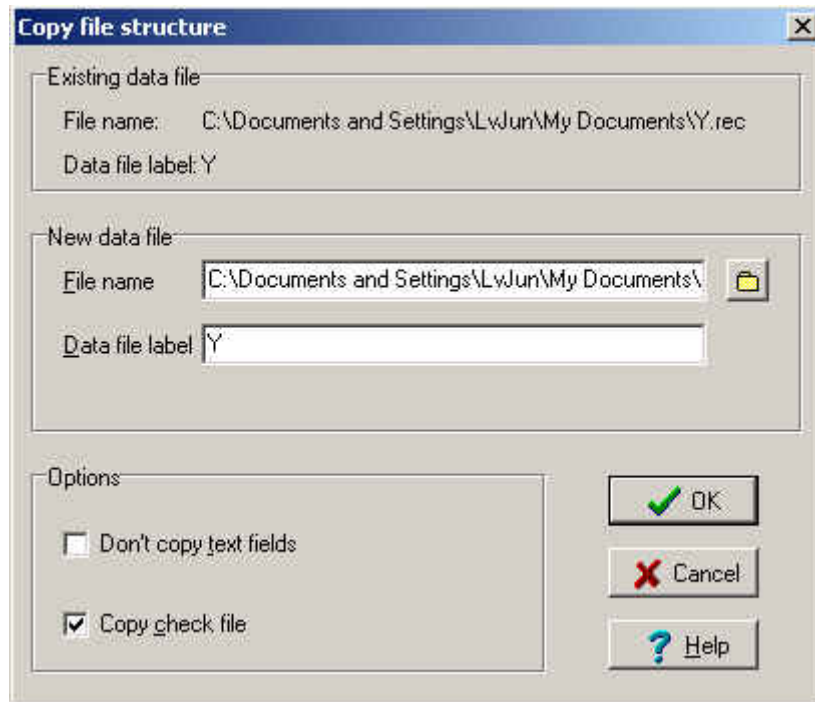


如果数据库 A 或数据库 B 有 CHECK 文件，追加/合并功能会将其引入新的合并后的数据库。使用者应该仔细检查和确认合并后的数据库的 CHECK 设置是否合适，特别留意 JUMPS、GOTO 和 IF...THEN...ENDIF 命令。

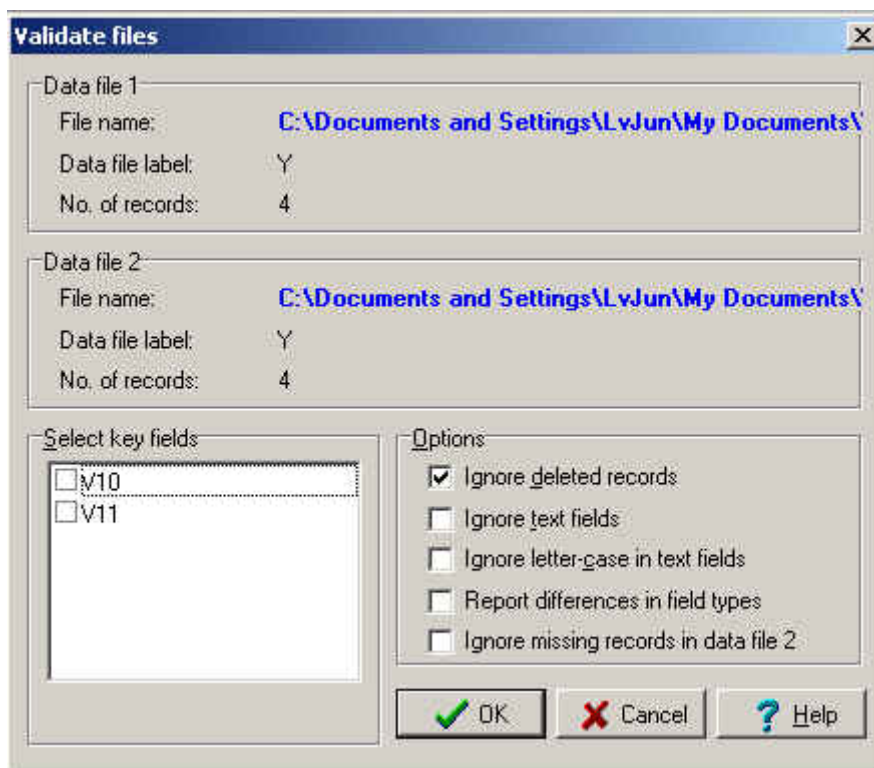
追加程序执行完毕后，程序会显示新建的这个合并数据库的简要情况。这些内容同时会被添加到新建的合并数据库的数据录入备忘录文件(data entry notes file)中。

## 6.2 双录入核查 (Double Entry and Validation)

为了保证数据录入的质量，我们一般会让两个人分别录入同一套数据，然后保存为两个不同的数据库。在准备双录入时，你可以利用菜单中的 **Tools** → **Copy Structure** 功能，将已经建立好的一个数据库的结构（可以包括已经建好的 CHECK 文件）拷贝、另存为一个新的数据库，而其中已经录入的数据不会被拷贝到新的数据库中。另外，如果情况允许，你可以选择不拷贝字符型变量（如被调查者的家庭地址、工作单位等）。确实，很少有人愿意将字符型变量录入两遍，而且即使录入完两遍，双录入核查过程中还会引出很多麻烦。



双录入完毕后，你可以进入菜单 **Document** → **Validate Duplicate Files**，或在工作流程栏上点击 **5.Document** → **Validate Duplicate Files**，然后选择要进行比较的两个数据库的文件名，并设置核查过程中的一些参数。



### 6.2.1 选择关键变量

要想比较两个数据库，你必须指定一个或多个关键变量。这个（些）关键变

量被用来匹配两个数据库中对应的记录，或者说是需要比较的记录。可供选择的关键变量列表中只包括那些在两个数据库中都存在的变量。在 CHECK 文件中设置了 KEY 命令的变量，列表中这样的变量旁边会显示钥匙样的记号。不过，双录入核查中选择的关键变量不一定是 CHECK 文件中设置了 KEY 的变量。

如果你不指定关键变量，那么程序就会按着记录顺序来比较两个数据库，即数据库 1 中的第一条记录与数据库 2 中的第一条记录进行比较，依此类推。这种情况下，两个数据库中记录录入的顺序必须一致。

### 6.2.2 选项设置

#### (1) 忽略删除的记录 (Ignore deleted records)

不核对标记为删除的记录。

#### (2) 忽略字符型变量 (Ignore text fields)

不核对字符型变量和大写字母的字符型变量。

#### (3) 忽略字符型变量中字母的大小写 (Ignore letter case in text fields)

程序会将 “Smith” 和 “sMiTh” 识别为输入一致。

#### (4) 报告变量类型的差异 (Report differences in field types)

如果选择了这一项，双录入核查报告中会报告如下信息：两个数据库中是否有相同的变量名、但是却是不同的变量类型。

#### (5) 忽略数据库 2 中缺失的记录 (Ignore missing records in data file 2)

勾选了这一项，程序就不会弹出类似 “records found in data file 1 are not found in data file 2” (数据库 1 中的某些记录在数据库 2 中没有发现) 样的信息。如果只重复录入原始数据库中部分记录，你可以勾选这一项。选择原始 (完整) 的数据库作为数据库 1，部分录入的作为数据库 2。

### 6.3 逻辑一致性核查 (Logical Consistency Check)

前面建立的 CHECK 核查文件主要是在数据录入过程中交互式地影响数据的录入，在一定程度上保证了录入数据的合理性和正确性。而这里提到的逻辑一致性核查，则是在数据录入完毕后，通过一次性设置一批核查命令，检查数据库中已有数据的逻辑一致性如何，而你又无需手动浏览整个数据库。最后，程序会列出所有不符合一致性核查命令的记录。

为了实现这个功能，你必须在该数据库的 CHECK 文件中，或在另外一个文本文件中，设置 CONSISTENCYBLOCK 命令块。CONSISTENCYBLOCK 是个块命令，所以必须以 END 结束。块中允许定义的命令数量没有限制。

运行菜单中的 Document → Consistency Checks，键入欲核查的数据库名，以及含有 CONSISTENCYBLOCK 的文件名 (可以是 CHECK 文件或其它文件)。不同的数据库可以共用同一个 CONSISTENCYBLOCK 文件，点击 OK，待核查过程完毕可以查看报告。CHECK 的语句可以写成下面这种形式：

```
CHECK "Text explaining the purpose of the check" logical /  
Boolean expression
```

例如：

```
CONSISTENCYBLOCK
```

```

REPORT ID1
CHECK "V1 is missing or unusually big" (V1<>.) AND (V1<112)
CHECK "ID-numbers are not identical" ID1=ID2
CHECK "Mother too young" (AGE<15) AND (HASCHILD="N")
CHECK "Ranges" CHECKRANGE
CHECK "Legal" CHECKLEGAL
CHECK "MustEnter" CHECKMUSTENTER
END

```

## REPORT

REPORT 命令是告诉程序如何报告那些不满足 CHECK 命令的记录。如果 CONSISTENCYBLOCK 中没有设置 REPORT 命令，则程序默认报告不满足条件的记录的记录号。在上面的例子中，程序会报告不满足条件的记录的 ID1 变量。

在上面的例子中，根据第一个 CHECK 命令，程序会报告 V1 变量缺失或大于 112 的记录。其中的说明性文字“V1 is missing or unusually big”与命令本身没有关系，只是作为提示用。

这里使用的逻辑表达式与 IF 表达式一样，可以使用相同的[运算符和函数](#)。表达式必须列出什么是满足条件的记录，而最后的报告中列出来的则是不满足条件的记录。除此之外，你还可以使用 4 种预设的 CHECK 命令：

## CHECKRANGE

检查所有变量的数值是否都符合 CHECK 文件中定义的 RANGE 的范围。

## CHECKLEGAL

检查所有变量的数值是否都符合 CHECK 文件中定义的 LEGAL 和 COMMENT LEGAL 的限制。有缺失值的变量会被忽略。

## CHECKRANGELEGAL

检查所有变量的数值是否符合 CHECK 文件中定义的 LEGAL、COMMENT LEGAL 和 RANGE。有缺失值的变量会被忽略。

## CHECKMUSTENTER

检查设置了 MUSTENTER 的变量中是否有缺失值。

## 6.4 根据数据库创建 QES 文件 (Make QES File from Data File)

如果创建数据库的 QES 文件已经找不到了，我们可以利用菜单中的 **Tools** → **QES File from REC File** 功能，根据已有的数据库反过来创建 QES 文件。键入数据库的文件名及准备新建的 QES 文件名，点击 **OK**。如果想利用新的 QES 文件创建新的数据库，则使用该项功能可以保证变量名与旧数据库中的完全一样。

## 6.5 重新编码数据库 (Recode Data File)

该项功能允许你改变数据库中所有记录的多个变量的数值。重新编码的命令是写在 RECODEBLOCK...END 命令块中，我们可以在 CHECK 文件中定义这



个命令块，也可以在另一个文本文件中定义。记住，不要将 RECODEBLOCK...END 命令块写在任何变量块中。RECODEBLOCK 中可以使用的命令包括：LET、IF...ELSE...THEN...ENDIF、CLEAR 和 EXIT。

如果要运行重新编码的命令，可以选择菜单 **Tools** → **Recode Data** → 键入准备重新编码的数据库名，以及含有 RECODEBLOCK 的文件名。所有记录都会被重新编码。程序会弹出一个信息框，显示会被修改的记录数。这个时候，你可以选择取消操作，数据库不会被修改。如果不取消操作，则原始数据库会被备份、另存为 *FILENAME.OLD.REC*，而 *FILENAME.REC* 中含有的是重新编码了的记录。



下面这个例子中，我们使用了 RecordNumber 这个函数，它可以表示当前的记录号。重新编码后，记录 1 的 ID 变量会变为 3001，记录 2 变为 3002。

```
RECODEBLOCK
  ID=RecordNumber+3000
  IF V2=. THEN
    CLEAR V3
    EXIT
  ENDIF
  V3=V3+1
END
```

重新编码的操作情况会载入数据录入备忘录中。你可以通过 **Document** → **Data Entry Notes** 浏览备忘录。

Data entry notes for C:\Documents and Settings\LvJun\My Documents\Untitled 1.rec

```
-----  
22 九月 2003 08:27  
Data file recoded. 6 records were changed  
RECODEBLOCK  
  ID=RecordNumber+3000  
  IF V2=. THEN  
    CLEAR v3  
    EXIT  
  ENDIF  
  V3=V3+1  
END
```

## 6.6 将年的表示方式从 2 位数转换为 4 位数

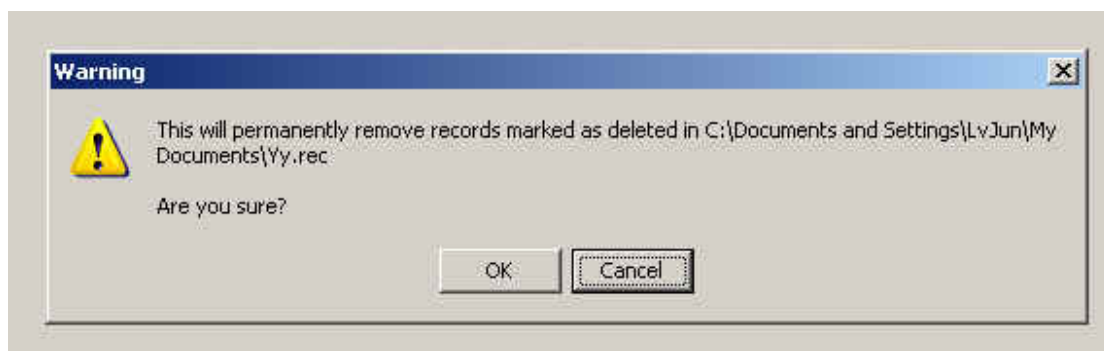
这个功能非常简单，EpiData 会将所有日期的年读作 4 位数：其中 50~99 读作 1950~1999 年，00~49 读作 2000~2049 年。

```
RECODEBLOCK  
  * to convert a 2 digit year to a four digit year:  
  fouryear=twoyear  
END
```

## 6.7 打包数据库 (Pack Data File)

在数据录入过程中，我们可以按 **Shift+Delete** 把某些记录标记删除。但实际上这些记录并没有真正的从数据库中删除，而只是被作了标记。为了永久地删除这些标记了的记录，我们可以用菜单 **Tools** → **Pack Data File** 功能。

在永久删除所有标记记录前，程序会弹出一个警告框，提示会有多少记录被删除。如果点击 **OK**，记录被永久删除。原始数据库的备份文件被另存为 FILENAME.OLD.REC。如果点击 **Cancel**，则数据库不会被修改。



## 6.8 压缩数据库 (Compress Data File)

选择菜单 **Tools** → **Compress Data File**，可以改变数据库中所有变量的长度，调整后的长度正好为该变量中最大数值的长度值。例如，如果一个整数变量被定义为##### (10 位数)，但该数据库中这个变量上最大的值仅为 9999，则

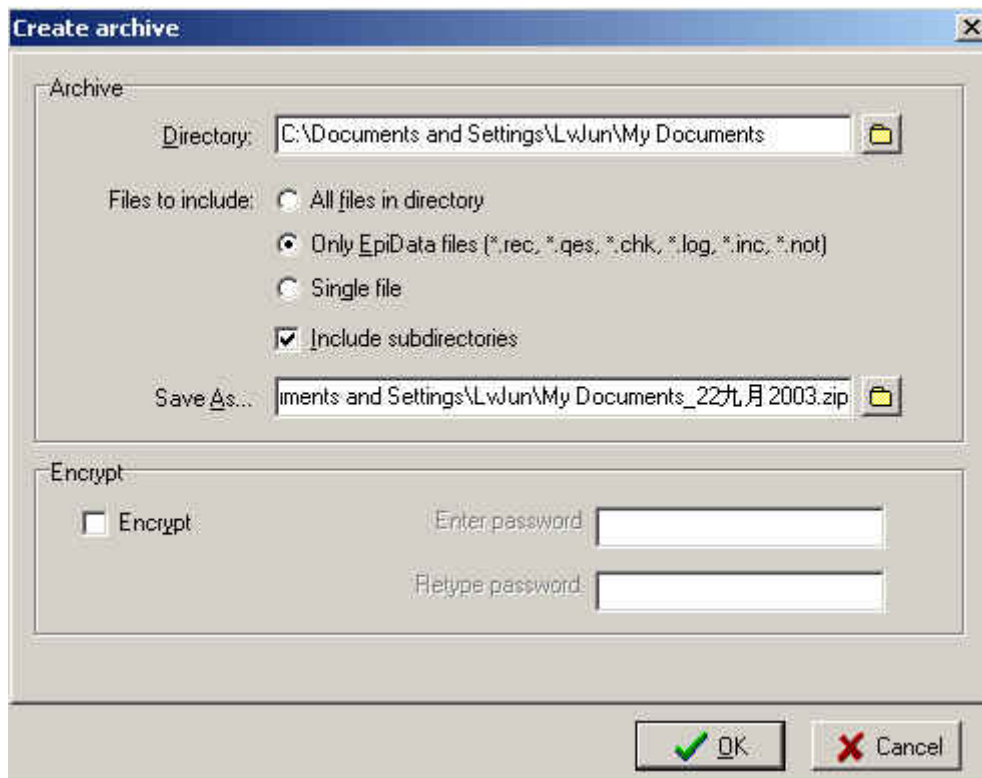
该变量会被重新定义为####（4 位数）。利用该项功能，你可以减小数据库的大小，所有变量浪费的空间被移走了。同样，原始数据库的备份文件会被另存为FILENAME.OLD.REC。

### 6.9 打印数据录入表格（Print Data Entry Form）

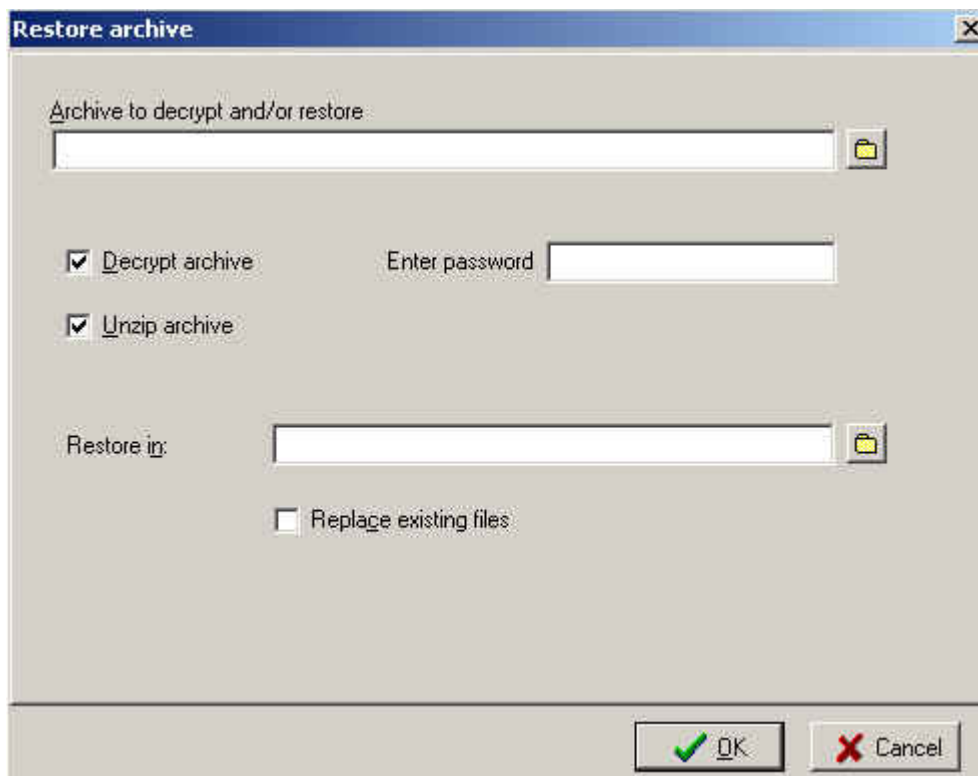
点击 **File** → **Print Data Form**，你可以打印当前屏幕上显示的数据录入表格，打印出来的效果和屏幕上显示的一样，包括当前记录的变量值。另外，在使用数据录入表格的预览功能时，即在编辑器状态下，点击工作流程栏上的 **2.Make Data File** → **Preview Data Form** 或菜单 **Data File** → **Preview Data Form**，也可以使用该打印功能。

### 6.10 将数据库存档（Archive）

选择菜单 **Tools** → **Create Archive**，你可以将指定文件内所有文件、仅 EpiData 文件（\*.rec、\*.qes、\*.chk、\*.log、\*.inc、\*.not）、或单一文件压缩、打包、保存起来，以便相关文件的传递、存档等。存档的文件会以压缩文件（\*.zip）形式保存，文件名由准备存档的文件夹名或文件名和保存时间（年月日）组成。你可以对这个存档文件设置密码。

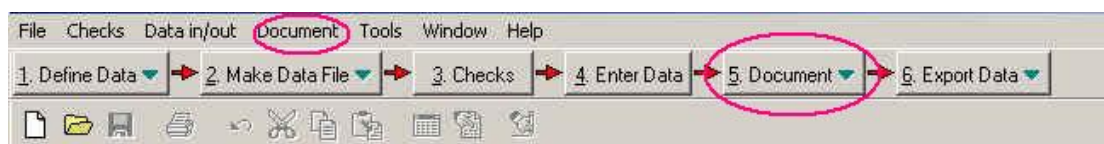


需要的时候，选择菜单 **Tools** → **Restore Archive**，你可以将这些存档文件恢复到指定的文件夹下，或破解密码，将其中的文件释放出来。



## 6.11 数据库相关信息的管理

通过生成某些报表，我们可以了解数据库及其录入变量的相关信息，并将这些信息存档、打印。与数据库有关的信息包括：数据库名称、文件大小、最后修改的日期、变量数、记录数、是否有相应的 **CHECK** 文件等。与录入变量有关的信息包括：变量名、变量标签、变量类型、变量长度、应用的 **CHECK** 命令等。



### 6.11.1 数据库结构 (File Structure)

选择菜单 **Document** → **File Structure** 或 workflow 栏上的 **5. Document** → **File Structure**，选择数据库。程序会输出一个文件，其中含有以下信息：数据库标签、数据库大小、最后修改日期、变量数、记录数、是否有配套的 **CHECK** 文件、变量名、变量标签、变量类型、长度、应用的 **CHECK** 语句等。

DATA FILE: C:\Documents and Settings\LvJun\My Documents\Untitled 1.rec  
 File label: [none]  
 File size: 250 bytes  
 Last revision: 22. 九月 2003 08:27  
 Number of fields: 3  
 Number of records: 6  
 Checks applied: Yes (Last revision 22. 九月 2003 12:44)

Fields in data file:

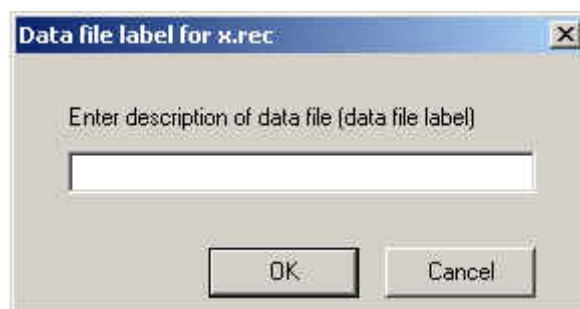
No.	Name	Variable label	Field type	Width	Checks
1	id		Number	4	Must enter
2	v2		Number	4	
3	v3		Number	4	

### 6.11.2 数据录入备忘录 (Data Entry Notes)

在数据录入过程中，如果需要作个短短的笔记或注释，可以选择菜单 **File** → **Data Entry Notes** 或按 **F5** 键，激活备忘录窗口。如果当前数据库没有备忘录文件，可以创建新的，当前的日期和时间会自动插入新建的备忘录 (\*.not) 中。另外，我们也可以在主窗口中选择菜单 **Document** → **Data Entry Notes** 进入备忘录。如果选择了菜单中的 **Data in/out** → **Backup** 备份数据库，则备忘录也会同时被备份。另外，备忘录还会保存“追加、合并”和“输入”等操作的信息，使我们可以掌握数据库的修改状况。在 CHECK 文件中设置 **WRITENOTE** 命令，可以在数据录入过程中向备忘录中添加某些内容。

### 6.11.3 数据库标签 (Data File Label)

在创建数据库之初，我们可以输入一段简短的描述性文字，作为数据库标签（最多 50 个字符）。该标签将作为 REC 数据库的一部分被保存起来。当数据库被输出到 Stata、Sas 或 SPSS 统计分析软件时，数据库标签也会作为数据库说明文件的一部分显示。你可以选择菜单 **Tools** → **Edit File Label** 修改数据库标签。



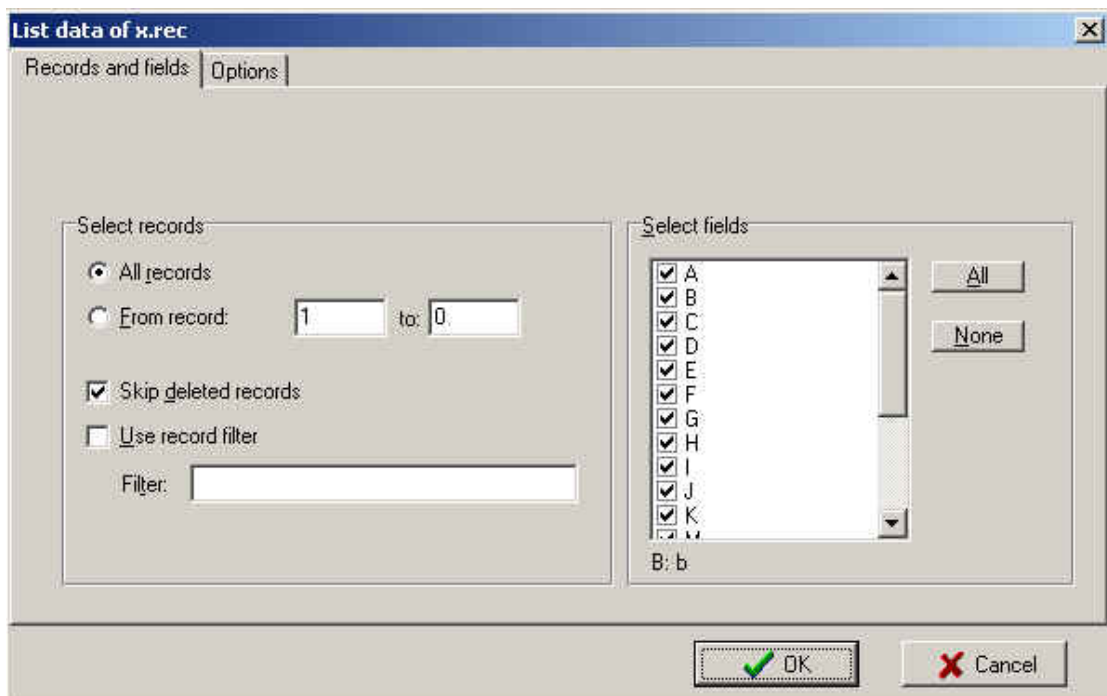
### 6.11.4 浏览数据 (View Data)

选择菜单 **Document** → **View data** 或工作流程栏上的 **5.Document** → **View data** 或按 **Ctrl+Alt+V**，选择数据库。结果数据库以如下的形式显示出来。

Rec.no.	id	v2	v3
1	3001	2	3
2	3002	2	3
3	3003	2	3
4	3004	2	3
5	3005	2	3
6	3006	2	3

### 6.11.5 数据列表 (List Data)

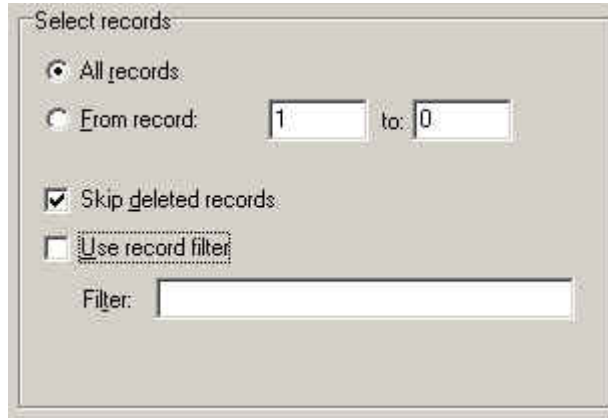
使用这个功能,我们可以将已有记录的所有或部分数据以可打印的格式列出来。选择菜单 **Document** → **List Data** → 选择数据库, 程序弹出一个对话框, 我们可以进行一些选项设置。



#### 选择记录

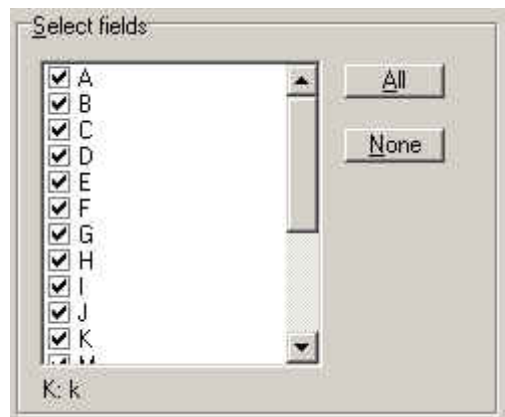
- (1) 列出所有记录或只是一段范围内的记录 (例如, 从第 100 条记录到第 199 条记录);
- (2) 忽略删除的记录;
- (3) 选择符合条件的记录: 设置布尔逻辑表达式, 只有满足上述条件的记录才会被列出来。例如:

```
ID>1000 or (V10<>.) AND (v100=20)
```



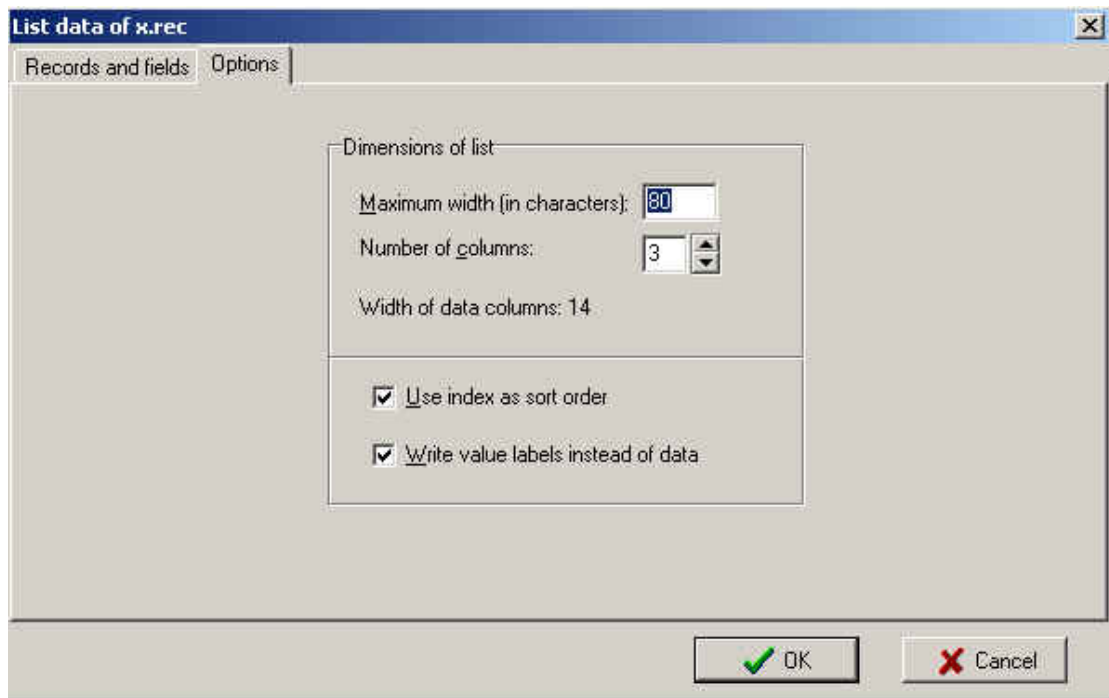
### 选择变量

通过选定或不选定的操作，在列表中选择需要列出的变量。点击 **All** 可以选择全部变量，点击 **None** 则表示所有变量都不选定。



### 其它选项

- (1) 列表尺寸 (Dimensions of list): 通过修改数据列表的行宽和列数，以保证打印出来的格式的美观性。
- (2) 使用索引进行排序 (Use index as sort order): 数据列表将按照 CHECK 文件中设置的 KEY 变量进行排序。只有设置了 KEY 变量，才能使用这个功能。
- (3) 列出数值标签，而不列数据 (Write value labels instead of data): 勾选该项后，如果一个变量定义了数值标签(在 CHECK 文件中用 COMMENT LEGAL 定义的)，则列表中只会列出数值标签，而不是列出数据。



输出的数据列表如下所示。



List of observations in C:\Documents and Settings\LvJun\My Documents\Y.rec  
List created 12. 九月. 2003 19:39

Records in file: 4  
Records in list: All  
Deleted records are skipped.

```

Observation 1 (record # 1)
      v10          110      v11          111

Observation 2 (record # 2)
      v10          210      v11              1

Observation 3 (record # 3)
      v10           99      v11          100

Observation 4 (record # 4)
      v10          210      v11              9
  
```

### 6.11.6 基本的统计表格 (Codebook)



选择菜单 **Document** → **Codebook** → 选择数据库 → 进行必要的设置 → 点击 **OK**。该功能可以向你提供数据库的关键信息和一些基本的描述性统计结果，如记录数、删除的记录数、变量标签、变量类型、选择的 **CHECK** 命令和缺失值、简要的描述性统计信息（是否显示要视变量类型而定）。

程序默认输出全部记录和变量，但是也可以选择输出部分记录、变量。不过，如果你选择了输出部分记录，可能会影响到均值等的计算。在选项页中，你可以选择显示的 **CHECK** 文件的内容。

```

CODEBOOK

Report generated          12. 九月 2003 20:38

Data file:               C:\Documents and Settings\LvJun\My Documents\Y.rec
File label:              Y
File date:               9. 九月 2003 21:46
Checks applied:         Yes (Last revision 9. 九月 2003 21:49)

Number of fields:       2

Records total:          4
Deleted records:        0
Used in codebook:       4 records

V10 -----
      type:  Number
      missing:  0/4
      range:  [99 ; 210]
      unique values:  3

      tabulation:
      Freq.    Pct.    Value    Label
          1     25.0     99
          1     25.0    110
          2     50.0    210

V11 -----
      type:  Number
      missing:  0/4
  
```

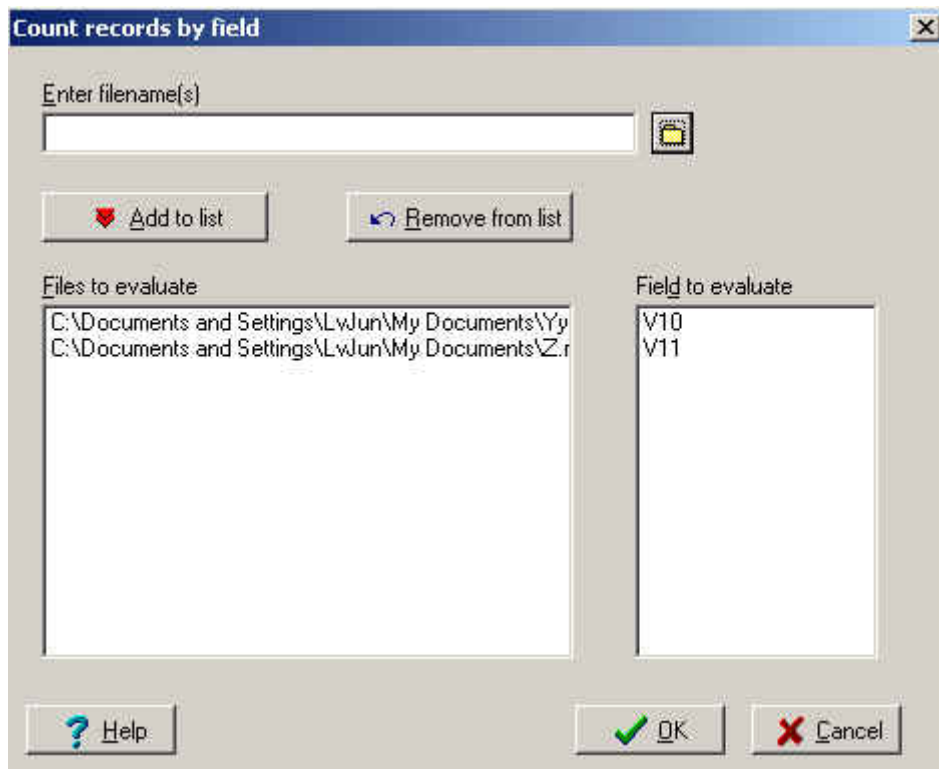
### 6.11.7 分变量统计记录数 (Count Records by Field)

这个功能可以统计数据库中某个变量中某个数值出现的次数。该功能可以同时几个数据库进行统计。

例如，一项研究同时建立了几个数据库。第一个数据库中包括了整项研究中所有病人的 ID 号和姓名。第二个数据库中包括了这些病人基本的数据（例如，年龄、身高、体重等），同时也包括了病人的 ID 号。第三个数据库包括病人的会诊资料。如果所有数据录入无误，则第一个数据库中的记录数应该等于整项研究的全部病人人数。每个记录号只对应唯一的一条记录。第二个数据库应该是每个病人有一条记录，也就是说第一个数据库中所有的 ID 号也只对应唯一的一条记录。第三个数据库的情况取决于研究本身。在这个例子里，所有病人应该至少有一条会诊记录，不过每个病人也可以同时有多条记录。也就是说，每个 ID 号必须至少对应一条记录，对应多条记录也是允许的。

利用 **Document** → **Count Records** 可以很方便地检查这些逻辑关系。在弹出的对话框中的右侧，是可供统计的变量列表，这些变量必须是在所有选择的数据库中共有的，否则将没有变量显示，这项功能也就无法使用。在变量列表中点击

一个变量，按 **OK** 键。



下面是一个输出结果的例子。两个数据库，**FileA.rec** 和 **FileB.rec** 都被选入数据库列表。选中 **ID** 变量进行统计。输出结果如下显示：

```
File 1 = d:\datafiles\FileA.rec
File 2 = d:\datafiles\FileB.rec
```

```
15 different values for ID found
```

ID	Files	
	1	2
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	.
8	1	.
9	1	.
10	1	.
11	.	1
12	.	2
13	.	1
14	.	1
15	.	1

输出结果中显示，两个数据库中有 15 个不同的 ID 值。值 1-6 在两个数据库文件中都有出现，各有 1 条记录。值 7-10 只在数据库 FileA.rec 中出现，各有 1 条记录。值 11-15 只在数据库 FileB.rec 中出现。11、13、14 和 15 各有 1 条记录，但值 12 出现在 2 条不同的记录。

## 7. 数据库的输出和输入

### 7.1 数据库的输出（Export Data）

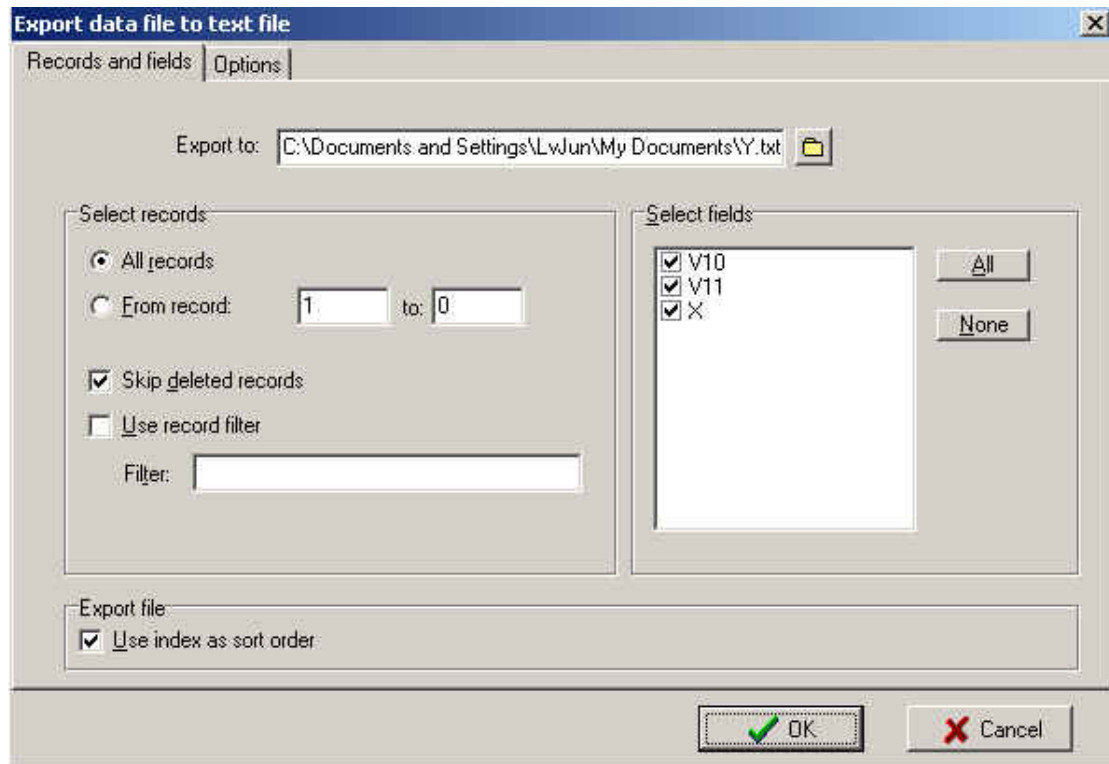
需要注意的是，如果你想输出旧的时间版本的数据库，其中的日期年还是 2 位数字，而不是 4 位数字，那么你最好先将其转为 4 位数字，然后再输出数据库。选择菜单 **Data in/out** → **Export** 或工作流程栏中的 **6.Export Data** 均可。

#### 7.1.1 备份数据库（Backup of Data）

选择 **Data in/out** → **Backup** 或工作流程栏中的 **6.Export Data** → **Backup**，选择要备份的数据库，指定保存路径，点击 **OK**，开始备份。该项功能在指定的文件夹中保存了同名的 \*.qes 文件、\*.rec 数据库、\*.chk 文件和 \*.not 文件（[备忘录文件](#)）。这项功能也可以通过 CHECK 文件命令 BACKUP 来实现。

#### 7.1.2 输出到文本文件（Export to Text File）

该功能将数据库输出为一个标准的文本文件，每行为一条记录。变量间可以以指定的字符相隔。输出文件的扩展名必须是 \*.txt。



### 选择记录 (Select records)

你可以选择输出所有记录或一定范围内的记录（例如，从第 100 条记录到第 199 条记录）；可以忽略掉标记删除的记录；可以筛选符合条件的记录。例如，

```
ID>1000
(v10<>.) AND (V100=20)
```

### 选择变量 (Select fields)

你可以选择输出的变量。

### 使用文本限定符 (Use text qualifier)

用双引号将所有非数值型的变量括起来。

### 变量分隔符 (Field separator)

选择分隔变量的符号，例如分号、逗号、TAB、或其它。

### 根据索引排序 (Use Index as sort order)

根据索引变量对记录进行排序，而不是按着记录输入顺序。

### 7.1.3 输出到 dBaseIII (Export to dBaseIII format)

将数据库输出到 dBaseIII 格式。很多数据库、电子制表软件和统计程序包都支持这种格式。变量输出对应的变量类型为：

EpiData 变量类型	dBaseIII 变量类型
整数、IDNUM	N-定数 (Fixed number)，小数点后面为 0 位
浮点数值	N-定数
文本、大写字符型文本、声索引变量、加密变量	C-字符
布尔逻辑变量	L-逻辑 (Y=真；N=假)
日期 (dmy 和 mdy)、当天日期 (dmy 和 mdy)	D-YYYYMMDD 格式的日期

注意，dBaseIII 格式限制每个数据库最多 128 个变量。选项设置类似于“[输出到文本文件](#)”中的设置。

### 7.1.4 输出到 Excel (Export to Excel)

输出数据库到 Excel 格式。Excel 的版本为 2.1，这个版本相对简单，更高版本的 Excel 程序和许多其它程序都可以兼容。变量输出对应的变量类型为：

EpiData 变量类型	Excel 单元格类型
整数、IDNUM、浮点数值	数值型
文本、大写字符型文本、声索引变量、加密变量	标注
布尔逻辑变量	逻辑 (1=真；0=假)
日期 (dmy 和 mdy)、当天日期 (dmy 和 mdy)	连续的日期数值 (按日期的格式)

注意，Excel 电子制表软件可以处理的行数和列数是有限制的。不同版本的 Excel 的限制不同，所以建议认真核对输出的文件，是否所有的数据都输出了。选项设置类似于“[输出到文本文件](#)”中的设置。

### 7.1.5 输出到 SPSS (Export to SPSS)

输出数据库到 SPSS 命令文件 (\*.sps) 和原始的数据文件 (\*.txt)。在 SPSS 中运行命令文件，将数据载入 SPSS 程序，然后将打开的数据库另存为一个真正的 SPSS 数据库。

注意，在产生的 SPSS \*.sps 文件中，“RECORDS=”的涵义不同于其在 EpiData 中的意义。在 EpiData 中，records (记录) 表示的是记录数；而在 SPSS 中，records 表示的是写下一个个人所有记录所需的行数。选项设置类似于“[输出](#)

[到文本文件](#)”中的设置。

### 7.1.6 输出到 SAS (Export to SAS)

输出数据库到 SAS 命令文件 (\*.sas) 和原始的数据文件 (\*.txt)。在 SAS 程序中提交命令文件，装载数据库。选项设置类似于“[输出到文本文件](#)”中的设置。

### 7.1.7 输出到 Stata (Export to Stata)

输出数据库到 Stata 格式 (版本 4、5、6、7、8)，版本 5 和 4 有相同的格式。在选项中，你可以设置输出数据库中变量名字母的大小写状态。Stata 数据库格式中包括数据库标签、变量标签和数值标签。不过，版本 4/5 只允许使用短标签 (数值标签最长 8 个字符)。如果标签太长，超出的字符会被截断。

EpiData 变量类型	Stata 变量类型
整数	
-长度<3	字节 (Byte)
-长度<5	整数
-长度<10	长整数 (Long integer)
-长度≥10	双实数 (Double real)
浮点数值	双实数
布尔逻辑变量	字节(0=假; 1=真)
日期 (dmy 和 mdy)、当天日期 (dmy 和 mdy)	连续的日期数值 (按日期格式, 使用%d)
文本、大写字符型文本、加密变量	字符串
声索引变量	字符串 (长度=5)

选项设置类似于“[输出到文本文件](#)”中的设置。

### 7.1.8 输出到新的 EpiData 数据库 (Export to New EpiData Data File)

输出数据库到一个新的 EpiData 数据库。利用这个功能，你可以很方便的在原始数据库的基础上，创建一个新库，其中包含原始数据库的部分记录或部分变量。例如，你想在 Epi Info v6 中分析数据，而你的 REC 数据库中的变量数又太多的情况下，就可以考虑使用这项功能。选项设置类似于“[输出到文本文件](#)”中的设置。

## 7.2 数据库的输入 (Import Data)

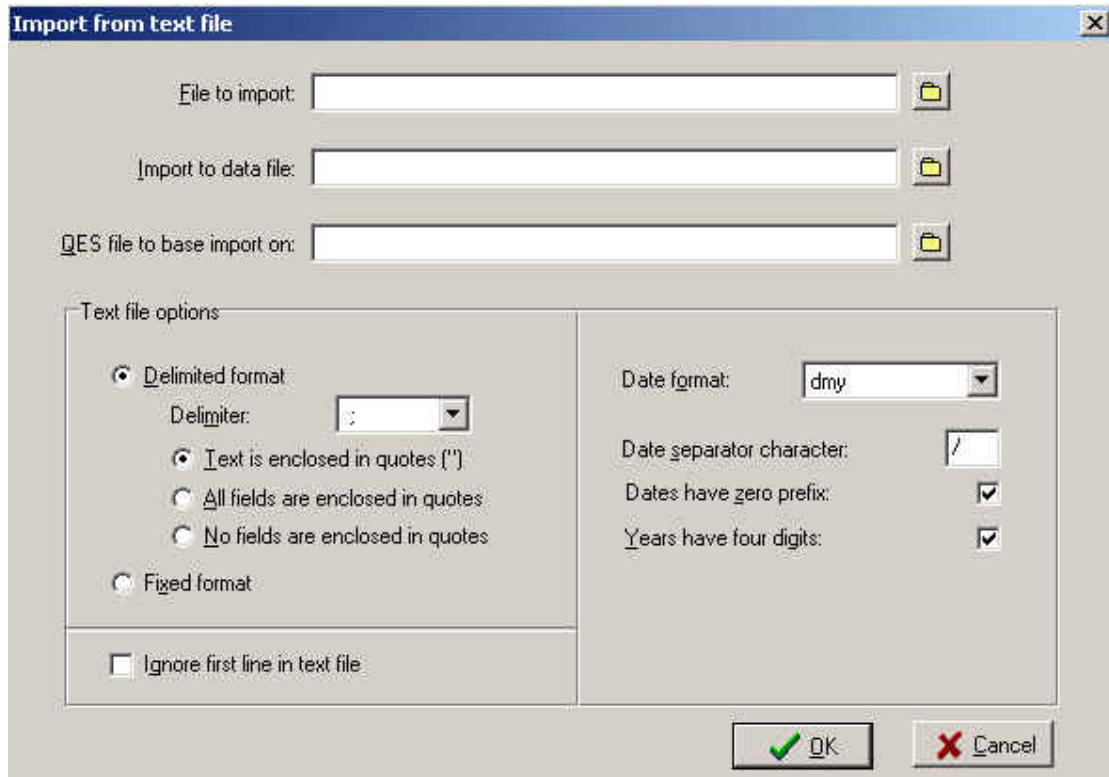
在菜单 **Data in/out** 中可以找到 **Import** 功能。

### 7.2.1 从文本文件输入 (Import Text Files)

EpiData 可以输入不同格式的保存数据的文本文件，包括分隔符 (逗号、分号、TAB 等) 分隔的或固定格式的，带文本限定符的，以及不同日期格式的。从文本文件输入数据，你必须首先创建一个 QES 文件，以此作为输入的模板，同时给文本文件中的变量一个变量名，并定义变量的类型。如果文本文件的第一行中已经包含了变量名，你必须在选项中勾选上 **Ignore first line in text file**。

如果 QES 文件中所含的变量比文本文件中的少，则文本文件中多出来的变

量将被忽略。如果 QES 文件中所含的变量比文本文件中的多，则这些多出来的变量会被设为缺失值。如果文本文件中含有日期，请确认日期格式是否正确，是否与 QES 文件中的设置一致。



选项设置如下：

#### 分隔符的格式（Delimited format）

选择文本文件中变量间使用的分隔符，可选逗号、分号、TAB 制表符、空格等。

#### 将字符/变量括起来（Enclosure of text/variables）

有三种情况可选，即在文本文件中，①所有变量，②没有变量，或③只有字符型变量要用双引号括起来的。

#### 固定格式（Fixed format）

在文本文件中，变量间未用分隔符相隔。如果选了这一项，请确认 QES 文件中定义的变量长度与文本文件中变量长度是否一致。如果创建文本文件的程序可以选择创建分隔符分隔的文本文件，还是建议你使用分隔符分隔的文本文件格式。

#### 忽视文本文件中的第一行（Ignore first line in text file）

出于方便，很多文本文件中的第一行都会列出变量名。如果存在这么一行，在输入 EpiData 时，你必须选择忽略这一行。变量名应该在 QES 文件中定义。

### 日期格式 (Date format)

选择日期中年、月、日的顺序。

### 日期分隔符 (Dates separator character)

选择用哪个字符来分隔年、月、日。如果文本文件中根本没有分隔符，请清除这个选项。

### 日期前加上 0 前缀 (Dates has prefixed zero)

如果文本文件中的日期格式采用的是如下这种格式：04052001 (2001 年 5 月 4 日)，请勾选这个选项。如果格式是 4052001，则清除这个选项。

### 年有 4 位数 (Years have 4 digits)

指出文本文件中的“年”是 4 位数 (2001) 表示的，还是 2 位数 (01) 表示的。如果用 2 位数表示，则当这 2 位数 < 50 时，程序默认为 21 世纪，即 20\*\* 年；如果这 2 位数 ≥ 50，则程序默认为 20 世纪，即 19\*\* 年。

## 7.2.2 从 dBase 文件输入

该功能将 dBase III 或 dBase IV 格式的数据库转换为 EpiData 的数据库。dBase 文件中的变量名长度可以达 11 个字符。而 EpiData 最多只允许 10 个字符的变量名，所以超出的字符会被截掉。dBase 文件中的变量类型与 EpiData 中的变量类型的对应关系如下表：

dBase 变量类型	EpiData 变量类型
C-文本	字符型变量
D-日期	日期变量，根据选项不同，可以是欧式 (dd/mm/yyyy) 或美式 (mm/dd/yyyy)
N/F-数值型	小数位数为 0，宽度小于 5 位数的变量将按整数变量输入，其它数值型变量将作为浮点变量输入
L-逻辑	布尔逻辑变量

只有上述这些变量类型可以输入 EpiData。在选项中选择输入的日期格式：<dd/mm/yyyy>、<mm/dd/yyyy>、<yyyy/mm/dd>。

## 7.2.3 从 Stata 文件输入

EpiData 可以输入 4、5、6、7、8 版本的 Stata 数据库。最多输入变量 800 个。7 和 8 版本的 Stata 文件中的变量名长度可以达到 32 个字符。而 EpiData 最多只允许 10 个字符的变量名，所以超出的字符会被截掉。6、7 和 8 版本的 Stata 文件中的变量标签最多允许 80 个字符。输入 EpiData 后只会保留前面的 50 个字符。Stata 文件中的变量类型与 EpiData 中的变量类型的对应关系如下表：



dBase 变量类型	EpiData 变量类型
字节	整数型变量 (3 位数)
整数	浮点型变量 (5 位数)
长整数	浮点型变量 (11 位数)
双实数、浮点数值	浮点型变量。如果格式编码为%fx.y, 则 x 表示变量长度, y 表示小数点后的位数。如果格式编码不是%f, 则变量长度默认设置为 18 位, 其中含 4 位小数。
字符串	字符型变量

如果数值型变量的格式为%d, 则该变量按日期变量输入。标准格式是<dd/mm/yyyy>, 不过如果指定格式为月在日前, 则也可以使用<mm/dd/yyyy>。如果 Stata 文件中含数值标签, 这些将被输入到 CHECK 文件中的 COMMENT LEGAL 命令下。在输入 Stata 8 数据库文件时, 如果发现有.a、.b 或.c 格式的缺失值, 则 CHECK 文件中会定义 MISSING VALUE。

## 8. 其它

### 8.1 选项设置 (Options)

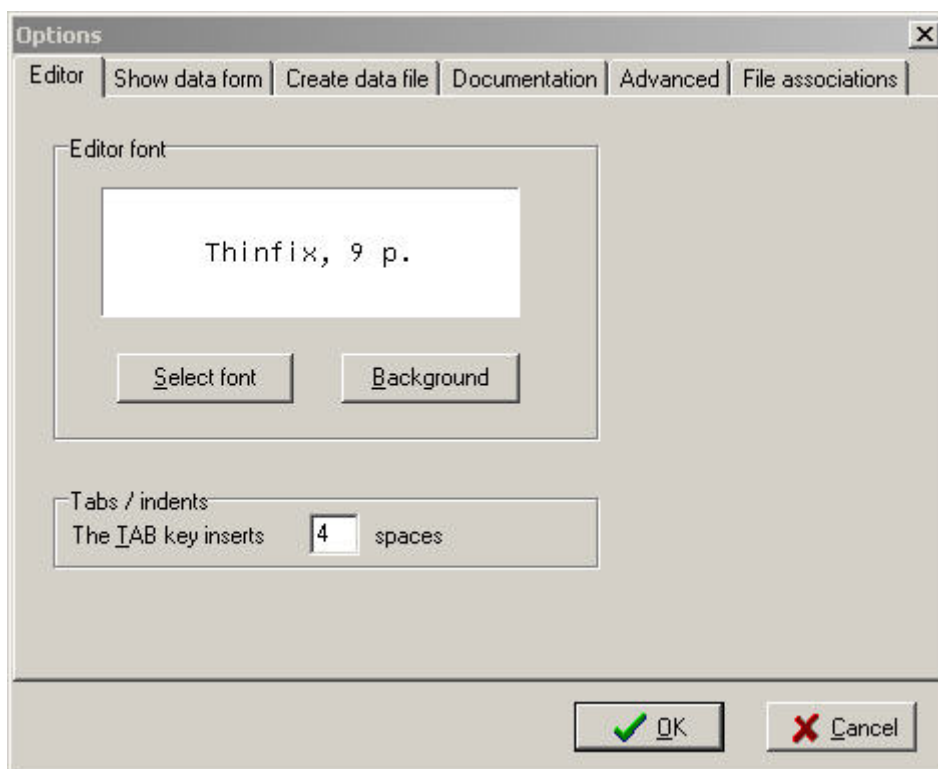
当所有窗口关闭或编辑器激活状态下, 我们可以从菜单 **File** 中选择 **Options** 进行相关的设置。所有的设置将保存在 EpiData.ini 文件中, 因此, 即使程序关闭, 再次启动时, 设置依然相同。

#### 8.1.1 编辑器选项 (Editor Options)

在这里, 你可以修改编辑器窗口中的字体和背景颜色。设置完毕后, 点击 **OK** 关闭选项设置窗口, 所有当前开启的编辑器窗口都会更新为新的字体和背景颜色。

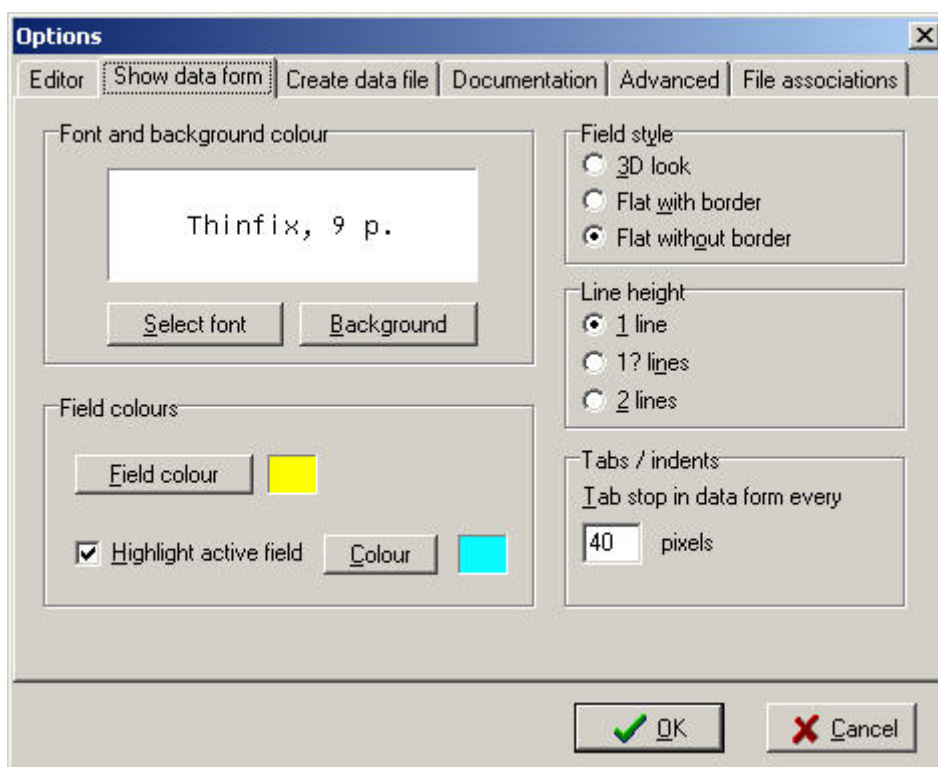
你可以修改按 **TAB** 键时, 插入的空格数。EpiData 中不能使用 Tab 字符, 所以当你打开一个文件, 或从另外一个程序中拷贝文本, 粘贴到 EpiData 的编辑器中时, 程序会自动用空格替代原有的 Tab 字符。

另外, 如果你发现用你的 EpiData 编辑器打开的含有中文的调查表文件显示为乱码, 你可以尝试在这里更换一下显示的字体, 很有可能是因为你的系统中不具备指定的字体, 才导致乱码的出现。



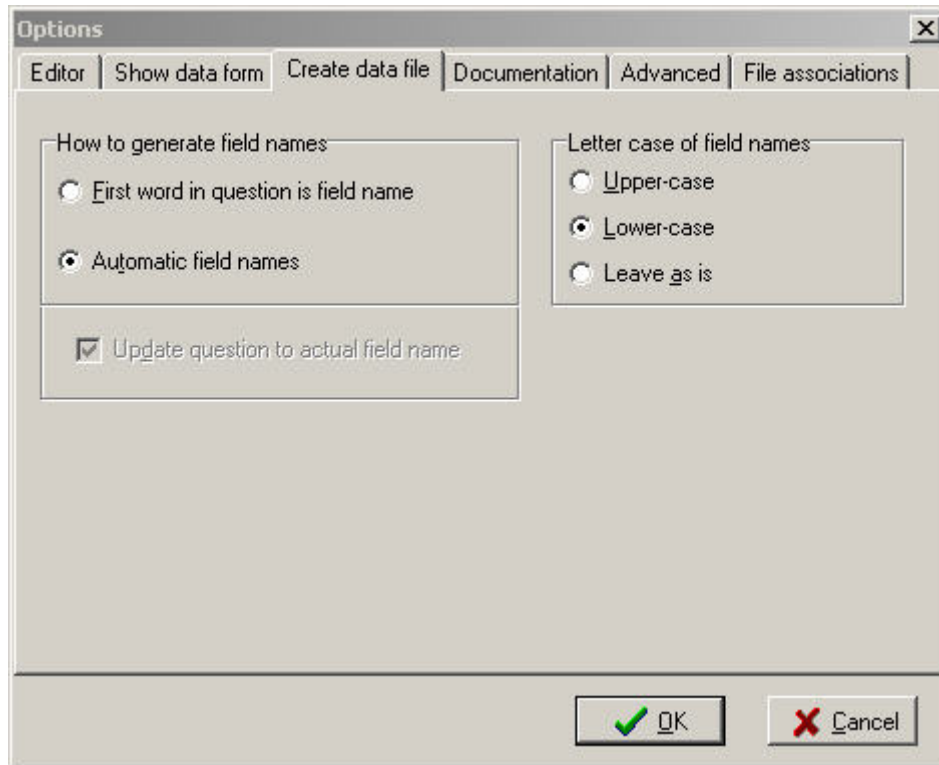
### 8.1.2 数据录入表格选项 (Show Data Form Options)

在这里，你可以定义数据录入表格的显示方式。你可以改变数据录入表格中显示的字体和背景颜色。你也可以指定录入变量的背景颜色，以及当变量被激活时的颜色。另外，还可以设置录入变量框的外观（3D 形式、平面式有边缘、平面式无边缘）、行间距、插入制表符（@）对应的制表位置间距。



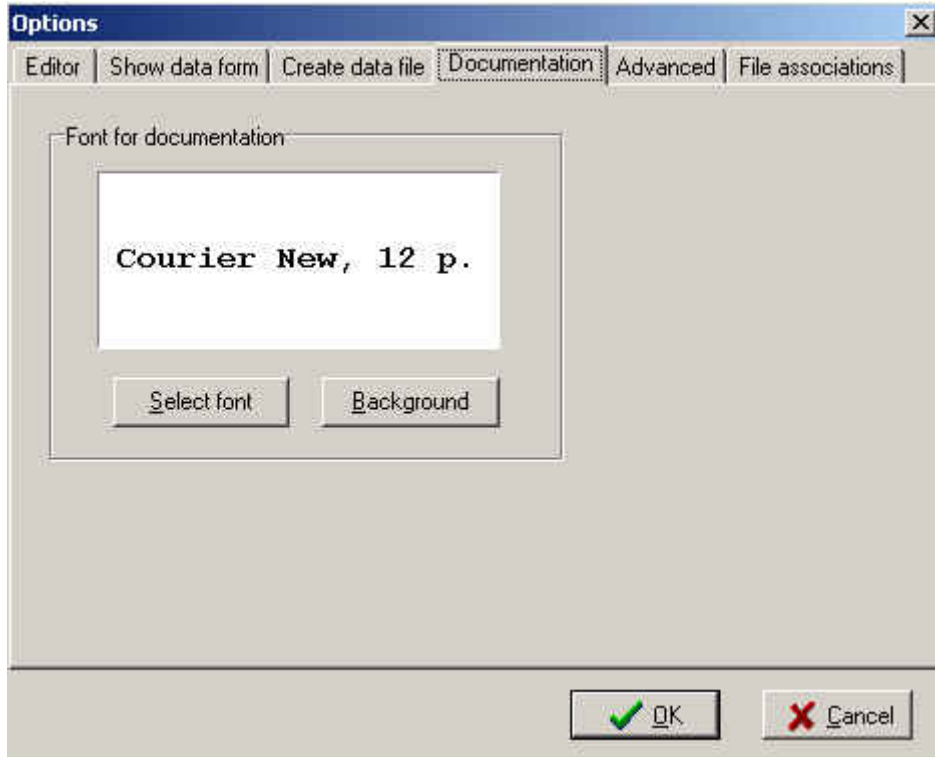
### 8.1.3 创建数据库选项（Create Data File Options）

在这里，你可以设置变量的命名方式，以及变量名是大写、小写、还是按 QES 文件中键入的实际情况显示。



### 8.1.4 输出报表选项（Documentation Options）

在这里，你可以定义输出报表（例如，`File Structure`、`List Data`、`Codebook`等）的编辑器窗口的外观（字体、背景）。



### 8.1.5 高级选项（Advanced Options）

#### ID 号变量（ID-Number Fields）

如果新数据库中含有 IDNUM 变量，该项功能可以定义第一个号从几开始。

#### 错误信息（Error Messages）

勾选此项后，CHECK 文件中设置的 IF 和 LET 命令如果有语法错误，程序会弹出错误信息提示。这样可以帮助你确定表达式不能正常工作的原因。如果没有勾选此项，则录入数据过程中，含有 IF 和 LET 的错误语句会被忽略。

#### 语言（Language）

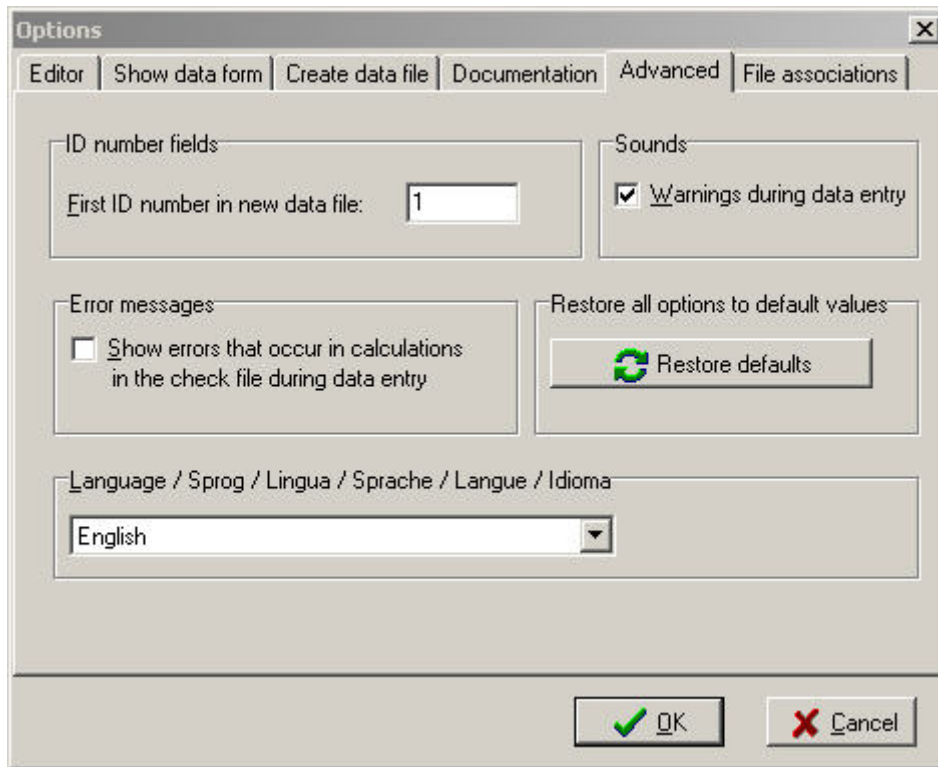
选择菜单、按钮、错误信息等中使用的语言。

#### 恢复默认选项（Restore Default Options）

将所有选项设置恢复到默认值（除了语言选项）。

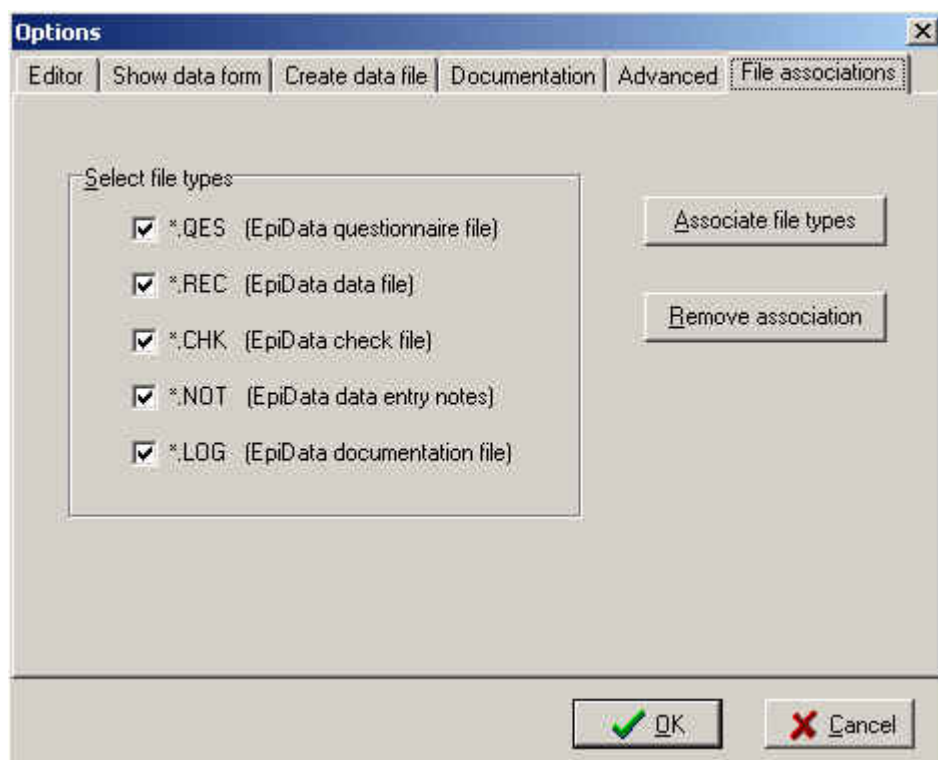
#### 声音（Sound）

如果勾选此项，当出现错误时，或有警告信息时，程序会发出标准的嘟嘟声。可参见 CHECK 文件中关于 [BEEP](#) 命令的讲解。



### 8.1.6 文件关联（File Associations）

当将 EpiData 文件类型与 EpiData 程序关联起来后，即使 EpiData 程序尚未启动，当你在资源管理器中直接双击一个数据库文件（\*.rec）时，也可以激活 EpiData 程序，打开数据库。关联的文件类型有专门的图标，你可以很容易地辨别出可用 EpiData 程序打开的文件。





## 8.2 其它

### 8.2.1 .INI 文件 (The .INI File)

所有的 EpiData 的设置都被保存在 EPIDATA.INI 文件中，该文件与 EPIDATA.EXE 在同一个文件夹中。使用者最好不要自己手动修改 INI 文件。不过，当你安装的是其它语言版本的 EpiData 程序时，你可以自己新建一个 EPIDATA.INI 文件，其中加上类似于下面这样的文字（注意不要再添加其它内容）：

```
Language=Francais
```

这么设置了以后，当你第一次运行 EpiData 时，无需再进入 **Options** 修改语言选项，程序会直接以指定的语言运行。

INI 文件中保存了最近使用的文件、窗口大小等信息。所有这些信息对每台计算机来说都是唯一的，如果强行将这个文件复制到另外一台计算机上，可能会出现奇怪的现象。

我们也可以指定程序参数 `/INI=inifilename` 运行 EpiData。这个参数指定从另外一个 .INI 文件（而不是默认的 EPIDATA.INI）装载程序运行设置，新的设置会保存在同一个文件中。当同一台计算机上有多个使用者时，利用这一功能，每个使用者可以有自己的程序设置（如窗口大小、颜色等）。

### 8.2.2 程序参数 (Program Parameters)

使用下面一个或多个参数，可以运行 EpiData。这些参数可以写在批处理 (\*.bat) 文件中，或者是使用 Windows 快捷方式。

<i>Filename.qes</i>	启动程序时打开指定的 QES 文件
<i>Filename.rec</i>	启动程序时打开指定的数据库
<code>/NOTOOLBARS</code>	隐藏两个工具条
<code>/AUTOSAVE</code>	不再显示“Save record to disk?”（是否存盘）的提示，无需询问，自动存盘。 <code>/AUTO</code> 也有相同的功能，与 Epi Info v6.xx 兼容。 <a href="#">AUTOSAVE</a> 也是一个 CHECK 命令。
<code>/CONFIRM</code>	当一个变量允许输入字符输满后，不允许程序自动激活下一个变量。 <a href="#">CONFIRM</a> 也是一个 CHECK 命令。
<code>/INI=inifilename</code>	载入指定 INI 文件中的程序设置，而不是使用默认的 EpiData.ini 文件。可参见 <a href="#">.INI 文件</a> 。

如果指定了多个文件，则只有最后一个文件会被打开。例如，建立一个批处理文件 `childprj.bat`，其中含有下列命令：

EPIDATA.EXE CHILDPRJ.REC /NOTOOLBARS

当运行该批处理文件后，EpiData 程序运行，打开数据库 childprj.rec，并隐藏工具条。

### 8.2.3 语言包

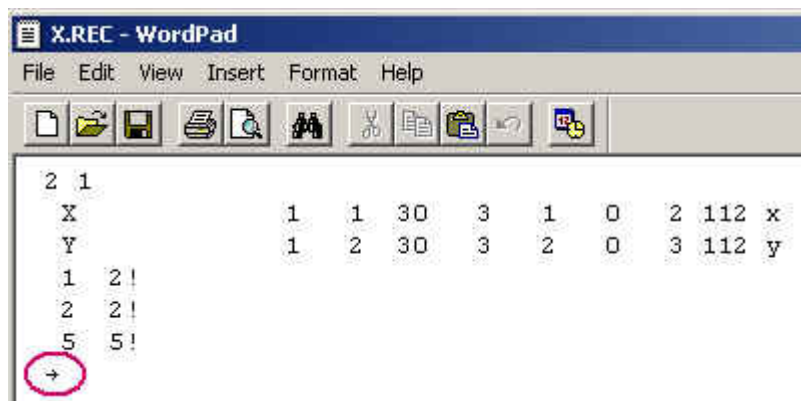
EpiData 程序主要是英语界面，但是开发者也提供了其它语言界面，使菜单、按钮、错误提示等可以以不同的语言显示。EpiData 使用哪种语言，可以在 **File** → **Options** → **Advanced** 中设置。

除了英语外，显示其它语言，需要在与 epidata.exe 相同的文件夹里有一个专门的语言文件，即 LANGUAGE.LANG.TXT。例如，中文 3.0 版本的对应文件是：chinese30.lang.txt。另外，其它的信息和帮助文件（例如，EPIDATA.HLP [帮助文件]等）也有不同的语言版本。

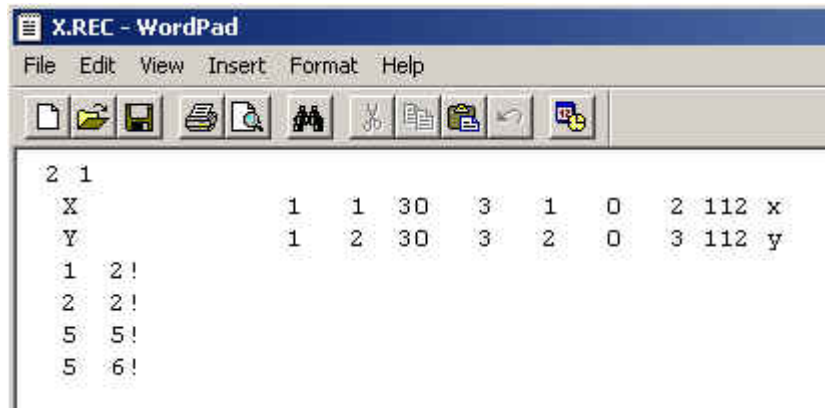
### 8.2.4 数据库结构（Datafile Structure）

EpiData 创建的数据库文件扩展名为.REC。数据库结构与 Epi Info v6 的基本一致，只是个别地方有所改动。

1. 由 Epi Info 创建的数据库中，只要不含①电话号码或②电话号码分机类型的变量，EpiData 就可以直接读取。
2. 由 EpiData 创建的数据库中，只要不含①声索引变量或②欧式当天日期（EuroToday）类型的变量，Epi Info 也可以直接读取。
3. EpiData 不会在数据库末尾添加表示文件结束（End-Of-File）的标记，但可以识别和处理由 Epi Info 在数据库中添加的这个标记。例如，下面这个是由 Epi Info 创建的数据库，可见最后有一个“→”标记。



下面这个则是经 EpiData 修改的数据库，文件最后已经没有“→”标记了。



4. 以下图为例介绍数据库文件中每行每列的涵义。数据库文件（你可以用写字板打开查看）中含有标题和所有录入的数据。标题是用来描述数据库中的变量的。标题中第一行是描述变量及标注（变量录入框旁边的解释性文字）的数目（本例中为15），以及一个编码表示录入表格的背景颜色（1表示录入框的背景颜色）。如果使用者输入了[数据库标签](#)，这一行中也会显示出来。之后，每个变量和标签各占一行。每一行中，从左至右依次表示：

- (1) 定义录入变量类型时使用的字符。
- (2) 变量名，最多10个字符，没有标点符号或空格。名字必须以字母开始，但其中可以包含数字。
- (3) 变量前解释性文字开始的列数。
- (4) 该变量距屏幕最顶端的行数。问卷的顶部为第一行。
- (5) 文字的颜色编码。
- (6) 该变量开始的列数。
- (7) 该变量开始的行数。
- (8) 变量类型的编码（见下表）。数值型变量的编码也会同时给出允许的位数。
- (9) 变量中的字符数，或变量的长度。如果是解释性文字，而非真正的可录入变量，则记录为“0”。
- (10) 录入变量的颜色编码。
- (11) 在屏幕上显示的解释性文字的内容，最多80个字符。



```

example1.rec - WordPad
File Edit View Insert Format Help
15 1 Filelabel: 一般情况调查表
_label1 1 3 30 0 0 0 0 112 调查表
#no 1 5 30 15 5 0 4 112 项目编码 no
_name 1 6 30 10 6 1 6 112 姓名 name
#sex 1 7 30 10 7 0 1 112 性别 sex
_birth 1 8 30 19 8 2 10 112 出生年月日 birth
#age 1 9 30 10 9 0 2 112 年龄 age
#marriage 1 11 30 21 11 0 1 112 1. 婚姻状况 marriage:
#year 1 12 30 25 12 0 2 112 与配偶共同生活了 year
_label2 27 12 30 0 0 0 0 112 年
#occup 1 14 30 18 14 0 1 112 2. 是否在业 occup:
#occup2 1 15 30 29 15 0 1 112 如果不在业, 目前属于 occup2
#occup1 1 16 30 18 16 0 1 112 职业是 occup1:
_other1 1 17 30 15 17 1 14 112 其它 other1
#hyp 1 19 30 25 19 0 1 112 3. 您是否患有高血压 hyp:
_today 1 21 30 16 21 10 10 112 录入日期 today
12 1 2 11 109/18/2003!

```

下表中列出了各种变量类型的编码。

编码	变量类型	变量长度	注释
0	整数	1-14	只包含数字0-9或空格。变量最长允许14个字符, 但EpiData通常会将长度 $\geq 5$ 的整数变量保存到双实数变量类型中, 该型编码为100。
1	字母	1-80	字符型变量。可以包含所有的ANSI字符
2	日期	5,8或10	美式日期变量, 即日期格式为mm/dd、mm/dd/yy或mm/dd/yyyy。具体使用的是哪种类型, 视变量长度而定。由EpiData创建的数据库通常使用4位数来表示年, 即长度为10个字符, 但是为了保证与Epi Info的兼容性, 短的日期类格式也是允许的。
3	大写字母	1-80	大写字母的字符型变量。可以包含所有大写的ANSI字符。
4	-	-	EpiData中不使用此编码。保留此编码, 以保证与Epi Info的兼容性。但据我们所知, Epi Info中也不使用这个编码。在Epi Info的源编码中, 4代表核查框“CHECKBox”。
5	布尔逻辑变量	1	布尔逻辑变量。或“是/否”变量, 可以包含空格(ANSI#32)、字母“Y”或“N”。

编码	变量类型	变量长度	注释
6	双实数	1-14	双实数变量。如果变量类型编码是6或100，则表示小数位数为0。在小数点后有≥1位数的双实数，其编码为100+小数点后的位数。小数分隔符一般用圆点(.)；当然，EpiData中也允许使用逗号(,)。例如，变量###.##，表示双实数变量，长度为6，变量类型编码为100+2=102。
7	-	-	EpiData中不支持，用于Epi Info中的电话号码变量。
8	-	-	EpiData中不支持，用于Epi Info中的时间变量。
9	-	-	EpiData中不支持，用于Epi Info中的电话号码分机变量。
10	当天日期	5,8或10	美式当天日期变量。
11	欧式日期	5,8或10	欧式日期变量，即dd/mm、dd/mm/yy或dd/mm/yyyy。具体使用的是哪种类型，视变量长度而定。由EpiData创建的数据库通常使用4位数来表示年，即长度为10个字符，但是为了保证与Epi Info的兼容性，短日期格式也是允许的。
12	IDNUM	5-14	自动编码的ID号变量，只允许输入数字0-9。
13	-	-	EpiData中不支持，此编码保留、用于Epi Info。
14	-	-	EpiData中不支持，此编码保留、用于Epi Info。
15	-	-	EpiData中不支持，此编码保留、用于Epi Info。
16	欧式当天日期	5,8或10	欧式的当天日期变量。
17	<a href="#">声索引变量</a>	5-80	声索引变量，格式为A-000，A可以是任何大写字母，000可以是任意3个数字。当将EpiData数据库转换到其它程序时，此型变量可以被转换为字符型变量，长度为5。Epi Info不支持这种变量类型。

5. 缺失值按空格 (ANSI #32) 保存，空格的数目等于对应变量的长度。
6. Epi Info只用大写字母表示变量名。而在EpiData中，你可以选择是用大写字母、小写字母或是混合大小写来表示变量名。
7. EpiData允许QES文件输入最多999行。
8. 数据库标签和[变量标签](#)：数据库标签被保存在第一行，“Filelabel:”后面。数据库标签中还会包含一个标记，显示是否使用了变量标签，因为这个会影响EpiData处理变量前“解释性文字”的方式。如果设置了变量标签的标记，则与变量前解释性文字中的第一个词会被解释为变量名，余下的文字作为变量标签。如果在数据库第一行，“Filelabel:”前面出现字符串“VLAB”（大小写不重要），或者虽然没有“Filelabel:”，但第一行中出现了“VLAB”，则表示已经设置了变量标签。

下面是一个数据库实例。可见该数据库中包含22个变量，没有单纯的解释性文字，已录入2条记录。第2条记录为标记删除（末尾以“?”结束）。没有标记删除的记录通常以“!”结束。数据库使用了变量标签。第一个变量的解释性文字是“Int1 The first integer”。但因为有VLAB标志，“Int 1”作变量名，“The first integer”作变量标签。数据库标签为“This is the filelabel”。

```

22 1 VLAB Filelabel: This is the filelabel
#INT1.....1...1..30...7...1...0...1.112.Int1.The first integer
#INT4.....1...2..30...7...2...0...4.112.Int4.A 4-digit integer
#INT6.....1...3..30...7...3.100...6.112.Int6.A 6-digit integer
#INT9.....1...4..30...7...4.100...9.112.Int9.A 9-digit integer
#INT13.....1...5..30...7...5.100...13.112.Int13.A 13-digit integer
#FIX11.....1...7..30...10...7.101...3.112.Fix1:1...
#FIX53.....1...8..30...10...8.103...6.112.Fix5:3...
#FIX104.....1...9..30...10...9.104...10.112.Fix10:4...
_TXT1.....1..11..30...7..11...1...1.112.Txt1...
_TXT4.....1..12..30...7..12...1...4.112.Txt4...
_TXT9.....1..13..30...7..13...1...9.112.Txt9...
_TXT14.....1..14..30...7..14...1..14.112.Txt14...
_UPP1.....1..16..30...7..16...3...1.112.Upp1...
_UPP4.....1..17..30...7..17...3...4.112.Upp4...
_UPP9.....1..18..30...7..18...3...9.112.Upp9...
_UPP14.....1..19..30...7..19...3..14.112.Upp14...
_BOOL.....1..21..30...7..21...5...1.112.Bool...
_SOUNDEX2.....1..22..30...11..22..17..20.112.Soundex20...
_EUDATE.....1..24..30...10..24..11..10.112.EU.date...
_USDATE.....1..25..30...10..25...2..10.112.US.date...
_EUTODAY.....1..26..30...10..26..16..10.112.EU.today...
_USTODAY.....1..27..30...10..27..10..10.112.US.today...
11111111111111111111111111111111.111.11111111.1111aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaAYF-623 01/05/200005/01/200025/05/2!
00005/25/2000!
22222222222222222222222222222222.222.22222222.2222bbbbbbbbbbbbbbbbbbbbbbbbbb!
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbBYA-535 02/05/200005/02/200025/05/2!
00005/25/2000?

```

### 8.2.5 快捷方式/鼠标 (Short-Cut Keys/Mouse)

注意，在数据录入的过程中不要使用鼠标。因为如果你通过鼠标点击来跳转变量时，CHECK 文件中的命令将无法发挥作用。只有一种情况，例如，当录入一个变量时，CHECK 设置的条件对当前记录来说确实难以满足，为了避免死循环，你可以利用鼠标直接跳到下一个变量。

如果你在某个菜单中看到有下划线，那么按 **Alt** 键，加上这个有下划线的字母，即可进入该菜单。例如，激活 **File** 菜单，可以按 **Alt+f**。其它快捷方式如下所示：

#### 1. 编辑器

<b>Ctrl+N</b>	新建编辑窗口
<b>Ctrl+O</b>	打开已有的 QES 文件
<b>Ctrl+S</b>	保存 QES 文件（不关闭窗口）
<b>Ctrl+P</b>	打印编辑器窗口中的内容
<b>Ctrl+A</b>	选择所有的文本
<b>Ctrl+C</b>	拷贝选择的文本到剪贴板
<b>Ctrl+X</b>	剪切选择的文本到剪贴板
<b>Ctrl+V</b>	将剪贴板上的内容粘贴到当前位置
<b>Ctrl+Z</b>	撤销最后一次操作

<b>Ctrl+G</b>	跳到指定的行(程序会弹出一个窗口,询问需要跳转的目标行数)
<b>Ctrl+F</b>	查找文本
<b>Ctrl+R</b>	查找和替代文本
<b>Ctrl+Q</b>	显示 <a href="#">变量类型选择列表</a> /激活变量类型选择列表
<b>Ctrl+T</b>	<a href="#">预览数据表格</a>
<b>F10</b>	关闭当前文件,回到 EpiData 的主界面。

## 2. 添加/修改 CHECK

### 2.1 当数据录入表格处于激活状态时

<b>Ctrl+Home</b>	激活第一个变量
<b>Ctrl+End</b>	激活最后一个变量
<b>F4</b>	查找变量
<b>F6</b>	激活 CHECK 功能窗口
<b>Ctrl+→</b>	激活 CHECK 功能窗口
<b>Ctrl+L</b>	编辑 Range/Legal
<b>Ctrl+J</b>	编辑 Jumps
<b>Ctrl+E</b>	在 <b>Must enter</b> 编辑窗口中的 Yes 和 No 间进行转换
<b>Ctrl+R</b>	在 <b>Repeat</b> 编辑窗口中的 Yes 和 No 间进行转换
<b>Ctrl+A</b>	编辑数值标签
<b>Ctrl+C</b>	拷贝当前变量的所有 CHECK 命令到剪贴板
<b>Ctrl+X</b>	剪切当前变量的所有 CHECK 命令到剪贴板
<b>Ctrl+V</b>	将剪贴板上的 CHECK 命令粘贴到当前变量上
<b>Alt+S</b>	保存 CHECK 文件
<b>Alt+D</b> 或 <b>F9</b>	编辑当前变量的所有 CHECK
<b>Alt+C</b>	退出 <b>Add/Revise</b> CHECK 程序
数字键盘上的 <b>+/-</b>	进入当前变量的数值标签编辑器

### 2.2 当 CHECK 功能窗口处于激活状态时

<b>F6</b>	激活数据表格窗口
<b>Ctrl+←</b>	激活数据表格窗口
<b>Enter</b>	进入下一个 CHECK 设置
<b>↑</b>	回到前一个 CHECK 设置
<b>↓</b>	跳到下一个 CHECK 设置
<b>Ctrl+↑</b>	激活前一个变量
<b>Ctrl+↓</b>	激活后一个变量
<b>Alt+S</b>	保存 CHECK 文件
<b>Alt+D</b>	编辑当前变量的所有 CHECK 命令
<b>Alt+C</b>	退出 <b>Add/Revise</b> CHECK 程序

## 2.3 录入数据

<b>Ctrl+N</b>	开始新记录
<b>Shift+Delete</b>	将当前记录标记删除/取消删除标记
<b>Ctrl+PgUp</b> 或 <b>F7</b>	显示前一条记录
<b>Ctrl+PgDn</b> 或 <b>F8</b>	显示后一条记录
<b>Ctrl+P</b>	打印数据表格
<b>Ctrl+Alt+Home</b>	显示第一条记录
<b>Ctrl+Alt+End</b>	显示最后一条记录
<b>Ctrl+Home</b>	回到当前记录的第一个变量
<b>Ctrl+End</b>	跳到当前记录的最后一个变量
<b>Ctrl+G</b>	跳到指定的记录号
<b>Ctrl+F</b>	根据当前变量的指定内容查找记录
<b>F3</b>	按照相同的搜索条件 ( <b>Ctrl+F</b> ), 继续寻找下一条记录
<b>F4</b>	查找数据录入变量
<b>Shift+F4</b> 或 <b>F4+F4</b>	查找相关变量
<b>Ctrl+←</b>	滚动到数据表格的左边缘
<b>F9</b> 或 数字键盘上的 <b>+/-</b>	打开允许录入数值的列表 (如果有的话)
<b>F5</b>	打开数据录入备忘录
<b>F10</b> (或 <b>Ctrl+R</b> )	只有在 <b>RELATE</b> 状态下: 回到 <b>RELATE</b> 的上一级数据库
<b>F10</b>	关闭数据库

## 8.2.6 主要菜单功能索引

下面列出了菜单中主要功能在本手册中讲解内容对应的章节号。



		章节号
File	<a href="#">Options</a>	8.1
Checks	<a href="#">Add/Revise</a>	4.1
Data in/out	<a href="#">Enter Data</a>	5
	<a href="#">New Data File</a>	3.1
	<a href="#">Backup</a>	7.1.1
	<a href="#">Import</a>	7.2
	<a href="#">Export</a>	7.1
	<a href="#">Append/Merge</a>	6.1
Document	<a href="#">File Structure</a>	6.11.1
	<a href="#">Data Entry Notes</a>	6.11.2
	<a href="#">View data</a>	6.11.4
	<a href="#">List Data</a>	6.11.5
	<a href="#">Codebook</a>	6.11.6

	<a href="#"><u>Validate Duplicate Files</u></a>	6.2
	<a href="#"><u>Consistency Checks</u></a>	6.3
	<a href="#"><u>Count Records</u></a>	6.11.7
Tools	<a href="#"><u>QES File from REC File</u></a>	6.4
	<a href="#"><u>Pack Data File</u></a>	6.7
	<a href="#"><u>Rebuild Indexes</u></a>	4.4 KEY
	<a href="#"><u>Revise Data File</u></a>	3.2
	<a href="#"><u>Rename Fields</u></a>	3.3
	<a href="#"><u>Edit File Label</u></a>	6.11.3
	<a href="#"><u>Copy Structure</u></a>	6.2
	<a href="#"><u>Color table</u></a>	4.4 COLOR
	<a href="#"><u>Recode Data</u></a>	6.5
	<a href="#"><u>Clear Checks</u></a>	4.1.1
	<a href="#"><u>Compress Data File</u></a>	6.8
	<a href="#"><u>Create Archive</u></a>	6.10
	<a href="#"><u>Restore Archive</u></a>	6.10