

## Exercise 9: Keeping track of data entry time

At the end of this exercise you should be able to:

- a. Edit the check file, and use temporary variables to calculate the amount of time required for data entry.

In this exercise you will learn to calculate the number of seconds which are required to complete one record. A field to retain this value as part of the database thus needs to be added to the questionnaire.

If you prepare a study, you should make it a rule to enter several hundred records by yourself (with the help of a colleague). This is important for two reasons:

- o Identify weaknesses in the data entry form and the CHK file
- o Recording the time needed to enter the information

You need the information on recording time to know what you can require from another data entry person. This is critical for making your budget. It is obviously not satisfactory to pay a data entry person according to the time the person claims to need: some people are slow and they should not receive the same remuneration as the faster person. If you take your achievement as the basis, you can objectively pay for the time required if a data entry person works at your speed: those slower than you will lose, those faster will win, and that is how it should be.

We can use EpiData Entry to monitor our data entry time and make a permanently available record that we can later analyze. In the EpiData\samples\ folder you find three files:

```
DateTime.qes  
DateTime.rec  
DateTime.chk
```

These three files are the basis to adjust our own sample files together with the **About time** in the EpiData Entry Help file. It is not an easy task and while you would certainly be able to figure it out by yourself, we will give you a helping hand.

We are going to make use of the clock of the computer. EpiData Entry has a function NOW which records the exact time of the computer. This time consists of the date and the time of the day. NOW writes the information on the date into the integer part of a field and the time of the day as a fraction of the 24-hour clock into the fractional part of the field. If we define thus such a field for the file and start counting the time when a new record is opened:

```
BEFORE FILE  
  DEFINE StartTime #####.#####  
END  
  
BEFORE RECORD  
  StartTime=NOW  
END
```

We could of course write the field `StartTime` into the questionnaire as a `NOENTER` field and then we would get, as an example:

39650.864672

NOW gives the computer time right at this point, writing the date as the integer part and the fraction of the day as the fractional part. The computer stores the date as the number of days passed since the internal anchor date. In EpiData, this anchor date is 31 December 1899. If we divide the above integer part of the number by the average number of days per year we get  $39650/365.25=108.56$ , that is a bit more than 108.5 years after the anchor date, or some time in July 2008. What about the fractional part? If we multiply the 24-hour day by this fraction we get  $0.864672*24=20.75$  which is roughly a quarter to nine in the evening, but it is much more precise than to the quarter (6 digits!) because what we really want is to have it expressed in seconds and one day has 86,400 seconds. To get the rounding proper we use even 6 rather than the bare minimum of 5 digits.

As the database only needs to retain the number of seconds required to complete one record, the number above is thus just something we need the CHK file to calculate in the background.

Assuming that we have a field SECONDS in the data file, we would then write after entering the value for the last field:

```
res3
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF seconds=. THEN
      seconds=(NOW-StartTime)*86400
    ENDIF
  END
END

AFTER RECORD
  IF idcode=. THEN
    HELP "Core information missing:\n      LABNAME, SERNO, and REGDATE\n must
all be available" TYPE=WARNING
    GOTO labname
  ENDIF
END
```

This NOW is a different NOW from the beginning (computer clock!) and all we do is to revert the difference between the two time points into seconds.

If we do it as above, the clock will stop right after the value of the last field RES1 has been entered and that value (number of seconds) will be written into the field SECONDS. Even when returning to the record, the original value in the field SECONDS will not change. If we were to take out the:

```
IF seconds=. THEN
  seconds=(NOW-StartTime)*86400
ENDIF
```

then the number of seconds would change to a new value if going again later through the record, and the value then might be lower. Ideally, however, the number of SECONDS would be cumulative, i.e. if a record is re-visited, the additional time during the revisit would be added to the pre-existing time. In other words, the time for corrections after validation would

be added. The change required to accomplish the latter is to delete it from the AFTER ENTRY statement in the RES1 field:

```
res3
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
  IF seconds=. THEN
    seconds=(NOW-StartTime)*86400
  ENDIF
END
END
```

Instead, the statements are integrated into the AFTER RECORD statement and extended to provide accumulation of time:

```
AFTER RECORD
  IF idcode=. THEN
    HELP "Core information missing:\n      LABNAME, SERNO, and REGDATE\n must
all be available" TYPE=WARNING
    GOTO labname
  ENDIF
  IF seconds=. THEN
    seconds=(NOW-StartTime)*86400
  ELSE
    seconds=seconds+(NOW-StartTime)*86400
  ENDIF
END
```

However, an even more informative approach is to have both the information about time required to enter one record for the first time and the information about the total amount of time (cumulatively) spent on a records, i.e. including the time spent on corrections. In the analysis one can then thus determine how much time is spent on entering one record, and how much additional time on correcting the record if a discordance must be resolved after validation (by subtracting the value in the field for first entry from that of the cumulative time). This is important because it has been shown that working faster is accompanied by making more errors, thus requiring revisiting more records again.

**Note on formatting:** The choice of a mixture of lower-case and upper-case letters (eg, in StartTime) is for easier visual recognition only. As mentioned earlier, EpiData Entry is not case-sensitive. This makes it particularly powerful as you can make use of formatting to visual ease. As a general rule, if you let EpiData Entry make the choices for you, commands are capitalized, all the rest is not. As for field names, lower-case is always preferred. While it does not matter in formatting, you can force its format in the REC file to any option you prefer, but we give preference to lower case (go to "File" "Options" "Create data file").

**Task:**

- o Start with the A\_EX08 QES-REC-CHK files, save them as A\_EX09 QES-REC-CHK and revise them accordingly. Don't just retype the above, try to consider the logic of it!*

- o Make two NOENTER fields, one that calculates the data entry time for the first entry (that will not change by revisiting the records) and another field that calculates the cumulative time resulting from one or more re-visits of the record*
- o Enter some data to check the functionality.*