

Part A. EpiData Entry

Part A: Quality-assured data capture with EpiData Manager and EpiData EntryClient

- Exercise 1 A data documentation sheet for a simple questionnaire
- Exercise 2 Create a basic data entry form
- Exercise 3 Create a value-label pair from external data
- Exercise 4 Create a composite identifier
- Exercise 5 Data entry and validation
- Exercise 6 Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file
- Exercise 7 Relational database

Acknowledgments:

We thank Ajay M V Kumar who had made valuable suggestions to improve the structure and flow of argumentation of the preceding version (using EpiData Entry 3.1) of Part A which we partially incorporated into this revised version.

Exercise 1: A data documentation sheet for a simple questionnaire

The first step in the process is to prepare a plan for data entry. This plan is called the **data documentation sheet**. This should not be confused with *data collection form* or *case report form* which is the proforma used for collecting the data from study participants or extracted out of the program records. The data documentation sheet is a codebook containing the details of all the variables (like field names, field labels, field type, field length, possible field values, and field value labels) to be entered.

Note: Please do not worry if you do not yet understand all the technical terms at this point in time. Be assured that you will appreciate these as you go along.

But let us proceed step by step and say that we have the following questionnaire:

Laboratory serial number: ____

Date specimen received (dd/mm/yyyy): ____/____/____

Sex: ____

Age in years: ____

Reason for examination: ____

Result of specimen 1: ____

Result of specimen 2: ____

Result of specimen 3: ____

This might present a typical simple questionnaire as used by an interviewer. Often such questionnaires are first completed on a paper *Case Report Form*. The above is actually an excerpt from the original Tuberculosis Laboratory Register proposed by The Union (note that the current version has been slightly changed):

Tuberculosis Programme

Form 2

Tuberculosis laboratory register

Year _____

Lab Serial No.	Date specimen received	Name	Sex M/F	Age	Name of referring facility	Address - patient for diagnosis	Reason for examination*		Results of specimen			Only for SS+ for diagnosis: TB Number or treatment centre**	Remarks
							Diagnosis (tick)	Month of follow up	1	2	3		

We will use this register as the basis for this course. For the time being, you plan to write a short and concise electronic data capture form, retaining only variables that are easy to capture and are likely to be useful for the analysis. *Please note this as a first principle in being efficient – capture only those variables which you will use for analysis!*

Each of the questions can be conceived of as a variable and the answer to the question as the value that the variable takes for a particular individual. **Variables** are also referred to as **'Fields'** in EpiData – both refer to the same and will be used interchangeably. We will give each variable a unique name. A completed entered data form for one study subject is called a **'Record'**. A set of such records is called a **'data file'**. The data file thus contains several records and each record contains information about one individual with respect to several variables. We are going

to describe each variable with respect to several attributes in the data documentation sheet. Let us now understand some terminology we are going to use.

- **Field name:** This is the name of the variable and in EpiData, there are certain rules to be followed in arriving at this name. We will come to these rules in a short while.
- **Field Label:** This is the descriptive name for the variable and contains a more detailed description than the variable name / field name can convey.
- **Field Type:** This describes the type of the variable – text, numeric or date being the major types.
- **Field length:** This describes the number of characters that a value can take.
- **Field values:** This describes the possible values that a variable can take.
- **Value labels:** These are descriptive names for the values. For categorical variables which are numerically coded, it is always useful to label them so that it is easier to read and understand what each of the codes mean.

“**Labels**” are also called “**metadata**” or “**data about data**”. They play a key role in data files. We may have entered a value “9” for a given field, but this number remains meaningless without specifying for what this value stands. It is important to get acquainted with these terms and understand them clearly since we will be using them frequently. We will be using several examples later in this chapter to clarify these terms.

Field name: There are some software-specific rules in naming a variable.

First, a Field name has to be **single word** that has **not more than ten characters**. This means that you cannot use a space in the name as a space makes it more than a word. Also, you cannot use any special characters like comma, semicolon, full-stop or underscore, and the first character should not be a number. Thus, do not start the variable name with a number. It cannot be ‘1v’, but it can be ‘v1’.

Note that some other analysis software may accept only a field length of eight characters. If you later plan to export your EpiData files for analysis to such a software package and you had used the full field length of ten, then your field names get truncated.

Second, use a name which is **intuitive** to understand what it means instead of generic field names like v1, v2 and so on.

Third, it may be a good practice to keep the field names in **lower case**. While EpiData is not case-sensitive, some other software is. Important among such are e.g. R and Stata® which are what we call “case-sensitive”. In the latter two, a field name of ‘sex’ (lower case), ‘SEX’ (Upper case), and ‘Sex’ (mixed case) are understood as different variables. If you later plan to export your EpiData files for analysis to such a software package, it may make life unnecessarily complicated if you have been inconsistent without a defined rule. Hence, the recommendation to keep it uniformly, “lower case”.

The following words ‘date’, ‘month’, and ‘year’ are functions in EpiData and are reserved names. Hence they cannot be used as variable names.

Field label: This is the full description of the variable and can be more than a word.

Field Type: There are different types of entry fields for the variables (we will follow the EpiData notation and call them “Fields”):

- **Text fields:** These fields take letters or numbers or a combination of these as possible values, like PETER, KOCH1882, giraffe, 45677 etc. You can type anything on the keyboard into this field. If you enter a number into such a field, it is accepted but you will not be able to make any calculation with it. These fields are also sometimes designated as character or alphanumeric fields, or most simply “**string**” (denoted by **S**) fields as they take any string of characters.
- **Numeric fields:** These are numbers. The numbers might be integers (denoted by **I**) like 885, 33, 1235 or real numbers like 3.4, 6.88, and 66.5 (also called **floats** and denoted by **F**). You can make calculations with numeric fields.
- **Date fields:** (denoted by **D**): In different countries, different ways of writing dates are used and this can be confusing for people from another culture. Some write *5 March 2005*, others *March 5 2005*, and again others *2005 March 5*. EpiData lets you choose the type of date you wish to take. We made our choice for European dates in the Preferences as we will be using European dates in this course, i.e. dates of the format *5 March 2005* or symbolized with DD/MM/YYYY.
- One other type of variables is called “**logic**” or “**Boolean**” variables. This is sometimes used in food-borne outbreak investigations. There, answers to questions on food items eaten is limited to “yes” and “no” and “missing”. There is no need for using this field type and we discourage its use as it might pose problems in analysis. The alternative is a numeric field with a label block.

While you are asked to limit the length of the field name, you have much more flexibility with the length of the value a field can take, but we will try to use it as efficiently as possible, that is we will limit the value length to the minimum needed.

Data Documentation Sheet

It is good practice to write what we call a **data documentation sheet** before you make your actual data entry form in EpiData Manager. As mentioned earlier, EpiData refers to this as **Codebook**.

In the past, fields like sex were commonly made text field with values “F” or “M” denoting Females and Males. It is efficient as it uses only a length of 1 to remain unambiguous (well, as long as the language is English or French, at least!). Things would get less efficient, if we would have to code treatment outcome with the possible values “cured”, “completed”, “died”, “failed”, “lost from follow-up”, “transferred out”, and “outcome not recorded”. Numeric coding is much simpler as there are up to ten possible values with a field length of just 1:

- 1 Cured
- 2 Treatment completed
- 3 Died of any cause
- 4 Failed bacteriologically
- 5 Lost from follow-up
- 6 Transferred out
- 9 Outcome not recorded

Later, in the analysis, you will also realize that it is very convenient to apply numeric selection criteria when you want to select a subset of data and undertake analysis only on the subset. Of course, a prerequisite is that the link between the numeric value and the text label is unambiguously clear. The role of labels is of enormous importance, also called meta-data or

“data about data” as mentioned above. We are going to make full use of numeric coding of field values and using explicit text as value labels.

Let us now go through a few examples of the variables (from the tuberculosis laboratory register example) and describe the various attributes of the variables in the data documentation sheet. As you go through, you will note that preparation of data documentation sheet requires thinking and knowledge of study data.

This is how we would write such a data documentation sheet:

Field name	Question [Field label]	Field type	Field length	Field values	Value labels	Notes [Comment]
serno	Laboratory serial number *	I	4	1-9000, 9001, 9002,		Serial number starting with 1 each year Enter 9001, 9002,... if serial number is <i>not unique</i> or <i>missing</i> , and write a data entry note
regdate	Registration date	D	10	01/01/2000-31/12/2005		Range of legal registration dates
sex	Examinee's sex	I	1	1 2 9	Female sex Male sex Sex not recorded	

* **Note:** Often, it will be preferable to make the identifier a text field. If it is a number, as in this case here with the laboratory serial number, precautions must be taken to distinguish e.g. “0001” from “1”, requiring that the numeric value is entered into one field. We can make use of the possibility in EpiData Manager to add leading zeros where appropriate.

Task:

- o Complete the data documentation sheet for all fields in the questionnaire. Note that you should always define a value if no answer was provided to a question.*
- o Think of the most efficient ways to code reason for examination and results of microscopic examination*

Solution to Exercise 1: A data documentation sheet for a simple questionnaire

Key Point(s):

- It is good practice to write a data documentation sheet before you make your actual data entry form with EpiData Manager.
- You should always define a value if no answer was provided to a question.
- “Date” is a reserved name in EpiData and cannot be used as a field name.

Task:

- o Complete the data documentation sheet for all fields in the questionnaire. Note that you should always define a value if no answer was provided to a question.*
- o Think of the most efficient ways to code reason for examination and results of microscopic examination.*

Solution:

There are many different solutions, but for the sake of uniformity, we will be using the following (but later revise some components of it) as shown on the next page.

Field name	Question [Field label]	Field type	Field length	Field values	Value labels	Notes [Comment]
serno	Laboratory serial number	I	4	1-9000 9001, 9002,		[Serial number starting with 1 each year] Enter 9001, 9002,... if serial number is <i>not unique</i> or <i>missing</i> , and write a data entry note
regdate	Registration date	D	10	01/01/2000-31/12/2005		
sex	Examinee's sex	I	1	1 2 9	Female sex Male sex Sex not recorded	
age	Examinee's age in years	I	3	0-125, 999		[Range of legal age years] Enter 999 if age not recorded
reason	Examination reason	I	1	0 1 2 3 4 5 6 7 8 9	Diagnosis Follow-up at 1 month Follow-up at 2 months Follow-up at 3 months Follow-up at 4 months Follow-up at 5 months Follow-up at 6 months Follow-up at 7 months or later Follow-up, month not stated Reason not recorded	
res1	Result of specimen 1	I	1	0 1 2 3 4 5 6 9	Negative 1+ positive 2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	[If the entered value is other than 6, then bypass next field and bypass it]
res1sc	Result of specimen 1 scanty	I	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	
res2	Result of specimen 2	I	1	0 1	Negative 1+ positive	[If the entered value is other than 6, then bypass next field and bypass it]

				2 3 4 5 6 9	2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	
res2sc	Result of specimen 2 scanty	1	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	
res3	Result of specimen 3	1	1	0 1 2 3 4 5 6 9	Negative 1+ positive 2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	<i>[If the entered value is other than 6, then bypass next field and bypass it]</i>
res3sc	Result of specimen 3 scanty	1	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	

Note the following here. For an unknown laboratory date (REGDATE), we must enter a legally existing (valid) date and we chose a legal but practically impossible date in the past, i.e. 01/01/1800. EpiData will not accept a date 99/99/9999 nor for that matter 29/02/2001. It is a personal preference of us to usually use 9 or 99 . 9 or the like to define unknown values, be this for text or numeric variables. We also introduced a “legal range” for some variables like REGDATE and AGE. We did this a bit arbitrarily, but still tried to keep it within what might be expected.

We made two fields for each result. There are 17 possibilities for a result, and therefore a length of 2 is the minimum required, but even with that, the values might not be intuitive, but they should be. An alternative version uses a float of length three to get for instance:

```
0.0      Negative
1.0      1+ Positive
2.0      2+ positive
...
0.1      Scanty, 1 AFB per 100 OIF
0.2      Scanty, 2 AFB per 100 AFB
```

Scanty results are relatively rare among positives (perhaps some 10% among diagnostic and some 20% among follow-up examinations in quality-assured high-burden country laboratories), and positives themselves are relatively rare among all (perhaps 10% to 20% among patients coming for diagnostic evaluation). Thus, scanty positive results might be only 1% of all results. We therefore chose to use integer variables and bypass the field scanty, unless the first field defines the result as quantified scanty.

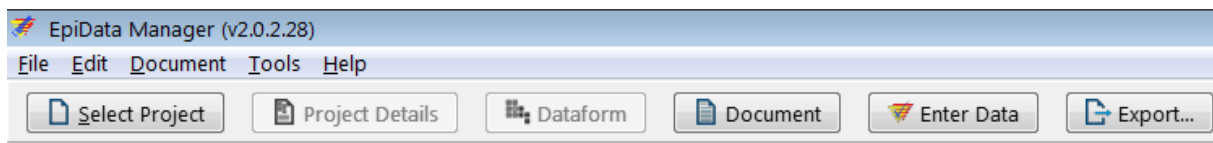
Exercise 2: Create a basic data entry form

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

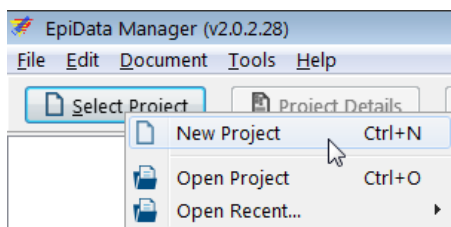
- Create and edit a data entry form with EpiData Manager.
- Include study information and password protection.
- Use headings and sections.
- Create basic label blocks, understand “Must enter”, jumps, ranges, using label blocks for missing / out of range numeric data.
- Align properly horizontally and vertically both for aesthetics and functionality.

You are now ready to start with the design of the questionnaire with EpiData Manager, based on the data documentation sheet prepared in Exercise 1.

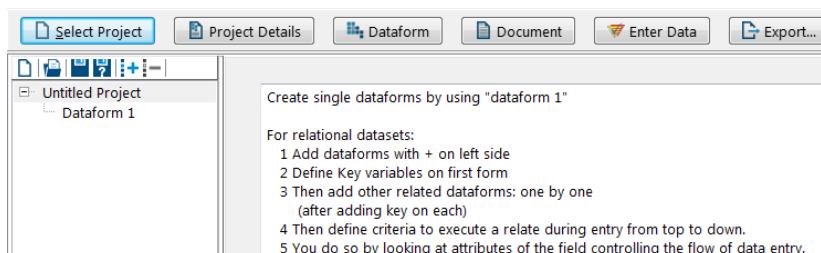
When we open EpiData Manager, we have the interface with three rows, displaying the version in the top row, menu items in the middle row and some specific menu items in the bottom row:



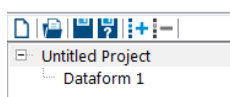
As we start from scratch, we begin with Select Project | New Project (or shortcut **CTRL+N**):



We note the opening of the set up:

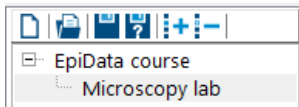


The information on the right side informs about the differences of using a single data entry form (as we will do now) or creating a relational dataset (as we will do in the last exercise of Part A). Hovering over the small icons:

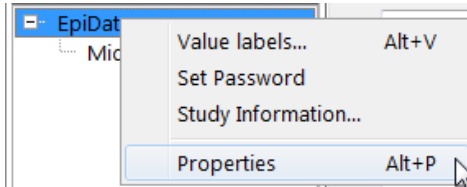


shows what they accomplish.

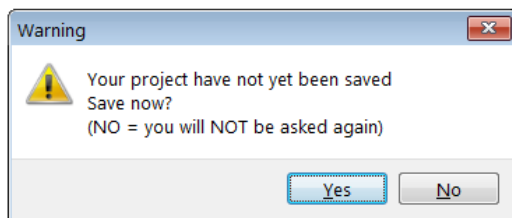
Clicking on Untitled Project and pressing function key F2 allows us changing the name to, say, “EpiData course”, and clicking on the Dataform 1 and pressing function key F2 allows us to change the name of the form to “Microscopy lab”, so that we will get:



Right-clicking on the Project name gives us options:

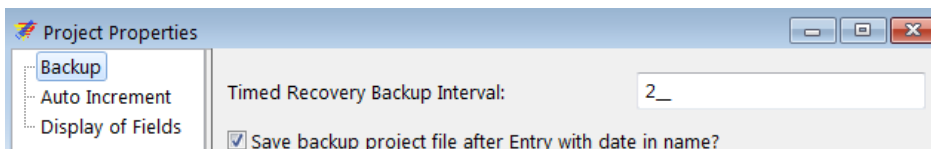


As we play around, at some point in time, EpiData Manager will prompt us to save the project:

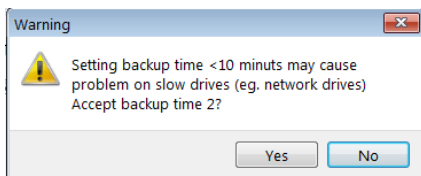


And we will do so promptly, saving the project as **a_ex02.epx** (the extension is supplied and the suggested folder is our chosen data folder C:\EPIDATA_COURSE).

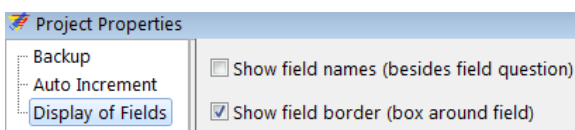
To resume from the above project options, let’s check Properties (shortcut **ALT+P**). You note that we get a window with something we had already defined earlier during set-up in Preferences:



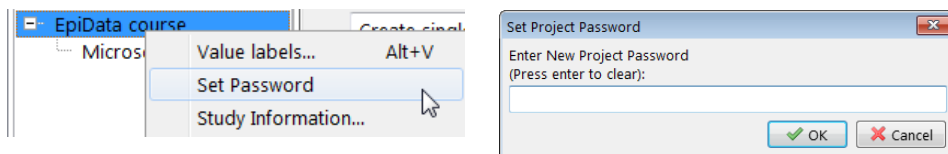
If we click on any of the other two, e.g. Display of Fields, we get a small warning (also mentioned during set-up):



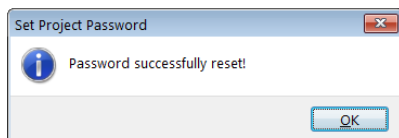
We accept and get what we had already accepted as default in Preferences:



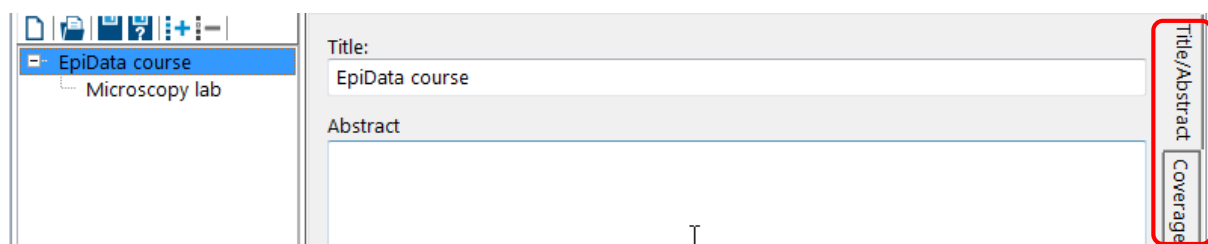
There are thus different possibilities to set our preferences and to change them here at any time without first going back to the initial settings. Thus we OK and check out Set Password with a right-click:



There are different levels of password setting, this is for the entire project, but a password can also be set for sections within a project. For the time being, we do not wish to set a password for the project and we clear with the Enter key and get:



Do you note the difference between when we click on the project name or the data form name respectively? Clicking on the project name “EpiData course” you get:

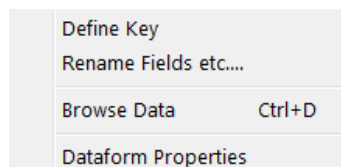
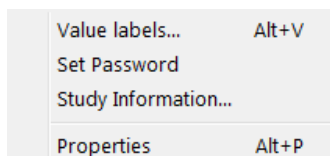


At the very right (circled red) you see that this is the first of the several vertically aligned tabs, further including Coverage, Description, Ownership, Funding, and Version Details. In the current tab we could add an Abstract, clicking on the second tab, Coverage, we can add information on Geographical location, Language, Data time coverage, Population, and Unit of observation. The third tab, Description, allows entering Keywords, Purpose, Citations, and Design. The forth tab, Ownership, allows entering Organisation/Institute, Agency, Authors/Contributors, and Rights. A fifth tab, Funding, is a free text field, and the final tab, Version, permits entering information about the study and the study form. Short, this menu option allows thorough documentation of the study. It is highly recommended that this is actually used to its fullest for a real study. For the purpose of this course, we will leave it unfilled.

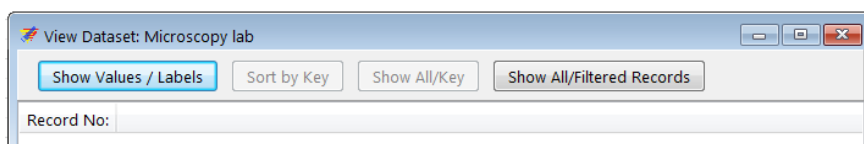
Contrasting right-clicking the Project name and the Dataform name we get:

Project name: EpiData course

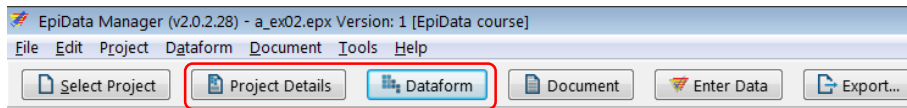
Dataform: Microscopy lab



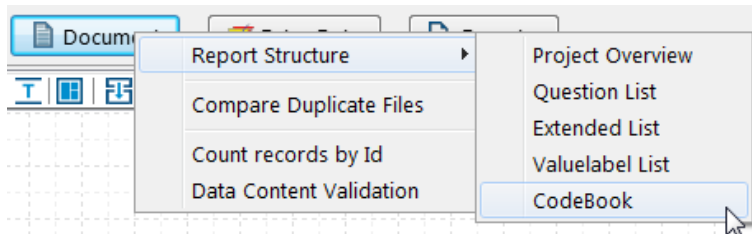
We have already checked out some of these options, while others will be used in what is coming once we start to actually make the data entry form, while further ones like Browse Data will only become available once we have data, while for now, as expected, an empty window opens:



The two sub-menus above can also be accessed any time from the Project Details and Dataform menus respectively:

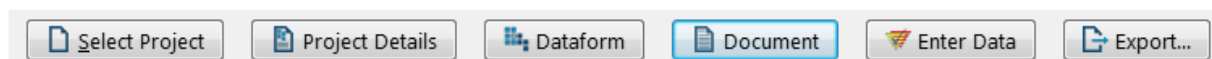


The third menu point above, Document, gives a sub-menu and the sub-menu item Report Structure has itself a sub-menu:

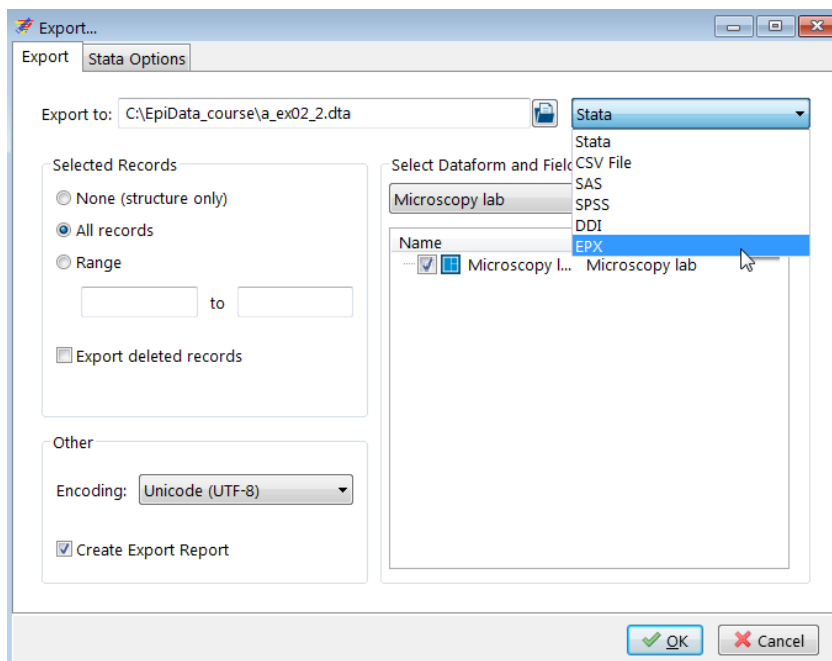


some of which we will be using at a later point of the course.

If the EpiData EntryClient software module is located in the same folder as the EpiData Manager as is the case in our set-up (and it is recommended not to keep them separately), we have direct access to it from the second-to-last menu item Enter Data:



The last menu item, Export, gives multiple options to export the structure or the data to various formats:



We will be making use of this menu item at a later point in time.

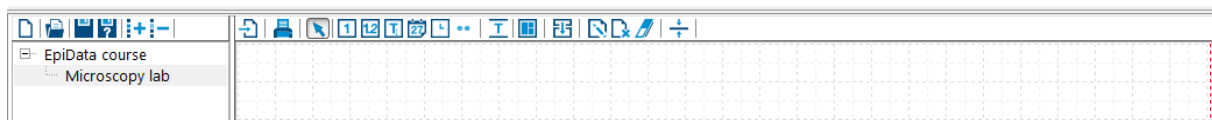
Designing the data entry form

Before starting with the actual design of our data entry form, let's clarify the terminology and where what is going to be placed. There are four elements that go with a field:

Field name	Field label	Field value	Comment / Value label
age	Person's age in years	23	Enter 999 if not recorded
sex	Person's sex	9	Not recorded

We call the variable the **Field name** (see Exercise 1). As shown in Exercise 1, the Field name may or may not be displayed. In setting up Preferences, we opted for not displaying it. To have more explicit information than what can be conveyed by the length 10, single-word Field name, we supplement the Field name with a more explanatory **Field label**. The third element on the line is the Field definition which receives the actual **Field value** during data entry. The Field value is the actual datum (singular of “data”) for that variable for that observation. For *continuous variables*, we might add an explanatory **Comment** to be displayed in a Note during data entry, here for instance what to do if the primary source has no information on this variable for this observation. For *categorical variables*, the comment is not necessary: a label block will comprehensively provide information on what is entered. As we use numeric coding, it would, however, be a nice touch if after picking a value from a pick list (label block), the **Value label** would appear to the right of the field, as a kind of visual confirmation that what had been intended was actually picked.

If we click in the Dataform tab “Microscopy lab”, we look now at the data form canvas:



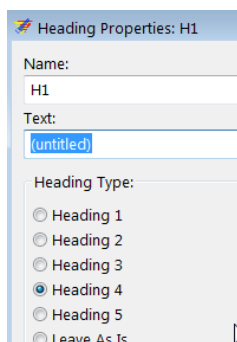
Above the grid we see various icons, most importantly right now those that symbolize different variable types which we can identify by hovering with the cursor over each one:



From the excerpt here of these icons, from left to right, these are:

- New Integer Field
- New Float Field
- New String Field
- New DMY Field
- New Time Field
- Other Field Types...
- New Heading
- New Section

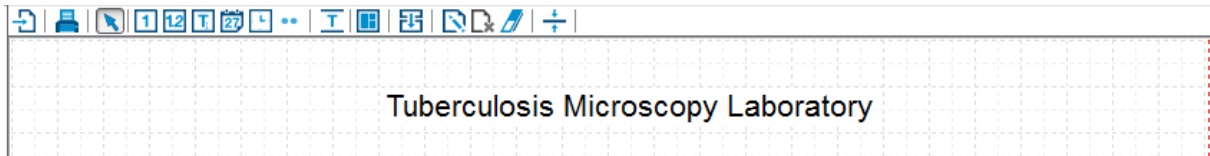
We start with the second-to-last on the left, the New Heading, click on it and get the menu:



We overwrite the Text with:

Tuberculosis Microscopy Laboratory

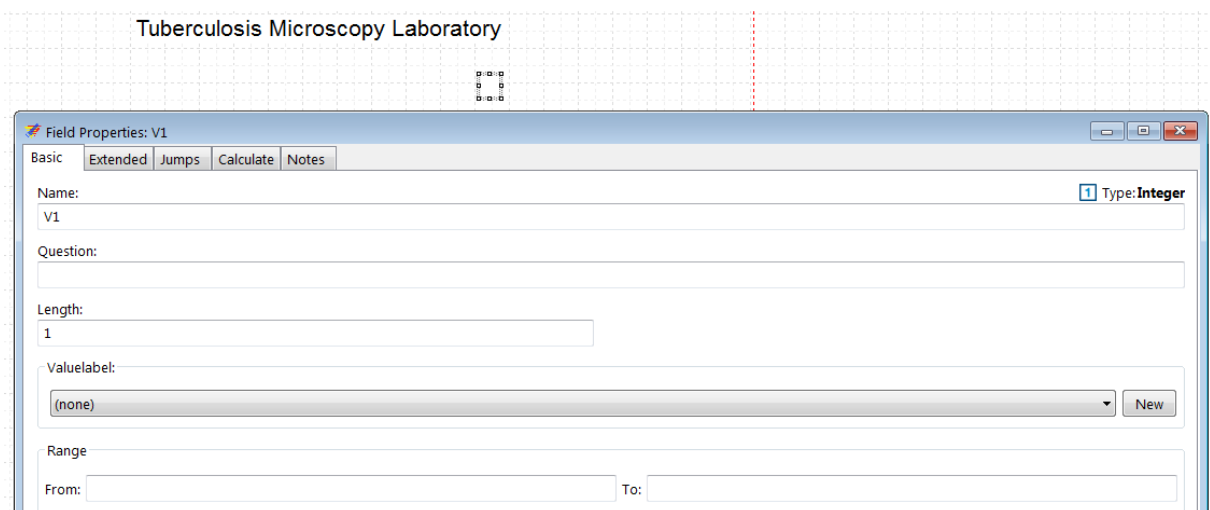
Those with some notion of HTML (the language of the internet) may recognize that text size is expressed relatively, where Heading 1 is the relatively largest and Heading 5 the relatively smallest. Let's choose (arbitrarily) Heading 2 and click Apply to get our heading:



You can drag this text box around to your preferred position. By right-clicking it, you can Edit it at any time to change the wording or the size if you wish to do so.

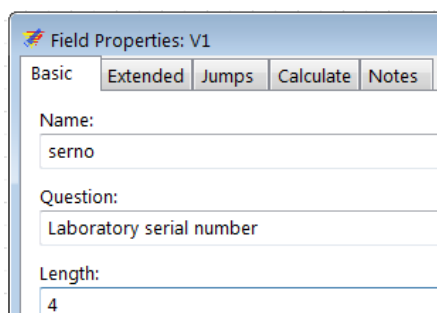
In some larger data entry forms, we then may next choose to make sections to which we could give different attributes, but for the time being, we will simply start with our fields.

Consulting the Data Documentation Sheet, we see the first field to be `serno`, the Laboratory serial number, an integer field of length 4. Clicking on the icon for the integer field, then clicking into a suitable place on the canvas, we get the Field Properties menu:

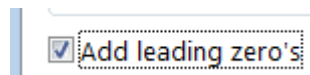


This is our main working tool for the design of the data entry form. We will therefore complete each page where applicable and study carefully the options on each tab.

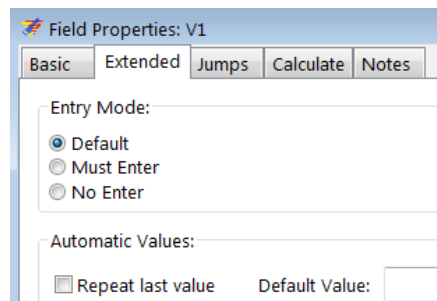
The field Name is given the default name V1 and we change this to `serno` as defined in the Data documentation sheet. The Question is for the Field label which we defined as Laboratory serial number, and its Length is 4. It has no Valuelabel (it is not a categorical variable) and we have no Range to assign, thus completed we get:



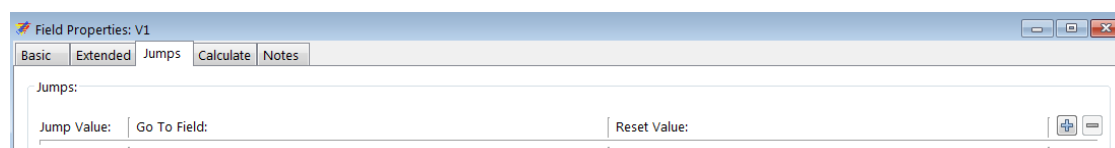
At the bottom of the Basic tab we tick to add leading zero's:



We switch to the tab Extended showing (at the top):



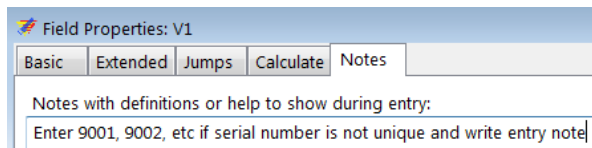
The Entry Mode is of critical importance and there are three possibilities. The Default is “you do as you wish”, i.e. the data entry person is allowed to enter or not to enter a value into the field. The second option Must Enter signifies that a value has to be entered else the data entry person cannot continue to work (or if bypassing the field by using the mouse, will be stopped latest when trying to save the record). This is our preferred method because “forgetting to enter” is not the same as “there is no information” and by forcing Must Enter a value must be entered. Therefore, whenever a value for a variable is missing, we must tell the data entry person what has to be entered in such a situation. We will thus make every field a Must Enter field with two exceptions. The first exception is for calculated fields, something we will use in a later exercise when we create a composite identifier. The second exception is for fields that can be bypassed with Jumps as we will see shortly. For the Laboratory serial number we thus tick Must Enter. There is a possibility for Automatic values, such as Repeat last value. If we tick the latter, the next new record will inherit the value from the current one for this field. This is convenient for instance if a long sequential series of records always has the same date before it changes. Whenever we change the inherited value by overwriting it in a new record, the new value will be passed on as a repeat value to the next record. Obviously, this is not what we need for the Laboratory serial number. After having ticked Must Enter, we thus move to the tab Jumps:



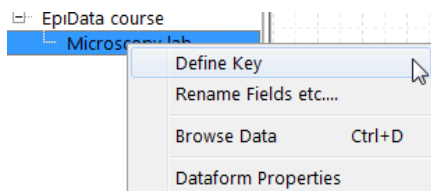
Jumps are used to define whether based on a value in the current field the next field is to be bypassed or not. If the field asks for smoking status and the response is non-smoker then the next field that would quantify the number of cigarettes per day is bypassed while it is not if the response in the current field is smoker. We will use this with the results fields.

The Calculate tab allows creating the value for a new variable from the values of one or more existing variables. For instance one could calculate the value of a date field from three date components. We generally discourage the use of calculations during data entry, they belong to the analysis, with one important exception though. Sometimes, a unique identifier for a record can only be defined by constructing from more than one variable. We will show this in a later exercise.

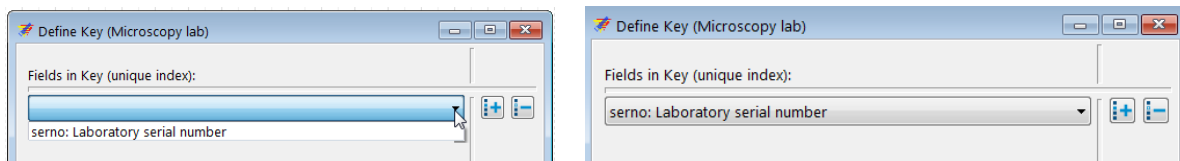
What we write into the Notes tab will briefly pop up if the cursor is in the field. For the Laboratory serial number we actually foresee such a situation, and this is to instruct to enter “artificial” serial numbers 9001, 9002, etc. if we happen to stumble upon a non-unique serial number. We thus enter a Note as for instance:



As an interlude here, we need to clarify key issues about the identifier. No variable is of more importance in a record than the unique identifier. We can even state that a record is useless without a unique identifier. For instance, if we have no value for age in a record, we can always check whether it is truly missing from the original case report form if we can identify that case report form by the unique identifier. Without it, we cannot possibly learn the truth. It is impossible to validate our data with double-entry if we do not have unique identifiers. For persons not familiar with the tuberculosis microscopy laboratory register, this is what you need to know: each examinee who presents to the laboratory for a microscopic examination is given a serial number which starts with 1 each calendar year. An examinee might thus provide a series of up to 3 specimens and the examinee (not the examination) is the unit of observation and “numbered” as such. The Laboratory serial number is unique for an examinee presenting him- or herself newly for a serial set of examinations usually over a one- to three-day period during the course of 1 year in 1 laboratory. We have not yet instructed EpiData Manager to treat the Laboratory serial number as a unique identifier. This is not done here in the Field Properties, it has to be done for the file (you may remember the options by right-clicking the data form name Microscopy lab):



It is of critical importance to Define Key, thus when done, we close the Field Properties dialog menu and go to the data form Microscopy lab, click on Define Key and select it:



What this will do is that whenever you enter a Laboratory serial number in a new record, EpiData will check the entire data set whether or not this identifier has already been used. If not, you can continue your work. If it had been used in another record, the EpiData EntryClient will tell you in which record and offer to go to it to change it if it had been entered erroneously or to enter an alternate in the current one (as suggested here in Notes). In any case, you cannot continue working unless the identifier is assuredly unique for the file.

Our next field is the Registration date with the field name regdate. We thus pick the icon for New DMY Field and start to complete the Basic tab of Field Properties. We did define a legal range. Enter it and make the field Must Enter.

The third field is `sex` which we defined as integer for length 1 and we need to create here a new Valuelabel:

The sub-menu for the Valuelabel:

Value	Label	Missing
1		<input type="checkbox"/>

It is quickly completed – note that we give a sensible, easily identifiable Valuelabel Name. Though we are quite free how we do that, taking the default is not very informative, thus:

Value	Label	Missing
1	Female	<input type="checkbox"/>
2	Male	<input type="checkbox"/>
9	Not recorded	<input type="checkbox"/>

We could tick the value for Not recorded as Missing, it might or might not come handy later in the analysis, but we leave it unticked for the time being. After accepting, the Valuelabel Name will get inserted automatically as Labelvalue into the foreseen box in the Basic tab:

else we can pick it from the drop-down list (red circled here). In the Extended tab we make the field Must Enter and tick the two boxes applicable for the Valuelabel Setting:

The first (Show valuelabel ...) is ticked by default, but the Always show picklist ... must be set. What the two accomplish is that the value goes into the field box and the Valuelabel is written to the right of the field box (for visual verification), but before this is done, the Picklist with the Value-Valuelabel pairs is popping up ready for selection as soon as the cursor gets into the field. In EpiData Manager, it gets indicated in green that we have a given label block:

Examinee's sex: label_sex

In the EntryClient, ticking both boxes will give the pop-up (left) and the writing of the value label (right) after the selection of the value:

Tuberculosis Microscopy Laboratory

Laboratory serial number 0001

Registration date 01-01-2000

Examinee's sex ☐

Value Labels

Value	Label
1	Female
2	Male
9	Not recorded

Tuberculosis Microscopy Laboratory

Laboratory serial number 0001

Registration date 01-01-2000

Examinee's sex ☒ Male

The fourth variable is age, the Examinee's age in years, with a field length of 3, and for which we allow a range from 0 to 125, and a legal value of 999 for missing age:

Field Properties: V4

Basic Extended Jumps Calculate Notes

Name: age 1 Type: Integer

Question: Examinee's age in years

Length: 3

Valuelabel: (none) New

Range: From: 0 To: 125

The out-of-range legal value is put into a Valuelabel, thus pick New and add as follows:

Field Valuelabel Editor

Valuelabel Name: label_age

Value	Label	Missing
999	Age not recorded	<input checked="" type="checkbox"/>

Then set to Must Enter in the Extended tab, but set / keep all boxes in Valuelabel Setting unticked: the value-label pair is just to “legalize” the value 999, not to have a rudimentary pop-up window. Instead, we instruct the data entry person in the Notes tab what to do if the value for age is missing:

Field Properties: age

Basic Extended Jumps Calculate Notes

Notes with definitions or help to show during entry:

Enter 999 if missing

The fifth variable is reason, the Reason for examination, an integer of length 1. The principle is the same as for sex and nothing is new here above what was learned before.

The sixth variable is res1, the Result of specimen 1, an integer of length 1. Again, the principle is the same as for the other categorical variables with label blocks, but we do have something new. We will insert a Jumps command for every value, except if it is 6, Scanty quantified. Because we need to tell EpiData where to go, if the field has or has not the value 6, we propose that everything else but the Jumps command for fields res1, res1sc, and res2 is completed first, and then we return to res1 to formulate the Jumps command (not

critical here as will be seen). We don't have to, but let's arrange the fields for `res1` and `res1c` (and the other "pairs") horizontally:

Note also that we do not need to write a label block for each result and for each scanty result, it suffices to write a generic label block for results as all three results have the same Value label blocks and we may name it `label_result` and name the Value label block for scanty results something like `label_scanty`. After we have it for the first, we can pick it from the drop-down list for the Value label:

Once we are done with these, we return to the field `res1` and open the Jumps tab to enter what we require. Because the field to skip is the next field (`res1sc` here), we do not actually need to state to which field to go. We just add all values except 6 and the instruction (Skip Next Field) under the Go To Field. Click the + sign to the right to add a new line and enter the next value:

Tasks:

- o *Finalize the data entry form for the remaining results variables. Note that the fields for scanty results cannot be **Must Enter**. Note further the options in the Drop down menu for what is the default "Skip Next Field" to pick the best option for the last result.*
- o *Align the field correctly using the Alignment icon. Note that correct vertical alignment is critical if two variables are on the same horizontal pane (EpiData Manager gives nice blue and red guiding lines). If two variables on the same horizontal pane are vertically mismatched, the sequence of data entry will go wrong!*
- o *Tell EpiData Manager to create the Codebook and save the output file as a text file "`a_ex02_codebook.txt`".*

Solution to Exercise 2: Create a basic data entry form

Key Point(s):

- The identifier is the most important variable in a record
- EpiData is not case-sensitive, but some other software is. It is therefore advisable to use a simple rule such as consistently using lower-case for field names.

Tasks:

- o *Finalize the data entry form for the remaining results variables. Note that the fields for scanty results cannot be **Must Enter**. Note further the options in the Drop down menu for what is the default “Skip Next Field” to pick the best option for the last result.*
- o *Align the field correctly using the Alignment icon. Note that correct vertical alignment is critical if two variables are on the same horizontal pane (EpiData Manager gives nice blue and red guiding lines). If two variables on the same horizontal pane are vertically mismatched, the sequence of data entry will go wrong!*
- o *Tell EpiData Manager to create the Codebook and save the output file as a text file “a_ex02_codebook.txt”.*

Solution:

The data entry form may look as follows:

The screenshot shows a data entry form titled "Tuberculosis Microscopy Laboratory" on a grid background. The form contains the following fields:

- Laboratory serial number:
- Registration date:
- Examinee's sex: label_sex
- Examinee's age in years:
- Reason for examination: label_reason
- Result of specimen 1: label_result
- Result of specimen 1 scanty: label_scanty
- Result of specimen 2: label_result
- Result of specimen 2 scanty: label_scanty
- Result of specimen 3: label_result
- Result of specimen 3 scanty: label_scanty

Once a data entry form has been prepared, it is best to test it right away in the EntryClient with some fake data (without saving) to identify quickly problems.

You can leave the data entry form open in EpiData Manager and access the EpiData EntryClient via the menu and after prompting the Manager will close.

Opening in the EntryClient:

Tuberculosis Microscopy Laboratory

Laboratory serial number

Registration date

Examinee's sex

Examinee's age in years

Reason for examination

Result of specimen 1 Result of specimen 1 scanty ☐

Result of specimen 2 Result of specimen 2 scanty ☐

Result of specimen 3 Result of specimen 3 scanty ☐

we see that all fields which should be Must Enter actually are (orange-brownish color) and that the fields which can be bypassed (for quantified scanty results) are not.

Entering fake data, we get:

Tuberculosis Microscopy Laboratory

Laboratory serial number

Registration date

Examinee's sex Female

Examinee's age in years

Reason for examination Diagnosis

Result of specimen 1 Negative Result of specimen 1 scanty ☐

Result of specimen 2 Scanty, quantified Result of specimen 2 scanty ☐ 8 AFB per 100 OIF

Result of specimen 3 Result not recorded Result of specimen 3 scanty ☐

Confirmation

Save Record?

Yes No Cancel

It all looks as it should, neat and nicely. Do not save the record and exit without saving.

We saved the CodeBook as text file and can look at it in a text editor. It must reflect what we defined in the Data documentation sheet but is now much more detailed and a superb document that can be shared with others who later collaborate in the analysis:

```
=====
Report: CodeBook
Created: 28-04-2015 18:33:16
=====

-----
File 1: C:\EpiData_course\a_ex02.epx
-----

File 1: C:\EpiData_course\a_ex02.epx
-----
Title      EpiData course
Created    28-04-2015 09:50:32
Last Edited 28-04-2015 18:06:40
Version    1
Cycle      13
-----
Backup on shutdown: yes
Encrypted data: no

Dataforms:
-----
```

Caption	Created	Structure Edited	Data Edited	Sections	Fields	Records	Deleted
Microscopy lab	28-04-2015 09:50:32	28-04-2015 18:06:40	28-04-2015 09:50:32	1	11	0	0

```
=====
Dataform: Microscopy lab
=====
```

Name	Type	Length	Missing	Value(s)	Value Label	Question / Caption
H1	Heading					Tuberculosis Microscopy Laboratory
serno	Integer	4				Laboratory serial number
regdate	Date (DMY)	10				Registration date
sex	Integer	1		label_sex		Examinee's sex
age	Integer	3		label_age		Examinee's age in years
reason	Integer	1		label_reason		Reason for examination
res1	Integer	1		label_result		Result of specimen 1
res1sc	Integer	1		label_scanty		Result of specimen 1 scanty
res2	Integer	1		label_result		Result of specimen 2
res2sc	Integer	1		label_scanty		Result of specimen 2 scanty
res3	Integer	1		label_result		Result of specimen 3
res3c	Integer	1		label_scanty		Result of specimen 3 scanty

Field: serno: Laboratory serial number

Field: regdate: Registration date

Field: sex: Examinee's sex

```
Value label: label_sex [I]: (Integer)
```

Field: age: Examinee's age in years

```
Value label: label age [I]: (Integer)
```



```

Value Label                Missing (M), set: label_age
-----
999   Age not recorded
-----

.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.

Field: reason: Reason for examination
-----
Type                Integer
Length              1
Entry Mode          Must Enter
Show Value Label    true
Show Picklist       true
-----

Value label: label_reason [I]: (Integer)
-----
Value Label                Missing (M), set: label_reason
-----
0      Diagnosis
1      Follow-up at 1 month
2      Follow-up at 2 months
3      Follow-up at 3 months
4      Follow-up at 4 months
5      Follow-up at 5 months
6      Follow-up at 6 months
7      Follow-up at 7 months or later
8      Follow-up, month not stated
9      Reason not recorded
-----

.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.

Field: res1: Result of specimen 1
-----
Type                Integer
Length              1
Entry Mode          Must Enter
Jumps               0 > Skip Next Field
                   1 > Skip Next Field
                   2 > Skip Next Field
                   3 > Skip Next Field
                   4 > Skip Next Field
                   5 > Skip Next Field
                   9 > Skip Next Field
Show Value Label    true
Show Picklist       true
-----

Value label: label_result [I]: (Integer)
-----
Value Label                Missing (M), set: label_result
-----
0      Negative
1      1+ positive
2      2+ positive
3      3+ positive
4      Positive, not quantified
5      Scanty, not quantified
6      Scanty, quantified
9      Result not recorded
-----

.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.^-.-^-.

Field: res1sc: Result of specimen 1 scanty
-----
Type                Integer
Length              1
Show Value Label    true
Show Picklist       true
-----

Value label: label_scanty [I]: (Integer)
-----
Value Label                Missing (M), set: label_scanty
-----
1      1 AFB per 100 OIF
2      2 AFB per 100 OIF
3      3 AFB per 100 OIF
4      4 AFB per 100 OIF
5      5 AFB per 100 OIF
6      6 AFB per 100 OIF
7      7 AFB per 100 OIF
8      8 AFB per 100 OIF
9      9 AFB per 100 OIF
-----

```

..^..^..^..^..^..^..^..^..^..^..^..^..^..^..^..

Field: res2: Result of specimen 2

```
-----
Type                Integer
Length              1
Entry Mode          Must Enter
Jumps               0 > Skip Next Field
                   1 > Skip Next Field
                   2 > Skip Next Field
                   3 > Skip Next Field
                   4 > Skip Next Field
                   5 > Skip Next Field
                   9 > Skip Next Field
Show Value Label    true
Show Picklist       true
-----
```

Value label: label_result [I]: (Integer)

```
-----
Value Label          Missing (M), set: label_result
-----
0      Negative
1      1+ positive
2      2+ positive
3      3+ positive
4      Positive, not quantified
5      Scanty, not quantified
6      Scanty, quantified
9      Result not recorded
-----
```

..^..^..^..^..^..^..^..^..^..^..^..^..^..^..^..

Field: res2sc: Result of specimen 2 scanty

```
-----
Type                Integer
Length              1
Show Value Label    true
Show Picklist       true
-----
```

Value label: label_scanty [I]: (Integer)

```
-----
Value Label          Missing (M), set: label_scanty
-----
1      1 AFB per 100 OIF
2      2 AFB per 100 OIF
3      3 AFB per 100 OIF
4      4 AFB per 100 OIF
5      5 AFB per 100 OIF
6      6 AFB per 100 OIF
7      7 AFB per 100 OIF
8      8 AFB per 100 OIF
9      9 AFB per 100 OIF
-----
```

..^..^..^..^..^..^..^..^..^..^..^..^..^..^..^..

Field: res3: Result of specimen 3

```
-----
Type                Integer
Length              1
Entry Mode          Must Enter
Jumps               0 > Save Record
                   1 > Save Record
                   2 > Save Record
                   3 > Save Record
                   4 > Save Record
                   5 > Save Record
                   9 > Save Record
Show Value Label    true
Show Picklist       true
-----
```

Value label: label_result [I]: (Integer)

```
-----
Value Label          Missing (M), set: label_result
-----
0      Negative
1      1+ positive
2      2+ positive
3      3+ positive
4      Positive, not quantified
5      Scanty, not quantified
6      Scanty, quantified
9      Result not recorded
-----
```

```

-----
.-^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.
Field: res3c: Result of specimen 3 scanty
-----
Type                Integer
Length              1
Show Value Label    true
Show Picklist       true
-----

Value label: label_scanty [I]: (Integer)
-----
Value Label          Missing (M), set: label_scanty
-----
1      1 AFB per 100 OIF
2      2 AFB per 100 OIF
3      3 AFB per 100 OIF
4      4 AFB per 100 OIF
5      5 AFB per 100 OIF
6      6 AFB per 100 OIF
7      7 AFB per 100 OIF
8      8 AFB per 100 OIF
9      9 AFB per 100 OIF
-----
.-^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.^-.

```

Exercise 3: Create a value-label pair from external data

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

- a. Create an EpiData two-variable dataset from a text file
- b. Use an external two-variable dataset to create a label block internally or externally

In the previous Exercise we worked with small label blocks of not more than ten category levels. These are quickly created directly in the data form. Often we encounter the need for larger standard label blocks. This might be a list of administrative units in a country and it may have hundreds of levels. Such lists might be publicly available on the internet from the government and they can come in different formats such as text files or spreadsheets. In order to standardize as much as possible for our data collection, we should take recourse to official lists whenever possible, rather than inventing our own parallel system. If we can get such a list into a formatted file for EpiData, it might also be used in more than a single project.

In this Exercise, we provide a spreadsheet which contains two variables, the name of a given tuberculosis laboratory paired with its code, not dissimilar to a list of communities paired with their zip codes.

Formatting the external file

The first thing to know is the file type, and into which format it should be brought. We have a spreadsheet `a_ex03_namecode.xls` with two columns:

	A	B
1	Awuna	ML_J
2	Beitbridge	MS_D
3	Bindura	MC_A
4	Binga	MN_G
5	Birchenough	ML_M
6	Bonda	ML_I
7	Brunapeg	MS_G
8	Chagutu	MM_I

You remember how the label block for the field `sex` was set up:

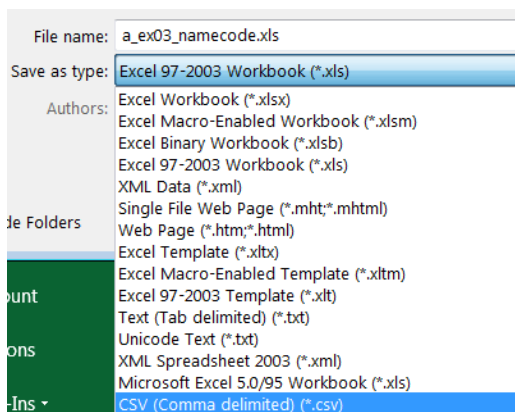
Value	Label
1	Female
2	Male
9	Not recorded

On the left is the Value that is written into the field and on the right the Label that provides the meta-data. The setup is thus the Value presenting a code on the left and the Label on the right. With short lists there is no issue because our eye captures quickly both Value and Label, but how about a very long list? How do we search a list of communities and their zip codes? People tend to know the name of a community, but not their zip codes. We are also used to search alphabetically by name not by zip code, yet we wish the code to be the value to be written into the field, not the name. EpiData makes it easy for us: we can search substrings, and it will find them in either column! If we start typing `birch...` in EpiData EntryClient, the software will

go to the first appearance of these letters be they in the left or the right column. Therefore, what we might do preferably, is to put the code of the laboratory into the left column and the name of the laboratory into the right and sort alphabetically by the name of the laboratory (not its code). This way the laboratory code is the value that is written into the field and the laboratory name is its label. Therefore invert to get:

	A	B
1	ML_J	Awuna
2	MS_D	Beitbridge
3	MC_A	Bindura
4	MN_G	Binga
5	ML_M	Birchenough
6	ML_I	Bonda
7	MS_G	Brunapeg
8	MW_J	Chegutu
9	MV_I	Chikombedzi
10	MC_H	Chimhanda

EpiData cannot read a spreadsheet file with an *.xls extension, but it reads text files with a *.txt extension or the common text standard of comma-delimited *.csv files. Conversely, most software, including proprietary Excel® or the non-proprietary LibreOffice Calc spreadsheets can save a file as a *.csv file. Use thus Save as (in Excel®) in search for the appropriate file type:



A word on delimiters: we do need a delimiter. A delimiter is an element that separates two variables. In normal text, a space is the delimiter separating two words. For our purpose, neither spaces nor tabs are good separators because they could be part of a variable. Indeed, the name for one laboratory is Collin Saunders. If we had a space as separator, these two components would erroneously be interpreted as two variables. A comma as in *.csv is better, but it is not necessarily fail-safe, best would be semi-colon delimited: it is easy to see and it is rarely used. In any case, in our 95 records (please visualize them in your text editor):

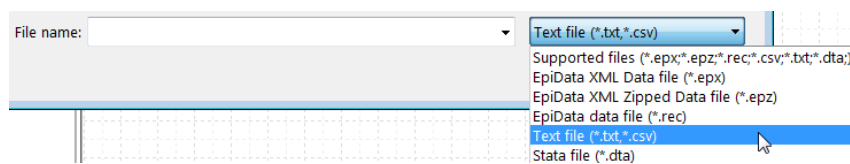
```
ML_J,Awuna
MS_D,Beitbridge
MC_A,Bindura
MN_G,Binga
ML_M,Birchenough
ML_I,Bonda
MS_G,Brunapeg
MW_J,Chegutu
MV_I,Chikombedzi
MC_H,Chimhanda
```

there are no commas apart from the desired ones to separate the two variables. We almost have now the format we require to make an EpiData *.epx file out of the *.csv file. **Important Note:** EpiData Manager interprets the first line as header but because the file does not have headers, the first line will erroneously be cut off. Therefore, we add a header to the *.csv file as follows:

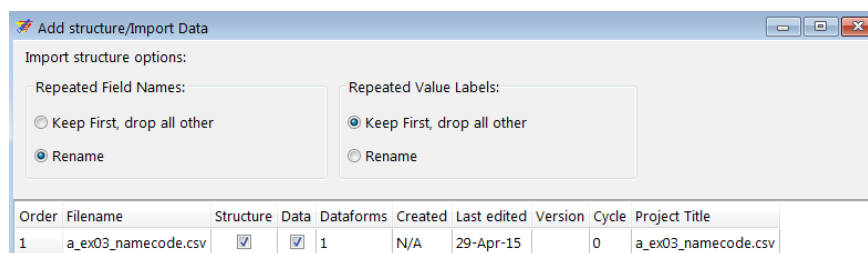
```
labcode,labname
ML_J,Awuna
MS_D,Beitbridge
MC_A,Bindura
MN_G,Binga
```

Creating an EpiData *.epx file from a *.csv file

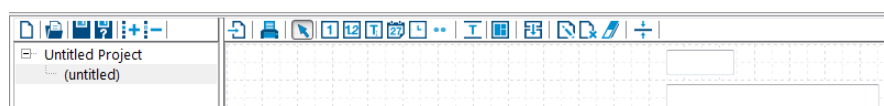
Open EpiData Manager and File | Import file ..., and select the correct file type:



Pick the a_ex03_namecode.csv file. This will open the dialog window:



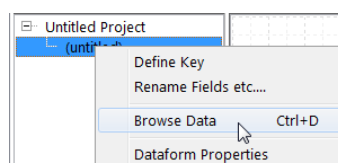
Note here the helpful options should you expect duplicates in Field Name – Field Value Labels. We can leave the defaults as they are and click OK. This gives us an Untitled Project with an (untitled) data form:



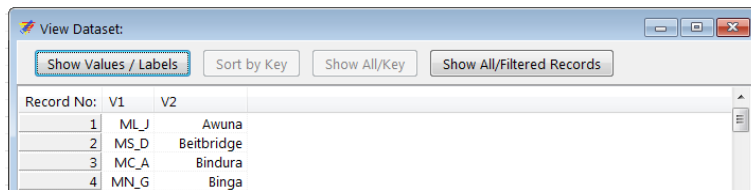
At the bottom of the screen we see that we have correctly 95 records:



Whether or not prompted to save, we save the file as a_ex03_namecode.epx. Right-clicking the data form (or with **CTRL+D**):

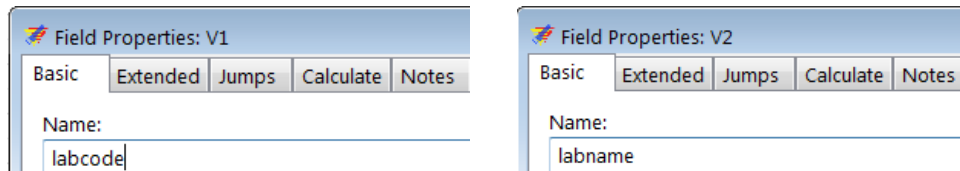


we browse to view the data set:



Record No:	V1	V2
1	ML_J	Awuna
2	MS_D	Beitbridge
3	MC_A	Bindura
4	MN_G	Binga

Closing and clicking on the first, and then on the second field respectively we can rename the two field names to labcode and labname to have a bit more meaningful information:



Field Properties: V1

Basic Extended Jumps Calculate Notes

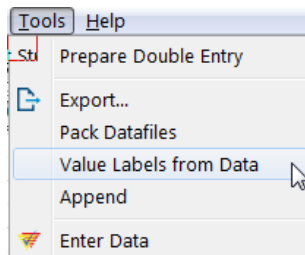
Name: labcode

Field Properties: V2

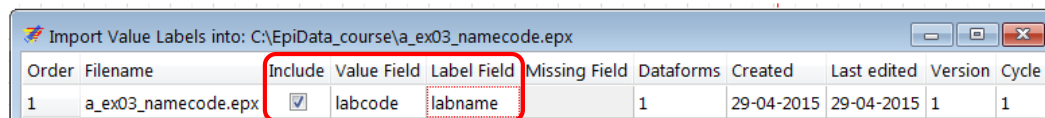
Basic Extended Jumps Calculate Notes

Name: labname

To make a label block set of the file go to Tools | Value Labels from Data:

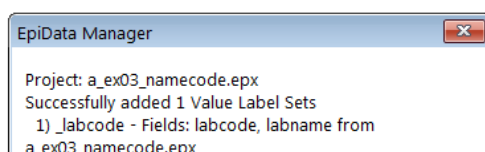


and pick the file itself, ensuring that Include is ticked and both Value Field and Label Field are correctly selected:



Order	Filename	Include	Value Field	Label Field	Missing Field	Dataforms	Created	Last edited	Version	Cycle
1	a_ex03_namecode.epx	<input checked="" type="checkbox"/>	labcode	labname		1	29-04-2015	29-04-2015	1	1

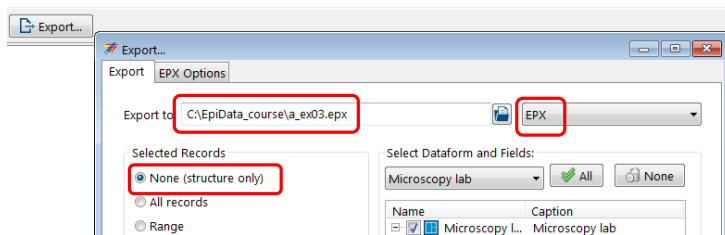
We get confirmation that the Value Label Set was created:



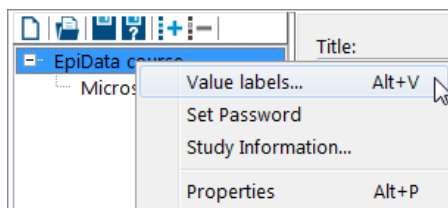
Save and close.

Updating the EpiData data form

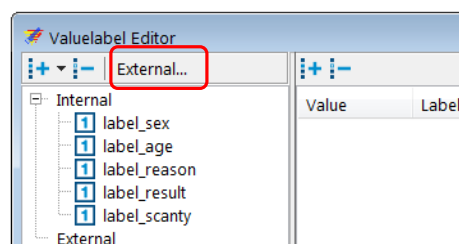
The first thing to do is to open the file a_ex02.epx from Exercise 2 and Export the structure only to an EPX file we name a_ex03.epx:



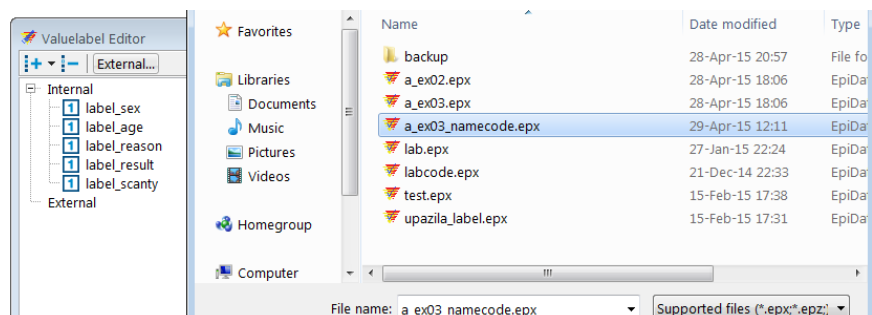
We then open the thus newly created `a_ex03.epx`. If we right-click on the project name EpiData Course we get a menu line Value labels (**ALT+V**):



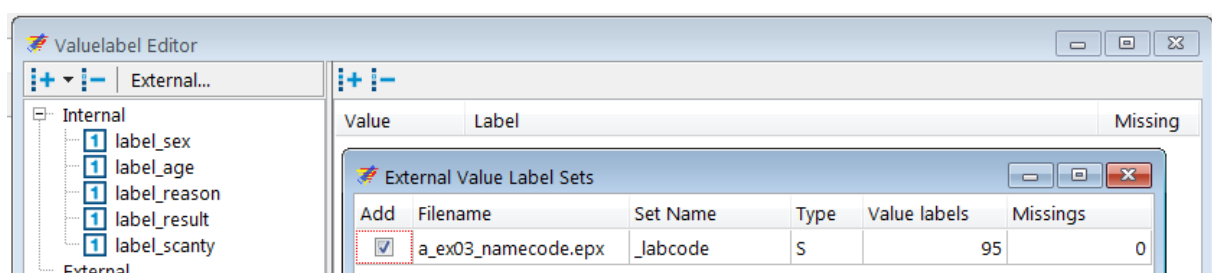
This gives us the Valuelabel Editor:



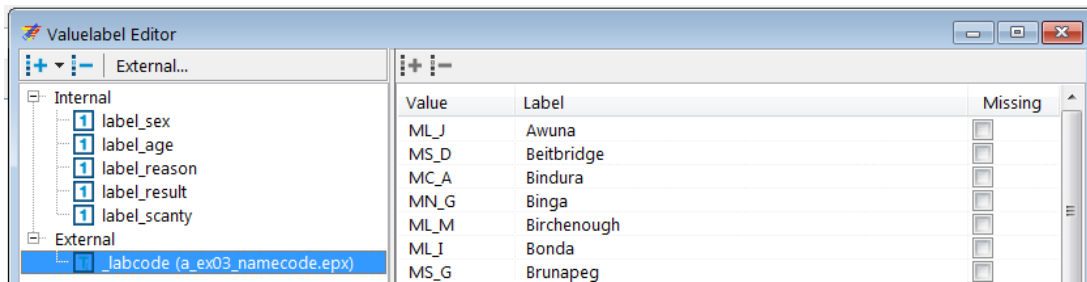
which allows the use of an external file as a label block. Using this approach (you will also look for an alternative to incorporate it), we can keep the label block externally. As long as it is in the same folder, the Manager / EntryClient will access it. Click and select the file:



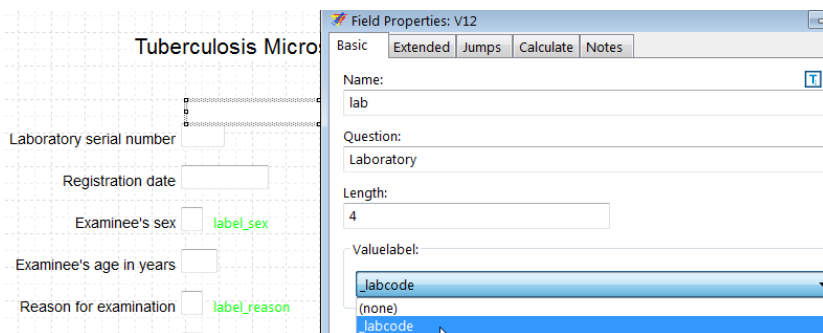
We are shown the name of the file, what the Set Name for the label block is (`_labcode`), that it is a string field with 95 Value labels:



We have now an external label block available and its content is visualized by clicking on it:



On the data entry form, we add a new string variable lab of length 4 with the field label Laboratory and add the now available Valuelabel:



Define the Extended as usual, save and test in EpiData EntryClient. Note particularly in EntryClient that for searching you start typing the name of a Field label and the cursor jumps to it.

Tasks:

- o Remove the external label block.*
- o Use the `a_ex03_namecode.epx` file and use it internally as a label block.*
- o Discuss advantages and disadvantages of External versus Internal label blocks.*

Solution to Exercise 3: Create a value-label pair from external data

Key Point(s):

- If possible, use an official list of, for instance, administrative units, and create a file format that can be imported into EpiData (such as *.csv)
- Create an EpiData *.epx file and make a label block of it
- You can use that label block internally or externally in your data collection form

Tasks:

- o *Remove the external label block.*
- o *Use the `a_ex03_namecode.epx` file and use it internally as a label block.*
- o *Discuss advantages and disadvantages of External versus Internal label blocks.*

Solution:

This is the modified data entry form with a fake record:

Tuberculosis Microscopy Laboratory

Laboratory MV_J Collin Saunders

Laboratory serial number 0001

Registration date 01-01-2001

Examinee's sex 1 Female

Examinee's age in years 45

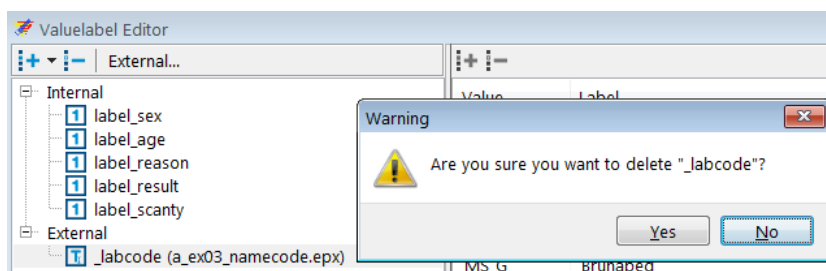
Reason for examination 1 Follow-up at 1 month

Result of specimen 1 0 Negative Result of specimen 1 scanty ☐

Result of specimen 2 6 Scanty, quantified Result of specimen 2 scanty 5 5 AFB per 100 OIF

Result of specimen 3 9 Result not recorded Result of specimen 3 scanty ☐

We access the Valuelabel Editor with **ALT+V**, click on it and press the Delete key:



and it's gone after accepting. To use the file internally, we close the Valuelabel Editor and use instead Tools | Value labels from data, select the file, and insure that all is ticked and selected correctly:

Import Value Labels into: C:\EpiData_course\a_ex03.epx				
Order	Filename	Include	Value Field	Label Field
1	a_ex03_namecode.epx	<input checked="" type="checkbox"/>	labcode	labname

Importantly, the deletion of the external also removed the link to the Valuelabel for the field:

Name:	<input type="text" value="lab"/>	<input type="button" value="T"/> Type: String
Question:	<input type="text" value="Laboratory"/>	
Length:	<input type="text" value="4"/>	
Valuelabel:	<input type="text" value="(none)"/> <input type="button" value="New"/>	

and this must then be set again. Conveniently, this internal label block can also be removed by accessing the Valuelabel Editor with ALT+V and proceeding analogously.

Advantages and disadvantages

The advantage of an internal label block is to have everything in a single file. Unless the label block is very large (how to define that?) it will thus mostly be advantages to do precisely that. If a label block is very large (say the ICD 10 codes), then it might be advantages to actually keep it externally as larger file names will slow down processing time. Furthermore, the current EpiData Analysis version cannot yet make the connection between the main file and an external label block.

Exercise 4: Create a composite identifier

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

- a. Recognize the efficiency of using date components instead of dates
- b. Creating a unique identifier from several variables
- c. Automatically capturing data entry time

Sometimes we have time information but it is incomplete, such as when a patient may remember the month but not the day of disease onset. In such a case a date field cannot be used sensibly because a date needs all three date components. It becomes thus useful to utilize instead three variables for day, month, and year respectively and then reserve it for the analysis to set rules what approximation might be used if one or more date components are missing. For data entry, this can also have the advantage that some components (like the year, or even the month) are set to Repeat if records are entered serially along a time axis and several sequential records have the identical year and perhaps also month.

A date component might also be used to construct a more complex identifier if the simple one does not suffice. In these exercises using the tuberculosis microscopy laboratory we defined the unique identifier as the laboratory serial number. However, the serial number starts with 1 every calendar year and is thus unique only for a single year. By combining the year plus the serial number, it becomes unique for the laboratory at any time, and adding the laboratory identifier, we can make a unique identifier for any examinee for any year, in any laboratory.

To learn about the workload of data entry, there is a possibility to instruct EpiData to record automatically the computer time at the time of opening a new record and to add another time stamp at the completion of the record.

We will introduce here these three concepts and integrate the computer-calculated fields into a special section.

Capturing time with date components instead of with dates

Export the structure of the `a_ex03.epx` file to a new `a_ex04.epx` file and open the new file. We then start with deleting the field `regdate`, dragging down the block of variables below it, and inserting the three new integer variables `regdd`, `regmm`, and `regyy` of lengths 2, 2, and 4 respectively, for Registration day, Registration month, and Registration year. We make each field Must Enter and provide a legal range and put the respective three values 99, 99, and 9999 for not recorded into two different Valuelabels. Remember from Exercise 2 that we need to untick the Show valuelabel text after ... in the Extended tab. We can make the fields for day and month set to Repeat as these two remain identical in most laboratories for quite a few sequential records.

Making an identifier composed of 3 existing variables

On the top we section off a part that will accommodate three calculated fields: the new identifier, the start time and the ending time of data entry for the record. Drag thus all fields down on the

canvass and insert a section above them that we call `Calculated fields`. The identifier `id`, labeled `Unique identifier` is a string field of length 14 as we will give it the form `YYYY-labcode-serno`, i.e. the 4-digit year, a hyphen, the 4-digit laboratory code, a hyphen, and the 4-digit laboratory serial number. It is placed inside the section and it has to be a `No Enter` field as we want to prevent a data entry person being able to ever touch a computer-calculated field. The top of our revised data entry form may thus look now as follows:

How do we go about making the value for this identifier? It can only be composed once all three required components have been entered, i.e. after entering the `Registration year`. We thus go to that field and go to the tab `Calculate`, where we choose the bottom option and fill by selection from the drop-down fields and inserting a hyphen in between, that's all there is to it:

A little more sophisticated is the approach to make `id` a unique identifier. Intuitively, we might think that we just make this field `id` the key field, but that is not how it's done: it is each of the components that is set to be the key and EpiData will then correctly interpret the resulting combined field required to be unique. It may appear a bit counter-intuitive, but that's how it's done. Remember where to `Define Key`? Right-click the data form name or go to the menu point `Dataform`:

Tasks:

- o Add two fields for *Starting time* and *Ending time* for data entry in the section *Calculated fields*.
- o Test the data entry form in *EpiData EntryClient*

Solution to Exercise 4: Create a composite identifier

Key Point(s):

- If one variable is insufficient to create a unique identifier, one can construct a composite identifier from more than one variable.
- Each of the contributing fields, but not the composite resulting variable is set to be Key.
- EpiData provides automatically calculated time fields that can be used to capture data entry time.

Tasks:

- o Add two fields for Starting time and Ending time for data entry in the section Calculated fields.
- o Test the data entry form in EpiData EntryClient

Solution:

A complete record may look as follows:

Tuberculosis Microscopy Laboratory

Calculated fields

Unique identifier 2002-MV_J-0034

Starting time 18:01:17 Ending time 18:02:11

Laboratory MV_J Collin Saunders

Laboratory serial number 0034

Registration day 12

Registration month 6

Registration year 2002

Examinee's sex 1 Female

Examinee's age in years 43

Reason for examination 0 Diagnosis

Result of specimen 1 0 Negative Result of specimen 1 scanty ☐

Result of specimen 2 6 Scanty, quantified Result of specimen 2 scanty 5 5 AFB per 100 OIF

Result of specimen 3 9 Result not recorded Result of specimen 3 scanty ☐

Exercise 5: Data entry and validation

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

- Know the three ways of reducing data entry errors
- Copy the structure of a EPX file
- Export data from EpiData files
- Validate duplicate data files

You have a line listing of 15 records on the page following the task description. These data should be entered in this exercise. But before you start working, a few considerations are in place.

Ensuring quality data entry

The motto for this course is:

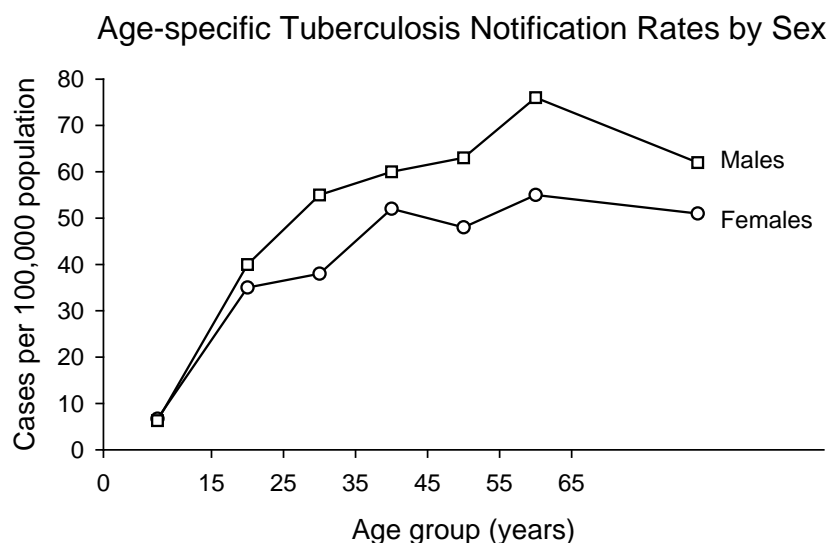
“You wish never to find yourself in a position to defend the quality of your data”

Michael B Gregg, formerly MMWR Editor, deceased

See also: Gregg M B. Field epidemiology (2nd ed). Oxford University Press. 2002: p 414

You might be challenged about the interpretation of your data, that is part of the scientific process, but your data should be of impeccable quality.

What do you think about the following graph?



It looks nice and we could talk about the differences between males and females and this and that. But we will keep it short: it is nonsense. The data underlying this graph have no basis, they were made up. Of course, if we were to present these data for real, it would be outright scientific fraud. Few people commit that (but it exists). *Nevertheless, often no assurance can be given that the computerized data are a true reflection of the original data source.* People may have in all honesty done “their best” and assume that they made no errors or so few that it really doesn’t matter. However, this is not good enough for science in general and public health and epidemiology in particular.

There are three ways how we reduce and ultimately eliminate data entry errors:

- o Defining well thought-through data entry controls in the EpiData Manager
- o Working together
- o Duplicate data entry and validation

Defining well thought-through data entry controls in the EpiData Manager

We have already a few inbuilt conditions that limit data entry errors by creating the `a_ex04.epx` file. For instance, a Must Enter field will prevent a data entry person to skip an actually recorded value, as one cannot continue without having entered a value for that field. For the field `sex`, we allowed only 1, 2, and 9 as legal values. It is thus not possible to enter “3” into this field. Combined with the pop-up menu during entry, no confusion can arise. The controls set in the EpiData Manager are an extremely powerful tool to control how data entry can be controlled through restrictions.

Working together

Entering data alone requires continuously shifting attention between the paper record and the computer screen. This will almost by necessity result in numerous errors, be it that a record is skipped or that it is forgotten what we just read. It should be routine that two persons work on data entry: one person reads aloud the Field value, the other repeats it aloud and enters the value.

Duplicate data entry and validation

Even with both of the above precautionary measures, data entry errors will still occur, and worse, to an unknown extent. ***The only way, and the only acceptable one, is to enter the data twice into two different files, and then to compare the two files for discordances.*** Any discordance uncovered will then be corrected the entry with the original paper record.

The rationale behind this process is: *the probability of committing the same error in the same field twice when data entry is done independently by two persons is very small.* Hence, if we list all the discrepancies by comparing the two databases and correct all of them, then we can be reassured that the remaining frequency of data entry errors is miniscule.

EpiData provides this powerful tool and we need a unique identifier to do this. We have made a provision that we have such an identifier (see previous exercises). Sometimes an identifier must be constructed from more than one variable as we have shown.

If a duplicate key is revealed (because there is a perhaps a problem with a component contributor), then a data entry note should be written, best as a text file that is kept open during data entry and amended as one proceeds. In this note, you must specify exactly with what identifier you have replaced the duplicate key, so that this note can be passed on to those who

enter the data the second time, enabling them to use the same alternative key when the necessarily stumble over the same problem.

Before you get to actually enter the data, you find here some assistance to make your data entry work more efficient.

Make duplicate EPX files

As we are entering the same data twice, we need two sets of the *.epx files, one for the first, the other for the second entry.

Task:

- o Download the solution of Exercise 4 and save the file as **a_ex05_a.epx** and **a_ex05_b.epx** files

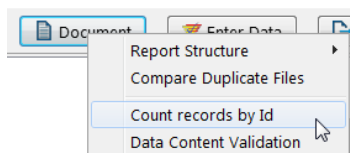
Double-entry

Enter the 15 records into the a_ex05_a.epx, then repeat data entry with the a_ex05_b.epx file.

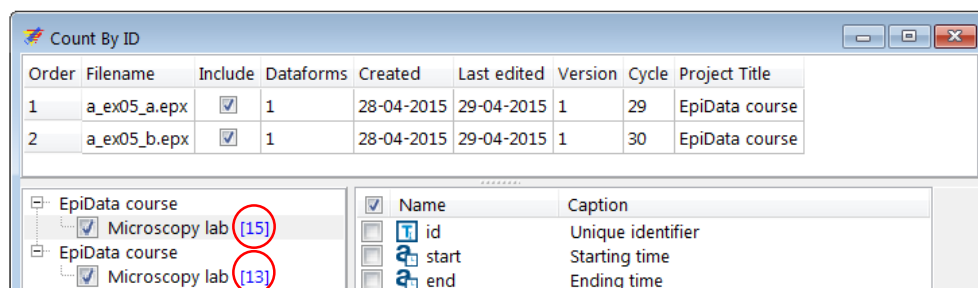
Data validation

After completing double-entry, the two data files are compared, a procedure termed “**data validation**”.

The first thing to verify is that we have the same number of records in both sets. If that is not the case, then this must be fixed first. Here for instance, we chose to count the records:

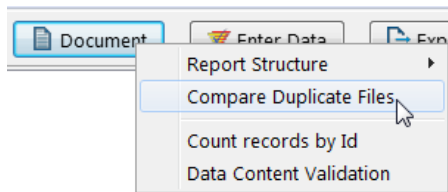


and found them to be unequal:

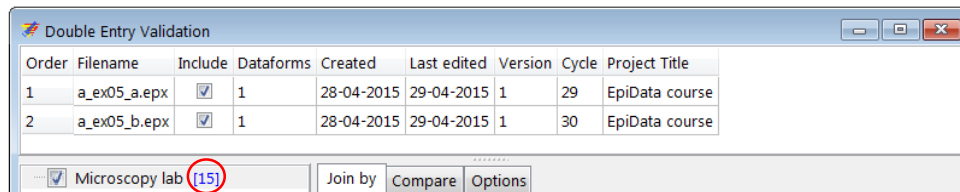


While we know from this small dataset that there actually are 15 records, and thus that the first set has the correct number and the second is missing 2, knowing the expected number of records is the exception in real life. It may also very well be that both sets have the same number of records but both are short of records because by chance each set is missing the same number of

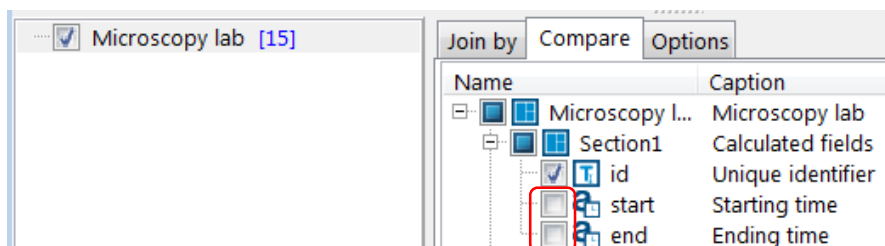
records. We thus have to check which records are missing and we do this by a validation, actually comparing the two sets:



The first of the two files is the “referent” and it is the number of records in this set that is displayed:



In the tab Compare, we untick the computer-provided time fields because they are likely to vary for virtually every record between the two files and would undesirably be listed as discordances:



In the report, we move to the Overview and find the confirmation that the duplicate file (the *_b.epx file) has 2 records missing:

```
-----
Result of Validation:
-----

Overview
-----
Test                                     Result
-----
Records missing in main file             0
Records missing in duplicate file        2
Non-unique records in main file          0
Non-unique records in duplicate file     0
Number of fields checked                  12
Common records                           13
Records with errors                       1
Field entries with errors                 1
Error percentage (#records)               7.69
Error percentage (#fields)                0.64
-----
```

Moving further down, we find the serial numbers missing from the second file and thus know which two records must be added first before a proper validation is possible:

```
Record no: 14
Key Fields:
lab = ML J
serno = 3310
regyy = 2003   Record not found
-----
Record no: 15
Key Fields:
lab = ML J
serno = 3311
regyy = 2003   Record not found
-----
```

The Validation report

We add these two records and then start again the validation process. This time we get the same number of records in the two files and find one discordance in serno 3302:

```
-----
Result of Validation:
-----

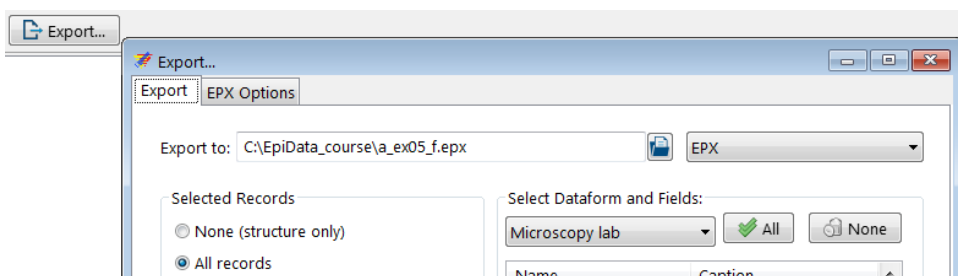
Overview
-----
Test                                     Result
-----
Records missing in main file             0
Records missing in duplicate file        0
Non-unique records in main file          0
Non-unique records in duplicate file     0
Number of fields checked                 12
Common records                          15
Records with errors                      1
Field entries with errors                1
Error percentage (#records)              6.67
Error percentage (#fields)              0.56
-----

Datasets comparison:
-----
Main Dataset:      Duplicate dataset:
-----
Record no: 5      Record no: 5
Key Fields:
  lab = ML_J
  serno = 3302
  regyy = 2003
Compared Fields:
  sex = 2          sex = 1
-----
```

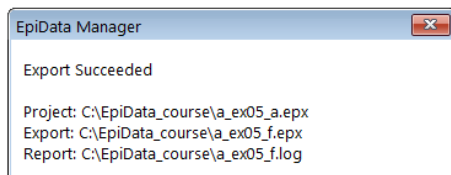
It is essential that this validation report is saved to ensure having a permanent record of the validation process. We propose to save it as a text file `a_ex05_validation.txt`.

Creating a final dataset

One might be tempted to make corrections of any errors that might be identified through discordances in either the `*_a.epx` or the `*_b.epx` file. Doing so would, however, break the “chain of evidence”: you could never repeat the validation process and get the same result, but data quality-assurance requires that the validation process is actually exactly reproducible. Therefore, the corrections must be made in a third file. To this end, we export the data from one of the source files to another EpiData file that we will call the `a_ex05_f.epx` file. To standardize as many things as possible, we always export the `a_ex05_a.epx` file to the `a_ex05_f.epx` file (even if in fact it is irrelevant whether we use the `a_ex05_a.epx` or the `a_ex05_b.epx` file, but consistency is good policy). We thus select from Export the `a_ex05_a.epx` file and define in the menu the name and type and All records:



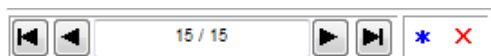
We get a report that export was successful:










Back from Manager to the EntryClient we open a_ex05_f .epx and search the record with serno 3302.

How to navigate through an * .epx file?

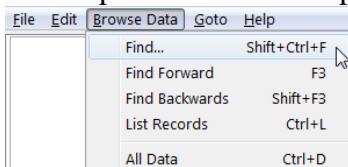
To navigate between the records of the * .epx file use the navigation panel on the left bottom end of the data entry screen which can be used to navigate through the records.



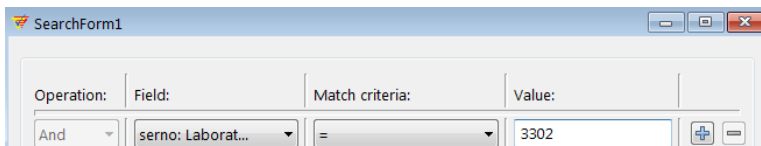
Shown vertically one by one:

-  Go to first record
-  Go to previous record
-  This is record 11 out of a total of 15 records
-  Go to next record
-  Go to last record
-  Insert a new record
-  Mark current record for deletion

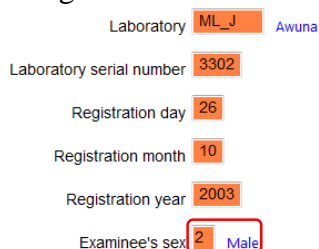
This is useful for a quick forth and back, mainly during data entry, but for the current task to find a specific record in a potentially large file, we use Browse Data | Find:



We enter our criterion:



and get thus to the record:



Comparing with the original paper record, we see that the true Examinee's sex should be female. Moving the cursor into the field and pressing **F9**, we can now pick the correcting value:

Registration month **10**

Registration year **2003**

Examinee's sex **2** Male

Examinee's age in years **38**

Value	Label
1	Female
2	Male
9	Not recorded

As this is the only discordance we save the revised record and exit. We now have a validated file with all discordances resolved.

How to delete a record?

Deleting a record consists of two steps – first, marking a record for deletion; second, permanently deleting it. This is just a safety feature in EpiData to ensure the deletion of record does not happen by chance.

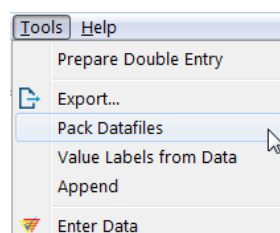
Steps in marking a record for deletion (Look at the screenshot below)

1. Open the *.epx file and go to the record you want to delete.
2. Click on the red 'cross' mark next to the navigation panel at the left bottom of the data entry screen. The word DEL appears at the side of the red 'cross' mark.
3. Click the arrow in the navigation panel to go to the next record. This will prompt you to save the record. Click 'Yes' and this successfully marks the record for deletion.
4. Note that the record is not yet permanently deleted from the database. If you realize that this record was not to be deleted, you can undo the action by clicking on the same button and saving the record. DEL will disappear now: the red "cross" is a toggle key:

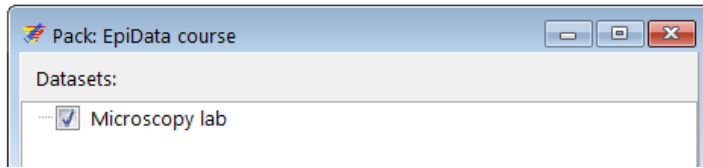


How to permanently delete a record? (Pack Data Files)

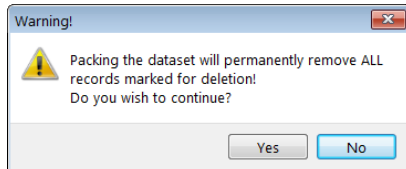
For data protection, it is not foreseen to permanently delete a record in the EpiData EntryClient. This must be done in the EpiData Manager. Close thus the file (if open) in EpiData EntryClient and go to the Manager to Tools | Pack Datafiles:



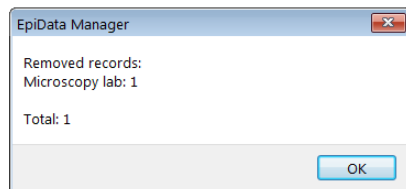
Choose the data file and tick the data form (it could be a relational data base with several forms and you got to choose which one):



Because this is going to be permanent, you receive a final Warning:



After accepting, you get confirmation that it is done:



No backup file is written, this is permanent.

Tasks:

- o Download the solution of Exercise 4 and save your a_ex04.epx file as a_ex05_a.epx and a_ex05_b.epx.*
- o Enter the 15 records using the a_ex05_a.epx file. After completing data entry, enter the same data again into the a_ex05_b.epx file.*
- o After you have completed the two files, proceed to validation as explained here.*
- o After ensuring that no record is missing in either file, export the a_ex05_a.epx file to a a_ex05_f.epx file, check out the discordances if any and correct them. This is your final dataset.*

On the next page you find the dataset with 15 records

Laboratory: Awuna

Tuberculosis laboratory register

Year: 2003

Lab Serial No.	Date specimen received	Name	Sex M/F	Age	Name of referring facility	Address - patient for diagnosis	Reason for examination*		Results of specimen			Only for SS+ for diagnosis: TB Number or BMU**	Remarks
							Diagnosis (tick)	Month of follow up	1	2	3		
3298	26 Oct	Mary	F	35	Bindura	Beijingstr. 6		5	neg	neg			
3299	26 Oct	John	M	20	Awuna	Tokyo Ave 5	√		neg	neg	neg		
3300	26 Oct	Petra	F	30	Birchenough	Bangkok Rd 108		5	neg	neg			
3301	26 Oct	Charles	M	24	Bindura	Hanoi Street 7a		2	neg	neg			
3302	26 Oct	Tiffany	F	38	Bindura	Hongkong Ave 8	√		neg	neg	neg		
3303	26 Oct	George	M	60	Bindura	Zurich Rd 923	√		neg	neg	neg		
3304	26 Oct	Luke	M	78	Awuna	Paris Street 18a	√		neg	neg	neg		
3304	26 Oct	Virginia	F	28	Birchenough	London Rd 24	√		neg	neg	neg		
3305	27 Oct	David	M	50	Awuna	Baltimore Str 1		6	neg	neg			
3306	27 Oct	Hans	M	50	Ganda Chivua	Bern Str 12	√		1+	1+	1+	Ganda Chivua No 342	
3307	27 Oct	Bill	M	68	Bindura	Berlin Ave 88	√		neg	neg	neg		
3308	27 Oct	Susan	F	29	Birchenough	Amsterdam Rd 3		5	neg	neg			
3309	27 Oct	Marc	M	36	Bindura	Vienna Str 76		2	neg	neg			
3310	27 Oct	Eve	F	15	Awuna	Rome Ave 4		5	neg	neg			
3311	27 Oct	Anthony	M	37	Birchenough	Antwerp Str 26c		6	neg	neg			

* Check the appropriate category from the Request for Sputum Examination

**TB register number

Solution to Exercise 5: Data entry and validation

Key Point(s):

- It should be routine that two persons work on data entry, and never one.
- The only and acceptable way to minimize data entry errors is to enter the data twice into two different files, and then compare the two files for discordances.
- Avoid using the mouse to move around fields during data entry, because the Check file cannot be applied to fields you skip by moving the mouse from one field to another.

Tasks:

- o Download the solution of Exercise 4 and save your a_ex04.epx file as a_ex05_a.epx and a_ex05_b.epx.*
- o Enter the 15 records using the a_ex05_a.epx file. After completing data entry, enter the same data again into the a_ex05_b.epx file.*
- o After you have completed the two files, proceed to validation as explained here.*
- o After ensuring that no record is missing in either file, export the a_ex05_a.epx file to a a_ex05_f.epx file, check out the discordances if any and correct them. This is your final dataset.*

Solution

Depending on the errors you made, you will get an output like the following:

```
=====
Report: Double Entry Validation Report.
Created: 29-04-2015 22:03:30
=====
```

```
-----
File 1: C:\EpiData_course\a_ex05_a.epx
File 2: C:\EpiData_course\a_ex05_b.epx
-----
```

```
File 1: C:\EpiData_course\a_ex05_a.epx
```

```
-----
Title      EpiData course
Created    28-04-2015 09:50:32
Last Edited 29-04-2015 20:33:53
Version    1
Cycle      29
-----
```

```
Backup on shutdown: yes
Encrypted data: no
```

```
Dataforms:
```

```
-----
Caption      Created      Structure Edited    Data Edited      Sections Fields
Records Deleted
```


Microscopy lab 28-04-2015 09:50:32 29-04-2015 20:33:53 29-04-2015 20:33:53 2 17
15 0

Caption Fields in key

Microscopy lab (serno:Laboratory serial number) + (regyy:Registration year) + (lab:Laboratory)

=====
DataForm: Microscopy lab
=====

Selections for validation:

Options:

Option Selected

Ignore deleted records No
Ignore missing records No
Add result to field No
Case sensitive text No

Key Fields:

lab serno regyy

Compared Fields:

id: Unique identifier
regdd: Registration day
regmm: Registration month
sex: Examinee's sex
age: Examinee's age in years
reason: Reason for examination
res1: Result of specimen 1
res1sc: Result of specimen 1 scanty
res2: Result of specimen 2
res2sc: Result of specimen 2 scanty
res3: Result of specimen 3
res3c: Result of specimen 3 scanty

Result of Validation:

Overview

Test Result

Records missing in main file 0
Records missing in duplicate file 0
Non-unique records in main file 0
Non-unique records in duplicate file 0
Number of fields checked 12
Common records 15
Records with errors 1
Field entries with errors 1
Error percentage (#records) 6.67
Error percentage (#fields) 0.56

Datasets comparison:

Main Dataset: Duplicate dataset:

Record no: 5 Record no: 5

Key Fields:

lab = ML_J
serno = 3302

```
regyy = 2003
Compared Fields:
sex = 2          sex = 1
-----
```

After making correction in the “F” file, your data should be correct, or are they not? While your final data file should be correct, there is still a slim chance that it has errors. How is this possible? If by chance the same error was entered in both files (which can happen particularly if the same person enters the data in both files), you will not be able to identify the error. For uniformity, you should overwrite your existing file with the `a_ex05_f.epx` file that is provided with the solution.

Exercise 6: Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file

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

- Understand how you import an EpiData Entry 3.1 REC / CHK file pair into EpiData Manager
- How to edit the new file and add the link to the metadata from the old CHK file to the new EPX file.

Importing an EpiData REC / CHK file containing data

In the required zip folder you find 4 data sets, a, b, c, and d, complete with all 16 files, i.e. for the “a” files a.qes, a.rec, a.chk, and a.eix, containing 75 records, and analogously for the “b”, “c”, and “d” sets. It contains microscopy laboratory data from four laboratories in Yangon (Myanmar), courtesy Dr Ti Ti. If you look at the A.QES file in EpiData Entry 3.1, you see the visual format:

```
id          Unique laboratory identifier  _____

serno       Laboratory serial number  ####
labcode      Laboratory code          _
regdate      Registration date <dd/mm/yyyy>
age          Examinee's age in years  ##
sex          Examinee's sex          #
reason       Examination reason       #
res1         Result of specimen 1    #.#
res2         Result of specimen 2    #.#
res3         Result of specimen 3    #.#
```

We note that the results were defined as floats with the following values (the definition of which we can best see by opening the A.CHK file (e.g. in a text editor):

```
LABEL label_result
0.0  Negative
0.1  "Scanty, 1 AFB per 100 fields"
0.2  "Scanty, 2 AFB per 100 fields"
0.3  "Scanty, 3 AFB per 100 fields"
0.4  "Scanty, 4 AFB per 100 fields"
0.5  "Scanty, 5 AFB per 100 fields"
0.6  "Scanty, 6 AFB per 100 fields"
0.7  "Scanty, 7 AFB per 100 fields"
0.8  "Scanty, 8 AFB per 100 fields"
0.9  "Scanty, 9 AFB per 100 fields"
1.0  "1+ positive"
2.0  "2+ positive"
3.0  "3+ positive"
4.0  "Positive, not quantified"
5.0  "Scanty, not quantified"
9.0  "No result recorded"
END
```

In EpiData Manager, go to File | Import File... and pick the a.rec to see it then in an editable box:

Order	Filename	Structure	Data	Dataforms	Created	Last edited	Version	Cycle	Project Title
1	a.rec	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1	N/A	30-10-2010		0	a.rec

You note that we encountered this box in an earlier exercise when we imported a value-label set. Note that we can import the structure only or both the structure and the data, as we plan to do here. Once confirmed with OK we get:

and see at the bottom left that there are indeed 75 records which we can browse in **Dataform | Browse Data** (shortcut **CTRL+D**) (first lines only shown):

Record No:	id	serno	labcode	regdate	age	sex	reason	res1	res2	res3
1	A-1001	1001	A	24-10-2003	35	1	5	0	0	9
2	A-1002	1002	A	24-10-2003	20	2	0	0	0	0
3	A-1003	1003	A	24-10-2003	30	1	5	0	0	9
4	A-1004	1004	A	24-10-2003	24	2	2	0	0	9

Note that the button Show Values / Labels is non-functional because the labels are not yet available, we thus see only the values at this point in time. Save the file when prompted as a_ex06a.epx.

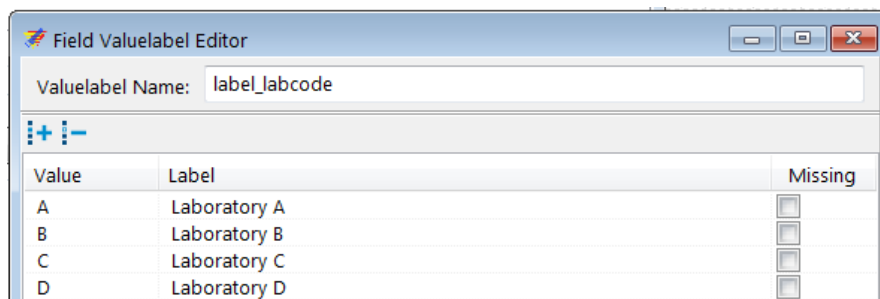
Make the metadata from the CHK file available to the EPX file

We work our way through the entry form from top to bottom.

The Unique laboratory identifier (field name `id`) is a calculated value from the Laboratory code and the Laboratory serial number. It is thus a field set to No Enter and calculated after both components that make it up are available.

The Laboratory serial number (field name `serno`) is Must Enter and we Add leading zeros.

The Laboratory code (field name `labcode`) is Must Enter. We noted above in browsing that the value was “A”. The field does also not have a Label Block (the Laboratory code was added only after data entry in the analysis module). As we will have also laboratories B, C, and D, we will add here a Label block with string values to later accommodate the three additional laboratories. Thus create a New Value Label as:

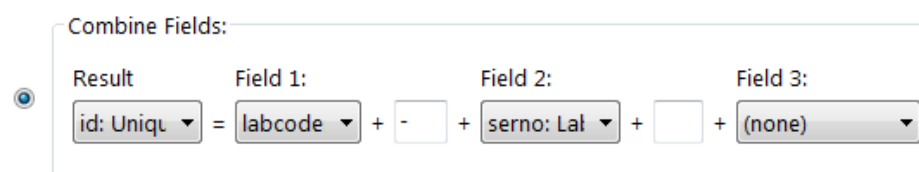


Field Valuelabel Editor

Valuelabel Name: `label_labcode`

Value	Label	Missing
A	Laboratory A	<input type="checkbox"/>
B	Laboratory B	<input type="checkbox"/>
C	Laboratory C	<input type="checkbox"/>
D	Laboratory D	<input type="checkbox"/>

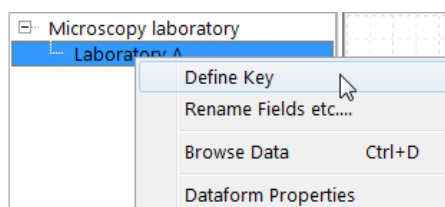
As we do with variables with label blocks, we tick Always show picklist during entry. As we have now all the necessary information we Combine Fields in the Calculate tab:



Combine Fields:

Result: `id: Uniqu` = Field 1: `labcode` + - + Field 2: `serno: Lal` + + Field 3: `(none)`

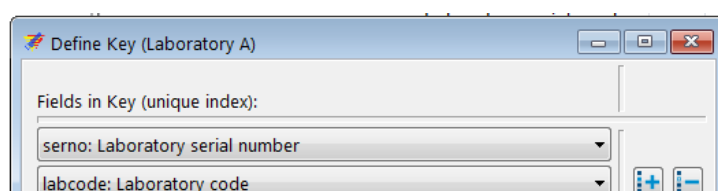
Let's also make both `labcode` and `serno` Key fields (right-click the data form [which we have here already properly relabeled]):



Microscopy laboratory

- Laboratory A
 - Define Key
 - Rename Fields etc....
 - Browse Data Ctrl+D
 - Dataform Properties

and make `serno` and `labcode` Key fields:



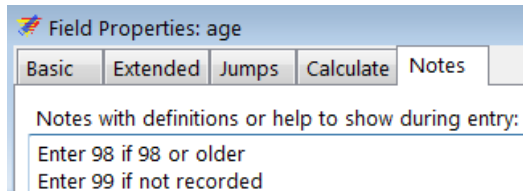
Define Key (Laboratory A)

Fields in Key (unique index):

- `serno: Laboratory serial number`
- `labcode: Laboratory code`

The Registration date (field name `regdate`) is Must Enter and as all laboratory results are from the year 2003 we might as well add a Range from 01/01/2003 to 31/12/2003.

The Examinee's age in years (field name `age`) is Must Enter and as a continuous variable of length 2 we may add Notes:



Field Properties: age

Basic Extended Jumps Calculate Notes

Notes with definitions or help to show during entry:

Enter 98 if 98 or older
Enter 99 if not recorded

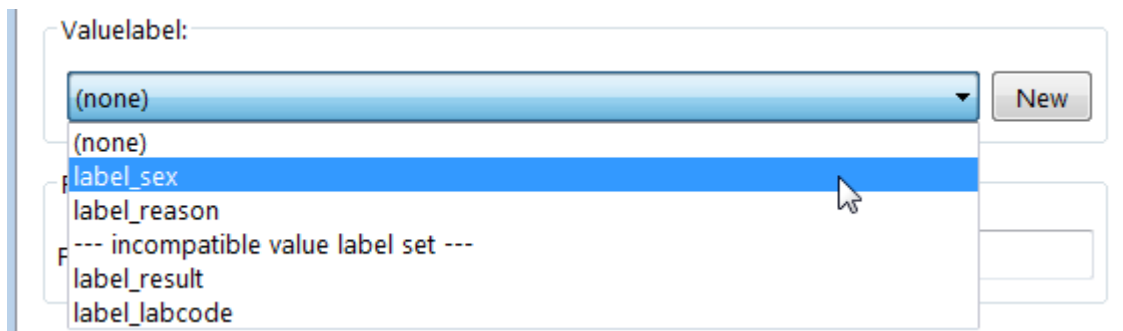
The Examinee's sex (field name `sex`) has no Valuelabel ticked:



Valuelabel:

(none) New

However, if you pick the drop-down list, you see that it actually exists (it is in the CHK file) and all we need to do is to pick it:

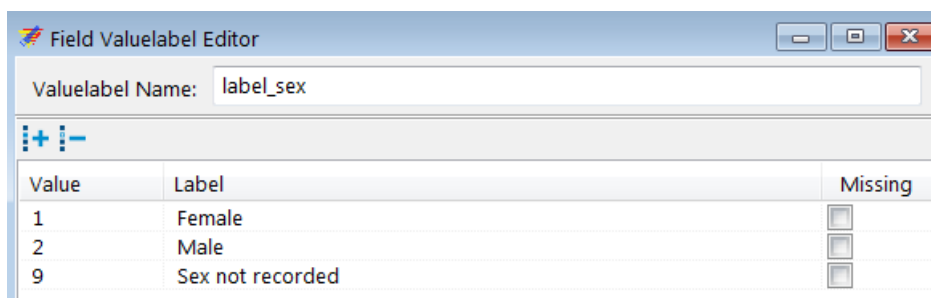


Valuelabel:

(none) New

(none)
label_sex
label_reason
--- incompatible value label set ---
label_result
label_labcode

and then we can view it with Edit that replaces New once picked:



Field Valuelabel Editor

Valuelabel Name: label_sex

Value	Label	Missing
1	Female	<input type="checkbox"/>
2	Male	<input type="checkbox"/>
9	Sex not recorded	<input type="checkbox"/>

The list of Value labels also shows that we have `label_reason` and `label_result` which we can use for the last four variables to complete updating the new EPX file with the metadata from the old EpiData Entry CHK file.

Finally, once the above is done, we may edit it nicely and check functionality in the EpiData EntryClient (without saving the record!!):

Tuberculosis Microscopy Laboratory

Laboratory A

Unique laboratory identifier

Laboratory serial number

Laboratory code Laboratory A

Registration date

Examinee's age in years

Examinee's sex Female

Examination reason Diagnosis

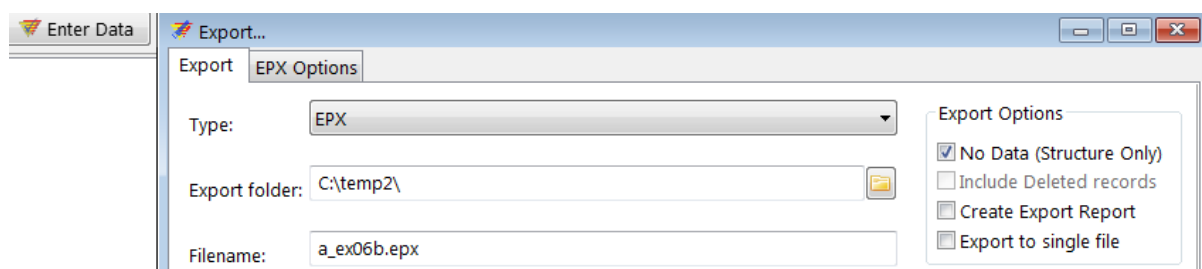
Result of specimen 1 1+ positive

Result of specimen 2 2+ positive

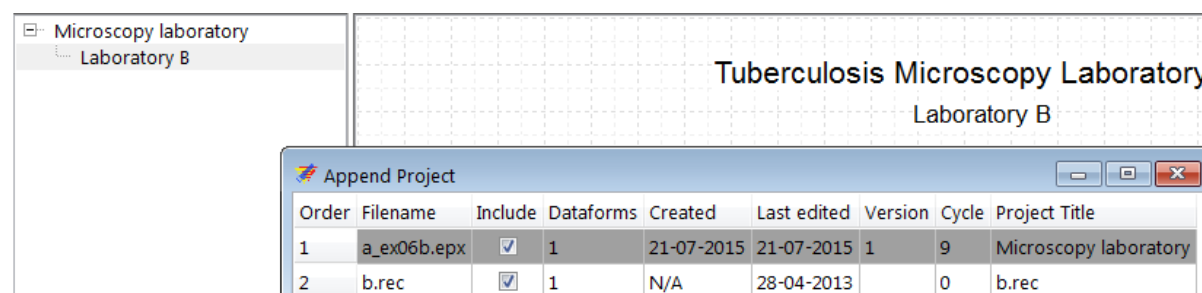
Result of specimen 3 No result recorded

Exporting the structure of an EPX file and importing REC file data into the new structure

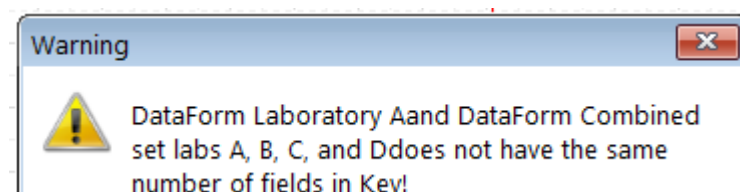
The data form structure of the now created a_ex06a.epx file can be exported without the data using Export and ticking the appropriate box. We export it to a new file a_ex06b.epx:



We then open this a_ex06b.epx file, edit it (changing the name of the data form and the header to become “Laboratory B”) and use Tools | Append and pick the b.rec file:

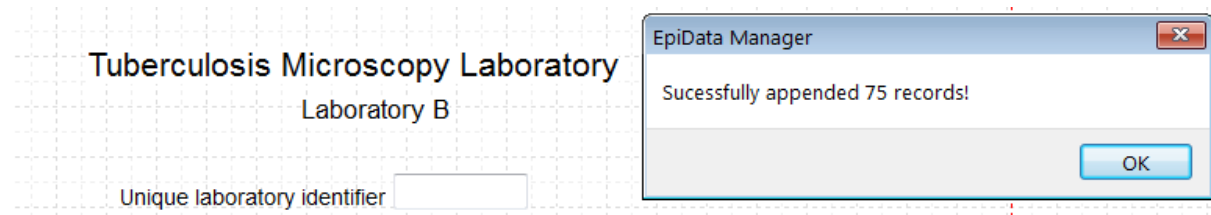


However, doing so, we get an error message:



The reason is that in the EpiData Entry REC file serno is the key, and in this EpiData Manager EPX file serno and labcode are the keys. The best approach is to remove the b.chk file

and to remove both keys in the EPX file and subsequently add them again after successful import which is prompt if these actions are done:



If we wanted to have all four laboratories in one single a_ex06abcd .epx file, we could have done that simply by adding all four REC files (after removing keys in the EPX file and removing the CHK files for the four REC files after successfully using their metadata). However, it cannot be done in one sweep, each must be added on its own.

Task:

- o Create the entire set of four EPX files, so that in the end we have a_ex06a.epx, a_ex06b.epx, a_ex06c.epx, and a_ex06d.epx*

Solution to Exercise 6: Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file

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

- Understand how you import an EpiData Entry 3.1 REC / CHK file pair into EpiData Manager
- How to edit the new file and add the link to the metadata from the old CHK file to the new EPX file.

Task:

- Create the entire set of four EPX files, so that in the end we have a_ex06a.epx, a_ex06b.epx, a_ex06c.epx, and a_ex06d.epx

Solution:

We have four files with an identical structure (the Headers differ, but these are just labels) with different records (but each file has 75 records):

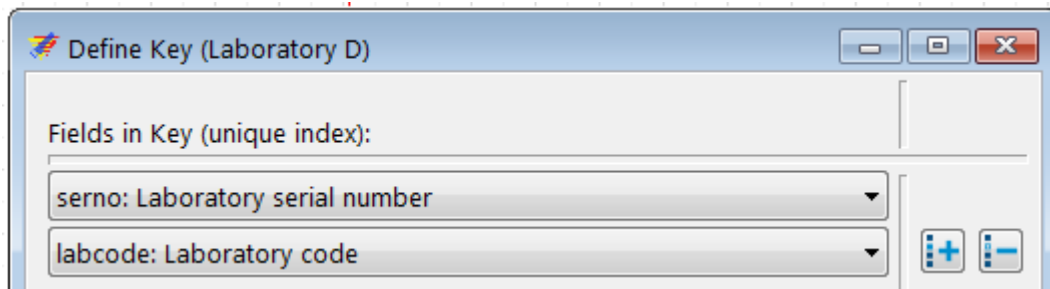
a_ex06a.epx
a_ex06b.epx
a_ex06c.epx
a_ex06d.epx

The final data form for a file looks like this:

The screenshot displays the EpiData Manager interface. On the left, a tree view shows the project structure with 'Microscopy laboratory' and 'Laboratory D'. The main area shows a data form titled 'Tuberculosis Microscopy Laboratory Laboratory D'. The form contains the following fields:

- Unique laboratory identifier
- Laboratory serial number
- Laboratory code (linked to label_labcode)
- Registration date
- Examinee's age in years
- Examinee's sex (linked to label_sex)
- Examination reason (linked to label_reason)
- Result of specimen 1 (linked to label_result)
- Result of specimen 2 (linked to label_result)
- Result of specimen 3 (linked to label_result)

You have ensured that the two keys are added again after import in each of the four files:



While we are not going to enter new records into any of these files and therefore this adding of the keys is not essential, it would be if new records were added to ensure that each resulting combined identifier is unique.

Exercise 7: A relational database

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

- a. Understand when a single data entry form and when a relational database is the preferred data collection instrument
- b. Create a relational database for a varying number of observations.

Not all laboratories keep their registers as The Union and WHO recommend for the Tuberculosis Laboratory Register, where 1 line corresponds to 1 examinee rather than to 1 examination. In fact, many laboratories enter the results for each examination on one line. If such an approach is chosen, we may find a register as follows:

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

This type of a register requires a different approach to data entry than we used before. Two important things need to be considered:

- 1) The same patient may appear again and again on sequential dates
- 2) Not every patient has the same number of visits

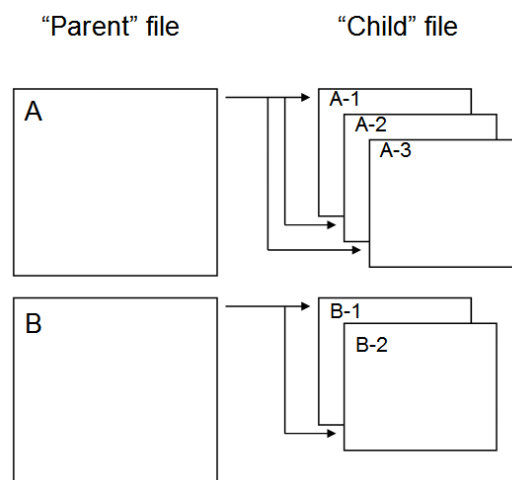
Some patient characteristics do not change over time such as, in this example, the identity of the patient, sex, and marital status (well, perhaps not during these short intervals). Others do change, such as the date of examination, blood sugar, the aspect of the sputum and the sputum smear examination result.

To capture such information in a single data entry form as was done in the previous exercises would be very inconvenient: 1) one would have to anticipate the maximum of allowable visits, and 2) if one patient has a single visit, one would still have to complete all fields with the codes for missing values up to the maximum allotted if we insist that all fields must usually be Must Enter fields.

Rule: *If an individual has a fixed number of observations for each variable, then a single EpiData form is the best solution. If an individual has a variable number of observations for each variable, the choice is a **relational data base**.*

Building a relational database requires deciding which information is static for an individual (during the observation period) and which information changes over repeated observations. We will illustrate a relational database with a very limited set of variables.

What is the structure of such a relational database? The figure below outlines a relational database with two levels.



At the Level 1 (the "Parent" file), we may have patients, denoted here with A and B. At Level 2 (the "Child" file), denoted here as A-1, A-2, and A-3, we may have different visits to a clinic where each time the date of the visit is recorded, and blood pressure blood sugar are measured. Each individual may have at least one, but up to an unlimited number of such visits, and the number of visits varies for each individual.

Note: *Because of the simplicity, it is always preferable to have a single data entry form. However, if the data available concern an individual with fixed information (e.g. age, sex, etc) on one hand, and variable information (e.g. serial examinations) and there are fixed sets of variables in serial examinations (e.g. repeated measures of blood sugar and blood pressure) and each examinee has a varying number of examinations, then it may become more efficient to use a relational database.*

In the following, we will call the records at Level 1 the parent records and those at Level 2 child records.

To build the relational database, we collect different information at each level and link the two levels, so that at some point during entering parent information will trigger the opening of the

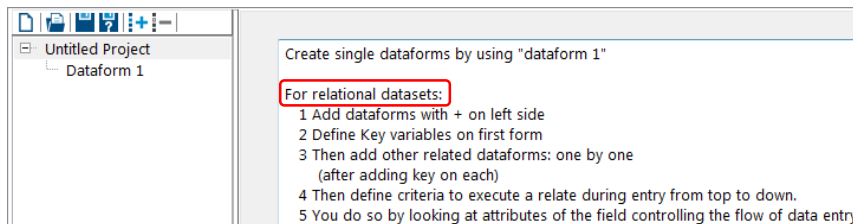
child form to enter one or more child records. One or more child records are added until one wishes to return back to the upper level of parent records.

Important components for the structuring of the relation between the two data forms are:

- The two forms have a common identifier field. This identifier might be called `idparent`.
- The child form must have its own unique identifier, which might be called `idchild`.

Illustrating it practically in EpiData Manager

We start a New Project. We note the brief information on the relational database in the opening window:



Change the name of the Untitled Project to Exercise 7 and Dataform 1 to `1_parent` and save the project as `a_ex07.epx`.

We add as first field the string variable `idparent`, the Unique parent identifier of length 2 as a Must Enter field. We add only two fields: `age`, an integer of length 2 for Patient's age in years and `sex` for Patient's sex, an integer of length 1. Both are Must Enter fields. For `age` as a continuous variable, we might write in Notes:

Enter 0 to 97 for exact age
Enter 98 if 98 or older
Enter 99 if not recorded

Finally, we define `idparent` as Key field. Next we make the `2_child` Dataform (rename it using **F2**) by clicking on the:



And we note that EpiData opens the new data form with the `idparent` identifier already written onto the canvass:



If you click on it, you see that in the Field Properties menu everything is grayed out and in its Extended tab we also see the grayed out setting to No Enter, all courtesy of EpiData Manager!

While we are in the `2_child` data form, we add the four variables `visit`, `syst`, `diast`, and `bs`, for Visit date, Systolic blood pressure, Diastolic blood pressure, and Plasma glucose in mMol respectively, a date, two integer and one float field, all Must Enter (and define some range, but assume no missing values). Preceding

Visit date, add idchild, the Unique child identifier, a No Enter string field of length 13, resulting from the combination of the values of idparent and visit.

This is all there is to it: the flow of the related forms is such that once the 1_parent is completed, the flow leads right over to the 2_child data form, where you can complete from one to many records before you move back up to a new 1_parent data form.

Other options

You can add more related forms. If there is more than one data form, you can choose under Properties of the 1_parent data form (right-click its name) in the After Record tab the order of flow which is to be followed after the parent record:



For a single related form as in our example, there is only one possibility, and this is pre-recorded as default.

You can also set conditions in a field of the 1_parent form, i.e. which value in the field should trigger access to the related field. As of now, there is still some bug that should be fixed shortly that prevents this extended functionality to work as intended.

Task:

- o Complete the related data set and check it for functionality in the EpiData EntryClient*

Solution to Exercise 7: A relational database

Key Point(s):

- A relational database is appropriate when a dataset contains fixed person information (such as age, sex, hair color, and the like and variable information like examination of weight and blood sugar over a variable number of examination visits.
- A unique identifier in the parent file serves as the key connecting the parent file with the child file

Task:

- o *Complete the related data set and check it for functionality in the EpiData EntryClient*

Solution:

A completed parent and child record set may look as this:

Relational database: Parent file

Unique parent identifier
Patient's age in years
Patient's sex Female

Relational database: Child file

Unique parent identifier
Unique child identifier
Visit date
Systolic blood pressure
Diastolic blood pressure
Plasma glucose in mMol

[Press F10 or click on the 1_parent form to return to it](#)