Efficient and effective management of big databases in Stata

Giovanni Marin[1,2]

[1]CERIS-CNR, Milano, Italy
[2]SEEDS Sustainability Environmental Economics and Dynamics Studies, Italy

2014 Italian Stata Users Group meeting
Milano, November 2014

Outline of the presentation

Storage

- Memory limits in Stata
- Variable formats

Importing

- insheet, infix, import excel

Combining

- Relational databases 1.0
- Joining and merging databases

Reshaping

- Long and wide formats
- Smart reshaping

Tips

Slides are complemented with practical exercises in `exercise_databases.do`

Memory limits in Stata

- **Stata** uses the **RAM** to store data and other information to be used
- Up to **Stata 11** the user was required to **decide the amount of memory** to be allocated ⇒ starting **from Stata 12** memory is allocated **dynamically** (the command `set memory` is now obsolete)
- **Memory limits** in Stata
  - Details can be found by typing `help memory`

Variable formats

- Each **variable** is **stored** in its **own format** ⇒ `help format`
- **Numeric** formats for numbers are substantially more **efficient** in terms of memory usage ⇒ `destring`
- Potential issue with **strings** ⇒ even if only one (or few) observation is a **long string** (e.g. 200 characters), **all other** observations are stored as 200 characters information ⇒ beware with long string!

Importing data stored in specific formats

- I here describe some possible commands to **import data** stored in formats different from `.dta`
- The description is **not comprehensive** ⇒ `File → Import →...`

`insheet`

- To import data stored with a **separator** (tab, comma, semicolon, space, etc)
- **Many formats** are allowed (.txt, .csv, .raw, .tsv, etc)

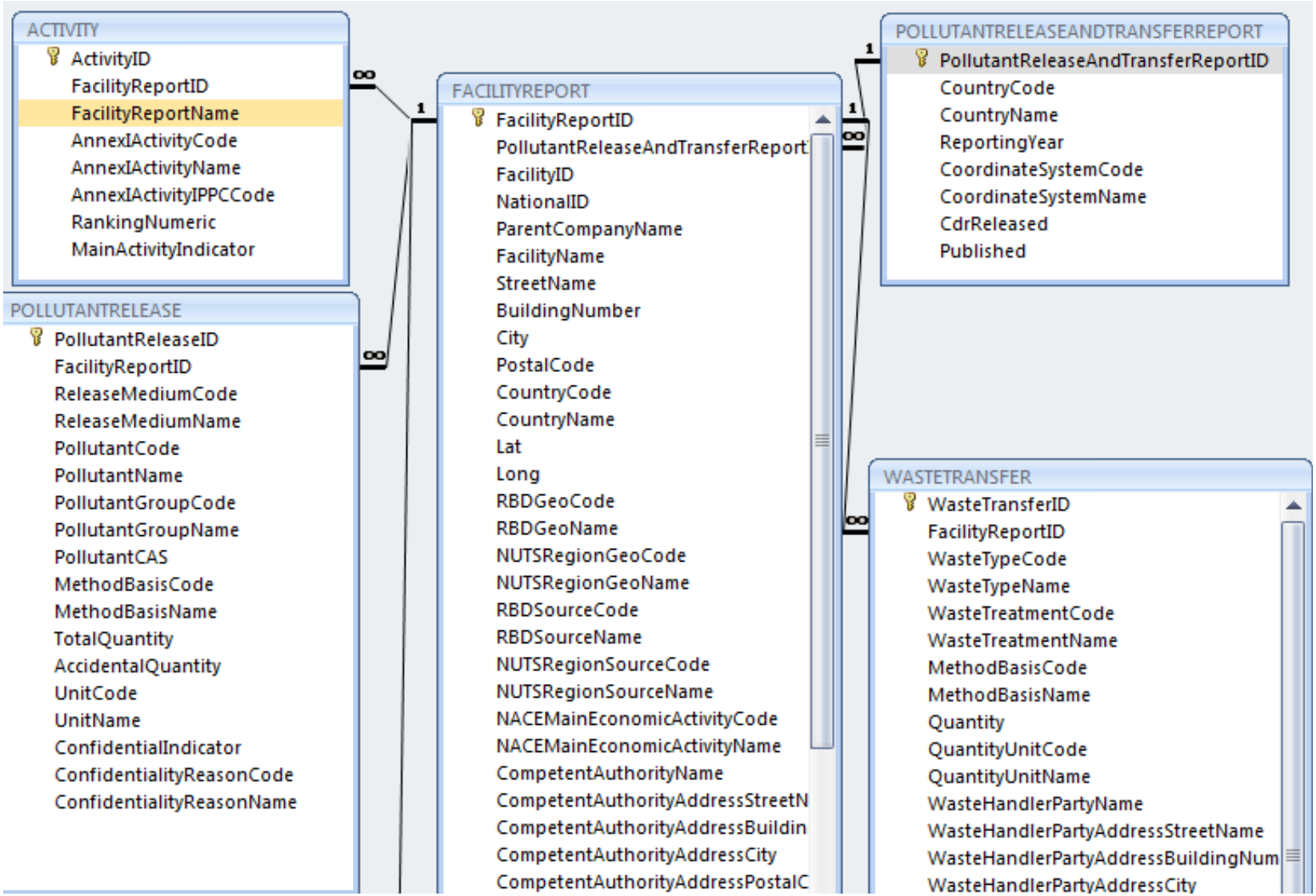
`infix`

- To import data stored in **fixed format** (with dictionary or with manual specification)

`import excel`

- Starting from **Stata 12** it is possible to import data from **MS Excel** files (both .xls and .xlsx)
- It is possible (and sometimes needed) to indicate the **cell range** and the **sheet**

What is a relational database



The role of identifiers

- **Tables** are **joined** by means of **identifiers**
- For example, if the dataset is a **panel** in **long format**, the **unique identifier** of each observation will be the **combination** of **firm id** (e.g. the registry code) and **year** (**1:1** identifiers)
- However, it could be the case that two tables should be joined by attributing **many observations for each identifier** (**m:1**) $\Rightarrow$ for example time-invariant information on the firm for a panel of firms
- Finally, it could be useful to have **m:m joins** $\Rightarrow$ to generate **all possible pairwise matches** between multiple identifiers
- In the following **examples** I will explore the various **cases**

The command `merge`

- `merge` is the basic command to **join two** Stata **datasets**
- It accommodates **1:1**, **1:m** and **observation-by-observation** join
- The variable(s) that should be used as **identifiers** need to have the **same variable name** and need to be in the **same** 'broad' **format** (e.g. either string or numeric) in both datasets
- Beware that if you have variables (excluding the identifiers) with the **same name** in **both** datasets, the **default** option **keeps** the information recorded in the '**master**' (in memory) dataset (also missing information!)
- By **default**, `merge` **keeps all observations** from both the master and the using dataset
- **m:m** merge is **problematic** (`joinby` is better)

The command `joinby`

- `joinby` is very **similar** to `merge` but:
  - It accommodates **m:m** merge
  - The **default** option is to **keep** just those observations that are **joined**

The command append

- append simply **adds** observation at the **bottom** of the dataset in memory
- Variables in the master and using dataset should have the **same 'broad' format** (e.g. either string or numeric)

Examples of long and wide formats

Table : Long format

id	t	var
1	1990	33
1	1991	11
2	1990	55
2	1991	77
3	1990	99
3	1991	91

Table : Wide format

id	var1990	var1991
1	33	11
2	55	77
3	99	91

The use of `reshape wide` and `reshape long`

- The command `reshape wide` transforms the **long** format **into** the **wide** format
- The command `reshape long` transforms the **wide** format **into** the **long** format
- Some **tips**
  - If the 't' variable is a **string**, remember to use the **option** `string`
  - When transforming the **wide** format **into** the **long** format, remember that all variables with the **same prefix** should be of the **same format** and that all variables **not** included in `varlist` should be **constant** within 'i'
- If your dataset has a **wide** format but many **missing** values, it could be useful to **reshape** it **into** a **long** format and **drop missing** observations
- You can **always return** to the **original** setting by using `fillin` and `reshape wide`

Some additional useful tips

- `compress` to **optimize** variables format
- **Encode** strings or leave unused strings aside (but ready to be re-joined if needed)
- Use the **cycles** `foreach` or `forvalues` to do repeated commands
- **Compress** and **extract** zipped datasets directly from Stata (`zipfile` and `unzipfile`)
- `fillin` to rectangularize a dataset
- Use `preserve` and `restore` to make **temporary changes** to datasets

THANK YOU FOR YOUR ATTENTION

For questions, doubts, comments, please contact me at

g.marin@ceris.cnr.it