Redatam7

1 Red7 Create *(Redatam7)*

Database Creation

Class material

REDATAM© is a software developed by the Latin American and Caribbean Demographic Center (CELADE), Population Division of the Economic Commission for Latin America and the Caribbean (ECLAC), United Nations.
www.eclac.cl/celade

Table of Contents

I. Introduction

Brief summary of data processing

Today's cellular telephone is one thousand times faster and more powerful than the computers of the 1970s. In those days, computers were enormous machines that weighed tons, with very little internal memory, which normally carried out processes according to instructions from punch cards. They stored large volumes of data on magnetic tape, which, because of its design, was read linearly.

Generally, population censuses contain millions of records and variables (microdata); they were managed and stored in these enormous machines, and they required a programmer to produce the tabulations needed for subsequent analysis. Because of the cost and time to read tapes with census microdata, it was almost mandatory to create, print and publish a series of documents with the official tabulations at the country level and for large administrative divisions.

This limited census analysis to defined or limited topics, and it reduced the possibility of exploring new analysis lines. Given the difficulties in programming and producing special tabulations, and the fact that only National Statistical Offices (NSO) have census microdata, it was nearly impossible for researchers to obtain additional specific tabulations. There was virtual no feedback for data and analysis.

In this technological context, and thanks to the economic backing of IDRC[1] of Canada, Arthur Conning, head of Information and Data Processing of CELADE[2] visited different national statistics and planning offices and other institutions of Latin America and the Caribbean, in pursuit of a project for using bibliographic descriptions to identify and locate existing official and specific census tabulations and publications.

The inspiration for this idea was the success of DOCPAL[3] which created and used a bibliographic database of summaries and keywords in a mainframe computer. Surely this was the correct path to follow; Conning visited the main NSOs of the region searching for collaboration and support.

Serendipity: Fortuitous discovery

Conducted in several countries in the region, the findings of this feasibility study were conclusive: The proposed plan DID NOT solve the real census information management

[1] IDRC The International Development Research Centre of Canada

[2] CELADE Latin American Demographic Centre -- the Population Division of the Economic Commission for Latin America and the Caribbean (ECLAC)

[3] DOCPAL Population Documentation System for Latin America and the Caribbean, which objective was to create a bibliographic database with summaries and key words to help in the analysis of surveys in mainframe computers at that time.

needs of the region. NSO interests pointed in another direction, toward: having the capacity to generate specific census tabulations which were not considered in the official findings published for each census, and having census information available for disaggregated divisions outside of the country's large administrative areas, and even for territories that do not necessarily fit into any official administrative arrangements.

The conclusion was clear: the original project had no future, but there was the unintentional and serendipitous discovery of a much more interesting and significant challenge for the region at that time.

REDATAM: a fast and friendly solution

In the mid 1980s, CELADE, with the support and funding of many international organizations, initiated the development of a more comprehensive tool. The REDATAM[4] project was born in 1985, aiming for the creation of a computer solution with the same name which main objective was to help in demographic data processing and analysis.

Based on a microcomputer (and no longer on mainframes), this solution allowed access to a combination of databases organized in hierarchical structure. REDATAM organizes and stores all the data so that the user can obtain quickly and friendly any tabulation or other basic statistics up to the smaller geographic area defined in the database. As seen in Figure 1 the motif of the latest version of REDATAM is “fast & friendly”.

Figure 1. Logo of the latest version of REDATAM

During the 25 years of the REDATAM project, giant computational leaps have been produced (DOS, Windows -16, 32 and 64 bits-), in view of which CEALDE has carried out great efforts to provide the countries with an adequate and actualized tool. Today we are presenting a new generation of REDATAM, denominated *Redatam7*. The next table shows the different versions of the software since its origin.

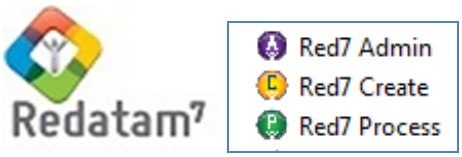
Table 1. REDATAM development (1985 – 2014)

REDATAM version	Generation	Year of Development	Operating System, environment, platform	Census round
REDATAM DOS	1st generation	1985	DOS	1980

[4] REDATAM (**RE**trieval of **DAT**a for Small **A**reas of **M**icrocomputer)

REDATAM 3.1	1st generation	1986	DOS	
REDATAM Plus	2nd generation	1991	DOS	1990
winREDATAM+	3rd generation	1997	Windows	
REDATAM +G4	4th generation	2001	Windows, Webserver	2000
REDATAM +SP	4th generation	2004	Windows, Webserver	
REDATAM 7	5th generation	2012	Windows, Webserver, 64 bits, multilingual code	2010

Redatam7: more support, results and customization



The development of REDATAM7 had three main objectives:

i. Increase the speed of data processing
ii. Facilitate the programming of indicators and
iii. Improve the user's experience

For which the software is based on programming languages, such as C++, Delphi and JavaScript. Currently Redatam7 is available only for Windows, the Linux, Macintosh and mobile operating systems versions are under construction and testing.

Unlike previous versions, Redatam7 is divided into three modules: *Create Process and Admin.* The first module generates a database and a REDATAM dictionary from an original input data file. The second module process and analyze these data, generating user-defined tabulations. And the third module allows you to administrate REDATAM databases, including functions that are seldom used.

Redatam7 incorporates new resources in comparison with the *Redatam+SP* version of 2004. The user network in the countries of the region and in other parts of the world has played an especially important role in this process, expressing us their difficulties and expectations with respect the use of the different generations of REDATAM. We have made improvements mainly in the following:

Resource	Benefits
✓ **XML language as**	Allowing synchronizing the documentation tasks, training and

standard	programming, as well as the connection with other computational tools, and so decreasing the gap between the development and the knowledge of the users.
✓ **New compiler**	Design and programming of a new compiler: departing from a definition of a grammar or syntax that adapts the language to the new requirements, achieving improved detection and display of errors in the use of the REDATAM language.
✓ **Unicode format supported**	Fundamental to generate dissemination applications in other languages used in some countries: Quechua, Creole, Guaraní, as well as other regions: Arabic, Chinese, etc., apart from the languages that already exist in REDATAM (english, spanish, portuguese, french and bahasa from Indonesia.
✓ **Customized tabulations**	Offers access to each of the elements involved in the presentation of tabulations, allowing the definition of rows and columns and other parameters in the table. The outputs are displayed in XML for export to other applications
✓ **Unlimited dimensions in tables**	At present the maximum number of variables to be included in tabulations is five, plus an area-break. With the new version you can obtain tabulation with an unlimited number of dimensions.
✓ **The Project is the new administrator**	In this version all the administration of databases, dictionaries, programs, maps, selections, table styles, etc is done by a project.

II. Basic concepts and definitions

The best way to work with REDATAM is to understand how its data is organized. Next we will explain some concepts and definitions to illustrate the REDATAM structure.

Entities

The variables describe **entities**, which are a set of logical objects that are hierarchically organized in the database. An entity can be a set of provinces, or a set of cities, or households, or persons. The database structure defines the hierarchical relations between entities, that is, mother-son relationships.

The picture shows a simple database structure with four entities organized hierarchically: COUNTRY, PROVINCE, DISTRICT and HOUSEHOLD

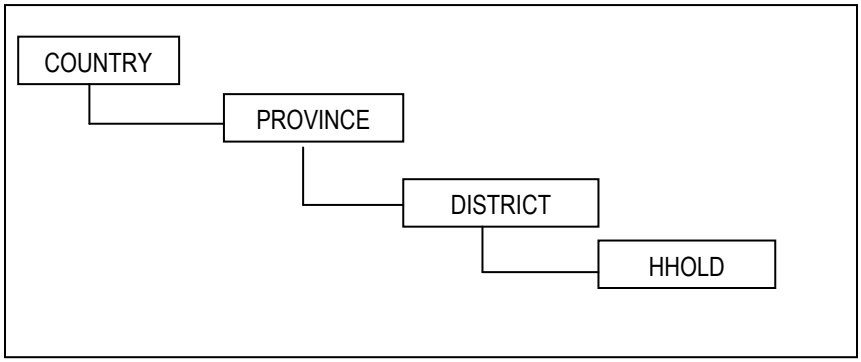


Figure 2. A simple hierarchical structure with four entities

Upper entities

Entities may be classified according to their position in the hierarchy. Higher entities are those that are over a certain entity. Taking again Figure 2 as a reference, we can say that the DISTRICT entity is above HHOLD. The next higher entity can also be called **mother entity**. By definition, each entity has only one mother entity, and therefore there is only one way back to the root entity.

Lower entities

Lower entities are those that are below a certain entity. Taking again Figure 2 as a reference, we can say that the DISTRICT entity is bellow PROVINCE. The next lower entity can also be called **child entity**. By definition, an entity may have a zero or many child entities.

Elements

The individual members of an entity are called entity elements. For example, the counties belonging to the "**county**" entity are the entity elements (or cases, instances or records) of this entity. An entity can have a few cases, for example, the regions of a country for the "**region**" entity, or it can have millions of elements, like the persons elements for the "**person**" entity in this same database.

For didactic purposes, when we refer to a REDATAM database structure we use the expression element for the cases. In the case of a physical file, we will use the term record.

The elements and entities of a databases always respect the hierarchical structure. This implies that each element of a mother entity is subdivided into the respective elements of its child entities. Returning to Figure 2, we can deduce that each element (province) of the PROVINCE entity is subdivided into elements (districts) of the DISTRICT entity.

Selectable or identifiable elements

When elements of an entity have names and/or codes that distinguish themselves individually, we say they are **identifiable**. When generating a database in REDATAM format, we can choose those entities containing names or codes for their elements in order to be identifiable; therefore the entity can be defined as selectable.

Although by definition, any entity can be selectable, in practice, it is customary to assign this property to geographic entities due to their statistical relevance. For policymakers, there is a greater interest in processing using administrative boundaries the same used as upper entities in a database. By checking the DISTRICT entity as selectable, it is possible to obtain indicators at the district level of a country[5].

In REDATAM, elements of selectable geographic entities can also be mapped graphically using the **geographic unique identifier code**.

[5] Add to that the fact that the information at smaller entities such as household or person -in many countries is protected by the principle of statistical confidentiality that ensures the inviolability of the privacy of individuals and legal entities (companies), which may not be subject to disclosure of data that identify statistically.

Geographic Code

The geographic code allows that all elements (including not selectable) to be aggregated and tabulated for each geography of the database and therefore, located and placed in a map. Based on this characteristic, the geographic code should be present for all geographic areas used in the hierarchical structure at the database generation process.

The geographic code comprises a specific alphanumeric sequence for each of the elements of an entity. Each part of the code reported belonging to a particular entity, which is why it is also known as compund code. An element from 07 district of the province 02 Country 0 (by definition), for example, may receive the geographic code 00207.

Non selectable/identifiable entity

If the individual elements of an entity do not have identification codes (visible in the dictionary by a small flag connected with the identifier), the entity is said to be non identifiable or **non selectable.** This type of entity cannot be used in a hierarchical selection. It cannot be used either as an output entity in an Arealist.

By definition, all the elements of a database are selectable, that is, can be individually analyzed. However, for security reasons (specifically to maintain the statistical nondisclosure or data confidentiality principle of the census databases), or for any subject matter needs, you can define any entity, and its lower entities as being **non selectable**, when creating the *REDATAM* database, or later when using the database Administration Tools.

There is a conflict of interests between security and data usability. For example, if the **Block** entity is defined as **non selectable**, it will not be possible to obtain results at the block level starting from the households and persons that live there. You can only get aggregated information of those blocks at a higher level such as district, count or region.

You can generate results for *individual* elements of selectable entities, but you **cannot have results for individual elements of non selectable entities**, for example for a specific household in the New Miranda database. Evidently, it is possible to generate tabulations of households and persons for a specific area, provided that the households and personas are not identified.

Branches and levels

A branch is the path from the root entity to a particular entity. By definition, an entity (and therefore its elements) belongs to a single branch. If you compare the structure of a multidisciplinary database with a tree, the branches in the structure are exactly the same as the ones in the tree, and the entities correspond to the tree leaves.

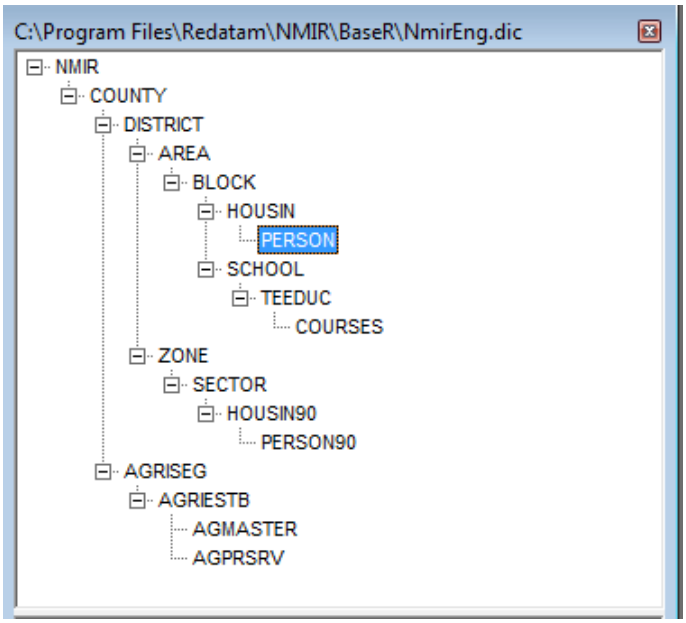


Figure 3. Several branches database structure

The **level** is the degree of depth of an entity within its industry. By construction, the "root" entity has zero level, (s) institution (s) immediately below have level 1 and so on.

Variables

The individual items of information for the entity elements are stored in **variables**. A variable is a property (characteristic, attribute) of each individual, that is, is a common property of all the individuals of that population: age, civil status, rainfall, etc. The property can be of qualitative type (attribute) or quantitative (variable). Consequently, each entity has its own value set for those variables. Thus, the ***Person*** entity could have, for example, a variable to store the ***sex*** of their individuals, and one of its elements could have the sex value equal ***female***.

Any entity can have variables in a database, for example, ***Province*** could have variables, like the yearly average rainfall, or the percentage of not married female in a certain age.

Table 2 shows the relationship between record of an entity and variables.

	ENTITY		
	Variable 1	**Variable 2**	**Variable 3**
Record 1 →			
Record 2 →			
Record 3 →			
Record 4 →			

Table 2. Relation between entity records and variables.

Data dictionary

All the information about the entities and variables is stored in a file called Database Dictionary (.dicx). This is the metadata, which means "information about information". The dictionary is the interface between the user and the data itself because the data can't be seen directly.

It allows the visualization of the data structure and the variables related to each entity. The dictionary contains a list of all the variables of a database for each of the hierarchical levels (entity), as well as the properties of each variable (values or categories, ranges, labels, types, not applicable and missing values, etc.). For example, the variable SEX at the person level presents two categories coded as 1 and 2, where 1 = male and 2 = female with a range of 1-2.

REDATAM employs the basic notion of freeing files and dictionaries. A dictionary is a conceptual data model. Its physical manifestation is the data file. It relies, for all the actual data administration, in the operating system functions and rules. The system follows the instructions of the data dictionary to recognize the file and reports, as anomalies, unrecognized formats.

REDATAM offers functions for reading and converting data dictionaries and the data itself written in CSPro, CSV, DBF and SPSS formats to REDATAM. This process reads the "meta data" from its original form and organizes the data according to the new structure given in the framework of the new REDATAM dictionary and database.

The dictionary is also saved in ASCII format with the extension wxp in order to import it from older version of REDATAM.

III. File structure of a REDATAM database

A REDATAM database is formed by three types of files:

1. Data files
2. Data dictionary file
3. Pointer or index files

REDATAM data files (.rbf)

REDATAM stores each variable (sex, age, relationship, etc.) of an entity (person) in a single binary file, with a "value" for each record in the entity. Then, each variable file (.rbf) looks like a vector of data but store in binary format so it can't be seen directly. Those files are known as transposed files, to distinguish them from the rectangular data structure, where all the values of one record are stored in the same row, and one single file has all the information for all the elements. These files have the .rbf extension (stands for "REDATAM binary files").

In the transposed structure used in REDATAM, there is a file for sex, another for household type, etc. The advantage of this organization is the fast processing and the secure of the data.

1. Pointer files (Indexes .ptr)

These files are the ones responsible for the connection between the entity elements and the lower entities. Each entity has an index file, with values that "point" from the elements in the higher entity (mother entity) to the elements of the entity itself. Therefore, the software extract the information exactly from the elements selected without having to go through the entire database.

4. Dictionary (.dicx)

All the information about the entities and variables is stored in a file called Database Dictionary (.dicx). This is the metadata, which means "information about information". The dictionary is the interface between the user and the data itself because the data can't be seen directly. It allows the visualization of the data structure and the variables related to each entity. The dictionary contains a list of all the variables of a database for each of the hierarchical levels (entity), as well as the properties of each variable (values or categories, ranges, labels, types, not applicable and missing values, etc.). For example, the variable SEX at the person level presents two categories coded as 1 and 2, where 1 = male and 2 = female with a range of 1-2.

The dictionary is also saved in ASCII format with the extension wxp in order to import it from older version of REDATAM.

IV. Basic principles of a REDATAM database

Logic structure of REDATAM

Internal number

Each one of the entities (and variables) is stored in a separate rbf file and is assigned a unique internal number when the database is created. By definition, the first entity ("root" entity, with the database name), is number 0. Those numbers can be seen (but not changed) in the Data Dictionary (see Properties screen).

Indexes/pointers

These files, also known as "**pointers**", are responsible for the physical connection between the entity elements and its inferior entities. Each entity has an index file, with the elements that "point" from the superior entity to the entity itself.
They have a .ptr extension (from "pointer"), and their names are formed by the database name as a prefix, added to the entity internal number (four digits, with leading zeros).
All the entities in the database (even the "root" entity) have an associated .ptr file. To understand how these files work, let us take as an example the database shown below.

Each one of these levels, such as "province", is defined as an entity. Each entity has one or more elements. The "province" entity can have, for example, two elements, the provinces called East Province and West Province. Following the hierarchy, below the "province" entity we find the "district" entity. Each province element is subdivided into district elements, which are subdivided into household elements:

If the entity elements have names and codes, the entity might be selectable, since their individual elements can be physically identified, and selected in REDATAM to be used later in a geographical selection. This is mainly the case for geographical entities like province, county, district, etc., allowing that their elements to be connected to a map.

In this example, below the "district" entity there is the "household" entity, made of individual households, each one of them belonging to a district. Entities such as household and person can have millions of elements.

Generally, those elements are non-identifiable in a REDATAM database, that is, we cannot select to process a specific household or any isolated individual. These entities are called non-selectable. However, a set of persons and households in an area can be

selected through its location inside one or more identified elements of higher level entities. This is done by means of the hierarchical selection process of R+SP.

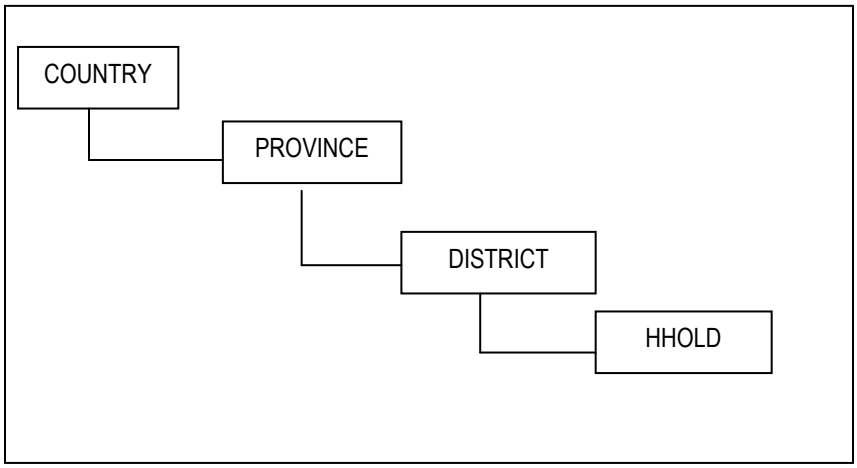


Figure 4. Hierarchical structure

This base has four entities, and therefore, it has four .ptr files. To simplify it, let us suppose that the entities have sequential internal numbers starting from zero (COUNTRY is entity 0, PROVINCE is 1, DISTRICT is 2, etc.). The HHOLD entity "belongs to" (or depends on) the DISTRICT entity, which "belongs" to PROVINCE, who "belongs" to COUNTRY, who, by definition, "belongs" to the system. This "dependency" concept, in a political division, for example, can be understood as: province 3 "of" the country, district 14 "of" province 2, household 358 "of" the district 21, etc.

Each entity can have any number of elements (cases), that is, x districts, y provinces, etc. For example, suppose we have the following number of cases for the entities:

ENTITY		CASES
COUNTRY (RROT)	1	
PROVINCE		5
DISTRICT		23
HHOLD		4587

NOTE: By definition, the "root" entity has ALWAYS only one element, it is like being the country. Also assume the following distributions, ie, the first province has three districts, the second five districts, etc., and the first district has 300 homes, 250 second homes, etc.

PROVINCE	DISTRICTS
1	3
2	5
3	4
4	6
5	5

Figure 5. Distribution of Districts by each Province

DISTRICT	HOUSEHOLDS

1	300
2	250
3	130
…..	
23	90

Figure 6. Distribution of households by each District

Logical contents of a pointer file

The logical concept of an index (pointer) file .PTR is a vector of counters, each record with the number of cases of the inferior entity that "belong" to the higher entity. In other words, each index file has as many elements as the elements of the entity "pointing at", and each of those elements contains the number of cases of the entity "being pointed".

For example, the logical contents of the index file corresponding to DISTRICT, that is, PAIS0002.PTR, is exactly the right column of Figure 4, that is, the file has five logical records (one for each province), with values 3, 5, 4, 6 and 5. The PAIS0003.PTR (for the HOUSEHOLD entity) corresponds to Figure 5, with 23 logical records, containing respectively 300, 250, 130, etc.

You can say that a pointer file, although it takes the number of the entity to whom it points at, the file works as a data file belonging to the superior entity, because it has the same number of elements of this entity.

Physical contents of a pointer file

In fact, the physical contents of an index file is a little different from the logical contents, it assumes its hole as holding pointers to the elements of the lower entity. Instead of the storing the number of elements, each record contains a sequence number (starting at zero) of the first element of the lower entity. Besides that, there is always an extra element at the end of the file, containing the total number of cases of the lower entity. Referring to Figure 5, the physical content of pais0002ptr file for the DISTRICT entity would be as in Figure 7, and the entity HOUSING would be like in Figure 8

RECORD	CONTENT
1	0
2	3
3	8
4	12
5	18
6	23

Figure 7. File PAIS0002.PTR

RECORD	CONTENT
1	0
2	300
3	550
4	680
…	…
23	4497
24	4587

Figure 8. File PAIS0003.PTR

There is always one additional record at the end of the list, to store the number of cases of the last case.

To have access to the elements of the lower entity, the system uses two records, the **record$_n$** for the start of the elements, and the **record$_{n+1}$** to calculate, by subtraction the number of cases. For example, the districts of the province 3 start at district 7 (in the record it is stored 8, but we have to subtract 1 because the lists are zero based), and there are 4 districts (12 – 8).

The figure 9 shows an example of this relationship in a graphic form, where you can see that each entity has variables (.rbf files), among them there is a special one containing the codes for its elements (if the entity is selectable). The entity can also have a variable containing the name associated with the code. To navigate through the entities hierarchical structure the system uses the indexes (.ptr files), establishing the connection between a superior element with their inferior elements, which can be seen by the arrows in the picture.

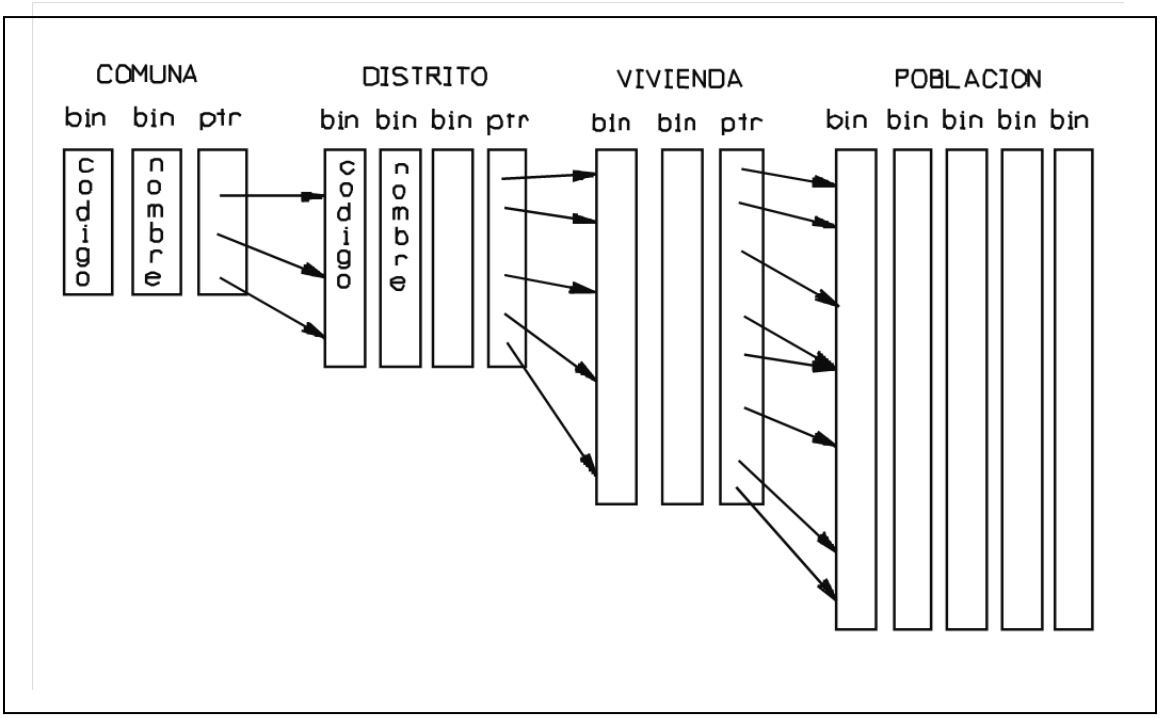


Figure 9. Relationship between pointers files

The index files are created at generation time. Each record (file element) has a 4-byte length and it is **NOT** in ASCII format. The format is proprietary of REDATAM and therefore it cannot be read by external programs.

Data

REDATAM stores each variable (sex, age, relationship, etc.) of an entity (person) in a single binary file, with a "value" for each record in the entity. Then, each variable file looks like a vector of data but store in binary format so it can't be seen directly. Those files are known as transposed files, to distinguish them from the rectangular data structure, where all the values of one record are stored in the same row, and one single file has all the information for all the elements.

In the transposed structure used in REDATAM, there is a file for sex, another for household type, etc. The advantage of this organization is the fast processing and the secure of the data. Each file will have as many values as the number of records in the entity where the variable belongs.

Going back to the example described in the Index files, a variable at the **PROVINCE** level will have 5 elements, while a variable at the **HOUSEHOLD** level will have 4,587 elements. These files have the .rbf extension (stands for "REDATAM binary files").

One of the advantages of this type of files is that, as each record, that is, each value has the same size, so the system uses a simple compression technique to minimize size of

the files. The decision to compress (or not) those files is at the user's hands, when defining the range and type of the variable in the properties window in the Base Scheme (wipX).

The compression algorithm consists of transforming the variable values into a binary number. The number of necessary "bits" for each variable depends on the maximum value the variable can take (also defined by the user when defining the range and type of the variable). A variable like sex, for example, having the values 1 and 2 (for male and female), is compressed into 2 bits (using 2 bits we can store a number lesser or equal 3). The age variable, whose maximum value is 99, needs 7 bits (7 bits an store a number up to 127), etc.

These binary files **cannot** be read outside REDATAM, since they are not in an ASCII format. Therefore, the values of the database can only be seen in an output table when processing a REDATAM database.

Creating a REDATAM Database

V. Preparing your input data

V.1 Files involved in the Database Generation

THE DATABASE SCHEME (.WIPX)

In order to create a REDATAM database, it is necessary to define a database scheme or framework (a wipx file) in the REDATAM Create Module, where the hierarchical structure and variables relationships are defined. This information is known as the Base Scheme and comprises the structure of the entities and information about the records and variables to be used from the source data to generate the *Red7* database.

In the generation process, this Base Scheme is the counterpart of the database dictionary. That is, it works like a dictionary file, containing all the information to build a REDATAM database but it doesn't have yet constructed the pointer or the variables files (ptr, rbf) to be considered a "true" dictionary, which will happen when the database is generated. This Base Scheme is used together with the metadata of the source database (the dictionary of *CSPro*, *DBF*, or *SPSS* files) to establish which variables will be generated within which entities.

INPUT DATA DEFINITION FILE (Dictionary)

This file is the dictionary of the source database, in Red7 only the following formats are read: *CSPro* format uses the *.ddf* dictionary file, *SPSS* format uses the *.sav* data and dictionary file and *xBase* format uses the *.dbf* data and dictionary file. These dictionaries store the properties or metadata of the source database. The information contained in the dictionary could be very basic: name, type and length of each variable as it is the case of the *xBase* format or it could be very lengthy: record type, name, type, starting position, (the field's physical information), the range, label, categories, not applicable and missing values, decimal places (the logical information about the fields), etc., as it is the case of the CSPro format. It also stores the type of the input file (simple or multiple records).

RELATIONSHIP BETWEEN .wipX AND THE INPUT DATA DEFINITION FILE

The input dictionary file contains the definition of the fields that are found in the input data set, and some information about its record types and the way the records are organized in the file. This definition is used to "populate" the future REDATAM database (feed the fields to be the variables according to the Base Scheme). The .wipX file, on the

other hand, is the file that stores the Base Scheme (entities and variables organization for the “to-be” REDATAM database) that will be use to migrate the input dataset into a REDATAM database creating at the same time its dictionary .dicx.

When you drag & drop a variable from the input dataset properties screen to the Base Scheme variables screen, the system moves the characteristics of the variable as it is in the source dictionary to the Base Scheme to be used when generation the REDATAM final database.

After executing the database generation process, a report is displayed on the screen and the dictionary (.dicX) is automatically generated together with the data (.rbf) and pointer (.ptr) files. This REDATAM dictionary is a working version of the Base Scheme (.wipX), so is connected to new database files and there is no connections whatsoever with the source data neither with the source dictionary.

V.2 Input file preparation: initial steps

A *REDATAM* database can have a very simple structure, with only one information branch, or as complex as you can imagine, with multiple branches in the hierarchy. It can be very small (a few Megabytes), or very huge, including census data, with hundreds or thousands of Megabytes. No matter the case you are handling, before actually starting the generation process, you need to consider the following steps:

(1) Define and review the source database (input file)
(2) Care to be taken with the input file
(3) Check acceptable formats for the input file definition
(4) Define the REDATAM structure and variables in a Base Scheme
(5) Proceed to create the REDATAM database based in the Base Scheme

INPUT FILE FORMAT

In order to generate a *REDATAM* database your source database should be in one single file and in one of these formats ***SPSS, CSPro or xBase***. The records in the source file can be stored in a simple or multiple record type file. If your input file is stored in a different format, such as the ones handled by commercial software like ACCESS, LOTUS, EXCEL, etc., you have to convert it to a valid format before starting the database generation.

SINGLE FORMAT

The input file has a single record type, that is, all its records have the same organization of variables and are related with a single entity. In this case, *Red7* assumes that those records belong to a single entity, for example "person". You can have variables from several entities in a single format, provided the records refer to the lowest entity. For example, if the single file represents persons, but it has also household variables, these variables must be repeated for every person record corresponding to the household.

Identification-variables						Data			
Province									
|	County								
|	|	District							
|	|	|	Block	Nhh	Nper	light	water	sex	age
|	|	|	|						
01	01	001	358	01	P1	1	1	2	23
01	01	001	358	01	P2	1	1	1	33
01	01	001	358	01	P3	1	1	2	12
01	01	001	358	02	P1	2	1	1	56
01	01	001	358	02	P2	2	1	2	52
01	01	001	358	02	P3	2	1	2	16

Figure 7. Single record type file

Other variables that need to appear in a single record format file are the ones that identify the geographic location of the person, the identification codes. For example, if you are loading the PERSON entity from a single record format file, each record must have, besides the variables related with the persons themselves, the variables that identify the higher entities like PROVINCE, DISTRICT, COUNTY and BLOCK

MULTIPLE FORMAT

If you are dealing with a multiple record format file, in the case of CSPro databases, you will find two or more record types in the input file. For example, having one record type for the household data, and another record type for the person data, allows to store values in the same column which belongs to different records. This file must have a specific column at the beginning of the row to identify the record type, for example, "1" for the household and "2" for the persons, as seen in the table below.

Rectype	Identification-variables				Data
	Province				
	|	County			
	|	|	District		
	|	|	|	Block	
	|	|	|	|	
1	01	01	001	358	Household
2	01	01	001	358	Person 1
2	01	01	001	358	Person 2
2	01	01	001	358	Person 3
1	01	01	001	358	Another household
2	01	01	001	358	Person 1

Figure 8. Multiple record type file

In this case, each record type (type 1 for the household and type 2 for the persons) will be referred to one entity in the REDATAM database, and the data from these rows will be loaded separately into the group of each entity. Each record type must have only the fields (variables) belonging to that specific entity, plus the geographic identification codes.

Those identification codes for the higher geographic entities are needed to build the relationships between the hierarchical levels in the REDATAM database, that is, the index files (.ptr). This is the only way the system has to "know" how to link data from one lower entity to the higher ones.

For the entities having no identification, like the HOUSEHOLD, the system will "know" when to create an element by detecting its record type in the input file. Using the same example from a typical census file, each type "1" record will create a household element, whereas each type "2" record following record type "1" will create person elements belonging to the previous household. Households with no persons will not have any type "2" records following its type "1" record. If your input file has households with no persons it is recommended to work with multiple format records, since in the single format the system still would create one person element even the variables might have not applicable values.

It is important to note that the household record type has to come BEFORE the corresponding person record type. That is, the input CSPro file should be ordered or indexed according to record type (plus geographic levels) in ascending order, having the household first and the persons within the household under.

The multiple record format is sometimes called a "hierarchical" file because there is a certain "hierarchy" between records type "1" for households and "2" for person. However, this is not a true hierarchical file because the identification codes (generally the geographical identification) is duplicated for every record. A true hierarchical file (REDATAM) would have one record type for each level in the hierarchy (entity), as shown in the following table.

Each entity would have a specific record type to store its data. The "geographical" information would not be repeated for every record in the file, but it would be stored only at its specific record. For example, the province information (like its code, name and other information at that level) would be stored only at type "1" records. This file type is not found regularly in statistical processes, although it is very useful as far as space saving.

Rectype	Data
1	Province (code and other information)
2	County
3	District
4	Block
5	Household
6	Person 1
6	Person 2
...	

5 Another household
6 Person 1
…

THINGS TO CONTROL IN THE INPUT FILE

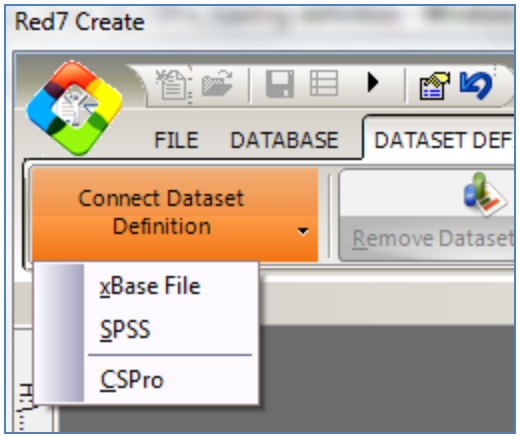
Before going to the generation process, it is necessary to evaluate the contents of the input file and check its consistency (verify that each household contain the same record type at the beginning of the row, sort the geographic codes and look for repetition of household numbers, missing head of the households, etc). We suggest to get a frequency distribution for each variable in order to be able to define all the needed information for the loading process (variables properties, identification codes, sorting of the data file, indexing, etc).

.

Check the data consistency:

a) Check if there are no duplicated records, that is, two records with the same codes and location. For example, check if there are not households declared twice.
b) Check to see if the variables defined as an entity identifier only have numbers, letters are not supported. For example, change the codes like 02001A to a number.
c) Check if the identifier code within an entity is unique for this entity. For example, that inside a household there are no two persons numbered 01, or within a block there are no two EA with the same code.
d) Decide which geographical variables will be used as codes in each entity of the REDATAM hierarchical structure, and define which levels will have the area names as well (for example, county, district, etc.).
e) Sort the input file according to the sequence defined by this REDATAM hierarchical structure.

ORIGINAL FILE FORMATS ACCEPTED BY REDATAM 7



The data in the source files are connected with the Base Scheme through its dataset definition file (dictionary), which can have one of the following acceptable formats:

CSPro The file definition was created by the CSPro software, and has the .dcf extension.

SPSS The file definition was created by the SPSS software, and has the .sav extension.

XBase File In this case the definition is the loading file itself (.dbf), there is no need to have another file definition.

Earlier versions of REDATAM accepted datasets in some additional formats:

IMPS4 The file definition (dictionary) was created by the IMPS4.1 software, and has the .ddw extension.

ISA The file definition was created by the ISSA software, and has the .map extension.

Chillan It is the *REDATAM* format (ASCII file) used to define the input data files that don't come with their own dictionaries, and has the .ddf extension.

Newer versions of SPSS can read excel databases as well.

VI. Generating the REDATAM Database

VI.1 BRIEF DESCRIPTION OF THE GENERATION

What does Database Generation mean?

In REDATAM language, database generation is simply a process of changing file formats, that is migrating data from the input file to a REDATAM database. To be successful, it is necessary to execute a series of steps and verified conditions.

First you have to check the consistency of the input file as described earlier. Then, as the generation is a reformatting process, you must define both (1) the input file comprised of the dictionary and data file and (2) the base scheme for the REDATAM database which requires the design of the base structure, its entities (hierarchical structure) and its variables location.

After that, you need to "map" the fields from one definition to the other, that is, which (and where) the input variables will be represented in the output REDATAM database. This consists in moving the input variables into each entity window of the base scheme. There will also be a need to check variable properties (labels, categories, ranges, not applicable and missing values) and define the geographic codes for the selectable entities.

Finally, you can run the generation process (five steps) and examining the generation report to evaluate the new database.

Let's now start the whole process of database generation from scratch.

VI.2 Preparing the original file

Results drawn from statistical models not only reflect their underlying assumptions but especially they considerably depend on the quality of the input data. This is a functioning principle of any statistical modeling system, and REDATAM is not an exception. Undoubtedly, REDATAM is one of the most powerful data- and user-friendly software system that can process and organize statistical information at the lower geographical levels of disaggregation, however to do so this will need the input data file to be as clean and consistent as it is required by the REDATAM hierarchical structure. It is worth to recall that the preparation of the input data file could be seen as the most determining of the whole process of starting and using REDATAM.

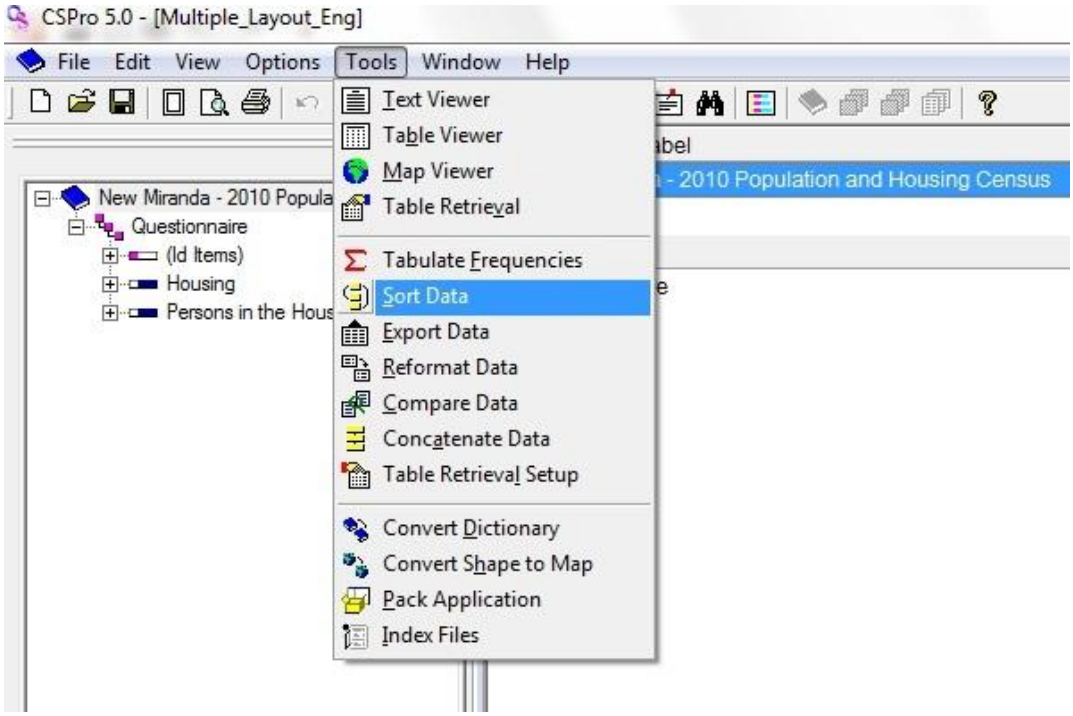
Such preparation of the input data file is generally done aside from REDATAM. Several software systems can be used for the purpose according to the source of the original file. For instance, SPSS and CSPro can be used internally for quality control with regard to the structure of data, checking the edits, cleaning the data and analysis of data quality while additional tools such as Microsoft Visual FoxPro and UltraEdit-32 may help in

identifying duplicate cases and the hierarchy and sorting of the database as well as in programming to count the total numbers of cases.

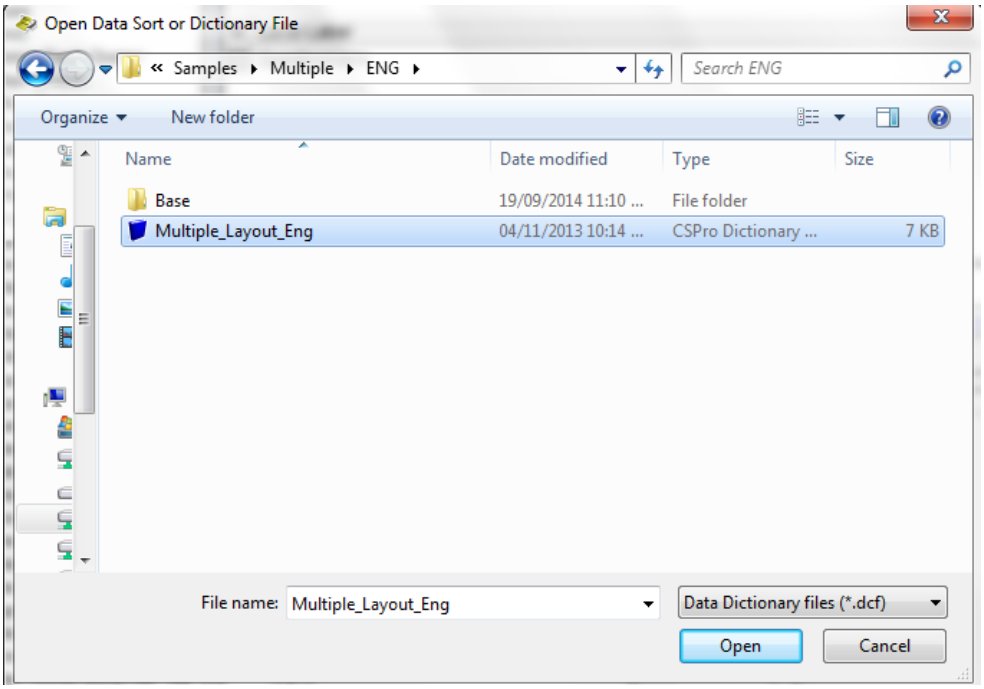
The input data file must necessarily respond to minimum conditions required for generating REDATAM database. It doesn't matter how long it will take to get through: one day, one week, one month or more! In clear, it is extremely important that the format of the input data file to be a self-documented file containing **sorted data** and descriptive information. The descriptive information is called the **dictionary** and contains variable names, type and locations (position and length), variable and value labels, and missing-value indicators.

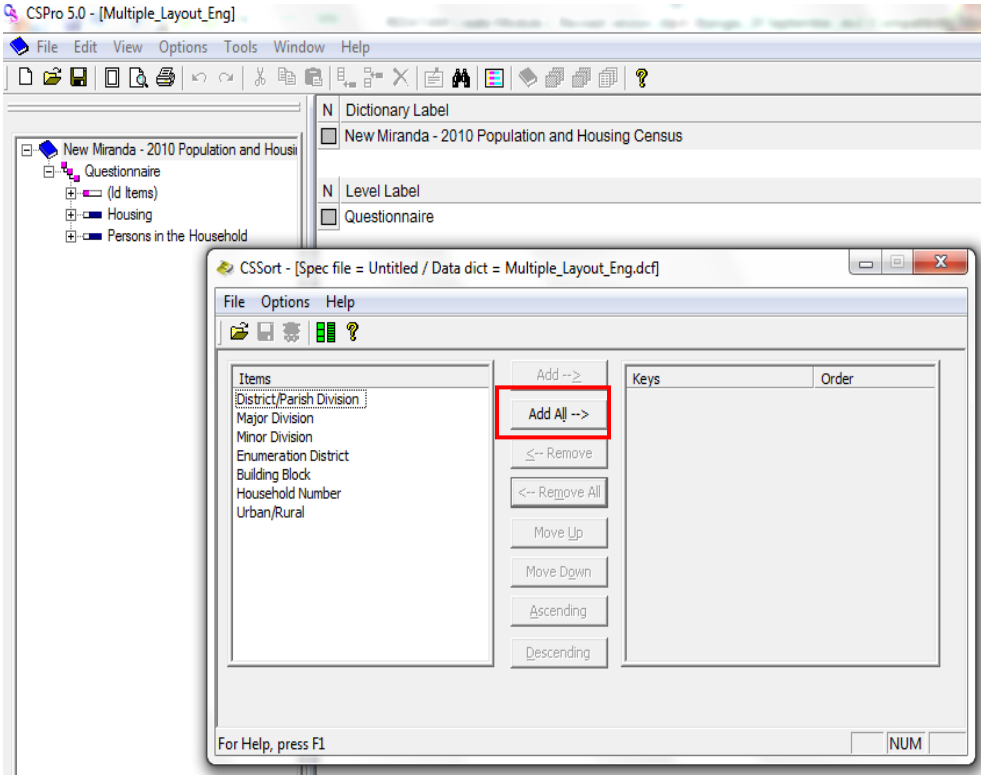
The exercises in this module will be illustrated using sample databases contained within the Red7 folder and which are among of the files downloaded with the software. The initial example is done using the multiple[6] sample file which is based on the New Miranda where the original file has been prepared using CSPro.

As shown below in CSPro we must sort the input data file using the Sort Data process, it may even not be possible to get a sorted input data file if the original file has errors such as duplicate cases.



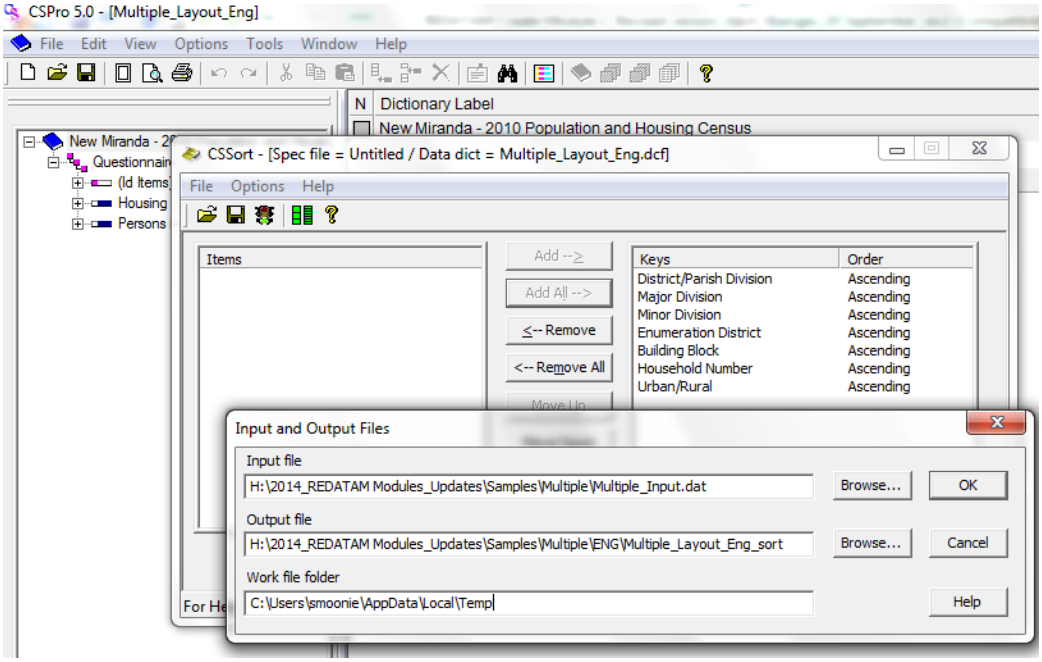
[6] Samples can be accessed within Program Files or in the location that the REDATAM7 software programme was installed (Redatam7/Samples/ Multiple).

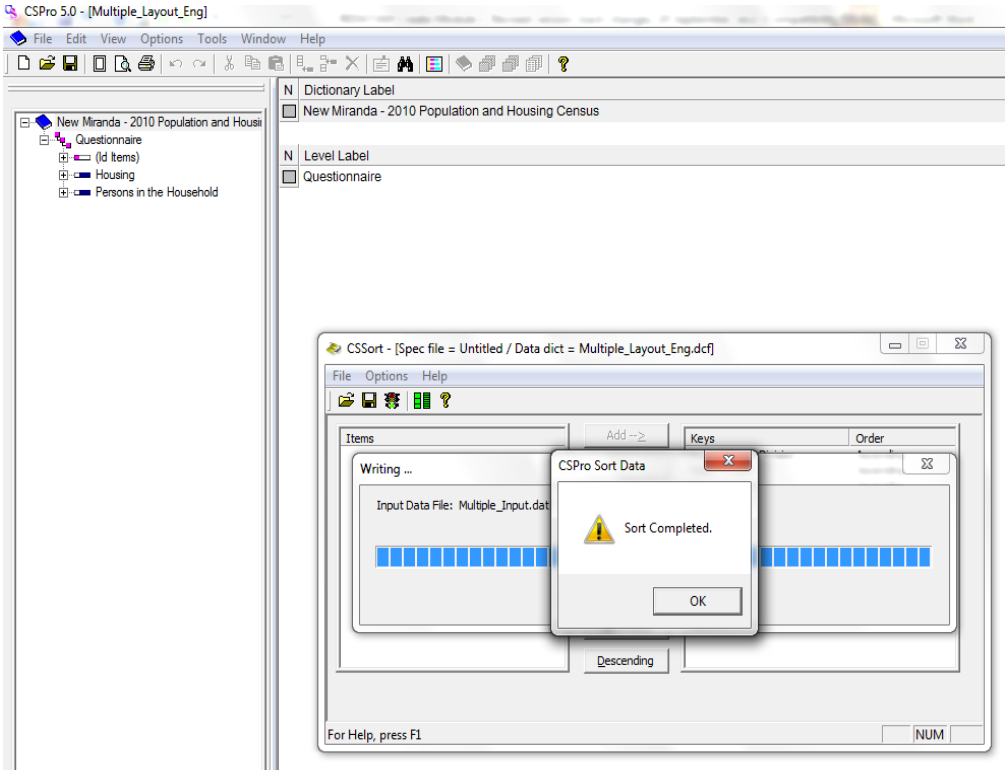




If there were no errors in the CSPro original data file, the Data Sort process will lead to the results as it follows:

Input File: Select the input data file with the .dat extension (Samples\Multiple**Multiple_Input.dat**)
Output file: Browse to working folder and provide a name for the sorted output file.





VI.3 Starting the RED 7 Create module

To start Red7 Create:

► From the Windows Start menu choose: All Programs > REDATAM7 > **Red7** Create

When you start a session on creation, you see the Red7 Create window with the toolbar as shown below.

New tool (for creating a new Scheme of REDATAM Creation Database)

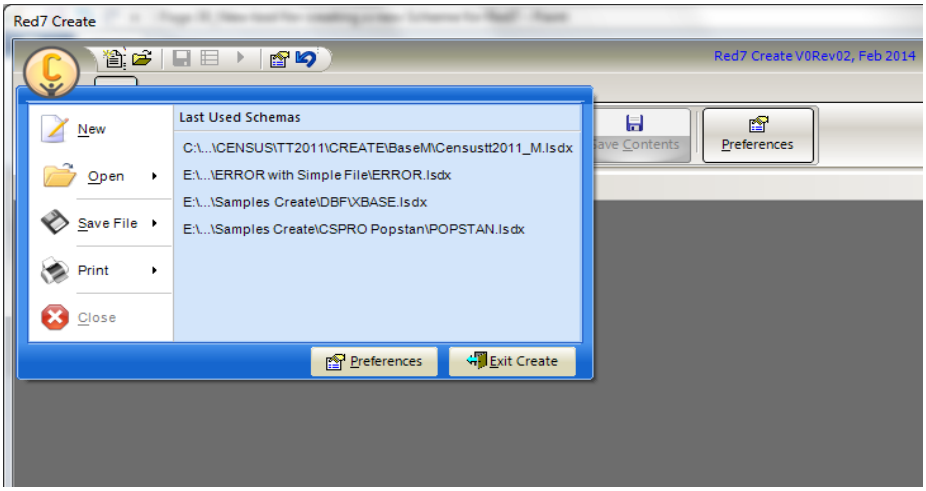
VI.4 Beginning to use Red7

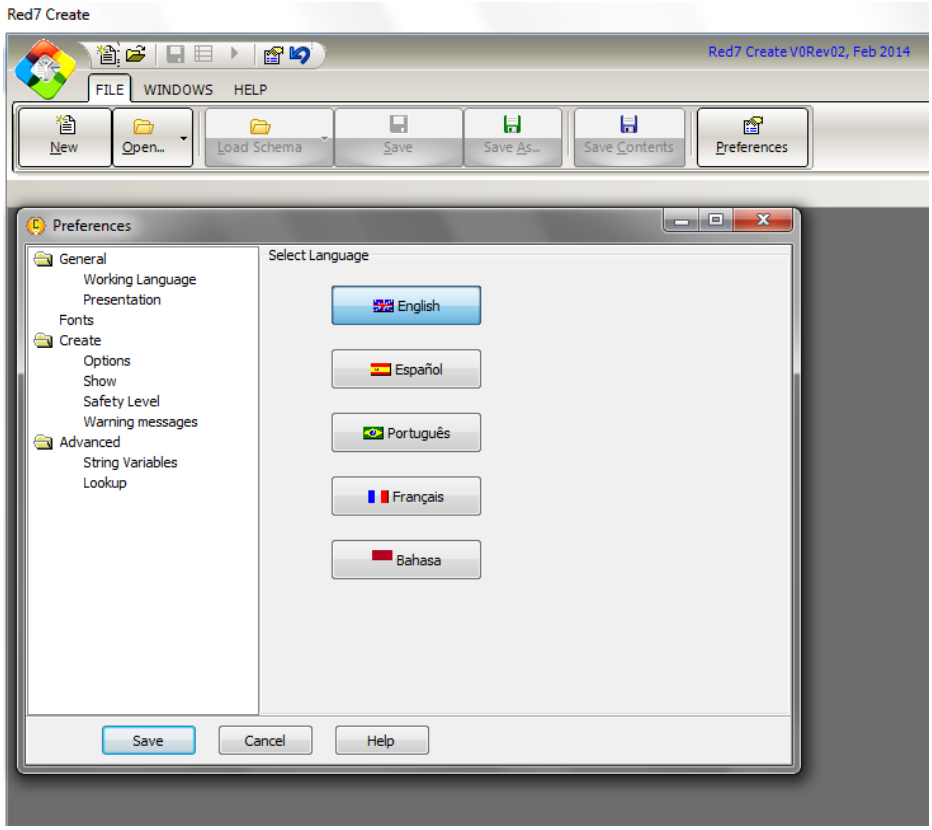
If desired, you can start by choosing your working language environment.

The text and screens of Red7 can be changed at any time among any of the following languages: Bahasa, English, French Portuguese or Spanish.

To change the language:
1. On the main menu bar, click on **File** > **Preferences**.
2. In the window that opens, click on the desired language tab.
3. Click on the **Save** button to save changes and return to the main menu.

NB: The selected language becomes the default language for the present and all sessions of Red7, although it can be changed at any time.





VI.5 Defining the database hierarchical structure

The database structure defines the hierarchical relations between entities, that is, mother-son relationships. This defines the relationships between the elements, since each entity can have elements, or specific entity instances. The database structure is then organized through entities or levels. Each entity is made of a set of entity elements. Each one of the levels represents a geographical level that is related hierarchically with the previous level.

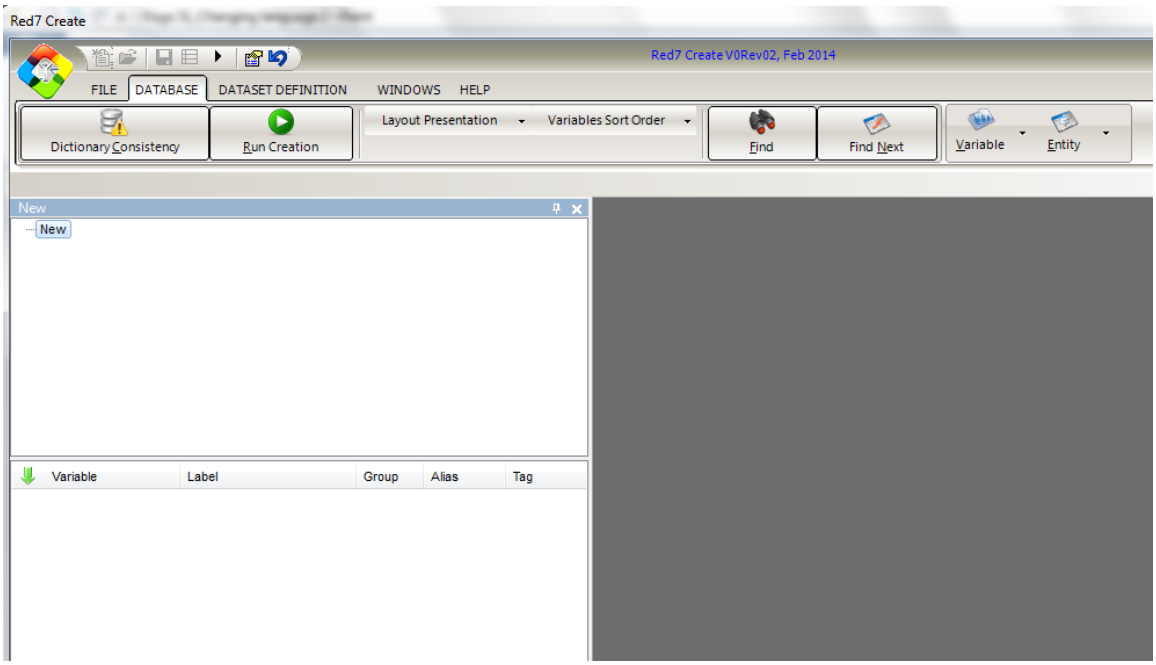
The *Red7* database generation process starts by defining the hierarchical entities, and its variables, in the Base Scheme, which is stored in a file with a .**wipX** extension.

To start a new Base Scheme:

► From the main menu choose: **File** > **New**

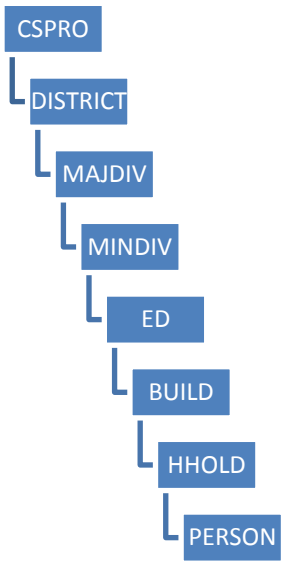
Alternatively, you can click on the ***New*** tool on the toolbar.

This opens an empty Base Scheme screen, where you can define the entities that form the hierarchical structure of the new database.



Then you can proceed to fill the screen with the hierarchical database structure. You should have already checked the data source in CSPro, for example, to verify the hierarchical structure to be reproduced in Red7 as shown here for the New Miranda 2010 Population and Housing Census file:
NOTE: For this example we continue to use the CSPro Multiple sample data file (Multiple_contained in the "**Samples**" folder.

NOTE: When renaming or adding names to entities, you need to decide on the text format for the naming conventions. It is advised to use upper case or capital letters for entities and variables instead of mixing them with small letters. Later on it facilitates the typing when programming, for instance.



1. ***Rename*** **the new entity (root entity): for example "NEW" to "CSPRO"**

NB: To be valid the entity or variable name must not have more than 8 characters.

► Highlight the root entity *NEW*, then right-click on it

► Select *Rename Entity*

► Type the new name (CSPRO, for instance)

Alternatively, you can just click on the entity name and than type the new name.

NB: This process is still the same when you want to rename any entity.

2. ***Add Child Entity*****: for example "DISTRICT"**

► Highlight an entity (CSPRO for example) and right-click on it to get the entity pop-up context menu

► Select Add Child Entity

► Type the entity name (Region, for instance)

3. **Continue repeating the same steps as mentioned above to add all child entities until the lowest level as defined in the hierarchical structure of the database**

In some cases, the questionnaire may contain a module or section that is applicable only to a subset of the population e.g. questions on migration (for members of the household who migrated) or mortality (for members of the household who died). In such cases, the corresponding entity would be included as a **brother entity**. To include this last entity, i.e add Person as child entity and insert Migration or Mortality as brother entity. These options are obtained in the same way, i.e by right-clicking on the highlighted entity and selecting *Add Child Entity* or *Insert Brother Entity*.

To add an entity, or to execute other maintenance functions, such as delete entities, move entities to another branch, etc., you should use the entities pop-up context menu.

Conclude this part of database creation by saving the database structure/ hierarchy. The Base Scheme file will have a .wipX extension. To save Base Scheme or database structure:

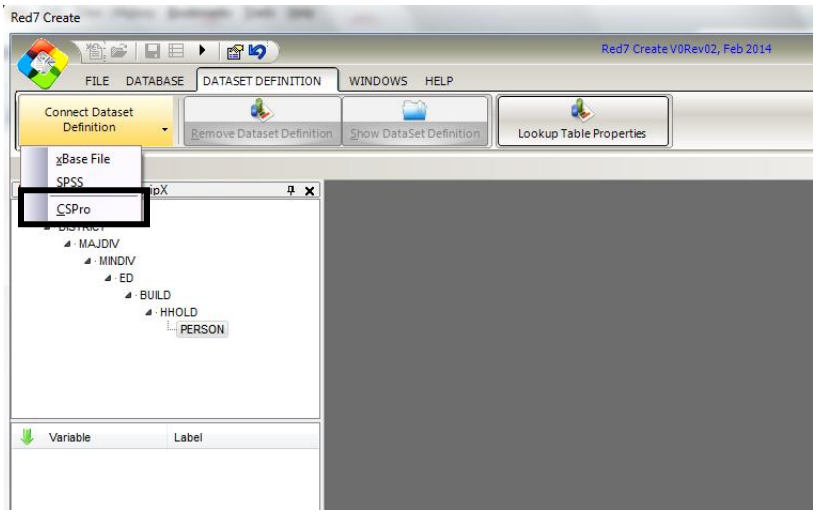
File > **Save As** > (include the name and save as file type Red7 Database Definition Scheme *.wipX).

VII. Creating a Database from CSPRO

VII.1 Defining Variables for each Entity

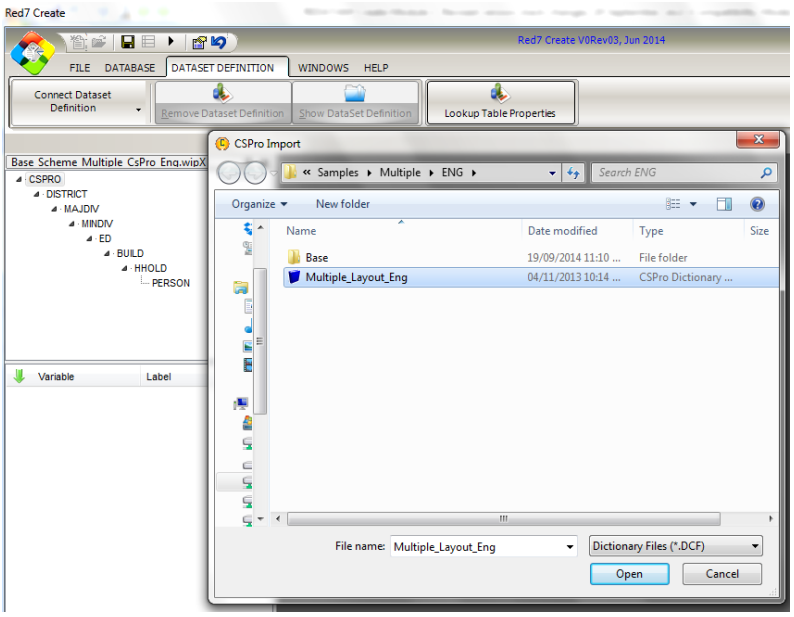
This activity consists of dragging the fields from the *DataSet Properties* screen and dropping them into the Variable dialogue box in the Base Scheme screen. To do so you need first to open the DataSet Definition option on the menu. How do you get there?

► First, from the main menu choose: Dataset Definition > Connect Dataset Definition > CSPro (for an input data file was prepared in CSPro format)



This opens a *CSPro Import* dialogue box, where you may browse to select the CSPro dictionary file (dcf) in order to conect the CSPro original data file information with the Base Scheme.

Select the data dictionary (**Multiple_Layout_Eng.dcf**) contained in the Samples folder.

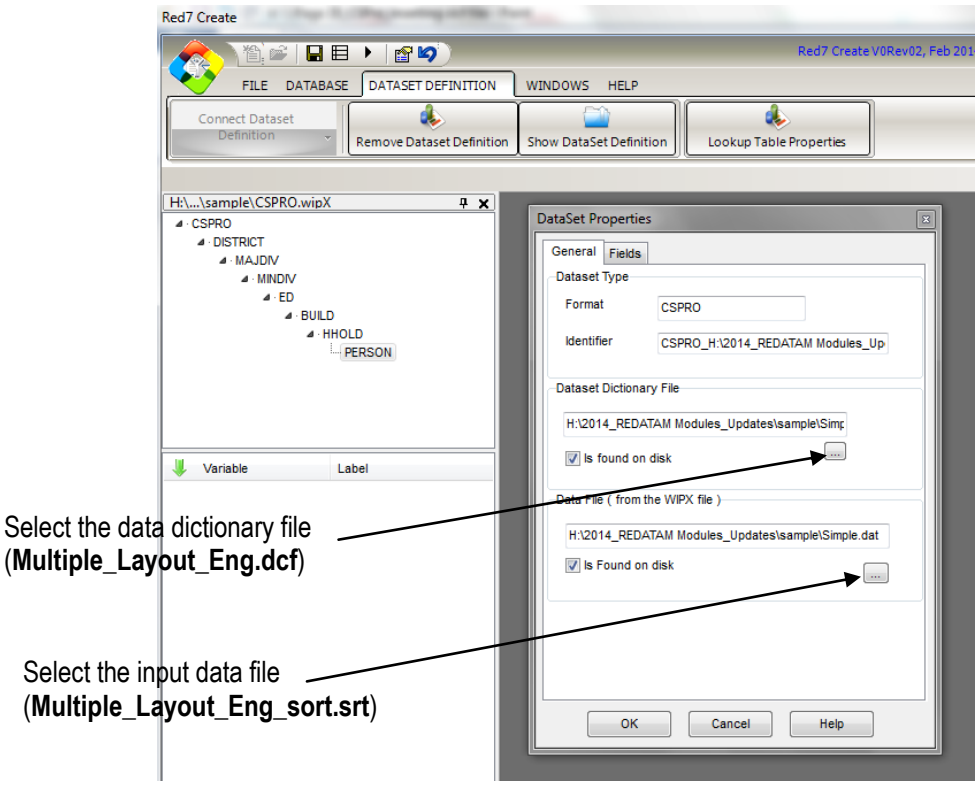


► Second, after selecting the CSPro dictionary file (Multiple_Layout_Eng.dcf, as shown above), click on *Open* button

You will get the *DataSet Properties* screen where, under *General*, you need to browse to search for the CSPro input sorted data file.

► Third, to search for the input data file, click on the isolated button as indicated above

You will obtain the *Input Data File* screen, where you will need to change the *Files of type* option to *All Files* (*.*) in order to find easily the input sorted data file with the extension *.srt (by default, it searches for *.dat files).



► Fourth, browse to find the input **sorted data** file (Multiple_Layout_Eng_sort.srt, for example), select it and click on the *Open* button

► Fifth, within the *DataSet Properties* window, click on the ***Fields*** tab to activate the related dialog box

► Sixth, for a selected entity to which you will add a variable in the Base Scheme, drag and drop fields from the DataSet Properties screen to the variables panel (or dialog box) in the Base Scheme

You then proceed as follows:

Select an entity: Click on the entity name in the Base Scheme.

Select the fields: In the DataSet Properties screen, find the field(s) you want to put in the selected entity. You can choose one or more fields, using Windows regular procedures, a click on the field name, or using the [Ctrl] key to choose separate fields, or the [Shift] key to choose a block of fields.

Drag & Drop the fields: Drag the chosen field(s) and drop them at the variables panel (or dialogue box) in the Base Scheme.

The field names will be copied to the variables part in the Base Scheme, becoming variables belonging to the selected entity in the Base Scheme and these new variables will be connected to the original fields in the Input Data File, from where they will get their values at generation time.

VII.2 Defining Variables and Entity Codes

Entity selectability is a property of an entity that allows the user to select independently every (geographic) entity in order to be able to use it afterwards in a hierarchical selection or as an output entity in an Arealist. By definition, all the elements of a database are *selectable*, that is they can be individually analyzed. However, for statistical confidentiality reasons for instance, an Entity can be *non selectable* notably when it refers to the lowest levels in order to avoid detailed and easily identifiable individual information.

Every selectable entity must have a variable to identify its elements. For an entity to be selectable it MUST have an alphanumeric variable (STRING type) whose value uniquely identifies its elements. In other words, it is compulsory that the variable is declared as STRING type (it doesn't matter if in the input file it is defined as integer) to be assigned an Entity Code.

It is advisable to proactively check first the characteristics of such variables in order to make sure they are of type STRING.

To do that:

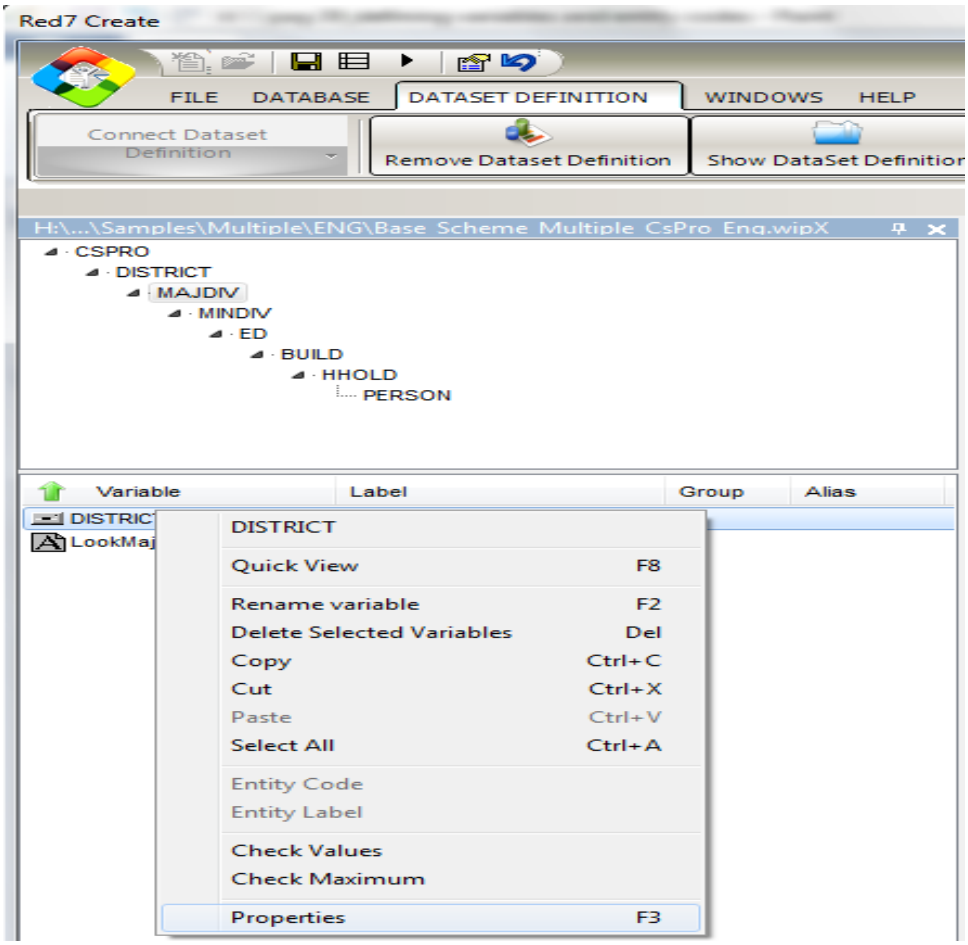
► From the main menu choose: **Dataset Definition** > **Show DataSet Definition**

► Click on an Entity (DISTRICT, for example). The assigned variable (DISTRICT) will be displayed in the contiguous Variable panel (or dialog box)

► Put the cursor over the selected variable (DISTRICT) and right-click on it

► Scroll down into the pop-up context menu that appears and choose Properties

Alternatively, you can double-click on the selected variable (DISTRICT).

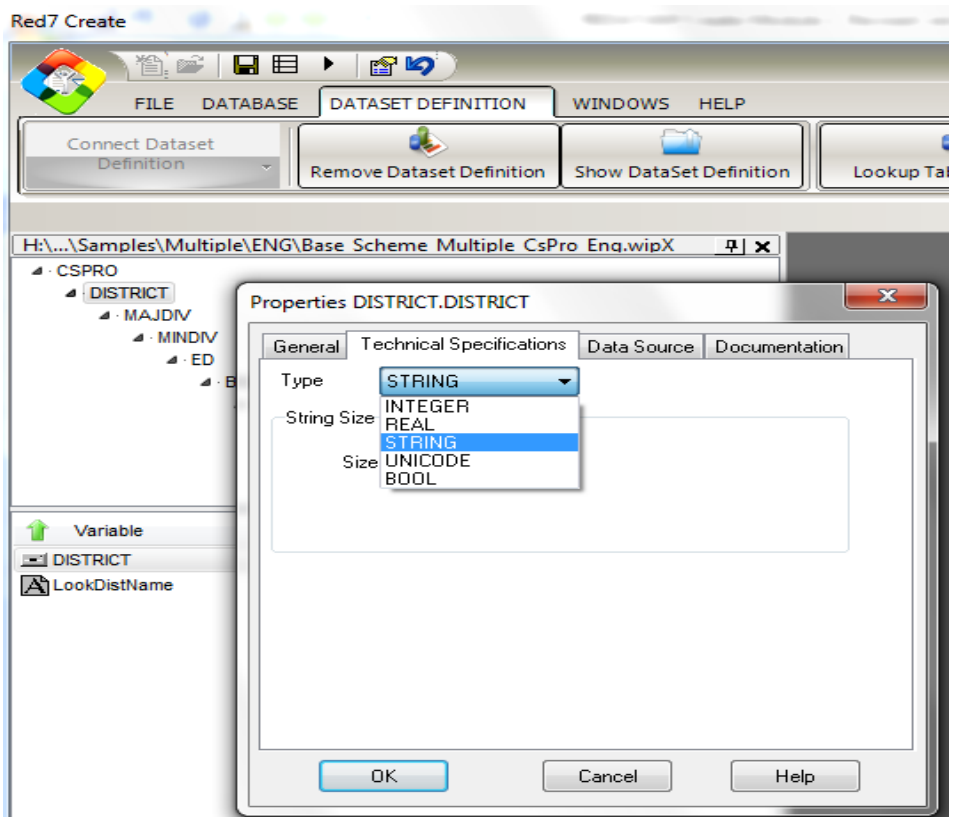


This opens the *Properties* dialog box for the selected variable.

► In the variable *Properties* dialog box, click on *Technical Specifications* tab

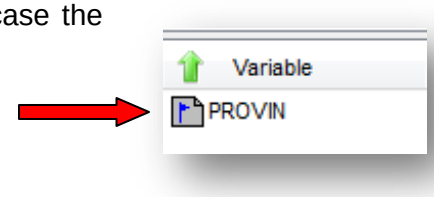
► And then, if applicable, change the variable *Type* to STRING

► Click OK to activate the change, if any.



VII.3 Assigning Entity Code to (Common) Variables

It is worth to recall that a variable MUST be declared as STRING type to be able to be assigned as an Entity Code in this case the icon next to the name is modified .



► In the Variable dialog box, put the cursor over the variable to be selected (DISTRICT for example) and right-click to obtain the pop-up context menu

► In the pop-up context menu, select *Entity Code.* The system acknowledges this command by displaying a (blue) flag at the left side of the variable name

NOTE: If the Entity Code is still not assigned to the variable (it means that probably the variable type is not a STRING):

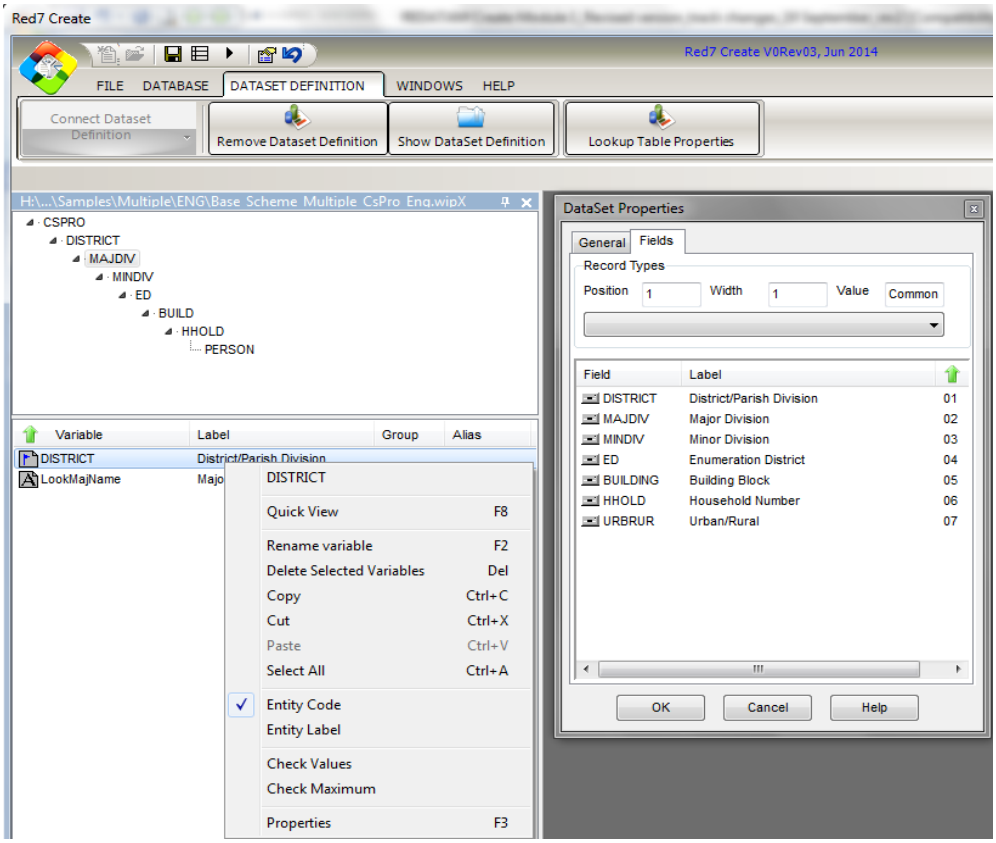
► Check its technical specifications by selecting by selecting *Properties* from the bottom of the list of options in the Properties pop-up context menu

Alternatively, you can double-click on the variable name to open the Properties pop-up menu

► In the *Properties* box, click on *Technical Specifications* tab

► Under *Type*, select STRING for the variable type

► The variable is now a STRING type, you may be able to assign an entity code. To do so, repeat the two first actions.



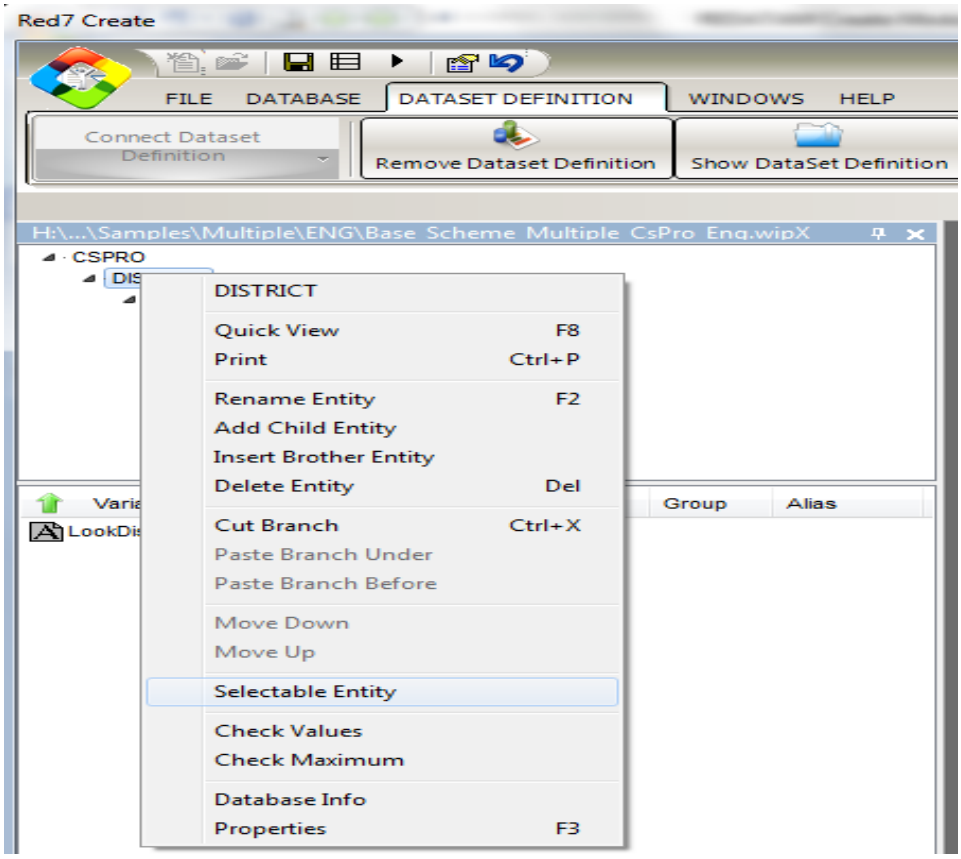
VII.4 Defining Entity Selectability

As stated above, to be *selectable* an Entity must have a variable of STRING type that identifies its elements and to which is assigned an Entity Code. In other words, an Entity cannot be selectable if its identification variable is not assigned an Entity Code. In the case of this New Miranda database, as an illustrative example, all the entities in the hierarchical structure (from DISTRICT to ED) must be selectable.

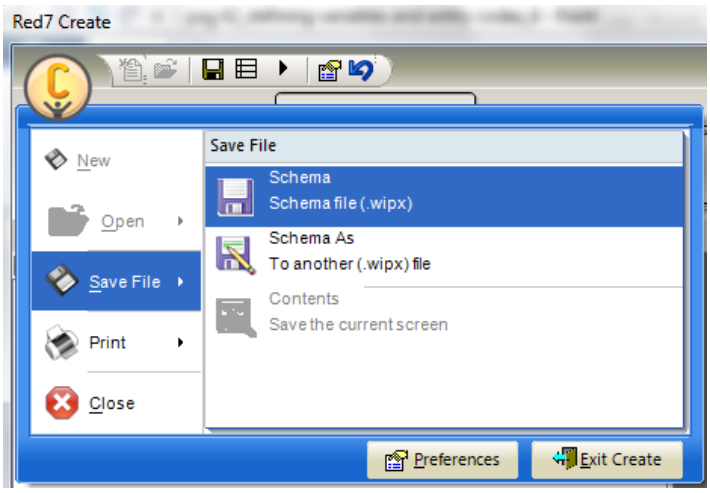
► To mark an entity as selectable, in the Entity dialog box put the cursor over the entity name and right-click to call the context pop-up menu

► In the pop-up menu, choose Selectable Entity

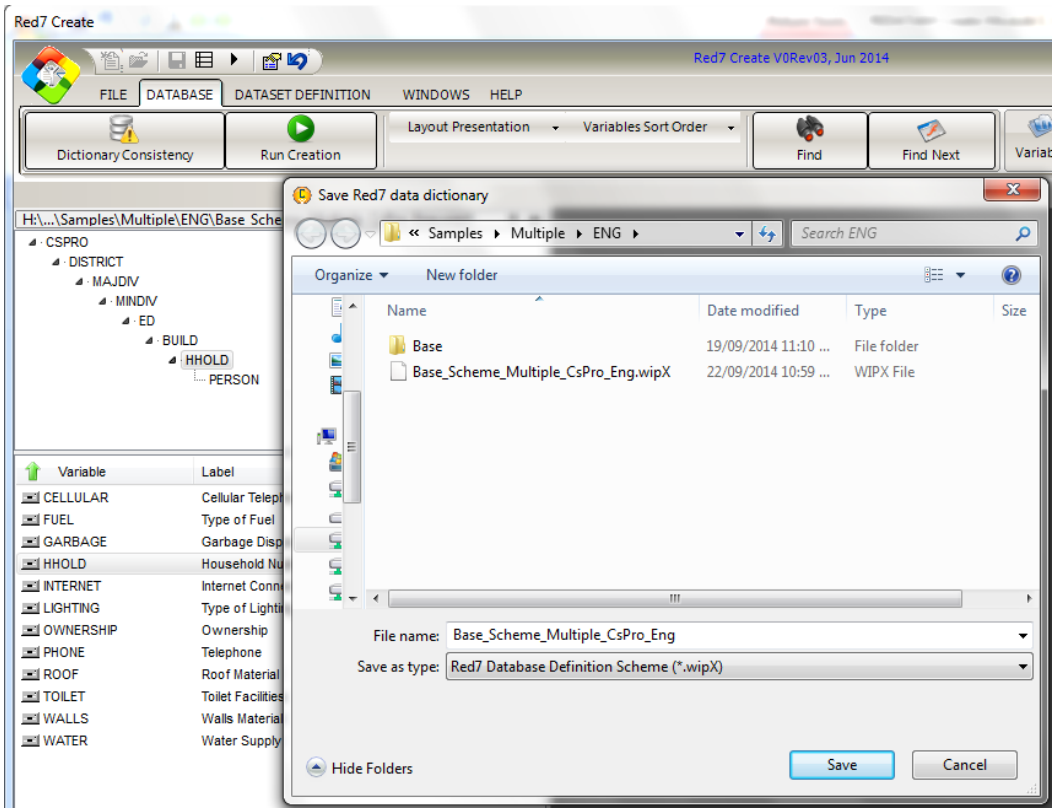
A check mark (√) will appear at the left side of the option, meaning that the entity is marked as being selectable. To unmark the entity (i.e to make it non selectable), call the pop-up context menu in the same way and click on the option (Selectable Entity) to remove the check mark.



When saving this scheme, the system writes a Base Scheme file, in the chosen directory, with a .wipX extension.



This leads to the REDATAM database structure as shown below.



Relationship between .wipX and .dcf files

The CSPro .dcf dictionary file contains the definition of the fields that are found in the input dataset and some information about its record types, common geographies and the way the records are organized in the file. This definition is used to "populate" (feed the fields to be the variables) the Base Scheme to be used in the future REDATAM database.

The .wipX file, on the other hand, is the file used to generate the REDATAM database. It is a template containing the structure, entities and variables that will exist in the REDATAM database. The template is used as a framework for REDATAM to grab the original variable and migrate it to the final location in the REDATAM database, transforming the organization of the data, never the values.

VII.5 REDATAM Database Generation

If the Base Scheme file is not the active file, you must open it first. To do that:

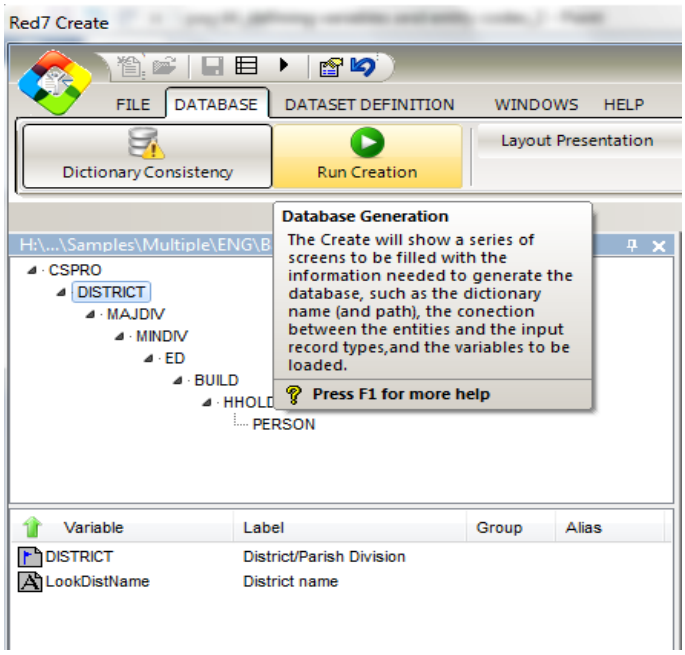
► On the main menu bar, click on File > Open, select the Base Scheme file with a .wipX extension (Base_Scheme_Multiple_CSPro_Eng.wipX, for example), and click on the *Open* button

Alternatively, you can click on the *Open* tool on the toolbar to call the Red7 Open Dictionary Definition pop-up menu

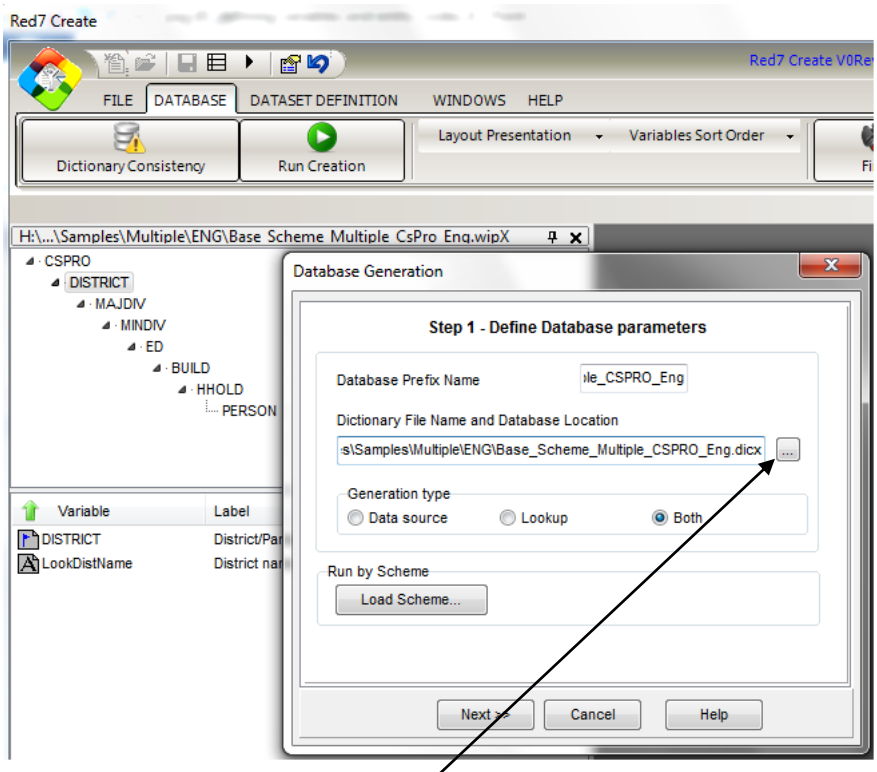
If the correct Base Scheme file is active, you can proceed as it follows:

► On the main menu bar, click on Database > Run Creation to call the Database Generation context pop-up menu (Step 1 – Define Database parameters)

Alternatively, you can press F9 to obtain the same result



From the Step 1 – **Define Database parameters** pop-up menu, you can locate the Dictionary File Name if necessary, by clicking on the "search tool" for the dictionary file as indicated below.

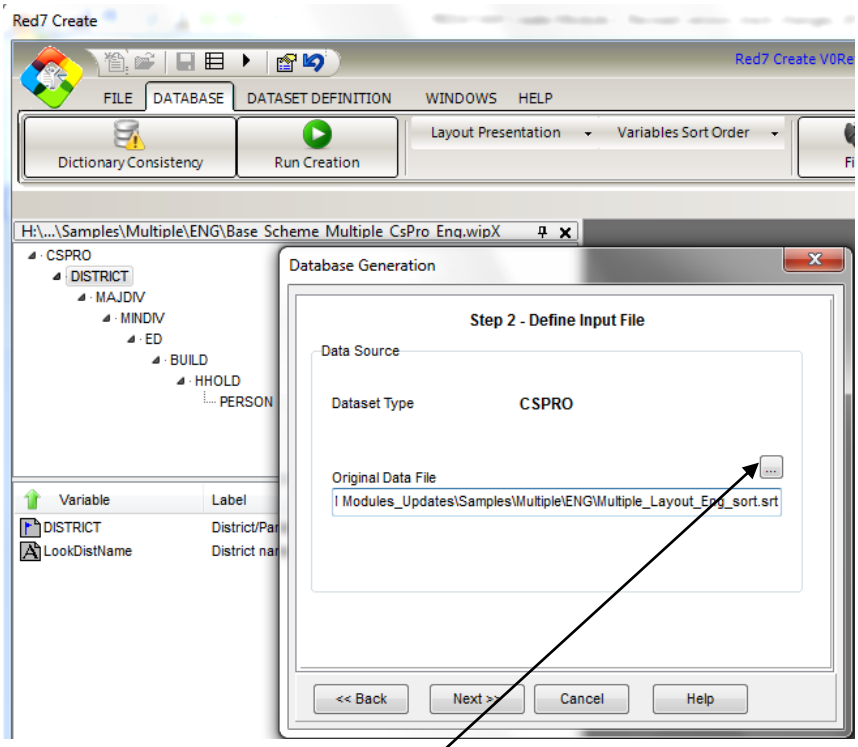


Search button for the dictionary file

After locating and selecting the dictionary file, click on *Next* at the bottom of the context pop-up menu to continue to the second step of the Database Generation process.

This opens the Step 2 context pop-up menu to define the input file.

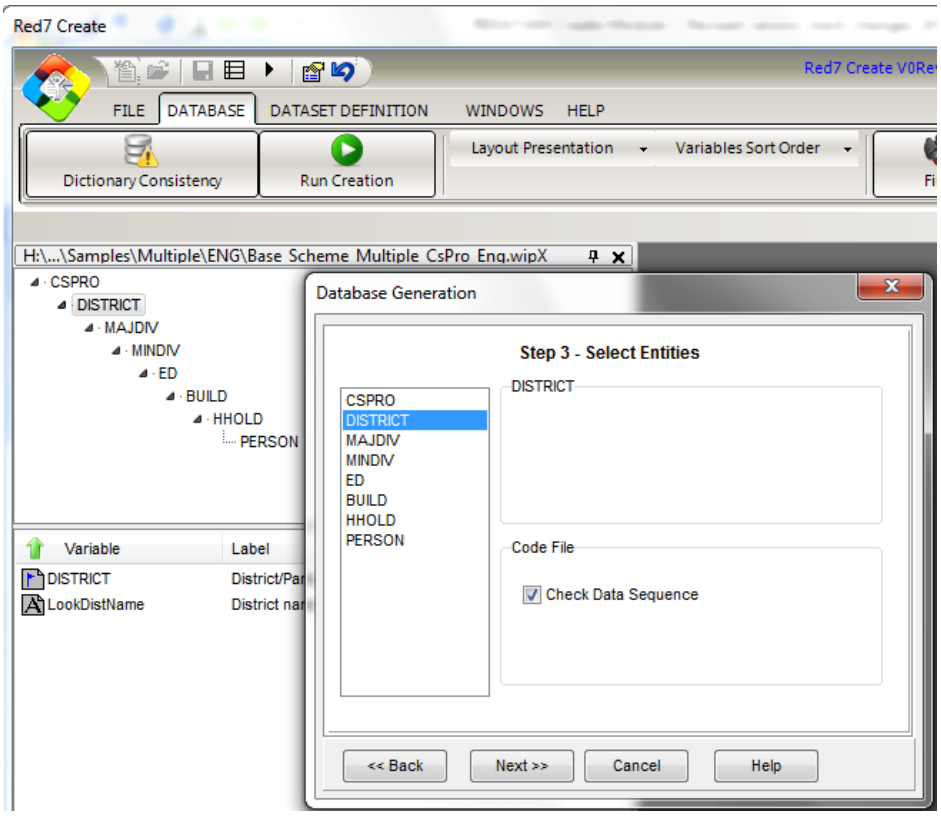
From the Step 2 – **Define Input File**, you can locate and select the original sorted data file according to the dataset type (for example, the CSPro sorted data file named Multiple_Layout_Eng_sort.srt) using the “search tool” for the original data file.



Search tool for the original sorted data file

After locating and selecting the original sorted data file, click on *Next* at the bottom of the context pop-up menu to go to the third step of the Database Generation process.

This opens the Step 3 – **Select Entities**.



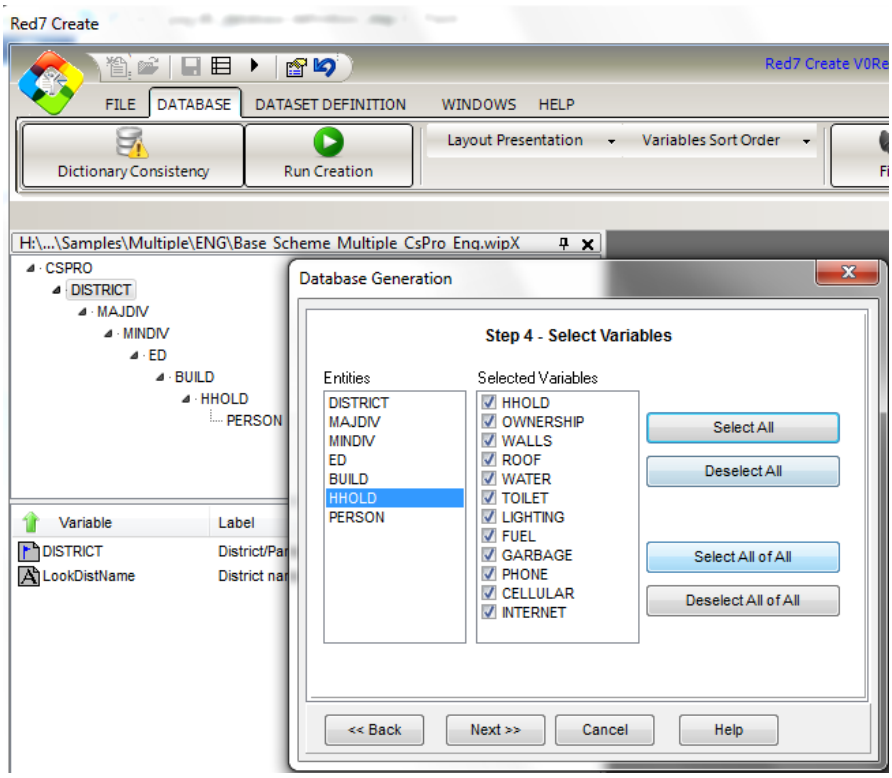
In the Step 3 – Select Entities pop-up menu, click on the entities one by one and:

► For the entities DISTRICT, MAJDIV, MINDIV, ED, BUILD, and the option *Check Data Sequence* has been selected by default

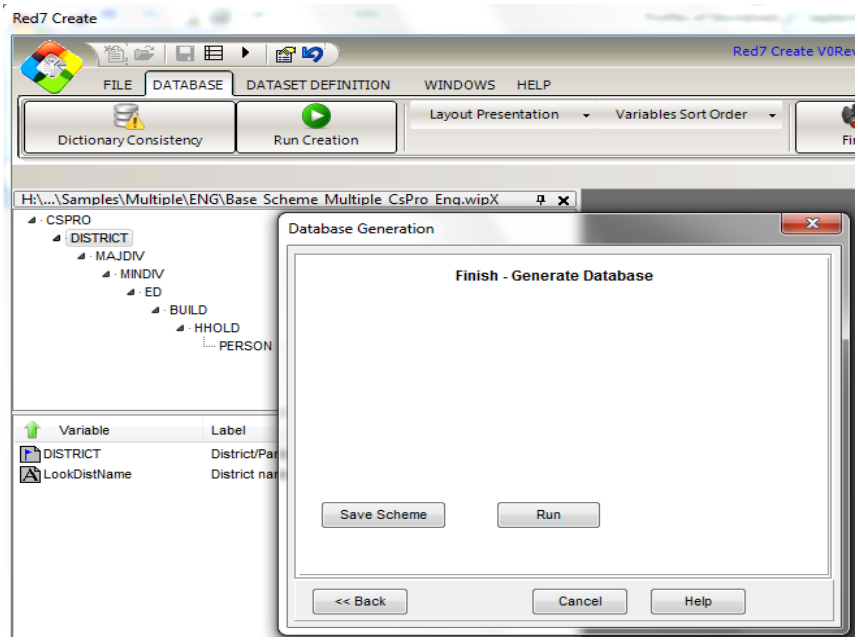
► For the entities HHOLD and PERSON, select *Create Pointer File* for each entity name and assign an input record type to the entity by clicking on the Back down-pointing triangle (▼), the symbol under Break on record type. For example, HOUSING will be assigned to HHOLD, POPULATION to Person.

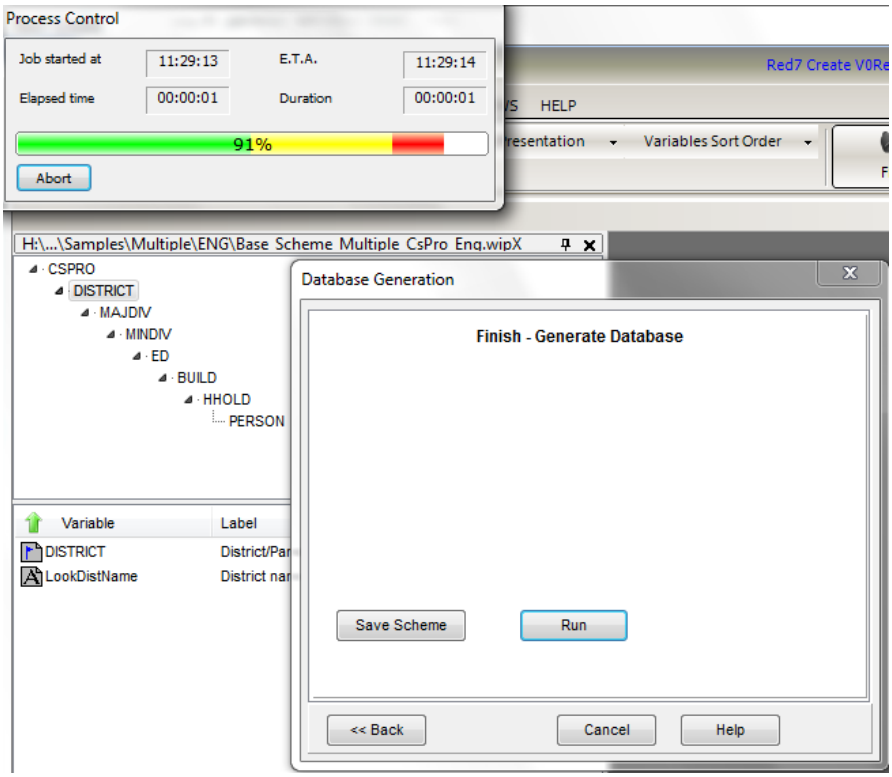
When completed, click on *Next* at the bottom of the context pop-up menu to proceed with the fourth step of the Database Generation process.

This opens the Step 4– Select Variables.
Step 4 – Select all the variables for the following entities: BUILD, HHOLD and PERSON


Red7 Create
FILE
DATABASE
DATASET DEFINITION
WINDOWS
HELP
Dictionary Consistency
Run Creation
Layout Presentation
Variables Sort Order
Database Generation
Step 4 - Select Variables
Entities
Selected Variables
DISTRICT
MAJDIV
MINDIV
ED
BUILD
HHOLD
PERSON
HHOLD
OWNERSHIP
WALLS
ROOF
WATER
TOILET
LIGHTING
FUEL
GARBAGE
PHONE
CELLULAR
INTERNET
Select All
Deselect All
Select All of All
Deselect All of All
<< Back
Next >>
Cancel
Help

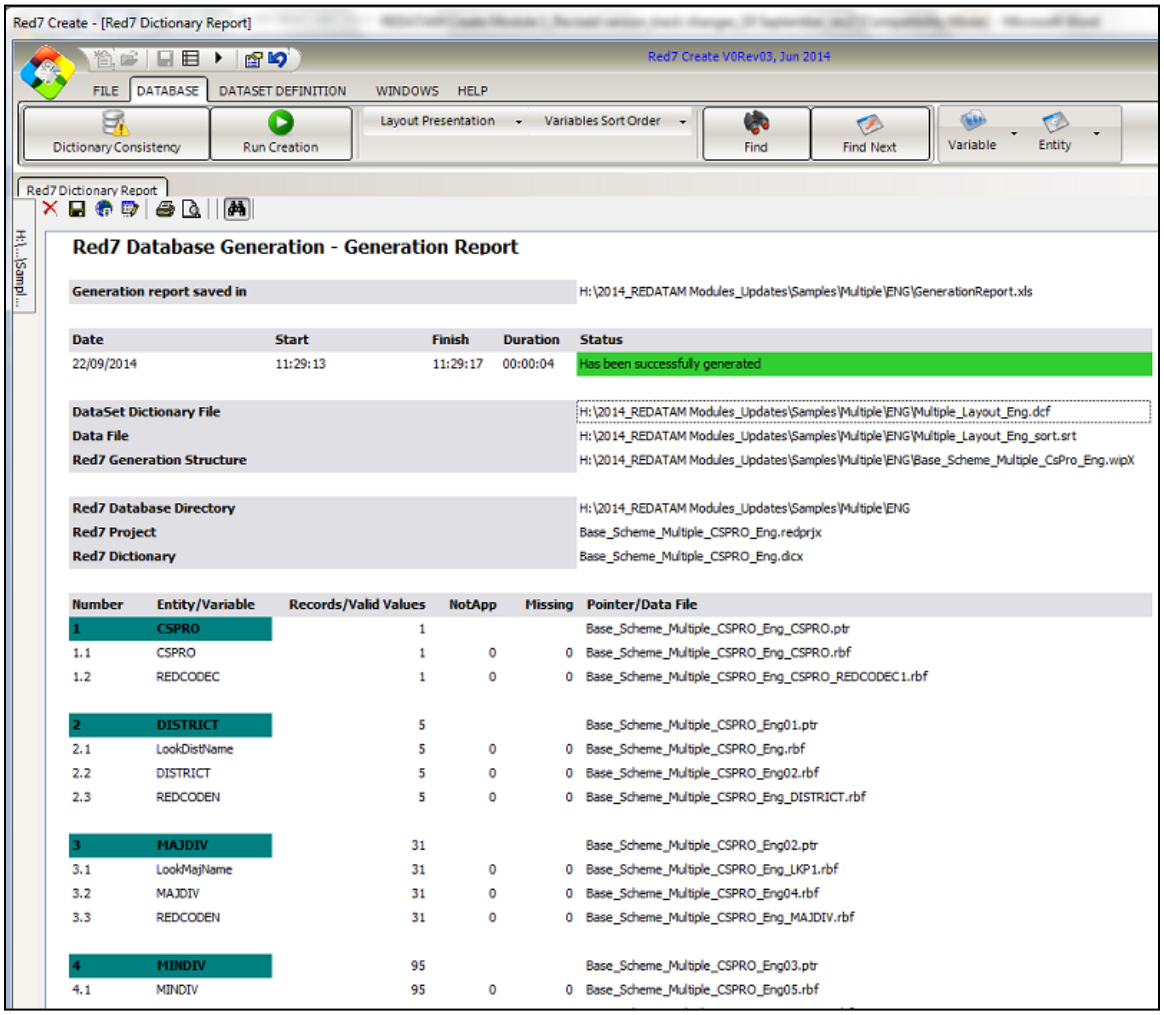
Select RUN:


Red7 Create
FILE
DATABASE
DATASET DEFINITION
WINDOWS
HELP
Dictionary Consistency
Run Creation
Layout Presentation
Variables Sort Order
Database Generation
Finish - Generate Database
Save Scheme
Run
<< Back
Cancel
Help


Process Control
Job started at
11:29:13
E.T.A.
11:29:14
Elapsed time
00:00:01
Duration
00:00:01
91%
Abort
Red7 Create V0Re
HELP
Variables Sort Order
H:\...\Samples\Multiple\ENG\Base Scheme Multiple CsPro Enq.wipX
CSPRO
DISTRICT
MAJDIV
MINDIV
ED
BUILD
HHOLD
PERSON
Variable
Label
DISTRICT
LookDistName
Database Generation
Finish - Generate Database
Save Scheme
Run
<< Back
Cancel
Help

VII.6 Database Generation Report

It is important to have a look on this report in order to control the quality of data generated by checking the total number of records for each entity and the values (valid in range, out of range, and not applicable) for all the variables.



Since you have successfully generated the REDATAM (Red7) database, you must obtain finally the following files.

VIII. Files of a REDATAM Database

In general, a REDATAM database is composed of its Dictionary, plus two other file sets: i) the data files; and ii) the index or pointer files.

Data files (.rbf)

REDATAM stores each variable of an entity in a single file, with a "record" for each element in the entity. Then, each variable file looks like a vector of data. Those files are known as transposed files, to distinguish them from the more common statistical data organizations, where all the variables of an element are stored in the same record, and one single file has all the information for all the elements.

One of the advantages of the transposed structure used in REDATAM is that the processing speed is very fast, since it allows the system to read and process only the variables that are involved in the program.

Pointer files (Indexes .ptr)

Those files are the ones responsible for the connection between the entity elements and the lower entities. Each entity has an index file, with values that "point" from the elements in the higher entity (mother entity) to the elements of the entity itself.

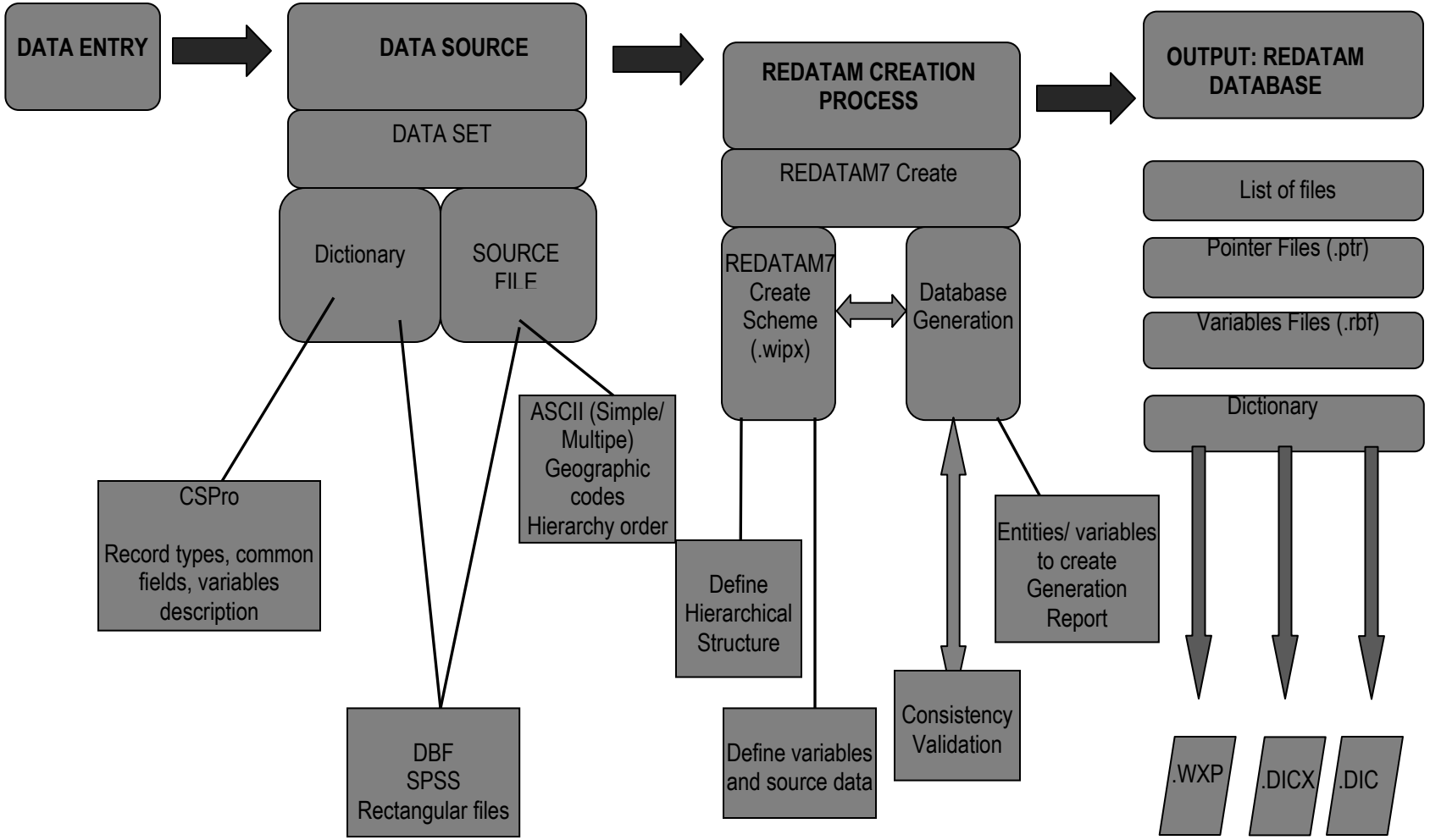
Dictionary (.dicX)

All the information about the entities and variables are stored in a file called Database Dictionary. This set can be defined as the metadata, which means "information about information". The dictionary is the bridge between the user and the data files. It allows the visualization of the data in terms of variables and records, freeing the user from the physical details of data storing and handling.

The dictionary contains a list of all the variables of a database for each of the hierarchical levels (entity), as well as the codes (values or categories) for each variable, with a brief description of the meaning of each code. For example, the codes for the sex variable of the person level are 1 and 2, where 1 = male and 2 = female. Generally, the database dictionary is defined only once. You can review it (search, print, browse, etc.) using the dictionary window that can be opened from the main menu.

Note: A REDATAM database generated using Red7 will have a .dicx extension. To read it in Redatam+SP Process module it is necessary to import its ASCII version (wxp file) instead of opening the dicx dictionary directly. Once in REDATAM Process module this dictionary can be save with the old .dic extension.

IX. Scheme of REDATAM Creation Process



X.

XI. Summary of steps to control in the Red7 database generation process

1. Select the original source file; verify it contains the data file as well as the dictionary file (for CSPro, IMPS formats), SPSS and DBF contains data and definitions in the same file.

2. Open the source file and analyze the variables as well as the geographic entities.

3. Check for the identification codes for each of the entities you will use in the REDATAM hierarchical structure.

4. Open the Red7 Create module.

5. Create the hierarchical structure (from NEW)

    a. Name the root entity of the base with a small, short name that will represent the name of the variables with the .rbf extension (8 characters).
    b. Use friendly names for the entities.

6. Save the Creation Scheme (.wipX extension).

7. Connect with the Dataset Definition.

8. Define the file containing the data (.dat, .txt, .srt file or .sav, .dbf file).

9. Identify the selectable entities.

10. Drag and drop the code variable for that entity.

11. Verify that the type of the variable is “STRING”.

12. Convert that variable as the “entity code”.

13. Make the entity itself “selectable” and move to the next entity, repeat from 10.

14. Drag and drop the rest of the variables to the window bellow the entity that will hold those variables.

15. Verify source values, maximum and minimum values using the option verify values from the popup menu. You can save the outputs for later comparison between the REDATAM database and the source database.

16. Verify that the ranges of the variables are adequate for each type of variable either INTEGER or REAL. Define Not Applicable and Missing values if you know them.

    a. Use 0 for NA in case these are values out of the valid range.

b. For MV (missing) use the maximum value plus one.
c. If the value 0 is valid, check for the existence of NA values and assign a different value.
d. Identify those variables declared as STRING but actually they will be used as INTEGER in REDATAM and change the type in the properties window.

17. If you need to split a variable into two, change the relative position number under the properties window as well as the size. Give a different name to each component.

18. If there are labels missing to any category add it in the properties window.

19. Open the root entity properties window and add the following information:

    a. Label
    b. Date of creation, author, information about the database

20. Save the Scheme. Review the warnings and errors messages and correct the errors, repeat this step until you do not get any error.

21. Run the execution process (run button), go step by step:

    a. Step 1:
       - Name and location of the dictionary to be created, this location will apply to the other files (rbf and ptr) as well.
    b. Step 2:
       - Name of the source data file.
    c. Step 3:
       - Verify that each entity that is going to be selectable has its pointer file defined.
    d. Step 4:
       - Select the variables to be included in the database for each entity.
    e. Step 5:
       - Save the run scheme (file .lsd) together with the scheme file (.wipX)
       - Run the process.....wait for the creation report.

22. Examine the Creation report:

    a. Compare the total number of variables and entities.
    b. Verify the NA and MV (not applicable and missing values). If they don't correspond it means that the variable didn't read the proper values from the source data file. Verify the start position and the size of that variable.
    c. Verify the total records number at the household entity and the person entity and compare to the source database. (you shouldn't lose records)

23. If there are inconsistencies in the report go back to the scheme and verify the definitions as well as the source file.

24. Before you create again the database, please delete previous files (.rbf and .ptr).

25. Run the process again, load the creation scheme (.lsd file), step N 21.

26. Review the creation report again.

27. Repeat until you get the REDATAM Database without error or inconsistencies.

28. Save the last creation report together with the .wipX file.

29. Go to REDATAM Process module and run a frequency for all variables.

30. Review the results and check for total number of records. If there are inconsistencies go back to step 15.

31. Save the REDATAM database and make a backup of the REDATAM data base.