STATISTICAL COMMISSION and
ECONOMIC COMMISSION FOR EUROPE

Working paper No. 20
English only

CONFERENCE OF EUROPEAN STATISTICIANS

UNECE Work Session on Statistical Data Editing
(27 – 29 May 2002, Helsinki, Finland)

Topic (iii): Editing of administrative data

CHYPRES, A PROTOTYPE OF DATA EDITING TOOL FOR HYBRID SURVEYS

Invited paper

Submitted by the University of Southampton, United Kingdom[1]

I. INTRODUCTION

1. The ins and outs of the statistical production process are undergoing a gradual evolution. The standard model, based on a single collection medium (the questionnaire), has in practice largely fragmented, and a change in methods seems eventually inevitable. In the medium to long term, the survey will cease to be the primary means of collecting information. Instead, existing sources of information, with different characteristics, will be combined using more or less automatic processes, with the survey perhaps serving to complete the field and/or the variables. Collection will become a mixed system, drawing on several sources of information, a «*hybrid data collection*».

2. This will make the data collection process more complex, more adaptable, so that we need now to start reconsidering the tools we use. We developed a small-scale prototype of generic tool, called CHYPRES (Collecte HYbride Pour REstitution Statistique), without being too ambitious as regards its functional scope or the quality of the user interface. What we wanted was to develop concrete methods of testing approaches. To do this, we needed a tool to facilitate the task of integrating multiple files, while giving the user overall control of the data collection and processing processes. Such a tool, based on a full and coherent representation of metadata, would have a transversal value in the statistical system, and especially for statistics collected from enterprises. Indeed, the introduction at INSEE of a so-called «statistical» register and of a system of registers, and more generally the desire for increased coordination of sources, all naturally leads us to examine the transmission of large quantities of information from a given source for incorporation into the system of registers.

3. However, to start with the basics, why a general tool for multiple data file integration? There are at least three major reasons to examine the question of tools:

i) **Time-saving**, since it removes the need for constant rewriting of the programs designed to integrate such files. It is the metadata that describe the information that should circulate within the statistical system, establish a common culture, fundamentally superior to a documentation system insofar as it serves as the basis of the file reading and integration programs;

ii) **Uniform practices** between the different departments;

iii) **Better control of information content**, since the knowledge stored over time (in the form of metadata) is a valuable source of information for identifying the true content of data files.

[1] Prepared by Pascal Rivière (riviere@socsci.soton.ac.uk).

II. THE PROBLEMS

4. Information collection is a process that by nature involves inputs and outputs. The essential characteristics of hybrid data collection are the *heterogeneity and variability of the inputs*: tax sources of different kinds, questionnaires, accounting data servers, ... The information retrieved will vary in its degree of order, and will not always be directly verifiable with the unit in question.

5. On the other hand, whether we are talking about *hybrid data collection* or not, the output of the collection process, before any statistical analysis, is always the same from a theoretical point of view. In the end, the essential aim is always to finish up with a database of individual data, a matrix:

- along the «rows», units: enterprises, or branches, but they must all be of the same type;
- down the «column», variables (quantitative, qualitative, textual) that describe these units: here again, a certain homogeneity should reign; in principle, a single variable should mean the same thing from one unit to another.

6. The final objective of this operation (data collection from existing sources, and simultaneously data editing) is to have a healthy database, as complete as possible, with a minimum of aberrations, or rather a minimum of major inconsistencies. Certain values might conceivably remain missing, as long as when they are subsequently processed this is done by recalculation and not by allocation.

7. The fact that the data collection is hybrid changes virtually nothing in terms of database content, except in one notable respect: if the collection is multi-source, additional metadata are required (e.g. for each datum, the information source it comes from). For example, one variable may come from tax returns, another from an accounting data server, another from the survey itself, another from the register clerk, … The source must be known. As we have seen, it is not the final outcome, but the actual collection process that changes.

III. REAL CONTEXT OF THE INFORMATION COLLECTION

8. In normal operation, the situation at the time a «file» is received from a certain information source can be described as follows:

i) At that moment, a reference database exists, but it is *incomplete*. At the start of the collection process, the database is essentially empty, and it fills up as information comes in from different sources.
ii) The incoming file contains *errors, missing data, incorrect identifiers, aberrant values*.
iii) The file variables do not always correspond to the database variables: they simply allow us to determine certain variables in the database, sometimes very imperfectly, since the *definitions* differ from those of the reference database.
iv) The *units* in the source file do not always correspond to the database units: for example, they might relate to a group of branches and not one branch.
v) From one source to another, the *reference periods* of the information are not really the same.
vi) With qualitative variables, the *modes* in the information source are not always the same as the reference modes, both in terms of meaning and notation.
vii) The file in question contains a whole stack of *useless information*, which it serves absolutely no purpose to collect. In fact, a majority of the source file may be useless.
viii) With the exception of any comments or headings, the file obeys a certain *structure, which is fixed but not perfectly documented*.
ix) The *identifiers* are not necessarily the reference identifiers (in this case the register identifiers).
x) We do not know what *processing* has been done on the data: we do not know if it has been adjusted, modified by a clerk, etc.; nor do we know why any such processing was done.
xi) Information *changes* constantly, and sometimes very quickly.

The situation described above is in no way an exaggeration: it is the norm. And that is why *data editing* is so important.

9. Given that the errors affect millions of pieces of data, it is absolutely impossible to verify everything by hand: verifications done by hand must be relevant, effective, and optimised to meet statistical objectives. The verification process must also be quick, so that statistics can be provided as rapidly as possible: the freshness of the statistics produced is an essential criterion of quality. Consequently, *trying to achieve «zero error» in data files is pointless, and even highly damaging*. Even if unlimited time were available, it could probably not be achieved in any case.

10. As can be seen, the context is not an easy one, but it is what we encounter in practice. It must be emphasised: information is dirty, that is its nature. To react to this characteristic by talking about the necessity of *cleaning up* data is a profound error, since it means emptying the subject, reducing it to a single thankless process that is supposedly easy to define. The perpetually imperfect nature of information is a permanent, intrinsic characteristic, which has to be lived with, and which it is pointless to try to eradicate: there really is no such thing as a big once-and-for-all *cleanup*, but instead a collective, multiform and above all, continuous process.

11. So what do we do? Technically, there is a key to managing what we do more effectively, to striking an acceptable balance between cost and quality: strict control of the metadata, combined with simple, high-speed tools for «viewing» all the data. Instead of trying to eliminate the impurities, we give ourselves the means to «see» these imperfections clearly, and to handle them advisedly. This article is about how this can be done.

IV. DEFINITIONS OF THE FILES OR DATABASES FOR INTEGRATION IN CHYPRES

12. Hybrid data collection and the data editing tasks derived from the comparison of several sources presupposes that we have information originating from several sources. Therefore, in the CHYPRES system we need to define what we have and, in particular, to develop a vocabulary specific to hybrid data collection.

The different files handled in CHYPRES

13. As we have seen, the aim is to integrate several data files so as to construct and gradually expand a database of individual data. These files will be retrieved one by one by the software.
NB: *The term «file» here is used imprecisely, so as not to complicate things: they are sources of information, which can be in the form of a file or database; the form in which the information source comes is of little importance to the concept; it could also come from websites.*

14. We will use the abbreviation **EF** (**external file**) for the data file we want to integrate, a file that «naturally» possesses a certain number of the defects listed above, for example tax files, survey files, etc.

15. The abbreviation **RDB** (**reference database**) will be used for the database gradually constructed (and updated) by integrating successive external files. Essentially, for a given hybrid data collection, there is one RDB and several external files such as survey database and register. That having been said, what is specific about a register is that we do not start from zero: there is an existing status that changes constantly as data arrive.

16. An **ERF** (**external reference file**) is a particular kind of EF, in that it constitutes a reference as a list of units. The difference between an ERF and another external file is this: when an EF is integrated into the RDB and this EF contains new units, if it is an ERF, the units are created, otherwise they are not included in the integration. Thus, only an external reference file can be used to create units in the RDB. For example, in the current annual enterprise surveys (AES), the ERF are files that come from the OCEAN system, which is used to manage the sampling frames.

17. The hybrid data collection process will involve integrating successive files EF_1, ..., EF_n, thereby updating the RDB to construct successive databases RDB_1, ..., RDB_n .

18. Or at least, that is how things will be described, although it is perfectly possible that we may have a machine capable of reading several files simultaneously. Technically, this type of approach is beginning to be conceivable.

19. Finally, we will use the abbreviation **MEF** for the metadata file relating to the external file, and **MRDB** for the metadata file relating to the reference database. The MRDB will be defined at the start of the process, and at most as many MEF as EF will be constructed. A metadata file is in a sense a «key» allowing us to «understand» a data file; there is therefore one and only one for each data file.

20. *Remember that the fundamental difference between a documentation and a metadata file, is that the latter can be read and used by a program, in the sense that the data processing system, during data collection but also during tabulation, will accept the metadata (for example, during collection, acceptance of the computation formulas used to calculate one variable from others, or of information on the location of variables in the file).*

21. We would point out that even though the name may be sometimes unfamiliar, there is nothing new about metadata: in fact, it already exists. However, it cannot be said that it is described and used systematically and uniformly from one source to another. That is the whole advantage of generic software.

Content of the CHYPRES metadata

a) General and individual RDB metadata

22. These are the ones that are fairly widely found today in survey databases, especially the French annual enterprise survey database. Note that for certain surveys, there exists «dictionaries of statistical data» containing much more detailed descriptions than the one below.

23. The metadata on the RDB (MRDB file) are thus, usually:
- identifiers of variables (e.g. TURNOVER);
- names of variables (e.g. «turnover»);
- types of variables (e.g. integer, text, variable with rules, etc.);
- record layouts;
- for variables with modes, the lists of variable modes, with associated description (e.g. {0,1}, 0 = «yes», 1 = «no»);
- the way of noting the «missing» value
- the definitions of variables (optional);
- other characteristics of the variables, such as the unit (F, kF, Euro, kEuro)
- certain information not linked to the variables, such as the title of the data collection, the period associated with the data (e.g. calendar year 1997), ...;
- the list of sources used: this is a metadatum that needs to be updated as the EF are read.

24. The metadata we have just listed are general metadata, in the sense that they never relate to a given unit: they apply to the «columns» of the matrix (variable), and not to the «rows» (individual). They can also apply to the source in general. This means that the metadata file is usually small. General metadata must have default values: for example, the default name for each variable is the same as the identifier. In addition, the application must allow these metadata to be displayed, and must be able to tell the user which ones have not been entered. The user must be able to save the work at any time: this means that he has created a metadata file in a format specific to the application. Obviously, if the application is started again, the user must be able to read the saved file. The general metadata should, as far as possible, be fully defined, for all the variables in the RDB. This is because the EF will subsequently be defined *in relation to the reference RDB*.

25. *Individual metadata*, by contrast, relate not to a variable, but to a unit, or to a pair (unit, variable). In particular, this means:
- the information source (some would come from the tax source, others from social data, for example),
- whether or not it is missing,
- whether the data has been kept in its raw form, or modified by a clerk, or imputed,

- the imputation code (imputation method used, if necessary).

26. By contrast with general metadata, individual metadata are not designed to be described via the interface. They are computed, deduced: for example, the name of the information source could be added automatically, for each variable and each unit, when the EF is «integrated» into the RDB. Moreover, and this is an essential point, individual metadata *are not located in the MRDB*: they appear within the RDB. They are (almost) RDB data like any others.

b) General and individual metadata relating to the external file

27. The first difference between the EF and the RDB, is that the EF does not need to be described exhaustively: only variables that are useful for the RDB need to be explained, i.e. to be given metadata. This is because when the statistician uses an administrative source, it often contains data that are not directly useful: specifying the location, the name, etc. of such data is therefore primarily a waste of time. In other words, the EF is exogenous, and you take from it what is useful and define only its useful metadata. By contrast, the RDB is the target, so its structure is perfectly within the control of the statistician: this means that it would be illogical to describe it imperfectly.

28. We would also note that the metadata necessary to *an* EF are not exactly the same as those of *the* RDB: true, we need the variable's location, its identifier, possibly its name; but the essential thing is to be able to bring the EF into the reference context, i.e. the RDB variables, the reference modes for these variables, etc. The whole art of describing the EF will be to define tables for transferring EF variables to RDB variables, for example, a formula for computing a RDB variable from EF variables. On the other hand, it is quite possible that a large majority of the EF variables may not be documented. This does not matter at all, since the variables in question will simply not be read. A whole portion of the MEF metadata will describe how to make the *projection* from the EF onto the RDB. The others relate to the EF as such, independently of the RDB, but the description will not need to be exhaustive.

29. Broadly speaking, therefore, the MEF can be divided into two parts:

- *an intrinsic part* independent of the RDB: the same metadata as in the MRDB, plus the identifier and name of this set of metadata, the name of the source, possibly a text describing it, the period associated with the source, the location of the comment lines, but also a code representing the quality level of the source (per variable or in general) ...

- *a part linked to the RDB*: whether the EF is a «reference» file or not (ERF or not ERF) for the RDB, the tables for transferring variable modes from the EF and the RDB, the formulas for calculating RDB variables from the EF variables.

30. Finally, the individual EF metadata function is very different from the RDB ones. In the case of the RDB, they were constructed, and are *endogenous* to the process. By contrast, the individual EF metadata are *exogenous*: if they are present, so much the better, they can be integrated into the RDB; if they are absent, this will not happen. In fact, there is absolutely no control in this case, since it is an external file: the information producer supplies what he wants.

31. Let us mention a few individual metadata that may be found in an EF:

- the date the information was obtained;
- the date (or period) the information relates to: this is different from the previous date, as the date t1 of the data can be obtained relative to the date t2, or to the period [t2 , t3];
- whether the data have been adjusted or not, and more generally a quality code associated with the data. For example, in the file we will have the workforce given as a 7 character string, from column 100 to column 106, then a code in column 107 (such as 0 if the workforce information is missing, 1 if the raw value has been kept, 2 if it has received additional verification, 3 if it is considered to be of poor quality, etc.).

32. *It is important to emphasise that the individual EF metadata must be treated as data, and must therefore, like other data, themselves be the subject of general metadata*. In the example above, this

simply means specifying that there is a variable QWF, named «quality code of the workforce variable», located in column 107, with the list of modes {0,1,2,3}, etc. This quality code can also be general information in the variable, in which case it is a general metadatum. This does not stop there also being an individual quality code: for example, we «know» that the AES (annual enterprise survey) code is «right» for the main activity; however, at the same time we may «know» that the activity code of particular units in the AES is not very reliable. Obviously, this is the information that needs to be retained: individual metadata override general metadata when there is a conflict, on the basis of a standard object-oriented principle of inheritance of attributes.

V. CREATING METADATA FILES AND VIEWING THE EXTERNAL FILE

33. The metadata files are the key, the heart of the system; but they do not spring fully formed into the world: they need to be created. In particular, in principle we have no metadata on the EF. By contrast, the RDB is often documented.

34. The first operation to be carried out before any other, is to bring this documentation into a complete and structured MRDB for each variable, identifier, name, location, type, modes list if the variable is qualitative.

35. On the other hand, developing the MEF will be an iterative process, because we may have very little information about the EF. At the very least (since without this, the system cannot function), the user must provide general file structure information (e.g. my first 25 lines are comments), to eliminate dross, then provide the identifier and location for each relevant variable. There is no obligation to describe all the relevant variables immediately. In fact, the basic idea is for the user to be able to *view* the EF and use it as a source of inspiration, before integrating it into the RDB. Developing the metadata and viewing the EF go together.

36. In practice, it is to the user's advantage to approach things in this way, variable by variable: he gives the program the identifier and location, and possibly the name. Once a variable X has been located in the file (e.g. position 110 to 114, from line 50), a certain number of verifications will be made. Is X always of the same type? If it has «modes», what modes are they? If it is quantitative, does it sometimes have nonnumeric values, and if so which and how many? When it is fully quantitative, what are its minimum and maximum values, total, average? How many units have incorrect identifiers (/what variable total for those units)? How many units have the correct identifier but do not belong to the RDB (/ditto)? All in all, we ask for very elementary statistical information; the difference compared with a «traditional» use of elementary statistical procedures, is that at this stage we are working with a file that we do not yet know very well.

37. Equipped with this information, the user will be able to build up his metadata file, for the variable in question (type, modes list, quality code, link with RDB variables) or in general (quality code again, notation of the «missing» mode, etc.). At the same time, he will also be able to extract from the EF a certain number of problematic units, in particular poorly identified ones. This task can only be done by successive iterations, by trial and error. The user thus adds metadata, but can also go back, modifying or deleting some of them. It should not be forgotten that the user here has only partial knowledge of the source.

38. The outcome of this process is the completion of all the metadata required to integrate the EF into the RDB. This set of metadata must have a name; a clear distinction must be made between the name given to the set of metadata, and the name of the file in which they appear. Each data file must be associated with one, and only one, set of metadata, which will be the «key» used to read it.

39. In practice, the user may also have to read an EF that in fact comes from the same type of information source as a previously read EF. For example, let us say he has already integrated a file of VAT declarations, and therefore constructed an MEF for this purpose; later, he finds himself dealing with a file of the same type. In this case, it is likely that the file structure is essentially the same. As a result, it will be associated with virtually the same set of metadata. This means that the program must allow a metadata file to be read and copied. Which is why every set of metadata must have a name.

40. Technical remark: during the (inevitably partial) reading of the file, everything should be uploaded into the memory for obvious reasons of execution speed. This means that memory capacity will have to be large.

41. The following are some remarks on questions of viewing:

- the viewing techniques described can equally well apply to the RDB as to the EF. It should be noted that viewing, as envisaged here, is of a «statistical» type (we are interested in individual data provided that they have a major impact, and we also want to have a global view with a few general statistics);

- the fact that the data are in XML format means that viewing tools exist (we refer here to traditional, individual viewing), since the latest Internet browsers accept XML in addition to HTML. Better still, unlike HTML, XML is only concerned with the representation of data (and not how they look on the screen, their appearance); XSL language makes it possible to create the layout for the page to be viewed, to describe its appearance formally. This means one can work at the same time on both the representation and presentation of the data, which is a considerable step, making it possible amongst other things to customise the interfaces without ever touching the data.

VI. INTEGRATING THE EXTERNAL FILE INTO THE REFERENCE DATABASE

42. This operation involves updating the reference database: the only output is therefore a new RDB. If we think in terms of skills, it is very different from the previous operations: developing metadata was an expert task, to be carried out by a statistician doing in-depth work on the integration sources.

43. By contrast, integrating a file into the RDB is a production task, done by a survey clerk, or rather by a "hybrid data collection" clerk: it is a manual editing task, in which multiples sources of information have to be handled. It will therefore probably require two tools, an interface from the expert giving the metadata as «output», an interface from the survey clerk retrieving these metadata as «input», amongst other things.

44. Integrating an external file into the RDB uses several inputs

i) The data and metadata files, upstream and downstream EF, MEF, RDB, MRDB (as described above, it is possible that there may be several simultaneously)
ii) The individual modifications «file»
iii) The integration filter
iv) The EF - RDB comparison criteria

i) The data and metadata files

45. These have been extensively discussed above. The EF is never modified. The MEF has just been created by the user. The RDB has not changed since the reading of the previous EF. The MRDB has been fixed for a long time. Note that it could be possible for each EF to be put into a single format, which also includes its MEF. This is a technical solution that has become possible with the emergence of XML as the reference language for the description of data *and metadata*.

ii) The individual modifications file

46. This is necessary because there may be correctable errors in the EF, and because we wish to avoid modifying the EF for obvious reasons of consistency. The modifications are done by the user. They are of two types:

- micromodifications: the individual modification of the data is described. Micromodifications are therefore based on the trio [unit, variable, value];
- macromodifications: the program is used to describe the modifications on a set of units and variables. The structure is therefore: If *condition C on unit U* Then *update variable X of unit U with value V*.

47. How do we construct such a file? Viewing what stands out in the EF (using tools such as sorting by descending order of importance on all units that meet a certain condition, or on membership of a given sector) makes it possible to highlight problem units which might contain errors. If it is possible, one then makes a micromodification. If, by means of viewing tools, one notices a general type of error in a population category, and if this type of error is correctable in general, one makes a macromodification.

iii) The integration filter

48. Here, the aim is to describe the logical filter to be applied to data for integration: not all units, or all variables, are integrated. There must be an implicit general filter: a RDB variable cannot be integrated from the EF if it is not subject to EF - RDB transfer rules, or if the transfer rules use at least one datum not described in the MEF.

49. There is also an implicit filter at an individual level: if, for a unit U and a variable Z of the RDB, at least one of the variables required for the calculation of Z is missing (or, more generally, unusable), Z cannot be calculated for U and therefore cannot be integrated. Obviously, if a unit U has an identifier unknown in the RDB, and if the EF is not a reference EF, no variable of U can be integrated.

50. However, the user must also be able to define an explicit integration filter:

- «in the column»: list of the RDB variables one wishes to extract from the EF;

- and especially «in the row»: the filter is then, globally or for each variable, a logical formula describing the conditions to be met by the pair (variable, unit) for the corresponding value to be integrated into the RDB (for example, the user may wish to filter on the workforce and turnover segments, or not to update the principal activity if the one appearing in the EF relates to the retail sector).

iv) Comparison criteria

51. These specify what should be done, when integrating data from the EF, if the variable is already present in the RDB. Variable by variable, the aim is to say which takes priority, EF or RDB.

52. A few obvious points are worth mentioning: if the variable is missing in the EF, it does not get integrated. Otherwise, if it is missing in the RDB, and unless explicitly advised to the contrary, it can be integrated into the RDB. In addition, it may be assumed by default that if the value was obtained following micromodification, it is of better quality than the value given in the RDB. Finally, if quality codes are present in both cases, and one is clearly superior to the other, the choice is obvious.

53. Apart from this, the comparison criteria must specify what should be done if the variable is already present in the RDB. One could imagine a general criterion on the relative deviation between EF datum and RDB datum: if this deviation is less than x, where x is a parameter to be set, no modification is made. Otherwise, the choice should probably be made variable by variable and be explained in a comparison rules file.

54. To be developed, the deviations need to be viewed: sum, average, min and max of absolute (/relative) deviations, sort by descending absolute deviation specifying unit names, etc.
Note: before the EF is integrated, it may be desirable to carry out certain, very basic, checks on internal consistency. These include in particular verifying that each identifier in the database is unique: if a single identifier appears twice, there may be an error.

VII. LIMITATIONS OF THE DESCRIPTION

55. The representation adopted in CHYPRES is exclusively matricial: two dimensions, variables and statistical units, and in each case, values and possibly (individual) metadata. In fact, this representation is not sufficiently general, and in particular does not take account of different levels of units (e.g. enterprises and establishments). In addition, the question of the time variable has been raised (in fact, we have a matrix with three levels: units, variables, period), but has not been discussed in depth. Nonetheless, matricial representation does apply to the large majority of cases.

56. Even if the formal structure of the information is indeed matricial, the external file may not comply with the property «one unit per row», which happens to be a very practical property for effective viewing of the

EF. If we want a very general software application, it is essential to allow for all possible syntaxes. From this point of view, again, the fact that XML (a data description language, not a programming language) allows information of a «semi-structured» nature to be described is a considerable advantage, which will probably have a major impact on developments in database management systems. In this case, there is neither a syntax problem nor a representation problem for information which comes outside the framework of the matricial structure described above (example of information without a matricial structure: dual-level data, enterprise/ establishment).

57. Variables with a complex structure need appropriate processing: we mean essentially variables that take their values in nomenclatures (notably activity): there are numerous nomenclatures, which have no real one-to-one correspondence (which complicates the question of transfer tables); in addition, there is in this case a continuum between reply and non-reply: for example, obtaining information at division level (2 figures) in the activities nomenclature when we want information at class level (4 figures). Not to mention the difficulties associated with coding this information from a name. More generally, the rules of transfer from the EF to the RDB are likely to be fairly complex if we want to be able to process all cases effectively.

58. Only very elementary statistical calculations have been discussed, since the purpose is not to carry out an in-depth statistical analysis, or to produce statistics: to introduce more and more functions so as to develop a new statistical application would be pointless. That having been said, it is obviously essential to consider the basic functions required for the *«intelligent» integration of an external file*. These functions could include the checking of consistency between data.

59. As we have seen, the calculations often have to be carried out on *subpopulations*, working by sector, by size, by region, etc. This is why the notion of a filter has been introduced. Nonetheless, as defined this concept remains imperfect, in that the filter only applies to EF variables which, as we have rightly said, can contain many errors. Strictly speaking, we need to be able to extend this logical filter to information from other databases (notably the RDB), which raises problems of performance, and brings us back to the difficulty posed by «joins» in database management systems.

60. In terms of space, storing all individual metadata can be very space intensive. Does it need to be done in all cases? There are other IT problems, linked to the structure of data in existing information systems, the difficulty of transferring data (this is, in fact, a problem to be considered with XML, since with a language like this, «tags» are added to each piece of data which take up a lot of space etc.). These technical problems are a very tricky question in themselves, which were not examined in CHYPRES, and which cannot be avoided if we one day wish to implement more widely the general principles discussed here. IT difficulties could even threaten the viability of certain approaches.

61. All these remarks prompt us to consider what the outlines of a general external file integration application should look like in a production environment. By trying to cover too many questions at the same time, we run the risk of constructing a hodgepodge application, with a few sophisticated functions designed simply to handle one or two rare special cases. In practice, we have considered the question of the minimum core functions that should go into the construction of a prototype.
