Distr.
GENERAL

Working Paper No.8
1 April 2008

ENGLISH ONLY

UNITED NATIONS STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION STATISTICAL OFFICE OF THE EUROPEAN COMMUNITIES (EUROSTAT)

ORGANISATION FOR ECONOMIC COOPERATION AND DEVELOPMENT (OECD) STATISTICS DIRECTORATE

Meeting on the Management of Statistical Information Systems (MSIS 2008)
(Luxembourg, 7-9 April 2008)

Topic (ii): Statistical information systems architecture

A FRAMEWORK FOR SDMX

Invited Paper

Prepared by Dario Camol and Laura Vignola, ISTAT, Italy

I. INTRODUCTION

1. The idea to offer a single dissemination point towards the international organizations was developed from the Istat short-term statistical database. This approach evolved over the last two years. The first period was dedicated to satisfying the requirements of the Eurostat SODI project. Later on, Eurostat launched other projects in which Istat was involved in both the "Demography Pilot Project" and "Census Hub Pilot Project". So the idea to design a new version of the Istat SDMX framework (version 2.0), released from a "proprietary" database schema took place. The evolution of the SDMX Istat Framework is based completely on SDMX standard format and architecture. It allows organizing the disseminating data according to Data Structure Definitions defined for a specific statistical Domain. The solution is based on a registry/repository accessible from a single web service. The version 2.0 is composed of a set of re-usable modules that guides the data flow from the reporting to dissemination. But some of these modules can be used independently of each other. The version 2.1 will render available most of those modules as API, so that they can be integrated from different systems.

II. ARCHITECTURE

A. The framework

2. The framework is composed of an application Client- Server realized in Visual basic .net with a database access (but it can access also other kinds of db server) and a web services realized in asp .net who access to the same database. The idea is to realize a repository SDMX compatible in which is possible to store data (actually the data input file can be in text format or in Gesmes STS format).

3. It can be accessed by a web services who take a SDMX query to retrieve data in SDMX compact or Cross Sectional format. A schema of the model is showed below (fig. 1).

Load of structure

4. The files, containing the Data Structure Definitions, are loaded into the system and all the information contained in these files is stored in specific tables created in the first loading and updated in every future loading if necessary. In this manner the code list structures, all their values and the DSD with all the attributes are stored in the database. In this way also all the information necessary to realize the two different formats (compact or cross sectional) of SDMX data file are stored.

5. Once created these tables will be the same for every data structure loaded. After the storage of the structure the user creates metadata table and one or more dataflow tables to contain the data, this must be done every time it is necessary to loads a new data structure or a new dataflow. These tables are specific for every DSD or dataflow loaded.

6. At the creation stage of the metadata table the user could decide if to insert a group of optional fields (title, obs_prebreak…etc), it depends if this optional information are or not present into the data file.
In the table called *Metadati* there will be stored all the information that concern the levels hierarchically superior to the observation level, in the tables of dataflow, instead, the information that is possible to find at the observation level.

Load of data

7. At this point the data file is loaded. Actually two formats can be loaded on the system: text format with separator fields and Gesmes format for STS.

8. The system with a WYSIWYG wizard follows the user during the importing process. After the selection of the dataset and the related DSD the user can choose the file to import. In this phase a mapping is needed between the fields present into the metadata and dataflow tables and the fields present into the text or Gesmes data file.

9. Through the visual aid of a grid and a selection box every field of the file must be linked to the related one presents in the dataflow table. This is the last step before the creation of the sdmx file and the real storage of the data.

10. If the file contains a series of values for update on the database the data will be overwritten.
Having all the data and metadata stored in its database the web service is ready to receive the sdmx query and send to the client application the specific Compact or Cross Sectional file.

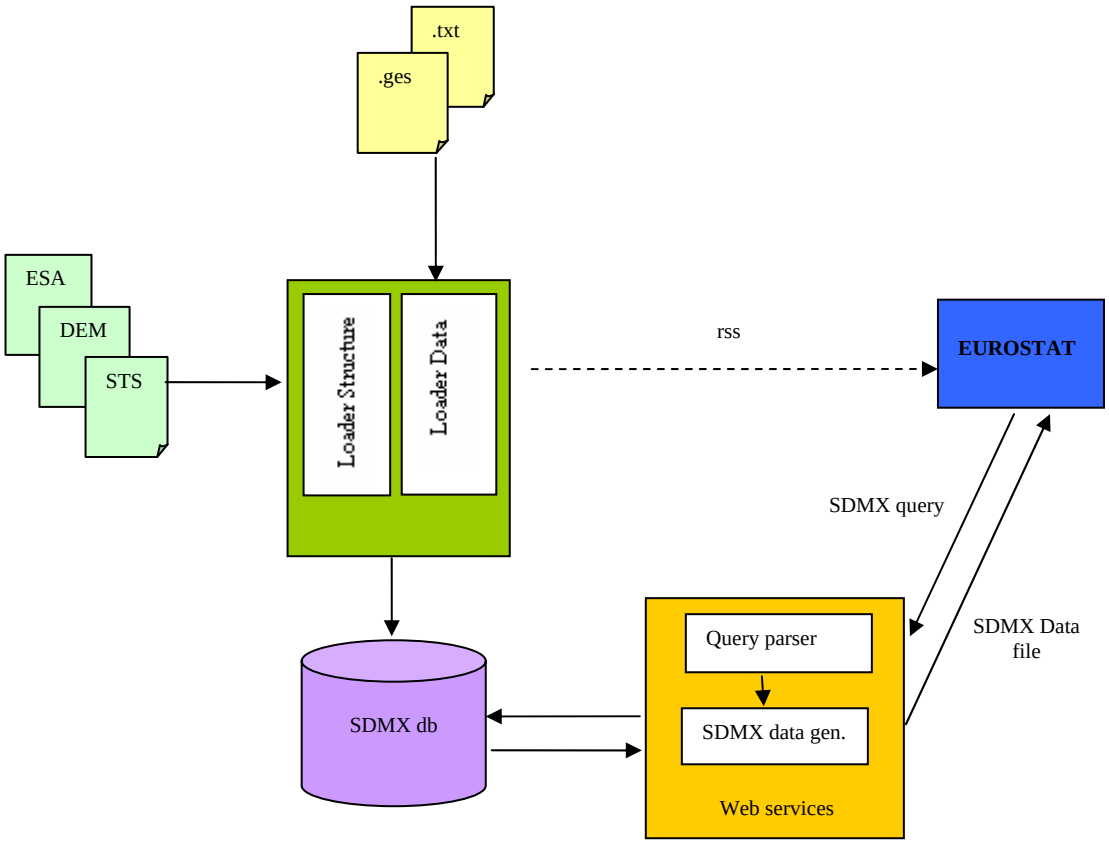

Figure 1.

B. The database

11. The database is initially empty and its tables are created and filled at runtime from the system when it imports the file. The first file needed is an xml file containing a link between dataflow and DSD. Importing this file the system will create the table Data_Flows.

12. The SDMX Structure files imported need to create the metadata tables (fig. 2). After the creation and the loading of this table the user must create a metadata table for each DSD loaded and one or more tables for data (the name of this table is represented by the dataflow name). (fig. 3)

13. During the creation of these tables the user could decide to include some fields for the optional attributes that are part of the DSD. This operation is more important because the consistency of the SDMX compact or cross sectional file (without attributes with null value) depends from it. The fields that are inserted in these tables must have a linked field in the imported data file.

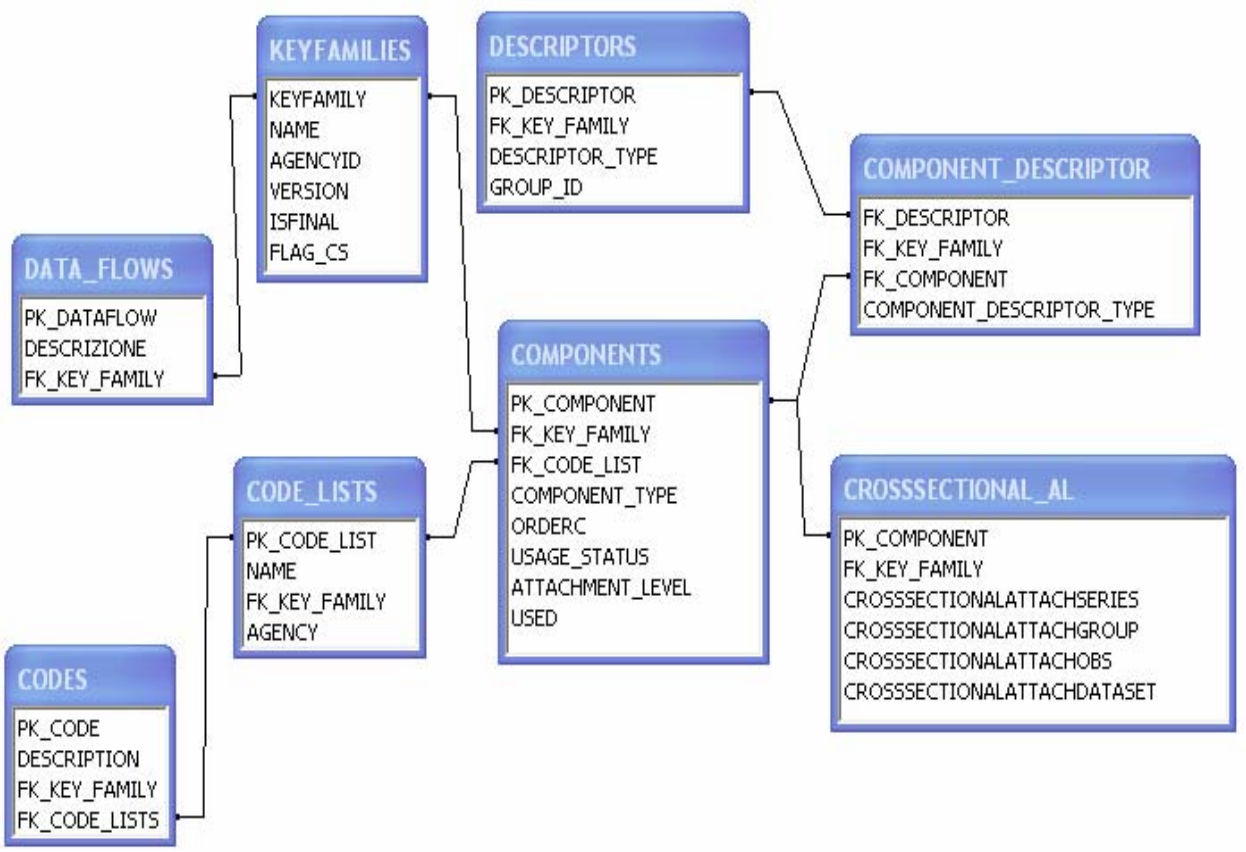
KEYFAMILIES
KEYFAMILY
NAME
AGENCYID
VERSION
ISFINAL
FLAG_CS
DESCRIPTORS
PK_DESCRIPTOR
FK_KEY_FAMILY
DESCRIPTOR_TYPE
GROUP_ID
COMPONENT_DESCRIPTOR
FK_DESCRIPTOR
FK_KEY_FAMILY
FK_COMPONENT
COMPONENT_DESCRIPTOR_TYPE
DATA_FLOWS
PK_DATAFLOW
DESCRIZIONE
FK_KEY_FAMILY
COMPONENTS
PK_COMPONENT
FK_KEY_FAMILY
FK_CODE_LIST
COMPONENT_TYPE
ORDERC
USAGE_STATUS
ATTACHMENT_LEVEL
USED
CODE_LISTS
PK_CODE_LIST
NAME
FK_KEY_FAMILY
AGENCY
CROSSSECTIONAL_AL
PK_COMPONENT
FK_KEY_FAMILY
CROSSSECTIONALATTACHSERIES
CROSSSECTIONALATTACHGROUP
CROSSSECTIONALATTACHOBS
CROSSSECTIONALATTACHDATASET
CODES
PK_CODE
DESCRIPTION
FK_KEY_FAMILY
FK_CODE_LISTS

Figure 2

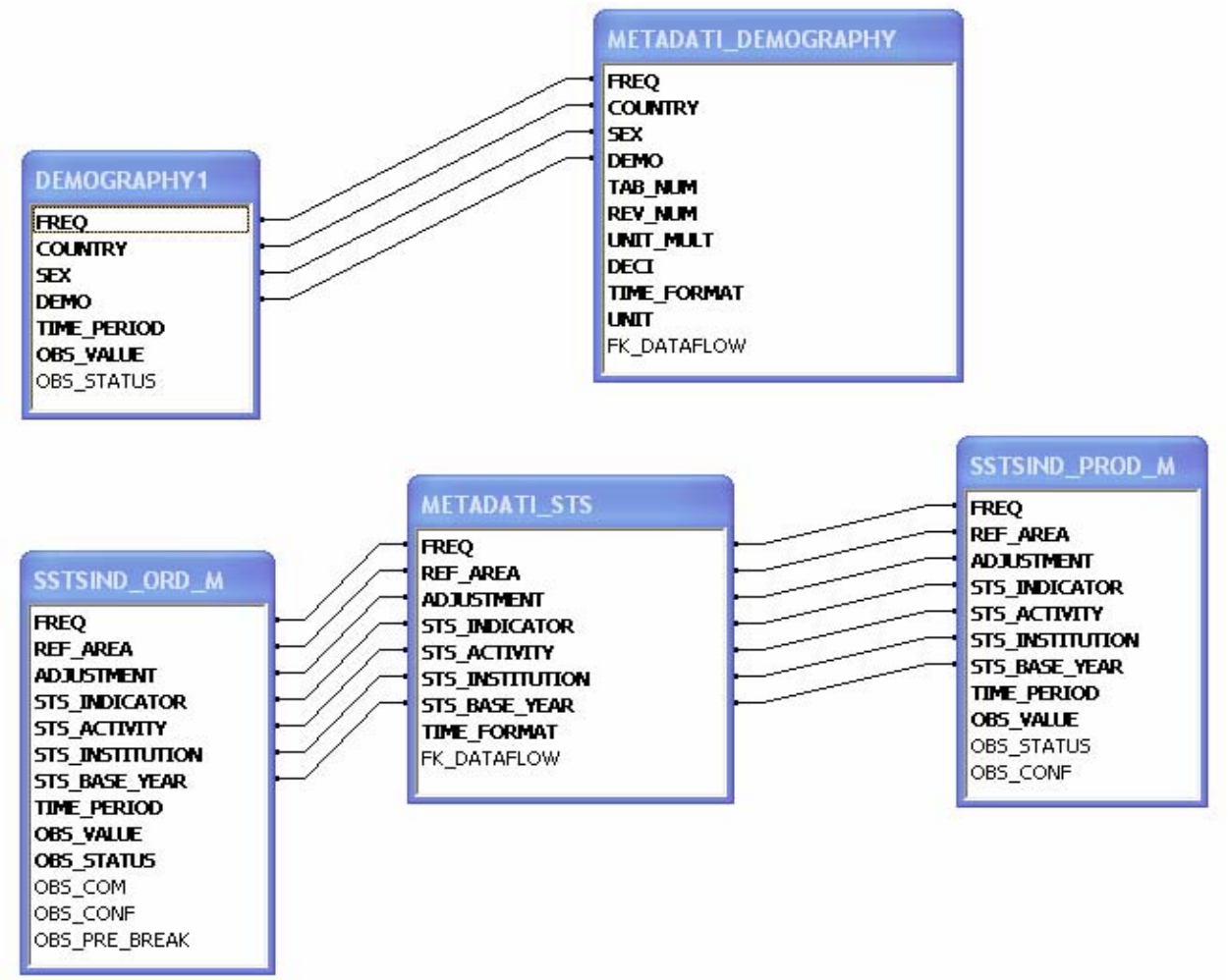

Figure 3. The case of STS and Demography

C. The web services

14. The web services can be considered composed of two parts. The first, not linked to the database, takes a sdmxquery as input and making a parsing of this query, generates a table containing all the required variables for each row. The rows have to be considered in "AND" relation and columns in "OR". From this particular structure the second part of the web services can create a sql query to retrieve data from the database with the structure showed above. Now we focus on the two modules and their principal characteristics.

The query parser

15. This module does not access to the database and elaborates the imported xml file giving as result a grid that has got, in its rows, a combination of values (columns) representing the DataWhere of the smdxquery. The first step required by Eurostat was to analyze a kind of SDMX query referred only to one dataset, the second step was to analyze a more complex query to retrieve data. A SDMX query is an xml file structured following the SDMX rules. We are interested by the DataWhere query. In this kind of query we can find these nodes:

- *Time*
- *And*
- *Or*
- Dimension
- Attribute

- DataFlow

16. We can define the nodes written in italics as “complex nodes” and the others ones “simple nodes”. The simple nodes contains only the value while the complex nodes can contains also other nodes, in particular the node “Time” contains only two child nodes “StartTime” and “EndTime” or only a node “Time”, while the node “And” and “Or” can contain all the nodes related above. What are the dimensions/attribute we can find inside a Dimension node? They are all the dimension/attribute referred to the required dataflow, as we can find in the dataflow table (see paragraph B).

17. The query parser proceeds in two steps:

(i) Rewriting of the query

In this step we worked with the SQL language, making the transformation of the file, to have a simpler structure. The structures related below are translated in other structures:
- If we have an “And” node with “And” child nodes inside, we can remove all the child nodes and put theirs conditions into the parent “And” node

Ex.
<DataWhere>
<And>
Cond 1
Cond 2
<And>Cond 4, Cond 5**</And>**
<And>Cond 6, Cond 7**</And>**
<Or>Cond 8, Cond 9**</Or>**
</And>
</DataWhere>

the new structure become

<And>
Cond 1
Cond 2
Cond 4
Cond 5
Cond 6
Cond 7
<Or>Cond 8, Cond 9**</Or>**
</And>

In the same way we can do for the “Or” node.
- If we have an “Or” node with “Or” child nodes inside, we can remove all the child nodes and put theirs conditions into the parent “Or” node
- The Time node may be viewed as an “And” node because the child nodes (EndTime, StartTime) are into the where clause, in an “And” condition. If the node time contains only a node time child (see the demografy query structure) we can change this child node into tow childs StartTime and EndTime with the same date and return to a precedent situation.

Ex.
<DataWhere>
<And>
Cond 1
Cond 2
<Time>
<StartTime>Date 1**</StartTime>**
<EndTime>Date 2**</EndTime>**

```
            </Time>
    </And>
</DataWhere>
```

the new structure become

```
<DataWhere>
    <And>
            Cond 1
            Cond 2
            <And>
                    <StartTime>Date 1</StartTime>
                    <EndTime>Date 2</EndTime>
            </And>
    </And>
</DataWhere>
```

- An “And” node with the “Or” child nodes can be translated as an “Or” node with “And” child nodes each of all have as child the cartesian product of all child node of the “Or” child nodes

Ex.
If we have this SDMX query fragment of file

```
<DataWhere>
    <And>
            Cond 1
            Cond 2
            <Or> Cond 3, Cond 4</Or>
            <Or> Cond 5, Cond 6</Or>

    </And>
</DataWhere>
```

The new SDMX query fragment of file may be written as:

```
<DataWhere>
    <Or>
            <And> Cond 1, Cond 2, Cond 3, Cond 5 </And>
            <And> Cond 1, Cond 2, Cond 4, Cond 5 </And>
            <And> Cond 1, Cond 2, Cond 3, Cond 6 </And>
            <And> Cond 1, Cond 2, Cond 4, Cond 6 </And>
    </Or>
</DataWhere>
```

At the end of this transformation at each node we put an identifying ID number and we create a recordset in this way: the columns of the recordset are the concept (dimension or attribute) of a specific dataflow, all the childs in an “And” node belong to the same recordset while the child in an “Or” node belong to different records.
Two columns of the records are the id value of a single node alias “pkmappa” and the id value of the parent node as “fkmappa”, so we have all the recordset linked by a parent-child relation.

(ii) Analyzing of the recordset

In this phase we start by the first record and we attach to itself the other record proceding in sequence by the value of fkmappa, when we arrived at the last record disposable we have a new record memorized in a new recordset structure and we restarted from the last value of fkmappa who has other record child disposable.

At the end, the recordset generated contains a query for each record in which all value of the columns not null are in a “And” relation.
This is an interesting result because we have reduced a complex query into a set of simple query, so we can use it for every database structure.

The SDMX Data file generator

18. From the result of the previously described module, depending on the structure of the database from which data must be taken, is easy to reconstruct a sql query. In our case the output grid given by the previous module is subdivided in more grids, one for each dataflow. Each row in this grid will represent the clause *where* of a *select* sql-instruction that must be used on Metadata e Data table. These tables have in common a primary key, represented by the DSD dimensions, so the *select* , previously mentioned, will act on both the tables, using the join relation on the key fields.

19. To understand better this concepts an example will be shown. If the web service receives the query below:

```
<Query>
    <query:DataWhere>
        <query:And>
            <query:DataFlow>STSIND_PROD_M</query:DataFlow>
            <query:Dimension name="FREQ">M</query:Dimension>
            <query:Dimension name="INDICATOR">PROD</query:Dimension>
            <query:Or>
                <query:And>
                    <query:Dimension name="ADJUSTMENT">S</query:Dimension>
                    <query:Or>
                        <query:Dimension
                            name="STS_ACTIVITY">N100DA</query:Dimension>
                        <query:Dimension
                            name="STS_ACTIVITY">N100C0</query:Dimension>
                        <query:Dimension
                            name="STS_ACTIVITY">N100D0</query:Dimension>
                    </query:Or>
                    <query:Time>
                        <query:StartTime>2000-01</query:StartTime>
                        <query:EndTime>2006-01</query:EndTime>
                    </query:Time>
                </query:And>
                <query:And>
                    <query:Dimension name="STS_ACTIVITY">N100C0</query:Dimension>
                    <query:Time>
                        <query:StartTime>1995-01</query:StartTime>
                        <query:EndTime>2003-01</query:EndTime>
                    </query:Time>
                </query:And>
            </query:Or>
        </query:And>
    </query:DataWhere>
</Query>
```

The grid showed is the result of the first part of the query parser.
In the imported query are required the data of only a dataflow: monthly industrial production.
Each row represents a different condition *where* : for example the first could be translated as a *where* of this type “ Dataflow=’SSTS_IND_PROD and adjustment=’S’ and frequency=’M’ and indicator=’PROD and sts_activity=’N100DA’ and time between 199001 and 200601”.
Each row is in “Or” relation with the others.

StartTime	EndTime	DataFlow	Adjustment	Freq	Indicator	Sts_activity
1990-01	2006-01	SSTSIND_PROD_M	S	M	PROD	N100DA
1990-01	2006-01	SSTSIND_PROD_M	S	M	PROD	N100D0
1990-01	2006-01	SSTSIND_PROD_M	S	M	PROD	N100C0
1995-03	2005-02	SSTSIND_PROD_M		M	PROD	N100C0

20. As we can see, in the third and fourth row, the ateco N100C0 and any adjustment in two different time periods are required, so the ateco N100C0 and adjustment S, is required two times in the overlapping time period. To give a solution to this problem the queries are bound in a *union* relation, so the database will eliminate the duplicates. Sending query to the database gives as result a compact or cross sectional sdmx file. The choice of the format is linked to the DSD stored into the database.

D. Future development

21. The future developer regards two different aspects, the introduction of new functionality like the load of other kind of data file, the creation of rss file during the load of data, the introduction into the query parser of the interpretation of other tags and the realization of some parts of this framework as API to make reusable by others.
