ETLA ELINKEINOELÄMÄN TUTKIMUSLAITOS
THE RESEARCH INSTITUTE OF THE FINNISH ECONOMY
Lönnrotinkatu 4 B 00120 Helsinki Finland Tel. 601322 Telefax 601753

Keskusteluaiheita - Discussion papers

No. 351

Mari Harni - Jukka Lassila - Heikki Vajanne

TRANSFORMATION AND GRAPHICS IN ETLAs ECONOMIC DATABASE SYSTEM

ISSN 0781-6847 25.01.1991

HARNI, Mari - **LASSILA,** Jukka - **VAJANNE,** Heikki, **TRANSFORMATION AND GRAPHICS IN ETLAs ECONOMIC DATABASE SYSTEM.** Helsinki : ETLA, Elinkeinoelämän Tutkimuslaitos, The Research Institute of the Finnish Economy, 1991. 12 p. (Keskusteluaiheita, Discussion Papers, ISSN 0781-6847; no. 351).

ABSTRACT: Transformation and graphic software, developed in ETLA, takes data from relational databases. The user can combine data from multiple sources and make transformations without worrying about possible undefined or missing values. The software also produces excellent quality business graphics in unix environment. Uniquely (in unix), the graphs are directly transformable to VideoShow system.

KEY WORDS: temporal data, data joins, merging

Transformation and Graphics in ETLAs Economic Database System

Table of Contents

1. Introduction

The growing masses of economic data necessitated ETLA (The Research Institute of the Finnish Economy) to start building a database in 1985. The base, now commercially available on-line, is relational and built on software called Mimer, with a Swedish origin. The information system is a mixture of ETLA's own software products and commercial packages.

Graphic representation of forecasts has been one of the strongholds of ETLA during the two decades of her macroeconomic forecasting activity. Presently, the graphics are VideoShow-based and fully integrated with a relational economic database.

Although relational database software offers many advantages to time-series data, practical applications are scarce. ETLA has put much effort in software development, to achieve a situation where the various and diverse needs of researchers and forecasters can be satisfied with the best available software and still benefit from the advantages of a centralized data base which is specially tailored to economic data and forecasts. Using a single integrated software would mean that some researchers could not use the best tools for their problems. Allowing many kinds of individual software choices leads, on the other hand, to communication problems and increased learning and training costs, but this was still considered the better alternative. ETLA's own software development has been directed to efficient linking of commercial software products. The result is an open unix environment, where ETLA's products are integrated with the most advanced micro software, among others VideoShow.

2. Time Series Data and SQL

Our starting point was a commercial DB program and, on the other hand, a large amount of economic time series data, which were ordered or grouped by tables. Most of the tables resemble a statistical publication so that each table is updated from only one statistical source. The original idea was that the final user is an economist who wants to use not only the raw data but also a large set of derived entities such as percentual changes, moving averages and so on. Furthermore, the aim of the DB project was and is to deliver the data to outside users as well as the researchers in our institute. The base consists of several hundreds of tables and about 25000 columns as of 1.6.1990 and is becoming bigger all the time.

At first sight the relational data model offers clear and well suited tools to deal with temporal data sets. One organizes the data into tables, which resemble the statistical sources. The columns consist of economic activities and among the key columns the most important is the set of time points when each economic activity was deemed to take place. Also it is easy to present data cubes in a tabular form by using two key columns, etc.

In any case the query lanquages are in trial use while considering the problem of combining data from multiple sources (banks, tables). Although they might perform the task in the case of time series data, the SQL expressions may become awkward. This is not to say that we should forget relational database altogether especially because currently almost all commercial data base products make use of the concept.

Now, there are several lines to follow. One is to rely totally upon the SQL. But even in that case one must invest a lot in the user interface because many operations on multiple tables become quite long to present while using the SQL in a time series context. Secondly we could purchase a system, which already is planned to cope with the temporal data. One way to break the limitations of the SQL has been to develop an extended SQL, which we have seen in the engineering applications (Jones, 1989). An important extension is to offer a proper set of functions in queries involving calculated values.

The third line is to make one's own software to support the existing relational data base system. We have chosen this line. The main reasons are the problem of missing values, the derived entities, the effiency and the graphical presentations. The name of the program is Table.

3. Missing Values and Derived Entities

In the context of scientific data bases one often stresses upon that it is important to consider the problem of undefined or missing values carefully (e.g. Wittkowski, 1988) and in this we concur. However, there are rather few undefined values in our base. The reason is simple: the statistical sources behind the tables have few undefined values. In our case the undefined values appear while joining multiple tables, e.g. considering the time as the common key.

Let us have three tables A, B and C, say and we assume that the tables are defined for the years as follows:

table	time span
A	1900 - 1950 ,
B	1950 - 2000 ,
C	1900 - 2000 .

The name of time column is OBS, which is unique over the tables, and we have a column STATUS. If its value is 'X' we have final or latest figures else preliminary ones. The traditional join operation is easily accomplished in SQL as follows (e.g. Ageloff, 1988):

```
select * from A,B,C where A.OBS = B.OBS and B.OBS = C.OBS; .
```

It is frustrating to incorporate the conditions concerning the STATUS columns, because one thing is certain, we are going to have only one row, which is for the year 1950. Of course we could define each table for this century and fill undefined values with a special code and then use the join operation. However, we feel that although some DB systems have efficient ways to store or record undefined values, the bookkeeping activities in the whole system would be too heavy (at least after 2000).

The most common way to perform the joining in time series systems is to define union of the final figures. It means that we want a table, where the years run from 1900 to 2000. Once again we can easily define in SQL:

```
select OBS from A where STATUS = 'X'
union
( select OBS from B where STATUS = 'X' );
```

We do not bother to incorporate the table C (and D,E,F,...) because we have too many problems to cope with. When we define union, we must have identical sets of elements both in the top select and in the subselect. Actually we want a new table on the fly. The key column of this new table is the union of the respective OBS columns in A,B and C. Furthermore the table consists of all the columns in A,B and C. Surely it is possible to join the tables in SQL or in embedded SQL but it seems to us that we are now trying to shoot a fly with an elephant gun. It seems to us that the joining methods of SQL are heavy procedures to carry out a simple match.

Nevertheless the relational data systems offer a rich arsenal and we take the advantage of it while handling single tables. The joining and derived entities are made by our own software such that the system appears to the user as a single program package.

In Table one may perform the task above as follows:

```
getdbs: SUHDANNE  #define the data bank
              OBS, STATUS, A1, A2 from A  #select from table A
                where STATUS = 'X';
              OBS, STATUS, B1, B2 from B  #select from table B
               where STATUS = 'X';
              OBS, STATUS, C1, C2 from C  #select from table C
               where STATUS = 'X'; ;;

 /* end of the retrieval, now the data is in a binary work file. It contains besides the actual data also the ex-
 planations of each table and column and the printing masks of every column. One may now load the data into
 the work space in various ways. We want to match the data sets using the first main key (OBS) of the tables
 A,B,C so we say only:   */

 mergeload: ;
```

As the outcome we have a table, the column of which are the augmented columns of the tables A, B and C. Currently the upper limit for the column number of a single table in Table is 1024. Furthermore we have the tables A, B and C. Anyway the combined table has as its key column a string vector (OBS by default). Its values are very near to an ordered union of the three key vectors of A, B and C, that is, observations are in the order determined by the lexical order of the combined key column.

We may take the the table to DOS/Macintosh as a Symphony/Excel worksheet by the command:

```
symphony:('.)  /* The notation inside the parentheses is an abbreviation for the current table  very much like
a dot is a shorthand notation for the current directory in Unix. Besides the command above one may define the
'coordinate' information on how to fill the worksheet.          */
```

Or if we want to get rid of the original tables A, B and C, we invoke a new session as follows:

```
spa:          /* Save the parameters of the current session for the new one, e.g. those  concerning the
graphics. It is possible to take all  graphical elements created in this session.          */
```

```
newstart:('.)  /* Drop all tables except the current one. */
ppr('.)     /* Now, we have a new session, we take a printout of the data of the current table folded both by
rows and also by columns if needed. */
```

In our tests the joining operations in Table are much faster than those of SQL in relational data base systems. Furthermore, the sorting algorithms in our system are 'stable' (Knuth, 1973), which has proven to be a useful feature while reordering the data in the case the sort key contains equal values.

Although our system offers greater flexibility while handling temporal data than the systems based on the standard SQL, there are also disadvantages. The most important one is that the maximum number of observations one processes in Table, must be known in advance. In our version it is currently defined to be at most 10,000 observations. However one may reconfigurate the number quite easily. The current number is not a severe restriction in our system, because in our base while there are plenty of columns the number of rows are still far from the current limit.

The register areas of vectoral calculations and sorting algorithms depend upon this figure. But on the other hand Table offers an almost unique way to accomplish the calculations in the sense that one may refer to vectoral statements in a way we have usually seen while handling scalar entities. Furthermore, Table keeps track of the missing values such that, in short, if the right-hand side of the expression contains an encoded missing value, so will the left-hand side as well. There are over 200 hundred built-in functions and operations in Table, and a large set of scripts available.

Furthermore, there is a link to the unix program Awk (a text pattern scanning and processing language) such that one may pass elements of Table's work space to Awk, perform the Awk-script and to get the outcome back to Table's work space. So Awk is one of the Unix products, which is a subprocedure of Table, like the stream editor Sed or the 'grep' functions.

4. Data Base and Graphics

A standard procedure to handle data is to retrieve the information from the base using menus or SQL-like tools e.g. to a Symphony worksheet and then move these files to local workstations like a Dos or Macintosh environment. The user then continues to refine the data for graphical presentations, reports etc. Besides this option we made graphical software for use on the HP computers. The early versions relied heavily upon the HP's graphical Starbase library. Nowadays the Starbase routines control only HP screens and the other screen, laser and VideoShow outputs are independent of Starbase. However Starbase contains some nice concepts, which were preserved in all versions of our software. One of them is the use of the world coordinates. We may define all graphical elements including texts using the coordinates of the user's application. Futhermore scaling the picture anew was made so easy that pictures which usually are drawn as separete elements on the micro software, can without any trouble be considered as a one unity in our software. Because the Starbase contains scalable fonts, the HP-screen is an accurate previewing device for VideoShow pictures.

The personal computers have quite advanced graphical tools. Furthermore, the general attitude is to avoid stressing the main computers with graphics. On the other hand , we feel that graphical presentations are as legal and important members of the database as are the original data, although we have not yet implemented the graphical elements as objects into the base.

Once again one must build a software application so that graphical presentations are possible. Our graphics belongs to the class of business graphics, however, containing capabilities for high-level graphical presentations.

Our graphics system creates picture files in the VideoShow form. We used the Software Developer's Kit to produce the NAPLPS code (ANSI, 1983). It has useful examples and it also describes NAPLPS primitives and extensions to NAPLPS supported by VideoShow Operating System.

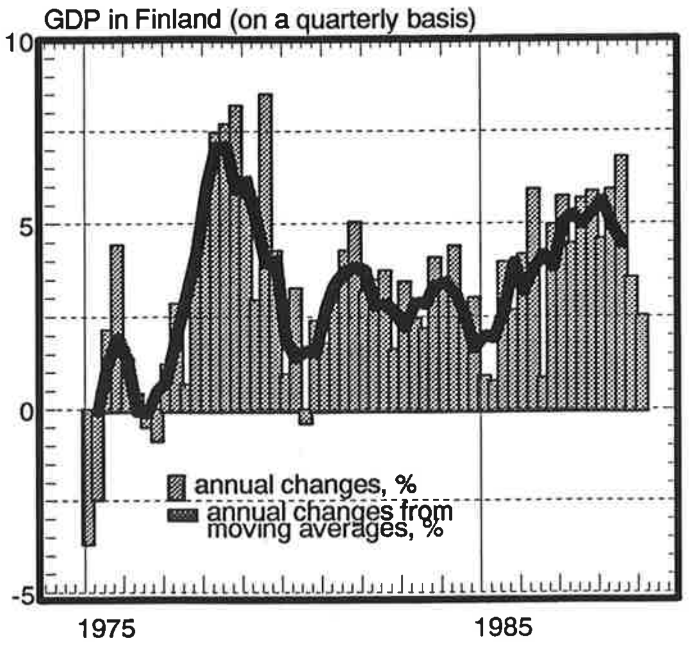
The system contains predefined graphical objects, over 80 at the moment. It is also possible to define one's own objects. In what follows we have a sample of a Table script, which contains data base retrieval and some tranformations to be drawn and finally the picture is shown in the VideoShow system:

```
 /* Get the Data from the Base : */
getdbs: SUHDANNE             #Name of the Bank
OBS, QQ from NKTHTQ    #The GDP in Finland
where STATUS = 'X';  #The Final Figures only
;;
/* End of Retrieval, now we load the data into the Work Space */
load:
/* Now for the sake of clarity we define the following entities: */
'OPEN_VIDEO_SHOW = -2;
'YES = 1;
'NO_ROUNDING_OF_CORNERS = 0;
'SOLID = 0; 'RATHER_THICK_LINE = 1.5;
'A_THICK_EDGE = 6; 'A_THIN_EDGE = 3;  'OUTER_EDGE = 'YES;
'LAVENDER = 6500 ; 'AQUA = 8720; 'BLACK = 1000;
'TAKE_INTO_PC = 'SHOW_IN_VIDEO_SHOW = 'YES;
'DON'T_SHOW_IN_VGA = 0;
/* Open The Graphical line to The VideoShow */
view: 'OPEN_VIDEO_SHOW
Dithering = 'YES;

     make("%text,0,102,"GDP in Finland (on a quarterly basis)");
     make("S_intr2", 'SOLID, 'OUTER_EDGE    , 'LAVENDER);
     make("S_edge2", 'SOLID, 'A_THIN_EDGE   , 'BLACK);
     make("Abars_2", startpoint(OBS), 0, endpoint(OBS), %(movavg(QQ,4),4));
     make("Lgbar", 20,18,2,4,"Annual changes, %");
     make("S_tape1", 'NO_ROUNDING_OF_CORNERS , 'RATHER_THICK_LINE);
     make("S_intr1", 'SOLID, 'OUTER_EDGE    , 'AQUA );
     make("S_edge1", 'SOLID, 'A_THICK_EDGE  , 'BLACK);
     make("Atape_1", midpoint(OBS), %(QQ,4))
     make("Lgtape", 20,13,4,1.5,"Annual changes from\nmoving averages, %");
     corners(); linyscale(); calxscale();
     draw()
     video('TAKE_INTO_PC, 'DON'T_SHOW_IN_VGA, 'SHOW_IN_VIDEO_SHOW);
/* END OF THE SCRIPT */
```

Figure 1.

This picture was imported into Ventura in the VideoShow form, however no extensions to NAPLPS were used.



The matching problem while representing graphically time series data is not difficult to cope with, if the entities are all in the base. We have the calendar values (see OBS above) and the calendar function, which determines the decimal values of the time points. However, if we are going to present transformations of the base data, especially when the constituents come from different sources (data banks, tables) we must have facilities to match the data before making calculations and drawing the transformations.

The following is a short list of the capabilities of the graphical software:

- In our version the maximum number of observations in the work space is defined to be 10,000. So we may define graphical elements of that size. For example, we may have a bar element consisting of 10,000 single bars. However, the number of vertices in polygons is device dependent.

- In one figure we may define 512 graphical objects, each of them containing 10,000 coordinate values or less. Among those there are different kind of line elements, pies, circles, arrows, markers, text elements, text files, bars, ellipses, tic marks, labels etc. The number is not a severe restriction, because one may delete the elements already drawn, after which we may define another set of 512 elements. Furthermore, it is possible to define elements of one's own. Table can also draw very thick lines in a device independent way. In VideoShow this feature is not an important one, because there are ready made capabilities for those kind of elements.

- At any stage while building the picture in the host, one may preview the figure on the PC or in the television set and get back to the point one left.

- In VideoShow pictures we may use 1000 different colors.

- It is possible to scale the picture automatically as well. One may scale the picture automatically according to all graphical elements or according to a certain subset elements only.

Furthermore it is possible to scale the x- or y-axis or both. At any stage you may rescale the picture anew, so that entities with totally different scales are easily presented in the same picture. It is also easy to define within one picture different effective drawing areas, which are rectangular subsets of the graphical output device. On the HP-screen and laser printer one can also define clip indicators to these subsets.

- The data of graphical elements are processed so that they burden the main memory as little as possible, but they are processed in an efficient way. (The software makes use of Table's binary work file.)
- Although the original entities in the base do not contain many undefined values, our experiences tell that after joining multiple tables one has generally a lot of these undefined values. Table's graphics automatically passes all those items, so that they are not a nuicance while drawing.
- One may locate the texts using the world coordinates. Furthermore, you can orientate the texts to the left, right or define them as centered. A text element contains at most 10,000 rows, each a maximum of 256 characters long. You may break those limits by using text files as input. Text fonts are scalable.
- You may easily define logarithmic scales using any base number, which is sometimes important in scientific applications (Cleveland, 1985).
- You may use the mouse or graphical arrow keys to get the coordinate information from the HP screen. This is performed by the ginxy()- function.
- It is possible to define special effects, like pauses, prompts, etc. into the VideoShow presentations. Furthermore, you may define a set of figures with the table of contents, thus yielding a ready made package for VideoShow presentations.
- After we have produced the pictures on the host side, the products of the General Parametric Corporation allow us to to add various kind of graphical objects to the same picture, objects, which have been created with some PC- graphics product on the PC side.

5. Preliminary figures and forecasts

Usually newest economic figures published are preliminary. In our database we don't delete any earlier data in databank when new revised data is saved.

The whole history of an observation is sometimes useful for research work. The reliability of preliminary figures can be controlled and it is possible to judge the uncertainty in newer preliminary figures. It is also possible to ask what was known about subjects in a specific point of time (Lassila, 1990). In the next paragraphs the solution to the last question is described.

Let us say we want to know ETLA's latest forecast and final figures for GDP in purchasers' values (volume, annual change %) when date was 4.12.89. The forecasts are in the table ETLHTQ_P and final figure are are in the table HTASEQ_P. We consider the figures from 1987.

We also need the table STATINFO, where we have recorded the sources of each observation and the date they were issued.

ETLHTQ_P, forecasts

OBS	SID	STATUS	QQ_P
. . .			
Y:1987	ETLA8605	1	3.00
Y:1987	ETLA8611	3	2.50
Y:1987	ETLA8703	5	3.00
Y:1987	ETLA8706	6	3.00
Y:1987	ETLA8709	7	3.50
Y:1987	ETLA8712	8	3.50
Y:1988	ETLA8703	1	2.00
Y:1988	ETLA8706	2	2.00
Y:1988	ETLA8709	3	2.50
Y:1988	ETLA8712	4	2.50
Y:1988	ETLA8803	5	3.00
Y:1988	ETLA8806	6	3.00
Y:1988	ETLA8809	7	4.00
Y:1988	ETLA8812	8	4.00
Y:1989	ETLA8803	1	2.00
Y:1989	ETLA8806	2	2.00
Y:1989	ETLA8809	3	2.50
Y:1989	ETLA8812	4	3.00
Y:1989	ETLA8903	5	4.00
Y:1989	ETLA8906	6	3.50
Y:1989	ETLA8909	7	4.00
Y:1989	ETLA8912	8	4.50
Y:1990	ETLA8903	1	3.00
Y:1990	ETLA8906	2	2.50
Y:1990	ETLA8909	3	2.50
Y:1990	ETLA8912	4	2.50
Y:1990	ETLA9003	5	3.00
Y:1990	ETLA9006	6	3.00
Y:1990	ETLA9009	7	2.50
Y:1991	ETLA9003	1	1.50
Y:1991	ETLA9006	2	1.50
Y:1991	ETLA9009	3	1.00
. . .			

STATINFO, date

SID	SIDDATE
. . .	
ETLA8605	D:1986/05/15
ETLA8611	D:1986/11/15
ETLA8703	D:1987/03/15
ETLA8706	D:1987/06/15
ETLA8709	D:1987/09/15
ETLA8712	D:1987/12/04
ETLA8803	D:1988/03/15
ETLA8806	D:1988/06/15
ETLA8809	D:1988/09/15
ETLA8812	D:1988/12/04
ETLA8903	D:1989/03/15
ETLA8906	D:1989/06/15
ETLA8909	D:1989/09/15
ETLA8912	D:1989/12/04
ETLA9003	D:1990/03/15
ETLA9006	D:1990/06/06
ETLA9009	D:1990/09/06
. . .	
KT8801	D:1988/01/26
KT8807	D:1988/07/04
KT8812	D:1988/12/12
KT8901	D:1989/02/16
KT8907	D:1989/07/05
KT8912	D:1989/12/11
KT9001	D:1990/01/26
KT9007	D:1990/07/12
KT9012	D:1990/12/19

HTASEQ_P, final figures

OBS	SID	STATUS	QQ_P
Y:1987	KT8801	-	3.23009
Y:1987	KT8807	-	3.80883
Y:1987	KT8812	-	3.80273
Y:1987	KT8907	X	3.98869
Y:1988	KT8901	-	4.6034
Y:1988	KT8907	-	5.17827
Y:1988	KT8912	-	5.23
Y:1988	KT9007	X	5.43386
Y:1989	KT9001	-	4.98
Y:1989	KT9007	-	5.16314
Y:1989	KT9012	X	5.22
...			

The table STATINFO contains a column SIDDATE, which tells the exact date when the source (every SID) was published. In the table above we have listed rows concerning this example.

As the result you get the following rows from the table ETLHTQ_P and HTASEQ_P:

OBS	SID	STATUS	QQ_P
Y:1987	KT8907	X	3.98869
Y:1988	KT8907	-	5.17827
Y:1989	ETLA8912	8	4.5
Y:1990	ETLA8912	4	2.5

Variable SID , ie. Source IDentification, tells the source, year and month of one row. STATUS is an auxiliary variable which in the case of short-term forecasts is defined as:

STATUS = 1	- forecast made in previous year's spring
STATUS = 2	- forecast made in previous year's summer
STATUS = 3	- forecast made in previous year's autumn
STATUS = 4	- forecast made in previous year's winter
STATUS = 5	- forecast made in same year's spring
STATUS = 6	- forecast made in same year's summer
STATUS = 7	- forecast made in same year's autumn
STATUS = 8	- forecast made in same year's winter

We get the result using the following Table script:

```
/* get rows from the STATINFO table where date of SID is earlier or equal to Jan 1, 1990 and rows from the
table ETLHTQ_P and HTASEQ_P since 1987: */

getdbs: SUHDANNE SID, SIDDATE FROM STATINFO
            WHERE SIDDATE <= 'D:1989/12/04' AND SIDDATE /= '-';
          OBS,SID,STATUS,QQ_P FROM ETLHTQ_P where OBS >= 'Y:1987';
          OBS,SID,STATUS,QQ_P FROM HTASEQ_P where OBS >= 'Y:1987';
        ;;

/* sort the STATINFO table by the SIDDATE column: */
load:STATINFO.
STATINFO.(')=STATINFO.(sortby(STATINFO.SIDDATE))

/* next paragraph of commands select the latest observations, which were available 4.12.1989. Operations are
made for both tables ETLHTQ_P and HTASEQ_P. */

load:ETLHTQ_P.
concat(match(ETLHTQ_P.SID,STATINFO.SID))  /*make index vector _,which has common SID's from tables
                        STATINFO and ETLHTQ_P and delete rows not found from the STATINFO table */
ETLHTQ_P.(')=ETLHTQ_P.(sortby(_))            /* sort the ETLHTQ_P table by the sorted index vector got
                                                from the previous command */
ETLHTQ_P.(')=ETLHTQ_P.(iuniq(ETLHTQ_P.OBS)) /* find latest observations from the ETLHTQ_P table
                                                 sorted by time */
'forec_end = 'end; setmiss(ETLHTQ_P.,'end+1)


load:HTASEQ_P.
concat(match(HTASEQ_P.SID,STATINFO.SID))
HTASEQ_P.(')=HTASEQ_P.(sortby(_))
HTASEQ_P.(')=HTASEQ_P.(iuniq(HTASEQ_P.OBS))
'final_end = 'end; setmiss(HTASEQ_P.,'end+1)


/* join observation, which are not found from final figures and are in table of forecasts (in this case: Y:1989 and
Y:1990), to table of final figures. */

span(1,'forec_end+'final_end)
nr(missing(match(ETLHTQ_P.OBS,HTASEQ_P.OBS)))
HTASEQ_P.('+'final_end) = ETLHTQ_P.(sortby(_))
```

The main difference between SQL and the Table script is that one may use the numbers of order in Table. For example, the function iuniq() of the script finds all multiple occurances and selects the latest one, thus yielding the indices of unique elements. This is not to say that is impossible to resolve the problem by SQL, although it might be difficult in our table setup.

6. Technical Dependencies

The software runs in HP-UX and the platform of ours is a set of HP9000/800 series computers.

Table has a full screen editor, the use of which is optional. Besides that the package has an user friendly text editor. These have been made by using the Curses-library in Unix.

The data retrieval of the base is a different module. The use of it implies that you have the package of Mimer database utilitities. The same concerns writing procedures to the base. If you want to link Table to another database, you have to write the interface by using the embedded SQL of that product. If the application resembles that of ours, i.e. the management of temporal database, our experiences show that all that is needed is the procedure which gets the data from an arbitrary single table. Table takes care of joining the data across the tables in various ways.

Our Canon drivers imply the use of Canon laser products. However, you may use the Video-Show or Ventura to get the pictures, e.g., to an HP Laser Jet.

The link between HP-UX and the local workstations is carried out by the Reflection emulation program. Reflection is a trademark of Walker Richer & Quinn Inc.

In order to get the VideoShow presentations to the PC, color printer or to the television, one must have a proper set of the products of General Parametrics Corporation.

The control of HP- graphical terminals makes use of the Starbase routines. However it is possible to close the HP- screen, if wanted, in which case we are independent of the Starbase.

7. What next?

The basic idea, open environment to allow efficient combination of best softwares and best quality products without compromising, does not seem to need reconsideration. The software has also led to new products and product ideas, besides allowing efficient graphics production from the database to publications and VideoShow presentations. All the forecast graphs of the Finnish Economy, over 50 in number, are available to ETLA's customers in VideoShow form both on-line and in disc form. Customers who have VideoShow can also download updated graphs from ETLA in custom-wise tailored form. Another product line is the software itself and the consultations around it.

Scientific data base management poses many problems but there are also important improvements to be expected. To mention only one, developments in object-oriented approach to databases may well lead to much more efficient ways to deal with data transformations and graphics.

References

(1) Ageloff Roy (1988) 'A primer on SQL'. Times Mirror/Mosby College Publishing, St. Louis 1988.

(2) ANSI (1983) 'American National Standard for Videotex/Teletext Presentation Level Protocol Syntax, (North American PLPS)'. ANSI X3.110-1983

(3) Cleveland William S. (1985) 'The elements of graphing data'. Wadsworth Advanced Books and Software, Monterey, California.

(4) Jones (1989) 'SQL in the engeneering applications'. A lecture given in Interex HP users conference, San Francisco 1989.

(5) Knuth Donald (1973) 'Sorting and searching'. Addison Wesley, Menlo Park, California. Especially pp. 164-166.

(6) Lassila Jukka (1990). 'Preliminary data in economic databases'. Lecture Notes in Computer Science 420, Springer-Verlag 1990, pp. 219-226.

(7) Wittkowski Knut M.(1988). 'Knowledge based support for the management of statistical databases' in Statistical and Scientific Database Management, Springer-Verlag 1988, ed. M.Rafanelli, J.C.Klensin, P.Svensson.