A Concept on Data Delivery for Large Enterprisewide Information Systems

Stephan Tobler
Release 1.2
May 1995

Abstract

This paper will discuss a practical approach to the gathering of information from a variety of data sources to any frontend application and focus on some typical problems in data delivery systems. The concept is based on a three level database architecture realized with the SAS system as a single point of control on a central platform. Source data may come from different platforms and databases and be integrated into an enterprisewide infobase which represents a neutral basis for distribution to application-specific subsets on central or decentral computing services. This concept is currently implemented under control of the SAS system on an MVS mainframe accessing SAPs R/2, DB2 and flat files, including data distribution to a management information system of the PTT Telecom on an OSF/1 server. Modularity in design and coding supports the universal usability of the system.
**Definitions**

In this document the terms listed below are to be understood as having the following meanings:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLAP</td>
<td>On-Line Analytical Processing. As distinct from OLTP systems (On-Line Transactional Processing), in OLAP applications, the emphasis is placed on information delivery and interactive retrieval. MIS (management information systems) and EIS (executive/enterprise information systems) are particularly important here as the end applications.</td>
</tr>
<tr>
<td>Data sources</td>
<td>Productive databases which deliver information to the frontend applications are defined as data sources. These may be financial and investment calculations, personnel information systems etc.</td>
</tr>
<tr>
<td>Data servers</td>
<td>All platforms on which data gathering and processing take place are referred to below as data servers. This heading therefore does not include the client systems on which the frontend application is ultimately performed.</td>
</tr>
<tr>
<td>Source data library</td>
<td>A source data library is the first stage in data gathering. The extracts from the data sources are inputted here after undergoing only coarse preconsolidation. They are then made available for further processing. A source data library provides the input for a particular problem area in the infobase.</td>
</tr>
<tr>
<td>Infobase</td>
<td>The infobase contains condensed and consolidated data from the relevant data sources. This is the central element in the entire data processing task.</td>
</tr>
<tr>
<td>Target data library</td>
<td>This is an application-specific database for the frontend applications. The data is compiled on the central data server for the current month and transferred to the application server via SAS Connect. Apart from normalized relational tables, a target data library may also contain denormalizations which are necessary for technical reasons and sequential files. The client part of the system accesses these tables either directly or via functional data layers (metabase).</td>
</tr>
<tr>
<td>Backend</td>
<td>The term backend denotes the part of the system which contains all the data delivery and consolidation. The boundary with the frontend is located at the application-specific database on the application server.</td>
</tr>
<tr>
<td>Frontend</td>
<td>This is the client application including access to the data from the application and data server.</td>
</tr>
</tbody>
</table>
From data source to frontend application

Purpose

The purpose of data processing from a wide range of data sources and for different application areas with different needs must be to make the desired information available in a suitable form after updating as far as possible to the latest situation. The imperative aim will therefore be to perform all the process control functions at a single, central point. It must not happen, for example, that human resources data are already available for a particular month while information on staff turnover is not yet available. To satisfy these criteria, a comprehensive concept must exist.

Multilevel concept

Information on every problem area which can be regarded as a self-contained unit must be supplied consistently, i.e. with the same degree of actuality. I have therefore chosen a multilevel data processing concept which enables me to effect individual verification of both source and target libraries, while nevertheless proceeding in a largely application-neutral manner.
System environment

The data sources as such and the frontend applications (including, where appropriate, the
decentral data and application servers) constitute the system environment. A separate
source data library ("raw database") must be provided for each data source without already
taking explicit account here of the intended purpose for which the data is to be used. The
aim is to obtain a database with the widest possible general validity applying equally to all
the target systems. In practical terms, this means, for example, that separate libraries must
be constituted for personnel, finance and stock inventory information.

Problem areas

These libraries are separately updated at specified intervals (daily, weekly or monthly
depending on the particular problem). It may therefore happen that financial data will
already be available for a particular month while human resources data will still date from
the previous month. However, as long as there are no discrepancies in respect of the
updating of the same problem area, the data will remain consistent as in any case only
generally valid reference tables will overlap in the normal situation.

In the event of a discrepancy between e.g. human resources data and staff turnover, a
blocking mechanism for further processing must be provided. The data shall not be passed
on until all the quantities in the source data library have been updated to an identical level.

Testing the attribute domains

When a source data library is ready for transfer, the attribute domains must be verified.
Should a key value not be present in a corresponding reference table (matching table),
processing of the problem area concerned must be stopped again and a fault list issued
(print or file) showing the mismatched key value. The attribute domains (at the physical
level these will generally consist of a reference table) must be completed at the earliest opportunity.

Infobase content

In other words, only entire problem areas are transferred as a single entity in each case. When the values in a source data library finally coincide, the data will be consolidated and added to the infobase. The infobase itself always contains several periods of data uploads. This is explained, in particular, by the need for a particular period to be transferred repeatedly to the frontend applications without excessive complications. This is intended to provide a minimum degree of security in the event of possible program and download errors. In this way, a previous period can readily be delivered at a later date if a data transfer is marred by faults.

As a matter of principle, the infobase contains a largely normalized database which is independent of the target systems. In time, an infobase may develop into an enterprisewide information source. In addition, it will be possible in large measure to avoid redundant databases in the context of the data processing described here. It is advisable to prepare a logical data model of the entire infobase and to adhere to that model wherever possible.

Data supply for frontend applications

The actual preparation of data for the target systems does not take place until transfer is effected to the application-specific target data libraries. Here, denormalized tables are made available for the frontend to provide best performance in data access.

If the frontend database remains on the host computer, the work of the backend services will be finished. The clients can now access their information directly.
If a decentral data server is used (as is the case with MIS Telecom), the target data library must be transferred 1:1 to the appropriate platform. This action can advantageously be performed via SAS Connect, as both resources (local and remote) can then be verified from the same (local) SAS session. This has the advantage that the final decisions as to whether or not transfer is to take place can then be taken shortly before the download (e.g. comparison of the modification date on both sides).

Verification via update profile

One central problem in data delivery using this concept resides in the verification of each single entity in order to determine whether it can now be updated and transferred or whether it still contains the same information as when it was last read.

This decision will preferably not be taken by reference to the "last modified" date in the table but via a special update profile dataset. If the "last modified" date alone is verified, there is a risk of initiating a read procedure in the event of unintended storage of a data set (even if the user only wanted to consult the latter), so your procedures would reload a period and jeopardize the consistency of the infobase.

Therefore a table must be provided which contains for each entity of the three processing stages (source data libraries, infobase and target data libraries) a marker showing the name, source, last period and update date. In the MIS Telecom project which will be described in more detail, the result was a table containing the following structure:

<table>
<thead>
<tr>
<th>UPDPROF</th>
<th>Name of source/target data library or infobase</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DBPOOL</td>
<td></td>
</tr>
<tr>
<td>MEMBER</td>
<td>Tablename / Entity</td>
</tr>
<tr>
<td>SOURCE1</td>
<td>Data source of member (source No. 1)</td>
</tr>
<tr>
<td>SOURCE2</td>
<td>Data source of member (source No. 2, where present)</td>
</tr>
<tr>
<td>SOURCE3</td>
<td>Data source of member (source No. 3, where present)</td>
</tr>
<tr>
<td>MATCHING</td>
<td>Name of reference table for key values (key matching)</td>
</tr>
<tr>
<td>PERIOD</td>
<td>Period of last update (or release of a generation file)</td>
</tr>
<tr>
<td>UPDAT</td>
<td>Date of last update</td>
</tr>
</tbody>
</table>

In actual fact, the indication of at most three data sources will be sufficient. Of course a denormalized table of this kind is not a model example of database design. However, we are not concerned here with an actual component of a database but merely with an auxiliary object which can be consulted for update checks and must above all not slow down processing performance. The date attribute UPDAT generally only has a statistical value as interrogation of the update level should always be effected via the PERIOD attribute. However, it is often helpful to see when a table was last updated through an upload task.
Nature of the source data

A basic distinction can be made between three types of source data. Firstly, sequential data files (flat files) are a current form of data exchange between proprietary systems and for data transfer across different platforms. Secondly, direct access to existing databases (generally hierarchical or relational) is a possibility which must, however, first be examined in great detail, particularly in the context of operational data sources. It must also be noted that various kinds of information cannot be obtained as a file but often in the form of lists, press articles etc.

Sequential data files

For reasons of data certainty, consistency and also of technical and meaningful feasibility, it is often impossible to access directly the data source. In such cases, a sequential file must therefore be made available from the data source, if such interface files do not already exist for other applications. The aim will be for these downloads already to contain a suitable degree of consolidation of the basic data. In that case it will be sufficient in the best assumption for the data to be read in and transferred directly without further consolidation. In addition, this guarantees accuracy of the consolidations as it is safe to suppose that experts in the different working areas are more familiar with the data. If consolidation is not effected until the infobase stage, you may risk having misunderstood the meaning of the data or you will not be able to react soon enough to effect changes in the source data logic because the necessary information for this purpose may still be lacking.

When the sequential data files are named it is important to make sure that the reference period of the data is contained in the file name. This facilitates verification as to whether the current file is new or has already been inputted into the infobase.

Direct access to DBs

No general rule can be laid down as to when an existing database should be accessed directly. Important questions which arise in this context are as follows:

- Is the database used intensively in online working?
- Are the data consistent?
- What is the data quantity?
- What losses of performance must be expected to occur in the event of access to the entire data stock?
- What is the relative complexity of permitting direct access or generating a sequential file or a specific transfer table?
- What maintenance costs must be expected for the interface?
- Do security problems arise?

When handling large quantities of data and in cases where the update level of the data in the frontend does not depend on the current time, the variant of direct access should as a
general rule be avoided. It often makes much more sense to prepare a specific interface database which contains the requisite information already largely updated at the stipulated intervals.

Data to be inputted manually

In cases where data must be inputted manually into the system from lists etc., it will not always be necessary to include such data in the central data processing. Often, information of this kind is highly application-specific and not relevant to other frontends. That is why manual inserts into a source data library will normally not be desirable.
Process design

Design and implementation

Each of the objects presented below represents a set of SAS macros which form a logical unit. It should be noted here that only the design of the application follows an object-oriented approach. At the implementation stage, important features such as inheritance, encapsulation etc. must be largely dispensed with because of the nature of SAS macros. However, the object-oriented approach in design enables many of the problems which occur with large information systems to be avoided and gives the system a high degree of re-usability, modularity and hierarchy.

General nature of the data delivery process

In the most general form, four control objects can be distinguished. These are the process control, process initialization, main process control and download control objects.

Process initialization

In process initialization, several utility files are normally made available; these files are required in the course of processing. A file of this kind might for example contain a directory listing of all the available FTP files. A subsequent decision can then be taken as to whether a new file exists and accordingly needs to be read.
Main process control

This object controls the main processing sequence. For each problem area, data must be drawn from the relevant sources to a source data library. The main processing operation includes the entire data delivery process from the data source via infobase to the target data library.

Download control

The target data libraries generally have to be transferred to decentralized platforms. In that case, the download object will establish a connection with the decentral computer via SAS/Connect and effect a comparison of the update level of the data files between the decentral and central target data libraries. This comparison is made via the dictionary tables of the two SAS sessions. If the data on the remote (decentral) host shows a later update date this data will selectively be downloaded.

Main process

A decomposition of the process control logic shows the following generally valid structure of the problem area-oriented data delivery.

The source data library load object must be provided individually for each problem area and one target data library load object has to exist for each of the target systems. All other objects can be used repeatedly. The cycle shown here will be run for each problem area.
General utility objects are shown in the lower part of the diagram. These perform actions which are repeatedly requested by different objects (library and file allocation, return code handling, verification of attribute areas etc.). This object group is not considered in more detail as it is heavily dependent on the particular central platform which is used.

In the source data library load object, new data are inputted from the given data source into the specific source data library. The data must normally be consolidated. This will preferably be done already in the first processing phase as far as this is permitted by general validity for different target systems. In frontend systems, concise and clear designations are preferred for text fields which often have a length of more than 50 characters in upper case. For many target systems, special matching tables are therefore used. These must of course be both complete and accurate. The key matching control object therefore primarily involves verification of these relationships.

Where new data are present in the source data library and the attribute domains have been verified, the infobase load object is able to add the corresponding entities to the infobase datasets. If the infobase is only to contain a particular time section (e.g. the past three months), it is essential to make sure that all records which are older than the specified period are deleted immediately after adding new entities.

Finally, all the updated infobase tables are loaded into the corresponding target data libraries via the target data load object. Depending on the requirements of the individual target systems, denormalizing followed by further consolidation will often be necessary here. In the case of data which has a critical relevance to performance (complex consolidation, large data quantities), it may be appropriate for technical reasons to compile summary extraction tables.
OLAP in practice

I had the opportunity to apply the concept described above in the context of the MIS Telecom project of the Swiss PTT Telecom and achieved a considerable increase in productivity as a result. On the one hand, there is no need to "reinvent" each new data source, while on the other the reference tables from the individual problem areas in the infobase were found to coincide in large measure.

MIS Telecom has the following broad technical structure:
Various sequential files, SAPs R/2 system and DB2 are relevant as data sources. These contain information on human resources and staff movements, cost and investment accounts, fault frequencies in the telecom network, the number of connections and subscribers to Telecom services etc. In order to give an idea of the scale of the system, part of the system as it has been implemented in practice is shown in the above diagram. The above data sources are shown as examples of the typical kinds of source systems which play a major role in OLAP systems.
Conclusion

An increasing demand exists for enterprisewide, compact information. That trend will no doubt continue for a long time to come. EDP service providers are finding it increasingly difficult to resolve the long-standing conflict between update levels, completeness and performance.

To enable the varying needs of different customers in respect of OLAP applications to be satisfied over an extended period, the greatest importance certainly attaches to the conceptual level of information delivery. This enables long-term application backlogs to be absorbed and the time taken to respond to the market to be shortened.

Regarding the concept described here, the fundamental principles must be to generalize and modularize as far as possible. The conceptual work for data delivery can and must account for a large share of the overall expenditure involved in setting up large information delivery systems.

There is actually a heated discussion on the profits of multidimensional vs relational database systems. I think both implementations have advantages and disadvantages, depending on the sort of data and work your datawarehouse is built for. Anyway, the following concept is not based on the use of one or the other implementation, but it’s a logical approach to an intelligent solution, giving you the ability to serve your customers as an information provider.

Since the SAS System does not support multidimensional database structures, this paper confines originally to the use of relational databases. Although it is not compelling to implement every part of the following inside the SAS System. You may mix different database management systems, even hierarchical, realtional and multidimensional ones as long as data exchange between these is possible.

Feedback welcome

If you have any suggestions to this paper...if you like to know more about the use of the SAS System at the Swiss PTT...if you know about other sources or documents on this topic

please e-mail me tobler@vptt.ch (office)
stobler@ping.ch (home)
or send me a fax +41 (0)31 338 90 69

newsgroups for further information comp.soft-sys.sas comp.databases.olap

or look for appropriate topics on The Data Mine Homepage (www.cs.bham.ac.uk/~anp/TheDataMine.html)