Paper 096-2009

Innovation, Implementation and Integration – Building a DataWarehouse with SAS DI Studio and MS SQL Server

Author: Søren Junk, Bankdata

WHAT IS BANKDATA

Bankdata is a service provider for 15 banks with 285 branches, and supplies both shared and individual solutions.

Bankdata builds systems for banks and is active from the initial analysis phase through implementation of the system. We help our customers integrate their solutions to central systems which are hosted by us, and offer technological advice and support when it is needed.

Bankdata participates in national and international infrastructure which enables our customers to make national and international transactions using standards such as Swift, Sepa or EBA. We receive currency rates every day from the Danish national bank along with stock rates from NASDAQ and OMX.

The DataWarehouse in Bankdata receives information from several operational system located in a DB2 database on a Mainframe system. We also receive external data from other data centers, the Danish government and from mortgage companies. This data is the basis of our DataWarehouse.

DataWarehouse supplies other systems, e.g. Basel II and Anti Money Laundering, with historical data which is used for trend analysis and reporting. Furthermore we supply data for reports used by the employees in the banks. For instance for use in reporting about which customers would be eligible for a certain product. In addition we also create large text files e.g. for use by our Credit Risk Management system.

OUR SITUATION

The previous setup of our DataWarehouse was using IBM's DataWareHouseManager as ETL tool along with COBOL stored procedures with an underlying DB2 database, all located on a mainframe. The problem with this setup was that developing our ETL flows took too long, and at the same time it was very hard to performance tune the setup. One of the few possibilities to increase performance was to upgrade the mainframe CPU. Making such an upgrade is very expensive due to the software licensing model used on a mainframe. The software licenses are tied to MIPS of the CPUs, and when upgrading the CPU to satisfy performance for one product, you will receive further licensing fees for all the other software products installed on the mainframe.

When we found out that IBM's DataWareHouseManager would be phased out and therefore not supported in the near future, a migration project was initiated with the goal of moving to a modern, future proof and flexible DataWarehouse, implemented on a Windows platform.

DEMANDS TO THE NEW DATAWAREHOUSE

We started by listing demands for the new architecture. The first demand was total data separation between all banks. At no time is it allowed for one bank to see another banks data.

Another demand was that we wanted to implement an architecture that supports the usage of metadata. Using metadata is a high priority because we believe we gain a lot of benefits from it. Before migrating the DataWarehouse, all information regarding data flow was manually maintained in a large spreadsheet. To keep the spreadsheet up to date is a lot of work, and it took a long time to find out where data for a certain sub model was coming from. To implement metadata in our architecture demands that the tools we use for development also should support the usage of metadata.

Furthermore an architecture that supports a wide range of tools and applications was demanded. The DataWarehouse should be able to supply systems with data regardless of the platform and other system specifics. We should also be able to supply users with data in a tool or application of their choice.

Being a service provider for 15 banks one of our great advantages is that we develop common solutions which are used by 15 different customers. In addition we also develop solutions to satisfy individual needs of one bank. This affects our architecture for the new DataWarehouse, because we have to be able to build both common solutions as well as satisfy individual demands in our implementation.

Another issue to be considered was the long development time. In the new setup we wanted to reduce the development time. We looked for a setup that made it possible to develop ETL flow at a faster pace. This means that the new ETL tool should be intuitive and easy to use, and at the same time be able to reduce the development time.

Moving from a mainframe to a Windows platform raised the issue of scalability. It is not possible to upgrade the CPU of Windows servers to the same extent as on a mainframe. Therefore our new setup should be able to handle scalability. How this is handled is described in the implementation section.

A FEW FACTS AND FIGURES

We have chosen an architecture using SAS DI Studio as ETL tool and a MS SQL Server as database. This architecture provides the power of SAS for processing large amounts of data, and we get the power of MS SQL Server for data selection.

The DataWarehouse setup is running on a Windows platform on HP ProLiant servers.

So far we have developed approximately 2000 ETL jobs per bank, and we are currently executing 30 000 jobs daily in our batch flow.

The size of the MS SQL Server is 30 TB and it will grow 5 TB per year without adding new business areas to the DataWarehouse. However, we are expecting to add data from new business areas at about 15 – 30 TB each year for the next many years.

Our Data Staging Area is 40 TB and will grow with 20 TB per year.

INNOVATION

PREPARING THE MIGRATION

A priority in the migration project was to find a new ETL tool. An analysis of the market was made and four products found that we believed could meet our demands. We then made a formal “Request For Information” where the four vendors had to fill out a form on how they could meet our demands and present their solution.

SAS was chosen as primary vendor, and having no knowledge of SAS in the project two employees were sent on a two day SAS DI Studio course to learn the basic functionality of the new product.

A Proof Of Concept was initiated to show that SAS DI Studio indeed was able to meet our demands to a new ETL tool. A sub model was selected where ETL flows should populate the star schema. In our previous setup the development was estimated to last 3 – 4 weeks. We were very positively surprised when after 2 days we had the first edition on the ETL flows. We were able to develop at a much faster pace even with only the basic knowledge of SAS DI Studio. The Proof Of Concept was accepted and installation of our development environment started.

Before starting the migration of the DataWarehouse, we established development standards to be kept during construction for setting a standard method on how to develop ETL flows in a uniform way. These standards were very helpful in the construction phase for both experienced programmers and for newly hired programmers who could use the development standards documentation for reference. The development standards helped to gain an overview over what should be done when and why.

BUILDING AN ARCHITECTURE

The first demand for our architecture was that it should ensure the further growth and usage of the DataWarehouse. It should be possible to add new business areas to the DataWarehouse without bumping into bottlenecks. Furthermore we

should be able to supply the users with data for reports and analysis in a wide range of tools and applications. To ensure this we began considering how to support scalability in our implementation.

Basically there are two ways of scaling: scale up or scale out. Scaling up means buying a server with more / faster CPUs and more RAM or adding CPUs to your existing server. This is not an option for us, partly because we are using one of the largest HP ProLiant Servers available, and partly because we do not consider it a sound strategy to make the performance of the DataWarehouse dependent on the availability of faster CPUs and servers.

Scaling out means adding one or more servers to your configuration and spreading the load across the servers. This way you can get the best and the most cost effective scalability. To conceive a scale-out strategy requires a lot of work, because you will have to consider a lot of performance scenarios to make sure the scalability works. To support a scale out architecture it is important to find a way for your ETL flows to execute across the servers without losing performance.

In order to meet scalability demands, we have chosen to partition our data. All of our customers have a unique bank number. We have created a set of databases for each bank and these databases contain data for one bank only. In other words: We have used the bank number as basis for our data partitioning. Knowing that we do not have ETL flows that read data from one bank and updates another, we are able to ensure the performance of our ETL flows because they always read data from only one database and will write to only one database.

Having created the same structure in 15 databases, one of our concerns was how to avoid developing ETL flows more than once. Being able to develop ETL flows at a faster pace, the time saved here should under no circumstances be used to develop ETL flow 15 times – once for each bank. We were able to configure our SAS Metadata Server to support that we only develop ETL flows once. How this is done is described later in this section.

Another issue was to be able to supply data to other systems as well as our customers on different platforms in multiple ways. This was accomplished by being able to use the open architecture in MS SQL Server using ODBC or OLE/DB as the communication method. In addition we could use other methods like Message Queues which is also supported by MS SQL Server. Another way to exchange data could be through SAS Metadata Server or using text files. Having all these options we were able to implement an open architecture.

METADATA USAGE

At Bankdata we use metadata in several ways. In our data modeling tool ERwin all tables are created with a business description of what business related data the table and columns contain. This description is exported from ERwin along with the rest of the table definition and imported in SAS Metadata Server. Here they will be used as source and target tables. The advantage of importing metadata from ERwin is that the business description on the tables/column will be imported to SAS Metadata Server in the description column of the table. This business description will follow the table when information is passed on to MS Share Point Server. We use MS Share Point Server as our front end tool to be used by our customer's developers. The developers are able to get a better idea of what types of business data the table and columns contain of from the business description. This will help them develop reports for the users faster and more accurately. As a bonus the load on the hotline function in the DataWarehouse department will be reduced.

Having metadata in SAS Metadata Server gives us another big advantage regarding documentation. Using Impact Analysis and Reverse Impact Analysis we are able to document where data comes from and what steps the data pass through before being presented to the users. This saves us a lot of manual work and we are able to retrieve correct metadata information very fast.

In addition we have a few jobs that extract metadata through the Open Metadata Interface from the SAS Metadata Server. These jobs gather information concerning jobs that have been checked in after they have been scheduled, or jobs that haven't been checked in for a longer period. An e-mail with the names of the jobs that require that we take action is then sent. This helps us to be sure the correct version of an ETL flow is executed.

SAS METADATA SERVER SETUP

The setup of our SAS Metadata Server required some analysis and planning. Having partitioned our data into 15 sets of databases (one set for each bank) we now have the same table 15 times, but in order to avoid developing ETL flows more than once we should have only one metadata table. The metadata table should then point out 1 of 15 physical database tables. We solved this problem by accessing only the tables of one bank per session. This way we are able to

point out LIBNAMES to the correct database by dynamically (pre-) assigning them with parameters. The same parameter is then used in our batch flow, where one ETL flow is executed up to 15 times – once for each bank.

IMPLEMENTATION

In this section the implementation of our architecture will be described.

SERVER ARCHITECTURE

In our server architecture we have 3 main servers:

- SAS Metadata Server where all metadata are stored
- SAS Workspace Server that executes all our SAS jobs
- MS SQL Server that contains all our databases

When a user logs on to the SAS Metadata Server using SAS DI Studio or SAS Enterprise Guide, a lookup is made to an external table containing information of which bank the user is employed by. This information is passed on to the SAS Workspace server which is able to dynamically assign a libname to the database of the bank. The user is then able to see the data of the bank he is employed by.

In our batch flow a similar setup has been implemented. Instead of retrieving information from an external table regarding bank number, this is passed on to the flow as a parameter.

All LIBNAMES are assigned per session which means we are able to execute ETL flows for several banks at the same time. That is due to that each ETL flow has its own session and therefore can point to different databases.

Furthermore a libname to a physical path is assigned – one for each bank. This is where all the SAS data sets are located. We consider all data in SAS tables temporary and this data is only used for debugging purposes. All the permanent data is stored in the MS SQL Server.

The architecture and setup is illustrated in Figure 1.

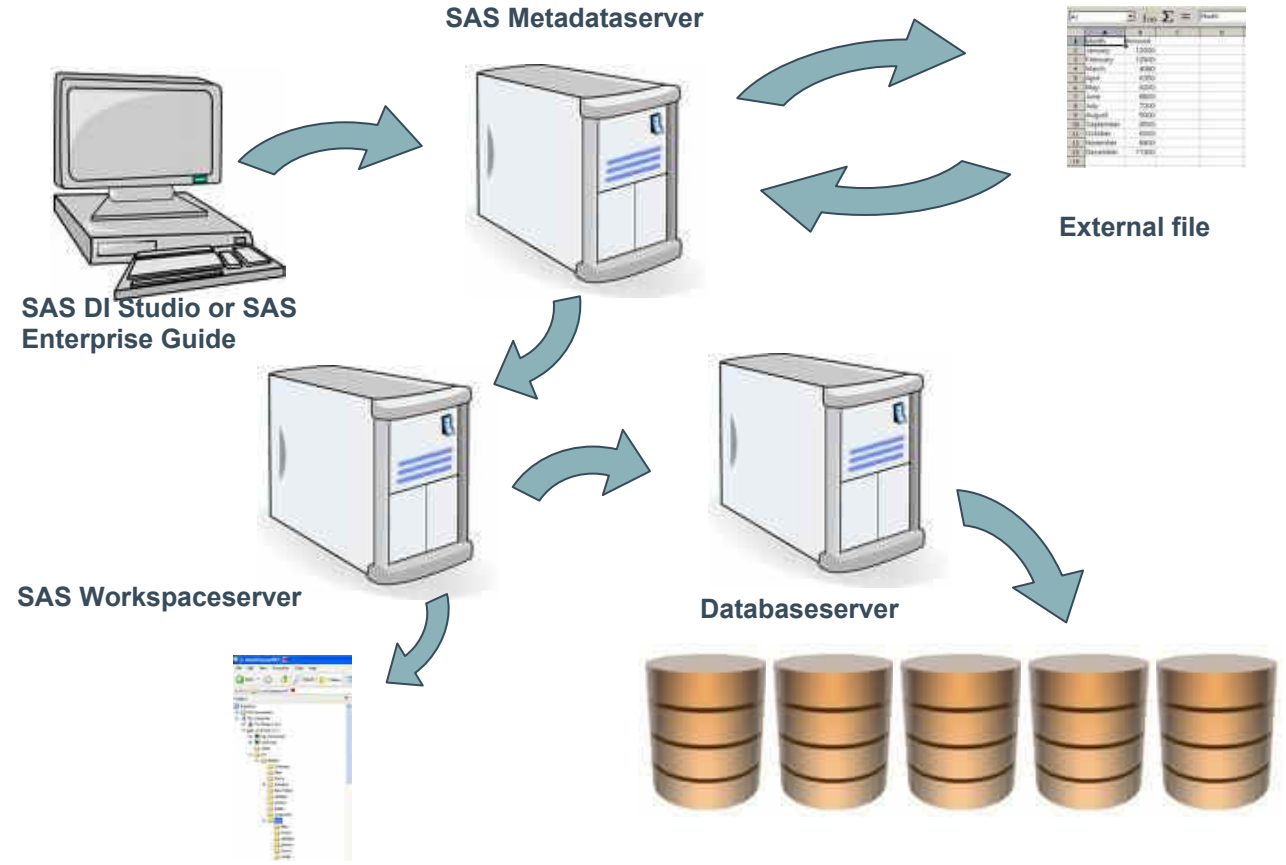

Figure1. *Server architecture and setup*

DATABASE ARCHITECTURE

Our database architecture has been implemented to support scaling issues. At the moment we are running on a single physical server for our databases. On this server 2 instances of MS SQL Server have been installed to simulate scaling and the databases has been divided into these instances so that the databases for one bank are always found on the same instance. Communicating between instances in our ETL flow is no different from communication between two physical servers. This way we can ensure that our ETL flow are able to perform when the scalability need arises.

The databases for one bank consist of one database for a base layer and one for a presentation layer. The base layer contains aggregated and summarized data from a system perspective, and the presentation layer contains data from the base layer which have been aggregated and summarized to satisfy customer needs. The users only have access to the data on the presentation layer.

Our batch flow starts by updating a database containing common data. The tables contain data regarding calendar information, geographic or demographic information which will not differ regardless of the bank. After these tables have been updated the data are copied to all the individual bank databases on the base layer. After the data have been copied, ETL flows are executed to update the banks individual data on the base layer. When the base layer is fully updated the flows to update the presentation layer are executed.

The database architecture is illustrated in Figure2.

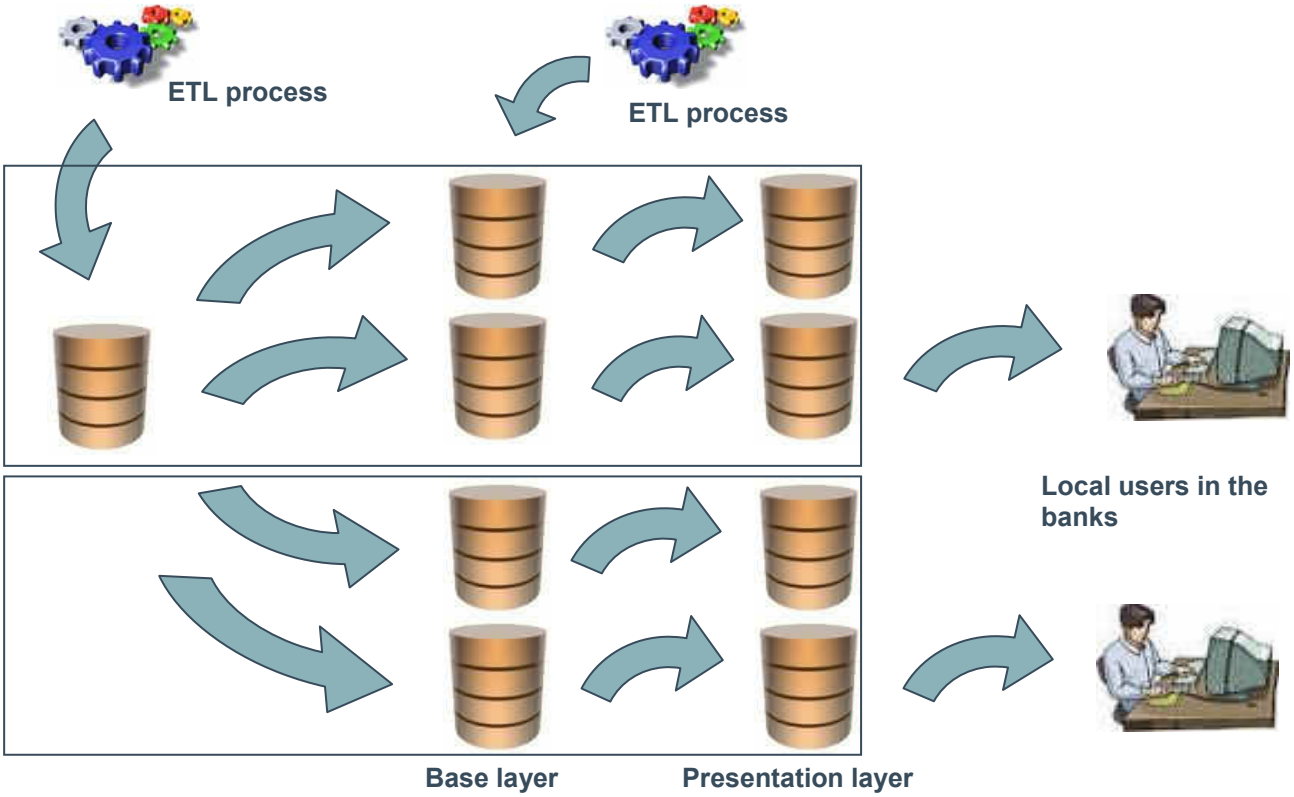

Figure2. *Database architecture*

USER ACCESS MAINTENANCE

To avoid manually updating the access rights for the users we have implemented a solution that handles this for us. Our solution is based on Windows Active Directory where a number of groups have been created for each bank. In the bank the local administrator can add or remove employees to the groups connected with that bank.

On our SAS Metadata Server we have created the same groups as in the Windows Active Directory. We have developed a synchronization script that will take the information from the Windows Active Directory and update the information on the SAS Metadata Server. That means that users added to Active Directory groups by the local administrator in the bank will automatically be added to the same groups on the SAS Metadata Server during synchronization.

At the SAS Metadata Server we are able to configure Access Control Templates (ACT) to handle metadata security. They are added at group level, and as users are moved to a specific group during synchronization, their metadata rights will be handled according to the group they have been added to.
The MS SQL Server is directly integrated into Window Active Directory. This way we can give the users in certain groups read access to the database. No further integration is needed here.

To summarize: All metadata security and access are handled with the synchronization script to the SAS Metadata Server, and all data security is handled in MS SQL Server.

CHANGE DATA CAPTURE PROCESS

Our Change Data Capture process is mainly based on the operational data we receive from a DB2 database on the mainframe. We have chosen to unload the needed tables to text files and copy them to our SAS Workspace Server. This is done to minimize data traffic between mainframe and Windows servers and to minimize mainframe CPU load. This provides us with a uniform way of preparing the operational data for the Change Data Capture process with minimum mainframe CPU usage.

Every day we extract table/column information from the DB2 data dictionary and compare it with the information from the previous day. If any of the tables have changed – whether if a column has been added or removed, whether a column's data type has changed or the sequence of columns has been altered, all execution concerning the affected table will stop. If one of the tables has changed we need to analyze the impact and alter the ETL flows accordingly. It is not possible to handle this situation automatically.

If the table structures haven't changed, we know we are dealing with consistent data structures and we proceed to unload the tables to text files. The text files are then copied to our SAS Workspace Server. We then proceed to compare today's text files with yesterday's. This allows us to find all relevant changes, and split them into 2 tables – one containing records to be created, and the other containing records to be updated. Since we are using slowly changing dimension type II, we do not physical delete any rows – we inactivate them. The tables containing the changed data will be used as input to our ETL flow to update the databases.

The Change Data Capture process is illustrated in figure 3.

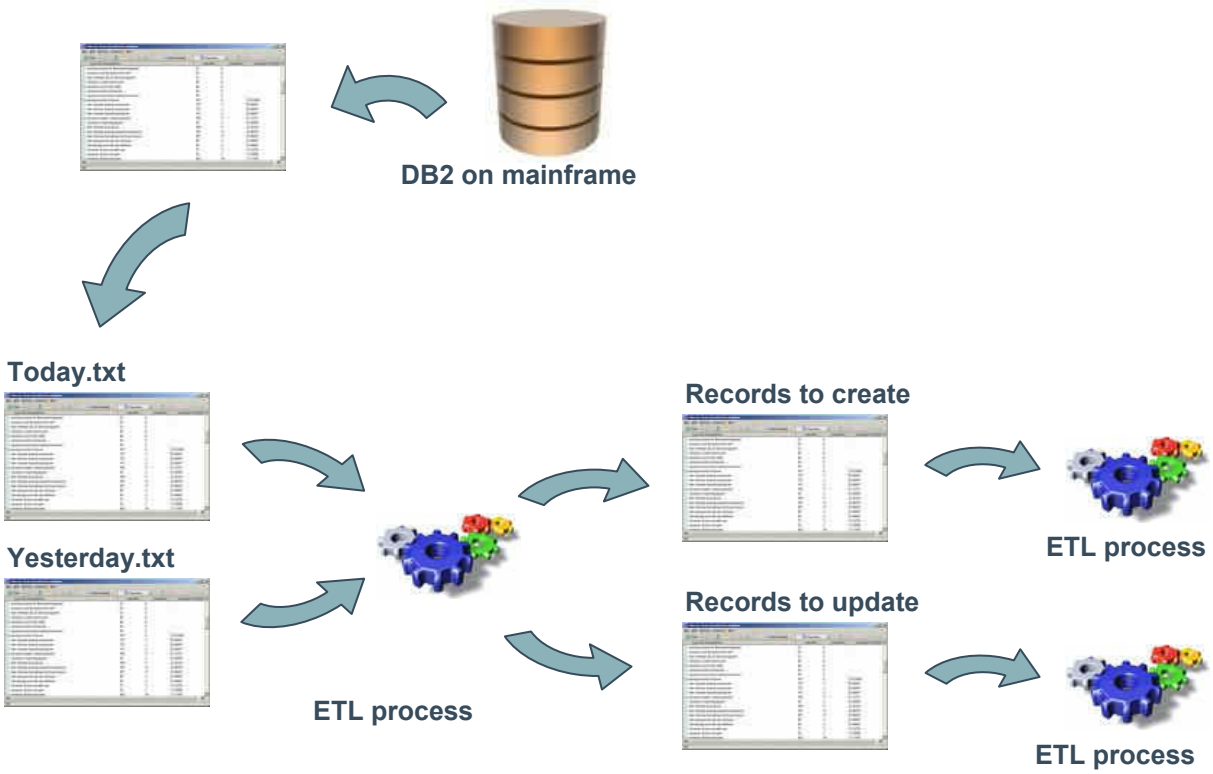

Figure3. *The Change Data Capture process*

INTEGRATION

INTEGRATION WITH SQL SERVER

After installing MS SQL Server, we were able to make the first libname from SAS to MS SQL Server within a few hours. We could read and write data to the tables, but the integration was far from complete.

The first problem we encountered was writing Unique Identifiers, a MS SQL Server data type containing our generated keys, using an OLE/DB connection. The SAS/ACCESS software added {} to the Unique Identifier which could not be interpreted by MS SQL Server. This was solved by switching to an ODBC connection which will not add {} to Unique Identifiers. This allowed us to keep on developing ETL flows, but the overall performance was very bad at the beginning.

We then started to focus on getting better performance and found that database performance is very dependant on libname options. We found that inserting rows in the database tables would perform much better when this was done in Bulk mode. With a few alterations in the libname, performance was enhanced.

Another option was to lock the database table during insert and update. Again this showed considerable performance gain on database transactions with very little work. Altering the libnames to use the new options does not require much work, but finding which options to use will take a lot longer.

Another problem we encountered was that the update functionality wasn't able to handle the situations we wanted. With a lot of workaround we got the update to work, but it wasn't able to perform satisfactory. Our solution to this problem was to develop a "User Written Transform" in SAS DI Studio which was able to update SQL Server tables. We found that using "User Written Transforms" is a very powerful tool and using them will not be at the expense of metadata usage which can occur in some cases when using "User Written Code".

INTEGRATING WITH OTHER SYSTEMS

Having implemented the open standard in MS SQL Server, we are now able to exchange data to users and other systems regardless of platform. One of the ways we receive and send data is through the use of text files. We receive text files containing mortgage data from external companies, data regarding people from the central person register and others. We send data through text files to other systems like credit risk analysis.

We can supply data through direct access to the database whether it is for another system, or if direct user access doesn't matter. Because the data security is handled by the database, we are able to grant read access for the end users who can extract the wanted data from the database to a tool or application of their choice, as long as they can communicate with the database (usually through OLE/DB or ODBC).

Another option is to let the users access the DataWarehouse through the SAS Metadata Server using SAS DI Studio, SAS Enterprise Guide or something similar. This way the users are able to use the entire DataWarehouse environment setup without having to create their own libnames. Using this option the users are given the possibility to see the business description contained on the SAS Metadata Server.

To be able to support both shared and individual solutions for our customers, we have placed common solutions in the foundation repository and individual solutions are placed in custom repositories. The individual solutions can contain data marts made for a specific bank, for a specific purpose, or a solution to deliver data for dashboards or other applications. Placing the individual solutions in custom repositories has the advantage that we are able to filter which users are allowed to se these repositories through Access Control Templates (ACT) in the SAS Metadata Server.

PEFORMANCE

PASS-THROUGH SQL

When using SAS DI Studio with a database, it is very important to use pass-through SQL. Pass-through SQL means that you build a query in DI Studio, pass the query to the database, and return the result. It is important to design your query according to the guidelines for pass through SQL otherwise SAS will retrieve the data from all tables involved in the join, process the join, and then present the result. The latter is much slower and has a negative impact on performance.

To use pass through SQL you can't use SAS functions like 'put' or 'input' in the query. The SAS ACCESS software is not able to translate these functions to the SQL Server, and the outcome is that pass through SQL is not used.

Another requirement is that all tables in the join are located in the same libname. During our migration project we developed a job that joined 25 SQL Server tables to a single SAS table. The SAS table contained less than 1000 rows, and the SQL Server tables contained from 100 rows to 45 million rows. The job executed for more than 5 hours before it was killed. We clearly saw that this would never work with the amounts of data in our production environment. We added an extra step to the job that inserted the data in the SAS table into an additional SQL Server table. The join now had to process 26 SQL Server tables, and execution time was reduced from +5 hours to 7 seconds.

Designing your flows to handle pass-through SQL will have a huge positive impact on performance.

Figure4 illustrates the difference between not using Pass-Through SQL (Example 1), and using Pass-Through SQL (Example 2)

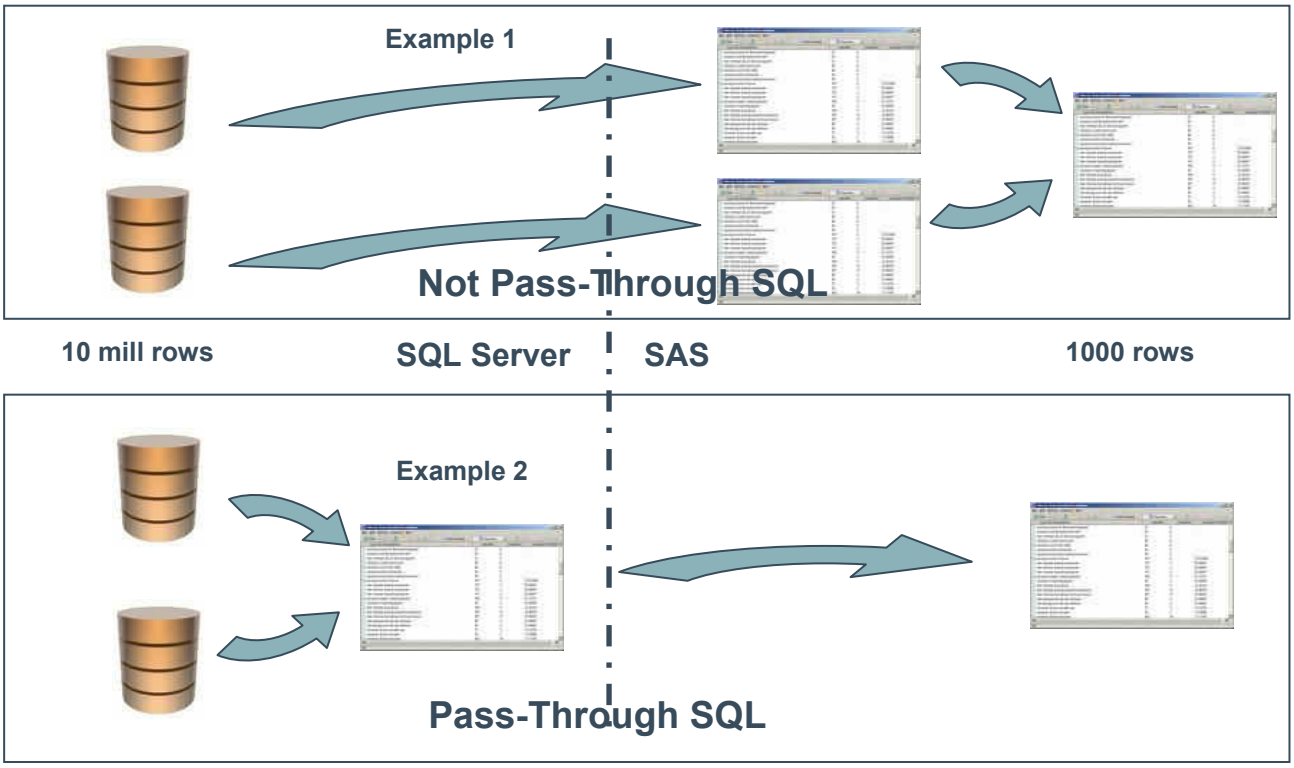

***Figure4**. The difference between not using Pass-Through SQL and using Pass-Through SQL*

OTHER PERFORMANCE ISSUES

When the new platform was implemented with the new ETL tool and the new database we experienced some overall system performance problems. The major problem was that no one could tell us how fast the servers were supposed to run, whether the database installation was optimal, whether there were any bottlenecks in the network, or whether there were problems with the SAN disks.

One of the major issues we faced was to optimize our usage of SAN disks. In cooperation with our storage manager we found out that even though our response time was very fast - less than 2.5 milliseconds, our throughput was less than 20 Mb/s. In order to optimize this we tried different configuration on the SAN disks.

We did this using a tool called IO Meter. Here we were able to create an IO profile of our setup, test this towards the SAN disk, and try alternative configurations. During this process we were able to change the throughput from 20 Mb/s to more than 600 Mb/s.

The ETL flows sometimes behave differently at the database than expected by the developers, and the database administrator can point out what to change. In cooperation with our database administrator we were able to set libname option that gave 10 – 15 % increase in overall database performance with only one hour work. Ask you local database administrator for advice and consult the SAS documentation on how to use the options in you libnames.

A general rule is that minimizing IO equals better performance. When developing ETL jobs, be sure to filter out the rows wanted as early as possible, and only to select the columns needed. The less data to be transferred, the faster the ETL job will execute.

When running SAS Metadata Server on a Windows 32-bit platform, the size of the foundation repository can be very important. Running 32-bit SAS is only able to address 2 GB of memory and because all metadata in foundation (and project repositories) are loaded into memory crossing the 2GB line will have an impact. Possible symptoms are that is takes a very long time to

- Check in/ check out
- Save a job

- Connect to the SAS Metadata Server
- Add a transform to an ETL job

There are a few ways to reduce the size of the repositories. One way is to format the project repositories when there are no jobs checked out to that repository. Another way is to specify the REORG=YES and RUNANALYSIS=YES when running the %OMABACKUP.

A third option is to export all jobs, folders, tables and libnames from the SAS Metadata Server, reinstall the SAS Metadata Server, and import the files again. This will remove the information regarding check in/check out, but at the same time it will reduce the size of foundation repository.

As a preventive measure it is a good idea to check the size of your repositories at regular intervals. This way it is possible to plan how to take the appropriate action.

The size of the repositories is only an issue when running a 32-bit SAS installation. In SAS 9.2 it is possible to run 64-bit SAS installations on most windows servers.

CONCLUSION

Implementing an environment of the scale and complexity used at Bankdata was no easy task. Adding the MS SQL Server made the setup more complex. It is important not to discard a solution because of the complexity. You have to look at what possibilities and benefits the users can gain from the solution.

We believe that with the migration of the DataWarehouse we have been able to build a future-proof and flexible solution. We have implemented an open architecture where we can supply data to systems and users regardless of platform, tool or application. We have furthermore implemented an architecture that supports both the possibility to scale up and to scale out. This ensures the further growth and usage of our DataWarehouse in the future. In addition we have the possibility to tune our setup to give the customer the best performance according to their needs.

We have seen in the migration project that it is very important to prepare the implementation. Spending time to focus on how to implement the architecture is indeed time very well spent as this will save you a lot of time later. Changing something in your architecture will have a big impact on the developed ETL flows and many flows will most likely have to be altered. Documenting which standards to be kept during construction will also help to build uniform ETL flows that are easier to maintain and debug. These standards can also serve as reference documentation to help share knowledge among the developers.

Furthermore we have seen a drastic reduction in development time of our ETL flows. Even with just the very basic knowledge of SAS DI Studio we were able to develop ETL flows very fast. Our experience is that SAS DI Studio is very easy to use and has an intuitive user interface. Migrating an existing DataWarehouse which had been developed over a period of five years to a new platform with a new ETL tool over a period of 18 months with a team of 5 developers who have no prior experience with SAS DI Studio shows how intuitive and easy to use SAS DI Studio is.

We have no doubt that our integrating MS SQL Server with SAS was the right choice for us. Using the power of SAS for processing and analyzing large amounts of data together with MS SQL Servers ability to execute selections and retrieve data at a very fast rate, has given us an architecture that meets our performance demands, as well as supplies the users with data with very little response time. Using this setup the workload was greater and it took longer time to develop the first ETL flows, but we feel the extra work was definitely worth the effort.

We have been able to provide our customers with maximum community in an individual world. We can support both common solutions as well as individual solutions for our customers. We have implemented this into SAS Metadata Server using custom repositories in addition to the foundation repository. We have placed common solutions in the foundation repository and when the need arises for an individual solution, this is placed in a custom repository. At the same time we are able to handle both data security and metadata security for the individual solutions with very little work. This means that not only are one bank’s data completely shielded from the others, but this also applies to the bank’s individual metadata.

So we feel it is safe to say, that with SAS DI studio and MS SQL-server we are ready for the future and ready to meet whatever need out customers might have. We can comply with their business needs in a fast and flexible way, and respond to the everyday changes in a changeable world.

Contact Information

Contact the author at:

Name: Søren Junk

Enterprise: Bankdata

Address: Erritsø Bygade 102

City, State ZIP: 7000 Fredericia, Denmark

Work Phone: +45 79 24 25 77

E-mail: sju@bankdata.dk