A PC interface for Mainframe SAS Datawarehouse

Author: Heidi Fog
Agenda

- Abstract
- Background
- The SAS ODBC Driver (v6.12.00.00)
- Mainframe Set Up
- PC Configuration
- Data Traffic
- SQL Conversion
- Date Issue
- Results
- Questions
Abstract

• The SAS OBDC Driver (Version 6.12.00.00)
• SQL and “Pass-Through”
• Dates
• Response Time
Background

• AXA Australia Health Insurance insuring the health of about 800,000

• Datawarehouse development has finished

• BASE SAS®, SAS/GRAPH®, SAS/CONNECT®, SAS/ACCESS®, SAS/FSP®, SAS/SHARE®, SAS/SHARE*NET™ AND SAS/AF®

• Increasingly skilled users

• Where to from here
The SAS ODBC Driver

- ODBC, Open Data Base Connectivity
- Facility to access SAS datasets or other databases
Mainframe Set Up

- SAS Server running on the mainframe
- PROC Operate
- Allocation of datasets, viewing active users and libraries
- BASE SAS®, SAS/SHARE® and SAS/SHARE*NET™ must be installed
PC Configuration

The SAS ODBC Driver Configuration on workstations
Data Traffic

- What users did prior the SAS ODBC Driver
- Accessing datasets from the mainframe
- Sending datasets to the mainframe
SQL Conversion

- PROC SQL and Pass-Through option
- SAS V8™ versus SAS V6™

Before

```
Select dataset_dtls.admitdate
from work_dtls inner join dataset_dtls
```

After V8™

```
Select b.admitdate
from work_dtls a inner join dataset_dtls b
```

After V6™

```
Select b.date
from work_dtls a inner join dataset_dtls b
```
Date Issue

• Initial test on linking two datasets

Test1 = 21/04/1999
Test2 = Wednesday, 21 April 1999
Test3 = 21-Apr-99

• All dates = 21APRIL1999:00:00:00

• Microsoft® Access 97 dates = 1240272000

• Solution

select put(a.date,datetime7.) as newdate
.. On put(a.date,datetime7.) = put(b.date,date7.)
Result

- Response time
- Easy access to data
- Increased Flexibility
- Greater utilisation of existing datawarehouse
- Users more independent
Questions

Heidi Fog
AXA Australia Health Insurance
1/447 Collins Street
Melbourne 3000
Ph: (+61) 3 9618 4867
E-Mail: HFOG@AXA.COM.AU
A PC Interface for Mainframe SAS Datasets
Heidi Fog, AXA Australia Health Insurance, Australia Melbourne

Abstract
This presentation will go through how AXA Australia Health Insurance uses the SAS OBDC Driver (v.6.12.00.00) to access our SAS Data Warehouse datasets on the mainframe via Microsoft® Access 97. It will be looking at what is needed in setting up The SAS ODBC Driver and what the users need to know in order to access their datasets.

A special focus on SQL and “Pass-Through” has improved the use of the SAS ODBC Driver. Users have been trained in changing a Microsoft® Access 97 SQL to the PROC SQL understood by a SAS® program. After SAS® V8™ is installed this change will be at a minimum. With the help of ODBC a user can send datasets to the mainframe, where their Data Warehouse datasets reside and then via Microsoft® Access 97/(SAS)SQL link datasets on dates and then run the query with "Pass-Through" with the machine power of the mainframe.

The response time matching more than a million records with work files of several thousand records using this method is unbelievable and has taken programmers and users by surprise. We have slashed response time from 4 or 5 hours to 2 or 3 minutes. Especially linking on dates was a breakthrough.

Background
AXA Australia Health Insurance insures the health of around 800,000 in Australia and New Zealand.

AXA Australia Health Insurance has a fully developed datawarehouse containing health insurance information such as hospital, medical, ancillary, product movements and membership details. This has been developed over the previous five years.

Our datawarehouse was developed using BASE SAS®, SAS/GRAPH®, SAS/CONNECT®, SAS/ACCESS®, SAS/FSP®, SAS/SHARE® and SAS/SHARE*NET™ and our interface is built in SAS/AF®. The datawarehouse datasets reside on an IBM® mainframe, to where the users remote submit their queries via the interface.

AXA Australia Health Insurance has 40 users of the datawarehouse from these 8 users are power users and provide AXA Australia Health Insurance with 80% of reports used by management.

The demand from these power users started in September last year to exceed the capacity of our user interface. The ways of accessing data no longer meet the needs. Our interface is still used to run queries for most of our 40 users, but we have provided our “Data Farmers” with an “add-on” opportunity. Installing the SAS ODBC Driver (version 6.12.00.00) is it.

The SAS ODBC Driver
The ODBC (Open Data Base Connectivity) facility allows users to access SAS datasets, the SAS data warehouse, DATACOM tables and sequential files on a mainframe via Microsoft® products on the user workstations such as Microsoft® Access 97 and Microsoft® Excel 97.
Mainframe Set Up

Access to the SAS files is provided via a SAS server, which is a SAS® procedure PROC SERVER, which runs from its own address space on the mainframe. We call it SASSERV, mainly because that is the example used in the installation notes. Submitting a mainframe job starts the SAS server. The server remains active for a specified duration. This job includes worklibrary sizes, memory options, behavioural options and allowing users to use SAS/SHARE*NET™.

Once the server is running, administration tasks can be performed to control the server, users and libraries, via the PROC OPERATE procedure.

We use the PROC OPERATE facility via TSO SAS to administer the server.

Administration tasks include:

a) viewing users currently logged on to the server
b) allocating libraries or files (ie. SAS datasets, SAS data warehouses, etc) to be accessed via the server
c) viewing libraries currently allocated to the server
d) terminating users from the server
e) de-allocating libraries or files from the server

We recycle the server weekly to manage storage-related issues. A job makes sure that all productionised datawarehouse datasets are automatically allocated after the recycling of the server.

The products BASE SAS®, SAS/SHARE®, SAS/Share*Net™ need to be installed.

TCP/IP is used as the transfer engine used by ODBC to send data streams to and from the remote host.

Two TCP configuration files need to be copied into personal libraries so they can be edited as required.


PC Configuration

The SAS ODBC Driver needs to be installed on each workstation

The SAS server on the mainframe needs to be defined in the SERVICES file on the C-Drive.

This can look like following:

```
SASSERV 1035/tcp #SAS Share Server Production
```

**Note: do not use the same port number for different servers.**

The mainframe data source needs to be defined via the ODBC Manager. The following 3 steps are performed for each the User DSN and the System DSN.
We must define the server.

SASSERV1 is set up for development purposes, where as SASSERV is our production version.

Configure the TCP options by entering your mainframe userid. We have left the Server User Password empty, so the user does not need to change this every time the mainframe password changes. Not including the password means though that a user will be prompted a password everytime he/she connects to the server.

We must provide a data source name and description for the defined server too.

Data Traffic

Sending data to and from the mainframe.

The alternative of not installing The SAS OBDC Driver (v.6.12.00.00) was having the users downloading an enormous amount of data to the PC. Which often they initiated before they would leave the office in the evening and hope for the LAN to remain active, during the 4-7 hours millions of records would find the way to their PC. Users could do this during the day but this would lock up their session while downloading. When the download had finished the user would link this dataset to another file on the PC. This would take an additional 4 – 5 hours.

Via Microsoft® Access 97 and the ODBC link a user can get a view of which datasets are available on the mainframe. Select the one(s) of interest and bring a view of this particular
dataset back into Microsoft® Access 97. The ODBC link is also used for the user to send a work dataset up to the SAS server on the mainframe. This dataset can then be linked up to a dataset already residing on the mainframe.

**SQL Conversion**

Microsoft® Access 97 is widely used by users at AXA Australia Health Insurance. Therefore it has not been an issue of making them familiar building a PROC SQL.

When in Microsoft® Access 97 we build a query, and from the SQL View we choose the “Pass-Through” option, we send the query straight to be processed on the mainframe by SAS® products. Because we chose to “Pass-Through”, the PROC SQL we send to the mainframe must be written in SAS® code.

The changes done to the SQL in Microsoft® Access 97 depends on whether SAS® V8™ is running or not. With the capabilities for long table names the changes to a SAS® V8™ program becomes minimal.

This example shows sections of the PROC SQL before the changes in Microsoft® Access 97. Variables from the two datasets, WORK_CEDAR, which is created by the user, and HOSPEPI_DTLS, a datawarehouse dataset, have been selected and fields have been linked to form a relationship.

```
SELECT WORK_CEDAR.ADMITDATE, 
    WORK_CEDAR.DISCHDATE, 
    WORK_CEDAR.COVNUMBER, 
    HOSPEPI_DTLS.HCLAIMNO, 
    HOSPEPI_DTLS.ADMITD, 
    HOSPEPI_DTLS.DISCHD 
FROM WORK_CEDAR INNER JOIN HOSPEPI_DTLS 
ON       (WORK_CEDAR.ADMITDATE = HOSPEPI_DTLS.ADMITD) AND 
(WORK_CEDAR.DISCHDATE = HOSPEPI_DTLS.DISCHD) AND 
(WORK_CEDAR.COVNO = HOSPEPI_DTLS.COVNO);
```

After the change the PROC SQL looks like following (SAS® V8™):

```
SELECT A.ADMITDATE, 
    A.DISCHDATE, 
    A.COVNUMBER, 
    B.HCLAIMNO, 
    B.ADMITD, 
    B.DISCHD 
FROM WORK.CEDAR A INNER JOIN HOSPEPI.DTLS B 
ON       (A.ADMITDATE = B.ADMITD) AND 
(A.DISCHDATE = B.DISCHD) AND 
(A.COVNUMBER = B.COVNO);
```

In some cases Microsoft® Access 97 sets these brackets “[]”. These need to change for the SAS® V8™ to process the code. This must of course also be done for SAS® V6™ to process the code.
Should SAS® V8™ not be installed the change will look like following:

```
SELECT A.ADMDATE, 
    A.DISDATE, 
    A.COVNO, 
    B.HCLAIMNO, 
    B.ADMITD, 
    B.DISCHD 
FROM WORK.CEDAR A INNER JOIN HOSPEPI.DTLS B 
on (A.ADMDATE = B.ADMITD) AND 
    (A.DISDATE = B.DISCHD) AND 
    (A.COVNO = B.COVNO); 
```

This is actually not working, because we are linking on dates. See section on dates below.

**Dates in the SQL.**

Our initial testing for linking 2 datasets where one dataset has been sent from Microsoft® Access 97 to the mainframe created speculation on how to solve this the most efficient way.

The following shows the tests on dates, which we carried out.

In Microsoft® Access 97 we create 3 test dates with different formats.

<table>
<thead>
<tr>
<th>Obs Nr</th>
<th>Test1</th>
<th>Test2</th>
<th>Test3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21/04/1999</td>
<td>Wednesday, 21 April 1999</td>
<td>21-Apr-99</td>
</tr>
<tr>
<td>2</td>
<td>1/01/1999</td>
<td>Friday, 1 January 1999</td>
<td>01-Jan-99</td>
</tr>
<tr>
<td>3</td>
<td>2/02/1999</td>
<td>Tuesday, 2 February 1999</td>
<td>02-Feb-99</td>
</tr>
<tr>
<td>4</td>
<td>1/01/2000</td>
<td>Saturday, 1 January 2000</td>
<td>01-Jan-00</td>
</tr>
</tbody>
</table>

We need to send these observations to the mainframe. This is done by via FILE, SAVE AS/EXPORT. Save to an External File or Database. Then under Save as Type we select ODBC Databases. From there we select the SASSERV. Now we have the dataset sitting on the mainframe.

What you will see on the mainframe is that all records in our dataset for example observation 1, Test1 – Test3 sit as 21APR1999:00:00:00

To bring back a view of this work dataset, we go back to Microsoft® Access 97 and go FILE, Get External Data/Link/Tables. Under Files of Type we select ODBC Databases. We pick up the dataset from the server. The following is what Microsoft® Access 97 receives:

<table>
<thead>
<tr>
<th>Obs Nr</th>
<th>Test1</th>
<th>Test2</th>
<th>Test3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1240272000</td>
<td>1240272000</td>
<td>1240272000</td>
</tr>
<tr>
<td>2</td>
<td>1230768000</td>
<td>1230768000</td>
<td>1230768000</td>
</tr>
<tr>
<td>3</td>
<td>1230768000</td>
<td>1230768000</td>
<td>1230768000</td>
</tr>
<tr>
<td>4</td>
<td>1262304000</td>
<td>1262304000</td>
<td>1262304000</td>
</tr>
</tbody>
</table>

This is a problem, as we can not use the dates in our dataset we send to the mainframe to link with dates on another dataset. The Hours:Minutes:Seconds seems to create a problem for us. And as it says in the SAS® ODBC Driver Technical Report: User’s Guide and Programmer’s Reference, Release 6.11:

> You can only compare equivalent literals against SAS date, time or datetime values since they each have a different unit of measure. (Page 35)

The purpose of having users themselves sending work datasets back and forward to the mainframe is lost if a programmer has to go and modify the uploaded dataset with the SAS
function DATEPART() on the mainframe before a user can use the dates in the dataset for linking.

This is the way we go about solving this problem.

We need to modify the PROC SQL a bit more that what we looked at before, in our case with the group of users, this did not take long for them to learn. The example below is the one from above.

With SAS® V8™
SELECT put(A.ADMITDATE,datetime7.) as newadm,
       put(A.DISCHDATE,datetime7.) as newdisd,
       A.COVNUMBER,
       B.HCLAIMNO,
       B.ADMITD,
       B.DISCHD
FROM WORK.CEDAR A INNER JOIN HOSPEPI_DTLS B
ON put(A.ADMITDATE,datetime7.) = put(B.ADMITD,date7.) AND
   put(A.DISCHDATE,datetime7.) = put(B.DISCHD,date7.) AND
   (A.COVNUMBER = B.COVNO);

With SAS® V6™
SELECT put(a.ADMITDATE,datetime7.) as newadm,
       put(a.DISDATE,datetime7.) as newdisd,
       a.COVNO,
       b.HCLAIMNO,
       b.ADMITD,
       b.DISCHD
FROM work.cedar a INNER JOIN hospepi.dtls b
ON put(a.DISCHD,datetime7.) = put(b.DISCHD,date7.) AND
   put(a.ADMITD,datetime7.) = put(b.ADMITD,date7.) AND
   (a.COVNUMBER = b.COVNO);

Result

Users would often spend hours running queries on the PC using data, which they had spend hours downloading. When we first ran tests with the SAS ODBC Driver, we thought that we had done something wrong, as a query linking millions of records with workfiles of a couple of thousand took 15 – 20 seconds, this was the time for a straight forward link though. Most reports which are now being build using this method has seen response time slashed from 4 – 5 hours to about 2 – 3 minutes.

Users, which like getting into some level of programming, have now got easy access to their datawarehouse datasets in a new and flexible way, and it is a fantastic addition to our SAS/AF® Interface.

By having slashed response time users have now time to work on larger forecasting and development of new reports. Users have become more independent in their report making, and their PC is not locked for half a day.
References


Contact details:

Heidi Fog
AXA Australia Health Insurance
1/447 Collins Street
Melbourne 3000
Australia
Phone (+61) 3 9618 4867
Fax (+61) 3 9618 4667
E-Mail HFOG@AXA.COM.AU