USING SAS® WITH ORACLE TO ADD A REPLICA OF A LARGE OPERATIONAL DATABASE TO A DATA WAREHOUSE

Clive Cooper, Department of Social Welfare, Wellington, New Zealand
Clare Somerville, Team Comtex, Wellington, New Zealand

ABSTRACT

In 1995 the New Zealand Department of Social Welfare began the implementation of a large data warehouse on a UNIX platform. One of the major components in the warehouse was to be the replication of a 60 gigabyte database from Unisys™ DMS II into Oracle®.

SAS software played a pivotal role in the project. SAS provided the tools first to explore and scrub the data, then to automate the process of producing data definition macros and other files for various aspects of the systems software, and finally to maintain the Oracle database using SAS/ACCESS® features.

This paper covers the topics that needed to be addressed to create and then maintain the operational database replication up to date.

INTRODUCTION

The Department of Social Welfare in New Zealand consists of a number of independent business units, responsible to the Minister of Social Welfare. One of the responsibilities of this department is the assessment and payment of money to beneficiaries receiving different government pensions and subsidies. To do this, it operates a computer system called SWIFTT.

SWIFTT is a Unisys DMS II database system, based on a Quad Unisys A19. It consists of around 63 gigabytes of data, stored in over 200 different tables.

Until the advent of the department's data warehouse, the only access to this data for analysts was through a Monthly Dump process, where selected fields of certain tables were extracted from the Unisys system and made available for analysis and reporting purposes.

In 1995 the Department of Social Welfare began the development of a data warehouse designed to make previously inaccessible or unavailable data readily accessible to analysts.

The data warehouse was to have two initial components. The first was the consolidation of the historical data which had been dumped out of the SWIFTT system each month. This historical data totalled over 300 gigabytes and had been available in SAS dataset format on IBM mainframe tape cartridges. The consolidation of these SAS datasets from 300 gigabytes to around 40 gigabytes, by the removal of duplicate data, was the subject of a presentation at last year's SUGI 21: "Moving Legacy Data to a Data Warehouse".

The second phase of development is the replication of all the department's operational systems. By far the largest and most important operational system in the department is the SWIFTT system. Because it is so central to the department, and the data has previously been available only in monthly snapshots, this system was the first system to be replicated in Oracle.

ENVIRONMENT

The data warehouse is based on a UNIX platform. Initially, the hardware was a four processor HP9000/K410 with one gigabyte of RAM and 320 gigabytes of available disk storage provided by ten disk arrays operating in RAID5. Two further disk arrays have now been added, providing a further 128 gigabytes.

The platform operating system was HP-UX version 10.10. The platform has now been moved to HP-UX 10.20 which removes the two gigabyte file size limitation, and has enabled an upgrade to K460 using four new 64 bit PA8000 processors. A dual DLT tape backup library has been added, giving one terabyte of on-line tape storage.

The warehouse platform is now equivalent to a 12 processor T500.

The database software is Oracle, version 7.3.2.2. SAS version 6.11 is used on the warehouse platform and on the warehouse users desktop. The desktop is currently running Windows 3.11, moving to Windows NT 4.0x in the near future. SAS provides access to the consolidated historical datasets, and to the Oracle database, for approximately 30 users.

REPLICATION SYSTEM DESIGN

The replication process had two stages: the first step was to extract a copy of the full operational database from DMS II; the second was to extract daily logs of all database changes, and use these logs to update the extract.

DATABridge software (described later) was used to perform the initial data extract and to maintain a copy of the daily logs. The extract would be loaded into Oracle on the warehouse platform, with the addition of certain fields. The logs would be processed to remove unwanted records, and valid “add” and “delete” records would be used to update the extract.

In the development of the system to maintain the replica database we had to consider five main issues:

1. The requirement for the operational use of the database was that it be up to date at the start of each business day.

2. The DMS II database was subject to frequent small changes involving a database re-organisation. These would have to be repeated for the replica within the time requirement described above.

3. The requirement for a customer transaction history meant that each record MODIFY on SWIFTT would be turned into a logical DELETE and physical CREATE on the data warehouse -- no data should be overwritten.

4. Earlier work during evaluation of the DATABridge software had shown that the DMS II logs contained many repeat ‘no change’ modifies for the same record during a single logical transaction.

5. The users of the data warehouse would utilise SAS/ACCESS to access the Oracle database.

The daily maintenance of the database required a number of application programs to meet these criteria. It was here that SAS provided the central role required to convert, analyse and process the daily changes from the DMS II logs. There were several design alternatives available in the replication process. Decisions had to be made on the extent to which SAS would be used, as opposed to Oracle PL/SQL.

DATABRIDGE SOFTWARE

DATABridge software was licensed from Unisys. Its purpose was to access the Unisys DMS II database by providing three facilities:

- A full extract of each table in the database
- Changes from the DMS II logs
- Documentation of the database tables.

Using DATABridge, the full extract of SWIFTT was carried out in June 1996, and took 5 days to complete. The extracted date was stored in ASCII flat files. Any changes to the tables in the database during that period of extraction were trapped in "fixup" and "update" files.

From the date of the extract, DATABridge was used to process the DMS II logs. All changes to the database tables were extracted to UNIX stream files.

UNIX FTP was used to move the extracted data and daily logs from the Unisys A series to the UNIX platform.

A text file called Lister was produced by DATABridge. The Lister text file documented all the tables in the database, with the variable names, their types, sizes, positions, decimal points, and signs.

PROCESSING THE INITIAL EXTRACT

After completing the full extract of each DMS II table, DATABridge processed the log files to identify any changes that took place during the extract. The nature of the process was such that some of these change records could be found duplicated in the initial extract. This anomaly required additional processing to remove these duplicates from the Oracle tables before beginning the update process.

At the time of the extract there was around 200 tables in the database; a total of 63 gigabytes.

DATABridge provided a comma delimited format flat file for each of the tables in the DMS II database. Two methods were considered for loading this data into Oracle: using Oracle SQL*Loader, or using SAS either via SAS/ACCESS or the SQL pass-through feature. We took note of other work done in this area (see SUGI 21 Proceedings).

Some processing of the data was required. The dates in the extract data were in DMS II format with a base of 1 January 1800. Oracle dates have a base of 1 January 4712 BC, and SAS dates have their base of 1 January 1960. Dates would require conversion before reading into SAS or Oracle. In addition to the date processing, some fields, which would be required during the update process, would need to be added and initialised. These fields included a file date, to record the date the record was found in the log, and a replacement date, which would record when the record was replaced. Indexes would then have to be created.

The different methods were tested and appraised. Using the SAS/ACCESS `append` method, the dates could be converted and indexes added in one data step. A further `proc append` would read the data into Oracle.

The SQL*Loader method would require the creation of control files to describe the tables of input data for SQL*Loader. These control files would be run, and would include the conversion of the date fields. The additional fields would be added and initialised. The indexes would then be created.

Both methods required the creation of Oracle scripts to create the tables spaces and the tables.

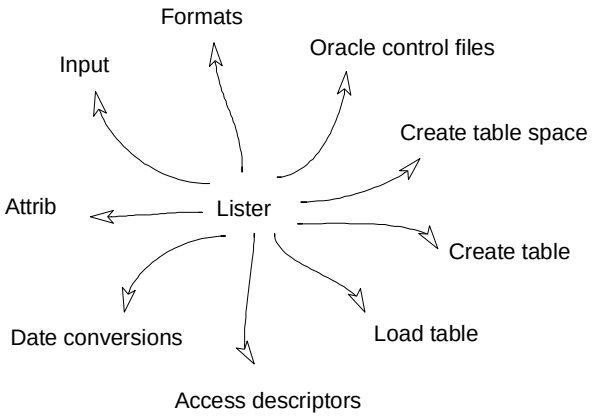
The time taken for each method was about the same -- the Metrics section discusses this in more detail. The SAS/ACCESS `append` method was chosen as it required less intervention and was therefore easier.

GENERATING SCRIPTS AND PROGRAMS

One of the outputs from DATABridge is the 'Lister' file which contains a description of all the database tables. This file is obtained electronically, and is updated each time the SWIFTT database is reorganised. After some experimentation we decided this file could provide all the information we needed to keep the system maintaining the Oracle database in step with the DMS II database tables.

The Lister file was used as a parameter file read by a set of SAS programs. These programs produced:

- Oracle SQL scripts to CREATE Tablespaces, Tables, Views, and to LOAD tables
- Control files for use by Oracle SQL*Loader
- SAS Macros containing INPUT statements for each of the log file record types (one per DMS II table)
- SAS Macro ATTRIB statements for each table
- SAS Macros to convert date fields
- SAS Formats to map table names and numbers
- SAS/ACCESS descriptors and views.



Additional formats required by the system were generated by other files. But the Lister file provided the ability to keep the Oracle system in step with the many changes made in the SWIFTT DMS II system.

LOG PROCESSING

The logs of all changes to the database were accumulated from the date of the extract. The multiple DMS II log files for each day were processed by DATABridge into one log file for each day, consisting of around 500 megabytes.

Each log record contained some additional fields: the table number, an add/delete flag, and an ID field which uniquely identified which record was being changed or added.

The records in each log could be a combination of any, or all, of the 208 SWIFTT tables. Each table had a different record layout. To process a log it was necessary to identify which record layout applied, so that it could be read and then used to update the corresponding Oracle table in the replica.

It was considered undesirable to read each log up to 208 times by the same number of SAS programs. A system was developed whereby each log file would be read twice. The first read identified all the tables to be found in that log, and created a list of SAS filenames and macro calls. On the second read, two main processes took place:

- The file was divided into individual log files; one per database table
- A SAS program was generated calling the appropriate `input` definition macro needed to read that table's data into a SAS dataset for the next stage of processing, and including `attrib` statements and date conversion macros.

Investigation of the logs revealed that they contained a lot of extraneous data. This log file 'clutter' was caused by the way the SWIFTT application maintained the source DMS II database: a database update was not just a simple delete and add of a new record. There were varying sequences of records. It was necessary to identify the 'clutter' and to identify which records were in fact new records, which were updates to already existing records, and which were deletes as opposed to updates.

These sequences could run up to 30 records for any one change. `First` and `last` processing was used to select the required records.

ORACLE UPDATE

Initially there were several months of daily logs accumulated which needed to be processed. There were two options available for clearing this backlog of logs: they could either be processed in SAS updating a SAS dataset, then appended into the Oracle database; or the extract could be read to Oracle and SAS used to update the extract in Oracle with each daily log.

Either way, the SAS programs had to be written to accommodate two different needs:

- The processing of several logs for updating a restricted selection of tables, to be used during the catch up with the backlog
- The processing of one daily log to update all the different tables, to be used once the backlog of logs was cleared and the system was running operationally.

A method of updating the extract data was developed and tested in SAS. This was done by creating a SAS dataset from the original DATABridge extract and then using SAS to apply the log changes to the dataset. The updated SAS dataset was then appended into the Oracle database.

This method was tested, and the processing times compared with the alternative method of reading the extract into Oracle, and using SAS to update the Oracle database. Early results showed that updating the Oracle database took over three times longer than updating the SAS datasets. Extensive work was done on improving these performance times (see System Metrics section in this paper).

The update process used a `modify` statement with a `select (_iorc_)` to ascertain whether a match of records between the log and the updated extract had been found:

```
data oracle.tablename ;
  ...
  modify oracle.tablename logfile ;
  by ids ;
  select (_IORC_) ;
    when %str((%sysrc(_sok))) do ;
* match located ;
    ....
    end ;
    when %str((%sysrc(_dsenmr))) do ;
* match not located ;
    ....
    end ;
    otherwise do ;
* unexpected condition ;
    ....
    end ;
run ;
```

Delete records were matched and the replacement date set; new records were added with a missing replacement date.

To confirm the update process the resulting updated dataset was compared against the Monthly Dump snapshots. The method used to compare the two sets of data required a set of data to be extracted from the Oracle database as at the date of the snapshot. The file date and replacement date fields were used to select the data. The same month was reconstituted (extracted) from the consolidated

history dataset. `Proc means` was used to produce a standard set of statistics from the data in each set. `Proc compare` was a useful tool to identify and locate data discrepancies in the testing process.

USER INTERFACE

SAS was used to provide the primary interface for the data warehouse users of the Oracle database. A set of SAS/ACCESS access and view descriptors were required for each structure.

Access descriptors and views provide a method of accessing data in different file formats, like Oracle, without using the SQL pass-through facility.

In this system, access descriptors describe the Oracle data to the SAS system. A view descriptor is used to select all, or a subset, of the data from a table and can then be referenced in a SAS program, as if it were a standard SAS dataset.

Using access descriptors, it is possible to map Oracle variable names to a matching set of SAS variable names. In this project, the users were accustomed to working with historical data variable names. They wanted to continue using the same names, saving them the trouble of unnecessary program changes, and providing the ability to easily match the new Oracle database data with the existing historical data. SAS formats were used in the creation of the access descriptors to map the Oracle variable names to the matching SAS variable names, where they existed.

Access descriptors can also be used to format variables, drop columns, and automatically generate shortened variable names where the Oracle variable name is greater than eight characters.

In this project, each table required three access descriptors. The first two descriptors were for the development team, and provided full update rights to the table. These two access descriptors had different buffsizes -- one to optimise the updating of the database, and the other to optimise the query access. One view descriptor was used to select all fields in the tables.

The third descriptor was created for the users and provided read only access. At this stage, the users have two view descriptors defined for them: one gives a view of the whole table, and the other provides a view of the fields and the subset of data they are familiar with in the historical data.

Each descriptor program, in addition to the mandatory statements, had the dates formatted and some variables renamed. The source code to create the descriptors for any one table could run into several pages.

Neither of the two standard methods of creating descriptors -- interactively or in batch -- was seen as satisfactory for this project. The large number of tables requiring descriptors, the amount of code required for each descriptor, and the need to maintain the descriptors in sync with a constantly changing database, meant that the descriptors needed to be automatically generated.

The Lister file was used to generate the access and view descriptors required for the 208 different tables. An example of the Lister file, for one of the smallest tables:

```
#070 Dataset: INCMS Format level 31 RecType 0 RecSz 8
EstRecs 2972067
  Num Name         DataType Offset Size DecSz Decimals Sign
  001 INCMS-APIND    Alpha     0000  0002  0001
  002 INCMS-COMMENT  Alpha     0002  0060  0030
  003 INCMS-MAINTS   Alpha     0062  0002  0001
  004 INCMS-AMOUNT   Number    0064  0010  0009    2      1
  005 INCMS-INCMCDE  Number    0074  0002  0002
  006 INCMS-INCMNBR  Number    0076  0009  0009
```

A SAS program reads the Lister file and uses `put` statements to output the required SAS code. For example:

```
put "proc access dbms=oracle %str(;)" ;
put "  create uporacle.&stname..access %str(;)" ;
put "  table = &stname %str(;)" ;
put "  user = 'XXX' %str(;)" ;
put "  orapw = 'XXX' %str(;)" ;
put "  path = '@IAP.world,buffsize=1' %str(;)" ;
put "  assign = yes %str(;)" ;
put "  rename " ;
...etc.
```

This results in the `proc access` source code to create descriptors for the INCMS table:

```
proc access dbms=oracle  ;
  create uporacle.INCMS.access  ;
  table = INCMS  ;
  user = 'XXX'  ;
  orapw = 'XXX'  ;
  path = '@IAP.world,buffsize=1' ;
  assign = yes  ;
  rename
    incmnbr = incno
    ;
  format
    FILEDATE = ddmmyy10.
    REPDATE = ddmmyy10.
   ;
  list all  ;
```

And two user views -- one of the full table and the other of the historical data subset:

```
* View of all fields  ;
create oracle.INCMS.view  ;
  select all  ;

* View of monthly dump fields  ;
create oracle.INCMSMD.view  ;
  subset where maints is null
    or maints ^= 'D' ;
  select
    FILEDATE
    REPDATE
    IDS
    APIND
    AMOUNT
    INCMCDE
    INCMNBR
   ;
run  ;
```

SYSTEM METRICS

During the development period a number of operational metrics have been gathered. With the development of the Oracle replication it was necessary to gather metrics for three components of the system:

- Oracle
- SAS
- HP-UX

Since the hardware was enhanced during the development process, the tests were repeated as a benchmark for performance, and to ensure the metrics were valid for comparison at different points.

At several conferences there have been papers which compared creation times for database tables against SAS data sets, particularly in the mainframe area. In this project we were also interested in understanding and optimizing the way in which SAS communicates with Oracle through the SAS/ACCESS gateways.

The metrics are given in terms of the CPU consumption and elapsed time. For SAS programs this information is provided in the log with the `-fullstimer` option set in `config.sas`.

Gathering the same type of information for Oracle is not so simple. An Oracle database involves a family of processes. A typical Oracle instance will have separate processes for writing the database files, the log files, the archive logs, a process monitor, one or more dispatchers, one or more servers. The Oracle monitor process starts and kills processes depending on demand for their services. For detailed performance analysis of the Oracle environment there are facilities in Oracle itself (trace files, dynamic details in the Data Dictionary, and the Oracle Server Manager monitor).

Other information is provided by the HP software MeasureWare products. GlancePlus is one of the components which provides a real-time display of 'the action'. MeasureWare also provides facilities to record the data for further analysis by products such as SAS/CPE.

The detailed analysis possible from all these sources is beyond the scope of this paper. A clear indication of the performance issues for Oracle and SAS on HP-UX are provided without that detail.

ORACLE ISSUES

The Oracle environment is now 7.3.2.2. The init.ora for the database has been modified significantly. The SGA (System Global Area) report shows:

Total System Global Area	331,573,104 bytes
Fixed Size	38,904 bytes
Variable Size	152,317,816 bytes
Database Buffers	178,954,240 bytes
Redo Buffers	26,714 bytes

The initialisation values that produce this size SGA follow the recommendations found in a set of tuning notes "Tuning and Performance: Tips and Sound Practices" provided by Oracle. Other parameters have been set to optimise the performance of Oracle. This includes the database block size setting which has been increased to 8K bytes.

One set of options which is significant for SAS/Oracle operation are those associated with inter-process communication (IPC). The database may also be operated in multi-threaded server (MTS) or 'dedicated' server mode. The dedicated server provides each application user (SAS user in this case) with an individual copy of the Oracle main engine. The demand for system resources, particularly memory, is high with this mode. Under MTS the dedicated server is replaced by a family of Oracle processes which support sharing of resources among a group of user applications. This approach has less resource demand and was the mode used in the initial stages of the development work.

For both modes of operation, the access to Oracle is managed through Oracle SQL*Net. There are two access protocols available, TCP or IPC. The TCP access uses the standard network access mechanism, even for processes communicating on the same machine. IPC uses UNIX Sockets and is only available to processes on the same machine as Oracle.

The following table shows the results of running the benchmark program to test these inter-process communication methods:

Compare of Inter Process Communication
Between SAS and Oracle
(HP-UX 10.20 PA8000/180 Mhz CPUs)

	-------CPU Time------			Real Time
	User mm:ss	System mm:ss	Total mm:ss	Total mm:ss
Dedicated/TCP	1:27	0:34	2:01	11:35
Dedicated/IPC	1:17	0:28	1:45	4:11
MTS/IPC	1:21	0:30	1:51	16:54

This clearly shows the best performance comes from using the Dedicated server with IPC communication.

HP-UX ISSUES

Over the course of the project so far we have moved from HP-UX 10.01 to 10.10 to 10.20. The move to 10.10 provided facilities for large file systems (128 gigabyte). Version 10.20 provided a number of improvements: increasing the file size limit from 2 gigabytes to 128 gigabytes and support for the new PA8000 180 Mhz 64 bit CPUs.

At the move to 10.10 we also did a hardware upgrade from a K400 to a K410 with a swap of the four 100 Mhz CPUs to 120 Mhz units. The K410 was upgraded to a K460 with the installation of four PA8000 180 Mhz as part of the move to 10.20. Metrics will be given later which show the improvement gained from the latest upgrade.

Additional disk arrays have also been purchased to give a total of 12 disk arrays configured to provide four large file systems:

1 x	100 gigabytes
2 x	120 gigabytes
1 x	128 gigabytes

All the arrays operate in RAID5 mode. An appropriate disk stripping regime is employed to maximize disk access performance.

During the installation and performance tuning on Oracle the HP-UX kernel default settings were changed by applying one of the HP supplied 'tuned parameter sets'. The one used was 'oltp-monolithic'. Two changes were made to that set; SWAPMEM was set back ON and TIMESLICE was restored to 10. The main reason for the changes was to provide the shared memory size required by Oracle and also to enable the increase in processes and related services that Oracle needs.

SAS ISSUES

The SAS environment changes were in two places: those made in `config.sas` and those made for SAS/ACCESS.

In `config.sas` the variable FULLSTIMER was set as described earlier. The MEMSIZE value was set to zero for 'no limit' which actually means the limit is set by the UNIX kernel MAXDATASEGMENTSIZE value. In our kernel this value has been left at the default of 67 megabytes. The SORTSIZE variable has been set to MAX.

In addition to the installation of the SAS/ACCESS to Oracle product, there are two UNIX environment variables that are important for SAS. Both are covered in SAS documentation but not necessarily in an easy place to find. The SASORA=V7 (for an Oracle version 7 database) is essential. The TWO_TASK variable is actually an Oracle variable associated with Oracle SQL*Net operation. This value needs to be set with the name (alias) for the IPC or TCP connection being used to connect to Oracle. The name is held in the Oracle network file `tnsnames.ora` on the machine running SAS. The file contains all the information needed by an application to connect to Oracle.

SAS/ACCESS TO ORACLE ISSUES

Our approach to maintaining the Oracle database uses SAS/ACCESS descriptors and views. From the various tests we had run we were concerned about the slow performance of the update processing. In SAS documentation there is a reference to the BUFFSIZE parameter in a descriptor and a general comment that adjusting the size may be beneficial.

This parameter can have a maximum value of 32,767 (rows); the default is 25. As an experiment we tried setting it to the minimum size of one. The result was a very slow query response. We then tried an update and found the opposite -- the small value gave the fastest response. Analysis of these and other tests led to the conclusion that in our update program there is an implied query which seeks the return of a single observation. A BUFFSIZE of one suited this query. For general query use the value needed to be large to maximize the number of observations returned.

By running a number of tests with different BUFFSIZE values we found the optimal values were one for the update programs and 5,000 for the query programs. There was no significant improvement in a query response time when using the maximum value. More detailed results are provided later.

LOAD AND UPDATE OF ORACLE ISSUES

The Oracle tables could be loaded either using Oracle SQL*Loader or using SAS/ACCESS. When the Oracle tables had been loaded they would be maintained using SAS/ACCESS.

Oracle SQL*Loader may be used in a direct load mode which is very fast. However, this mode cannot be used where SQL processing is required. The data we were loading had date values which needed to be converted from the Unisys date base to the Oracle date base, so a standard load was needed.

With the quantity of data to be loaded we needed to test for the best method to use. In our case 'best' meant in the fastest time, using the least CPU and involving the least number of processing steps.

A number of tests were performed to identify the best approach. To provide an environment where other users of the system would not be impacted, nor would they impact the results of the tests, the majority of the tests were carried out during weekends or at other times when the user community were not using the machine. The objective was to standardize the results as far as is practicable in a multi tasking UNIX environment.

For the tests a small table of 32,000 rows, 9.6 megabytes in size was used. From other testing we have done the results shown here are representative for the larger tables which exceed five million rows and are in excess of a gigabyte in size.

Metrics for each of the following methods are presented:

- Creation of SAS datasets from flat files
- Update of SAS datasets from SAS datasets
- Loading Oracle tables from flat files using SQL*Loader
- Loading Oracle tables from flat files using SAS/ACCESS
- Update of Oracle tables from SAS datasets using SAS/ACCESS

The following table shows the results of the tests:

Compare CPU and Elapsed Times -- SAS/Oracle
(HP-UX 10.10 and PA7200/120 Mhz CPUs)

	CPU Time hh:mm:ss	Elapsed Time hh:mm:ss
Create SAS Dataset	0:00:10	0:00:14
Update SAS Dataset	0:01:02	0:03:56
SAS/ACCESS Append into Oracle	0:02:19	0:30:57
Total	0:03:31	0:34:17
Load Oracle:		
- SQL*Loader	0:00:13	0:00:48
- Update Filedate	0:00:08	0:00:08
- Create Index	0:00:03	0:00:03
- Update using SAS/ACCESS	0:03:08	0:20:00
Total	0:03:32	0:20:59
Update Oracle BUFFSIZE effects:		
BUFFSIZE = 25 (default)	0:03:24	
BUFFSIZE = 32767 (max)	1:07:04	
BUFFSIZE = 1 for update and 5000 for query	0:03:08	

Note: SAS/ACCESS update CPU time does not include any of the Oracle database CPU time.

It was decided to use SAS to both load and then update the Oracle database. The table also shows the difference in CPU resource consumed between creating and maintaining a stand alone SAS dataset as against an Oracle table. It must also be acknowledged that a component in the Oracle overhead comes from its provision of recovery logging and associated features. These features are not currently built into SAS datasets.

Testing and monitoring is continuing to help identify further improvements in the processing cycle. Issues over the behaviour of the `modify` process and the way it operates when using SAS/ACCESS are being discussed with SAS Institute as part of this work.

CONCLUSION

This paper describes the way in which SAS was used to create and maintain the replica of a large operational database. SAS provided the tool to manage the data manipulation, and a method for maintaining the Oracle database.

In addition, it provided a smooth user interface. SAS was used to match the Oracle data to the existing standards of variable names and formats, providing a seamless and transparent interface with the Oracle data.

References

SAS Institute Inc., *SAS/ACCESS® Interface to Oracle®: Usage and reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc., 1993. 261pp

SAS Institute Inc., *SAS/ACCESS® Software for Relational Databases:, Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1994.

Oracle 7 Server Reference, Release 7.3, January 1996, Part No. A32589-1

SUGI 21 Proceedings of the Twenty-First Annual *SAS® Users Group International Conference,* Chicago, Illinois March 10-13 1996

Author Contact:

Clive Cooper
Data Administrator
Information Systems Co-ordination Unit
Department of Social Welfare
Private Bag 21
Wellington
New Zealand
Phone 64 4 25 444 144
Fax 64 4 916 3916
email clic@actrix.gen.nz
email clive.cooper@dsw.govt.nz

Clare Somerville
Senior Consultant
Software Solutions
Team Comtex
PO Box 2390
Wellington
New Zealand
Phone 64 4 25 501 575
Fax: 64 4 472 6796
email clares@actrix.gen.nz
email clare.somerville@dsw.govt.nz

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. Unisys is the registered trademark or trademark of Unisys Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.