VARIOUS METHODS FOR UPDATING ORACLE TABLES USING SAS® ACCESS TO ORACLE®, EVALUATING EFFICIENCY
Brian Mayeux, Mercer Human Resource Consulting, Phoenix, Arizona

ABSTRACT
At Mercer Human Resource Consulting Inc., it is often necessary to update our Medical Insurance Claims Oracle Data Warehouse. To identify the Oracle records to be updated, identification (ID) variables are typically matched between a SAS data set and an Oracle table. Four methods are introduced for updating Oracle records from a SAS data set. The processing efficiency of the four methods is quantified and sample code for each method is given. An optimal updating method is recommended.

INTRODUCTION
Purpose and Scope
It is often necessary for our Medical Insurance Claims Oracle Data Warehouse to be updated. The number of updates per request is typically in the thousands, however, it is common to have our updates be in the millions. Because of the tremendous numbers of updates and tight client deadlines we need to minimize claims processing time. We have found that updating and deleting Oracle records has been the largest processing time challenge. There have been some instances where updates could have taken weeks or even months if circumventing inefficient data processing methods did not occur.

This paper will discuss three distinct methods of updating Oracle records and evaluate each method's processing efficiency. A fourth method will be introduced that uses one of two methods depending on number of updates taking place. This fourth method will be presented after the efficiency results are given.

Computer Architecture and Software
SAS version 8.2 resides on it's own server that is running Microsoft Windows 2000. The SAS Server has four 1 GHz processors and 8 GB of RAM. Oracle version 8i resides on it's own server that is running Microsoft Windows 2000. The Oracle Server has four 500 MHz processors and 4 GB of RAM. PC clients running various Microsoft Windows versions access the SAS server using SAS/CONNECT. SAS ACCESS to ORACLE resides only on the SAS Server. The wiring between all servers and data storage is fiber-gigabit.

METHODS
Update Oracle Example
The task is to update insurance paid amounts (variable named PAID_AMT) for certain combinations of variables CLAIM_ID and LINE_NO in an Oracle Table. The values for CLAIM_ID and LINE_NO to be updated reside in a SAS data set. The SAS data set has the following attributes:

1) Name: UPD_SAS.
2) Number of Records: up to 2.5 million.
3) Number of variables: 3.

The Oracle Table has the following attributes:
1) Name: CLAIMS.
2) Number of Records: 6.7 million.
3) Number of Variables: 16.

The SAS data set UPD_SAS and the Oracle table CLAIMS both have a composite primary key of CLAIM_ID and LINE_NO. Therefore only unique value combinations of CLAIM_ID and LINE_NO can exist in both of these data sets.

Method 1: SAS-to-Oracle Method
In Method 1, the SAS data set UPD_SAS updates the Oracle table CLAIMS directly using the following PROC SQL code:

LIBNAME
ora
oracle
user=dcgwwas_dev
password=**********
pool=phx_dev
schema=dcgwwas_dev;
run;
libnarne dc_clm 'N:\DCGWAS\test';
PROC SQL;
UPDATE ora.CLAIMS as a
SET paid_amt =
(SELECT b.paid_amt
FROM dc_clm.UPD_SAS as b
WHERE a.claim_id=b.claim_id
and a.line_no =b.line_no)
WHERE exists
(select * from dc_clm.UPD_SAS as b
WHERE a.claim_id=b.claim_id and
a.line_no =b.line_no);
QUIT;

Method 2: One-by-One SAS-to-Oracle Method
Method 2 consists of the following steps.
1) The number of updates (records in UPD_SAS) are determined and set to the macro variable NUM.
2) One update iteration is executed for each update (record in UPD_SAS) going from 1 to NUM.
3) Values of PAID_AMT, CLAIM_ID, and LINE_NO are copied to macro variables P, C, and L respectively.
4) Field PAID_AMT in Oracle table CLAIMS is assigned the value for the macro variable P where Oracle variables CLAIM_ID equals macro variable C, and Oracle variable LINE_NO equals macro variable L.

The code in macro form is as follows (libnames are as in Method 1 code):

%MACRO method2(sas_dat,ora_dat);
The macro variables SAS_DAT and ORA_DAT represent the SAS Data set that contains the update records and the Oracle table that is to be updated, respectively.

Method 3: Oracle-to-Oracle Method
In Method 3, the SAS data set UPD_SAS is first copied to an Oracle table called UPD_ORA that resides in the same database and schema as CLAIMS. Then using the SQL Pass-through facility, the Oracle table CLAIMS is updated using values from the Oracle table UPD_ORA. After the updating is done the Oracle table UPD_ORA is truncated for later use. The code for Method 3 is as follows (libnames are as in Method 1 code):

```
PROC SQL;
    insert into ora.UPD_ORA
    select claim_id,
           line_no,
           paid_amt
    from dc_clm.UPD_SAS;
QUIT;

PROC SQL;
    connect to oracle (user=dcgwas_dev orapw=**********);
    execute (truncate table UPD_ORA) by oracle;
    QUIT;
```

Oracle tables UPD_ORA and CLAIMS both must have a primary composite key and a composite index of CLAIM_ID and LINE_NO. The absence of this composite index in either one of the Oracle tables will drastically increase the update processing time.

Test Runs
Test runs for the task described in this (the Methods) Section were performed for all three methods and for eleven different update record quantities.

It was impossible for us to perform the test runs on a server free from other processes. Therefore the results may be shifted faster or slower by approximately 10% depending on server processing load. Test runs were performed for each number of updates around the same time. Therefore, comparisons between methods (for a fixed number of updates) should not have been affected that much by server processing load.

RESULTS
Test-Run Results
The following table displays the update processing time for various amounts of updates:

<table>
<thead>
<tr>
<th>Updates</th>
<th>Method 1: Sas-to-Oracle</th>
<th>Method 2: One-By-One</th>
<th>Method 3: Oracle-to-Oracle</th>
<th>Method 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.12</td>
<td>0.00</td>
<td>4</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>11.67</td>
<td>0.02</td>
<td>4.6</td>
<td>0.02</td>
</tr>
<tr>
<td>50</td>
<td>14.83</td>
<td>0.07</td>
<td>4.2</td>
<td>0.07</td>
</tr>
<tr>
<td>100</td>
<td>19.78</td>
<td>0.15</td>
<td>4.25</td>
<td>0.15</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>1.60</td>
<td>4.5</td>
<td>1.80</td>
</tr>
<tr>
<td>10000</td>
<td>862</td>
<td>18.60</td>
<td>5.26</td>
<td>6.26</td>
</tr>
<tr>
<td>50000</td>
<td>76.20</td>
<td>3.51</td>
<td>3.51</td>
<td>3.51</td>
</tr>
<tr>
<td>100000</td>
<td>154.20</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>500000</td>
<td>936.00</td>
<td>11.9</td>
<td>11.9</td>
<td>11.9</td>
</tr>
<tr>
<td>1000000</td>
<td></td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>2500000</td>
<td></td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
</tbody>
</table>

A log-log plot of processing time by number of updates is given on the next page. Test runs were not done for method 1 beyond 10,000 updates and method 2 beyond 500000 updates since it would have taken more than 1 day to accomplish each run.
The results show the following:

1) Method 1 has the longest processing time of all methods across all update amounts.
2) Method 2 uses the least processing time for all methods across all methods for under approximately 3000 updates.
3) Method 3 uses the least processing time for over approximately 3000 updates.

An optimal method is to use Method 2 for less than 3000 updates and Method 3 for more than 3000 updates. This method is referred to as Method 4 and the results for this method are given in the above table. The code for this method is as follows (libnames are as in Method 1 code):

```sas
%MACRO method4(sas_dat,ora_dat,upd_ora);
proc contents noprint data=&sas_dat
  out=templ(keep=nobs);
run;

data _null_; set templ;
call symput('num',nobs);
run;

%IF &num ge 3000 %THEN %DO;
PROC SQL;
  insert into ora.&upd_ora
    select claim_id,
        line_no,
    from dc_clm.&sas_dat;
QUIT;
PROC SQL;
  connect to oracle (user=dcgwas_dev
  orapw=**********);
  execute;
    UPDATE CLAIMS a
    SET paid_amt = (select b.paid_amt
                     from ora.upd_ora b
                     WHERE a.claim_id=b.claim_id and
                     a.line_no =b.line_no)
    WHERE exists (select * from
                   ora.upd_ora b
                   WHERE a.claim_id=b.claim_id and
                   a.line_no =b.line_no )
    by oracle;
QUIT;
PROC SQL;
  connect to oracle (user=dcgwas_dev
  orapw=**********);
  execute (truncate table UPD_ORA) by oracle;
quit;
%END;

%ELSE %DO
  **** Method 2;
  %method2(dc_clm.UPD_SAS,ora.CLAIMS);
%END;
%MEND method4;
```

```sas
paid_amt
from dc_clm.&sas_dat;
QUIT;
PROC SQL;
  connect to oracle (user=dcgwas_dev
  orapw=**********);
  execute;
    UPDATE CLAIMS a
    SET paid_amt = (select b.paid_amt
                     from ora.upd_ora b
                     WHERE a.claim_id=b.claim_id and
                     a.line_no =b.line_no)
    WHERE exists (select * from
                   ora.upd_ora b
                   WHERE a.claim_id=b.claim_id and
                   a.line_no =b.line_no )
    by oracle;
QUIT;
PROC SQL;
  connect to oracle (user=dcgwas_dev
  orapw=**********);
  execute (truncate table UPD_ORA) by oracle;
quit;
%END;

%ELSE %DO
  **** Method 2;
  %method2(dc_clm.UPD_SAS,ora.CLAIMS);
%END;
%MEND method4;
```
CONCLUSION
Because of tight client deadlines and enormous amounts of update requests, update processing time reduction is an urgent issue at Mercer Human Resource Consulting as well as other businesses that perform Data Warehousing.

We have researched three methods for updating Oracle tables using SAS Access to Oracle. The best method is one that utilizes the strengths of two methods. For less than approximately 3000 updates, it is best to update an Oracle table one-by-one (iteratively) using the SAS macro given earlier for Method 2. And for more than approximately 3000 updates, it is best to first insert the update records into Oracle and perform the updates in Oracle using the SAS SQL pass-through facility. Certain indexes explained in the Method 3 section, must be in place for this Oracle-to-Oracle method to be superior for over 3000 updates.

ACKNOWLEDGMENTS
The following persons were instrumental in helping us to research and write this paper.
From Mercer Human Resource Consulting:
1) Dave Velde
2) Anabelle Reyes
From the SAS Institute:
1) Kelly Godfrey.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Brian Mayeux
Mercer Human Resource Consulting
3131 E. Camelback Rd. Suite 500
Phoenix, AZ 85016
Phone: (602) 522-6511
E-mail: Brian.Mayeux@mercer.com