Paper 027-2010

Complex Data Combinations on Large Volumes: how to get the Best Performance

Henri THEUWISSEN, BI Knowledge Sharing, Belgium

ABSTRACT

Developers are often challenged to build complex combinations of large files with a good performance.

In most cases the SQL procedure is the obvious choice. However, the DATA step often provides better alternatives with outstanding performance improvements.

An extra level of complexity is added when data can not just be combined on common key fields, but time slices or weighting parameters have to be taken into account.

INTRODUCTION

We will learn how to build solutions with a good performance for complex combinations of tables with millions of records.

We will compare the following methods:

- SQL inner joins
- MERGE - BY statements
- hash tables
- arrays
- multiple SET statements

The following business cases will be discussed:

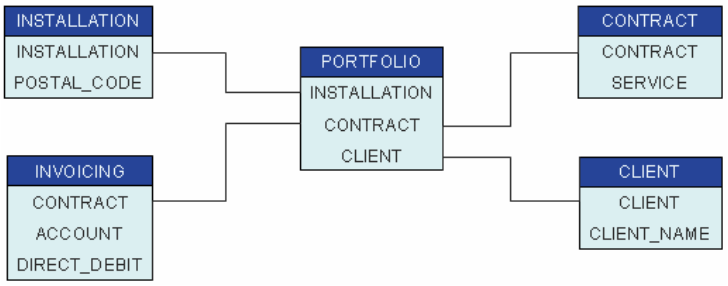
- How can we efficiently treat data in a star schema with a fact table and several dimension tables?
- How can we efficiently treat time-sliced data where every record must be split into several records using a weight variable from another table?
- How can we efficiently combine two time-sliced tables and search for gaps in the combination of these tables?

BUSINESS CASE 1

Combining a Fact Table with Several Dimension Tables

INPUT TABLES

We have data in a star schema with a fact table (PORTFOLIO) and several dimension tables (CLIENT, INSTALLATION, CONTRACT and INVOICING) with unique key values.



INSTALLATION

INSTALLATION	POSTAL_CODE
INST_001	1370
INST_002	3360
INST_003	3000

CONTRACT

CONTRACT	SERVICE
CONTR_021	ELE
CONTR_018	GAS
CONTR_310	ELE

PORTFOLIO

INSTALLATION	CONTRACT	CLIENT
INST_001	CONTR_021	CLIENT_A
INST_002	CONTR_018	CLIENT_B

INVOICING

CONTRACT	ACCOUNT	DIRECT_DEBIT
CONTR_021	ACC_3210	Y
CONTR_018	ACC_9876	N
CONTR_310	ACC_5678	Y

CLIENT

CLIENT	CLIENT_NAME
CLIENT_A	CROONEN NANCY
CLIENT_B	THEUWISSEN HENRI
CLIENT_C	NOTEN JASPER

We will combine the data from the PORTFOLIO fact table with the CLIENT, INSTALLATION, CONTRACT and INVOICING dimension tables using the following methods:

- SQL inner joins
- DATA step MERGE - BY statements
- hash tables

The resulting table will look as follows:

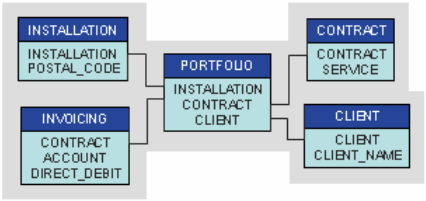
PORTFOLIO			CLIENT	CONTRACT	INSTALLATION	INVOICING	
INSTALLATION	CONTRACT	CLIENT	CLIENT_NAME	SERVICE	POSTAL_CODE	ACCOUNT	DIRECT_DEBIT
INST_001	CONTR_021	CLIENT_A	CROONEN NANCY	ELE	1370	ACC_3210	Y
INST_002	CONTR_018	CLIENT_B	THEUWISSEN HENRI	GAS	3360	ACC_9876	N

SOLUTION 1

Using SQL Inner Joins

Combine all tables at once in one query using SQL inner joins.

```
PROC SQL;
   CREATE TABLE BC1_SOL1 AS
      SELECT POR.*,
             CLI.CLIENT_NAME,
             CON.SERVICE,
             INS.POSTAL_CODE,
             INV.ACCOUNT,
             INV.DIRECT_DEBIT
         FROM PORTFOLIO    POR,
              CLIENT       CLI,
              CONTRACT     CON,
              INSTALLATION INS,
              INVOICING    INV
         WHERE POR.INSTALLATION = INS.INSTALLATION AND
               POR.CONTRACT     = CON.CONTRACT     AND
               POR.CLIENT       = CLI.CLIENT       AND
               POR.CONTRACT     = INV.CONTRACT;
QUIT;
```

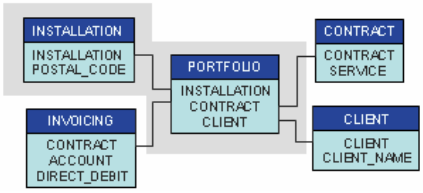


SOLUTION 2

Using DATA Step MERGE – BY Statements

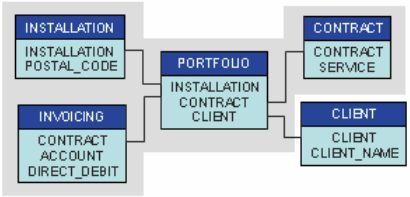
Combine the PORTFOLIO fact table with the INSTALLATION dimension table by INSTALLATION using a first DATA step MERGE statement.

```
PROC SORT DATA = PORTFOLIO;
   BY INSTALLATION;
RUN;

PROC SORT DATA = INSTALLATION;
   BY INSTALLATION;
RUN;

DATA PORTFOLIO_INSTALLATION;
   MERGE PORTFOLIO (IN = POR)
         INSTALLATION;
   BY INSTALLATION;
   IF POR;
RUN;
```



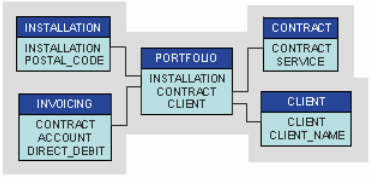
Combine the intermediate result with the CONTRACT and INVOICING dimension tables by CONTRACT using a second DATA step MERGE statement.

```
PROC SORT DATA = PORTFOLIO_INSTALLATION;
   BY CONTRACT;
RUN;

PROC SORT DATA = CONTRACT;
   BY CONTRACT;
RUN;

PROC SORT DATA = INVOICING;
   BY CONTRACT;
RUN;

DATA PORTFOLIO_INSTALLATION_CONTRACT;
   MERGE PORTFOLIO_INSTALLATION (IN = POR)
         CONTRACT
         INVOICING;
   BY CONTRACT;
   IF POR;
RUN;
```



Combine the intermediate result with the CLIENT dimension table by CLIENT using a third DATA step MERGE statement.

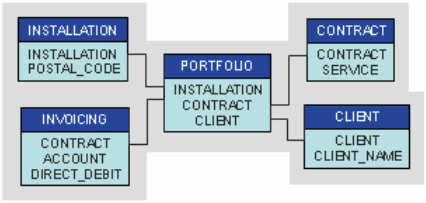
```
PROC SORT DATA = PORTFOLIO_INSTALLATION_CONTRACT;
   BY CLIENT;
RUN;

PROC SORT DATA = CLIENT;
   BY CLIENT;
RUN;

DATA BC1_SOL2;
   MERGE PORTFOLIO_INSTALLATION_CONTRACT (IN = POR)
         CLIENT;
   BY CLIENT;
   IF POR;
RUN;
```



SOLUTION 3

Using Hash Tables

Combine the PORTFOLIO fact table with all the dimension tables at once in one DATA step by loading the dimension tables into hash objects.




```
DATA BC1_SOL3;
   SET PORTFOLIO;
   LENGTH CLIENT_NAME  $ 40
          SERVICE      $  3
          POSTAL_CODE  $  4
          ACCOUNT      $  8
          DIRECT_DEBIT $  1;
   IF _N_ = 1 THEN DO;
      DECLARE HASH CLI (DATASET : 'CLIENT');  ❶
      CLI.DEFINEKEY  ('CLIENT');
      CLI.DEFINEDATA ('CLIENT_NAME');
      CLI.DEFINEDONE ( );
      DECLARE HASH CON (DATASET : 'CONTRACT');  ❷
      CON.DEFINEKEY  ('CONTRACT');
      CON.DEFINEDATA ('SERVICE');
      CON.DEFINEDONE ( );
      DECLARE HASH INS (DATASET : 'INSTALLATION');  ❸
      INS.DEFINEKEY  ('INSTALLATION');
      INS.DEFINEDATA ('POSTAL_CODE');
      INS.DEFINEDONE ( );
      DECLARE HASH INV (DATASET : 'INVOICING');  ❹
      INV.DEFINEKEY  ('CONTRACT');
      INV.DEFINEDATA ('ACCOUNT', 'DIRECT_DEBIT');
      INV.DEFINEDONE ( );
   END;
   DROP RC;
   RC = CLI.FIND ( );  ❺
   RC = CON.FIND ( );
   RC = INS.FIND ( );
   RC = INV.FIND ( );
RUN;
```

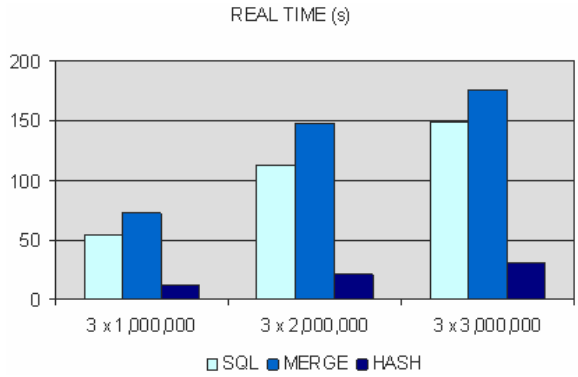
❶ Load the CLIENT table into a hash object using the CLIENT variable as the key and the CLIENT_NAME values as the data.
❷ Load the CONTRACT table into a hash object using the CONTRACT variable as the key and the SERVICE values as the data.
❸ Load the INSTALLATION table into a hash object using the INSTALLATION variable as the key and the POSTAL_CODE values as the data.
❹ Load the INVOICING table into a hash object using the CONTRACT variable as the key and the ACCOUNT and DIRECT_DEBIT values as the data.
❺ Look up the dimension values in the hash objects.

RESOURCE USAGE

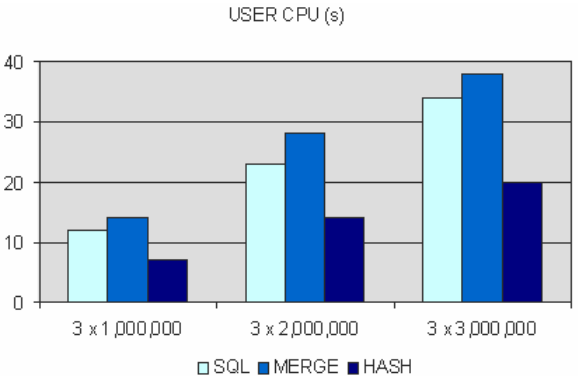
The following tables and charts illustrate resource usage in terms of:

- real time
- user CPU
- memory

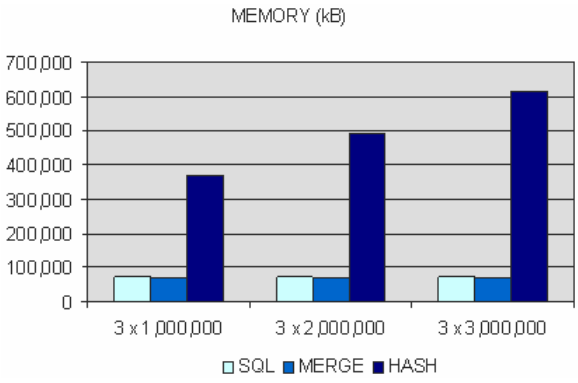
REAL TIME	SQL	MERGE	HASH
3 x 1,000,000	54 s	72 s	12 s
3 x 2,000,000	113 s	148 s	21 s
3 x 3,000,000	149 s	176 s	31 s



USER CPU	SQL	MERGE	HASH
3 x 1,000,000	12 s	14 s	7 s
3 x 2,000,000	23 s	28 s	14 s
3 x 3,000,000	34 s	38 s	20 s



MEMORY	SQL	MERGE	HASH
3 x 1,000,000	74,018 kB	68,878 kB	369,476 kB
3 x 2,000,000	74,043 kB	68,878 kB	492,372 kB
3 x 3,000,000	74,171 kB	68,878 kB	615,268 kB



BUSINESS CASE 2
Splitting Time-sliced Data Using a Weight Variable

INPUT TABLES

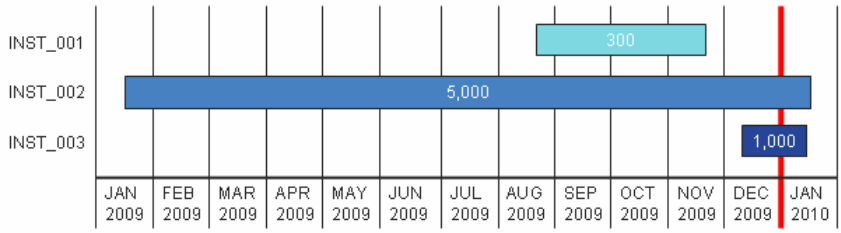
The METERING table contains the CONSUMPTION measured for an INSTALLATION between 2 dates (MET_FROM and MET_TO).

The SLP_DAY table contains the weight (SLP) for every day (DATE).

We have to split the CONSUMPTION from the METERING table into records per year using the weight variable SLP from the SLP_DAY table.

METERING

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION
INST_001	21/09/2009	23/11/2009	300
INST_002	18/01/2009	16/01/2010	5,000
INST_003	12/12/2009	11/01/2010	1,000



SLP_DAY

	DATE	SLP	
2009	01/01/2009	0.003249822	
	02/01/2009	0.003265299	
	03/01/2009	0.003389773	Σ SLP = 1
	...	...	
	31/12/2009	0.003570652	
2010	01/01/2010	0.003293231	
	...	...	
	29/12/2010	0.003360963	Σ SLP = 1
	30/12/2010	0.003317455	
	31/12/2010	0.003561818	

We will split the CONSUMPTION per year using the following steps:

1. Make the daily SLP values available using one of the following methods:
   - SQL joins
   - a hash table
   - an array
2. Calculate the total SLP value for the consumption period (TOTAL_SLP) using the following formula:

$$TOTAL\_SLP = \sum_{DATE = MET\_FROM}^{MET\_TO} SLP_{DATE}$$

3. Calculate the total yearly SLP values (YEAR_SLP) using the following formulas:

2009

$$YEAR\_SLP = \sum_{DATE = MET\_FROM}^{31/12/2009} SLP_{DATE}$$

2010

$$YEAR\_SLP = \sum_{DATE = 01/01/2010}^{MET\_TO} SLP_{DATE}$$

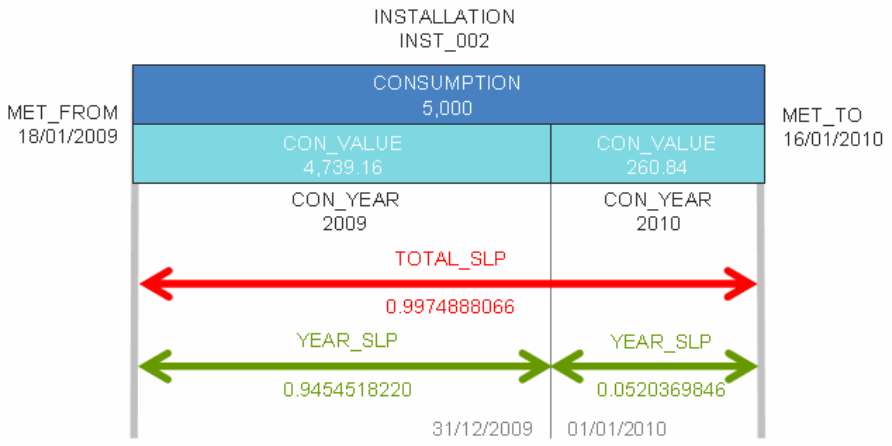
4. Split the CONSUMPTION value per year (CON_YEAR + CON_VALUE) using the following formula:

```
CON_VALUE = YEAR_SLP / TOTAL_SLP * CONSUMPTION
```

The resulting table will look as follows:

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION	CON_YEAR	CON_VALUE
INST_001	21/09/2009	23/11/2009	300	2009	300.00
INST_002	18/01/2009	16/01/2010	5,000	2009	4,739.16
INST_002	18/01/2009	16/01/2010	5,000	2010	260.84
INST_003	12/12/2009	11/01/2010	1,000	2009	650.22
INST_003	12/12/2009	11/01/2010	1,000	2010	349.78

The following table shows how the CONSUMPTION for INSTALLATION INST_002 is split across 2009 and 2010:



SOLUTION 1

Using SQL Joins

Use traditional SQL processing. Three PROC SQL steps will be required. The execution of the first two queries involves performing a Cartesian product join that can not be optimized.

Calculate the total SLP value for the consumption period.

```
PROC SQL;
   CREATE TABLE BC2_SOL1A AS
      SELECT INSTALLATION,
             MET_FROM,
             MET_TO,
             CONSUMPTION,
             SUM (SLP) AS TOTAL_SLP
         FROM METERING,
              SLP_DAY
         WHERE DATE BETWEEN MET_FROM AND MET_TO
         GROUP BY INSTALLATION,
                  MET_FROM,
                  MET_TO,
                  CONSUMPTION;
QUIT;
```

```
NOTE: The execution of this query involves performing one or more Cartesian
      product joins that can not be optimized.
```

BC2_SOL1A

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION	TOTAL_SLP
INST_001	21/09/2009	23/11/2009	300	0.2581367003
INST_002	18/01/2009	16/01/2010	5,000	0.9974888066
INST_003	12/12/2009	11/01/2010	1,000	0.1030559305

Calculate the total yearly SLP values.

```
PROC SQL;
   CREATE TABLE BC2_SOL1B AS
      SELECT INSTALLATION,
             YEAR (DATE) AS CON_YEAR,
             MET_FROM,
             MET_TO,
             SUM (SLP) AS YEAR_SLP
         FROM METERING,
              SLP_DAY
         WHERE DATE BETWEEN MET_FROM AND MET_TO
         GROUP BY INSTALLATION,
                  CALCULATED CON_YEAR,
                  MET_FROM,
                  MET_TO;
QUIT;
```

NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.

BC2_SOL1B

INSTALLATION	CON_YEAR	MET_FROM	MET_TO	TOTAL_SLP	YEAR_SLP
INST_001	2009	21/09/2009	23/11/2009	0.2581367003	0.2581367003
INST_002	2009	18/01/2009	16/01/2010	0.9974888066	0.9454518220
INST_002	2010	18/01/2009	16/01/2010	0.9974888066	0.0520369846
INST_003	2009	12/12/2009	11/01/2010	0.1030559305	0.0670087163
INST_003	2010	12/12/2009	11/01/2010	0.1030559305	0.0360472142

Split the CONSUMPTION value per year.

```
PROC SQL;
   CREATE TABLE BC2_SOL1 (DROP = TOTAL_SLP YEAR_SLP) AS
      SELECT A.*,
             CON_YEAR,
             YEAR_SLP,
             CONSUMPTION * YEAR_SLP / TOTAL_SLP AS CON_VALUE
         FROM BC2_SOL1A A,
              BC2_SOL1B B
         WHERE A.INSTALLATION = B.INSTALLATION AND
               A.MET_FROM = B.MET_FROM AND
               A.MET_TO = B.MET_TO;
QUIT;
```

BC2_SOL1

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION	CON_YEAR	CON_VALUE
INST_001	21/09/2009	23/11/2009	300	2009	300.00
INST_002	18/01/2009	16/01/2010	5,000	2009	4,739.16
INST_002	18/01/2009	16/01/2010	5,000	2010	260.84
INST_003	12/12/2009	11/01/2010	1,000	2009	650.22
INST_003	12/12/2009	11/01/2010	1,000	2010	349.78

SOLUTION 2

Using a Hash Table

Load the SLP values into a hash table.

```
DATA BC2_SOL2 (KEEP = INSTALLATION MET_FROM MET_TO CONSUMPTION CON_YEAR CON_VALUE);
   SET METERING;
   DROP RC;
   IF _N_ = 1 THEN DO;
      DECLARE HASH HT (DATASET : 'SLP_DAY');  ❶
      HT.DEFINEKEY  ('DATE');
      HT.DEFINEDATA ('SLP');
      HT.DEFINEDONE ();
   END;
   TOTAL_SLP = 0;
   DO DATE = MET_FROM TO MET_TO;
      RC = HT.FIND ();  ❷
      IF RC = 0 THEN TOTAL_SLP = TOTAL_SLP + SLP;  ❸
   END;
   CON_VALUE = 0;
   YEAR_SLP = 0;
   DO DATE = MET_FROM TO MET_TO;
      RC = HT.FIND ();
      IF RC = 0 THEN YEAR_SLP = YEAR_SLP + SLP;  ❹
      IF DATE = MET_TO THEN DO;
         CON_VALUE = YEAR_SLP / TOTAL_SLP * CONSUMPTION;  ❺
         CON_YEAR = YEAR (DATE);
         OUTPUT;
      END;
      ELSE DO;
         IF YEAR (DATE) NE YEAR (DATE + 1) THEN DO;
            CON_VALUE = YEAR_SLP / TOTAL_SLP * CONSUMPTION;  ❺
            CON_YEAR = YEAR (DATE);
            OUTPUT;
            YEAR_SLP = 0;
         END;
      END;
   END;
RUN;
```

❶ Load the SLP_DAY table into a hash object using the DATE variable as the key and the SLP values as the data.
❷ Look up the SLP values for the consumption period in the hash object.
❸ Calculate the total SLP value for the consumption period.
❹ Calculate the total yearly SLP values.
❺ Split the CONSUMPTION value per year.

BC2_SOL2

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION	CON_YEAR	CON_VALUE
INST_001	21/09/2009	23/11/2009	300	2009	300.00
INST_002	18/01/2009	16/01/2010	5,000	2009	4,739.16
INST_002	18/01/2009	16/01/2010	5,000	2010	260.84
INST_003	12/12/2009	11/01/2010	1,000	2009	650.22
INST_003	12/12/2009	11/01/2010	1,000	2010	349.78

SOLUTION 3

Using an Array

Perform a table lookup in an array that contains the SLP values.

```
PROC SQL NOPRINT;  ❶
   SELECT MIN (DATE),
          MAX (DATE)
      INTO :ARR_START,
           :ARR_STOP
      FROM SLP_DAY;
QUIT;

%LET ARR_START = &ARR_START;
%LET ARR_STOP  = &ARR_STOP;
%LET ARR_RANGE = &ARR_START : &ARR_STOP;

DATA BC2_SOL3 (KEEP = INSTALLATION MET_FROM MET_TO CONSUMPTION CON_YEAR CON_VALUE);
   SET METERING;
   FORMAT CON_VALUE COMMA12.2;
   ARRAY SLP_VAL (&ARR_RANGE) SLP_VAL_&ARR_START - SLP_VAL_&ARR_STOP;  ❷
   RETAIN SLP_VAL_&ARR_START - SLP_VAL_&ARR_STOP;
   IF _N_ = 1 THEN DO;
      DO INDEX1 = 1 TO NOBS;
         SET SLP_DAY POINT = INDEX1 NOBS = NOBS;
         SLP_VAL (DATE) = SLP;  ❸
      END;
   END;
   TOTAL_SLP = 0;
   DO INDEX2 = MET_FROM TO MET_TO;
      TOTAL_SLP = TOTAL_SLP + SLP_VAL (INDEX2);  ❹
   END;
   CON_VALUE = 0;
   YEAR_SLP = 0;
   DO INDEX3 = MET_FROM TO MET_TO;
      YEAR_SLP = YEAR_SLP + SLP_VAL (INDEX3);  ❺
      IF INDEX3 = MET_TO THEN DO;
         CON_VALUE = YEAR_SLP / TOTAL_SLP * CONSUMPTION;  ❻
         CON_YEAR = YEAR (INDEX3);
         OUTPUT;
      END;
      ELSE DO;
         IF YEAR (INDEX3) NE YEAR (INDEX3 + 1) THEN DO;
            CON_VALUE = YEAR_SLP / TOTAL_SLP * CONSUMPTION;  ❻
            CON_YEAR = YEAR (INDEX3);
            OUTPUT;
            YEAR_SLP = 0;
         END;
      END;
   END;
RUN;
```

❶ Create macro variables to specify the index range of the array.

GLOBAL SYMBOL TABLE

MACRO VARIABLE	VALUE	
ARR_START	17898	01/01/2009
ARR_STOP	18627	31/12/2010
ARR_RANGE	17898 : 18627	01/01/2009 : 31/12/2010

❷ Define the array to contain the daily SLP values.

ARRAY NAME	SLP_VAL				
ARRAY ELEMENT	1 17898	2 17899	3 178900	... ...	730 18627
ARRAY REFERENCE	SLP_VAL (17898)	SLP_VAL (17899)	SLP_VAL (17900)	...	SLP_VAL (18627)
VARIABLE	SLP_VAL_17898	SLP_VAL_17899	SLP_VAL_17900	...	SLP_VAL_18627
VALUE	0.003249822	0.003265299	0.003389773	...	0.003561818

❸ Process the array to create the SLP_VAL variables containing the daily SLP values.

PDV

SLP_VAL_17898	SLP_VAL_17899	SLP_VAL_17900	...	SLP_VAL_17900	TOTAL_SLP	_N_
0.003249822	0.003265299	0.003389773	...	0.003561818	0	1
RETAIN	RETAIN	RETAIN	...	RETAIN		

❹ Calculate the total SLP value for the consumption period.

PDV

INSTALLATION	MET_FROM	MET_TO	...	TOTAL_SLP	INDEX2
INST_001	21/09/2009	23/11/2009	...	0 + 0.0023402165 = 0.0023402165	18130
INST_001	21/09/2009	23/11/2009	...	0.0023402165 + 0.0024498335 = 0.0047900500	18131
...	...	...	...	...	...
INST_001	21/09/2009	23/11/2009	...	0.2581367003	18224

❺ Calculate the total yearly SLP values.
❻ Split the CONSUMPTION value per year.

PDV

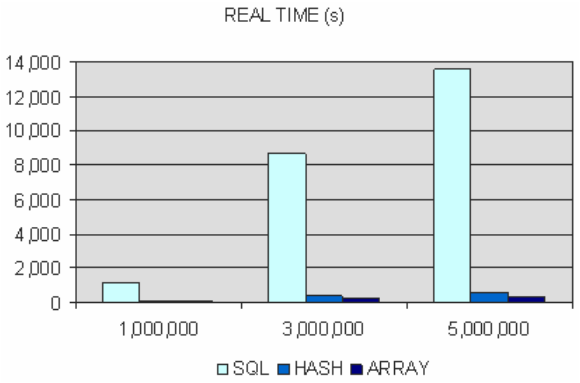
INSTALLATION	MET_FROM	MET_TO	CONSUMPTION	YEAR_SLP	CON_YEAR	CON_VALUE
INST_001	21/09/2009	23/11/2009	300	0.2581367003	2009	300.00
INST_002	18/01/2009	16/01/2010	5,000	0.9454518220	2009	4,739.16
INST_002	18/01/2009	16/01/2010	5,000	0.0520369846	2010	260.84
INST_003	12/12/2009	11/01/2010	1,000	0.0670087163	2009	650.22
INST_003	12/12/2009	11/01/2010	1,000	0.0360472142	2010	349.78

RESOURCE USAGE

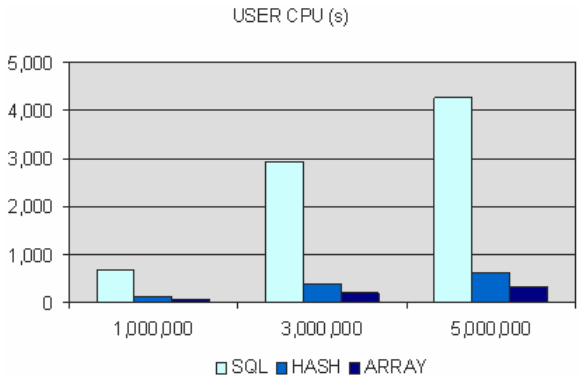
The following tables and charts illustrate resource usage in terms of:

- real time
- user CPU
- memory

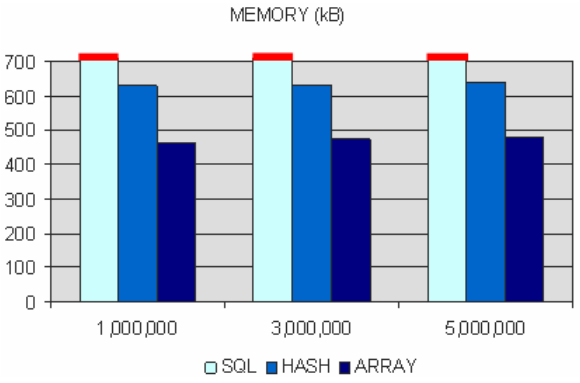
REAL TIME	SQL	HASH	ARRAY
1,000,000	1,155 s	112 s	57 s
3,000,000	8,681 s	398 s	201 s
5,000,000	13,564 s	623 s	313 s



USER CPU	SQL	HASH	ARRAY
1,000,000	663 s	111 s	56 s
3,000,000	2,944 s	396 s	199 s
5,000,000	4,261 s	621 s	310 s



MEMORY	SQL	HASH	ARRAY
1,000,000	73,628 kB	628 kB	464 kB
3,000,000	73,628 kB	632 kB	474 kB
5,000,000	73,628 kB	636 kB	480 kB



BUSINESS CASE 3
Combining Two Time-sliced Tables

INPUT TABLES

The CONTRACT table contains the period (CON_FROM – CON_TO) that an INSTALLATION is active.

The METERING table contains the CONSUMPTION measured for an INSTALLATION between 2 dates (MET_FROM and MET_TO).

We have to search for the periods in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

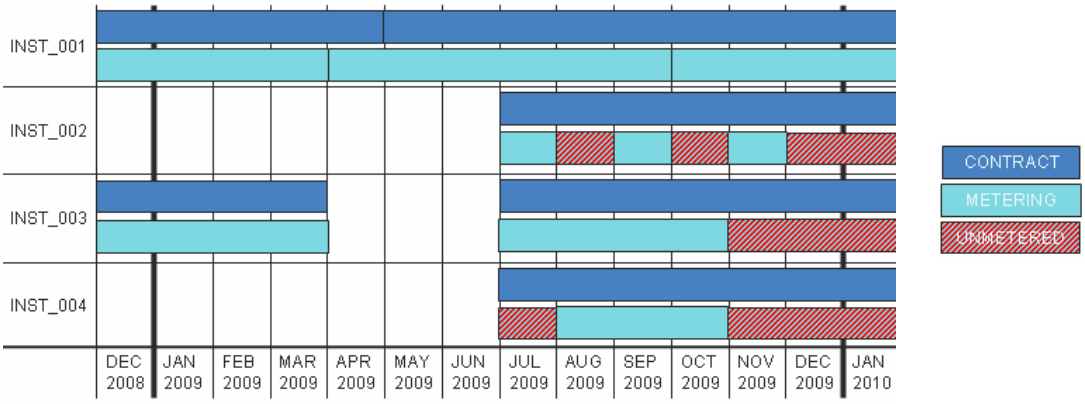
CONTRACT

INSTALLATION	CON_FROM	CON_TO
INST_001	01/07/2008	30/04/2009
INST_001	01/05/2009	31/12/2012
INST_002	01/07/2009	31/12/2010
INST_003	01/12/2008	31/03/2009
INST_003	01/07/2009	31/12/2010
INST_004	01/07/2009	31/12/2010

METERING

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION
INST_001	01/10/2008	31/03/2009	100
INST_001	01/04/2009	30/09/2009	80
INST_001	01/10/2009	31/03/2010	150
INST_002	01/07/2009	31/07/2009	1,000
INST_002	01/09/2009	30/09/2009	2,000
INST_002	01/11/2009	30/11/2009	2,500
INST_003	01/12/2008	31/03/2009	1,000
INST_003	01/07/2009	31/10/2009	700
INST_004	01/08/2009	31/10/2009	800

The following table shows the periods in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured:



The resulting table will look as follows:

UNMETERED

INSTALLATION	UNM_FROM	UNM_TO
INST_002	01/08/2009	31/08/2009
INST_002	01/10/2009	31/10/2009
INST_002	01/12/2009	31/12/2009
INST_003	01/11/2009	31/12/2009
INST_004	01/07/2009	31/07/2009
INST_004	01/11/2009	31/12/2009

SOLUTION 1

Using the DATA Step MERGE – BY Statement

Use DATA step MERGE – BY statements to search for the periods in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

```
DATA _NULL_;  ❶
   CALL SYMPUT ('START_RANGE', TRIM (LEFT ('01JAN2009'D)));
   CALL SYMPUT ('END_RANGE', TRIM (LEFT ('31DEC2009'D)));
RUN;

PROC SORT DATA = CONTRACT;
   BY INSTALLATION CON_FROM;
RUN;

DATA CON_DAY (KEEP = INSTALLATION DATE);  ❷
   SET CONTRACT;
   DO DATE = MAX (&START_RANGE, CON_FROM) TO MIN (&END_RANGE, CON_TO);
      OUTPUT;
   END;
RUN;

PROC SORT DATA = METERING;
   BY INSTALLATION MET_FROM;
RUN;

DATA MET_DAY (KEEP = INSTALLATION DATE);  ❸
   SET METERING;
   DO DATE = MAX (&START_RANGE, MET_FROM) TO MIN (&END_RANGE, MET_TO);
      OUTPUT;
   END;
RUN;

DATA BC3_SOL1 (KEEP = INSTALLATION UNM_FROM UNM_TO);
   MERGE CON_DAY (IN = C) MET_DAY (IN = M);  ❹
   BY INSTALLATION DATE;
   IF C AND NOT M;  ❺
   RETAIN UNM_FROM PREV_DATE .;
   FORMAT UNM_FROM UNM_TO DDMMYY10.;
   IF FIRST.INSTALLATION THEN DO;
      UNM_FROM = DATE;
      PREV_DATE = DATE;
   END;
   IF DATE - PREV_DATE GT 1 THEN DO;
      UNM_TO = PREV_DATE;
      OUTPUT;  ❻
      UNM_FROM = DATE;
   END;
   IF LAST.INSTALLATION THEN DO;
      UNM_TO = DATE;
      OUTPUT;  ❻
   END;
   PREV_DATE = DATE;
RUN;
```

❶ Create macro variables to restrict the analysis to 2009.

GLOBAL SYMBOL TABLE

MACRO VARIABLE	VALUE	
START_RANGE	17898	01/01/2009
END_RANGE	18262	31/12/2009

❷ Split the CONTRACT table per day in 2009.

CONTRACT

INSTALLATION	CON_FROM	CON_TO
INST_001	01/07/2008	30/04/2009
INST_001	01/05/2009	31/12/2012
INST_002	01/07/2009	31/12/2010
INST_003	01/12/2008	31/03/2009
INST_003	01/07/2009	31/12/2010
INST_004	01/07/2009	31/12/2010

CON_DAY

INSTALLATION	DATE
INST_001	01/01/2009
INST_001	02/01/2009
…	…
INST_001	30/04/2009
INST_001	01/05/2009
INST_001	02/05/2009
…	…
INST_001	31/12/2009
INST_002	01/07/2009
INST_002	02/07/2009
…	…
INST_002	31/12/2009
…	…

❸ Split the METERING table per day in 2009.

METERING

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION
INST_001	01/10/2008	31/03/2009	100
INST_001	01/04/2009	30/09/2009	80
INST_001	01/10/2009	31/03/2010	150
INST_002	01/07/2009	31/07/2009	1,000
INST_002	01/09/2009	30/09/2009	2,000
INST_002	01/11/2009	30/11/2009	2,500
INST_003	01/12/2008	31/03/2009	1,000
INST_003	01/07/2009	31/10/2009	700
INST_004	01/08/2009	31/10/2009	800

MET_DAY

INSTALLATION	DATE
INST_001	01/01/2009
INST_001	02/01/2009
…	…
INST_001	31/03/2009
INST_001	01/04/2009
INST_001	02/04/2009
…	…
INST_001	30/09/2009
INST_001	01/10/2009
INST_001	02/10/2009
…	…
INST_001	31/12/2009
…	…

❹ Merge the CON_DAY and MET_DAY tables by INSTALLATION and DATE.
❺ Only keep the observations from the CON_DAY table that do not exist in the MET_DAY table.
❻ Output only one observation per period (UNM_FROM – UNM_TO).

INSTALLATION	DATE
INST_002	01/08/2009
INST_002	02/08/2009
…	…
INST_002	31/08/2009
INST_002	01/10/2009
INST_002	02/10/2009
…	…
INST_002	31/10/2009
INST_002	01/12/2009
INST_002	02/12/2009
…	…
INST_002	31/12/2009
…	…

BC3_SOL1

INSTALLATION	UNM_FROM	UNM_TO
INST_002	01/08/2009	31/08/2009
INST_002	01/10/2009	31/10/2009
INST_002	01/12/2009	31/12/2009
INST_003	01/11/2009	31/12/2009
INST_004	01/07/2009	31/07/2009
INST_004	01/11/2009	31/12/2009

SOLUTION 2

Using Arrays

Use arrays to search for the periods in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

```
DATA _NULL_;  ❶
   CALL SYMPUT ('START_RANGE', TRIM (LEFT ('01JAN2009'D)));
   CALL SYMPUT ('END_RANGE', TRIM (LEFT ('31DEC2009'D)));
RUN;

DATA BC3_SOL2 (KEEP = INSTALLATION UNM_FROM UNM_TO);
   MERGE CONTRACT (IN = C) METERING;  ❷
   BY INSTALLATION;
   IF C;
   FORMAT UNM_FROM UNM_TO DDMMYY10.;
   RETAIN CON_&START_RANGE - CON_&END_RANGE ' '
          MET_&START_RANGE - MET_&END_RANGE ' '
          UNM_&START_RANGE - UNM_&END_RANGE;
   ARRAY CON (&START_RANGE : &END_RANGE) $ 1 CON_&START_RANGE - CON_&END_RANGE;  ❸
   ARRAY MET (&START_RANGE : &END_RANGE) $ 1 MET_&START_RANGE - MET_&END_RANGE;  ❹
   ARRAY UNM (&START_RANGE : &END_RANGE) $ 1 UNM_&START_RANGE - UNM_&END_RANGE;  ❺
   IF FIRST.INSTALLATION THEN DO;
      DO IR = &START_RANGE TO &END_RANGE;
         CON (IR) = '-';  ❻
         MET (IR) = '-';  ❻
         UNM (IR) = '-';  ❻
      END;
   END;
   DO IC = MAX (&START_RANGE, CON_FROM) TO MIN (&END_RANGE, CON_TO);
      CON (IC) = 'X';  ❼
   END;
   DO IC = MAX (&START_RANGE, MET_FROM) TO MIN (&END_RANGE, MET_TO);
      MET (IC) = 'X';  ❽
   END;
   IF LAST.INSTALLATION THEN DO;
      DO IC = &START_RANGE TO &END_RANGE;
         IF CON (IC) = 'X' AND MET (IC) NE 'X' THEN UNM (IC) = 'X';  ❾
      END;
      DO IC = &START_RANGE TO &END_RANGE;
         IF UNM (IC) = 'X' THEN DO;
            IF (IC NE &START_RANGE AND UNM (IC - 1) = '-')
               OR IC = &START_RANGE THEN UNM_FROM = IC;
            IF (IC NE &END_RANGE AND UNM (IC + 1) = '-')
               OR (IC = &END_RANGE) THEN DO;
               UNM_TO = IC;
               OUTPUT;  ❿
            END;
         END;
      END;
   END;
RUN;
```

❶ Create macro variables to restrict the analysis to 2009.

GLOBAL SYMBOL TABLE

MACRO VARIABLE	VALUE	
START_RANGE	17898	01/01/2009
END_RANGE	18262	31/12/2009

❷ Merge the CONTRACT and METERING tables by INSTALLATION.

CONTRACT

INSTALLATION	CON_FROM	CON_TO
INST_001	01/07/2008	30/04/2009
INST_001	01/05/2009	31/12/2012
INST_002	01/07/2009	31/12/2010
INST_003	01/12/2008	31/03/2009
INST_003	01/07/2009	31/12/2010
INST_004	01/07/2009	31/12/2010

METERING

INSTALLATION	MET_FROM	MET_TO	CONSUMPTION
INST_001	01/10/2008	31/03/2009	100
INST_001	01/04/2009	30/09/2009	80
INST_001	01/10/2009	31/03/2010	150
INST_002	01/07/2009	31/07/2009	1,000
INST_002	01/09/2009	30/09/2009	2,000
INST_002	01/11/2009	30/11/2009	2,500
INST_003	01/12/2008	31/03/2009	1,000
INST_003	01/07/2009	31/10/2009	700
INST_004	01/08/2009	31/10/2009	800

NOTE: MERGE statement has more than one data set with repeats of BY values.

❸ Define the array CON based on the CONTRACT table to contain the days in 2009 that an INSTALLATION is active.

ARRAY NAME	CON				
ARRAY ELEMENT	1 17898	2 17899	3 178900	... ...	365 18262
ARRAY REFERENCE	CON (17898)	CON (17899)	CON (17900)	...	CON (18262)
VARIABLE	CON_17898	CON_17899	CON_17900	...	CON_18262

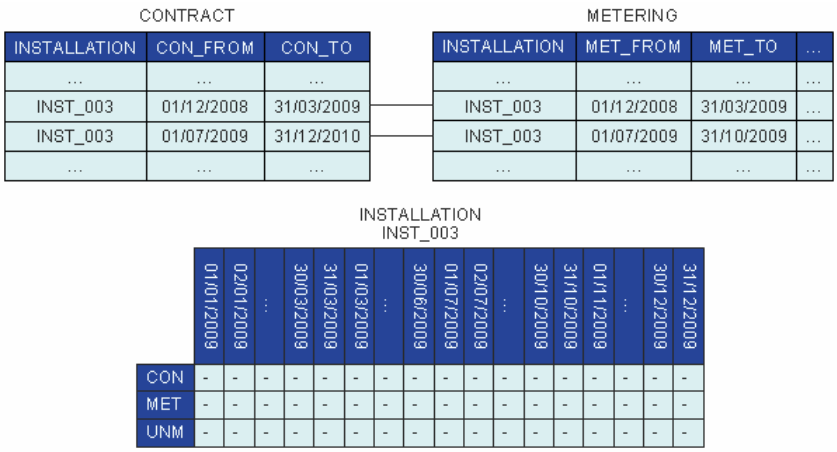
❹ Define the array MET based on the METERING table to contain the days in 2009 that CONSUMPTION has been measured.

ARRAY NAME	MET				
ARRAY ELEMENT	1 17898	2 17899	3 178900	... ...	365 18262
ARRAY REFERENCE	MET (17898)	MET (17899)	MET (17900)	...	MET (18262)
VARIABLE	MET_17898	MET_17899	MET_17900	...	MET_18262

❺ Define the array UNM to contain the days in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

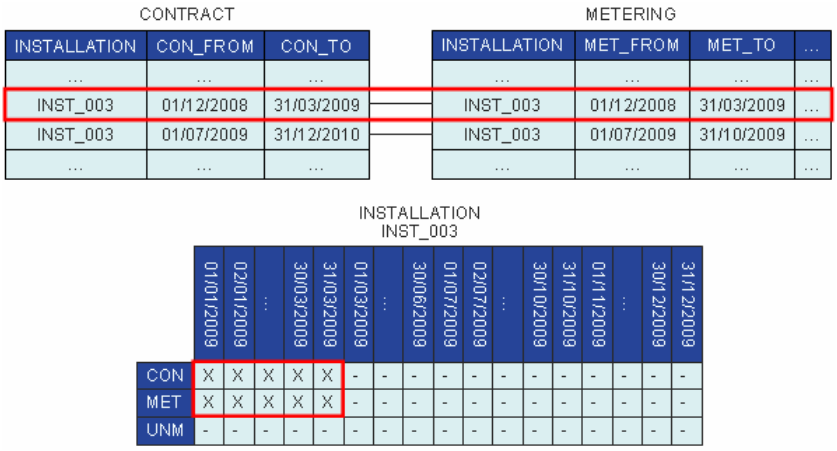
ARRAY NAME	UNM				
ARRAY ELEMENT	1 17898	2 17899	3 178900	... ...	365 18262
ARRAY REFERENCE	UNM (17898)	UNM (17899)	UNM (17900)	...	UNM (18262)
VARIABLE	UNM_17898	UNM_17899	UNM_17900	...	UNM_18262

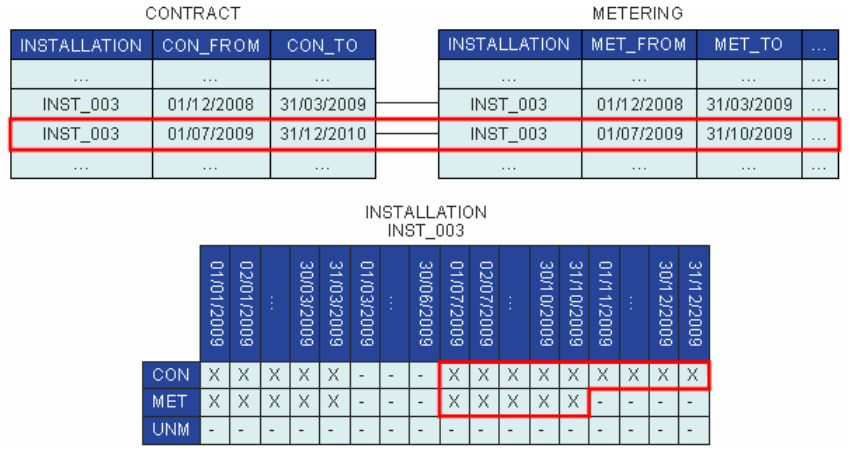
❻ Process the CON, MET and UNM arrays to initialize the CON, MET and UNM variables to "-" for each INSTALLATION.



❼ Process the CON array to assign the value "X" to the CON variables for the days in 2009 that an INSTALLATION is active.

❽ Process the MET array to assign the value "X" to the MET variables for the days in 2009 that CONSUMPTION has been measured.





❾ Process the CON, MET and UNM arrays to assign the value “X” to the UNM variables for the days in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

INSTALLATION
INST_003

	01/01/2009	02/01/2009	...	30/03/2009	31/03/2009	01/03/2009	...	30/06/2009	01/07/2009	02/07/2009	...	30/10/2009	31/10/2009	01/11/2009	...	30/12/2009	31/12/2009
CON	X	X	X	X	X	-	-	-	X	X	X	X	X	X	X	X	X
MET	X	X	X	X	X	-	-	-	X	X	X	X	X	-	-	-	-
UNM	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X

❿ Output only one observation per period (UNM_FROM – UNM_TO).

INSTALLATION
INST_003

	01/01/2009	02/01/2009	...	30/03/2009	31/03/2009	01/03/2009	...	30/06/2009	01/07/2009	02/07/2009	...	30/10/2009	31/10/2009	01/11/2009	...	30/12/2009	31/12/2009
														UNM_FROM			UNM_TO
CON	X	X	X	X	X	-	-	-	X	X	X	X	X	X	X	X	X
MET	X	X	X	X	X	-	-	-	X	X	X	X	X	-	-	-	-
UNM	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X

BC3_SOL2

INSTALLATION	UNM_FROM	UNM_TO
INST_002	01/08/2009	31/08/2009
INST_002	01/10/2009	31/10/2009
INST_002	01/12/2009	31/12/2009
INST_003	01/11/2009	31/12/2009
INST_004	01/07/2009	31/07/2009
INST_004	01/11/2009	31/12/2009

SOLUTION 3

Using Multiple SET Statements

Use multiple SET statements to search for the periods in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

```
DATA _NULL_;  ❶
   CALL SYMPUT ('START_RANGE', TRIM (LEFT ('01JAN2009'D)));
   CALL SYMPUT ('END_RANGE', TRIM (LEFT ('31DEC2009'D)));
RUN;

DATA BC3_SOL3 (KEEP = INSTALLATION UNM_FROM UNM_TO);
   SET CONTRACT;  ❷
   FORMAT UNM_FROM UNM_TO DDMMYY10.;
   RETAIN OBS 0;
   CON_FROM = MAX (&START_RANGE, CON_FROM);
   CON_TO = MIN (&END_RANGE, CON_TO);
   FLAG = 0;
   DO WHILE (FLAG = 0 AND OBS LT NOBS);
      IF INSTALLATION GT MET_INSTALLATION
         OR (INSTALLATION = MET_INSTALLATION AND CON_FROM GT MET_TO) THEN DO;
         OBS = OBS + 1;
         SET METERING (RENAME = (INSTALLATION = MET_INSTALLATION))
                      POINT = OBS NOBS = NOBS;  ❸
         MET_FROM = MAX (&START_RANGE, MET_FROM);
         MET_TO = MIN (&END_RANGE, MET_TO);
      END;
      IF MET_INSTALLATION GT INSTALLATION
         OR (MET_INSTALLATION = INSTALLATION AND MET_FROM GT CON_FROM) THEN DO;  ❹
         UNM_FROM = CON_FROM;
         IF MET_INSTALLATION = INSTALLATION THEN UNM_TO = MIN (CON_TO, MET_FROM – 1);
         ELSE UNM_TO = CON_TO;
         OUTPUT;
         IF MET_INSTALLATION GT INSTALLATION THEN FLAG = 1;
      END;
      IF FLAG = 0 AND INSTALLATION = MET_INSTALLATION THEN DO;  ❹
         IF CON_TO GT MET_TO THEN DO;
            IF CON_FROM LE MET_TO THEN CON_FROM = MET_TO + 1;
         END;
         ELSE FLAG = 1;
      END;
   END;
   IF OBS = NOBS AND FLAG = 0 THEN DO;  ❹
      IF INSTALLATION = MET_INSTALLATION THEN DO;
         IF MET_FROM GT CON_FROM THEN DO;
            UNM_FROM = CON_FROM;
            UNM_TO = MET_FROM - 1;
            OUTPUT;
            CON_FROM = MET_FROM;
         END;
         IF MET_TO LT CON_TO THEN DO;
            UNM_FROM = MET_TO + 1;
            UNM_TO = CON_TO;
            OUTPUT;
         END;
      END;
      ELSE DO;
         UNM_FROM = CON_FROM;
         UNM_TO = CON_TO;
         OUTPUT;
      END;
   END;
RUN;
```

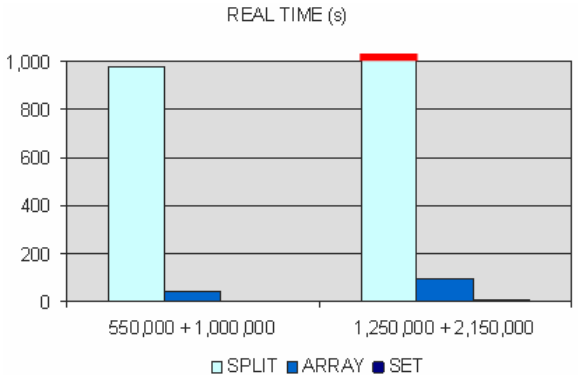
❶ Create macro variables to restrict the analysis to 2009.
❷ Read the CONTRACT table sequentially.
❸ Read the METERING table using direct access.
❹ Use complex conditional processing to search for days in 2009 that although an INSTALLATION is active no CONSUMPTION has been measured.

RESOURCE USAGE

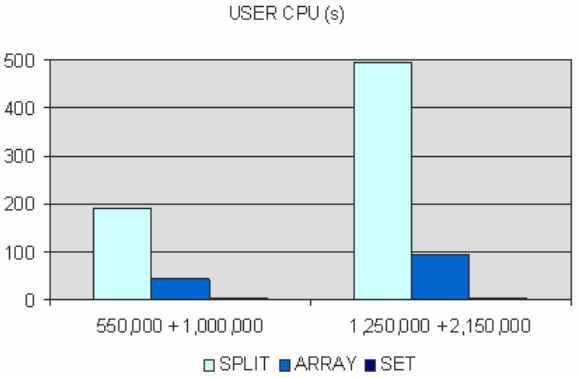
The following tables and charts illustrate resource usage in terms of:

- real time
- user CPU
- memory

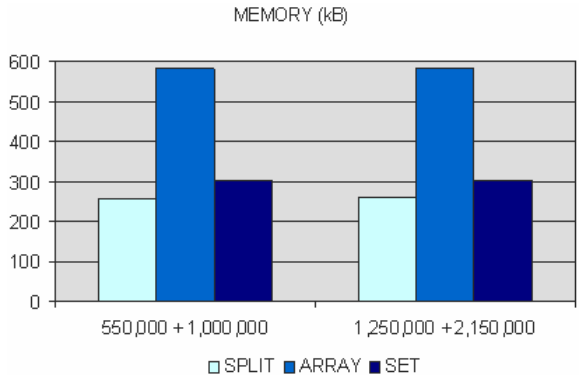
REAL TIME	SPLIT	ARRAY	SET
550,000 + 1,000,000	976 s	44 s	1 s
1,250,000 + 2,150,000	2,507 s	94 s	2 s



USER CPU	SPLIT	ARRAY	SET
550,000 + 1,000,000	191 s	43 s	1 s
1,250,000 + 2,150,000	495 s	93 s	2 s



MEMORY	SPLIT	ARRAY	SET
550,000 + 1,000,000	257 kB	582 kB	303 kB
1,250,000 + 2,150,000	259 kB	583 kB	303 kB



CONCLUSION

You should consider spending more programmer time to reduce computer resource usage for complex combinations of tables with millions of records. So, instead of using the traditional programming techniques like SQL inner joins or DATA step MERGE – BY statements, you should consider using more complex programming techniques like hash tables, arrays and multiple SET statements.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name:	Henri THEUWISSEN
Enterprise:	BI Knowledge Sharing
Address:	Sterrenlaan 40
	3360 BIERBEEK
	BELGIUM
Work Phone:	+32 (0)496 28 45 28
Fax:	+32 (0)16 46 37 74
E-mail:	henri.theuwissen@biknowledgesharing.be