SAS loader for Valid-time state tables

A valid-time state table is defined in the book „Developing Time-Oriented Database Applications in SQL” by Richard T. Snodgrass (2000) as a table that

*„records information valid at some time in the modeled reality, and it records states, that is, facts that are true over a period of time. The FROM\_DATE and TO\_DATE columns delimit the period of validity of the information in the row.”*

# Task

Suppose we make observations at a predefined time frequency (e.g. on a daily basis). At each time the observations have a natural key TK (that differentiates them from one another). We disregard the *exact time* of the state change of a given observation. Instead, we suppose that a state of a given record has been valid exactly until the time a different state has been recorded. In order to store this type of information:

* besides the fields of the natural key (TK) and other, state-relevant but not key attributes (OTH) we establish two technical fields in our table indication the *beginning* and *end* of a (closed) validity range:
  + let’s call them VALFROMD and VALTOD when they are (SAS) dates
  + or call them VALFROMDT and VALTODT when they are (SAS) datetime values
  + neither of them can take the NULL value
    - technically, VALFROMD=7000-12-31 represents a currently open validity range
* for all records VALFROMD <= VALTOD (or VALFROMDT <= VALTODT, respectively)
* as a result, {TK, VALFROMD} (or {TK, VALFROMDT}) would be a primary key in our table
* on each an every snapshot pertaining to a ponint in time T, defined by the filter (VALFROMD <= T <= VALTOD) (or VALFROMDT <= T <= VALTODT, respectively), TK must be unique

Our task is to define such a loading process that:

* given a set of NEW observations (fields: TK + OTH) pertaining to a point in time T
* loads them into the above defined table (fields: TK + OTH + validity range technical fields) so that
  + when a given observation identified by its natural key TK\_1 exists in our state table and has an open validity range („existing record”) then:
    - nothing happens if all OTH fields are the same for TK\_1
    - if the existing record differs from the new record on at least one OTH field then:
      * the existing record’s validity range should be closed at (T-1)
      * the new record should be inserted with an open validity range beginning at T
  + when a given observation identified by its natural key TK\_1 does NOT exist in our state table OR its validity is already closed by T then
    - the new record should be inserted with an open validity range beginning at T
  + when a given observation identified by its natural key TK\_2 does NOT exist in our new observation set but does exist in our state table with an open validity range then it should be decided (i.e. driven by an external parameter) whether
    - its validity range should be closed by (T-1)
    - OR it should be kept open until we obtain an observation with natural key TK\_2 and different attributes (OTH)

Through a simple example:

|  |
| --- |
|  |

# SAS solution

In our SAS solution two macros would do this general task:

1. **%etl\_load\_sas\_simple\_vfvt** would load the target table provided that the New/To-Be-Modified/To-Be-Closed records have already been separated in different temp tables
2. **%etl\_simple\_vfvt\_wrapper** would provide an easy-to-understand „interface” to the user, in which only the target and source tables, natural key and other attributes list and some other parameters should be provided

Obviously this latter calls the former while running.

Another helper macro would check the existence of one or more given datasets:

1. **%etl\_test\_connection**

## Macro %etl\_test\_connection

|  |  |  |
| --- | --- | --- |
| **Param** | **Type** | **Content** |
| inTables | MANDATORY | fully qualified table references separated with blanks |
| outResult | MANDATORY | returning macro variable |

Checks the existence of the given tables and generates return code in a predefined macro variable:

0 if all tables exist

1 if at least one table doesn't exist

nothing if a mandatory parameter was omitted

Example:

data apple; length x 8; x=1; run;

%let tmpallexists=;

%etl\_test\_connection(inTables=work.apple, outResult=tmpallexists);

%put tmpallexists=|&tmpallexists.|;

## Macro %etl\_load\_sas\_simple\_vfvt

| **Param** | **Type** | **Content** |
| --- | --- | --- |
| inTargetLib | MANDATORY | target library |
| inTargetTblName | MANDATORY | target table to be loaded/updated |
| inToCloseDs | CONDITIONAL | dataset containing records to be closed when left empty: no records' validity would be closed |
| inToModifyDs | CONDITIONAL | dataset containing records to be modified when left empty: no records would be modified (i.e. validity closed and new record with open validity range added) |
| inNewDs | CONDITIONAL | dataset containing new records when left empty: no new records will be added |
| inVFVTDateFormat | MANDATORY | DATE or DATETIME can be accepted.  DATE => {VALFROMD, VALTOD} supposed DATETIME => {VALFROMDT, VALTODT} supposed default: DATE |
| inKeyCols | MANDATORY | natural key columns separated by space WITHOUT VALFROMD(T)/VALTOD(T) cols |
| inValidDate | OPTIONAL | validity date in inVFVTDateFormat format (MUST be comparable to VALFROMD(T)-VALTOD(T) cols) default: empty when left empty then it is calculated from the system date |
| outRC | OPTIONAL | name of macro variable containig return code potential values: {SUCCESS, ERROR} |

Loads into the target table by setting record validity properties

WARNING! re-creates the target table but doesn't re-creates indices!

Supposes that

- fields indicating the validity range borders exist in Target table

- their name is {VALFROMD, VALTOD} or {VALFROMDT, VALTODT}

Example:

%let etl\_upd\_d=%sysfunc(date());

data basetable;

length valfromd valtod 8

key $3

value 8

;

format valfromd valtod yymmdd10.;

key = "K1";

valfromd = '15JAN2000'd; valtod='31JAN2005'd; value=10; output;

valfromd = '01FEB2005'd; valtod='15JAN2010'd; value=20; output;

valfromd = '16JAN2010'd; valtod='31DEC7000'd; value=30; output;

key = "K2";

valfromd = '15SEP2004'd; valtod='31DEC2004'd; value=100; output;

valfromd = '01JAN2005'd; valtod='16JAN2010'd; value=200; output;

valfromd = '17JAN2010'd; valtod='01JAN2012'd; value=300; output;

key = "K3";

valfromd = '15SEP2004'd; valtod='31DEC2004'd; value=1000; output;

valfromd = '01JAN2005'd; valtod='16JAN2010'd; value=2000; output;

valfromd = '17JAN2010'd; valtod='31DEC7000'd; value=3000; output;

run;

data toclose;

length key $3

value 8

;

format valfromd valtod yymmdd10.;

key = "K1"; output;

run;

data tomod;

length key $3

value 8

;

format valfromd valtod yymmdd10.;

key = "K3"; value=50; output;

run;

data reopen;

length key $3

value 8

;

format valfromd valtod yymmdd10.;

key = "K2"; value=500; output;

run;

data basetable\_befupdate;

set basetable;

run;

%etl\_load\_sas\_simple\_vfvt(

inTargetLib=WORK

,inTargetTblName=basetable

,inToCloseDs=WORK.toclose

,inToModifyDs=WORK.tomod

,inNewDs=WORK.reopen

,inKeyCols=key

,inValidDate=&etl\_upd\_d.

,inVFVTDateFormat=DATE

);

## Macro %etl\_simple\_vfvt\_wrapper

| **Param** | **Type** | **Content** |
| --- | --- | --- |
| inSourceLib | MANDATORY | source library |
| inSourceTblName | MANDATORY | source table to be loaded/updated |
| inTargetLib | MANDATORY | target library |
| inTargetTblName | MANDATORY | target table to be loaded/updated |
| inKeyCols | MANDATORY | key columns separated by space WITHOUT VALFROMD(T)/VALTOD(T) cols |
| inNoKeyCols | MANDATORY | NON-key columns separated by space WITHOUT VALFROMD(T)/VALTOD(T) cols |
| inCloseUnmatchedRecords | MANDATORY | whether validity of old records without an update in the new set should be closed (YES) or not (NO) default: NO |
| inVFVTDateFormat | MANDATORY | wether VALFROMD or VALFROMDT to be used DATE or DATETIME can be accepted default: DATE |
| inValidDate | OPTIONAL | validity date in inVFVTDateFormat format (MUST be comparable to valfrom-valto cols) default: empty when left empty then it is calculated from the system date |
| inIndexToRestore | OPTIONAL | indices to be restored past MERGE default: empty example: key pk=(valfromd key)/unique |
| outRC | OPTIONAL | name of macro variable containig return code potential values: {SUCCESS, ERROR} default value: tmpvfvtwrapper |

Wrapper for validity range loader (etl\_load\_sas\_simple\_vfvt)

Example:

data basetable

basetable\_befupdate

;

length valfromdt valtodt 8

key $3

value 8

;

format valfromdt valtodt yymmdd10.;

key = "K1";

valfromdt = '15JAN2000'd; valtodt='31JAN2005'd; value=10; output;

valfromdt = '01FEB2005'd; valtodt='15JAN2010'd; value=20; output;

valfromdt = '16JAN2010'd; valtodt='31DEC7000'd; value=30; output;

key = "K2";

valfromdt = '15SEP2004'd; valtodt='31DEC2004'd; value=100; output;

valfromdt = '01JAN2005'd; valtodt='16JAN2010'd; value=200; output;

valfromdt = '17JAN2010'd; valtodt='01JAN2012'd; value=300; output;

key = "K3";

valfromdt = '15SEP2004'd; valtodt='31DEC2004'd; value=1000; output;

valfromdt = '01JAN2005'd; valtodt='16JAN2010'd; value=2000; output;

valfromdt = '17JAN2010'd; valtodt='31DEC7000'd; value=3000; output;

run;

data newset;

length key $3

value 8

;

key = "K3"; value=50; output;

key = "K2"; value=500; output;

run;

%etl\_simple\_vfvt\_wrapper(

inSourceLib=WORK

,inSourceTblName=newset

,inTargetLib=WORK

,inTargetTblName=basetable

,inKeyCols=key

,inNoKeyCols=value

,inCloseUnmatchedRecords=NO

,inVFVTDateFormat=DATETIME

,inValidDate=

,inIndexToRestore=key pk=(valfromdt key)

,outRC=tmpvfvtwrapper

);