Paper 112-2012

Developing a Flexible ETL Process to Let SAS® Capture Data Changes Efficiently in a Data Warehouse and Clean Up the Mess

Stephen Overton, Zencos Consulting, Cary, NC, USA

ABSTRACT

Data sources integrated into an Enterprise Data Warehouse can and most likely will have different extraction frequencies as well as differing time granularities. One efficient way to load a data warehouse is to detect changes in data sources and only keep new data or data which has changed. This paper presents a flexible change data capture process to extract and load new data during any phase of loading a data warehouse. The process can run dynamically at any time and requires no set schedule. This paper will also demonstrate a data retention process using Base SAS®. Both processes are centrally managed and operate independent of each other.

INTRODUCTION

The Change Data Capture (CDC) process can be one of the most challenging tasks of designing an Extract, Transform, and Load (ETL) process in a data warehouse. Developers can treat each data source independently but this approach has a much longer development life cycle with more points of potential failure. This paper focuses on a standardized coding technique which synchronously filters incoming data for new observations each time the ETL process is run. Conceptually the process would remain the same if the data warehouse has additional staging areas or data stores. No matter how many data sources exist, these control tables always maintain the integrity of time-based transactional data. If data records in the source data do not have an identifying time attribute, this process will not work.

The core components of the CDC process are described first. An example Base SAS process follows to demonstrate this process using SAS datasets. Relational databases can be used as long as they are licensed and can be assigned as a SAS library. The concepts in this paper can also be implemented using custom transformations in SAS Data Integration Studio® and many other ETL development tool. Full sample code that is used in this paper can be found at the URL listed in the References section.

KEY TERMINOLOGY

The CDC process discussed in this paper uses tables which are separate from business data to manage data flow. These tables are critical to the CDC process and will be identified as control tables. Other important components which are important to the CDC process are the data sources and logic used to filter new data and update the control tables.

SOURCE DATA

When the source data is initially extracted, it may need to be converted to a delimited format if not from a relational database source or SAS dataset. The CDC process requires an identifying time attribute in the data.

There can be two different aspects of time in source data:

- **Observation specific**: Each row contains a timestamp variable which identifies the timeframe for the record. An example of this type of data would be credit card transactions which occur over time in a dataset. The transaction date would be the observation specific date.
- **Point in time:** Data is only applicable for the point of extraction or query and may not contain a timestamp. In this situation, the timestamp is derived as of the current date time for which the population of data is extracted. This timestamp represents the age of the data as a whole. An example of this type of data would be a balance sheet data for a company.

Important Note: In order to control the data flow from different sources and identify new data incrementally, you must identify the timestamp. If the source data does not have a timestamp attribute, point in time data for example, this CDC process cannot be used.

CONTROL TABLES

Control tables defined in this paper are physical tables that contain key pieces of metadata about the sources and target tables in the data warehouse, staging area, operational data store, or any landing area. This paper describes a source and table dimension that are used to control the CDC process and manage data retention respectively.

Additional columns could be added to monitor other attributes of the data or to have more control points during processing.

Source Dimension

The source dimension contains details about every logical data source used in the data warehouse. Sources can be relational database tables, external data, SAS datasets, or any logical data that is used to load a table in a data warehouse. As source data is extracted, it must be transformed into a tabular format if it is not already in a readable format.

Table 1 represents the structure of the source dimension used in this paper.

Name	Type	Length	Format	Purpose
source_key	Numeric	8	16	Primary key
source_type	Character	50	$50.00	Type of data: SAS Dataset, Text File, Oracle, DB2
source_library	Character	50	$50.00	SAS library name for reference (if applicable)
source_desc	Character	50	$50.00	Name of the data source
key_date_column	Character	50	$50.00	Control date for the data source
last_key_date_extracted*	Date	8	DATETIME22.	Last date extracted within data
last_extraction	Date	8	DATETIME22.	Point in time which data was last extracted

Table 1. Structure of Source Dimension

***** The "last_key_date_extracted" must be date or timestamp. Careful consideration must be taken into account depending on the nature of source data. If the source data uses the timestamp format, this format must be used across all sources.

Figure 1 contains an example of the source dimension used in this paper.

	source_key	source_type	source_library	source_desc	key_date_column	last_key_date_extracted	last_extraction
1	0	SAS Dataset	WORK	SOURCE_SALES	date	.	.
2	1	SAS Dataset	SGF2012	STG_SALES	last_update	.	.

Figure 1. Sample Contents of Source Dimension

Table Dimension

The table dimension contains details about every table in the data warehouse. It is not required for CDC processing but will be required for data retention management as described later in this paper in the Data Retention Cleanup section. This dimension can also contain descriptive metadata, such as ownership and last update dates.

Table 2 represents the structure of the table dimension used in this paper.

Name	Type	Length	Format	Purpose
table_key	Numeric	8	16	Primary key
library	Character	50	$50.00	SAS library name for reference (if applicable)
physical_table_name	Character	50	$50.00	Name of the physical table
table_name	Character	50	$50.00	User friendly name of the table
last_update	Date	8	DATETIME22.	Point in time which data was last updated
data_retention_days	Numeric	8	16	Number of days the data is retained

Table 2. Structure of Table Dimension

Figure 2 contains an example of table dimension used in this paper.

	table_key	dw_location	libref	physical_table_name	table_name	last_update	data_retention_days
1	0	Data Warehouse	SGF2012	FACT_SALES	Sales Fact Table	.	.
2	1	Staging	SGF2012	STG_SALES	Sales Staging Ta...	.	90

Figure 2. Sample Contents of Table Dimension

UNDERSTANDING THE CDC PROCESS LOGIC

Once data is extracted, it is filtered for data that has not already been extracted successfully. This is done by comparing a timestamp in the incoming source data with the "last_key_date_extracted" value in the source dimension for the respective source extracted. Only records that have a timestamp after the "last_key_date_extracted" will flow into the target table. Data quality checks should be performed prior to inserting into the target table. If the data quality checks fail, the data should not be inserted and therefore the control tables will not be updated.

This methodology has a few major benefits:

- ETL process can run at any frequency.
- The same process can be used across the entire data warehouse environment and is centrally managed.
- There is no need to develop a "historical" or "initial" load process separate from incremental loading in a data warehouse project.

UPDATING CONTROL TABLES EFFICIENTLY WITH SAS MACRO® PROGRAMMING

Control tables must be updated immediately after source data is inserted to prevent duplication of data from another subsequent run of the ETL process. For the example used in this paper, this consists of identifying where the data originated, the last point of data extracted according to the timestamp, and the target table and library. The following SAS Macro is defined to update control tables more efficiently.

```
/** Macro to Update Control Tables After Adding Data **/
%macro update_control(source,last_key_date_extracted,target,target_library);
/* source = Data source extracted                                    */
/* last_key_date_extracted = Last identifying date extracted         */
/* target = Target data source                                       */
/* target_library = Target library of data source updated            */
proc sql;
  update sgf2012.dim_source
  set last_key_date_extracted=round("&last_key_date_extracted"dt,'0:00:01'T)
  where source_desc = "&source";

  update sgf2012.dim_source
  set last_extraction=round(datetime(),'0:00:01'T)
  where source_desc = "&source";

  update sgf2012.dim_table
  set last_update=datetime()
  where physical_table_name = "&target" and libref = "&target_library";
quit;
%mend update_control;
```

CONCEPTUAL PROCESS

To summarize, the overall process uses the components described above in the following order:

1. Read source data
2. Use CDC logic to keep new data
    a. Lookup the last key date extracted for source data in control table
    b. Identify new data by comparing current key dates in source data with last key date extracted
3. Insert new data in target table
4. Update Control Table

Each component is critical to the overall process and shown below in Figure 3.

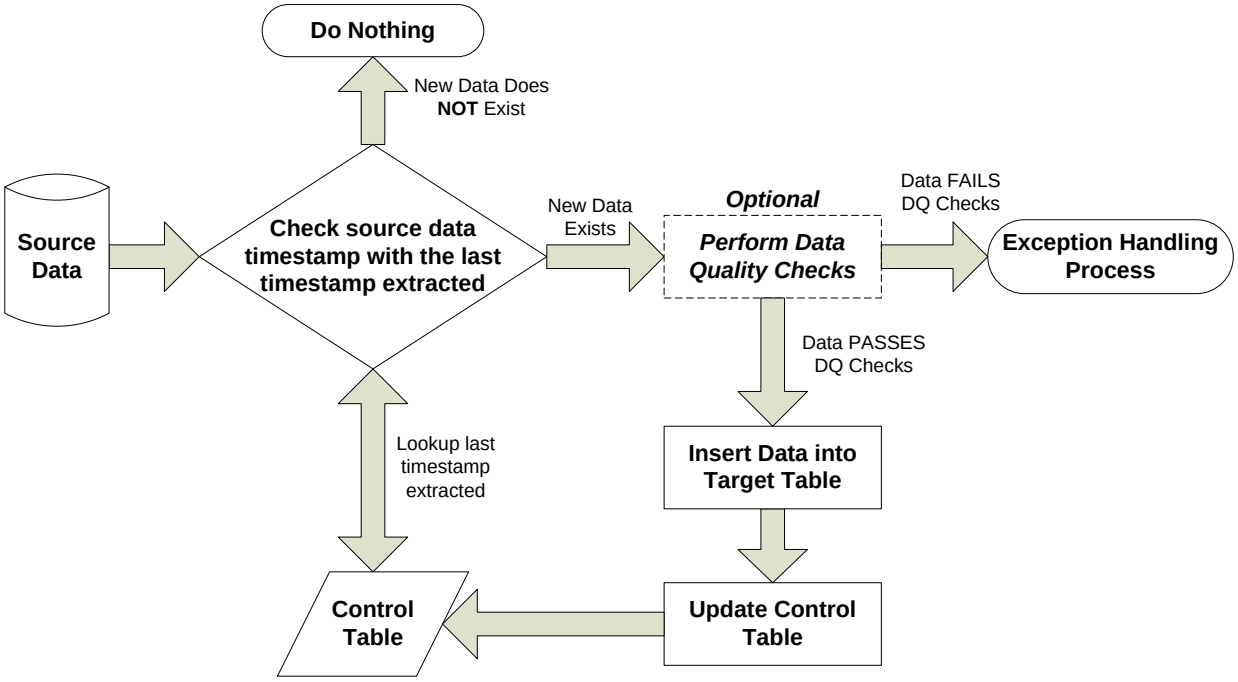


Figure 3. Conceptual Process Flow

EXAMPLE CHANGE DATA CAPTURE PROCESS

To demonstrate the CDC process described in this paper, simulated monthly Sales data is built from the SASHELP.PRICEDATA and inserted into a staging table. The initial load inserts the bulk of sample data. Subsequent runs of the same ETL process demonstrates the CDC process, which inserts new data or nothing at all.

SOURCE DATA EXTRACTION

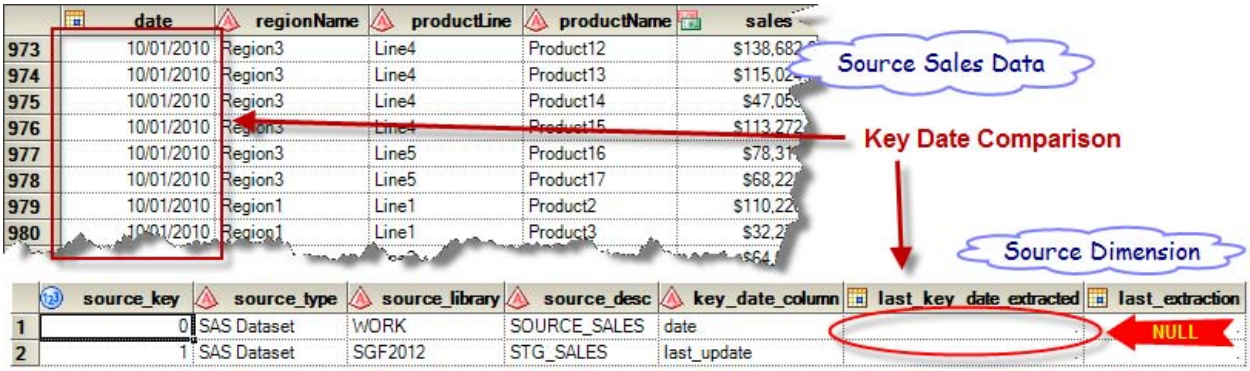
This paper uses the PRICEDATA SAS dataset, which is already in a useable format. The PRICEDATA dataset is included with SAS. The date column, highlighted in Figure 4, identifies the record date. This is an example of an observation specific date.

	date	regionName	productLine	productName	sales
1	01/01/2006	Region1	Line1	Product1	$55,706.5
2	01/01/2006	Region2	Line3	Product10	$65,154.34
3	01/01/2006	Region2	Line3	Product11	$69,561.38
4	01/01/2006	Region3	Line4	Product12	$153,195.86
5	01/01/2006	Region3	Line4	Product13	$124,572.1
6	01/01/2006	Region3	Line4	Product14	$56,308.6
7	01/01/2006	Region3	Line4	Product15	$125,891.8
8	01/01/2006	Region3	Line5	Product16	$72,210.94
9	01/01/2006	Region3	Line5	Product17	$83,644.5
10	01/01/20	Region1	Line1	Prod	123

Figure 4. Sample Contents of Source Sales Data

LOADING THE STAGING TABLE

When initially loading the staging table, the "last_key_date_extracted" value obtained from the source dimension for the data source is null. This allows all historical data to flow into the staging table because the key dates are greater than a null value. This is shown in Figure 5 below.

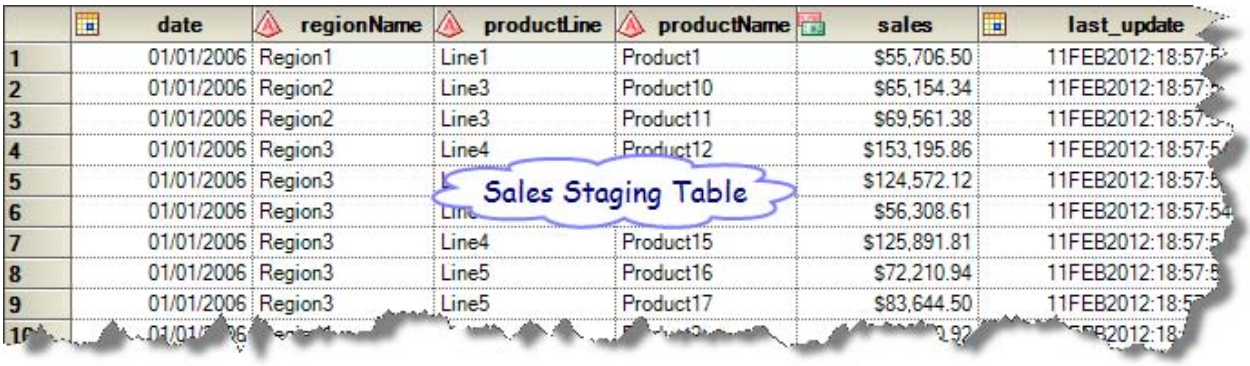


	date	regionName	productLine	productName	sales
973	10/01/2010	Region3	Line4	Product12	$138,682
974	10/01/2010	Region3	Line4	Product13	$115,02
975	10/01/2010	Region3	Line4	Product14	$47,05
976	10/01/2010	Region3	Line4	Product15	$113,272
977	10/01/2010	Region3	Line5	Product16	$78,31
978	10/01/2010	Region3	Line5	Product17	$68,22
979	10/01/2010	Region1	Line1	Product2	$110,22
980	10/01/2010	Region1	Line1	Product3	$32,2

	source_key	source_type	source_library	source_desc	key_date_column	last_key_date_extracted	last_extraction
1	0	SAS Dataset	WORK	SOURCE_SALES	date		
2	1	SAS Dataset	SGF2012	STG_SALES	last_update		

Figure 5. Key Date Comparison

The current timestamp (11FEB2012 for this example) is stored as the “last_update” attribute in the target staging table to normalize time used in CDC processing within the data warehouse. The Sales Staging Table is loaded with the following data as shown in Figure 6. You can see the last_update column was added.



	date	regionName	productLine	productName	sales	last_update
1	01/01/2006	Region1	Line1	Product1	$55,706.50	11FEB2012:18:57:5
2	01/01/2006	Region2	Line3	Product10	$65,154.34	11FEB2012:18:57:
3	01/01/2006	Region2	Line3	Product11	$69,561.38	11FEB2012:18:57:
4	01/01/2006	Region3	Line4	Product12	$153,195.86	11FEB2012:18:57:5
5	01/01/2006	Region3			$124,572.12	11FEB2012:18:57:5
6	01/01/2006	Region3			$56,308.61	11FEB2012:18:57:54
7	01/01/2006	Region3	Line4	Product15	$125,891.81	11FEB2012:18:57:5
8	01/01/2006	Region3	Line5	Product16	$72,210.94	11FEB2012:18:57:5
9	01/01/2006	Region3	Line5	Product17	$83,644.50	11FEB2012:18:5

Figure 6. Sales Staging Table

The code used to load the Sales Staging Table is shown below.

```
proc sql;
  /** Get last date extracted from source to join to incoming data **/
  create table last_key_date_extracted as
    select last_key_date_extracted
    from sgf2012.dim_source
    where source_desc = 'SOURCE_SALES';

  /** Create temporary table with incoming data to append in next step **/
  create table stg_insert_sales as
    select
      source.*,
      datetime() as last_update format datetime22.
    from
      source_sales source,
      last_key_date_extracted control
    where
      dhms(source.date,0,0,0) > control.last_key_date_extracted;

  /** Get count of incoming records for macro logic later **/
  select count(*) into :stg_sales_cnt from stg_insert_sales;

  /** Get last date in source data to use for control table **/
  select
    dhms(max(date),0,0,0) format=datetime. into :last_key_date_extracted
  from stg_sales_temp;
quit;
```

UPDATING THE CONTROL TABLE

Once the data is inserted into the staging table, the Source Dimension control table can be updated with the latest timestamp, which represents the last piece of data inserted into the target table. The entire process can be run any time after this timestamp and inserts only new data. A macro is used in this demonstration as a checkpoint to make sure there is incoming data to insert into the target table. Continuing the example above, the SAS code used to add data to the target table and update all respective control tables is shown below.

```
%macro load_data;
  /** If incoming data exists, add to target table and update control tables **/
  %if &stg_sales_cnt > 0 %then %do;
    /** Add data to staging table **/
    proc append base=sgf2012.stg_sales data=stg_insert_sales; run;

    /** Update Control Tables After Adding Data **/
    %update_control(SOURCE_SALES,&last_key_date_extracted,SGF2012,STG_SALES);
  %end;
%mend load_data;

%load_data;
```

After the process completes, the Source Dimension is updated and contains the following data:

	source_key	source_type	source_library	source_desc	key_date_column	last_key_date_extracted	last_extraction
1	0	SAS Dataset	WORK	SOURCE_SALES	date	01NOV2010:00:00:00	12FEB2012:11:33:54
2	1	SAS Dataset	SGF2012	STG_SALES	last_update	.	.

Figure 7. Updated Source Dimension After Change Data Capture Process

If the same SAS code is re-submitted to load the Sales Staging Table, no data will be loaded because all key dates in the source table are equal to or before the “last_key_date_extracted” value.

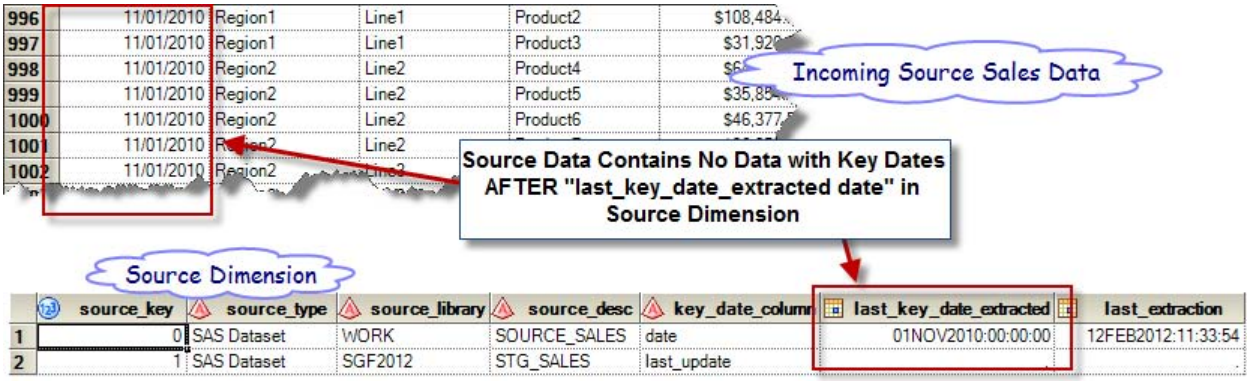
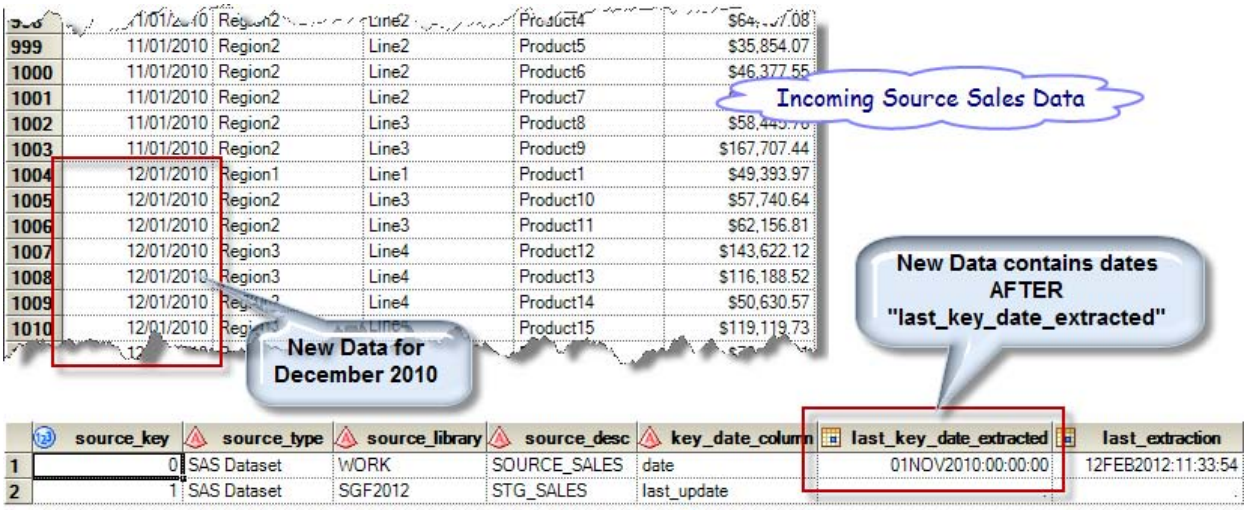


Figure 8. Key Date Comparison with No Change

CAPTURING CHANGED DATA

Additional data is added to the incoming Source Sales data to demonstrate the CDC process. As long as the key date column in the source data is greater than the “last_key_date_extracted” value from the source dimension, the data will be extracted and added to the staging table. This is shown below in Figure 9. The ETL process can be run at any time to capture this data. This is beneficial for source data which is lagged or is not updated on a regular basis.



	source_key	source_type	source_library	source_desc	key_date_column	last_key_date_extracted	last_extraction
1	0	SAS Dataset	WORK	SOURCE_SALES	date	01NOV2010:00:00:00	12FEB2012:11:33:54
2	1	SAS Dataset	SGF2012	STG_SALES	last_update		

Figure 9. New Data Extracted from Incoming Source Data

New data is inserted into the staging table and the "last_key_date_extracted" column in Source Dimension is updated to December 1, 2010.

	source_key	source_type	source_library	source_desc	key_date_column	last_key_date_extracted	last_extraction
1	0	SAS Dataset	WORK	SOURCE_SALES	date	01DEC2010:00:00:00	12FEB2012:11:48:13
2	1	SAS Dataset	SGF2012	STG_SALES	last_update		

Figure 10. Updated Source Dimension After New Data is Extracted

DATA RETENTION CLEANUP

The Table Dimension control table can be used to maintain data retention policies in the data warehouse. Staging tables should be purged regularly. Data that exceeds a time threshold should be removed to minimize the overall physical size of the data warehouse. This paper will demonstrate a simple SQL process which uses an attribute of the Table Dimension to remove obsolete data. SAS Macro Programming is used to loop through a pre-defined list of tables.

```
proc sql;
  /** Get a list of all tables in a specified library using the SAS dictionary
  tables **/
  create table table_list as
    select
      tables.libname,
      tables.memname as tablename,
      control.data_retention_days,
      control.libref
    from dictionary.tables tables
    left join sgf2012.dim_table control on
      trim(tables.memname) = trim(control.physical_table_name) and
      trim(tables.LIBNAME) = trim(control.libref)
    where
      control.dw_location = 'Staging'
  ;

  /** Get list of tables from desired schema or library **/
  select tablename into :table1-:table99999 from table_list;

  /** Get data retention thresholds from same list of tables **/
  select data_retention_days into :retention1-:retention99999 from table_list;

  /** Get how many tables are in the list to loop through **/
  select count(*) into :table_count from table_list;
quit;
```

```
/**Macro to loop through a single information map at a time and return all data fields
and key data about the field**/
%macro PurgeTables;
  %do x=1 %to &table_count;
    proc sql;
      delete from sgf2012.&&table&x
      where last_update < intnx('dtday',datetime(),-&&retention&x);
    quit;
  %end;
%mend;

%PurgeTables;
```

CONCLUSION

Requirements of the project should always drive the design specifications. The control tables discussed in this paper were created to show the conceptual process. More attributes can be added or used when processing data for more control. However complex the control tables become, using this CDC process provides a flexible method to extract data and maintain the data warehouse.

REFERENCES

SAS code, which builds sample data and demonstrates the change data capture process and data retention process described in this paper can be found at the following URL.

http://www.stephenoverton.net/SASCode/SGF2012/

RECOMMENDED READING

Kimball, Ralph. April 2002. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* 464 Pages. John Wiley & Sons, Inc.

CONTACT INFORMATION

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

Stephen Overton
Zencos Consulting (http://www.zencos.com)

Work Email: soverton@zencos.com
Personal Email: stephen.overton@gmail.com
Personal Website: http://www.stephenoverton.net
LinkedIn: http://www.linkedin.com/in/overton