Abstract

This paper describes an application for the processing of Nielsen Scantrack data gathered on items sold in stores using laser check out scanners. This information can be very useful for marketing research.

As delivered, the data is in the form of a hierarchical file on tape. In order to manipulate the data for analysis, relate it to other corporate data, and handle database functions, such as backup and update, an application was developed to transform the information to a relational (table-like) format using SAS software.

The overall process and techniques involved are discussed along with hierarchical-to-relational transformations in general.

What Is Nielsen Scantrack Data?

Scantrack data is gathered from retail check out systems employing laser UPC (Universal Product Code) bar code scanners. These scanners read a bar code that uniquely identifies each item. The code is used as a look up key for the item’s price, which is automatically totaled on a cash register.

Above and beyond the value of quick price look up, the fact that an item has been sold is itself a valuable piece of information. For the retailer this information can be used for inventory management and marketing feedback. For a distributor or manufacturer the effects of price, advertising, and competitor activity can be tracked at the "micro" level. Of particular value is the fact that the information is in machine readable form from its inception. Computers can be used to gather, transform, and process the data.

Data gathered via scanners is processed by Nielsen and others for resale to interested clients. Vendors of scanning data, generally offer it in various value-added formats including aggregation across brand names, markets, and store chains; typically including information relating to advertising activity.

This paper deals with Nielsen "store level" data which includes item sales and advertising information by store within market areas. The data is provided on tapes at 4 week intervals with all information aggregated by week.

Data Format

Scantrack data is delivered as a hierarchical file. In this case hierarchical means that the file consists of several record types. Data fields on differing record types do not necessarily have analogous meanings and many record types have unique field layouts. As a simplification we can say that record types fall into 2 groups:

- detail information records
  - records giving sales volume or advertising information for an item, for example: the dollar sales for an item for a certain week
- descriptive records
  - records that describe an attribute of the detail data, for example: a text description of the item sold.

While Scantrack is "value-added" relative to the raw UPC scanner information, it will normally require processing by the receiving client before use. At a very minimum the tapes would be backed up, cataloged, or unloaded to disk. Additional processing may be necessary to meet particular client requirements. What follows is a discussion of an application built to satisfy a specific Scantrack client’s need for molding the data to a more useable, and maintainable form.
The Problem

A marketing research organization that uses Scantrack is presented with several problems.

First, each 4 week tape must be integrated with existing information from previous periods. Scantrack may contain updates to data from earlier periods so this integration process must include an update function as well as an add function.

A more critical problem involves structuring the information for access by the intended end-users. User requirements include systematic and ad hoc analysis, and the joining of Scantrack data to corporate data from other sources including a relational database. To these ends, the Scantrack data itself is transformed to a relational format.

The central problem with this hierarchical to relational transformation lies in the fact that the information we want for a particular table may come from several different Scantrack record types.

For example, the text description of an individual item, identified by it's UPC code, is held in one record type while the sales data for that item are in another record type. We want a single table with columns for item codes (UPC's), text description, and sales.

Two Approaches

When considering the design of an application to convert this series of hierarchical records to a series of relational tables, there are 2 possible approaches.

One is to rely on hard coded relationships between record types and tables. This offers the advantage of simplicity and a smaller investment in design time. Such a design would be acceptable if the configuration of the data were stable..this is not the case with Scantrack.

Each tape has records that describe which data are on the tape. This information must be utilized to adjust to variations in the data actually provided. This would be rather tedious and error prone as a manual task.

The other approach is dynamic code generation, making use of the Scantrack descriptive record types to build data relations on the fly, at database load time, without manual intervention. Here the trade off is between higher initial investment in system design and development vs. less ongoing maintenance and a reduced burden on the end user.

Since the application discussed below was developed for end users whose time was much more valuable when applied to data analysis than to system maintenance the second approach was favored.

While the actual application deals with several aspects of the use of Scantrack we will concentrate on hierarchical to relational data transformations, the specific techniques involved, and their application to general data transformation problems.

Applicability of SAS

There were several reasons for using SAS software in this application. Most important was the fact that the end users were quite knowledgeable in SAS and would be using it for data analysis. SAS datasets also provide an intuitive, table-like data layout, and the SAS Data Step language is a powerful tool for a fairly complex application developed under a deadline.

Process

The resulting application process can be broken down into 3 steps:

1. reformat the descriptive and detail record types as "tables", i.e., normalize the records
2. join descriptive and detail records by a key, such as the UPC code
3. build transaction files and apply them to master files.

We will be concentrating on steps 1. and 2., joining information from descriptive and detail records.
Techniques

Since Scantrack record types appear in a particular sequence, one way of viewing the data is as a sequential stack of relational tables. The layout of the columns in each "table" may be totally different but there is likely to be at least one common column, or field. This stack of tables can be visualized as a tower of odd sized blocks.

A problem arises when we want descriptive data from a table near the top of the stack and detail data from a table near the bottom of the stack...we need to "unstack" the tables so they can be used in parallel and joined by a common field.

The problem and two solutions are illustrated by the following examples. First, using SAS macro variables, and second user Formats.

Data Value Attribute (SAS Macro Variables)

There are many Scantrack detail data elements; dollar sales, customer counts, and average selling price are three. Each of these elements is held in a "flattened" 2 dimensional array that can be thought of as a table. One column of this table is a data element id code, another is the data value. The information in these tables applies to a particular UPC code/week/store combination.

Another, descriptive, record type holds information about the data element value, i.e., an attribute of the data value, for example, its type: character or numeric. Grouped together, these records constitute a logical attribute description table.

As data elements are read we need to know if their values are to be read as character or numeric data. On a particular tape, the data attribute description record types could be read and a report generated listing the data codes and their respective types: character or numeric. The program that reads the detail records could then be edited to reflect these types with the proper SAS INFORMAT's and edits.

As mentioned above, this "hard code" approach is prone to errors and would require manual intervention for each tape. This approach was abandoned in favor of dynamic generation of SAS INPUT and INFORMAT statements via SAS macros.

The problem can be viewed as one of joining the "table" of descriptions and data codes to the "table" of data codes and data values by their common attribute: data code. In order to do this the code/description table must be available when the code/data table is processed. This can be handled with SAS macro variables.

As the Scantrack tape is read and the code/description records encountered, the information in these records is held in an array of macro variables that is analogous to a table of the attribute codes and attribute descriptions. These macro variables are used to generate SAS SELECT/WHEN statements that execute appropriate formatted INPUT statements, and, in effect, act as a runtime look-up table of attribute data type (numeric or character) given attribute code.

Relating Market To Store (User FORMAT's)

One level of Scantrack detail information is by store, or more properly, store code. Since much of marketing analysis is done on a market territory level it is useful to include the market territory name as a column of the store information table.

In Scantrack there is a record type that describes each store including the market it belongs to. These records are read to create a SAS dataset containing two variables: store code and market.

As the Scantrack tape is read we need to know if their values are to be read as character or numeric data. On a particular tape, the data attribute description record types could be read and a report generated listing the data codes and their respective types: character or numeric. The program that reads the detail records could then be edited to reflect these types with the proper SAS INFORMAT's and edits.

While this store/market dataset could be searched to find the market a given store is associated with, this would be very inefficient since, on average, half the dataset would have to be read for every detail record. In order to speed things up the store/market information is converted to a temporary SAS user FORMAT in which the RANGE is the store id and the LABEL is the market. This is accomplished by using a Data Step to write a system file that constitutes a SAS PROC FORMAT Step with the appropriate RANGE'S and LABEL'S. The PROC FORMAT Step is then INCLUDE'd into the overall SAS job to make the format available as the detail records are read. The user
FORMAT becomes a fast look-up table for joining store data with store description. Since this FORMAT is built dynamically, at runtime, it requires no user intervention and is always current to the Scantrack tape being processed.

CONCLUSIONS

Nielsen Scantrack files are moderately complex hierarchical data structures. Processing of Scantrack data often requires the joining of information from various, out of sequence record types. This is a problem to the developer of any Scantrack application. The task is easier if the Scantrack data can be viewed in the context of a relational (table-like) data model.

If visualized as a series of dissimilar tables stacked end to end the information from various record types can be joined as needed and the Scantrack application design can employ relational methodology. SAS macros and user FORMAT's provide powerful tools to facilitate the automated, runtime, "unstacking" of Scantrack data into a relational structure. Applications designed along these lines can dynamically adjust to changes in the Scantrack tapes.

If you have comments or would like more information please contact:

Tom Miron
Systems Seminar Consultants
6014 Gateway Green
Madison, WI 53716
(608)255-3531, (608)222-7081

SAS software is a registered trademark of SAS Institute Inc., Cary, NC, USA.