*Entity-Relationship Approach to*
*Information Modeling and Analysis, P.P. Chen (ed.)*
*ER Institute, 1981*
*© ER Institute*

# DECLARATIVE AND PROCEDURAL FEATURES OF A CODASYL ACCOUNTING SYSTEM

William E. McCarthy and Graham Gal
Department of Accounting
Graduate School of Business Administration
Michigan State University
East Lansing, Michigan 48824
U.S.A.

This paper presents selected aspects of an information system whose data files have been specified in accordance with "events" accounting principles and maintained with a CODASYL database management system (DBMS). The accounting constructs employed in this implementation are based on entity-relationship modeling, and the particular DBMS used is the GPLAN system developed at Purdue University. The paper reviews derivation and use of both declarative and procedural features of the events model.

## 1. INTRODUCTION

During the last two decades, accounting researchers have been repeatedly faced with the challenge of adapting conventional accounting models to fit well with the capabilities provided by modern computer systems [1] [2]. A prominent stream of research concerned with these adaptations has come to be known as "events accounting," a term that is now used to describe any accounting system whose primary distinguishing feature is the storage and maintenance of transaction data in disaggregate form. Sorter [24] first proposed the events philosophy in 1969 as an alternative approach to public reporting of financial data, but later work in this area [9] [16] [14] [12] [18] has concentrated almost exclusively on the development of multidimensional accounting systems and the integration of such systems with the shared use and modeling concepts of database management. Readers interested in a review of the events accounting literature may consult [22].

In this present paper, we describe an actual events accounting implementation that was built using the network or CODASYL [8] approach to database management. The particular system implemented in this case is an events model based on an entity-relationship (E-R) view of accounting [18]. Such a view discards traditional conventions such as debits, credits, and accounts in favor of data models describing economic resources, events, and agents. Data

---

This project was funded by a grant from the Peat, Marwick, Mitchell Foundation through its Research Opportunities in Auditing program. The views expressed herein are those of the authors and do not necessarily reflect the views of the Peat, Marwick, Mitchell Foundation. We also would like to acknowledge the assistance provided by Dennis Miller, Brenda Spiewak, and Debbie North.

definitions are specified by mapping E-R constructs (such as entity-relationship tables [7]) into CODASYL constructs (such as record types and owner-coupled sets). Following exhibition of these definitions, procedural aspects of the implemented system are illustrated using process description tools recommended by DeMarco [11]. These procedural aspects include transaction processing and information retrieval for both accounting and non-accounting decision makers.

The events implementation described herein has been programmed using the GPLAN database management system (DBMS), a CODASYL system developed at Purdue University in FORTRAN [15]. In the examples presented in the paper however, we have chosen to use more English-like descriptions of both data elements and processes in an effort to make our illustrations more readable. Readers interested in seeing the actual data definitions and program code may consult [23].

## 2. DECLARATIVE FEATURES OF THE EVENTS SYSTEM

### 2.1 Introduction

The declarative features of a commercial database system can be developed by (1) analyzing the entity-relationship structure of the corporate enterprise's data environment, (2) translating that E-R structure into the logical model employed by a particular DBMS, and (3) mapping that logical model into physical storage definitions. An overview of this process is illustrated in Figure 1 and explained below.

### 2.2 Object System → E-R Data Model

The object system [25] is a term used to describe those aspects of a certain reality that are of interest to potential database users. For accountants, the object system consists primarily of phenomena that directly concern a given corporate enterprise's financial status, phenomena such as economic resources ("cash" and "inventory"), economic events ("sales" and "services"), and economic agents ("customers" and "vendors"). The object system also will include both relationships among the economic phenomena (vendors "participating in" purchases or cash receipts "paying for" sales) and detailed characteristics of the phenomena (the "name," "address," and "current balance" of customers).

The steps involved in mapping elements of an accounting object system to an E-R data model are explained in detail by McCarthy [18]. This abstraction process is represented by the two top blocks of Figure 1, and its final output is a data model consisting of entity and relationship tables (also referred to as entity and relationship relations [7]) plus an E-R diagram showing how the various tables correspond to each other. In the model shown in Figure 1, there are three entity sets (rectangles) and two relationship sets (diamonds).

## 2.3 E-R Data Model → CODASYL DBMS Processable Schema

The translation of an E-R data model to a CODASYL schema is illustrated by the conversion of an E-R diagram to a data structure diagram [4] in Figure 1. Chen [7, pp. 29-32] has established rules for this conversion which can be summarized as follows:

(1) All entity sets (such as A, B, and C) become CODASYL record types.

(2) All m-to-n relationship sets (such as Y) also become record types; this kind of record type is usually called a link record.

(3) All 1-to-n relationships (such as X) are represented by declarations of owner-coupled sets with the "1" entity as owner and the "n" entity as member. This is shown in the data structure diagram by the arrow pointing from the owner (A) to the member (B) with the set name (Set 1) written beside it.

(4) All m-to-n relationships are represented by declaring two sets (Set 2 and Set 3) with the related entities (B & C) as owners and the link record (Y) as members.

Illustrated in Figure 2 is the same type of conversion just discussed. The top part of the figure contains a subset of an E-R accounting model taken from [20]. The bottom part of the figure was derived using the four rules summarized above plus declarations of three "system-owned" sets (S1, S8, and S10). These system-owned sets are entry points to the database and allow access to customers, inventory, and vendors in a manner analogous to sequential reading of tape files.

Fields of selected record types for the CODASYL structure are illustrated in Figure 3. These fields represent selected characteristics of the modeled entities and relationships, and similar fields would have to be specified for all record types used in a particular implementation.

## 2.4 CODASYL DBMS Processable Schema → Storage Structure Definition

The mapping from record types and owner-coupled sets to storage structures is an implementation feature peculiar to the specific database management system being used. The physical storage features of the GPLAN DBMS (on which the events accounting system described herein is implemented) are explained by Haseman and Whinston [15, Chap. 11].

## 2.5 Declarative Features Summary

The derivation of the declarative features of our events system, as illustrated in Figure 1, has now been explained completely. In the rest of the paper, we will concentrate on explaining procedural aspects of the CODASYL structure outlined in Figures 2 and 3. These procedural aspects are concerned primarily with the maintenance and use of data defined in the sections above.

## 3. PROCEDURAL FEATURES OF THE EVENTS SYSTEM

### 3.1 Introduction

This section illustrates transaction processing and information retrieval for the events system. For example cases, we answer the questions: "how do we get the data into the system?" and "how do we get it out for a particular use?" Transaction processing logic is displayed first, using sale events as an example.

### 3.2 Transaction Processing

In Figure 2(b), the 1-to-n relationship between customers and sales was effected in the database by declaring a set (S7) with the "customer" record type as owner and the "sale" record type as member. Two occurrences of this set are illustrated in Figure 4: customer White was the party to whom sales 1, 4, & 5 were made and customer Nelson was the party to whom sales 2 & 9 were made. The dotted lines connecting members of S1 (system-owned) simply signify that not all members of that set instance are shown in the diagram.

The processing logic that inserts information concerning a particular sale into the database is illustrated in Figure 5(a) along with a dictionary enumeration of the data elements contained on a sale invoice. Both the process description and the dictionary entry are outlined using methods recommended by De Marco [11]. In Figure 5(b), an example sale invoice is shown; the paragraphs below explain how data contained on that document is fitted into the system.

(1) First of all, a sale record occurrence is created containing field values for sale characteristics (the particular characteristics were shown in Figure 3).

(2) Next, an owning customer is found for the sale (in this case, customer White) by searching through the customer records via set S1 and then the customer and sale occurrences are linked. This linking is illustrated in Figure 4 by an arrow. Because the set ordering in this instance is specified as FIFO (first-in, first-out), "SALE 1" is inserted first in line next to the owning customer. Sales 4 & 5 are events that occur later and consequently will be added further down the chain extending from the owner record.

(3) Finally in the processing, each line item on the sale document is added to the database and linked in turn to both its sale owner (set S6) and its inventory owner (set S5). Illustrations of line item occurrences and linkages for "SALE 1" are shown at the bottom of Figure 6.

In the actual DBMS implementation [23], the algorithm illustrated in Figure 5(a) is coded using a programming language (FORTRAN) and a data manipulation language (GPLAN DML). Additionally, readers should realize at this point that similar processing of all economic events (including the "cash receipts" and "purchases" shown in Figure 2) would be needed in the full implementation.

### 3.3 Information Retrieval

An important feature of the entity-relationship view of accounting, on which the described CODASYL system is based, is its deliberate exclusion of many double-entry conventions. Conventions such as debits, credits, and accounts are not used at all, but in spite of their absence, traditional accounting numbers can be retrieved without difficulty. Two examples of such retrieval are outlined in Figure 7 and explained briefly below. These explanations are followed by a discussion of non-accounting decision use.

Accounts-receivable (A/R) retrieval for the data structure outlined in Figure 2(b) can be computed using the following summary steps.

(1) Those events that generate claims to future receipts of monetary resources are determined. For the modeled enterprise, this simply means identifying sales transactions.

(2) Requited sales are determined. This means identifying cash receipts and then using that information further to determine the sales for which the receipts paid.

(3) Unrequited sales are determined. This is done by subtracting the set of paid-for sales from the set of all sales. Following determination of this difference, the dollar amounts of the remainder are summed to get total receivables.

Execution of the first two steps explained above requires a sequential scan of all customers (to determine their sales and their cash receipts); therefore, the steps have been grouped together in one repetition block of Figure 7(a). The set difference operation is then accomplished by sorting and matching the two outputs to identify the set of transactions to be summarized.

The record-by-record orientation of the accounts-receivable retrieval is a feature characteristic of CODASYL database languages; readers interested in seeing a set-theoretic specification for the same retrieval may consult [19, p. 633]. It is also interesting to note at this point that, despite the absence of debits and credits, the controls inherent in a conventional subsidiary-control account structure are still present. The subsidiary receivables in this case are characteristics of the customers and are updated by transactions while the controlling figure is simply a redundant check of the data calculated at a certain time for verification purposes.

Cost-of-goods-sold (COGS) retrieval involves three record types--"inventory," "purchases," and "P-line-item"--and three owner-coupled sets--S8, S9, and S12. Some occurrences of these components are illustrated at the top of Figure 6. The additional instances of purchase line items shown in the "cloud" will allow us to be more specific about part of the COGS calculation.

Before studying the COGS retrieval procedure in Figure 7(b), the reader should understand the following aspects of the CODASYL system.

(1) The cost of goods sold for this company is calculated on a LIFO (last in, first out) basis. To facilitate the calculation, we have designated the ordering of set S9 also as LIFO. If this ordering were not present, an additional sort[1] would be necessary for the process.

(2) In Figure 3, the field "line-item-remnant" of "P-line-item" represents that quantity of the particular line item that has not yet been charged off to cost of goods sold.

(3) The variable "sold-quantity" represents the total quantity sold for each inventory item during the reporting period in question. Again to facilitate the calculation, we have assumed that correct values for this variable have been previously materialized from the member instances of set S5.

The COGS derivation is another procedure involving sequential processing of a system-owned set, in this case set S8. For this procedure, each inventory type is retrieved in turn, and its cost is calculated by looking at its purchase history. For inventory type "A" with 12,000 units sold this period, the COGS calculation would involve these components (objects used in the calculation are illustrated in Figure 6):

Total sold = 12,000 at a cost of $22,600 consisting of:

| | | |
|---|---|---|
| 6,000 units @ 2.00 = | $12,000 | (Purchase 10) |
| 4,000 units @ 1.90 = | 7,600 | (Purchase 9) |
| 2,000 units @ 1.50 = | 3,000 | (Purchase 8) |

When the procedure is finished, "Purchase 8" would have 3,000 items of "A" (that is, "line-item-remnant" = 3,000) left to be charged off to future COGS.

Non-accounting data retrieval can be accomplished either by the same technique used above with accounting data--record-by-record processing with a programming language--or by another technique that emphasizes non-programmer use--processing with a query language. GPLAN implementation of both methods are explained by Haseman and Whinston [15, Chaps. 7-9].

Some potential data needs for non-accounting decisions that would use the accounting framework of Figure 2(b) include those shown below.

(1) To analyze the effects of discount policy on cash flows, a manager might request, "For certain groups of customers, plot the lag time between sales and cash receipts against the dollar amount of the

---

[1] This sort would require a time stamp on the "P-line-item" record type; this stamp could be effected either by having a "time" field whose virtual source was its owning record or by retrieving the "purchase" record itself. Since GPLAN does not support the former, we have used the latter in [23].

receipts." This query would involve navigation[2] along the paths of S1, S33 and S28.

(2) To assess vendor performance, a manager might request, "For each vendor, construct a histogram of the quality ratings of our purchases with that firm during a certain period." This query would involve a navigation along the paths of S10 and S11.

(3) To obtain a preliminary indication of the desirability of using distributed warehousing, a manager might ask, "List the names and dollar sales of customers who could have been supplied with their purchased products directly from our vendors located in the same city (or state or area)." This query would involve first a navigation through S1, S7, S6, & S5, then a navigation through S10, S11, S12, & S9, and finally a matching of the data obtained.

### 3.4 Summary of the Procedural Features of the Events System

We are now finished reviewing various types of transaction processing and information retrieval for the events database. Before moving on to a summary, we discuss briefly an important procedural aspect of this type of accounting system: its maintenance of transaction histories.

The CODASYL system described in this paper, with a few exceptions, incorporates completely the concept of events accounting; that is, it maintains all transaction or flow data indefinitely in disaggregate form. Such an implementation is compatible with Sorter's [24] original principles which asked that accounting systems be designed so as to maximize the reconstructability of detailed event histories and thus make the outputs of the system available to a wider range of unspecified decision makers. Sorter's thinking in this regard is echoed in the following quote that deals with information systems development in a different functional area--marketing.

> Care must be taken not to accumulate data in ways that appear reasonable at one time, but preclude analysis at a subsequent time. A disaggregated data file is an important feature of any MIS that hopes to respond to needs or problems that are identified only after the MIS is operational. Disaggregation refers to the maintenance of individual data in a detailed time sequence as they are generated, so that new data are not combined with existing data. [13, p. 3]

Despite these opinions however, there will certainly be times when a cost-benefit assessment of both accounting and non-accounting data usage will identify situations where the desirability of using events principles ought to be

---

[2] The analogy of a CODASYL database user to a navigator in n-dimensional data space is due to Charles Bachman, the originator of most network data model concepts. See [5].

analyzed further.[3] Two such situations were used in our data retrieval examples; they were (1) the "current-balance" field of "customer" and (2) the "sold-quantity" variable which we materialized procedurally but which alternatively could have been maintained as a field in the "inventory" record type. In the former case we decided (for control purposes) to temporally aggregate data, while in the latter case we decided (somewhat arbitrarily) not to. The important point about transaction recording to remember during decision requirements analysis is that the data usage needs of one functional area, such as accounting, should not be allowed to dictate the aggregation levels presented to the other functional areas. This is precisely why an accounting system built with data modeling is better suited to a shared use environment than one built with a debit-credit orientation. Certain accounting actions, such as the preparation of adjusting and closing journal entries, should be performed independent of normal transaction maintenance if they result in aggregations that lose or obscure information for other users.

As a final point in our discussion of procedural features, we should note that our emphasis on events principles, coupled with the very simple nature of the example enterprise which we chose to use, led us to build a system that needed no chart of accounts or general ledger processing. In [23], we illustrate financial statements produced without these conventions. However, with relatively minor alterations, the accounting system described herein could produce (as a byproduct of transaction processing) the information needed to maintain a traditional general ledger [10, Chaps. 15-19]. If such a situation were necessary, we recommend that the general ledger elements (the accounts) be either specified as a view (subschema) or maintained as a separate file outside the scope of the centrally-defined data model.

## 4. CONCLUSION

This paper has described part of an events accounting system, one which was implemented using the GPLAN database management system [15]. CODASYL data definitions and data manipulation programs were developed from an entity-relationship view which considers that accounting object systems are modeled best as sets of economic resources, events, and agents plus relationships among those economic phenomena. The accounting theories underlying this view are explained in [21].

This implementation has demonstrated, at least in a partial sense, the feasibility of constructing data-modeled accounting systems, accounting systems designed from the start to encourage the shared use and maintenance of economic data. For at least two reasons however, the generalizability of our results is hard to assess. First of all, the DBMS used was decidedly limited and less complex than most commercial systems. GPLAN is a research/education system, and it does not support features such as CALC (direct) access, virtual data fields, and subschema specifications. Certainly, an accounting system built on

---

[3] Quite apart from technical considerations, there are other behavioral factors that might also necessitate this further analysis. See [6].

a better equipped DBMS might differ in many ways. Second, the enterprise we modeled was very simple [20]. It consisted of a retail company with a limited capital structure, a limited set of products and services, and no departmental structure. Future implementation efforts will have to concentrate on the modeling of more complex object systems (such as a manufacturing firm with multiple departments) and on the use of more detailed and wide-ranging design methodologies (such as those outlined in [3] and [17]).

## 5. REFERENCES

[1] American Accounting Association. "Report of the Committee on Managerial Decision Models." The Accounting Review, Vol. 44 (Supplement 1969), pp. 43-76.

[2] American Accounting Association. "Report of the Committee on Non-Financial Measures of Effectiveness." The Accounting Review, Vol. 46 (Supplement 1971), pp. 164-211.

[3] Armitage, H., and W. E. McCarthy. "An Entity-Relationship Methodology for the Analysis and Design of Integrated Accounting Systems." Department of Accounting, Michigan State University, 1981. (Working Paper).

[4] Bachman, C. W. "Data Structure Diagrams." Data Base, Vol. 1 (Summer 1969), pp. 4-10.

[5] Bachman, C. W. "The Programmer as Navigator." Communications of the ACM, Vol. 16 (November 1973), pp. 653-658.

[6] Benbasat, I., and A. S. Dexter. "Value and Events Approaches to Accounting: An Experimental Evaluation." The Accounting Review, Vol. 54 (October 1979), pp. 735-49.

[7] Chen, P. P. "The Entity-Relationship Model--Toward a Unified View of Data." ACM Transactions on Database Systems, Vol. 1 (March 1976), pp. 9-36.

[8] CODASYL Programming Language Committee, Data Base Task Group Report. New York: Association for Computing Machinery, 1971.

[9] Colantoni, C. S.; R. P. Manes; and A. B. Whinston. "A Unified Approach to the Theory of Accounting and Information Systems." The Accounting Review, Vol. 46 (January 1971), pp. 90-102.

[10] Cushing, B. E. Accounting Information Systems and Business Organizations. Reading, Mass.: Addison-Wesley, 1978.

[11] De Marco, T. Structured Analysis and System Specification. Englewood Cliffs, N. J.: Prentice-Hall, 1979.

[12] Everest, G. C., and R. Weber. "A Relational Approach to Accounting Models." The Accounting Review, Vol. 52 (April 1977), pp. 340-359.

[13] Gibson, L. D.; C. E. Nugent, E. Christopher; and T. E. Vollman. "An Evolutionary Approach to Marketing Information Systems." Journal of Marketing, Vol. 37 (April 1973), pp. 2-6.

[14] Haseman, W. D., and A. B. Whinston. "Design of a Multidimensional Accounting System." The Accounting Review, Vol. 51 (January 1976), pp. 65-79.

[15] Haseman, W. D., and A. B. Whinston. Introduction to Data Management. Homewood, Ill.: Richard D. Irwin, 1977.

[16] Lieberman, A. Z., and A. B. Whinston. "A Structuring of an Events-Accounting Information System." The Accounting Review, Vol. 50 (April 1975), pp. 246-258.

[17] Lum, V. S.; G. M. Schkolnick; D. Jefferson; S. Su; J. Fry; T. Teorey; and B. Yao. "1978 New Orleans Data Base Design Workshop Report." Research Report RJ2554, San Jose, California: IBM Research Laboratories, 1979.

[18] McCarthy W. E. "An Entity-Relationship View of Accounting Models." The Accounting Review, Vol. 54 (October 1979), pp. 667-686.

[19] McCarthy, W. E. "Construction and Use of Integrated Accounting Systems with Entity-Relationship Modeling." In Entity-Relationship Approach to Systems Analysis and Design. edited by P. Chen. Amsterdam: North-Holland, 1980. pp. 625-637.

[20] McCarthy, W. E. "A Case Study Demonstrating the Applicability of Data Modeling to Accounting Object Systems." Proceedings of the 1980 Southeast Regional Meeting of the American Accounting Association, (April 1980), pp. 319-324.

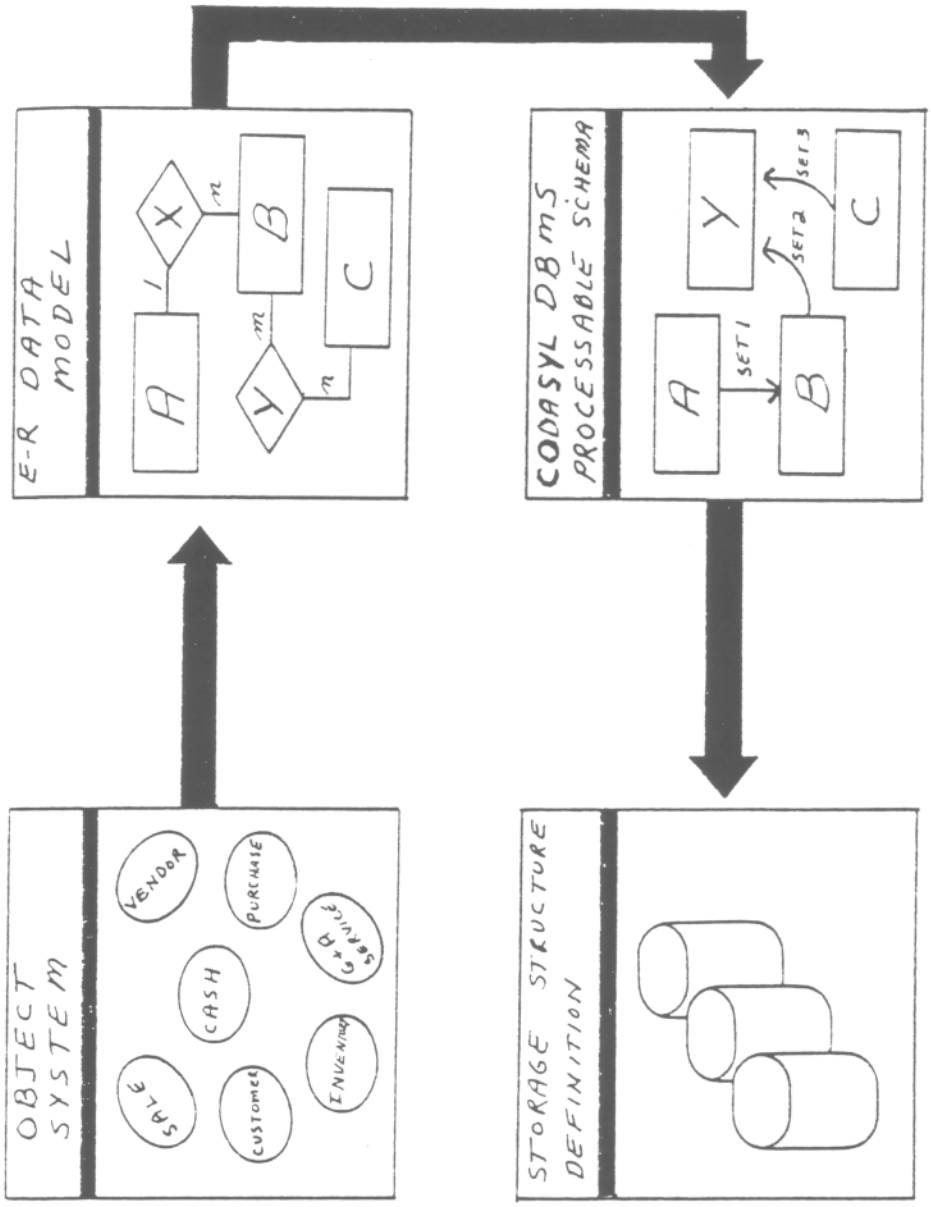
[21] McCarthy, W. E. "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment." Department of Accounting, Michigan State University, 1980. (Working Paper).

[22] McCarthy, W. E. "Multidimensional and Disaggregate Accounting Systems: A Review of the 'Events' Accounting Literature." MAS Communication, Vol. 5 (July 1981), pp. 7-13.

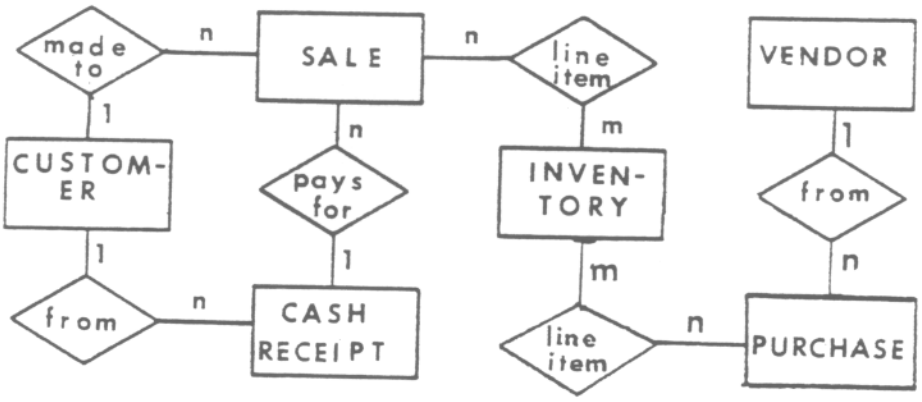
[23] McCarthy, W. E. and G. Gal. "Declarative and Procedural Features of a CODASYL Accounting System." Department of Accounting, Michigan State University, 1981. (Working Paper).

[24] Sorter, G. H. "An 'Events' Approach to Basic Accounting Theory." The Accounting Review, Vol. 44 (January 1969), pp. 12-19.

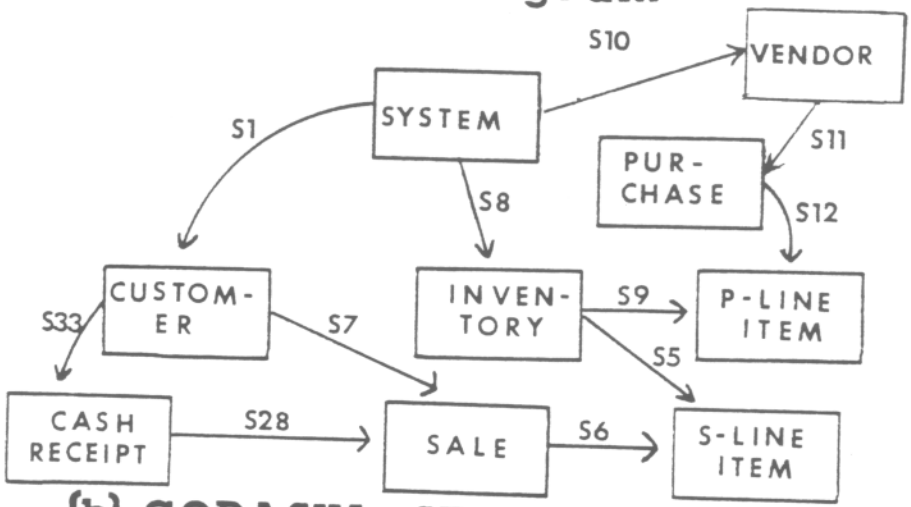
[25] Sundgren, B. "Conceptual Foundation of the Infological Approach to Data Bases." In Data Base Management. edited by J. W. Klimbie and K. L. Koffeman. Amsterdam: North-Holland Publishing Company, 1974. pp. 61-96.




**DEVELOPMENT OF DECLARATIVE FEATURES**
**FIGURE 1**




(a) E-R Diagram




(b) CODASYL STRUCTURE

E-R TO CODASYL TRANSLATION

FIGURE 2

CUSTOMER

| NAME | NUMBER | ADDRESS | BALANCE |
|---|---|---|---|

INVENTORY

| STOCK NUMBER | QOH | DESCRIPTION | COST |
|---|---|---|---|

SALE

| INVOICE NUMBER | DATE | AMOUNT | ORDER NUMBER |
|---|---|---|---|

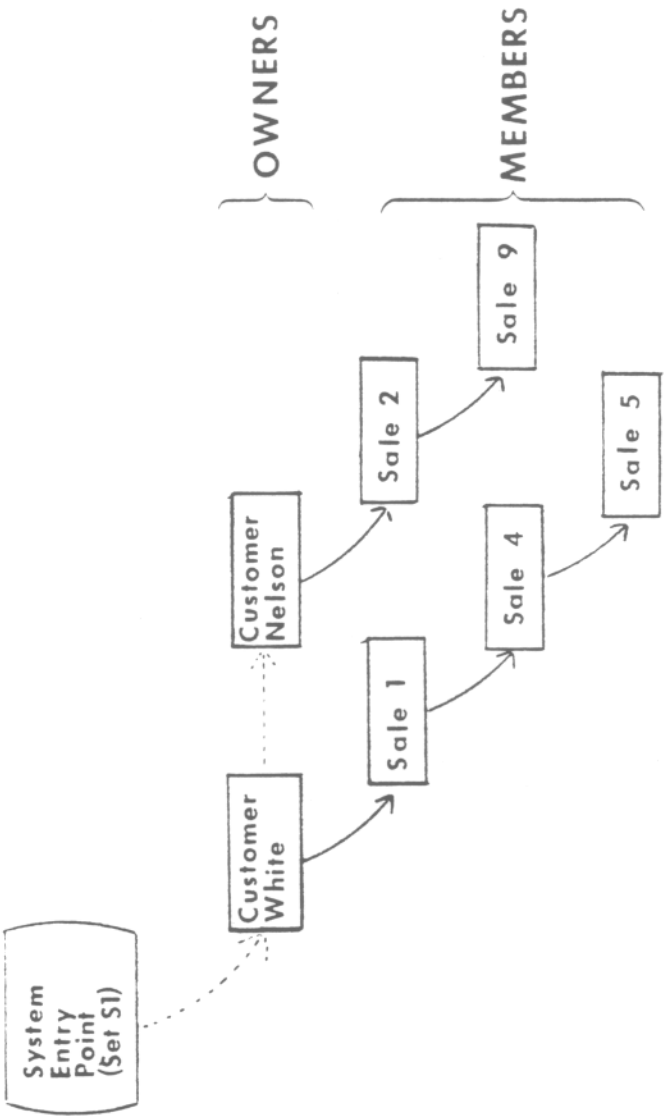
S-LINE-ITEM

| NUMBER | COST |
|---|---|

P-LINE-ITEM

| NUMBER | COST | LINE ITEM REMNANT |
|---|---|---|

**CODASYL RECORD FIELDS**
**FIGURE 3**




**SET S7 OCCURRENCES( FIFO ORDERING)**
**FIGURE 4**

```
Perform the following steps for each new sale invoice
    Create a SALE record type occurrence
    Find the appropriate CUSTOMER member of S1
    Update customer's account
    Add this SALE to S7
    Perform the following steps for each line-item on this sale invoice
        Create an S-LINE-ITEM record type occurrence
        Add this S-LINE-ITEM to S6
        Find the appropriate INVENTORY member of S8
        Update inventory status information
        Add this S-LINE-ITEM to S5
```

```
Sale-Invoice = Sale-Invoice-Number + Customer-Name +
               Customer-Number + Customer-Information +
               1{Inventory-Number + Quantity + Unit Price}7 +
               Sale-Amount
```

## (a) Sale Processing

**WILSON CORP.**

Sales Invoice

invoice no. 1

Customer name White

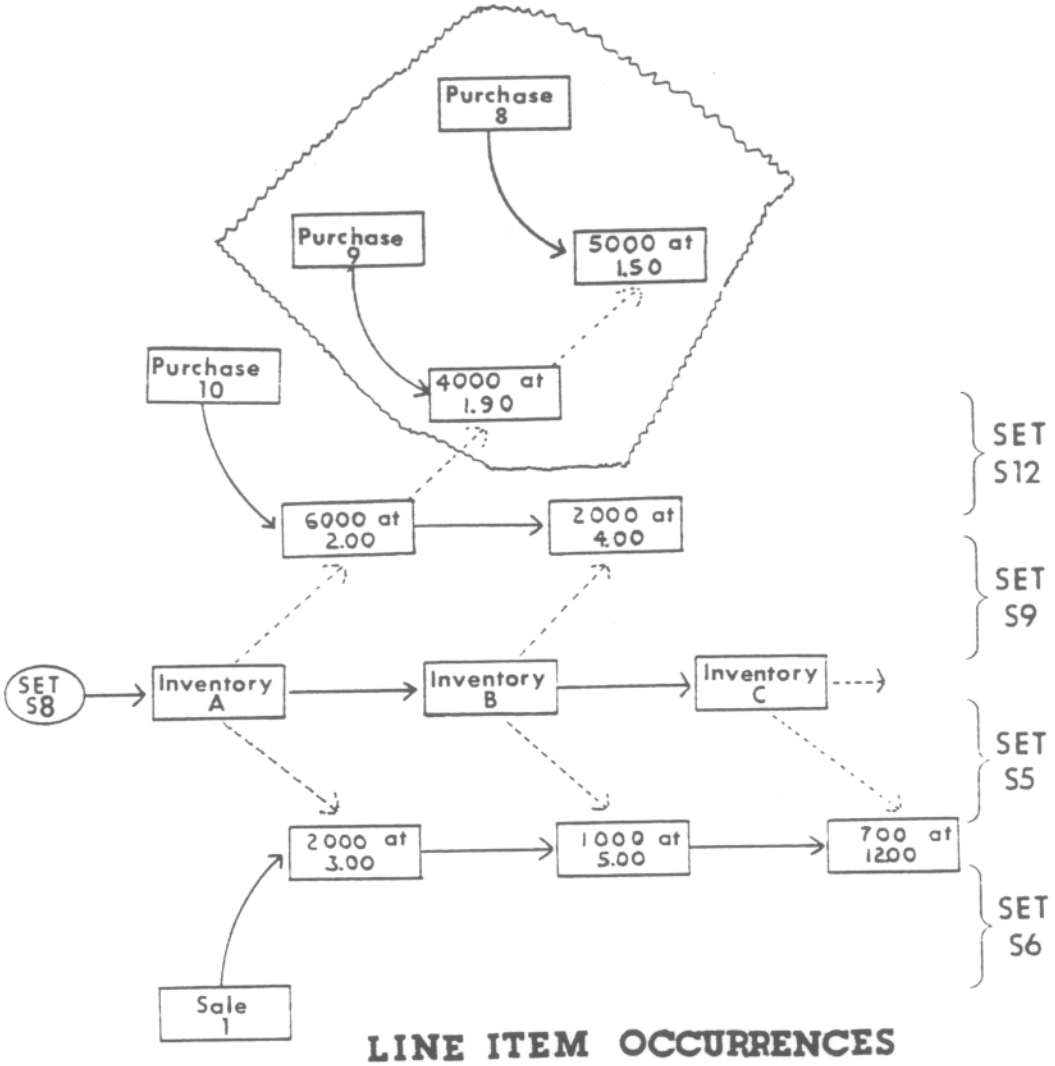
Customer no. 100

Customer info. _____

| line-item | inv# | quantity | price | extension |
|---|---|---|---|---|
| 1 | A | 2,000 | 3.00 | 6,000 |
| 2 | B | 1,000 | 5.00 | 5,000 |
| 3 | C | 700 | 12.00 | 8,400 |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| | | | TOTAL | 19,400 |

## (b) Source Document for Sale

# FIGURE 5




**LINE ITEM OCCURRENCES**

**FIGURE 6**

```
Perform the following steps for each CUSTOMER member of S1
    Get a CUSTOMER occurrence
    Accumulate detail A/R balance
    Perform the following steps for each CASH-RECEIPT member of S33
       with this CUSTOMER owner
        Get a CASH-RECEIPT occurrence
        Perform the following steps for each SALE member of S28 with
           this CASH-RECEIPT owner
            Get a SALE occurrence
            Output sale invoice number and sale amount to paid-list
    Perform the following steps for each SALE member of S7 with this CUSTOMER
       owner
        Get a SALE occurrence
        Output sale invoice number and sale amount to total-list
Sort paid and total lists
Match lists sequentially to identify unpaid sales
Total unpaid sales and compare with accumulated detail balance
```

**(a) A/R Retrieval**

```
Perform the following steps for each INVENTORY member of S3
    Get an INVENTORY occurrence
    For each P-LINE-ITEM member of S9 with this INVENTORY as owner,
      perform the following steps until sold-quantity is gone
        Get a P-LINE-ITEM occurrence
        If line-item-remnant < sold-quantity
              Then, contributed-quantity equals line-item-remnant
              Otherwise, contributed-quantity equals sold-quantity
        Subtract contributed-quantity from line-item-remnant
        Subtract contributed-quantity from sold-quantity
        Multiply contributed quantity by unit-price and add product
          to COGS for this inventory element
    Output Inventory-number and COGS amount
Add COGS for each inventory item and output final figure.
```

**(b) COGS Retrieval**

**FIGURE 7**