AN INVENTORY CONTROL SYSTEM BASED ON ALDAT

FELIX MANUEL NEVRAUMONT
SCHOOL OF COMPUTER SCIENCE
McGill UNIVERSITY

APPLICATION PROJECT II

SUPERVISED : PROF. T.H. MERRETT

WINTER 1984

TABLE OF CONTENTS

I N D E X O F R E L A T I O N S

ACKNOWLEDGEMENT

I would like to give my gratitude to Prof. T.H. Merrett for his guidance and helpful comments and discussions which contributed to the completion of this work.

1.- INTRODUCTION

In this report, we present an inventory control system based on ALDAT using some of the relational algebra operations. In section 2, a brief introduction to relational algebra and ALDAT is given as well as a short introduction to relations and the relational algebra operations. In section 3, a brief introduction to inventory theory is given, as is description of the model used in our system. Section 4 provides a general view of the system diagram with its inputs and outputs. In section 5, we give a design of the inputs to the system. Section 6 describes in detail the system relations (database files) used later in the creation of the database. In section 7, a detailed description of the outputs from the system is given. Finally, in section 8, we describe briefly how to build the inventory control database on the IBM PC using MRDS/FS.

2.- RELATIONAL ALGEBRA AND ALDAT

The Relational algebra is a collection of high-level operations on relations, first introduced by Codd in the early 70's.
It is developed from mathematical set theory. Data is organized in another form of data structure, a relation or a table, which resembles a traditional sequential file. Each column of the table corresponds to a field of the file and each row corresponds to a record. The relational algebra treats the relation as a unit and manipulates all the tuples (rows) at once, with the aid of a set of operators.

The Domain algebra is independent of and complementary to the relational algebra. It operates on domains to define new domains. It is a facility whereby new attributes can be created as functions of existing attributes in a given relation.

ALDAT, an algebraic data language, is a high-level language equipped with a set of powerful relational operators, and a pascal-like programming structure. It provides the facility for easy programming and general relation manipulations; and more importantly, it has the facility for domain algebra operations which operate on attributes of relations to create other new attributes, independently of the relations they are in.
For a detailed description consult [1] and [2].

RELATIONS

A relation is defined to be a set of tuples satisfying the following properties :

1) All rows (tuples) are distinct.
2) The ordering of the rows is immaterial.
3) Each column is labelled and so the ordering of the columns is insignificant.
4) The value in each row under a given column is simple.

The columns are called attributes, which are names associated with a set of values called a domain.

A key of a relation is a minimal subset of its attributes which can be used to identify uniquely each tuple.

Through this report, we will use operations of the relational algebra as well as those of the domain algebra. A detailed description of all the operations may be found in [2] and [4]. We now informally describe a few which are used in this report.

RELATIONAL ALGEBRA OPERATIONS

These functions retrieve data from the database after it has been converted into relational form. We discuss only the PROJECT and the SELECT functions, and the MU-JOIN.

PROJECT

The PROJECT function creates a result relation which is a vertical subset of the operand relation, that is a subset of attributes (columns) from the operand relation are copied into a new relation. Any duplicate rows created in this process are eliminated.

Example 1 : Consider the following two relations and the query : list all items with their corresponding prices.

REL1

WH	ITEM	PRICE	QUANTITY	REORDER LEVEL	EOQ
01	radio	50.00	33	15	30
01	television	600.00	8	9	22
01	stereo	425.00	20	17	14
02	blender	75.00	17	12	27
02	radio	50.00	39	23	35
02	stereo	425.00	6	16	28

REL2

SUPPLIER	ITEM	COST/UNIT
GENERAL ELECTRIC	blender	62.00
GENERAL ELECTRIC	radio	41.00
SONY	television	549.00
SONY	stereo	405.00
SONY	radio	40.00
PHILIPS	blender	64.00
PHILIPS	television	537.00

then

PRICE_LIST <- ITEM, PRICE in REL1

PRICE_LIST

ITEM	PRICE
blender	75.00
radio	50.00
stereo	425.00
television	600.00

SELECT

The SELECT function creates a new relation which is a subset of the operand relation, by including only those tuples (rows) which satisfy a given condition. There are many types of operations on an attribute which can be used as a select condition.

Example 2 : Consider the two relations described before and the query : list all columns for items below their reorder level.

then

ITEMS_BELOW_REORDER_LEVEL <- where QUANTITY <= REORDER LEVEL in REL1

ITEMS_BELOW_REORDER_LEVEL

WH	ITEM	PRICE	QUANTITY	REORDER LEVEL	EOQ
01	television	600.00	8	9	22
02	stereo	425.00	6	16	28

MU-JOINS

The MU-JOIN is a family of functions which perform generalized set operations on pairs of operand relations. Given two relations defined on common domains, then a mu-join can be performed to produce a result relation. This result relation will be composed of different sets of tuples depending on the mode of the mu-join. Through this report, we will use only the intersection or natural join (ijoin) which concatenates tuples where they have the same values for the joining attributes.

Example 3 : Consider the relations ITEMS_BELOW_REORDER_LEVEL and REL2, and the query : list all the suppliers of items which are below their reorder level.

then

ITEM_SUPPLIERS <- ITEMS_BELOW_THEIR REORDER_LEVEL ijoin REL2

ITEM_SUPPLIERS

WH	ITEM	PRICE	QUANTITY	REORDER LEVEL	EOQ	SUPPLIER	COST/UNIT
01	television	600.00	8	9	22	SONY	549.00
01	television	600.00	8	9	22	PHILIPS	537.00
02	stereo	425.00	6	16	28	SONY	405.00

DOMAIN ALGEBRA OPERATIONS

These fall into two categories, horizontal and vertical. Horizontal domain operations have operands which do not cross tuple boundaries, that is all the operands for that operation are found in the same tuple. Some of them are subtraction, multiplication, minimum, etc. On the other hand, vertical domain operations have operands that are a result of a function on values from one or more tuples. Through this report, we will use two classes of vertical domain operations, reduction and equivalence reduction.

REDUCTION (red)

This produces a simple result from the values from all tuples of a single attribute in the relation. Totalling is an example.

EQUIVALENCE REDUCTION (egv)

This is like simple reduction but produces a different result for different sets of tuples in the relation. Each set is characterized by all tuples having the same value for some specified attribute - an "equivalence class" in mathematical terminology. Subtotalling is an example.

Example 4 : We consider the two relations of Example 1 and the query : calculate revenue for each item (quantity * price), the total revenue, and the subtotal revenue per item.

then

```
Let REVENUE be QUANTITY x PRICE
Let TOTAL be red + of REVENUE
Let SUBTOTAL be egv + of REVENUE by ITEM
REVENUES <- WH, ITEM, QUANTITY, PRICE, REVENUE, TOTAL, SUBTOTAL
            in REL1
```

REVENUES

WH	ITEM	QUANTITY	PRICE	REVENUE	TOTAL	SUBTOTAL
01	radio	33	50.00	1650.00	20715.00	3590.00
01	stereo	20	425.00	8500.00	20715.00	11050.00
01	television	8	600.00	4800.00	20715.00	4800.00
02	blender	17	75.00	1275.00	20715.00	1275.00
02	radio	39	50.00	1940.00	20715.00	3590.00
02	stereo	6	425.00	2550.00	20715.00	11050.00

Note how we use projection to "actualize" the results of the domain algebra computations. Otherwise, they are "virtual" attributes, with no physical storage in any relation.

3.- A BRIEF INTRODUCTION TO INVENTORY THEORY AND A DESCRIPTION OF THE MODEL USED

Inventory control is a key element in manufacturing since it is responsible for determining material requirements, providing direction and support to production planning, stores, warehouses, etc., as well as minimizing inventory costs (holding, ordering, storage, etc.), and avoiding loss or theft of merchandise. The main objectives are the following :

- maintain adequate control of the physical movement of materials between various store locations
- maintain proper control of all materials on hand
- provide input data and output data into the invertory record file on all receipts
- control surplus and shortage
- provide the user with current stock status of available inventory, updated purchase orders, updated suppliers' price lists, customers' accounting, etc.
- indicate when to order an item for replenishment of stock and in what quantity an item should be ordered.

BASIC TYPES OF INVENTORY

* Raw materials - materials that will be transformed, cut, formed, or stamped to produce component parts.

* Semifinished component parts in stock (work-in-process) - materials in various stages of production still in a partially completed state.
* Finished goods - end items in stock awaiting shipment to customers.

Our model used in this report will concentrate (focus) on finished goods inventory because it satisfies the requirements for stores (pharmacy stores, supermarkets, car-dealer agencies, etc.) as well as warehousing control (electronic spare parts, furniture, etc.); therefore, this model will be used in the implementation of our system.

Once we have selected the basic type of inventory, then we proceed to select the inventory system category for our model. These categories are summarized in Table 1.

		component demand	
		forecast	formula
maintenance of status data	quantity only	STATISTICAL ORDER POINT	LOT REQUIREMENTS PLANNING
	quantity and timing	TIME-PHASED ORDER POINT	MATERIAL REQUIREMENTS PLANNING

Table 1. INVENTORY SYSTEM CATEGORIES

From the above Table, the best alternative is the TIME-PHASED ORDER POINT because of the following reasons :

- it is subject to independent demand (demand for a given inventory item is unrelated to demand for other units)
- it is suitable for service parts, finished products in factory stock, and field warehouse items
- the requirements for independent-demand items are forecast using any forecasting method the user selects because they can not be calculated.

Finally, we have to select the type of inventory stock that our model will use. These types are summarized in Table 2.

economical order size or fixed order quantity
buffer or safety stocks using reorder point
anticipation of seasonal stocks
ABC inventory classification

Table 2. TYPES OF INVENTORY STOCK

From Table 2, the first two types provide the best choices which our model requires because of the following reasons :

- the number of units to be ordered each time (lot size) and the stock level which triggers a replenishment order (reorder point) are fixed by the decision-maker.
- the purpose of the reorder point is to provide for protection

against variations in item demand as a measure against stock-out. Failure to have stock on hand to fill customer orders routinely from stock could result in lost sales revenue and opportunity for profit.

MODEL DESCRIPTION

The description of the model used can be summarized in Fig. 1. In addition, the model assumptions are the following :

- our model is based upon forecasts for such factors as user demand and lead time. Forecasting is a method whereby one uses a projection of the past demand pattern as a base indicator of the future (more details may be found in [7])
- replenishment orders arrive as single, complete shipments

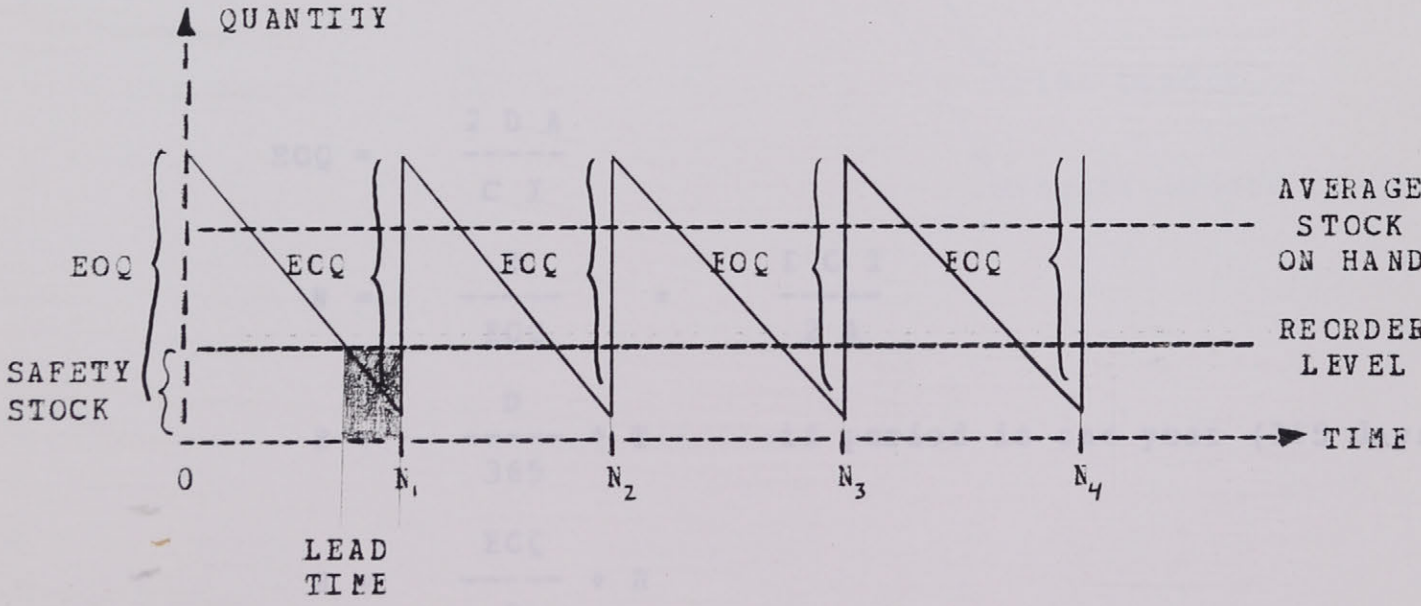


Fig. 1 TIME-PHASED ORDER POINT WITH SAFETY STOCK

MODEL ABBREVIATIONS AND FORMULAS

The derivation of the formulas for our inventory model and a complete description is found in [1].

D = demand (in units) per period (quarter, year, etc.)

A = cost per order new inventory

C = cost per unit

I = inventory holding cost per period

EOQ = economic order quantity (in units)

N = optimal number of orders to be placed per period

T = acquisition lead time (in days) = the time period between the requisition of and the receipt of a replenishment shipment at a place and in a form which makes the new units available to meet user demands

R = reorder stock level (in units)

H = average stock on hand (in units)

$$EOQ = \frac{2\,D\,A}{C\,I}$$

$$N = \frac{D}{EOQ} = \frac{D\,C\,I}{2\,A}$$

$$R = \frac{D}{365} * T$$ if period is one year (365 days

$$H = \frac{EOQ}{2} + R$$

4.- SYSTEM DESCRIPTION

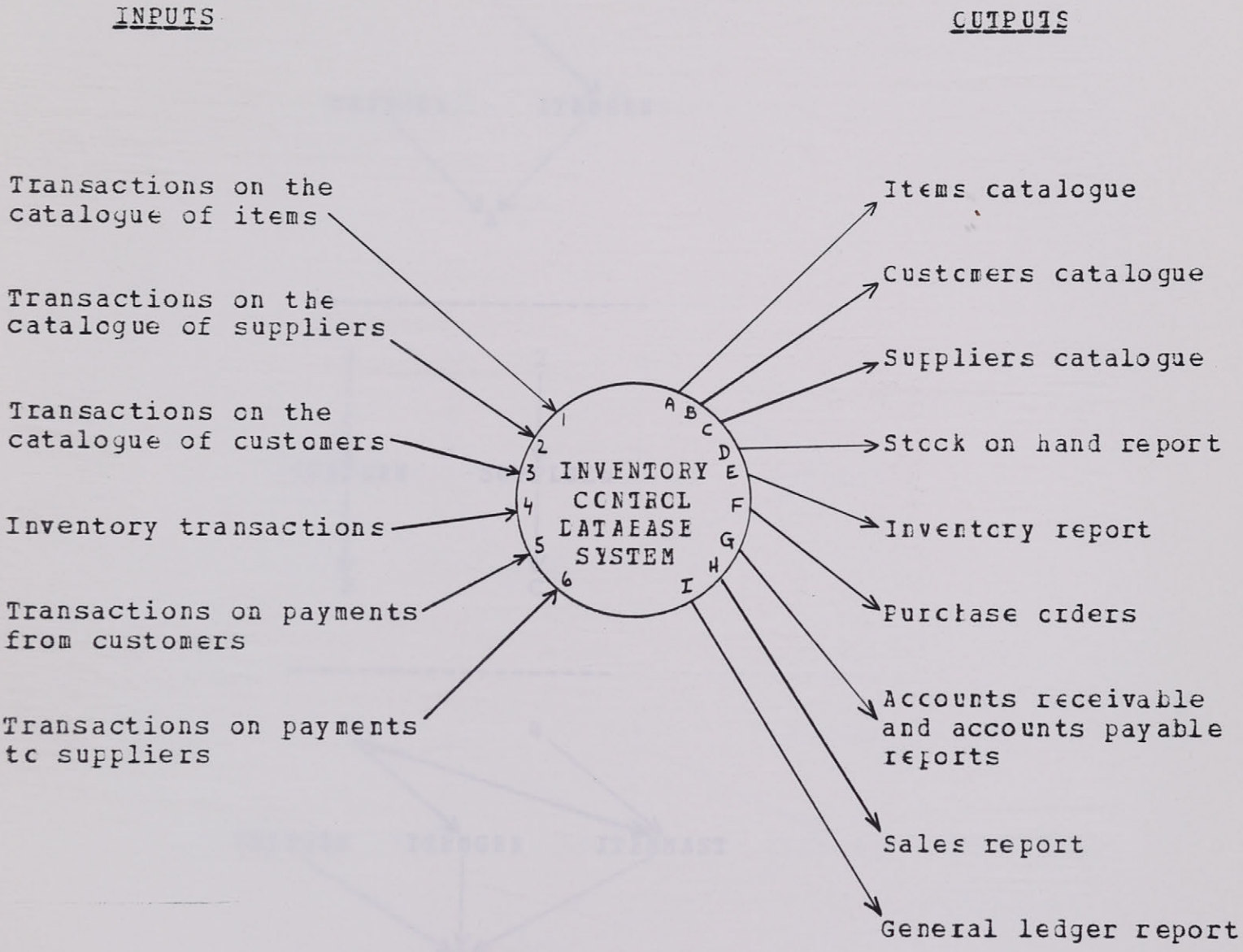
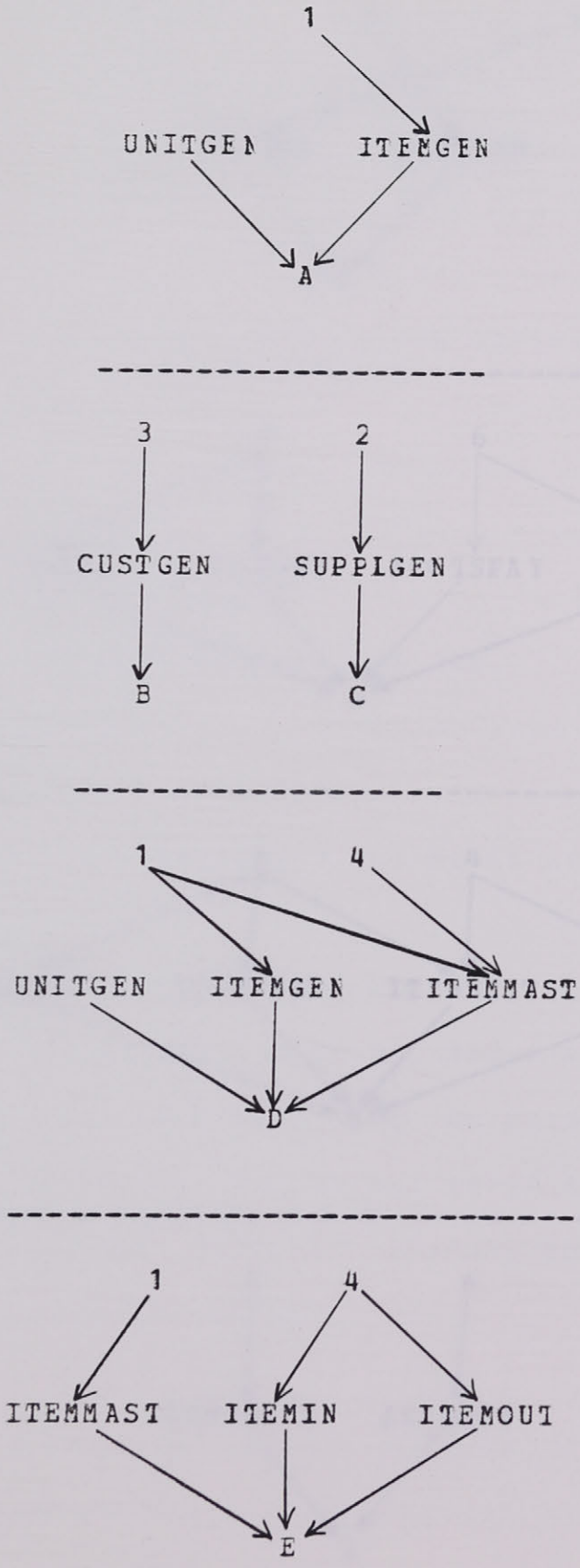
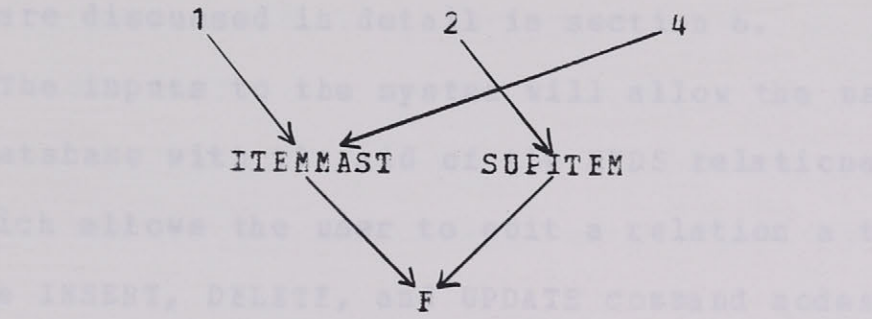


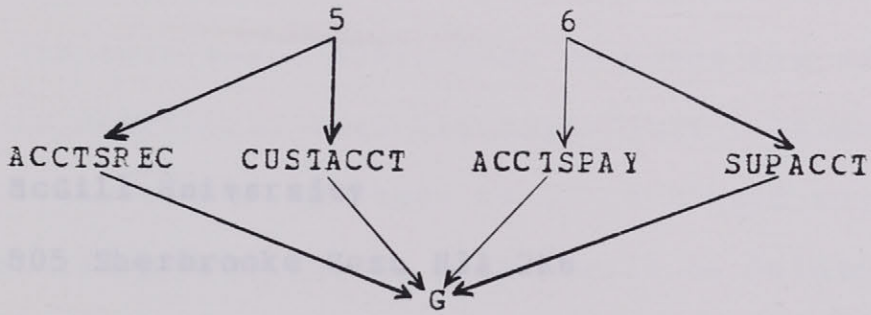
Fig. 2 SYSTEM DIAGRAM

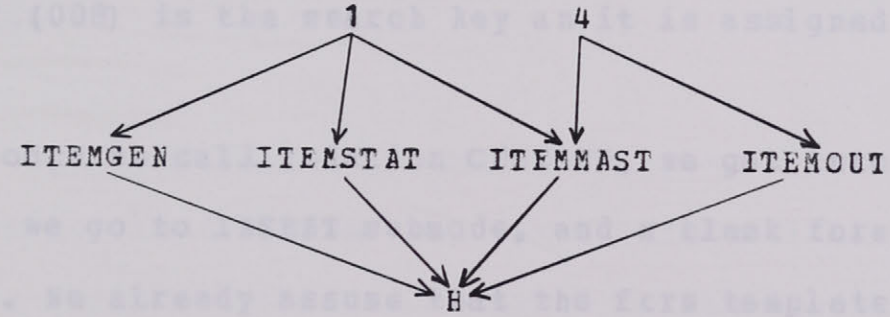
Fig. 2.1 SYSTEM CONFIGURATION

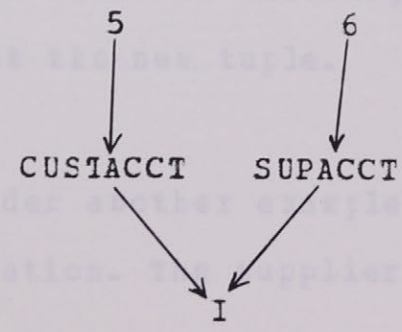


(CONT) SYSTEM CONFIGURATION









The inventory control database system will store the relations (database files), both permanent and temporary. These relations are discussed in detail in section 6.

The inputs to the system will allow the user to enter data into the database with the aid of the MRDS relational editor (refer to [3]) which allows the user to edit a relation a tuple at a time through the INSERT, DELETE, and UPDATE command modes. Let's take an example of a customer entry into the CUSTGEN relation (see section 6). The customer info is

CUST : 008
CUSTNAME : McGill University
CUSTADDR : 805 Sherbrooke West H3A 2K6
CUSTPH : (514)392-5073

where CUST (008) is the search key an it is assigned by the user.

1) First, once we call relation CUSTGEN, we get into PROCESS mode.
2) Second, we go to INSERT submode, and a blank form will appear on the screen. We already assume that the form template of the relation was designed previously. Now we can enter the tuple by filling out the form. When we have finished entering the tuple, we hit CRTL-C on the keyboard to accept the new tuple.

Let's consider another example of a supplier deletion from the SUPPLGEN relation. The supplier info is

SUPPL : 339

SUPNAME : DOMTAR PAPER

SUPADDR : 950 Maisonneuve West H1C 1S6

SUPPH : (514)681-3377

where SUPPL (339) is the search key, as assigned by the user.

1) First, once we call relation SUPPLGEN, we get into PROCESS mode. 2) Second, we go to FIND submode and the editor will ask us for the search key (in this case, 339). Once we enter the search key, if the search is successful then the tuple just retrieved will be displayed on the screen and then we go to DELETE submode and the current tuple displayed on the screen will be deleted automatically, otherwise we get an error that the desired tuple does not exist.

ITEMS CATALOGUE

This report will list general information on all items which are handled in the different warehouses. It will be generated on paper every month and will be received by the warehouse managers in order to fill out the inventory forms (see Fig. 6 in section 5). Moreover, the user can check if the listed items are still in use and make the necessary updates.

CUSTOMERS CATALOGUE

This report will list general information on customers. It will be generated every month and will be received by the user in order to check if the information listed is correct. He can otherwise make the necessary changes.

SUPPLIERS CATALOGUE

This report will list general information on suppliers which supply the necessary items to the warehouses. It will be generated on paper every month and received by the user to revise the information and make the updates if needed.

STOCK ON HAND REPORT

With this report, the user will be informed on the current stock point of all items in the warehouses. It will be generated on paper every two or four weeks and received by the warehouse managers to compare the obtained results against the physical inventory in order to reduce loss or theft of merchandise.

INVENTORY REPORT

This report will list all the inventory transactions performed in the different warehouses. It will be generated on paper every week and received by the user to compare the listed results against supplier invoices and customer factures, and make adjustments if necessary.

PURCHASE ORDERS

With this report, the user will know precisely which items are below their reorder level and their associated economic order quantity (EOQ) and the supplier that provides the minimum cost per unit for the item to be supplied. It will be generated on paper daily at the end of the day and received by the user to take the appropriate action.

ACCOUNTS RECEIVABLE AND ACCOUNTS PAYABLE REPORTS

These reports will list all payments made to suppliers and all payments received from customers. They will list subtotals by customer and by supplier. They will be generated on paper every month and recived by the user to check if the information listed is correct and make adjustments if needed.

SALES REPORT

This report will inform the user of how all items in the warehouses are being sold. It will list subtotals by item and by warehouse. It will be generated on paper every month and received by the user in order to help him for a better planning in the future.

GENERAL LEDGER REPORT

This report will inform the user on the present economic and financial situation. It will be generated at anytime the user requests it and could be obtained on paper or displayed on the screen.

5.- ANALYSIS AND DESIGN OF INPUTS TO THE SYSTEM

1) TRANSACTIONS ON THE CATALOGUE OF ITEMS (Fig. 3)

2) TRANSACTIONS ON THE CATALOGUE OF SUPPLIERS (Fig. 4)

3) TRANSACTIONS ON THE CATALOGUE OF CUSTOMERS (Fig. 5)

4) INVENTORY TRANSACTIONS (Fig. 6)

5) TRANSACTIONS ON PAYMENTS FROM CUSTOMERS

6) TRANSACTIONS ON PAYMENTS TO SUPPLIERS

The input forms in Figs. 3, 4, and 5 are used by the user (Administration department, Sales department, etc.) every time he wants to enter data into the database. These three input forms are first filled out by the user and then later given to the person who will be in charge of updating the system relations in the database. The user has the task of assigning a code number to WAREHOUSE#, ITEM#, SUPPLIER#, and CUSTOMER#. In addition, the user has also to calculate (forecast) the parameters of the input form in Fig. 3, based on historic data. The input form in Fig. 4 will help the user to record all price lists from suppliers in order to select the minimum cost per unit when the user prepares the purchase orders.

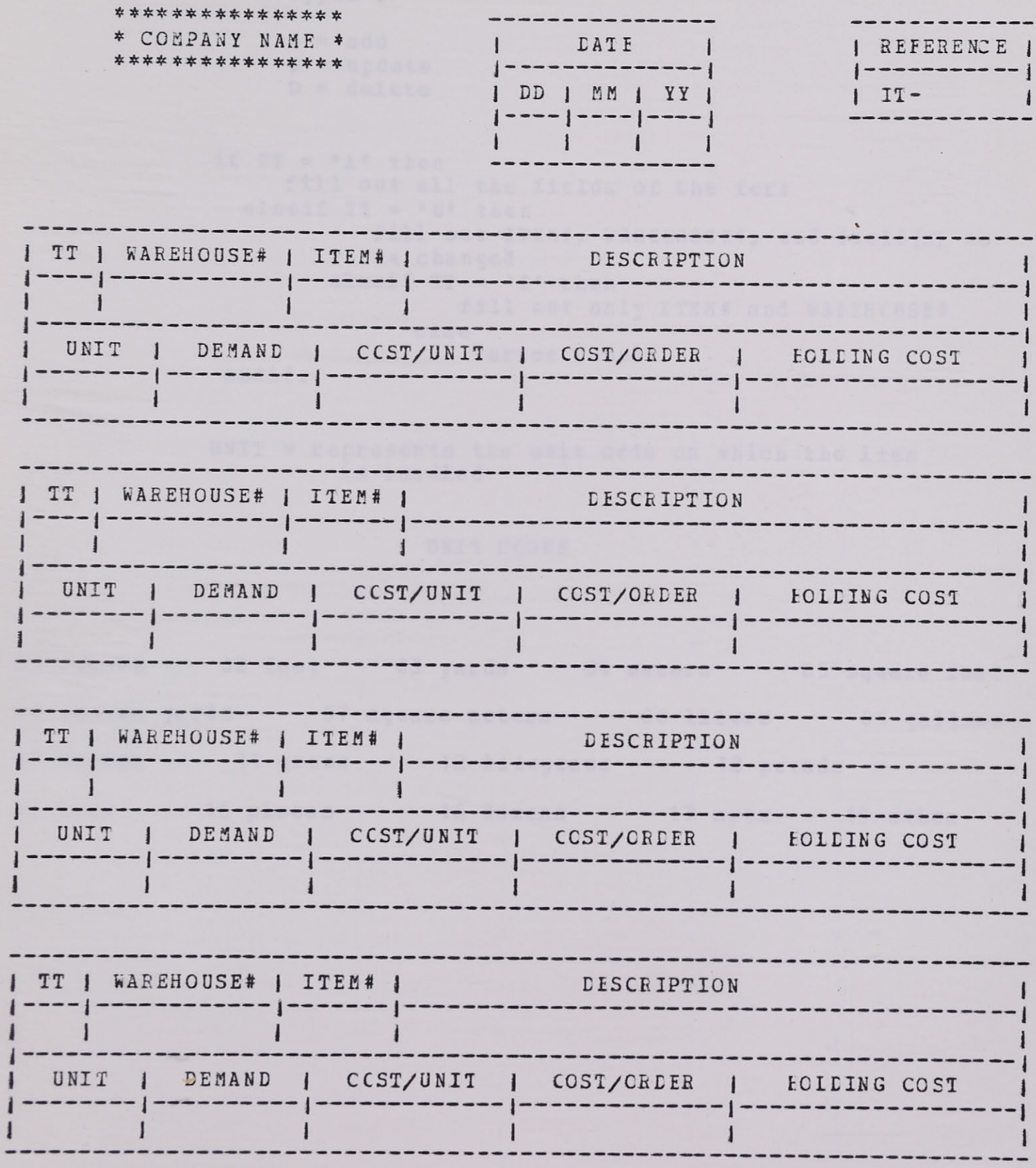
With the input form in Fig. 5, the user will keep track of all customers when he bills them later.

The input form in Fig. 6 is used by the warehouse managers who fill it out every time there is an inventory transaction coming in and out of the warehouses.

The input forms for payments from customers and payments to suppliers are already on paper in the form of customer bills and supplier invoices which both contain the necessary information (total amount of the payment, date, etc.). Therefore, we did not have to design the input forms (layouts) for these two types of transactions.

A brief discussion to all these input forms with examples was given previously in section 4.

Fig. 3 TRANSACTIONS ON THE CATALOGUE OF ITEMS



TT = transaction type

Types :

A = add
U = update
D = delete

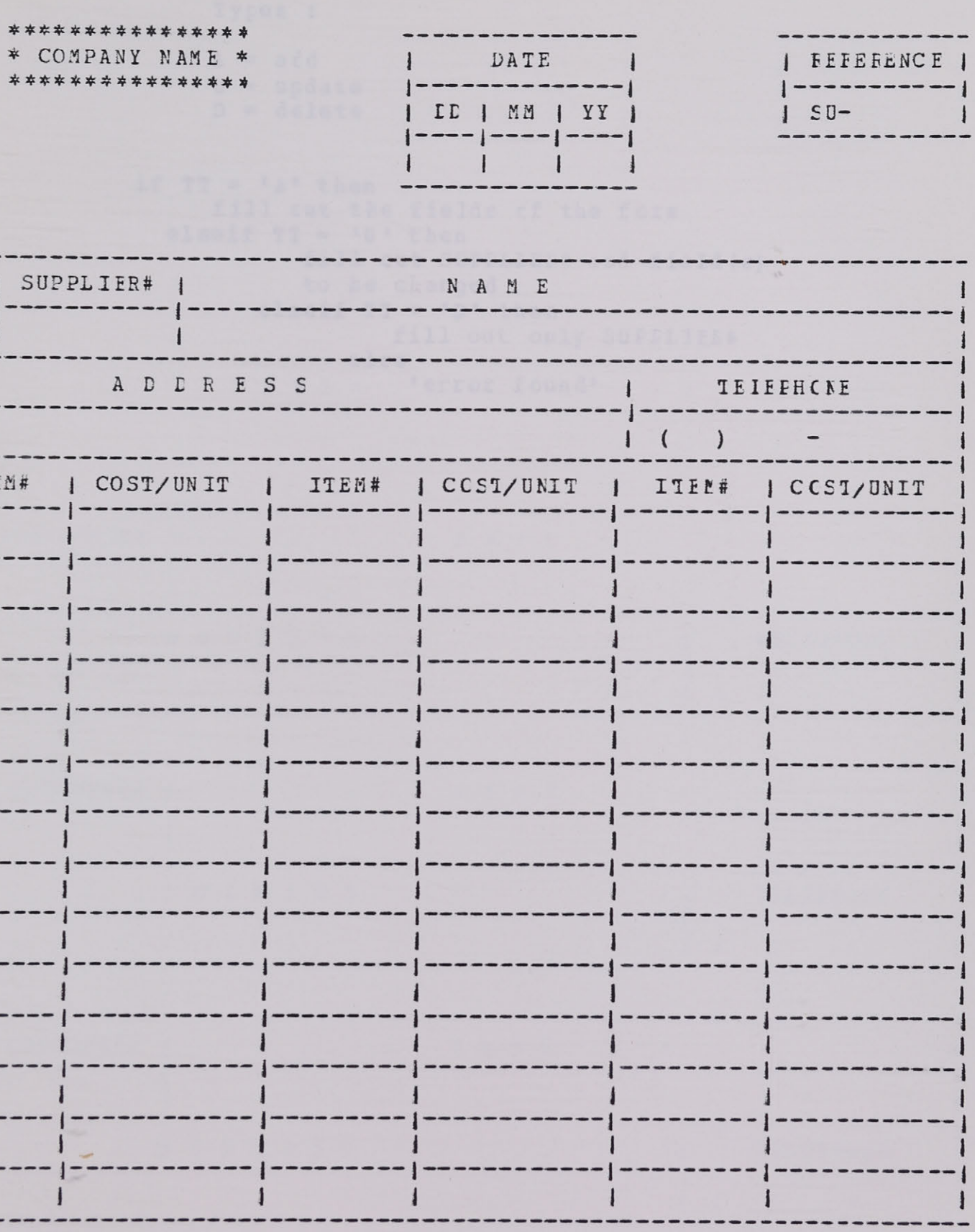
```
if TT = 'A' then
     fill out all the fields of the form
  elseif TT = 'U' then
           fill out ITEM#, WAREHOUSE#, and field(s) to
           be changed
       elseif TT = 'D' then
                fill out only ITEM# and WAREHOUSE#
            else
                'error found'
 endif.
```

UNIT = represents the unit code on which the item is handled

UNIT CODES

01 inches	02 feet	03 yards	04 meters	05 square feet
06 square yards	07 square meters	08 liters	09 gallons	
10 ounces	11 grams	12 kilograms	13 pounds	
14 tons	15 pieces	16 dozens	17 sets	19 other

Fig. 4 TRANSACTIONS ON THE CATALOGUE OF SUPPLIERS



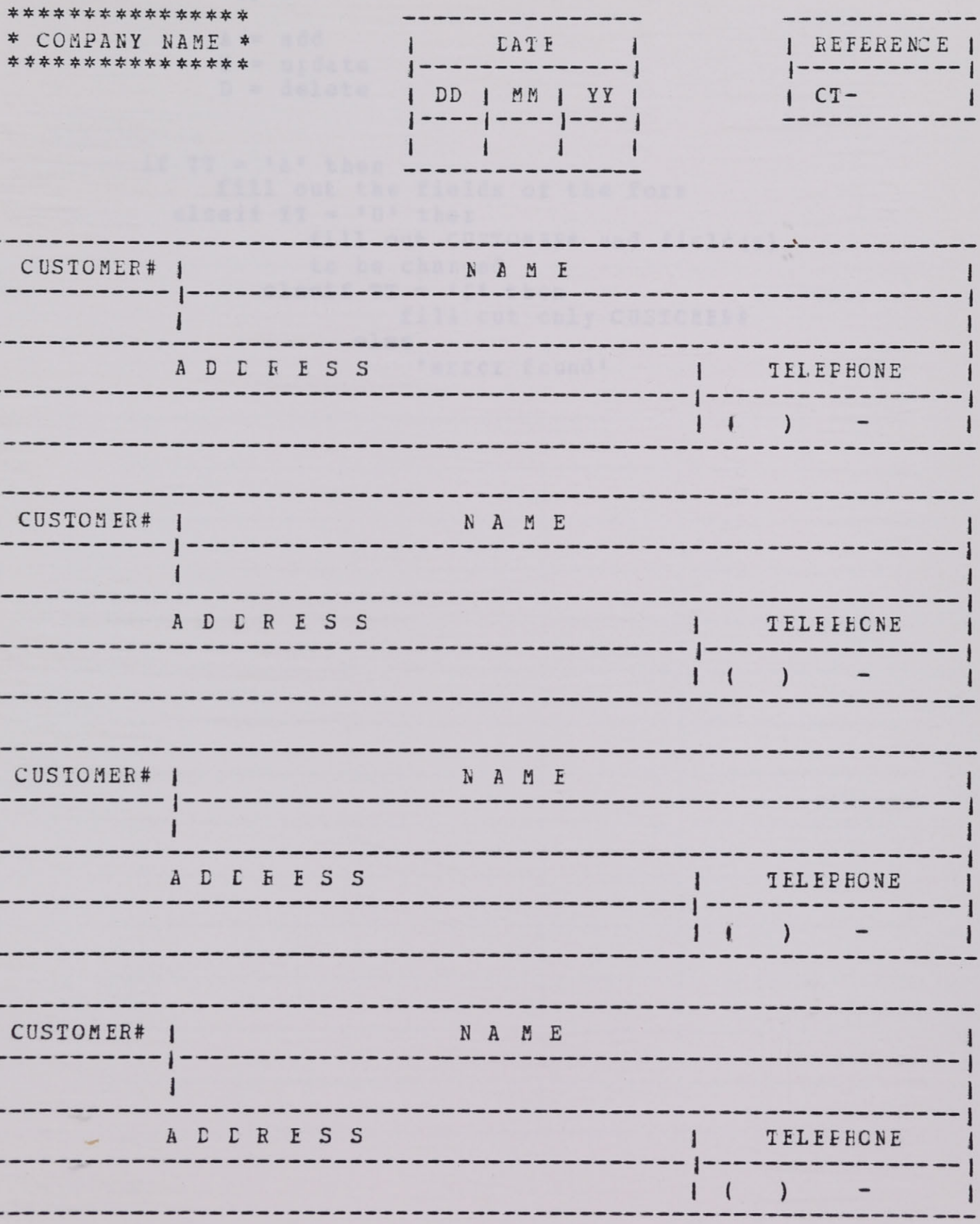
TT = transaction type

Types :

A = add
U = update
D = delete

```
if TT = 'A' then
     fill cut the fields cf the form
  elseif TT = 'U' then
           fill cut SUPPLIER# and field(s)
           to be changed
        elseif TT = 'D' then
                 fill out only SUPPLIER#
              else
                  'error found'
```

Fig. 5 TRANSACTIONS ON THE CATALOGUE OF CUSTOMERS



```
TT = transaction type

        Types :

        A = add
        U = update
        D = delete


   if TT = 'A' then
        fill out the fields of the form
     elseif TT = 'U' then
              fill out CUSTOMER# and field(s)
              to be changed
          elseif TT = 'D' then
                   fill out only CUSTOMER#
               else
                    'error found'
```

Fig. 6 INVENTORY TRANSACTIONS

```
**********************
* COMPANY NAME *
**********************

-----------------
| WAREHOUSE# |
|---------------|
|               |
-----------------

-----------------
|     DATE      |
|---------------|
| DD | MM | YY  |
|----|----|-----|
|    |    |     |
-----------------

-------------
| REFERENCE |
|-----------|
| IN-       |
| OUT-      |
-------------
```

ITEM#	DESCRIPTION	UNIT	QUANTITY

WAREHOUSE MANAGER

6.- DESCRIPTION OF THE RELATIONS USED IN THE SYSTEM

RELATION ITEMMAST

ITEMMAST (WH, ITEM, EOQ, QTY, REORDER)

where <WH, ITEM> is the key of the relation. It contains information on current stock on hand per item and the reorder level associated with each item.

attribute-name	attribute-description	length(bytes)
WH	warehouse code	2/N
ITEM	item code	5/N
EOQ	economic order quantity	4/N
QTY	stock on hand	4/N
REORDER	reorder level	4/N
	record length	19

N = numeric data C = character data

Example 5 : Display relation ITEMMAST.

ITEMMAST

WH	ITEM	EOQ	QTY	REORDER
01	20116	28	58	35
01	55724	76	22	25
01	25279	9	8	3
02	20401	4	4	4
02	23138	6	5	1
03	09316	375	120	77
03	20116	31	39	42
03	20401	3	5	3
04	20116	24	28	25
04	27662	17	8	7
04	55724	65	53	17

RELATION ITEMGEN

ITEMGEN (ITEM, ITEMDESC, UNIT)

where <ITEM> is the key of the relation. It contains general information on all the items.

attribute-name	attribute-description	length (bytes)
ITEM	item code	5/N
ITEMDESC	item description	20/C
UNIT	unit code of the item	2/N
	record length	27

Example 6 : Display relation ITEMGEN.

ITEMGEN

ITEM	ITEMDESC	UNIT
20116	blender 14-speed	15
20087	food processor	17
20401	microwave oven	15
68005	luxor rug	6
55724	white paint	8
23138	portable recorder	15
27662	table lamp 25" h	15
25279	pocket camera instant	15
09316	brown sugar	12

RELATION WHGEN

WHGEN (WH, WHDESC)

where <WH> is the key of the relation. It contains general information on all the warehouses.

attribute-name	attribute-description	length (bytes)
WH	warehouse code	2/N
WHDESC	warehouse description	20/C
	record length	22

Example 7 : Display relation WHGEN.

WHGEN

WH	WHDESC
01	PASCAL Longueuil
02	PASCAL Montreal
03	PASCAL Laval
04	PASCAL Quebec

RELATION UNITGEN

UNITGEN (UNIT, UNITDESC)

where <UNIT> is the key of the relation. It contains general information on the type of units that are handled (see Fig. 3).

attribute-name	attribute-description	length (bytes)
UNIT	unit code	2/N
UNITDESC	unit description	15/C
	record length	17

RELATION CUSTGEN

CUSTGEN (CUST, CUSTNAME, CUSTADDR, CUSTPH)

where <CUST> is the key of the relation. It stores general information on all our customers.

attribute-name	attribute-description	length (bytes)
CUST	customer code	3/N
CUSTNAME	customer name	20/C
CUSTADDR	customer address	30/C
CUSTPH	customer phone number	13/C
	record length	66

Example 8 : Display relation CUSTGEN.

CUSTGEN

CUST	CUSTNAME	CUSTADDR	CUSTPH
061	Nikos Leoust	Park 1320 Montral,Que	(514)286-4359
144	Frank Lok	Guy 850 Quebec,Que	(610)396-1086
296	Robert Amper	Reed 662 Laval,Que	(514)417-2233
580	Gerard Marelle	Stanley 730 Montreal,Que	(514)937-8447
631	Frank Wayne	Peel 3150 Montreal,Que	(514)392-5691
655	Chris Statis	Clark 490 Montral,Que	(514)845-2086

RELATION SUPPLGEN

SUPPLGEN (SUPPL, SUPNAME, SUPADDR, SUPPH, SUPCRED)

where <SUPPL> is the key of the relation. It stores general information on all our suppliers.

attribute-name	attribute-description	length (bytes)
SUPPL	supplier code	3/N
SUPNAME	supplier name	20/C
SUPADDR	supplier address	30/C
SUPPH	supplier phone number	10/N
SUPCRED	supplier credit cond.	10/C
	record length	73

Example 9 : Display relation SUPPLGEN.

SUPPLGEN

SUPPL	SUPNAME	SUPADDR	SUPPHH
012	General Electric	200 Lincoln Houston,Tx	(664)266-1359
069	Osterizer Prod.	865 St Laurent Toronto,Ont	(416)356-9086
087	SONY Ltd.	3020 Pine Buffalo,NY	(212)443-3233
105	Duppont Inc.	655 Lapierre Kingston,Ont	(613)927-2447
214	Panasonic	Km. 22 TransCanada	(514)300-8691
426	Moulinex Inter.	Km. 160 Aut. Laurentians	(514)866-0086

RELATION ITEMSTAT

ITEMSTAT (WH, ITEM, DEMAND, COSTUNIT, COSTORD, COSTHOLD, LEADTIME, EOQ, N, REORDER, H)

where <WH, ITEM> is the key of the relation. It contains statistical information on all the items which are used in the different warehouses.

attribute-name	attribute-description	length (bytes)
WH	warehouse code	2/N
ITEM	item code	5/N
DEMAND	item demand per period	5/N
COSTUNIT	unit cost per item	7.2/N
COSTORD	cost to order new inven	6.2/N
COSTHOLD	inventory holding cost	2.2/N
LEADTIME	acquisition lead time	2/N
EOQ	economic order quantity	4/N
N	number of orders	2/N
REORDER	reorder stock level	4/N
H	avg. stock on hand	4/N
	record length	43

EOQ, N, REORDER, and H are derived from other other domains by calculation (see section 3).

Example 10 : Display relation ITEMSTAT.

ITEMSTAT

WH	ITEM	DEMAND	COSTUNIT	COSTORD	COSTHOLD	LEADTIME	ECQ	N	REORDER	H
01	20116	318	54.00	10.00	.15	10	28	11	35	14
01	55724	750	10.35	6.00	.15	3	76	10	25	38
01	25279	36	49.95	9.00	.15	7	9	4	3	5
02	20401	23	659.00	33.00	.15	15	4	6	4	2
02	23138	19	104.50	12.00	.15	5	6	3	1	3
03	09316	6980	2.65	4.00	.15	1	375	19	77	188
03	20116	381	54.00	10.00	.15	10	31	12	42	16
03	20401	17	659.00	33.00	.15	15	3	6	3	2
04	20116	229	54.00	10.00	.15	10	24	10	25	12
04	27662	170	37.60	5.00	.15	4	17	10	7	9
04	55724	550	10.35	6.00	.15	3	65	8	18	33

RELATION ITEMIN

ITEMIN (WH, ITEM, CHRONCS1, QTYIN)

where <WH, ITEM, CHRONCS1> is the key of the relation. It records all the item transactions coming into the warehouses.

attribute-name	attribute-description	length (bytes)
WH	warehouse code	2/N
ITEM	item code	5/N
CHRONOS1	date and time of the transaction (YYMMDDHHMM)	14/C
QTYIN	input quantity	4/N
	record length	25

Example 11 : Display relation ITEMIN

ITEMIN

WH	ITEM	CHRONOS1	QTYIN
01	55724	84/04/12,10:35	45
02	20401	84/04/12,09:22	3
03	20116	84/04/12,16:10	20
04	20116	84/04/12,18:24	20
04	55724	84/04/12,12:00	167

RELATION ITEMOUT

ITEMOUT (WH, ITEM, CHRONOS2, QTYOUT)

where <WH, ITEM,CHRONOS2> is the key of the relation.

It records all the item transactions coming out of the warehouses.

attribute-name	attribute-description	length (bytes)
WH	warehouse code	2/N
ITEM	item code	5/N
CHRONOS2	date and time of the transaction (YYMMDDHHMM)	14/C
QTYOUT	output quantity	4/N
	record length	25

Example 12 : Display relation ITEMOUT.

ITEMOUT

WH	ITEM	CHRONOS2	QTYOUT
01	20116	84/04/12,11:03	10
02	23138	84/04/12,13:44	1
03	20401	84/04/12,10:50	1
03	09316	84/04/12,15:12	5
04	20116	84/04/12,16:36	2

RELATION SUPITEM

SUPITEM (SUPPL, ITEM, COSTUNIT)

where <SUPPL, ITEM> is the key of the relation. It stores all the price lists of the suppliers.

attribute-name	attribute-description	length(bytes)
SUPPL	supplier code	3/N
ITEM	item code	5/N
COSTUNIT	unit cost per item	7.2/N
	record length	15

Example 13 : Display relation SUPITEM

SUPITEM

SUPPL	ITEM	COSTUNIT
012	20116	45.00
012	20401	623.35
069	20116	52.00
069	20087	127.20
087	23138	88.00
105	55724	8.95
105	68005	26.60
214	20401	589.50
214	23138	86.40
426	20116	44.00
426	20087	123.75

RELATION ACCTSREC

ACCTSREC (COLREF, DATE, CUST, FACTURE, AMOUNT)

where <COLREF> is the key of the relation. It records all the payments made from customers.

attribute-name	attribute-description	length (bytes)
COLREF	bill reference	5/N
DATE	date of the payment(YYMMDD)	6/C
CUST	customer code	3/N
FACTURE	facture number	4/N
AMOUNT	amount of the payment	8.2/N
	record length	28

Example 14 : Display relation ACCTSREC.

ACCTSREC

COLREF	DATE	CUST	FACTURE	AMOUNT
00005	84/04/06	144	0066	100.00
00017	84/04/06	580	0085	323.00
00062	84/04/06	655	0103	95.50
00083	84/04/13	144	0146	72.00
00141	84/04/15	296	0159	250.00
00210	84/04/21	580	0231	170.25

RELATION ACCTSPAY

ACCTSPAY (PAYREF, DATE, SUPPL, INVOICE, AMOUNT)

where <PAYREF> is the key of the relation. It records all the payments made to suppliers.

attribute-name	attribute-description	length (bytes)
PAYREF	invoice reference	5/N
DATE	date of the payment (YYMMDD)	8/C
SUPPL	supplier code	3/N
INVOICE	invoice number	4/N
AMOUNT	amount of the payment	8.2/N
	record length	28

Example 15 : Display relation ACCTSPAY.

ACCTSPAY

PAYREF	DATE	SUPPL	INVOICE	AMOUNT
00016	84/04/09	087	0020	3275.00
00022	84/04/10	105	0135	4826.00
00027	84/04/10	012	0260	5910.75
00035	84/04/15	214	0348	6131.00
00094	84/04/16	087	0357	10552.00
00159	84/04/27	105	0563	1630.35

RELATION CUSTACCT

CUSTACCT (CUST, CUSTBAL)

where <CUST> is the key of the relation. It stores accounting information on all customers.

attribute-name	attribute-description	length
CUST	customer code	3/N
CUSTBAL	customer balance	8.2/N
	record length	11

Example 16 : Display relation CUSTACCT.

CUSTACCT

CUST	CUSTBAL
144	-1520.00
580	-600.00
655	-950.00

RELATION SUPACCT

SUPACCT (SUPPL, SUPPLBAL)

where <SUPPL> is the key of the relation. It stores accounting information on all suppliers.

attribute-name	attribute-description	length
SUPPL	supplier code	3/N
SUPPLBAL	supplier balance	8.2/N
	record length	11

Example 17 : Display relation SUPACCT.

SUPACCT

SUPPL	SUPPLBAL
087	-35200.00
105	-18900.00
214	-10650.00

7.- ANALYSIS AND DESIGN OF OUTPUTS FROM THE SYSTEM

A) ITEMS CATALOGUE

B) CUSTOMERS CATALOGUE

C) SUPPLIERS CATALOGUE

D) STOCK ON HAND REPORT

E) INVENTORY REPORT

F) PURCHASE ORDERS

G) ACCOUNTS RECEIVABLE AND ACCOUNTS PAYABLE REPORTS

H) SALES REPORT

I) GENERAL LEDGER REPORT

A brief discussion on these reports was given in section 4.

NOTE : All the following examples are based on data from the relations in section 6.

ITEMS CATALOGUE

Let TOT/WH be egv + of ITEM by WH

ITEMCAT <- WH, ITEM, ITEMDESC, UNITDESC, TOT/WH in ITEMSTAT ijoin
(ITEMGEN ijoin UNITGEN)

Example 18 : Display items catalogue.

ITEMCAT

WH	ITEM	ITEMDESC	UNITDESC	TOT/WH
01	20116	blender 14-speed	pieces	3
01	55724	white paint	liters	3
01	25279	pocket camera instant	pieces	3
02	20401	microwave oven	pieces	2
02	23138	portable recorder	pieces	2
03	09316	brown sugar	kilograms	3
03	20116	blender 14-speed	pieces	3
03	20401	microwave oven	pieces	3
04	20116	blender 14-speed	pieces	3
04	27662	table lamp 25" h	pieces	3
04	55724	white paint	liters	3

CUSTOMERS CATALOGUE

Let TOT/CUST be red + of 1

CUSTCAT <- CUST, CUSTNAME, CUSTADDR, CUSTPH, TOT/CUST in CUSTGEN

(see example 18)

SUPPLIERS CATALOGUE

Let TOT/SUP be red + of 1

SUPPLCAT <- SUPPL, SUPNAME, SUPADDR, SUPPH, TOT/SUP in SUPPIGEN

(see example 18)

STOCK ON HAND REPORT

```
Let QTY/ITEM be eqv + of QTY by ITEM

STOCKREP <- WH, ITEM, ITEMDESC, UNITDESC, QTY, QTY/ITEM

            in ITEMMAST ijoin (ITEMGEN ijoin UNITGEN)
```

Example 19 : Display current stock on hand.

STOCKREP

WH	ITEM	ITEMDESC	UNITDESC	QTY	QTY/ITEM
01	20116	blender 14-speed	pieces	58	125
01	55724	white paint	liters	22	75
01	25279	pocket camera instant	pieces	8	8
02	20401	microwave oven	pieces	4	9
02	23138	portable recorder	pieces	5	5
03	09316	brown sugar	kilograms	120	120
03	20116	blender 14-speed	pieces	39	125
03	20401	microwave oven	pieces	5	9
04	20116	blender 14-speed	pieces	28	125
04	27662	table lamp 25" h	pieces	6	6
04	55724	white paint	liters	53	75

INVENTORY REPORT

```
Let SUBTOTIN be eqv + of QTYIN by ITEM

Let SUBTOTOU be eqv + of QTYOUT by ITEM

MERCHIN <- WH, ITEM, CHRONOS1, QTYIN, SUBTOTIN in ITEMIN

MERCHOUT <- WH, ITEM, CHRONOS2, QTYOUT, SUBTOTOU in ITEMOUT

Let NEWQTY be QTYIN - QTYOUT

INVREP <- WH, ITEM, ECQ, NEWQTY, REORDER in MERCHIN rjoin ITEMMAST

          ljoin MERCHOUT

ITEMMAST [ WH, ITEM, EOQ, QTY, REORDER, <- WH, ITEM, ECQ, NEWQTY,

         REORDER ] INVREP
```

Example 20 : Display invertcry repcrt with the updates to ITEMMAST

MERCHIN

WH	ITEM	CHRONOS1	QTYIN	SUBTCTIN
01	55724	84/04/12,10:35	45	212
02	20401	84/04/12,09:22	3	3
03	20116	84/04/12,16:10	20	40
04	20116	84/04/12,18:24	20	40
04	55724	84/04/12,12:00	167	212

MERCHOUT

WH	ITEM	CHRONOS2	QTYOUT	SUBTCTCU
01	20116	84/04/12,11:03	10	12
02	23138	84/04/12,13:44	1	1
03	20401	84/04/12,10:50	1	1
03	09316	84/04/12,15:12	5	5
04	20116	84/04/12,16:36	2	12

ITEMMAST

WH	ITEM	EOQ	QTY	REORDER
01	20116	28	48	35
01	55724	76	67	25
01	25279	9	8	3
02	20401	4	7	4
02	23138	6	4	1
03	09316	375	115	77
03	20116	31	59	42
03	20401	3	4	3
04	20116	24	46	25
04	27662	17	8	7
04	55724	65	220	17

PURCHASE ORDERS

```
Let MINCOST be eqv min of COSTUNIT by ITEM
DUMMY1 <- WH, ITEM, EOQ where QTY <= REORDER in ITEMMAST
DUMMY2 <- SUPPL, ITEM, COSTUNIT where COSTUNIT = MINCOST in SUPITEM
PURCHASE <- WH, ITEM, SUPPL, EOQ, COSTUNIT in DUMMY1 ijcin DUMMY2
```

Example 21 : Display purchase orders.

PURCHASE

WH	ITEM	SUPPL	EOQ	COSTUNIT
01	55724	105	76	8.95
02	20401	214	4	589.50
03	20116	426	31	44.00

ACCOUNTS RECEIVABLE AND ACCOUNTS PAYABLE REPORTS

```
Let SUBTOT/C be eqv + of AMOUNT by CUST
Let SUBTOT/S be eqv + of AMOUNT by SUPPL
Let TOT/C be red + of AMOUNT
Let TOT/S be red + of AMOUNT
ACCTREC <- CUST, COIREF, DATE, AMOUNT, SUBTOT/C, TOT/C in ACCTSREC
ACCTPAY <- SUPPL, PAYREF, DATE, AMOUNT, SUBTOT/S, TOT/S in ACCTSPAY
Let NEWCBAL be CUSTBAL + SUBTOT/C
Let NEWSBAL be SUPPLBAL + SUBTOT/S
ARECREP <- CUST, NEWCBAL in CUSTACCT ljoin ACCTREC
APAYREP <- SUPPL, NEWSBAL in SUPACCT ljoin ACCTPAY
CUSTACCT [ CUST, CUSTBAL <- CUST, NEWCBAL ] ARECREP
SUPACCT [ SUPPL, SUPPLBAL <- SUPPL, NEWSBAL ] APAYREP
```

Example 22 : Display accounts receivable and accounts payable reports with the updates to CUSTACCT and SUPACCT.

ACCTREC

COLREF	DATE	CUST	FACTURE	AMOUNT	SUBTOT/C	TOT/C
00005	84/04/06	144	0066	100.00	172.00	610.75
00017	84/04/06	580	0085	323.00	497.25	610.75
00062	84/04/06	655	0103	95.50	95.50	610.75
00083	84/04/13	144	0146	72.00	172.00	610.75
00141	84/04/15	296	0159	250.00	250.00	610.75
00210	84/04/21	580	0231	170.25	497.25	610.75

ACCTPAY

PAYREF	DATE	SUPPL	INVOICE	AMOUNT	SUBTOT/S	TOT/S
00016	84/04/09	087	0020	3275.00	13827.00	32325.00
00022	84/04/10	105	0135	4826.00	6456.25	32325.00
00027	84/04/10	012	0260	5910.75	5910.75	32325.00
00035	84/04/15	214	0348	6131.00	6131.00	32325.00
00094	84/04/16	087	0357	10552.00	13827.00	32325.00
00159	84/04/27	105	0563	1630.35	6456.25	32325.00

CUSTACCT

CUST	CUSTBAL
144	-1348.30
580	-102.75
655	-854.50

SUPACCT

SUPPL	SUPPLBAL
087	-21373.00
105	-12443.75
214	-4519.00

SALES REPORT

```
Let SALES be QTYOUT * COSTUNIT

Let SUBTOTWH be eqv + of SALES by WH

Let SUBTOTIT be eqv + of SALES by ITEM

Let TOTSALES be red + of SALES

DUMMY3 <- WH, ITEMDESC, QTYOUT, COSTUNIT in ITEMMAST ijcin
          (ITEMSTAT ijoin (ITEMOUT ijoin ITEMGEN))

SALESREP <- WH, ITEMDESC, QTYOUT, COSTUNIT, SALES, SUBTOTWH,
            SUBTOTIT, TOTSALES in DUMMY3
```

Example 23 : Display sales report.

SALESREP

WH	ITEMDESC	QTYOUT	COSTUNIT	SALES	SUBTOTWH	SUBTOTIT	TOTSALES
01	blender 14-speed	10	54.00	540.00	540.00	658.00	1324.75
02	portable recorder	1	104.50	104.50	104.50	104.50	1324.75
03	microwave oven	1	659.00	659.00	672.25	659.00	1324.75
03	brown sugar	5	2.65	13.25	672.25	13.25	1324.75
04	blender 14-speed	2	54.00	108.00	108.00	658.00	1324.75

GENERAL LEDGER REPORT

```
Let CREDIT be red + of SUPPLBAL

Let DEBIT be red + of CUSTBAL

Let BALANCE be DEBIT - CREDIT

LEDGER <- DEBIT, CREDIT, BALANCE in CUSTACCT ujoin SUPACCT
```

Example 24 : Display ledger report.

```
LEDGER
```

DEBIT	CREDIT	BALANCE
2305.00	38335.75	36030.20

8.- BUILDING THE INVENTORY CONTROL DATABASE ON THE IBM PC USING MRDS/FS

MRDS/FS (McGill Relational Database System with Functional Syntax) is an interactive relational expression interpreter built by Ted Van Rossum [4] for his master's thesis in 1983. It provides the user with a complete set of relational programming functions such as the relational algebra operations, domain algebra operations, conditional execution functions, branching functions, and housekeeping functions. MRDS/FS has a facility for relation manipulation in an interactive environment as well as a facility to create new user defined functions (UDF), which allow the user to define a new function as a series of MRDS/FS functions which will be executed one after the other without user intervention.

MRDS/FS is an extension of MRDSA, a Pascal data sub-language built by George Chiu [3] for his master's thesis in 1982, which provides the user with a library of subroutines for the execution of relational operations such as project, nu-join, etc.

MRDS/FS requires the following hardware and software environment :

- 8080, 8065, or Z-80 based microprocessor system with UCSD P-System version 4.0 (operating system)
- 128 K bytes of RAM
- 2 double sided disk drives or hard disk
- MRDS/FS software (object code)

Our system prototype will be implemented on the IBM PC using sample data.

We will show one example of how MRDS/FS works.

Example 25 : Let's consider the stock on hand report described in section 7.

Using ALDAT notation

```
Let QTY/ITEM be eqv + of QTY by ITEM
STOCKREP <- WH, ITEM, ITEMDESC, UNITDESC, QTY, QTY/ITEM
            in ITEMMAST ijoin (ITEMGEN ijoin UNITGEN)
```

Using MRDS/FS functions

```
LET [ QTY/ITEM ] [ EQV + OF QTY BY ITEM ]

TEMP <- MJOIN [ I ] [ ] [ ] ITEMGEN  UNITGEN
TEMP1 <- MJOIN [ I ] [ ] [ ] ITEMMAST TEMP
STOCKREP <- PROJECT [ WH ITEM ITEMDESC UNITDESC QTY QTY/ITEM ] TEMP1
PRINT_REL [ STOCK ON HAND REPORT ] [ PRINTER: ] STOCKREP
```

For a detailed description of the MRDS/FS functions refer to [4].

9.- CONCLUSIONS

The Inventory Control System presented in this report represents a new approach to coping with the inventory control problem. It uses the tools of the relational algebra, where data is organized in the form of relations or tables. The domain algebra, which creates new attributes as functions of existing attributes in a given relation is used. MRDS/FS is an interactive relational expression interpreter, which provides the user with a set of relational programming functions such as relational algebra operations, domain algebra operations, etc. And ALDAT, a high-level programming language, provides the facility for easy programming and general relation manipulations.

We have selected finished goods (end items in stock awaiting shipment to customers) as the basic type of inventory because it does not have many variables attached to inventory model. In addition, we also have selected safety stocks using reorder point as the restocking technique since it is the best suitable for finished products in factory stock and field warehouse items.

At the present time, few commercial systems support inventory control and these provide very easy types of inventory stock such as ABC classification and the prices of these systems are very expensive.

Our inventory control system is capable of generating many reports such as stock-on-hand reports, purchase orders, sales reports, etc., which are very useful to the user for better decision making; moreover, it has the capability of allowing the user to enter data

(transactions on items, on suppliers, on payments, on inventory, etc.) interactively into the system relations (database files) with the aid of the relational editor.

Our system has been designed in such a way that it is feasible to implement it in a microcomputer, such as the IBM PC with a minimum required configuration (refer to section 8), and consequently become affordable for future use.

In contrast, one of the drawbacks of our system is that it is not easy to use by the user, who needs knowledge of the relational algebra and ALDAT; however, it has higher operational power than many commercial systems available in the market.

There is still a great deal of work to do in the MRDS/FS software. A report generator facility could be added to the existing one which is very rudimentary, as well as a facility to enable the user make validation process in the input data edition. Furthermore, a replacement (update) function is also needed to be implemented in order to allow the user to make replacements of domains in the existing relations.

REFERENCES

[1] MERRETT T.H., ALDAT - AUGMENTING THE RELATIONAL ALGEBRA FOR PROGRAMMERS. McGill UNIVERSITY, TECHNICAL REPORT SOCS-78.1, November 1977

[2] MERRETT T.H., RELATIONAL INFORMATION SYSTEMS. RESTON PUBLISHING CO. Reston,Virginia 1984

[3] GEORGE CHIU, MRDSA USER'S MANUAL. McGill UNIVERSITY, TECHNICAL REPORT SOCS-82.9, May 1982

[4] TED VAN ROSSUM, IMPLEMENTATION OF A DOMAIN ALGEBRA AND A FUNCTIONAL SYNTAX. McGill UNIVERSITY. TECHNICAL REPORT SOCS-83.18, August 1983

[5] R. STANSBURY STOCKTON, BASIC INVENTORY SYSTEMS : CONCEPTS AND ANALYSIS. ALIYN AND BACON, 1965

[6] JEROME H. FUCHS, COMPUTERIZED INVENTORY CONTROL SYSTEMS. PRENTICE-HALL, 1978

[7] HILLIER AND LIEBERMAN, INTRODUCTION TO OPERATIONS RESEARCH. HOLDEN-DAY, 1980