PROFIT ANALYSIS AND ACCOUNTING WITH ALDAT

the application of
relational algebra to
business information systems

by

Chin Yun Chen

Pyunghee Kim

Technical Report SOCS - 79.13

July 1979

McGill University
School of Computer Science
Montreal, Canada

PROFIT ANALYSIS AND ACCOUNTING WITH ALDAT

the application of

relational algebra to business information systems

Advisor

Prof. T. H. Merrett

By

Chin Yun Chen

Pyunghee Kim

July 1979

Presented in partial fulfillment of the requirements
for the Degree of Master of Science (Applied)

Acknowledgement

We would like to express our deep gratitude to Prof. T.H. Merrett for his substantial help and guidance during the period of this work.

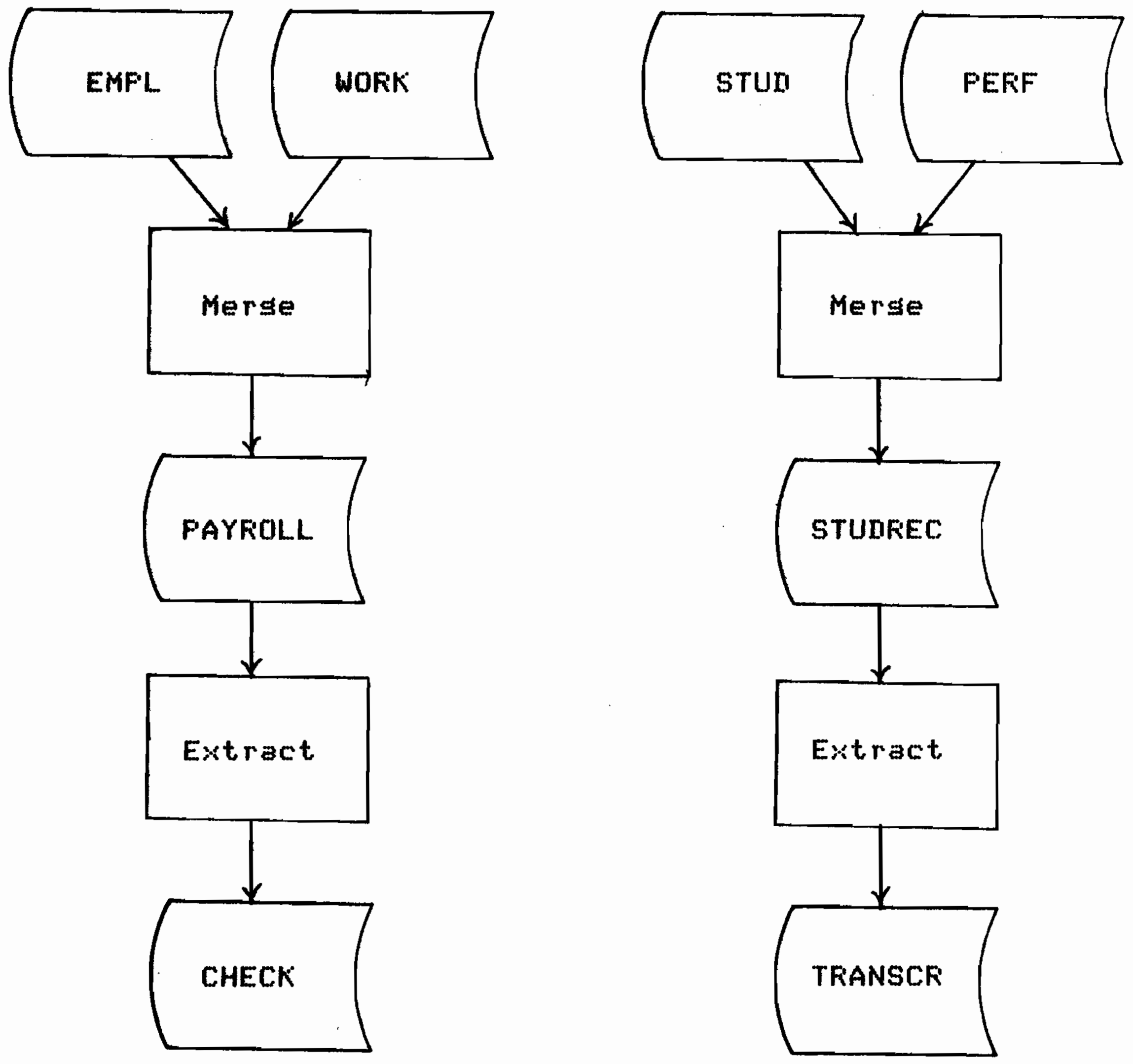
Contents

1. Introduction

Applications programs using existing business data processing languages, like COBOL or PL/I , deal with files by writing loops to handle one record after another. For example, to print a report, a program would extract the necessary information from a file, calculate totals or subtotals and then format the result for a record. This would be repeated for each record. Thus, for instance, a program for payroll check printing and another program for student transcripts are viewed and designed as quite separate problems. Similarly, to merge EMPLoyee file with WORKlog file; and STUDent record with PERFormances are treated differently, with a lot of duplicated effort. Considered from an operational point of view, each of these pairs of problems is essentially one problem.

The desired approach is to be able to deal with each file as a unit and to operate on the file rather than on records. We should treat files the way FORTRAN treats numbers - as fundamental, indivisible units - or as APL treats arrays. For instance, instead of using loops for check printing we could have a single operation on the payroll file. Secondly we should have a framework of standard operations so that each problem can be seen as an application of one or more basic file operations. Instead of using loops to merge two files , EMPL with WORK and STUD with PERF, we could have a single operation on the two files.

The relational algebra developed by E.F.Codd provides this desired approach. Relations are essentially files, and algebraic operations form a framework of standard basic operations on files. These topics are elaborated on in chapter 2. In these terms, a payroll system and a student record system could consist of a merge followed by an extract. These are shown schematically below:



This report shows how the relational algebra, formulated as the programming language Aldat, applies to a manufacturing profit analysis system and to a general accounting system. It demonstrates the conciseness and ease of programming of Aldat in business applications as compared to existing data processing languages.

This report comprises two works - Profit Analysis by Chin Yun Chen; and Accounting Applications by Pyunghee Kim - merged as shown below:

$$\begin{tikzcd}
 & 1. \arrow[d] & & \text{Introduction} \\
 & 2. \arrow[dl] \arrow[dr] & & \text{Aldat in Business} \\
C3. \arrow[d] & & K3. \arrow[d] & \text{(Problem Description)} \\
C4. \arrow[dr] & & K4. \arrow[dl] & \text{(The Application of Aldat)} \\
 & 5. \arrow[d] & & \text{Conclusion} \\
 & A: \arrow[d] & & \text{List of Relations} \\
 & B: & & \text{List of Attributes}
\end{tikzcd}$$

Each application can be read separately except Accounting Applications uses some of the Profit Analysis results as input file. Due to the independent writing of this report, in the following cases different attributes were given for the same relation name:

EQUIP, SALHIST

In most cases, names are consistent.

2. Aldat in Business

Aldat is based on the relational algebra. The relational algebra provides a set of operators for manipulating relations. We choose relations because they are simple, general, and have well-defined mathematical properties. The algebra operates on whole relations not just tuples. It treats the relations as primitive data structures in much the same way that FORTRAN treats scalar numeric variables or APL treats arrays.

The advantages of Aldat are its conciseness and standardization. Applications programs will be at least ten times shorter than equivalent PL/1 or COBOL programs. What was designed as a whole program in a Cobol-oriented system specification will appear as a statement in Aldat. The limited, but powerful, set of operations available in the relational algebra ensures that there is a limited number of ways of doing any particular task, and often only one, standard, way.

This project is an illustration of the applicablity of the relational algebra to accounting and bill-of-materials data processing.

2.1 The relation

In traditional terms, a relation resembles a file, a tuple a record, and a domain a field. To put it another way, relations may be thought of as highly disciplined files - the discipline concerned being one that results in a considerable simplification of the data structures the user has to deal with, and hence in a corresponding simplification of the operators needed to manipulate them.

We illustrate a relation called EQUIP, defined on domains EQUIP# (equipment number), CHRONOL (the date of purchasing), EQCOST (the cost of equipment), SALVAGE (the value of salvage), and LIFE (life limit of the equipment). The domain EQUIP#, for example, is the set of all equipment. It is convenient to represent the relation as a table.

EQUIP (EQUIP#	CHRONOL	EQCOST	SALVAGE	LIFE)
1	730101	20,500	500	10
2	731201	18,500	1,500	5
3	740801	24,400	400	8
4	760101	29,000	1,000	7
5	760601	25,000	1,000	10

Each row of the table represents one tuple of the relation. The number of tuples in this relation is 5.

2.2. The relational operations

The operations of the relational algebra fall into two classes: the monadic operators - which include project and select, and the dyadic operators - which include the μ-join and the range join. We discuss these below.

2.2.1 The select

The select operator constructs a new relation consisting of those tuples of an existing relation that satisfy some conditions (which we defined in the domain expression). A selection function can specify the tuples which we need. For example,

<u>select</u> EQUIP[LIFE10]

in which LIFE10 is a boolean expression. A tuple will be selected only if the expression for the tuple is true. Therefore, the tuples with EQUIP# equal to 1 and 5 are selected.

2.2.2 The project

The project operator, forms a new relation which extracts specified domains from an existing relation and removes any redundant duplicate tuples in the set of domains extracted. For example,

<u>project</u> EQUIP[LIFE]

would give a unary relation consisting of tuples containing 10, 5, 8, and 7.

2.2.3 T-selectors

It is convenient to combine select and project into a single operation. To find the equipment life of 10, we normally write:

<u>project</u> (<u>select</u> EQUIP[LIFE10])[EQUIP#]

which is select on LIFE10 followed by a projection on domain EQUIP#. By using T-selector, we can write this as:

EQUIP# <u>if</u> LIFE=10 <u>in</u> EQUIP.

2.2.4 The μ-join

The μ-join include natural join and union join. They are illustrated with the relations:

ROUTCOST (OP#	ASSEMBLY#	RCOST)
1	T	4.0
2	N	2.4
3	N	1.0

LABOUR (EMP#	OP#	EQUIP#	EMPCOST)
1	1	4	5.0
3	2	5	8.0

Suppose we wish to merge the manufacturing costs into a relation MFGCOST (OP#,ASSEMBLY#,RCOST,EMPCOST). It may be that some operations do not have RCOST or EMPCOST: such operations would not appear if MFGCOST were created using the natural join:

ROUTCOST[OP#] <u>ijoin</u> LABOUR[OP#]

after projection, the result would give

MFGCOST (OP#	ASSEMBLY#	RCOST	EMPCOST)
1	T	4.0	5.0
2	N	2.4	8.0

A complete summary of operations would require a union join, in which all operations with or without costs would appear, accompanied if necessary by a "null" value for one or both costs,

ROUTCOST[OP#] <u>uJoin</u> LABOUR[OP#]

which gives

MFGTEMP (OP# ASSEMBLY# RCOST EMPCOST EMP# EQUIP#)

OP#	ASSEMBLY#	RCOST	EMPCOST	EMP#	EQUIP#
1	T	4.0	5.0	1	4
2	N	2.4	8.0	3	5
3	N	1.0			

Projecting this relation over the domain OP#, ASSEMBLY#, RCOST, and EMPCOST would yield the required information:

MFGCOST (OP# ASSEMBLY# RCOST EMPCOST)

OP#	ASSEMBLY#	RCOST	EMPCOST
1	T	4.0	5.0
2	N	2.4	8.0
3	N	1.0	

2.2.5 The range join

The range join is illustrated by the tax table. Suppose we want to find the base tax, given in the relation TAXTABLE, for employees in the relation SALHIST:

SALHIST (EMP# SALARY)

EMP#	SALARY
1	500
2	475
3	400

TAXTABLE (FLOOR BASE TAX%)

FLOOR	BASE	TAX%
350	50.0	20
400	60.0	30
450	75.0	35
500	92.5	40

The result we want would give a floor of 400 for employee 1, 450 for employee 2, and 500 for employee 3:

PAYTAX (EMP#	SALARY	FLOOR	BASE	TAX%)
1	500	500	92.5	40
2	475	450	75.0	35
3	400	400	60.0	30

The range join is defined, so that paytax is given by:

TAXTABLE[FLOOR] <u>loJoin</u> SALHIST[SALARY].

2.3 The domain operations

The domain algebra is independent of and complementary to the relational algebra. It operates on domains to define a new domain. The resulting new domain has to be virtual. That is, no storage is allocated for or evaluation performed on the result until the result is actualized for a specific relation which includes all the operand domains. There are five forms of domain algebra expressions. We discuss three below.

2.3.1 Scalar operations

In the scalar operation, an arithmetic expression can be evaluated entirely within a single tuple by considering only the values of the operand domain from that tuple, such as

<u>let</u> TAX <u>be</u> BASE + (SALARY - FLOOR) * TAX%

which will give tax payable for each employee.

(EMP#	SALARY	FLOOR	BASE	TAX%	TAX)
1	500	500	92.5	40	92.50
2	475	450	75.0	35	83.75
3	400	400	60.0	30	60.00

2.3.2 Simple reduction

Simple reduction is an aggregation and it is similar to the reduction operation in APL. For example

<u>let</u> TOTAX <u>be</u> <u>red</u> + <u>of</u> TAX

will give the sum of all values of TAX.

(EMP#	SALARY	FLOOR	BASE	TAX%	TAX	TOTAX)
1	500	500	92.5	40	92.50	236.25
2	475	450	75.0	35	83.75	236.25
3	400	400	60.0	30	60.00	236.25

In Aldat, the operation must be a commutative and associative binary operator which can be one of $\vee$, $\wedge$, +, *, <u>max</u>, and <u>min</u>. The usual database aggregation function COUNT can be written as <u>red</u> + <u>of</u> 1 and AVERAGE as (<u>red</u> + <u>of</u> X) / (<u>red</u> + <u>of</u> 1).

2.3.3 Equivalence reduction

Equivalence reduction is a sub-aggregation. It is a generalization of subtotalling. For instance, the total sales profits for each customer would be written as,

<u>let</u> $CPROFIT <u>be</u> <u>equiv</u> + <u>of</u> $PROFIT <u>by</u> CUST#.

For example,

PROFIT (CUST# FGPART# NQTY $SALE $PROFIT SALES#)

8	E	100	800	300	9
10	T	3	105	30	8
10	R	3	45	9	8
17	N	20	3,000	432	7
5	T	80	2,800	800	7

after we project $CPROFIT on CUST#, which would give us the result:

(CUST#	$CPROFIT)
8	300
10	39
17	432
5	800

2.4 History relations

A history relation is a relation in which tuples are never deleted or changed. Updating is accomplished only by adding new tuples.

A record is changed by adding a tuple with the new information but the same key value. A record is deleted by adding a tuple with the same key and empty values.

This leads to the problem of duplicated "keys". We resolve this by augmenting the key by a "chronology" attribute. This gives the date the tuple was added. The date must have fine enough resolution (eg. to the nearest minute) to distinguish all tuples.

We need the notion of the "surface" of the relation. For example, consider the following relation:

SALHIST

EMP#	CHRONOL	SALARY	OVERSALARY
1	740507	3	5
2	750101	4	6
3	750101	3	5
4	760101	3	5
3	771001	4	6
4	771201	5	7
5	781001	3	5
6	781101	2	3
1	790101	5	7

If we want to find the most recent salary for each employee, we get:

<u>let</u> SCHRON <u>be</u> CHRONOL = <u>equiv</u> <u>max</u> <u>of</u> CHRONOL <u>by</u> EMP#;

by: <u>project</u> (<u>select</u> SALHIST[SCHRON])[EMP#,SALARY]

will give the result,

EMP#	SALARY
1	5
2	4
3	4
4	5
5	3
6	2

we can also use T-select:

(EMP#,SALARY) <u>if</u> CHRONOL = <u>equiv</u> <u>max</u> <u>of</u> CHRONOL <u>in</u> SALHIST

which would give the same result as above.

A second need is to find information for a specified period. To find the record of any employee after 1977, we can write

<u>let</u> SCHRON <u>be</u> CHRONOL > 771231

the result of a selection would be:

(EMP#	CHRONOL	SALARY	OVERSALARY)
5	781001	3	5
6	781101	2	3
1	790101	5	7

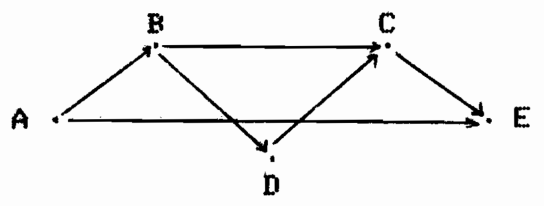
The justification for the history file is that it is easy to update and can be served to keep the business archives for auditing purpose.

2.5 Relational inner product

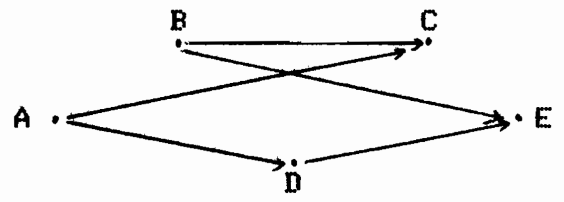
The relational inner product is a combination of natural join, project, and equivalent reduction. It can be illustrated by concatenation of graphs and networks.

A graph can be represented by a matrix, R, where R_{ij} = 1, when i and j are two points connected by an edge and R_{ij} = 0 otherwise. The product, R^2 = R*R, represents paths of length 2, R^3 represents paths of length 3 and so on. These notions may be translated into relational terms, where "product" is read "relational inner product". For example, if we have a graph of five points, we can generate five distinct product graphs.

Step 1. R

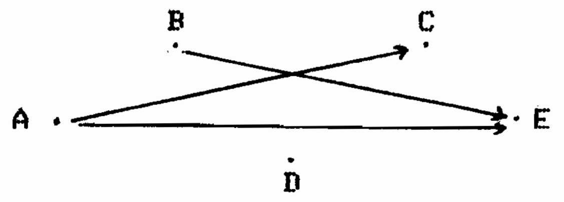


Step 2. R^2



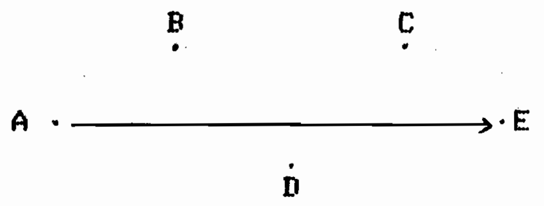
Here the chord AC represents the path ABC in R, BC represents BDC.

Step 3. R^3



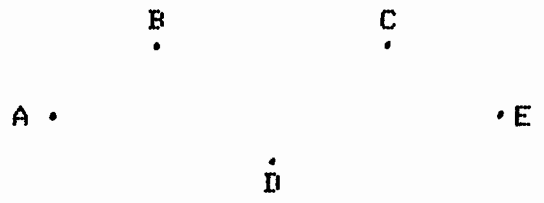
AE represents ABCE and AC represents ABDC in R

Step 4. R^4



Here AE represents ABDCF in R

Step 5. R^5



Empty because no paths of length 5 in R

These graphs can be represented by relations. For example,

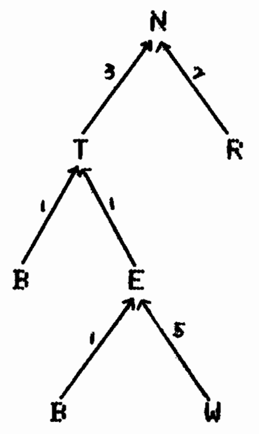
R (I	J)	RN (J	K)	RN (J	K)	RN (J	K)	RN (J	K)
A	B	A	C	A	E	A	E		
A	E	A	D	A	C				
B	C	B	E	B	E				
B	D	B	C						
D	C	D	E						
C	E								
(a)		(b)		(c)		(d)		(e)	

The procedure for the above example is written as:

```
relation R on (I,J); RN on (J,K);
RN:= R;
while (¬ empty RN) do begin
      RN:= project (R[J] iJoin RN[J])[I,K];
end
```

It starts with R (a) and gives in sequence RN (b), RN (c), RN (d), terminating at the empty relation RN (e).

A second illustration is a network, used to solve the bill of materials problem. A list of assemblies, subassemblies, parts and quantity in a manufactured product can naturally assume a tree structure (or a graph or a network).



We can represent the tree structure by a relation,

BOM (ASSEMBLY#	SUBASSY#	QTY)
N	T	3
N	R	2
T	E	1
T	B	1
E	B	1
E	W	5

associated with a raw materials table (which records raw materials).

RM (PART#)
R
B
W

We are able to calculate the quantity of raw materials for each assembly (where assemblies consist of many parts and subassemblies). First, we use natural join to find the quantity of raw materials for each assembly, then project on the RQTY (the required raw material quantity: QTY * RQ). After that, we use equivalence reduction to add up the raw materials quantity for each assembly. The loop will be terminated when no linkage items can be created. The procedure is as follow,

```
relation BOM on (ASSEMBLY#,SUBASSY#,QTY);
         RM on (PART#);
         TEMP on (SUBASSY#,PART#,Q);
         RQ on (ASSEMBLY#,PART#,TOTQTY);
let RQTY be QTY*Q;
let TOTQTY be equiv + of RQTY by ASSEMBLY#,PART#;
TEMP:= project (BOM[SUBASSY#] iJoin RM[PART#])
             [ASSEMBLY#,PART#,Q];
while (¬ empty TEMP) do begin
      RQ:= project (RQ[ASSEMBLY,PART#,RQTY] uJoin
                   TEMP[SUBASSY#,PART#,Q])
                   [ASSEMBLY#,PART#,TOTQTY];
      TEMP:= project (BOM[SUBASSY#] iJoin TEMP[SUBASSY#])
                     [ASSEMBLY#,PART#,RQTY];
end
```

The required raw materials quantity for each assembly are stored in relation RQ:

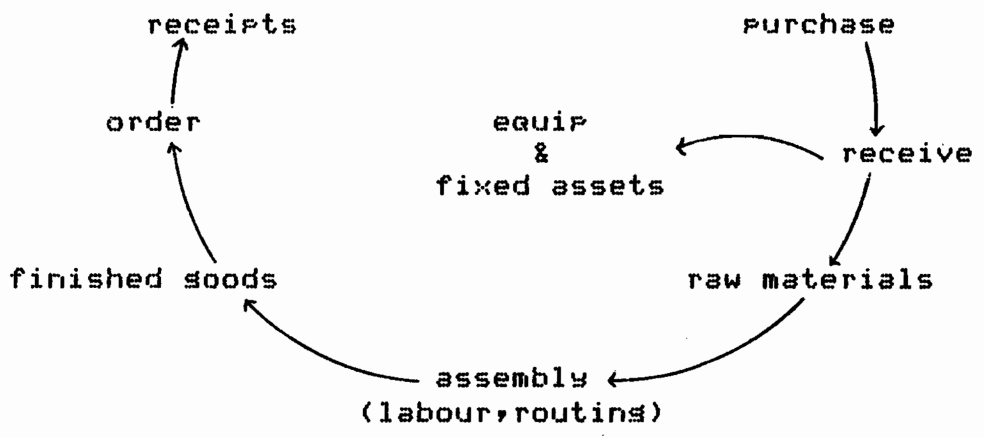
RQ (ASSEMBLY#	PART#	TOTQTY)
T	B	2
N	R	2
E	W	5
E	B	1
N	B	6
T	W	5
N	W	15

For instance, to make a NAND gate (N) we need two resistors, six bases, and fifteen wires.

The relational inner product is evidently a valuable construct. We use it further in a bill-of-materials context in chapter C3.

C3. Profit Analysis

The information model which we used to represent our business is as follows,



After we purchase raw materials or assets, we receive them. Raw materials are stocked until they are assembled into finished goods. These are ordered by customers and then paid for. From this information, a sales analysis can be performed.

Our sales analysis consists of two parts. In part one, we deal with cost and profit for each item produced. In part two, we use this to do the profit analysis by each customer and salesman. We need two parts because the cost of finished goods does not change very often. This means we execute the cost program only if there is a cost variation. The profit analysis we have to do routinely.

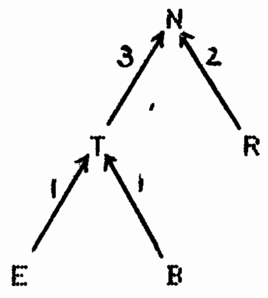
C3.1 Profit by part

When we convert raw materials into finished goods, the finished goods costs reflect the costs of raw materials, direct labour, and routing. These costs associated with product are referred to as product cost.

The costs of raw materials are recorded as the unit costs at the time of purchasing. The information of labour time reports and equipment-use time reports are also used in cost evaluation. They serve to accumulate raw material costs, labour costs and routing costs which contribute to the product cost.

With all the costs of each subassembly and the bill of materials (which identifies the finished goods by subassembly and component number), we are able to calculate the finished goods cost.

The graph shows that the NAND gate is made of three



transistors and two resistors. The transistor is made of one emitter and one base. The resistor, the emitter, and the base are all raw materials.

For raw materials, only material costs are considered, because they are the primitive materials which we use as such during the manufacture. When the raw materials are converted into finished goods, besides the material costs, labour costs and equipment costs are added to the total cost of the finished goods.

For example, each emitter and base cost five and seven dollars respectively. To assemble the transistor, nine dollars of labour cost and four dollars of equipment cost are required. Therefore, the final cost of the transistor is twenty-five dollars. The same principle is applicable to the calculation of the cost of the NAND gate.

The profit by part can be calculated simply by applying the known equation, profit = price - cost.

C3.2 Customer and sales profit analysis

A sales analysis serves to analyze the contribution to profit of company sales personnel and customer. It can be determined, for example, that a customer who does considerable business with a company does not necessary contribute much of the overall company profit, because of low gross margins.

When the finished goods are shipped to the customer, a sales transaction is recorded on a form called "shipment report". This report provides information for the sales analysis. Since some items sold may be returned by the customer, it is essential to make an occasional adjustment of the report. In this way a more accurate date of sale can be obtained. The returned goods are recorded in the credit memo.

Combining the adjusted shipment report with the finished goods profit record, we will be able to produce the dollar profit analysis report by each customer and salesman.

For example, in the customer profit report:

(CUST#	CHRONOL	QTY	$CSALE	$CPRFT)
8	790430	100	800	300
10	790430	6	150	39
17	790430	20	3,000	432
5	790430	80	2,800	800

The profit from customer 5 is almost double than that of customer 17, although customer 17 bought more goods in cash value. This makes customer 5 a better customer. It is possible to sell large quantities of merchandise to customers and yet not realize much profit (for example, customer 8).

C4. The Application of Aldat to Profit Analysis

We will describe our sales analysis program with examples using Aldat.

C4.1 Routing cost

From the routing work time report, we first find the costs of equipment and then calculate the routing cost. The cost of equipment per hour is derived from the relation EQUIP, which contains only the information total cost, salvage, life. The formula is the following:

$$\frac{\text{equipment cost} - \text{salvage}}{\text{life} * \text{work weeks per year} * \text{work hours per week}}$$

we can write,

```
let ECOST be (EQCOST - SALVAGE) / (LIFE * 50 * 40);
EQUIPCOST:+ project EQUIP [EQUIP#,ECOST];
```

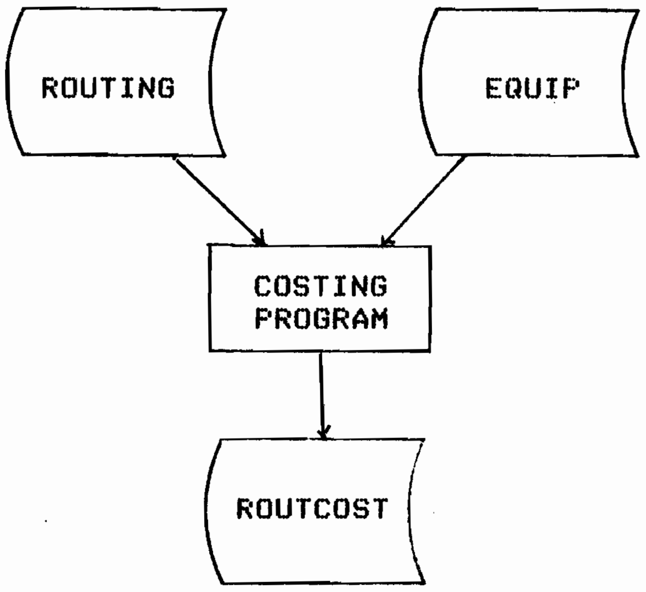
Let us try some simple data:

EQUIP (EQUIP#	CHRONOL	EQCOST	SALVAGE	LIFE)
1	730101	20,500	500	10
2	731201	18,500	1,500	5
3	740810	24,400	400	8
4	760101	29,000	1,000	7
5	760601	25,000	1,000	10

gives EQUIPCOST (EQUIP# ECOST)

EQUIP#	ECOST
1	1.0
2	1.7
3	1.5
4	2.0
5	1.2

Since we now know the hourly cost of each piece of equipment, we can do a natural join with the relation ROUTING, which contains the record of equipment working time. Then we derive the routing cost.



```
let RCOST be RTIME * ECOST;

ROUTCOST:+ project (ROUTING[EQUIP#] iJoin EQUIPCOST
                     [EQUIP#])[OP#,ASSEMBLY#,RCOST];
```

The relation ROUTING contains the following:

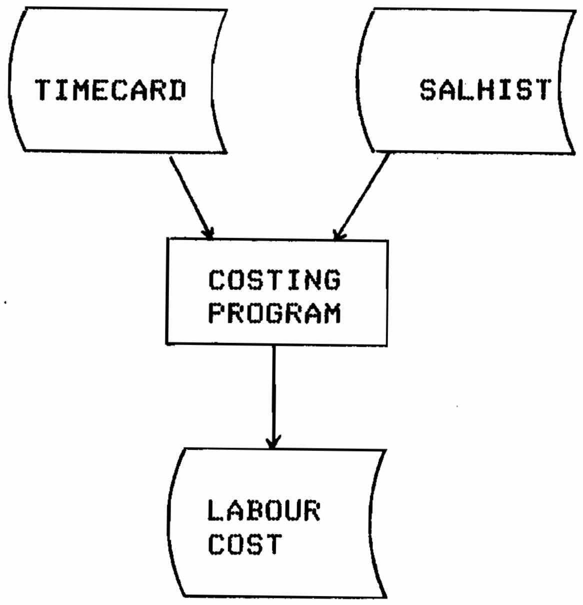
ROUTING (OP#	ASSEMBLY#	EQUIP#	CHRONOL	RTIME)
1	T	4	790215	2
2	N	5	790215	2
3	N	1	790215	1

After the operation, the result of ROUTCOST contains:

ROUTCOST (OP#	ASSEMBLY#	RCOST)
1	T	4.0
2	N	2.4
3	N	1.0

C4.2 Labour cost

From employee time cards, which are source documents recording the operation number and use of equipment and hours worked, we can simply get the cost of labour by a natural join with the salary history relation.



Since the salary history file contains all records of employees salaries, we have to select the most recent salary for each employee. In Aldat, we write

<u>let</u> SCHRON <u>be</u> CHRONOL = <u>equiv</u> <u>max</u> <u>of</u> CHRONOL <u>by</u> EMP#;

and then do a selection.

For example, we have TIMECARD and SALHIST, two relations which are involved in the labour cost.

TIMECARD (EMP#	CHRONOL	OP#	EQUIP#	HOURS	OVERTIME)
1	790319	1	4	1	
2	790319	1	4	1	
3	790319	2	5	2	
4	790319	2	5	2	
5	790319	2	5	1	1

SALHIST (EMP#	CHRONOL	SALARY	OVERSALARY)
1	740507	3	5
2	750101	4	6
3	750101	3	5
4	760101	3	5
3	771001	4	6
4	771201	5	7
5	781001	3	5
6	781101	2	3
1	790101	5	7

We can write

```
        let SCHRON be CHRONOL= equiv max of CHRONOL by EMP#;
        let EMPCOST be HOURS*SALARY + OVERTIME*OVERSALARY;
by LABOUR:+ project (TIMECARD[EMP#] iJoin (select SALHIST
                              [SCHRON])[EMP#])[EMP#,OP#,EQUIP#,EMPCOST];
```

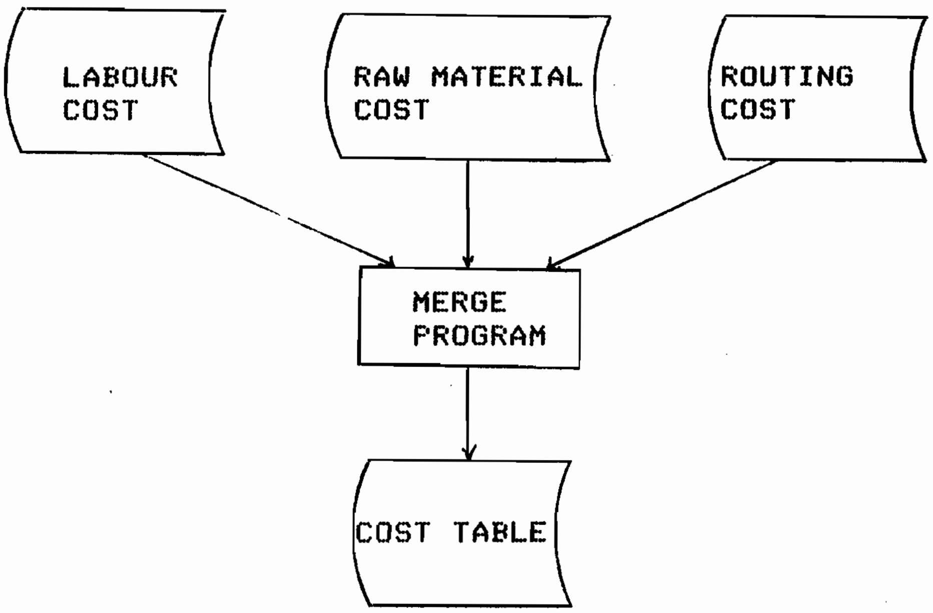
It would give us the cost of labour:

LABOUR

(EMP#	OP#	EQUIP#	EMPCOST)
1	1	4	5
2	1	4	4
3	2	4	8
4	2	5	10
5	2	5	8

C4.3 Finished goods cost

For the finished goods cost, besides the labour and routing costs, we need the cost of raw materials. To put these files together, we use union join, in case some costs are zero.



First of all, we merge the two relations LABOUR and ROUTCOST together. Since we can have more than two employees working on one operation, we must add up all employees' contribution to each operation. We use equivalence reduction in domain algebra:

<u>let</u> LABCOST <u>be</u> <u>equiv</u> + <u>of</u> EMPCOST <u>by</u> OP#;

by TEMP1:+ <u>project</u> (ROUTCOST[OP#] <u>uJoin</u> LABOUR[OP#])
[OP#,ASSEMBLY#,LABCOST,RCOST];

gives,

TEMP1 (OP#	ASSEMBLY#	LABCOST	RCOST)
1	T	9.0	4.0
2	N	26.0	2.4
3	N		1.0

The raw materials cost RMCOST as given:

RMCOST (RMPART#	CHRONOL	$RMCOST)
R	750608	10.0
E	760503	5.0
B	770101	7.0
R	780609	12.0

Since the cost of raw materials stored in RMCOST is a history relation, we have to find the latest cost for each part first, then union Join with relation TEMP1.

<u>let</u> RCHRON <u>be</u> CHRONOL = <u>equiv</u> <u>max</u> <u>of</u> CHRONOL <u>by</u> RMPART#;

<u>let</u> $LABCOST <u>be</u> <u>equiv</u> + <u>of</u> LABCOST <u>by</u> ASSEMBLY#;

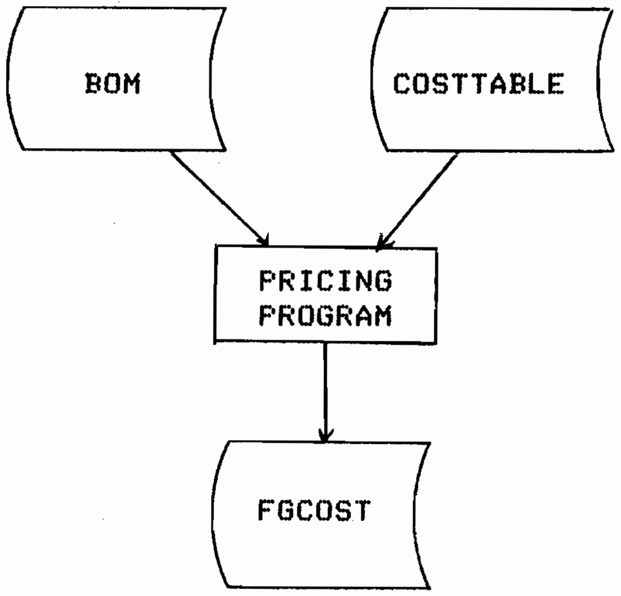
<u>let</u> $ROUTCOST <u>be</u> <u>equiv</u> + <u>of</u> RCOST <u>by</u> ASSEMBLY#;

```
COSTTABLE:+ project ((select RMCOST[RCHRON])[RMPART#]
                         uJoin TEMP1[ ASSEMBLY# ]);
```

It gives the result

COSTTABLE (RMPART#	$RMCOST	$ROUTCOST	$LABCOST)
E	5.0		
B	7.0		
R	12.0		
T		4.0	9.0
N		3.4	26.0

Now, we have a table which contains all the information of costs for each subassembly, and by natural join with the relation BOM, we can produce the finished goods cost. The bill of materials file is used to break down finished goods requirements into subassembly and part requirements, as discussed in chapter 2.5.



Let us illustrate the relation BOM as follows,

BOM (ASSEMBLY#	SUBASSY#	QTY)
N	R	2
N	T	3
T	E	1
T	B	1

We can write

```
        let RMPRICE be equiv + of (QTY*$RMCOST) by ASSEMBLY#;
        let ROTPRICE be equiv + of (QTY*$ROUTCOST) by ASSEMBLY#;
        let LABPRICE be equiv + of (QTY*$LABCOST) by ASSEMBLY#;
        T1:= project (BOM[SUBASSY#] iJoin COSTTABLE[RMPART#])
                     [ASSEMBLY#,RMPRICE,ROTPRICE,LABPRICE];
```

which would give us:

T1 (ASSEMBLY#	RMPRICE	ROTPRICE	LABPRICE)
N	24.0	12.0	27.0
T	12.0		

we are not finished yet, since NAND gate (N) is made of three transistors (T). Assembly T is in turn a subassembly even though it is derived from other subassemblies. This problem can be solved by the following procedure,

```
TEMPCOST:+ COSTTABLE;
while (¬ empty T1) do begin
      TEMPCOST:+ project (TEMPCOST[RMPART#,$RMCOST,ROUTCOST,
                  $LABCOST] uJoin T1[ASSEMBLY#,RMPRICE,
                  ROTPRICE,LABPRICE])
                  [RMPART#,$RMCOST,$ROUTCOST,$LABCOST];
      T1:= project (BOM[SUBASSY#] iJoin T1[ASSEMBLY#])
                    [ASSEMBLY#,RMPRICE,ROTPRICE,LABPRICE];
end
```

The results of the first loop are the following:

```
TEMPCOST (RMPART#  $RMCOST  $ROUTCOST  $LABCOST)
             E         5.0
             B         7.0
             R        12.0
             T                  4.0          9.0
             N                  3.4         26.0
             N        24.0     12.0         27.0
             T        12.0
T1 (ASSEMBLY#  RMPRICE  ROTPRICE  LABPRICE)
        N        36.0
```

The assemblies of T1 may in turn be subassemblies. Thus the operation is repeated until T1 is empty.

In our case, we stopped after the second loop, and the final cost of the finished goods is obtained in relation TEMPCOST:

TEMPCOST (RMPART#	$RMCOST	$ROUTCOST	$LABCOST)
E	5.0		
B	7.0		
R	12.0		
T		4.0	9.0
N		3.4	26.0
N	24.0	12.0	27.0
T	12.0		
N	36.0		

From the TEMPCOST, we add up the costs by each part:

```
        let FGPART# be RMPART#;
        let RMTOT be equiv + of $RMCOST by RMPART#;
        let ROTOT be equiv + of $ROUTCOST by RMPART#;
        let LABTOT be equiv + of $LABTOT by RMPART#;
        let $FGCOST be RMTOT + ROTOT + LABTOT;
        let PERIOD be today;
        FGCOST:+ project TEMPCOST [FGPART#,PERIOD,RMTOT,ROTOT,
                                   LABTOT,$FGCOST];
```

FGCOST (FGPART#	PERIOD	RMTOT	ROTOT	LABTOT	$FGCOST)
E	790320	5.0			5.0
B	790320	7.0			7.0
R	790320	12.0			12.0
T	790320	12.0	4.0	9.0	25.0
N	790320	60.0	15.4	53.0	128.4

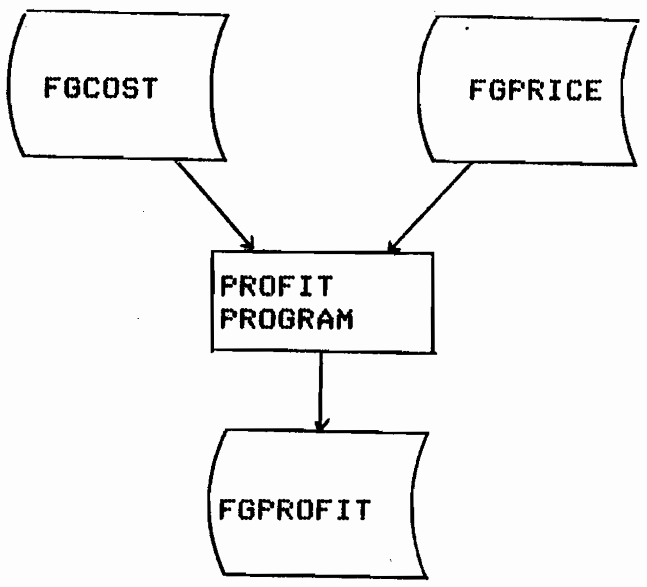
This equivalence reduction could be done within the while loop, which gives shorter intermediate results and a final TEMPCOST such as

TEMPCOST (RMPART#	RMTOT	ROTOT	LABTOT)
E	5.0		
B	7.0		
R	12.0		
T	12.0	4.0	9.0
N	60.0	15.4	53.0

The relation FGPRICE contains:

FGPRICE (FGPART#	CHRONOL	$FGPRICE)
N	790218	125.0
T	790218	30.0
N	790320	150.0
T	790320	35.0
R	790320	15.0
E	790320	8.0
B	790320	10.0

To obtain the profit by each part, we can simply use the formula: $FGPROFIT = $FGPRICE - $FGCOST.



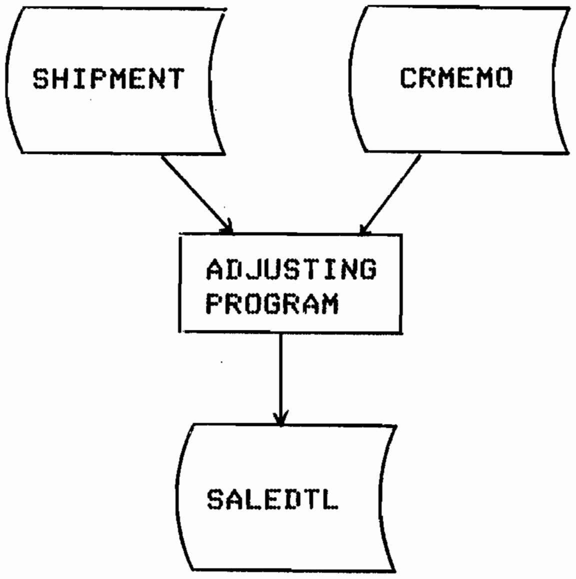
```
let PERIOD be today;

let $FGPROFIT be $FGPRICE - $FGCOST;

let RECENT be CHRONOL = equiv max of CHRONOL by FGPART#;

FGPROFIT:+ project ((select FGPRICE[RECENT])[FGPART#]

           iJoin (select FGCOST[RECENT])[FGPART#])

           [FGPART#,PERIOD,$FGPRICE,RMTOT,ROTOT,

            LABTOT,$FGPROFIT];
```

The result would be the following:

```
FGPROFIT (FGPART# PERIOD $FGPRICE $FGCOST RMTOT ROTOT LABTOT $FGPROFIT)

              N     790320   150.0    128.4   60.0  15.4   53.0      21.6

              T     790320    35.0     25.0   12.0   4.0    9.0      10.0

              R     790320    15.0     12.0   12.0                    3.0

              E     790320     8.0      5.0    5.0                    3.0

              B     790320    10.0      7.0    7.0                    3.0
```

C4.4 Sales detail

The sales detail file which contains the current monthly net sales is derived from the shipments and the credit memo details file.



For shipment details, we merge SHIPMENT and SHIPLINE together, and since they are history files, the data for the current period must be selected.

SHIPMENT (CUST# ORD# CHRONOL B_LAD# SALES#)

CUST#	ORD#	CHRONOL	B_LAD#	SALES#
4	99	790212	1	5
5	101	790301	2	4
17	131	790309	3	1
9	143	790321	4	6
8	105	790419	5	9
10	131	790420	6	8
17	161	790428	7	7
5	101	790430	8	7

SHIPLINE (B-LAD#	FGPART#	SHQTY)
1	N	20
2	T	5
2	R	5
3	N	20
3	T	10
4	N	30
4	T	6
4	R	6
5	E	100
6	T	4
6	R	4
7	N	20
8	T	80

```
let SHIPCHRON be CHRONOL>lastperiod;
SHIPDTL:+ project ((select SHIPMENT[SHIPCHRON])[B_LAD#]
         iJoin SHIPLINE[B_LAD#])
         [CUST#,ORD#,CHRONOL,B-LAD#,SALES#,FGPART#,SHQTY#];
```

The shipment details now contains:

SHIPDTL (CUST#	ORD#	CHRONOL	B-LAD#	SALES#	FGPART#	SHQTY)
8	105	790419	5	9	E	100
10	131	790420	6	8	T	4
10	131	790420	6	8	R	4
17	161	790428	7	7	N	20
5	101	790430	8	7	T	80

Similary, we can derive credit memo details from CRMEMO and CRLINE:

CRMEMO (MEMO#	CHRONOL	CUST#	ORD#	INV#)
1	790301	4	99	90
2	790319	4	99	90
3	790401	5	101	101
4	790419	9	142	103
5	790425	10	131	104

CRLINE (MEMO#	FGPART#	CRQTY)
1	N	1
2	N	4
3	N	3
3	T	1
4	T	2
5	T	1
5	R	1

```
let CRCHRON be CHRONOL>lastperiod;
CRDTL:+ project((select CRMEMO[CRCHRON])[MEMO#] iJoin
          CRLINE[MEMO#])[MEMO#,CUST#,ORD#,FGPART#,CRQTY];
```

CRDTL (MEMO#	CUST#	ORD#	FGPART#	CRQTY)
3	5	101	N	3
3	5	101	T	1
4	9	142	T	2
5	10	131	T	1
5	10	131	R	1

The shipment details can now be adjusted according to the credit memo details, by using the adjust formula:

net sales quantity = shipped quantity - returned quantity.

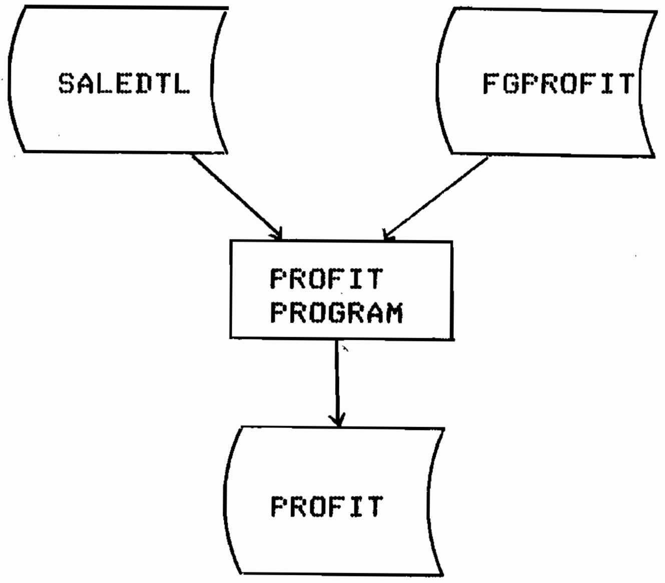
```
    let NQTY be SHIPQTY - CRQTY;
    SALEDTL:+ project (SHIPDTL[CUST#,ORD#,FGPART#] uJoin
                            CRDTL[CUST#,ORD#,FGPART#])
                       [CUST#,ORD#,FGPART#,SHQTY,CRQTY,NQTY,SALES#];
```

SALEDTL	(CUST#	ORD#	FGPART#	SHQTY	CRQTY	NQTY	SALES#)
	5	101	N		3	-3	
	5	101	T		1	-1	
	9	142	T		2	-2	
	8	105	E	10		10	9
	10	131	T	4	1	3	8
	10	131	R	4	1	3	8
	17	161	N	20		20	7
	5	101	T	80		80	7

If the net sales quantity is negative, then the unmatched credit memo details must be not belong to the current month. This is because the credit memo details may contain returned items which were shipped in the last period.

C4.5 Profit by customer and salesman

The profit analysis by the current period can be produced from SALEDTL and FGPROFIT.

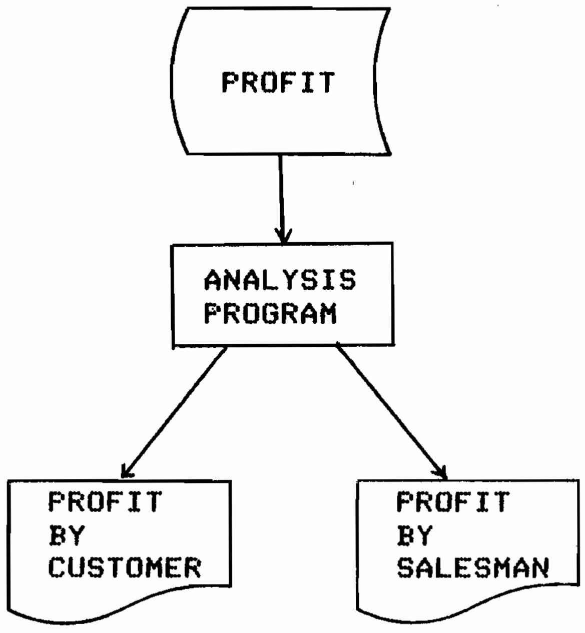


The net sales quantities which are negative in SALEDTL are items sold during the last period. For the current period's sales, we have to select the net sales quantity which is greater than zero.

```
let PFCHRON be CHRONOL = equiv max of CHRONOL by FGPART#;
let CURSALE be NQTY > 0;
let $SALE be $FGPRICE * QTY;
let $PROFIT be $FGPROFIT * QTY;
PROFIT:+ project (select SALEDTL[CURSALE])[FGPART#]
         iJoin (select  FGPROFIT[PFCHRON])[FGPART#])
         [CUST#,FGPART#,NQTY,$SALE,$PROFIT,SALES#];
```

PROFIT (CUST#	FGPART#	NQTY	$SALE	$PROFIT	SALES#)
8	E	100	800	300	9
10	T	3	105	30	8
10	R	3	45	9	8
17	N	20	3,000	432	7
5	T	80	2,800	800	7

The customer profit report and salesman profit report can be produced from PROFIT.



<u>let</u> PERIOD <u>be</u> today;

<u>let</u> $SSALE <u>be</u> <u>equiv</u> + <u>of</u> $SALE <u>by</u> CUST#;

<u>let</u> $CPRFT <u>be</u> <u>equiv</u> + <u>of</u> $PROFIT <u>by</u> CUST#;

CUSTRP:+ <u>project</u> PROFIT [CUST#,PERIOD,NQTY,$CSALE,$CPRFT];

The customer profit report contains:

CUSTRP (CUST#	PERIOD	NQTY	$CSALE	$CPRFT)
8	790430	100	800	300
10	790430	6	150	39
17	790430	20	3,000	432
5	790430	80	2,800	800

<u>let</u> $SSALE <u>be</u> <u>equiv</u> + <u>of</u> $SALE <u>by</u> SALES#;

<u>let</u> $SPRFT <u>be</u> <u>equiv</u> + <u>of</u> $PROFIT <u>by</u> SALES#;

SALERPT:+ <u>project</u> PROFIT [SALES#,PERIOOD,$SSALE,$SPRFT];

The report of salesman profit is as follows,

SALERPT (SALES#	PERIOD	$SSALE	$SPRFT)
9	790430	80	30
8	790430	150	39
7	790430	5,800	1,232

We can also analyze the contribution of parts to the profit of the company.

<u>let</u> TOTQTY <u>be</u> <u>equiv</u> + <u>of</u> NQTY <u>by</u> FGPART#;

<u>let</u> $PARTPFT <u>be</u> <u>equiv</u> + <u>of</u> $PROFIT <u>by</u> FGPART#;

<u>let</u> $PARTSALE <u>be</u> <u>equiv</u> + <u>of</u> $SALE <u>by</u> FGPART#;

PARTRPT:+ <u>project</u> PROFIT [FGPART#,PERIOD,TOTQTY,
$PARTSALE,$PARTPFT];

```
PARTRPT (FGPART#  PERIOD  TOTQTY  $PARTSALE  $PARTPFT)
            E     780430   100        800        300
            T     790430    83      2,905        830
            R     790430     3         45          9
            N     790430    20      3,000        432
```

C4.6 Corrections to previous sales detail reports

Some returned items from last period's sales will show up on this month's credit memo details. Therefore, a report of corrections is required.

```
let $CRPROFIT be $FGPROFIT * CRQTY;
let $CR be $FGPRICE * CRQTY;
CORRECTION:+ project ((select SALEDTL[CURSALE])[FGPART#]
                       iJoin FGPROFIT[FGPART#])
                      [CUST#,ORD#,PERIOD,FGPART#,CRQTY,$CR,
                       $CRPROFIT];
```

```
CORRECTION (CUST# ORD# PERIOD FGPART# CRQTY $CR $CRPROFIT)
              5    101 790430    N       3   450    64.8
              5    101 790430    T       1    35    10.0
              9    142 790430    T       2    70    20.0
```

From this correction report, we can go back to the appropriate period sales detail report (see page 38) and check manually.

K3. The Accounting Applications

K3.1 Accounting network

We present the simplified accounting network of our firm from a funds flow point of view:

$$\begin{tikzcd}
 & & C \arrow[drr, bend left] \arrow[dd, dashed] & & \\
R \arrow[urr, bend left] & & & & P \arrow[ddll] \arrow[ddl] \arrow[ddd, bend left] \\
 & & T \arrow[d] & & \\
 & & E \arrow[uull] \arrow[dll, dashed] \arrow[drr, dashed] \arrow[dd] & F \arrow[l] & \\
G \arrow[uuu, bend left] & & & & M \arrow[dll, bend left] \\
 & & A \arrow[ull, bend left] & &
\end{tikzcd}$$

DR <-- CR

The accounts in the above network are, together with their beginning balances as example:

	Assets			Liabilities	
C	Cash	10000	P	Accounts payable	800
M	Raw materials	3500	T	Taxes payable	125
A	Goods in assembly	150			
G	Finished goods	200		Equity	
R	Accounts receivable	0			
F	Fixed assets	100	E	Stockholders' account	13025
		13950			13950

The network represents a set of transactions or summary of transactions for a period. The transactions are marked by arrows from the account credited to the account debited. The details of transactions are as follows:

accounts debited <--	credited	transactions
C	R	Collect accounts receivable
P	C	Pay accounts payable
M	P	Purchase of raw materials
A	M	Raw material costs
G	A	Assembling costs
R	G	Cost of goods sold
R	E	Gross profit on sales
A	E	Variable cost charged to goods in assembly (labour cost, routing cost)
F	P	Purchase of fixed assets
E	F	Depreciation of fixed assets
E	P	Manufacturing & operating expenses
E	T	Taxes - payroll

The arrows marked with broken lines are possible existing transactions such as:

G <-- E	Finished goods inventory costs
M <-- E	Raw material inventory costs
T <-- C	Cash payment of taxes at end of year

but will be ignored in further discussions to simplify the event. Some transactions are directly involved in the daily running of the firm (eg. cash receipts and payments), but some are to be extracted from the daily operation. We can keep the record of transactions separately depending on their sources of operation.

The daily shipment information contains the quantity of goods sold. If we know the unit cost and profit per item, we can extract the cost of goods sold (R <-- G) and the gross profit (R <-- E). In our approach, we also extract, further, the quantities G <-- A, A <-- E and A <-- M.

From the employees' salaries and tax deduction calculation at each pay period, we have the information of total accrued income taxes (E <-- T) and net salaries (E <-- P and P <-- C).

The rest of the transactions shown in the network diagram are recorded in separate relations as they arise each day.

We will summarize all transactions, which were kept separately, at the end of the accounting period.

K3.2 Spread sheet and financial statements

We will display the summary of transactions in an accounting period in the spread sheet with examples on the next page.

The amount appearing at the intersection of rows and columns represents the total of the transactions debited to the account named by the row head and at the same time credited to the account named by the column head. The row sum represents total debit, Δ^{+}, in the row head account and the column sum represents total credit , Δ^{-} , in the column head account. We summarize the changes separately for Assets and for Liabilities and Equities according to conventional accounting practice. These changes in Balance Sheet accounts (i.e. current general ledger balance, which summarizes the total transactions for the period) can be used as a rough approach to funds flow analysis.

[Spread sheet]

DR \ CR		ASSETS						LIABILITIES		EQUITY	
		C	M	A	G	R	F	P	T	E	Δ^+
A	C					1122.00					1122.00
	M							580.25			580.25
	A		2222.00							2404.00	4625.00
	G			4625.00							4625.00
	R				4625.00					960.00	5585.00
	F							301.50			301.50
L	P	3837.50									3837.50
	T										
E	E						150.42	2277.50	472.50		2900.42
	Δ^-	3837.50	2222.00	4625.00	4625.00	1122.00	150.42	3159.25	472.50	3363.00	

Changes in accounts

$(\Delta^{+} - \Delta^{-})$ Assets		$(\Delta^{-} - \Delta^{+})$ Liabilities		Source of funds		Use of funds	
C	-2715.50	P	-678.25	C	2715.50	P	678.25
M	-1641.75	T	472.50	M	1641.75	R	4463.00
A	0			A	0	F	151.08
G	0	Equity		G	0		
R	4463.00			T	472.50		
F	151.08	E	462.58	E	462.58		
	256.83		256.83		5292.33		5292.33

The Source and use of funds statement can be derived directly from the funds flow statement as shown above: decreases in assets and increases in liabilities or equity are sources of funds, while increases in assets and decreases in liabilities or equity are uses of funds.

The sum of flow of funds (summary of total transactions for a period) with the starting balance sheet gives the final Balance Sheet at the end of this accounting period.

Assets		Liabilities	
C	7284.50	P	121.75
M	1858.25	T	597.50
A	150.00		
G	200.00	Equity	
R	4463.00		
F	251.08	E	13487.58
	14206.83		14206.83

Note that the result balances. To achive this traditional balance (rather than sum zero) is the reason for treating asset account differently from the liabilities and equity under debit and credit.

The spread sheet gives the Income statement for the period directly:

Sales (RG + RE)		5585.00
Cost of goods sold (RG)		4625.00
Gross profit (RE)		960.00
Manufacturing cost and operating expenses (EP)	2277.50	
Depreciation (EF)	150.42	
Variable cost charged to goods in assembly (AE)	-2403.00	
Unabsorbed manufacturing cost and operating expenses		24.92
Net income before taxes		935.08
Accruals of income taxes (ET)		472.50
Net income after taxes (E)		462.58

Note that the final amount is just the changes in the equity account. Also note that most of the lines in the income statement are entries in the spread sheet or sums of entries: these entries are not in themselves considered accounts in our discussion, although in other discussions they are. For example, sales, cost of goods sold, depreciation, etc. are often called Revenue or Expense accounts.

The income statement can be rewritten in a more systematic form. Note that except for RG (cost of goods sold) all expenses and revenues correspond to transactions directly linked with the equity account and RG appears only because it is added to RE (gross profit) to give the traditional revenue of sales.

Revenues (column E)	
Profit (RE)	960.00
Variable cost (AE)	2403.00
Total revenue	3363.00

Expenses (row E)	
Direct	
Manuf. & oper. exp. (EP)	2277.50
Indirect	
Taxes (ET)	472.50
Depreciation (EF)	150.42
Total expenses	2900.42

Net profit after tax 462.58

Figures in the income statement can be given in alternate form as a percent of total revenues with those from the previous period or the same period a year ago.

If we summarize the derivation of financial statements from the spread sheet, we can express the process in a tree structure as follow:

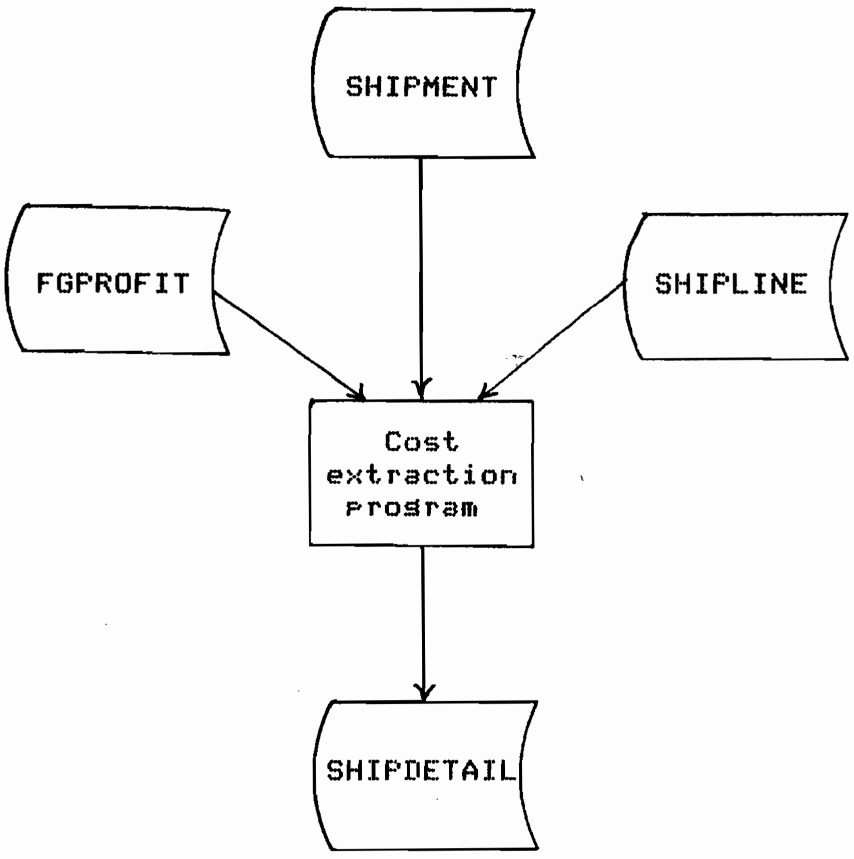
$$\begin{tikzcd}
 & \text{Spread sheet} \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{Starting balance} \arrow[d] & \text{Funds flow statement} \arrow[dl] \arrow[d] \arrow[dr] & \text{Revenue} \arrow[d] & \text{Expense} \arrow[dl] \\
\text{Balance Sheet} & \text{Source and use statement} & \text{Income statement} &
\end{tikzcd}$$

K4. The Application of Aldat to Accounting

K4.1 Detail file preparation

Cost extraction.

The daily shipment information contains the quantity of goods sold. We will extract various cost and profit for shipped goods using a file FGPROFIT which contains unit profits and costs of each finished part. Then the extracted detail will be kept as a permanent file, SHIPDETAIL, which contains profit and cost per parts shipped. This file will be summarized at the end of accounting period for general ledger by accounting period.



We will extract the information for accounts M, A and G:

```
 << cost Extraction Program >>

 let LASTIME be last_time;  << the last operation date >>
 let PERIOD be today;       << assume function today   >>
 let CURNT be equiv max of CHRONOL by FGPART#;
 let QTY be equiv + of SH_QTY by FGPART#;
 let QTYFGCOST   be QTY*$FGCOST;  << R <-- G and G <-- A >>
 let QTYRMTOT    be QTY*RMTOT;                 << A <-- M >>
 let QTYVARCOST  be QTY*(ROTOT + LABTOT );     << A <-- E >>
 let QTYFGPROFIT be QTY * $FGPROFIT;           << R <-- E >>

 SHIPDETAIL:+ project(((B_LAD# if CHRONOL>LASTIME in SHIPMENT)
                   [B_LAD#] iJoin SHIPLINE[B_LAD#])[FGPART#] iJoin
                     (if CRONOL=CURNT in FGPROFIT)[FGPART#]
     [PERIOD,FGPART#,QTY,QTYFGCOST,QTYRMTOT,QTYVARCOST,QTYFGPROFIT]
```

We will illustrate the last computation step by step with example:

SHIPMENT(CUST#	ORD#	CHRONOL	B_LAD#	SALES#)
4	99	790212	1	5
5	105	790301	2	4
17	131	790329	3	1
9	142	790329	4	6
8	105	790403	5	9
10	131	790403	6	8
17	161	790403	7	7
5	101	790403	8	4

SHIPLINE(B_LAD#	FGPART#	SH_QTY)
1	N	20
2	T	5
2	R	5
3	N	20
3	T	10
4	N	30
4	T	6
4	R	6
5	E	10
6	T	4
6	R	4
7	N	20
8	T	5

FGPROFIT(FGPART#	CHRONOL	$FGPRICE	$FGCOST	RMTOT	ROTOT	LABTOT	$FGPROFIT)
E	780901	.	.	.	.	.	.
B	780901	.	.	.	.	.	.
R	780901	.	.	.	.	.	.
E	790320	8	5	5	0	0	3
B	790320	10	7	7	0	0	3
R	790320	15	12	12	0	0	3
T	790320	35	25	12	4	9	10
N	790320	150	128.4	60	15.4	53	21.6

We elaborate on two of the subexpressions in the derivation of the SHIPDETAIL.

(a) B_LAD# if CHRONOL>LASTIME in SHIPMENT

This statement projects bill of lading number, B_LAD#, if chronology is greater than last time in relation SHIPMENT. Therefore if we assume last_time as 790402 then the result is:

B_LAD#
5
6
7
8

An intersection join of this with SHIPLINE[B_LAD#] would be:

B_LAD#	FGPART#	SH_QTY
5	E	10
6	T	4
6	R	4
7	N	20
8	T	5

(b) if CHRONOL=CURNT in FGPROFIT

This statement will select tuples in FGPROFIT, which chronology is maximum, then ijoin with above respect to FGPART#:

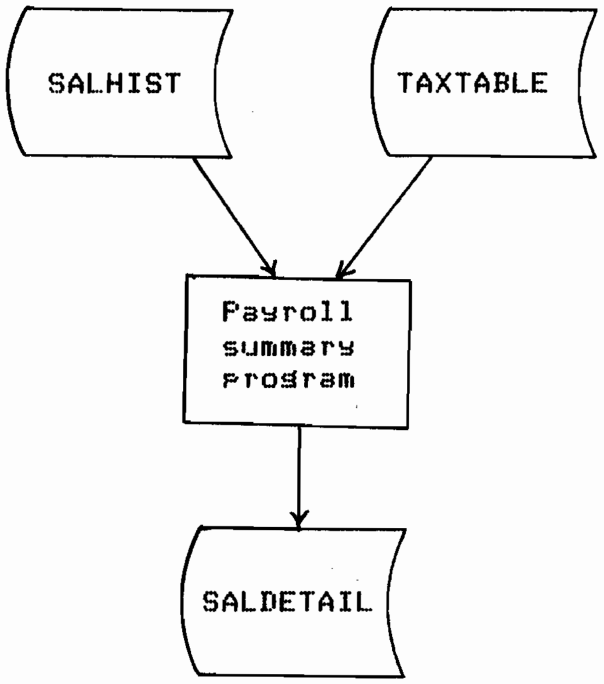
B_LAD#	FGPART#	SH_QTY	CHRONOL	$FGPRICE	$FGCOST	RMTOT	ROTOT	LABTOT	$FGPROF
5	E	10	790320	8.00	5.00	5.00	0.00	0.00	3.0
6	T	4	790320	35.00	25.00	12.00	4.00	9.00	10.0
6	R	4	790320	15.00	12.00	12.00	0.00	0.00	3.0
7	N	20	790320	150.00	128.40	60.00	15.40	53.00	21.6
8	T	5	790320	35.00	25.00	12.00	4.00	9.00	10.0

projected final result: (assume function, 790403, April 3,1979)

PERIOD	FGPART#	QTY	QTYFGCOST	QTYRMTOT	QTYVARCOST	QTYFGPROFIT
790403	E	10	50	50	0	30
790403	T	9	225	108	117	90
790403	R	4	48	48	0	12
790403	N	20	2568	1200	1368	432

Payroll Summary

We calculate tax deductions for each employee at each pay period. We use the salary history file, SALHIST, and the tax deduction table, TAXTABLE, for calculation and the result will be kept in SALDETAIL, which will be summarized at the end of each accounting period.



```
<< Payroll Summary Program >>

 let LATESTSAL be equiv max of CHRONOL by EMP#;

 let LATESTRATE be equiv max of CHRONOL by FLOOR;

 let TAX be BASE + (SALARY - FLOOR) * TAX% ;   << E <-- T >>

 let NET be SALARY - TAX;           << P <-- C and E <-- P >>

 let PAYDATE be payday;      << assume function on payday >>

 SALDETAIL:+ project(( if CHRONOL=LATESTSAL  in SALHIST )[ SALARY ]
                lojoin ( if CHRONOL=LATESTRATE in TAXTABLE )[ FLOOR ])
                       [ PAYDATE, EMP#, SALARY, TAX, NET ];
```

(assume function on April 15, 1979)

SALHIST(EMP#	CHRONOL	SALARY)	LATESTSAL
1	751119	250	770513
1	760601	350	770513
2	770301	400	770301
1	770513	500	770513
3	780801	400	780801

TAXTABLE(FLOOR	CHRONOL	BASE	TAX%)	LATESTRATE
350	740101	50.00	20	740101
400	740101	60.00	20	740101
450	740101	72.50	30	770101
500	740101	87.50	35	770101
400	770101	60.00	30	770101
450	770101	75.00	35	770101
500	770101	92.50	40	770101

(a) if CHRONL=LATESTSAL in SALHIST

will give:

EMP#	CHRONOL	SALARY	LATESTSAL
1	770513	500	770513
2	780601	475	780601
3	780801	400	780801

(b) if CHRONOL=LATESTRATE in TAXTABLE

will give:

FLOOR	CHRONOL	BASE	TAX%	LATESTRATE
350	740101	50.00	20	740101
400	770101	60.00	30	770101
450	770101	75.00	35	770101
500	770101	92.50	40	770101

low range join, loJoin, of above two sample (a) and (b) respect

to SALARY and FLOOR will be:

EMP#	...	SALARY	FLOOR	...	BASE	TAX%
1		500	500		92.50	40
2		475	450		75.00	35
3		400	400		60.00	30

and project with following domains will be the final result.

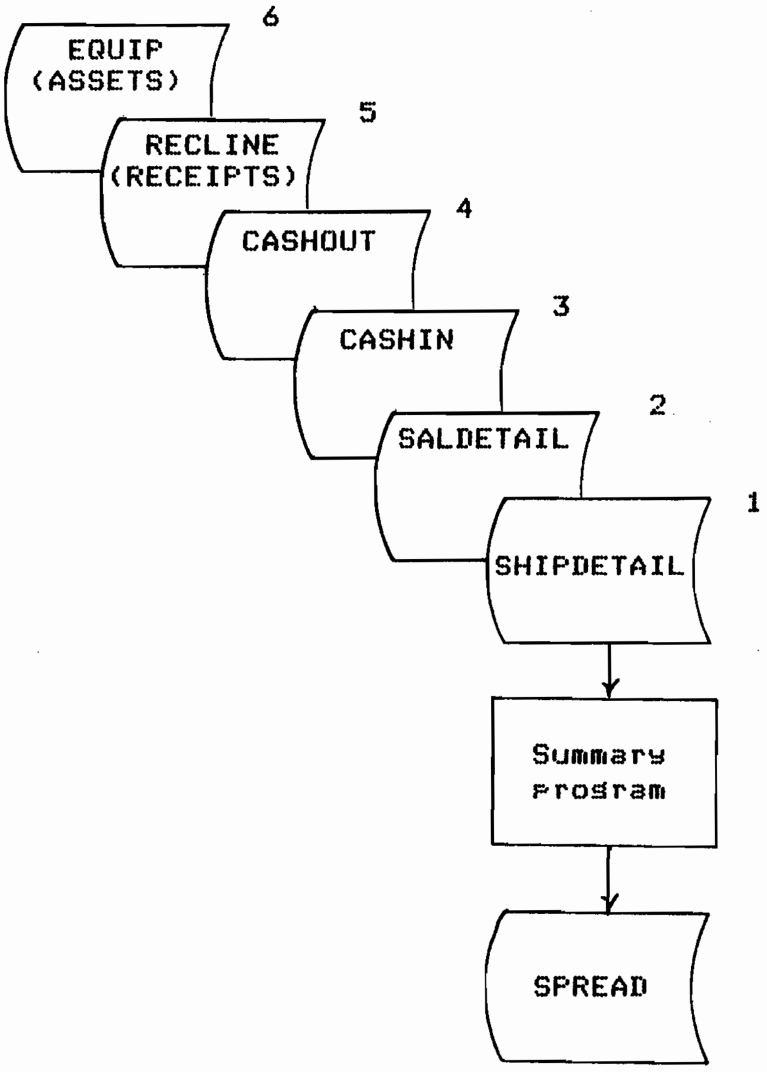
PAYDATE	EMP#	SALARY	TAX	NET
790415	1	500	92.50	407.50
790415	2	475	83.75	391.25
790415	3	400	60.00	340.00

(TAX of EMP#2 = 75 + (475 - 450) * 35/100 = 83.75)

K4.2 Accounting period summary

This is a conventional posting program. All transactions, which were kept separately during the period, will be summarized into the spread sheet, SPREAD.

We will introduce the summary program in sections to illustrate related Aldat statements at the same time. We collect each data into temporary relation SUMMARY first, then we store the summary of SUMMARY into permanent file SPREAD.



Summary Program - 1: SHIPDETAIL

We will illustrate with our example SHIPDETAIL:

```
(CHRONOL FGPART# QTY QTYFGCOST QTYRMTOT QTYVARCOST QTYFGPROFIT) RECENT
 790330    B       5     35        35         0          15          0
 790403    E      10     50        50         0          30          1
 790403    T       9    225       108       117          90          1
 790403    R       4     48        48         0          12          1
 790403    N      20   2568      1200      1368         432          1
 790420    T      18    450       216       234         180          1
 790430    N      10   1284       600       684         216          1
```

(Assume function at April 30,1979; lastday as March 31,1979)

```
<< Summary of SHIPDETAIL >>

 let LASTPERIOD be lastday; << lastday of last accountin period >>

 let TFGCOST   be red+ of QTYFGCOST;    << R <-- G & G <-- A >>

 let TRMTOT    be red+ of QTYRMTOT;     << A <-- M >>

 let TVARTOT   be red+ of QTYVARCOST;   << A <-- E >>

 let TFGPROFIT be red+ of QTYFGPROFIT; << R <-- E >>

 let ACCTR be 'R';    << account R >>

 let ACCTG be 'G';    << account G >>

 let ACCTA be 'A';    << account A >>

 let ACCTM be 'M';    << account M >>

 let ACCTE be 'E';    << account E >>

 let RECENT be CHRONOL > LASTPERIOD;

 CURNTDATA:= select SHIPDETAIL[ RECENT ];

 SUMMARY:+ project CURNTDATA [ ACCTR, ACCTG, TFGCOST ];
                                 R      G      4625

 SUMMARY:+ project CURNTDATA [ ACCTG, ACCTA, TFGCOST ];
                                 G      A      4625

 SUMMATY:+ project CURNTDATA [ ACCTA, ACCTM, TRMTOT ];
                                 A      M      2222

 SUMMARY:+ project CURNTDATA [ ACCTA, ACCTE, TVARTOT ];
                                 A      E      2403

 SUMMARY:+ project CURNTDATA [ ACCTR, ACCTE, TFGPROFIT ];
                                 R      E      960
```

Summary program - 2: SALDETAIL

The relation SALDETAIL contains detailed payment of salaries and tax deductions at each payday.

We will illustrate the step with example:

```
 SALDETAIL( PAYDATE   EMP#   SALARY  TAX     NET   )    CURNTPAY
            790331     3      400   60.00  340.00          0
            790415     1      500   92.50  407.50          1
            790415     2      475   83.75  391.25          1
            790415     3      400   60.00  340.00          1
            790430     1      500   92.50  407.50          1
            790430     2      475   83.75  391.25          1
            790430     3      400   60.00  340.00          1
```

```
<< Summary of SALDETAIL >>

 let TOTAX  be red+ of TAX;   << E <-- T >>

 let TOTNET be red+ of NET;   << P <-- C & E <-- P >>

 let ACCTC  be 'C';    << account C >>

 let ACCTP  be 'P';    << account P >>

 let ACCTT  be 'T';    << account T >>

 let CURNTPAY be PAYDATE > LASTPERIOD;

<< select all tuples with CURNTPAY is true >>

 CURNTDATA:= select SALDETAIL[ CURNTPAY ];

 SUMMARY:+ project CURNTDATA[ ACCTE, ACCTT, TOTAX  ];
                              E      T    472.50

 SUMMARY:+ project CURNTDATA[ ACCTP, ACCTC, TOTNET ];
                              P      C    2277.50

 SUMMARY:+ project CURNTDATA[ ACCTE, ACCTP, TOTNET ];
                              E      P    2277.50
```

Summary program - 3 & 4: CASHIN and CASHOUT

The detail relation CASHIN contains daily transactions of cash debit and accounts receivable credit information. The relation CASHOUT contains daily transactions of accounts payable debit and cash credit information. We illustrate the operation with examples.

```
CASHIN( INV# CHRONOL $AMT )  RECENT  TOTAMT
         5   790330   100      0
         1   790331    50      0
         2   790401    80      1       1122
         3   790401   120      1       1122
         6   790421   850      1       1122
         4   790429    25      1       1122
         7   790429    15      1       1122
         8   790430    32      1       1122
```

<< Summary of CASHIN >>

```
 let TOTAMT be red+ of $AMT;

 SUMMARY:+ project( select CASHIN[RECENT] )[ACCTC, ACCTR, TOTAMT];
                                            C      R      1122
```

```
CASHOUT(B_LAD# VEND# CHRONOL $AMT)  RECENT  TOTAMT
         B1     V1   790331   100      0
         B5     V1   790331   800      0
         B2     V5   790401   350      1      1560
         B3     V3   790401   110      1      1560
         B6     V4   790428  1000      1      1560
         B4     V4   790430    20      1      1560
         B7     V2   790430    80      1      1560
```

<< Summary of CASHOUT >>

```
SUMMARY:+ project( select CASHOUT[RECENT] )[ACCTP, ACCTC, TOTAMT];
                                            P      C      1560
```

Summary program - 5: Receipts (RECLINE)

The detail relation RECLINE contains the purchase cost of the received raw material or assets. We assume that we can distinguish it from a domain in RECLINE, CLASS - M for raw material, F for fixed assets.

```
<< Summary of RECLINE >>

 let TOTCOST be equiv+ of PURCHASECOST by CLASS;

 SUMMARY:+ project( select RECLINE [ RECENT ] )
                  [ CLASS, ACCTP, TOTCOST ];
```

where CLASS and ACCTP corresponds to accounts to be debited and accounts to be credited respectively. We illustrated the above with the following example RECLINE:

					UNIT	FREIGT	PURCHASE		
(B_LAD#	VEND#	CHRONOL	PART#	QTY	PRICE	RATE	COST	CLASS)	TOTCOST
B1	V1	790331	123	30	0.50	0.01	15.15	M	
B2	V2	790401	070	1	200.00	0.005	201.00	F	301.50
B3	V3	790415	208	500	0.50	0.01	252.50	M	580.25
B3	V3	790415	103	200	1.00	0.01	202.00	M	580.25
B3	V3	790415	100	2	50.00	0.005	100.50	M	580.25
B4	V4	790415	010	1	100.00	0.005	100.50	F	301.50
B5	V1	790429	123	30	0.50	0.01	15.15	M	580.25
B5	V1	790429	110	100	0.10	0.01	10.10	M	580.25

Summary program - 6: EQUIP (Assets)

We will show fixed assets depreciation calculation using straight-line method.

$$\frac{\text{Cost - Salvage value}}{\text{Service life in years}} \times \frac{\text{accounting period in month}}{12}$$

(accounting_period_in_month should be predetermined value)

```
<< Summary of EQUIP >>

 let ACCTPERIOD be accounting_period_in_month;

 let DEPRE be (EQCOST - SALVAGEVAL)*ACCTPERIOD/(LIFE*12);

 let TOTDEPRE be red+ of DEPRE;    << E <-- F >>

<< consider the assets currently in service >>

 let CALCULATEDEPRE be PERIOD <= DISCARDATE;

 let ACCTF be 'F';    << account F >>

 SUMMARY:+ project( select EQUIP[ CALCULATEDEPRE ] )
                  [ ACCTE, ACCTF, TOTDEPRE ];
                      E      F      150.42
```

We illustrate the above process with an example which shows the basic domains which are necessary for illustration.
(assume accounting period as one month and the computation is for April 30, 1979 - 790430)

The sample relation EQUIP:

				SALVAGE	DISCAR	CALCULATE	
(EQUIP#	CHRONOL	EQCOST	LIFE	VAL	DATE)	DEPRE	DEPRE
E1	730101	10000	10	500	830101	1	79.17
E2	731201	500	5	20	781201	0	
E3	750504	9000	20	500	950504	1	35.42
E4	760101	4500	10	200	860101	1	35.83

All along, we have collected all transactions from each detail file. We can illustrate the contents of SUMMARY with our examples.

SUMMARY(	ACCTDR	ACCTCR	AMT)	TOTAL
	R	G	4625.00	4625.00
	G	A	4625.00	4125.00
	A	M	2222.00	2222.00
	A	E	2403.00	2403.00
	R	E	960.00	960.00
	E	T	472.50	472.50
	P	C	2277.50	3837.50
	E	P	2277.50	2277.50
	C	R	1122.00	1122.00
	P	C	1560.00	3837.50
	F	P	301.50	301.50
	M	P	580.25	580.25
	E	F	150.42	150.42

* Note that there are two P <-- C , one from payroll and one from cash payments, in above illustration.

Now we summarize it and illustrate with the same example, assuming today is April 30, 1979.

```
 let PERIOD be today;

 let TOTAL be equiv+ of AMT by (ACCTDR, ACCTCR);

 SPREAD:+ project SUMMARY[ PERIOD, ACCTDR, ACCTCR,  TOTAL];
                           790430      R       G      4625.00
                           790430      G       A      4625.00
                           790430      A       M      2222.00
                           790430      A       E      2403.00
                           790430      R       E       960.00
                           790430      E       T       472.50
                           790430      P       C      3837.50
                           790430      E       P      2277.50
                           790430      C       R      1122.00
                           790430      F       P       301.50
                           790430      M       P       580.25
                           790430      E       F       150.42
```

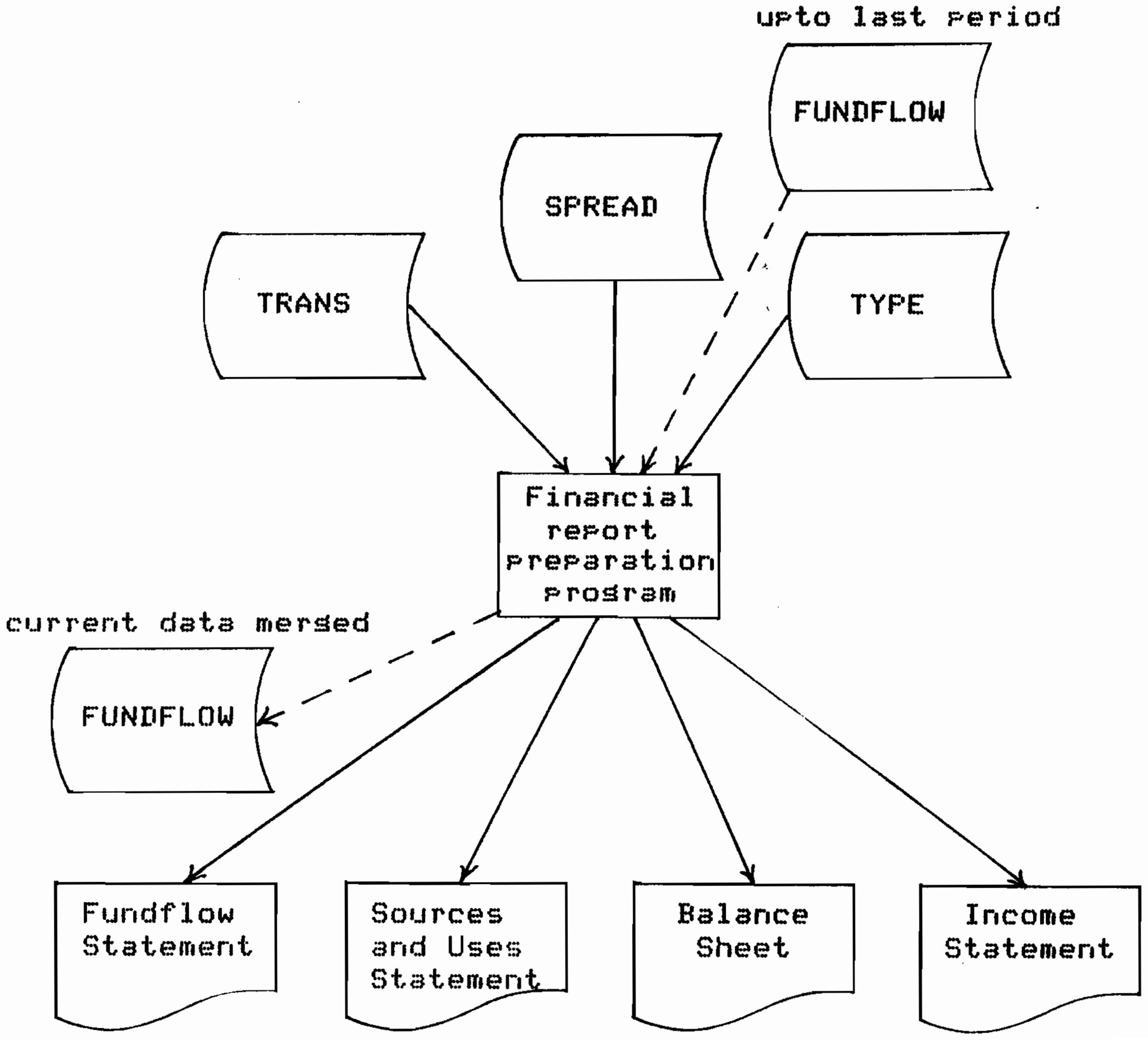
The permanent relation SPREAD has layers of information as follows:

```
SPREAD( PERIOD, ACCTDR, ACCTCR, TOTAL )
             .       .       .      .
             .       .       .      .
             .       .       .      .
        790131  transactions in Jan.79
             .       .       .      .
             .       .       .      .
             .       .       .      .
        790228  transactions in Feb.79
             .       .       .      .
             .       .       .      .
             .       .       .      .
        790331  transactions in Mar.79
             .       .       .      .
             .       .       .      .
             .       .       .      .
        790430  transactions in Apr.79
```

K4.3 Financial reports

The principle relation is SPREAD (a summary of all transactions for each period), with some supporting information in TYPE (account types - Asset, Liability, Equity - and descriptions) and TRANS (legal transactions and their descriptions - the topology of the accounting network).

We present the derivation of the basic financial reports, Fund flow statement - Sources and uses statement, Balance Sheet, Income statement.



Funds flow statement

We select all transactions of current period from SPREAD. The FUNDFLOW is really the balance of the current period of the SPREAD sheet, with signs adjusted for assets or liabilites and equities.

```
 let RECENT be CHRONOL > LASTPERIOD;
 let PERIOD be today;      << assume function today >>
 let TOTIN  be equiv + of TOTAL by ACCTDR;
 let TOTOUT be equiv + of TOTAL by ACCTCR;
 let FF be if ACTYPE='Asset' then TOTIN - TOTOUT else TOTOUT - TOTIN;

 CURNTSPREAD:=project(select SPREAD[RECENT])[ACCTDR,ACCTCR,TOTAL];

 FUNDFLOW:+ project(((project CURNTSPREAD [ACCTDR,TOTIN])[ACCTDR]
                uJoin (project CURNTSPREAD [ACCTCR,TOTOUT])[ACCTCR])
                [ACCTCR] iJoin TYPE[ACCT])[ACCT,ADESCR,ACTYPE,FF];
```

The printed funds flow report might omit the ACCT number, which is an internal identifier, and print only the descripion of the account, ADESCR.

We can illustrate this last computation step by step from our example:

```
          CURNTSPREAD( ACCTDR ACCTCR  TOTAL   )
                         C      R    1122.00
                         M      P     580.00
                         A      M    2222.00
                         A      E    2403.00
                         G      A    4625.00
                         R      G    4625.00
                         R      E     960.00
                         F      P     301.50
                         P      C    3837.50
                         E      F     150.42
                         E      P    2277.50
                         E      T     472.50
```

```
project CURNTSPREAD[ACCTDR, TOTIN]
                       C     1122.00
                       M      580.25
                       A     4625.00
                       G     4625.00
                       R     5585.00
                       F      301.50
                       P     3837.50
                       E     2829.18
```

```
project CURNTSPREAD[ACCTCR, TOTOUT]
                          R     1122.00
                          P     3159.25
                          M     2222.00
                          E     3363.00
                          A     4625.00
                          G     4625.00
                          C     3837.50
                          F      150.42
                          T      472.50
```

uJoin on ACCTDR,ACCTCR:

```
                          ACCTCR
                          ACCTDR  TOTIN      TOTOUT
                             C   1122.00    3837.50
                             M    580.25    2222.00
                             A   4625.00    4625.00
                             G   4625.00    4625.00
                             R   5585.00    1122.00
                             F    301.50     150.42
                             P   3837.50    3159.25
                             T               472.50
                             E   2829.18    3363.00
```

iJoin with TYPE:

```
 TYPE(ACCT    ACTYPE       ADESCR            )     ( Chart of accounts )
        C     Asset      Cash
        M     Asset      Raw material
        A     Asset      Goods in assembly
        G     Asset      Finished goods
        R     Asset      Accounts receivable
        F     Asset      Fixed assets
        P   Liability    Accounts payable
        T   Liability    Taxes payable
        E    Equity      Stockholders' Equity
```

Final result:

```
FUNDFLOW( ACCT    ADESCR                ACTYPE         FF       )

           C    Cash                    Asset     -2715.50

           M    Raw material            Asset     -1641.75

           A    Goods in assembly       Asset         0

           G    Finished goods          Asset         0

           R    Accounts receivable     Asset      4463.00

           F    Fixed assets            Asset       151.08

           P    Accounts payable      Liability    -678.25

           T    Taxes payable         Liability     472.50
```

Sources and uses statement

The Sources and uses statement, SORUSE, is a rearrangement of FUNDFLOW. Decrease in assets and increases in liabilities are sources of funds. The opposite is use of funds.

```
 let SUAMT be abs(FF);
 let SU be if ACTYPE='Asset' and FF <= 0
           or ACTYPE='Asset' and FF > 0 then 'Source' else 'Use';
 SORUSE:+ project FUNDFLOW [ ACCT, ADESCR, SUAMT, SU ];
```

the result would be:

```
 SORUSE( ACCT    ADESCR                SUAMT      SU    )
          C    Cash                  2715.50  Source
          M    Raw material          1641.75  Source
          A    Goods in assembly        0     Source
          G    Finished goods           0     Source
          R    Accounts receivable   4463.00   Use
          F    Fixed assets           151.08   Use
          P    Accounts payable       678.25   Use
          T    Taxes payable          472.50  Source
          E    Stockholders' equity   462.58  Source
```

Balance Sheet

Starting again with SPREAD, we can extract BALANCE. We use entire contents of SPREAD, instead of the current period of SPREAD.

```
 let BAL be TOTIN - TOTOUT;
 let DRCR be if BAL > 0 then 'DR' else 'CR';
 let ABSBAL be abs(BAL);

 BALANCE:+ project((project SPREAD[ACCTDR,TOTIN])[ACCTDR]
          uJoin (project SPREAD[ACCTCR,TOTOUT])[ACCTCR])
          [ ACCTCR, BAL ];
```

This gives total balance assuming zero starting balance.

```
 BALANCESHEET:+ project( TYPE[ACCT] uJoin BALANCE[ ACCTCR ])
                  [ PERIOD, ACCT, ADESCR, ABSBAL, DRCR ];
```

The result would be a traditional Balance Sheet. Depending on DRCR, if it is 'DR' then print the balance at left otherwise print it at right. This is practically the same as FUNDFLOW, except for the range of period selected.

Income statement

The INCOME statement derivation is more easily understood if split into a derivation of REVENUE and EXPENSE. The revenues are given by the columns of equity accounts and expenses are given by the rows (see the matrix representation of spread sheet of one period in page 46).

To get REVENUE, we select the E column of the SPREAD sheet for the current period, and to add description of the transaction, join it with the E column of TRANS (the legal transactions according to our accounting network)

TRANS(ACCTDR	ACCTCR	TDESCR)
C	R	Collect Accounts receivable
M	P	Purchase of Raw materials
A	M	Raw material costs
A	E	Variable costss charged to acc. A
G	A	Assembling costs
R	E	Gross profit on sales
F	P	Purchase of Fixed assets
P	C	Pay Accounts payable
E	F	Depreciation
E	P	Manufacturing & operating expenses
E	T	Taxes - Payroll

```
let CRE be ACCTCR='E';

REVENUE:+(project(select CURNTSPREAD[ CRE ])[ACCTDR,TOTAL])
        [ACCTDR] iJoin ( select TRANS[ CRE ] )[ ACCTDR ];
```

Similary, EXPENSE

```
let DRE be ACCTDR='E';

EXPENSE:+(project(select CURNTSPREAD[ DRE ])[ACCTCR,TOTAL])
         [ACCTCR] iJoin ( select TRANS[ DRE ] )[ ACCTCR];
```

The INCOME statement is the union of these two, suitably flagged as revenues or expenses, with the net changes in equity (which we can get from the FUNDFLOW). We inject a new description for the profit and loss line.

```
 let REV be 'Revenue';
 let EXP be 'Expense';
 let FFE be ACCT='E';
 let PL be if FF >= 0 then 'Profit' else 'Loss';
 let PLDESCR be 'Net' || PL || 'after Tax';
 let PLAMT be abs(FF);

 INCOME:+ project((REVENUE[ACCTDR,TDESCR,TOTAL,REV] ujoin
             EXPENSE[ACCTCR,TDESCR,TOTAL,EXP])[ACCTCR,TDESCR,TOTAL,EXP]
          ujoin (project(select FUNDFLOW[FFE])[ACCT,PLDESCR,PLAMT,PL]))
                [ ACCT, TDESCR, TOTAL, REV ];
```

The INCOME statement looks like as follows if we use our example:

```
 INCOME( ACCT      TDESCR                             TOTAL      REV   )

           A    Variable costs charged to acc. A    2403.00   Revenue

           R    Gross profit on sales                960.00   Revenue

           F    Depreciation                         150.42   Expense

           P    Manufacturing & operating expenses 2277.50   Expense

           T    Taxes - payroll                      472.50   Expense

           E    Net Profit after Tax                 462.58   Profit
```

The printed income statement might omit ACCT number which is an internal identifier, and print only the rest.

For a more elaborate income statement, see extensions.

K4.4 Finanacial reports - extensions

The relation FUNDFLOW which we derived in the previous discussion is really the changes in our Balance sheet accounts during an accounting period. If we had stored each monthly result of funds flow calculation in FUNDFLOW, then the entire contents of FUNDFLOW, which originated when the firm is established, would give the firm's financial position at this time - Balance sheet at today. Depending on the selection of date, we can calculate the firm's financial position at any given time.

Comparative balance sheet

We will demonstrate the derivation of Comparative Balance sheet, which consists of the item amounts from two of the firm's successive Balance sheets and the same period a year ago arranged side by side in a single statement.

We assume the FUNDFLOW was stored as

FUNDFLOW(PERIOD, ACCT, FF)

For illustration purposes, we assume the dates we are interested in are at Dec.31,1978 (31/12/78) and 31/12/77, and assume function at 31/05/79.

```
 let TONOW be PERIOD <= today;           << assume function today >>
 let JANMAY79 be TONOW and PERIOD > end78;      << Jan - May '79 >>
 let TO78 be PERIOD <= end78;                << up to end of '78 >>
 let JANDEC78 be TO78 and PERIOD > end77;      << during the '78 >>
 let TO77 be PERIOD <= end77;             << up to end of year 77 >>
 let UPTONOW be equiv+ of FF by ACCT;  << total balance by ACCT >>
 let UPTO78 be UPTONOW;  << total balance up to end of '78 >>
 let UPTO77 be UPTONOW;  << total balance up to end of '77 >>
 let FF79 be UPTONOW;     << changes in '79 >>
 let FF78 be UPTONOW;     << changes in '78 >>
 let PERCENT79 be FF79*100/UPTO78;
                               << percentage of changes during Jan - May '79 >>
 let PERCENT78 be FF78*100/UPTO77; << percent. of changes dur. '78 >>
```

```
COMPARATIV:+ project ((((((project TYPE[ ACCT,ADESCR ])[ ACCT ]
       uJoin (project FUNDFLOW[ ACCT,UPTONOW ])[ ACCT ])[ ACCT ]
  uJoin( project( select FUNDFLOW[ TO78 ])[ACCT,UPTO78])[ACCT])
  [ACCT] uJoin(project(select FUNDFLOW[TO77])[ACCT,UPTO77])[ACCT])
 [ACCT] uJoin(project(select FUNDFLOW[JANMAY79])[ACCT,FF79])[ACCT])
 [ACCT] uJoin(project(select FUNDFLOW[JANDEC78])[ACCT,FF78])[ACCT])
 [ACCT,ADESCR,UPTONOW,UPTO78,UPTO77,FF79,PERCENT79,FF78,PERCENT78];

balances of accounts                 increase or decrese
  at present                           during Jan - May '79
  year ended Dec.31,1978                  amount
  year ended Dec.31,1977                  percentage
                                       during 1978
                                          amount
                                          percentage
```

Income statement - extension 1

The figures on the income statement can be given in an alternate form as a percent of total revenues. We will show the process for the income statement for the year ending today using the same technique as before. We summarize

SPREAD(PERIOD, ACCTDR, ACCTCR, TOTAL)

which is a collection of monthly summaries of all transactions.

```
 let TAMT be equiv+ of TOTAL by (ACCTDR, ACCTCR);
 << let CURNTYEAR be PERIOD > lastday; >>
 SPREAD_Y:+ project( select SPREAD[ CURNTYEAR ])[ACCTDR,ACCTCR,TAMT]
```

Now we derive the revenue for the year

```
 let CRE be ACCTCR='E';
 REVENUE_Y:+ project((project(select SPREAD_Y[CRE])[ACCTDR,TAMT])
             [ ACCTDR ] iJoin ( select TRANS[ CRE ])[ ACCTDR ]
             [ ACCTDR, TDESCR, TAMT ];
```

We need the total revenue to calculate the percentage.

```
 let TOTREV be red+ of TAMT;
 let BLANK be ' ';
 let REVDESCR be 'Total Revenue';
 TOTALREV:+ project REVENUE_Y[ BLANK, REVDESCR, TOTREV ];
 REVAMT:+ project TOTALREV[ TOTREV ];   << single valued relation >>
```

We will use REVAMT to calculate percentage later.

```
 let DRE be ACCTDR='E';
 EXPENSE_Y:+project((project(select SPREAD_Y[DRE])[ACCTCR,TAMT])
           [ACCTCR] iJoin (select TRANS[DRE])[ACCTCR])
           [ ACCTCR, TDESCR, TAMT ];
```

We get total expenses

```
 let EXPDESCR be 'Total Expenses';
 let TOTEXP be red+ of TAMT;

 TOTALEXP:+ project EXPENSE_Y[ BLANK, EXPDESCR, TOTEXP ];
```

The usual income statement of the year is the union of REVENUE_Y, TOTALREV, TXPENSE_Y, TOTALEXP with the netchanges in equity which we can get from FUNDFLOW_YEAR. Then we make Cartesian product with total revenue REVAMT.

```
 let TOTFF be equiv+ of FF by ACCT;

 FUNDFLOW_YEAR:+ project( select FUNDFLOW [ CURNTYEAR ])
                         [ ACCT, TOTFF ];

 let FFE be ACCT='E';
 let PL be if TOTFF >= 0 then 'Profit' else 'Loss';
 let PLDESCR be 'Net' || PL || 'After Tax';
 let PLAMT be abs(TOTFF);

 INCOME_Y:+ project(( REVENUE_Y[ ACCTDR, TDESCR, TAMT ]  uJoin
                     TOTALREV[ BLANK, REVDESCR, TOTREV ] uJoin
                     EXPENSE_Y[ ACCTCR, TDESCR, TAMT ]   uJoin
                     TOTALEXP[ BLANK, EXPDESCR, TOTEXP ] uJoin
   (project(select FUNDFLOW_YEAR[ FFE ])[ ACCT, PLDESCR, PLAMT ])
                     [ ACCT, PLDESCR, PLAMT ]) product REVAMT)
                     [ ACCT, TDESCR, TAMT, TOTREV ];
```

Finally we can get the income statement of the year ending today with figures on percent of total revenue.

```
 let PERCENT be 100*TAMT/TOTREV;

 INCOME_PERCENT:+ project INCOME_Y[ ACCT, TDESCR, TAMT, PERCENT ];
```

We illustrate the final step with the previous example:

The relation, INCOME_Y, would be as follows:

(ACCT	TDESCR	TAMT	TOTREV)	PERCENT
A	Variable costs charged to acc. A	2283.00	3243.00	70.40
R	Gross profit on sales	960.00	3243.00	29.60
	Total Revenue	3243.00	3243.00	100.00
F	Depreciation	79.18	3243.00	2.44
P	Manufacturing & operating expenses	2277.50	3243.00	70.23
T	Taxes - payroll	472.50	3243.00	14.57
	Total Expenses	2829.18	3243.00	87.24
E	Net Profit After Tax	413.82	3243.00	12.76

Income statement with percent of total revenues, INCOME_PERCENT:

(ACCT	TDESCR	TAMT	PERCENT)
A	Variable costs charged to acc. A	2283.00	70.40
R	Gross profit on sales	960.00	29.60
	Total Revenue	3243.00	100.00
F	Depreciation	79.18	2.44
P	Manufacturing & operating expenses	2277.50	70.23
T	Taxes - payroll	472.50	14.57
	Total Expenses	2829.18	87.24
E	Net Profit After Tax	413.82	12.76

The printed income statement might omit ACCT number which is an internal identifier, and print only the rest.

Income statement - extension 2

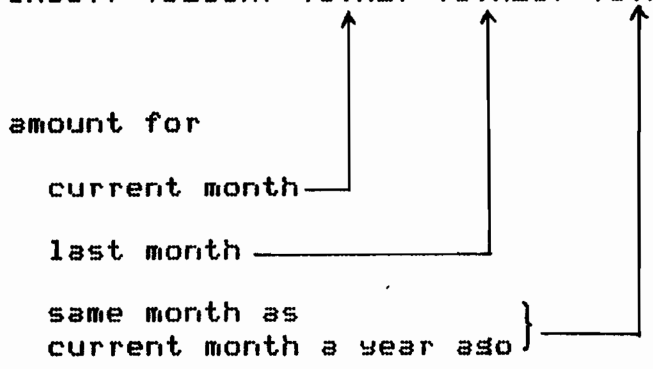
The comparative income statement with the amount of increases or decreases and their percentages can be derived with the same technique used to prepare the Comparative Balance Sheet. Now we will derive the Comparative Income Statement using QT-expressions which are self explanatory and it is easy to see the process.

We will display amounts for the current month, May '79, the last month, April '79, the same month as the current month a year ago, May '78, the changes and the percentage of changes.

Note that the relations are stored as follows:

```
  SPREAD( PERIOD, ACCTDR, ACCTCR, TOTAL )

  FUNDFLOW( PERIOD, ACCT, FF )

 let ENDMAY79 be end_of_May'79;
 let ENDAPR79 be end_of_April'79;
 let ENDMAR79 be end_of_March'79;
 let ENDMAY78 be end_of_May'78;
 let ENDAPR78 be end_of_April'78;

 let TOTAL1 be TOTAL;
 let TOTAL2 be TOTAL;

 REVENUE:+(((ACCTDR,TOTAL  if ENDMAY79>=PERIOD and PERIOD>ENDAPR79
               and ACCTCR='E' in SPREAD )[ ACCTDR ] uJoin
            (ACCTDR,TOTAL1 if ENDAPR79>=PERIOD and PERIOD>ENDMAR79
               and ACCTCR='E' in SPREAD )[ ACCTDR ])[ ACCTDR ] uJoin
            (ACCTDR,TOTAL2 if ENDMAY78>=PERIOD and PERIOD>ENDAPR78
               and ACCTCR='E' in SPREAD )[ ACCTDR ])[ ACCTDR ] iJoin
            (ACCTDR,TDESCR if ACCTDR='E' in TRANS )[ ACCTDR ];

 EXPENSE:+(((ACCTCR,TOTAL  if ENDMAY79>=PERIOD and PERIOD>ENDAPR79
               and ACCTDR='E' in SPREAD ) uJoin
            (ACCTCR,TOTAL1 if ENDAPR79>=PERIOD and PERIOD>ENDMAR79
               and ACCTDR='E' in SPREAD )[ ACCTCR ])[ ACCTCR ] uJoin
            (ACCTCR,TOTAL2 if ENDMAY78>=PERIOD and PERIOD>ENDAPR78
               and ACCTDR='E' in SPREAD )[ ACCTCR ])[ ACCTCR ] iJoin
            (ACCTCR,TDESCR if ACCTDR='E' in TRANS )[ ACCTCR ];
```

```
let FF1 be FF;
let FF2 be FF;
let NDESCR be 'Net Income';

NETIN:+(((ACCT,FF  if ENDMAY79>=PERIOD and PERIOD>ENDAPR79
             and ACCT='E' in FUNDFLOW )[ ACCT ] ujoin
         (ACCT,FF1 if ENDAPR79>=PERIOD and PERIOD>ENDMAR79
             and ACCT='E' in FUNDFLOW )[ ACCT ])[ ACCT ] ujoin
         (ACCT,FF2 if ENDMAY78>=PERIOD and PERIOD>ENDAPR78
             and ACCT='E' in FUNDFLOW )[ ACCT ];

let INC1 be TOTAL - TOTAL1;
        << amount of inc./dec. during ENDMAY79>=PERIOD>ENDAPR79 >>
let PERCNT1 be INC1*100/TOTAL1;
    << percentage of inc./dec. during ENDMAY79>=PERIOD>ENDAPR79 >>
let INC2 be TOTAL1 - TOTAL2;
        << amount of inc./dec. comapre with same period a year ago >>
let PERCNT2 be INC2*100/TOTAL2;
    << percentage of inc./dec. compare with same period a year ago >>

INCOMEX:+ project((REVENUE[ACCTDR,TDESCR,TOTAL,TOTAL1,TOTAL2] ujoin
                   EXPENSE[ACCTCR,TDESCR,TOTAL,TOTAL1,TOTAL2])
                          [ACCTCR,TDESCR,TOTAL,TOTAL1,TOTAL2] ujoin
                   NETIN[ACCT,NDESCR,FF,FF1,FF2])
[ACCT, TDESCR, TOTAL, TOTAL1, TOTAL2, INC1, PERCNT1, INC2, PERCNT2];
```



5. Conclusion

The advantages of the use of Aldat in applications programs are its conciseness and its standardization. As we have demonstrated in our examples, most programs are short and simple.

Despite the power of Aldat, as we have presented, there are some drawbacks. They are: we never mentioned how to prepare input, or how to correct wrong tuples created through input errors. Nor do we mentioned output printing. Since the problem of editing input to the database has not been completely solved in Aldat, we have to assume that the input file exists with no errors. Similarly we have not dealt with the details of output formatting and prints, so we content ourselves with producing all the proper attributes needed to print the report. Thus the most important needs in Aldat are input verification and editing and printing procedures.

Further improvements to the language that we would like to suggest are as follows:

a) To select "surface" of the relation (cf. section C2.4)

eg. SALHIST(ENP#, CHRONOL, SALARY, OVERSALARY)

Instead of two statements

```
let SCHRON be CHRONOT = equiv max of CHRONOL by EMP#;
select SALHIST[ SCHRON ];
```

We prefer

```
surface SALHIST;
```

b) The syntax of a period selection

eg. Instead of ENDMAY79>PERIOD <u>and</u> PERIOD>ENDAPR79

We prefer ENDMAY79>PERIOD>ENDAPR79

(as in mathematics $a > b > c$)

c) Rename expression

eg. Instead of two expression

```
let TOTAL1 be TOTAL;
let TOTAL2 be TOTAL;
```

We prefer

```
let TOTAL1,TOTAL2 be TOTAL;
```

With the above limitations in mind, there is no doubt that Aldat is good at any data processing problem which has to deal with file manipulation. There are many other commercial applications that could be attempted as further work.

Appendix A: List of Relations Used

relation name on	(attributes)	page
BALANCE	(ACCT, BAL)	66
BALANCESHEET	(PERIOD, ACCT, ADESCR, ABSBAL, DRCR)	66
BOM	(ASSEMBLY#, SUBASSY#, QTY)	16,30
CASHIN	(INV#, CHRONOL, $AMT)	58
CASHOUT	(B_LAD#, VEND#, CHRONOL, $AMT)	58
COMPARATIV	(ACCT, ADESCR, UPTONOW, UPTO78, UPTO77, FF89, PERCENT79, FF77, PERCENT78)	70
CORRECTION	(CUST#, ORD#, CHRONOL, FGPART#, CRQTY, $CR, $CRPROFIT)	42
COSTTABLE	(RMPART#, $RMCOST $ROUTCOST, $LABCOST)	29
CRDTL	(MEMO#, CUST#, ORD#, FGPART#, CRQTY)	37
CRLINE	(MEMO#, FGPART#, CRQTY)	37
CRMEMO	(MEMO, CHRONOL, CUST#, ORD#, INV#)	37
CURNTDATA	(CHRONOL, FGPART#, QTY, QTYFGCOST, QTYRMTOT, QTYVARCOST, QTYFGPROFIT)	56
CURNTSPREAD	(ACCTDR, ACCTCR, TOTAL)	64
CUSTRP	(CUST#, CHRONOL, NQTY, $CSALE, $CPRFT),	41
EQUIP	(EQUIP#, CHRONOL, EQCOST, LIFE)	5
EQUIP	(EQUIP#, CHRONOL, EQCOST, LIFE, SALVAGE)	60
EQUIP	(EQUIP#, CHRONOL, EQCOST, LIFE, SALVAGEVAL, DISCARDATE)	60
EQUIPCOST	(EQUIP#, ECOST)	24
EXPENSE	(ACCT, TOTAL)	67
EXPENSE_Y	(ACCTCR, TDESCR, TAMT)	71

FGCOST	(FGPART#, CHRONOL, RMTOT, ROTOT, LABTOT, $FGCOST)	32
FGPRICE	(FGPART#, CHRONOL, $FGPRICE)	33
FGPROFIT	(FGPART#, CHRONOL, $FGPRRICE, $FGCOST, RMTOT, ROTOT, LABTOT, $FGPROFIT)	33
FUNDFLOW	(ACCT, ADESCR, ACTYPE, FF)	64,65
FUNDFLOW (extention)	(PERIOD, ACCT, FF)	69
FUNDFLOW_YEAR	(ACCT, TOTFF)	71
INCOME	(ACCT, TDESCR, TOTAL, REV)	68
INCOMEX	(ACCT, TDESCR, TOTAL, TOTAL1, TOTAL2, INC1, PERCNT1, INC2, PERCNT2)	74
INCOME_PERCENT	(ACCT, TDESCR, TAMT, PERCENT)	71,72
INCOME_Y	(ACCT, TDESCR, TAMT, TOTREV)	71,72
LABOUR	(EMP#, OP#, EQUIP#, EMPCOST)	7,27
MFGCOST	(OP#, ASSEMBLY#, RCOST, EMPCOST)	7
NETIN	(ACCT, NDESCR, FF, FF1, FF2)	74
PARTRPT	(FGPART#, CHRONOL, TOTQTY, $PARTSALE, $PARTFIT)	42
PAYTAX	(EMP#, SALARY, FLOOR, BASE, TAX%)	9
PROFIT	(CUST#, FGPART#, NQTY, $SALE, $PROFIT, SALES#)	11,40
RECLINE	(B_LAD#,VEND#,CHRONOL,PART#,QTY,UNITPRICE, FREIGHTRATE, PURCHASECOST, CLASS)	59
REVAMT	(TOTREV)	70
REVENUE	(ACCT, TOTAL)	67
REVENUE_Y	(ACCTDR, TDESCR, TAMT)	70
RMCOST	(RMPART#, CHRONOL, $RMCOST)	28

ROUTCOST	(OP#, ASSEMBLY#, RCOST)	7,25
ROUTING	(OP#, ASSEMBLY#, EQUIP#, CHRONOL, RTIME)	25
SALDETAIL	(PAYDATE, EMP#, SALARY, TAX, NET)	53
SALEDTL	(CUST#, ORD#, FGPART#, SHQTY, CRQTY, NQTY, SALES#)	38
SALERPT	(SALES#, CHRONOL, $SSALE, $SPRFT)	41
SALHIST	(EMP#, CHRONOL, SALARY)	53,54
SALHIST	(EMP#, CHRONOL, SALARY, OVERSALARY)	12,26
SHIPDETAIIL	(CHRONOL, FGPART#, QTY, QTYFGCOST, QTYRMTOT, QTYVARCOST, QTYFGPROFIT)	51
SHIPDTL	(CUST#, ORD#, CHRONOL, B-LAD#, SALES#, FGPART#)	38
SHIPLINE	(B_LAD#, FGPART#, SH_QTY)	36
SHIPMENT	(ÇUST#, ORD#, CHRONOL, B_LAD#, SALES#)	35
SORUSE	(ACCT, ADESCR, SUAMT, SU)	66
SPREAD	(PERIOD, ACCTDR, ACCTCR, TOTAL)	61
SPREAD_Y	(ACCTDR, ACCTCR, TAMT)	70
SUMMARY	(ACCTDR, ACCTCR, AMT)	56,61
TAXTABLE	(FLOOR, CHRONOL, BASE, TAX%)	8
TEMP1	(OP#, ASSEMBLY#, LABCOST, RCOST)	28
TEMPCOST	(RMPART#, $RMCOST, $ROUTCOST, $LABCOST)	31
TIMECARD	(EMP#, CHRONOL, OP#, EQUIP#, HOURS, OVERTIME)	26
TOTALEXP	(BLANK, EXPDESCR, TOTEXP)	71
TOTALREV	(BLANK, REVDESCR, TOTREV)	70
TRANS	(ACCTDR, ACCTCR, TDESCR)	67
TYPE	(ACCT, ACTYPE, ADESCR)	64,65
T1	(ASSEMBLY#, RMPRICE, ROTPRICE, LABPRICE)	30

Appendix B: List of Attributes Used

attributes:	meaning	page
ABS		66
ABSBAL		66
ACCT	ACCounT number	64,65
ACCTA		56
ACCTA		56
ACCTC		57
ACCTCR	ACCounT number to be CRedited	61
ACCTDR	ACCount number to be Debited	61
ACCTE		60
ACCTF		60
ACCTG		56
ACCTM		56
ACCTP		57
ACCTR		56
ACCTT		57
ACTYPE	ACcount TYPE	64,65
ADESCR	Account number DESCRiption	64,65
ASSEMBLY#	ASSEMBLY number	7,25
BAL		66
BASE	BASE tax	8
BLANK		70
B-LAD#	Bill of LADing number	35
CALCULATEDEPRE		60

CHRONOL	CHRONOLogy of tuples created	5,23
CLASS		59
CRE		66
CRQTY	Returned quantity	37
CRCHRON	returned goods of the current period	37
CURNT		51
CURNTPAY		57
CURSALE	CURRent month's SALE	39
CUST#	CUSTomer number	35
DEPRE		60
DISCARDATE	DISCARDing DATE of equipment	60
DRCR		66
DRE		67
ECOST	Equipment COST per hour	24
EMPCOST	EMPloyee COST	7,27
EMP#	EMPloyee number	7,27
EQCOST	EQuipment COST	5,23
EQUIP#	EQUIPment number	5,23
EXP		68
EXPDESCR		71
FF	Fund Flow amount	64,65
FF1,F12		74
FF78,FF79		69
FFE		68
FGPART#	Finished Goods PART number	32
FLOOR	Lower limit of the salaries	8

SU		66
SUAMT		66
SUBASSY#	SUBASSemblY number	16,30
TAMT	Total AMounT	70
TAX%	percentage of TAX	8,54
TDESCR	Transaction DESCRiption	67
TFGCOST		56
TFGPROFIT		56
TO77,TO78,TONOW		69
TOTAL1,TOTAL2		73
TOTAMT		58
TOTAX		57
TOTCOST		59
TOTDEPRE		60
TOTEXP		71
TOTFF		71
TOTIN		64
TOTNET		57
TOTOUT		64
TOTQTY	TOTal QuantiTY	17,42
TOTREV	TOTal REVenue	70
UPTONOW,UPTO77,UPTO78		69
VEND#	VENDor number	58
$AMT	AMounT	58
$CPRFT	PRoFiT by Customer	11,41
$CR	CRedit amount	42

References

1. Merrett, T.H.

   Aldat - Augmenting the Relational Algebra for Programmers.

   McGill, Technical Report SOCS - 78.1, Nov. 1977.

2. Eliason, A.L. & Kitts, K.D.

   Business Computer Systems and Applications, Chap. 4 - 18.

   SRA, 1974.