duplicate

4 A

IBM Research Center
P. O. Box 218
Yorktown Heights, N. Y.

January 13, 1961

Subject: Tabular Techniques Development

In the past two years a number of people have explored the possibility of using tabular form as a means of expressing decision processes so as to present these logical decisions in a more understandable way. In order to keep IBM personnel acquainted with this area of development, we are planning to distribute appropriate material from time to time, reviewing current work in developing tabular techniques. The people receiving this material have been selected because of their interest in programming methods.

You may well ask, "What do you mean by tabular techniques?" The full meaning of these techniques will be described in the various papers to be distributed. For the present, let us define tabular techniques as being the use of a table form to present the decision logic or operating procedures. In other words, tabular techniques will present programming and system descriptions in a table format. The material we distribute will be of four types: (1) material obtained from customers experimenting with tabular form, (2) material obtained from the Committee on Data Systems Languages (CODASYL) concerning the work on tabular form development, (3) material produced within IBM describing technical developments or explaining the use of tabular form, and (4) material produced by competitors, describing their developments and applications.

Since this is the first release, we would appreciate suggestions as to others who should be receiving this material, and any specific comments or ideas concerning the attached work. If you have any questions concerning these items, please call or write me.

This first distribution includes two items:

(1) A status report on current developments in tabular techniques.

(2) A copy of a speech given by Mr. T. F. Kavanagh, of General Electric, at the Eastern Joint Computer Conference on December 14, 1960. (Note that this speech differs from the paper printed in EJCC proceedings).

Burton Grad

Burton Grad
Project Coordinator

UNIFIED SYSTEMS APPROACH

CLEARINGHOUSE REPORT

TABULAR TECHNIQUES DEVELOPMENT

STATUS -- DECEMBER, 1960

January 13, 1961
Ref. No. 1A2

Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

December, 1960

Tabular Techniques Development Status

Burton Grad
IBM

During 1960 the use of tabular techniques in systems and programming languages has grown to become an area of significant experimentation. The one thing all these developments share is a tabular (or table) layout, in which the decision or program logic is recorded: information has positional significance as well as meaning contained within the statements.

However, just as machine languages are not the same, tabular techniques are not all the same. A number of people have contributed to these developments, and each person (or group) has followed a somewhat different path. Most of the developments have been limited to the particular application for which they were intended and have not been generalized.

This report serves to record in one place what has transpired during the past two or three years in the development of tabular techniques, and attempts to express major features and differences between various techniques.

Orren Evans, of Hunt Foods and Industries, first published work on using tabular form for computer programming. He had gained experience in the use of tabular form in his work with Sutherland and Company. The Hunt Foods material was released in December of 1959 to CODASYL, and later was presented at a Guide meeting and to the NMAA; it has been published by IBM as a General Information Manual. The decision structure tables are of a "limited entry" variety; this means that a complete condition or action statement is made in the stub (argument) of the table while the tabular entries only make assertions concerning the truth or falsity of the condition or indicate whether an action should be executed.

The Evans work corresponds in many ways to the Sutherland material. Copies of a current Sutherland proposal will be made available. Sutherland has prepared a number of tables describing a particular customer's decision rules; a 7070 program is being written from these tables instead of from flow charts. The Sutherland tables are still of the limited entry variety, though they are somewhat simpler in structure than the Hunt Foods' tables.

The CODASYL Systems Committee, prompted by the Hunt Foods work, has also decided to exploit tabular form to provide a systems-oriented language. I am a member of this Committee which also includes Les Calkins of U. S. Steel, Jack Strong of North American Aviation, Carl Byham of Southern Railway, Sol Pollack of RAND and some 8 - 10 others including representatives of RCA, Remington-Rand, and GE. The work which has been published to date in various intra-committee reports describes tabular form, data description, and certain systems-level operators. The tabular form material incorporates the limited entry approach of Hunt Foods, but also takes care of "extended entry" tables like those developed at General Electric: the table entries contain actual values, names or functions.

The General Electric work was initiated by the Integrated Systems Project in their Manufacturing Services Division (a staff group). As part of the major project aimed at designing an automatic factory, the need for describing complex, sequential, decision rules led this group to the creation and use of decision structure tables.

For variables (named fields) which have many values (more than two), the extended entry approach offers certain advantages; it is still quite easy to teach and relatively easy to implement. In contrast, the limited entry table may have substantial advantages for problems involving primarily two state variables. Up to recently, General Electric had not been willing to release any of the material which they have developed. However, at the last CODASYL meeting (12/1/60) Charlie Katz and Don Klick of General Electric's Computer Department, presented a paper, "Preliminary Reference Manual, TABSOL - 225 - A Tabular Systems Oriented Language for the GE 225 Information Processing System". This paper proposes a complete and quite comprehensive tabular form language which is to be directly processed on a GE 225. I should like to quote briefly from the introduction to this manual:

"Recent investigations by The Integrated Systems Project of General Electric's Manufacturing Services uncovered an area of applications which require neither extensive data file processing nor profound mathematics, but rather an unwieldy number of sequential decisions. To cope effectively with these decisions, the ISP team devised a tabular language. The purpose of this language was to depict, by means of tables, the relationships of logical decisions... Since its creation, TABSOL has been used in many departments of G. E. to analyze and solve problems of product engineering, manufacturing methods, cost accounting, and production control. The application of decision tables is continually growing. Recent studies show that they provide a concise method for supporting the logic of other data processing applications. For example, decision tables may be used to specify the transfer of control associated with the values of one or more fields, to control the printing of detail and summary lines of a report, or to interrogate the sort keys in a multi-file system. At the Computer Department we have found decision tables a valuable tool in designing and implementing the General Compiler."

"Decision tables represent a third language for the General Compiler. These may be used by themselves or in conjunction with the features of the compiler language. The specifications outlined in this manual pertain mainly to the table entries and imply and require a knowledge of the General Compiler ..."

General Electric has also permitted Mr. T. F. Kavanagh, who worked on the Integrated Systems Project, to present a paper entitled, "The TABSOL Concept" at the Eastern Joint Computer Conference on December 14, 1960. It is known that General Electric has probably thirty different departments (out of a total of 100) actually involved in experimenting on the practical use of decision tables. The specific experience of the ISP team is such as to indicate that the use of tables could save significant time in the programming and debugging of decision rules. Work in General Electric up through 1959 involved the preparation and use of interpreters for the 702, 704, 650, and 305 RAMAC. It would be reasonable to assume that in 1960, work on the NCR 304 would have progressed far enough to have a processor available, and there may be processors for other machines such as the Burroughs 205.

As a result of the CODASYL work, IBM was requested by North American Aviation to support their development work on tabular form. P. W. Knaplund, then Manager, Systems Marketing, DP Division, obtained the half-time services of M. D. Rayner who was assigned by R. V. Woodworth, then of the Inglewood office. Mr. Rayner is spending the other half of his time working with Northrop (Norair Division) on the development of another form of tables involving variable operation sequence. Neither of these programs are far enough along yet to have formal reports available.

W. M. Selden of IBM Corporate Systems Standards has been located at Rochester to work with Eastman Kodak in testing and developing concepts in the use of tables. Specifically, there are three projects either under way or ready to start there, with Eastman Kodak providing the bulk of the experimental work. One project has to do with the presentation of production control rules in their camera division. The second project has to do with the validation and updating of files in the Data Processing group. The third project is concerned with quality control decisions.

UNIFIED SYSTEMS APPROACH

CLEARINGHOUSE REPORT

TABSOL

A FUNDAMENTAL CONCEPT FOR SYSTEMS ORIENTED LANGUAGES

Text of Speech Presented at E J C C
December 14, 1960

January 13, 1961
Ref. No. 1A1

T. F. Kavanagh
Manufacturing Services
General Electric Company

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

TABSOL

A Fundamental Concept for Systems Oriented Languages

T. F. Kavanagh
Manufacturing Services
General Electric Company

Bulging file cabinets, the flow chart jungle, mounting clerical costs, and the vast world where electronic computers haven't been successfully applied -- that's really what TABSOL and decision structure tables are all about. Structure tables have special meaning for information systems designers and programmers and they also have implications for hardware engineers because both computer user and computer designer must work together on the same information processing problems.

To date, the difficulties of communicating with electronic computers have received much attention. The various pseudo-languages represent great advances in this area, but a language is a great deal more than the basic tool of communication. A good language, -- a good symbology, -- is an essential element in man's thought processes. In a sense it defines his capacity for conceptualization and for abstract thought. It's no mystery that the telephone wasn't invented in Tahiti or the airplane in Afghanistan. Today we face a similar language restriction in trying to analyze and think about the complex decision-making systems required to operate a business or control an industrial process. Our traditional techniques seem inadequate. Flow charts quickly become a puzzle of lines, balloons, and boxes whose secret lies hidden in the mind of the creator. Frequently, programmers complain they would rather reprogram the job than take over someone else's flow charts.

In addition to flow charts, you often see matrix- type displays. They appear under a variety of names--collation charts, tabulated drawings, standard time data sheets, and so on. Often large and unwieldy, they usually represent listings of past decisions or answers rather than the logic used in making them. But none of these methods for thinking about and communicating complex decision logic have been really effective. Most business and professional people still communicate with the computer world through an elaborate hierarchy of flow charters and programmers. This is the problem which we feel TABSOL greatly simplifies. It combines some of the characteristics of earlier methods and introduces a few new features of tis own. After using TABSOL's decision structure tables in numerous applications, we feel they are both good communicating tools, and also valuable thinking tools. As one G.E. computer wag put it:

TABSOL is a thinking man's language.

Decision structure tables provide a standard, uniform methods for clearly describing complex, multi-variable multi-result decision systems.

A structure table consists of a rectangular array of terms, sub-divided into four quadrants. The vertical double line separates the decision logic on the left from the result functions or actions which appear on the right. A horizontal double line separates the structure table column headings above from the table values recorded in the horizontal rows below. Thus, the upper left quadrant records the names of the parameters effecting the decision while the lower left quadrant records the specific values which a decision parameter may have in a given situation. Similarly, the upper right hand quadrant records the names of result functions -- or actions to be performed -- once the decision has been made, and the lower right quadrant shows the actual result values which pertain directly opposite the appropriate set of decision parameter values. Thus, each horizontal row completely and independently describes one possible decision situation. Each structure table becomes a complete statement of the logical and quantitative relationships supporting a particular elementary decision.

There is no limit to the structure table columns or rows. The dimensions of any specific structure table are completely flexible, and are a logical consequence of the decision being described. A series of these structure tables taken in combination will describe a complete decision system.

Now let's look at a simple example (figure 1). Here we want to make an elementary decision on transportation from New York to Boston. There are three significant decision parameters: Weather, Plane Space, and Hotel Room. Weather has only two value states, Fair or Foul; Plane Space is either OK or Sorry; and Hotel Room can be either Open or Filled. In terms of results, Plane or Train are the only permissible means of Transportation. If the weather was Foul, despite an OK on plane space and an Open, Hotel Room, then we see by inspection that the solution appears in the second row. Train is the correct Transportation. We are also instructed to Cancel Plane, and this is the End of the decision.

This simple structure table provides a general solution to this particular decision-making problem. If afternoon trips to Boston ever occur -- and one must assume that they frequently do -- then an operating decision can quickly be made by supplying the current value of Weather, Plane Space, and Hotel Room and solving the structure table.

Solving a structure table consists of comparing or "testing" specific values assigned decision parameters in the problem statement against the corresponding sets of decision parameter values recorded in the structure table. If all tests in a row are satisfied, then the solution is in that row. The correct result values or actions appear in the same row, to the right of the double line.

Once a particular structure table has been solved and the result functions executed, it is often necessary to make more decisions. For this reason, the last result column of the structure table provides a firm link to the next decision structure table. Notice the last row specifies that for all values of Weather, with no Plane Space, and no Hotel Room, the decision-maker is directed to solve another structure table, Transportation, New York-Boston in the morning.

Similarly, a system designer can build a whole system of structure tables. He completely controls the make-up of each table, as well as its position in the sequence of total problem solution. He may decide to skip tables, or, he may re-solve tables to achieve the effect of iteration.

Getting from New York to Boston is a rather prosaic problem to say the least; we certainly don't need a computer to make decisions like this.

So let's look at how a systems designer might structure a real operating decision.

Table 2015 (figure 2) completely describes time standards determination for a certain coil winding operation. In this situation if the number of turns is less than 10, the operator's time allowance in seconds is equal to the number of turns. However, if the number of turns is greater than 10 but less than 15, the operator is allowed an additional 88 hundredths of a second.

So you see that problem values and decision parameter test values need not be simple identities. Actually, the problem values may be equal to the table value, greater than, less than, not equal to, greater than or equal to, less than or equal to the test value. This broad selection of test types greatly increases the power of individual structure tables and sharply reduces their size. Note than we can put the test type right in the test block immediately proceding the test value, or in the column heading after the decision parameter name. Of course, the test type in the column heading applies to all test values appearing below. It is also possible to formulate complex test blocks involving two or more decision parameters.

Structure table results are not limited to simple assignments of alphabetic or numeric constants. As we've already seen if the solution occurs in the first row, the current value of TURNS is assigned TIME. If the solution occurs in the second row, the result of the arithmetic expression TURNS $\neq$ 0.88 is assigned TIME. If the solution occurs in the third or fourth row the result of the formula evaluation TIME 1 or TIME 2 will be assigned to TIME. This is the significance of the equal sign, appearing after the name in the result value block. These formulas are recorded in the area just prior to the structure table proper.

In the next action column the result function PERFORM appears. This means that the data processing or arithmetic operations named in the result value block are to be executed. Notice that one of the result values is another structure table. Should the solution occur in this row, Table 2016 will be solved just as any other, only control will remain within the framework of Table 2015 which is our illustrative table. When completed, the next result function will be processed. In the next column, the result function GO links this structure table to the next structure table to be solved. If there is no solution row found in the structure table proper, then control passes to the area directly below the structure table. This is usually regarded as an "error", and most often indicates a failure of the decision logic to cope with a certain combination of problem values. The systems designer can -- and should -- notify himself whenever such an error occurs by arranging for an error printout, identifying the table that failed and the problem being solved at the time. With this source language printout and other structure tables, the systems designer has all the data he needs to trouble-shoot the system in his own professional terminology.

We can also use the areas immediately before and after the structure table proper to record any additional language statements that may be required -- input output operations, data movement, operator instructions or any other data processing activities.

Of course, I cannot even attempt to completely describe decision structure tables in this short talk; a much more complete explanation appears in the Proceedings. There are many more features available for formulating concise, complete decision systems. I can only give a quick introduction to inherent Gestalt in this method of describing decision logic.

Structure tables did not start at their present state of development. This language concept evolved through a series of experimental tabular systems-oriented languages developed for the 305, 650, 702 and 704.

These experimental languages proved remarkably adequate; however, the added power of a conventional language seemed very appealing, particularly as the prospects for structure table application in all sorts of problem areas brightened.

At this point, General Electric's Computer Department joined the effort. The Computer Department had been developing a new compiler, called GECOM, for use with General Electric computers. The first version of this new General Compiler, will be avilable for the GE 225 in May, 1961. It has been designed primarily around COBOL, with some of the basic elements of ALGOL. It will now contain all of TABSOL. Simply stated, joining TABSOL with GECOM places the power of a full-fledged conventional language at the command of every structure table block.

We now have a rather substantial amount of experience in applying structure tables to a wide variety of operating decision-making problems. But perhaps the most interesting, at least from the researcher's point of view, was the very work which led to decision structure tables themselves. In 1957 we were investigating the possibility of automating the essential information and material processing required to directly transform customer orders into finished products. We studied customer order editing, product engineering, drafting, manufacturing methods, and time standards, quality control, cost accounting, and production control. This accounts for a fairly substantial portion of the operating decision system in a manufacturing business. Fortunately, the inputs and outputs to this system are simple and well-defined: the customer order comes in and the finished product goes out. So it was possible to treat all activities within these bounds as one integrated, goal-oriented operating decision system and develop decision structure tables accordingly. Working with a small product section in one of the Company's operating components, a significant portion of the functional decision logic was successfully structured. Then the resulting structure tables were directly incorporated into a computer-automated operating decision system which transformed customer orders for a wide variety of finished products directly into factory instructions for operators and numerically programmed machine tools. This prototype system was demonstrated to General Electric management in November, 1958. Since then, structure tables have been used to describe the operating decision logic in many different applications. Structure tables appear to have great potential in compilers and also in computer simulation programs.

As a result of these efforts, we have come to believe that decision structure tables are broadly applicable to nearly all classes of information processing and decision-making problems because:

1. Structure tables for a disciplined decision analysis. The precise structure table format highlights illogical statements and emphasizes the reasons why results are different.

2. They are easy to understand. The structure table format is so simple and straight-forward that engineers, planners, and other functional specialists can write structure tables for their own decision-making problems with very little training. They provide an excellent basis for program documentation and communication.

3. Debugging simplified. Structure table errors can be reported at the source language level, thus permitting the functional specialist to debug without a knowledge of computer coding.

4. And structure tables are easy to maintain. Instead of changing all the precalculated answers in all the files, it is often only necessary to change a single value in a single table. Structure tables solved automatically in an electronic computer offer levels of accuracy unequalled in manual systems.

This discussion encompasses the efforts of over seventy-five men and women representing five Service Components and some fifteen different Operating Components within General Electric. In particular, credit is due Burt Grad, who though no longer with General Electric, was a principal originator of the decision structure table concept. Also Mal Boggs, Charlie Katz, Dan Langenwalter, Herb Nidenberg, and Ted Schultz and many others.

The best way to understand TABSOL is to try it yourself. Seriously, let me recommend that you demonstrate the effectiveness of decision structure tables to yourself by "structuring" a few simple decisions. You might write a structure table to help your wife to decide how to pack your suitcase for a business trip. Frankly, if you will only take the time to "structure" a few decisions and actually experience the deeper insight and clarity which this technique provides, then decision structure tables will speak for themselves.

December, 1960,

Problem Statement: Select Transportation, New York - Boston, p. m.

Weather: Foul

Plane Space: OK

Hotel Room: Open

Decision Structure Table: Transportation, New York - Boston, p. m.

Weather	Plane Space	Hotel Room	Trans-portation	Other In-structions	Next Decision
Fair	OK	Open	Plane		End
Foul	OK	Open	Train	Cancel Plane	End
	Sorry	Open	Train		End
	OK	Filled		Cancel Plane	NY-Bost. a. m.
	Sorry	Filled			NY-Bost. a. m.

Solution:

If the value of Weather is Foul, and

the value of Plane Space is OK, and

the value of Hotel Room is Open,

Then

the value of Transportation is Train, and

the value of Other Instructions is Cancel Plane, and

the value of Next Decision is End.

Figure 1

```
TABLE 2015.  DIMENSION  C2 A3 R4.
NOTE TIME STANDARDS FOR COIL WINDING.
TIME~1  =  125*DIA*TURNS.
TIME~2  =  1000*DIA/SQRT (TURNS).
BEGIN
```

TURNS	TURNS LS	TIME	PERFORM	GO
LS 10		TURNS		TABLE 2020
GREQ 10	15	TURNS + 0. 88	SETUP	TABLE 2025
GREQ 15	100	TIME~1 =	SETUP	TABLE 2025
GREQ 100	1000	TIME~2 =	TABLE 2016	TABLE 2030

```
IF NOT SOLVED GO ERROR~COIL.
END TABLE 2015.
```

Figure 2

DP Systems Engineering Services
200 Mamaroneck Avenue
White Plains, N. Y.

February 15, 1961

Memorandum to:

Subject: Tabular Techniques Development
Distribution #2

This is the second release of material concerning the development of tabular techniques for systems and programming description. Enclosed are two items:

(1) A working paper by Mr. Earl Althoff of Eastman-Kodak describing a tabular approach to a file updating problem.

(2) A preliminary report on TABSOL 225 by Mr. D. Klick of General Electric's Computer Department. This paper was given at the CODASYL Systems Committee meeting in December, 1960.

Reference is also made to a third item which is available through IBM Stationary Stores in Endicott and hence not attached:

(3) General Information Manual "Advanced Analysis Method for Integrated Electronic Data Processing" by Mr. Orren Y. Evans of Hunt Foods & Industries. This is Report No. F20-8047.

Burton Grad

Burton Grad
Project Coordinator

SYSTEMS
ENGINEERING
SERVICES

CLEARINGHOUSE REPORT

A PRELIMINARY APPROACH

TO

TABULAR PROGRAMMING

February 1, 1961
Ref. No. 1B1

E. O. Althoff
Data Processing Service
Eastman - Kodak

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

POINTS ABOUT PRELIMINARY APPROACH TO TABULAR PROGRAMMING

1. For each element used, prepare a 15-digit title to use in the English text and a four-digit abbreviation to use in formulae. The four-digit abbreviation either starts with a letter or is numbered sequentially 0001, 0002, 0003, ...

2. Do not strain to over-abbreviate. For example, CT01, CT02, ... can be used to stand for control totals of various types. It is usually best to give mnemonic abbreviations only for the hundred or less most used elements.

3. Data sets can be listed on a data element sheet if desired. For example, the data set "Target Date" abbreviated TRGT consists of the data elements:

"Target Month"	abbreviated TMON
"Target Day"	abbreviated TDAY
"Target Year"	abbreviated TYR

   In the above, four entries are made, one for each data element and one for the data set.

4. The definition should be clear and unambiguous, but above all must be complete. Differentiate clearly between similar data elements.

5. Prepare a data file for each set of data (not going directly to a report). Do not consider the machine in your preparation. As an example consider a tape with records of Type A followed by several records of Type B; prepare two data files, A and B, since having these on the same tape is pure machine method.

6. For each data set or element listed, record a reference to set number and page number of the data element sheets. Thus, 03-01 refers to data element set 03, page 01. Record both the title and the abbreviation. Record the length for that file. A given element can require four digits on one data file and six on another.

7. Give each data file a letter designation A, B, C, .., record whether input or output. In the case of an updated data file, assign two letters, say A for input and B for output.

8. Obtain a Data Processing Spacing Chart for all report lines (messages as well as fancy reports). Label each report A, B, etc. Use a second letter for each different type of line. Thus, Report A may have lines AA, AB, AC, ...

9. Tables must give all logic except how to start and how to stop. All the statements which follow must be accomplished completely — no exceptions can be permitted.

10. The table is divided into conditions and actions. On the left, one gives the English statement of the condition or action, and on the right, one records the precise formula fully and completely. Thus, on the right;

    A-STAT = B-STAT clearly shows that the condition is <u>true</u> if and only if the STAT element of data File A equals the STAT element of data File B.

    C-0001 $>$ 0 shows the condition is true only if the data element 0001 of data File C is greater than the constant 0.

11. The formula for a condition can include any connectors desired to complete "a single condition". Examples are:

    F-TAX = 10 or 15
    F-TAX = 10 and G-TAX = 15

12. The actions can be varied also. In general, one records data movement or arithmetic actions first, then all data file advance actions, then all table transfer actions.

13. Typical data movement actions are D → E (ASG#, PROG, and TT01) meaning move from data File D to data File E the data elements ASG#, PROG, and TT01. In case of one move, D-ASG# → E-ASG#. Others are D-PDHR add to E-YRHR, etc.

14. When data is posted to report lines, increment is used as: D-ASG# → AB14 meaning post data element ASG# of data File D to position 14 (right-hand increment) of line B of Report A.

15. Another action may be to do an action or actions from other tables. Thus, Action 2 of Table 01-A5 can simply be "Do actions 3 and 6 of Table 03-B7.

16. Another action may follow the actions for a data rule from another table in its entirety; if so, simply transfer to Rule XX of Table XX-XX.

17. The advance data files actions are abbreviated GIV X for input and TAK X for output. In some cases posting a data element to a control total is included as: C-AMT add to TOT1; TAK C. When an advance action is given, the next action calling on that file from <u>any</u> table will be from the next record.

18. Tables are numbered NN-XN where NN denotes the project area; NN runs from 01, 02, ,., X is a letter denoting a sub-project and runs from A to Z, while the rightmost N is 1 to 9 and denotes table within sub-project. The last action for any data rule is always a transfer to some table. (Do not transfer to a rule within a table — leave this to the programmer.)

19. On any given table, possible entries opposite condition are Y, N, and -. Y = Yes, N = No, and - means "does not apply".

    The matrix $\begin{array}{|c|c|c|}\hline Y&Y&N\\\hline Y&N&N\\\hline\end{array}$ would indicate an analyst omission since the combination $\begin{array}{|c|}\hline N\\\hline Y\\\hline\end{array}$ is not specified. One must specify enough data rules to account for every combination of the conditions, whether <u>possible</u> or <u>impossible</u> (is it really impossible???)

    The - is used primarily in two cases:

    A. If condition 1 is A-STAT = 0 and condition 2 is A-STAT = 1, then a $\begin{array}{|c|c|}\hline Y&-\\\hline -&Y\\\hline\end{array}$ entry would show that, if A-STAT = 0, we don't need to test for A-STAT = 1 and vice versa.

    B. The - may be used to indicate a plain transfer to another table, when the only alternative would be to over-run the 6 X 10 matrix. Example:

Condition 1	Y	Y	Y	Y	Y	Y	Y	Y	N
Condition 2	Y	Y	Y	N	Y	N	N	N	-
Condition 3	Y	Y	N	Y	N	Y	N	N	-
Condition 4	Y	N	Y	Y	N	N	Y	N	-

The data rule on the right simply transfers to another table where the NO's for Condition 1 are spelled out.

20. In summary, the preliminary approach is designed to obtain from a job analyst actions to follow for every combination of conditions. The conditions and actions are not to be vague — but must be 100 percent precise to every data element involved. There is no thought given in the preliminary approach to automating any of the steps: tables $\rightarrow$ programs. Only a person with two - three years active programming and computer systems experience can prepare tables containing many subtle traps which develop only in automatic E.D.P. systems; for the next year or so, it is expected that these people will return expanded tables (with these subtle points included) to the job analyst and will, in addition, write programs in KodaKoder.

EOAlthoff:rds
October 11, 1960

DATA PROCESSING SERVICE

ELEMENTS DEFINITION

SET # 01 PAGE #01

Job No.

Name: Earl O. Althoff

Project: D.P.S. Billing

·DIGIT TITLE One Letter ASGN — 4-DIGIT ABBREV. ASGL

DEFINITION: A letter code used to differentiate between several types of perpetual assignments. Refer to the back of a D.P.S. Time-Reporting Sheet for full details.

15-DIGIT TITLE Assignment No. — 4-DIGIT ABBREV. ASG#

DEFINITION: A four-digit number given in sequence to non-perpetual assignments as they occur. The number has no structure of any sort.

15-DIGIT TITLE Billing Number — 4-DIGIT ABBREV. BIL#

DEFINITION: A five-digit number assigned by the D.P.S. Accountant to each account or sub-account which D.P.S. bills. It is structured as desired to yield a meaningful report order.

15-DIGIT TITLE PROG-SYSTEM NO. (DATA SET) — 4-DIGIT ABBREV. UJ#

DEFINITION: A uniform job number which serves a variety of purposes. It is organized primarily r computer run and program within computer run. (See Chapter 27 - Self-Teach.) Consists of elements MFC, RUN#, and PRGL.

15-DIGIT TITLE Project Title — 4-DIGIT ABBREV. TITL

DEFINITION· A 45-character title given to each project having a four-digit assignment number.

15-DIGIT TITLE Project Type Code — 4-DIGIT ABBREV. TYPE

DEFINITION: A two-character code enabling us to group a project by new programs (N), changes (C), or revision (R). The units position is 1 for a business project, 6 for a program research project.

15-DIGIT TITLE Major Fctn. Code — 4-DIGIT ABBREV. MFC

DEFINITION: A two-digit code used by D.P.S. to roughly distinguish between basic major project functions such as Merchandise Billing, Paper Finishing Scheduling, etc. It is the first two digits of Prog-System No.

1ꜰ·DIGIT TITLE Target Date — 4-DIGIT ABBREV. TRGT

INITION: The date by which an assignment should be completed to the point that production results are obtainable. Six digits as 011560.

D A T A   P R O C E S S I N G   S E R V I C E

ELEMENTS DEFINITION

SET # 01 PAGE # 02

Job No.:

Name: Earl Althoff

Project: D.P.S. Billing

DIGIT TITLE Programmer 4-DIGIT ABBREV. PROG

DEFINITION: An official ten-digit name assigned to each individual of the Programming and Methods Staff

15-DIGIT TITLE Registration # 4-DIGIT ABBREV. REG#

DEFINITION: A six-digit number given to each employee of Kodak Rochester. The first three digits indicate department and the last three are sequentially given by various rules.

15-DIGIT TITLE Prog-Syst.Descr. 4-DIGIT ABBREV. DESC

DEFINITION: Refers to a 29-digit alphanumeric title or description given to each specific program or computer systems sub-assignment.

15-DIGIT TITLE EST. Man Months 4-DIGIT ABBREV. ESTM

DEFINITION: Refers to a time estimate given for each program in an assignment. The time is ven in four digits (one decimal place).

15-DIGIT TITLE Due Date for V 4-DIGIT ABBREV. DUEV

DEFINITION: A date given for each program to be ready for system volume testing. Six digits as 011560 or 12B161.

15-DIGIT TITLE Department 4-DIGIT ABBREV. DEPT

DEFINITION: A four-character alphanumeric abbreviation of the department a programmer belongs to. Examples are DPS, MSDD, A&O, DC.

15-DIGIT TITLE Computer Run # 4-DIGIT ABBREV. RUN#

DEFINITION: The third and fourth digit of Prog-System No.. Delineates the programs consititutin a scheduled computer run.

DIGIT TITLE Program Letter 4-DIGIT ABBREV. PRGL

INITION: The fifth digit of Prog-System No. Letters from A to Z are given to programs of a given computer run.

DATA PROCESSING SERVICE

DATA FILE LAYOUT

FILE DESCRIPTION Assignment Master

Job No.:

Project: D.P.S. Billing

Data File: A in B out

Name: Earl Althoff

TITLE	REF.	ABBREV.	LENGTH	For Programmer Use Onl REC #	INCR.
1. Assignment No.	01-01	ASG#	4		
2. Billing Number	01-01	BIL#	5		
3. Proj. Ldr. Name	01-05	PLDR	10		
4. Project Title	01-01	TITL	45		
5. Proj. Type Code	01-01	TYPE	2		
6. Major FCTN Code	01-01	MFC	2		
7. Target Date	01-01	TRGT	6		
8. Bill-out Code:	01-01	BILC	1		
9. Completion Code	01-03	CMPL	1		
10.					
11.					
12.					
13.					
14					
15.					
16.					
17.					
18.					
19.					
20.					
21.					
22.					
23.					
24.					
25.					
26.					
27.					
28.					
29.					
30.					
31.					
32					
33.					
34.					
35.					

D A T A P R O C E S S I N G S E R V I C E
DATA FILE LAYOUT

FILE DESCRIPTION Current Time Records

Job No.:
Project: D.P.S. Billing
Data File: C in
Name: Earl Althoff

	TITLE	REF.	ABBREV.	LENGTH	REC #	INCR.
					For Programmer Use Onl	
1.	Assignment No.	01-01	ASG#	4		
2.	Prog-System No.	01-01	UJ#	5		
3.	Programmer	01-02	PROG	10		
4.	Department	01-02	DEPT	4		
5.	Progress Code	01-03	STAT	1		
6.	Est. Date for V	01-03	ESTV	6		
7.	Hrs. This Period	01-03	HRTP	4		
8.	CPU MIN V-Test	01-05	VTST	4		
9.	K-10S This Pd.	01-04	K10P	2		
10.	K-20S This Pd.	01-04	K20P	2		
11.	K-30S This Pd.	01-03	K30P	2		
12.	K-40S This Pd.	01-04	K40P	2		
13.	K-50S This Pd.	01-04	K50P	2		
1	ASGL can be M and N only	01-01	ASGL	1		
5.						
6.						
7.						
8.						
9.						
0.						
1.						
2.						
3.						
4.						
5.						
6.						
7.						
8.						
9.						
0.						
1.						
3.						
4.						
5.						

DATA PROCESSING SERVICE
DATA FILE LAYOUT

FILE DESCRIPTION Program System Master

Job No.:
Project: D.P.S. Billing
Data File: K in L out
Name: Earl Althoff

	TITLE	REF.	ABBREV.	LENGTH	For Programmer Use Only REC #	INCR.
1.	Assignment No.	01-01	ASG#	4		
2.	Prog-System No.	01-01	UJ#	5		
3.	Programmer	01-02	PROG	10		
4.	Prog-System Description	01-02	DESC	29		
5.	Est. Man Months	01-02	ESTM	4		
6.	Due Date for V	01-02	DUEV	6		
7.	Department	01-02	DEPT	4		
8.	Progress Code	01-03	STAT	1		
9.	Est. Date for V	01-03	ESTV	6		
10.	HRS to Date	01-03	HFTD	5		
11.	Bill-out Code	01-03	BILC	1		
12.	V-TEST To Date	01-05	VTTD	6		
13.	K-10S To Date	01-04	K10S	3		
14.	K-20S To Date	01-05	K20S	3		
15.	K-30S to Date	01-04	K30S	3		
16.	K-40S To Date	01-04	K40S	3		
17.	K-50S To Date	01-04	K50S	3		
18.	ASGL will be M and N only	01-01	ASGL	1		
19.						
20.						
21.						
22.						
23.						
24.						
25.						
26.						
27.						
28.						
29.						
30.						
31.						
32.						
33.						
34.						
35.						

DATA PROCESSING SERVICE
TABLE LAYOUT

TITLE Update Assignment Master

Job. No.:
Name: Earl Althoff
Project: D.P.S. Billing
Table No: 01-A1

CONDITIONS	FREQUENCY / RULE NO.	1	2	3	4	5	6	7	8	9	10	CONDITION ABBREVIATIO
	FREQUENCY	0	10	1	10	0	200					
	RULE NO.	1	2	3	4	5	6	7	8	9	10	
1. Is there a new assignment master?		Y	Y	N	N	N	N					D-ASG# < A-ASG#
2. Is there a change to the assignment master?		-	-	Y	Y	Y	N					D-ASG# = A-ASG#
3. Is the change record a deletion?		Y	N	Y	-	-	-					D-CMPL = 2
4. Is the change record a completion notification?		-	-	-	Y	-	-					D-CMPL = 1
5. Are we posting changes to a completed assignment?		-	-	-	-	Y	-					D-CMPL = A
6.												
ACTIONS												
1. Move entire master record to serve as base.					Y		Y					A → B
2. Post change assignment no. to updated master.			Y	Y		Y						D-ASG# → B-ASG#
3. Post corresponding parts of change to master.			Y			Y						BIL#, FLDR, TITL, TYPE, MFC, TRGT, BILC D → B
4. Set-up and write delete error message.		Y										D-ASG# → AA-39 Write AA
5. Post completion code to master			Y		Y	Y						D-CMPL (Numeric portion) → B-CMPL
6. Set-up and write deleting message				Y								D-ASG# → AB-18 Write AB
7. Advance controls to next change		Y	Y	Y	Y	Y						GIV D
8. Advance controls to next input master				Y	Y	Y	Y					GIV A
9. Transfer to Table 01-A1		Y										TR 01-A1
10. Transfer to Table 01-A3				Y								TR 01-A3
11. Transfer to Table 01-A2			Y		Y	Y						TR 01-A2
12. Transfer to Table 01-A4							Y					TR 01-A4
13.												

DATA PROCESSING SERVICE
TABLE LAYOUT

TITLE Second Change Check

Job. No.:
Name: Earl Althoff
Project: D.P.S. Billing
Table No: 01-A2

CONDITIONS	FREQUENCY	0	0	0	10							CONDITION
	RULE NO.	1	2	3	4	5	6	7	8	9	10	ABBREVIATION
1. Is there a change to the updated master?		Y	Y	Y	N							D-ASG# = B-ASG#
2. Is this change a deletion?		Y	-	-	-							D-CMPL = 2
3. Is this change a completion notification?		-	Y	-	-							D-CMPL = 1
4. Are we posting changes to a completed assignment?		-	-	Y	-							D-CMPL = A
5.												
6.												
ACTIONS												
1. Follow action of Rule 3 Table 01-A1		Y										TR Rule 3 - 01A1
2. Post completion code to master.			Y									D-CMPL (numeric part) → B-CMPL
3. Advance controls to next change.			Y	Y								GIV D
4. Follow actions 3 and 5 of Table 01-A1				Y								Duplicate 3 and 5 of 01-A1
5. Transfer to Table 01-A2			Y	Y								TR 01-A2
6. Transfer to Table 01-A4					Y							TR 01-A4
7.												
8.												
9.												
10.												
11.												
12.												
13.												

D A T A   P R O C E S S I N G   S E R V I C E
TABLE LAYOUT

TITLE Update Prog-System Master

Job. No.:
Name: Earl Althoff
Project: D.P.S. Billing
Table No: 01-B1

C O N D I T I O N S	FREQUENCY	0	100	10	1	10	2000	200				C O N D I T I O N
	RULE NO.	1	2	3	4	5	6	7	8	9	10	A B B R E V I A T I O
1. Is there a new Prog-System Master?		Y	Y	Y	N	N	N	N				E-FLD1 < K-FLD1
2. Is the new master for this assignment?		Y	Y	N	-	-	-	-				E-ASG# = B-ASG#
3. Is this a delete?		Y	N	-	-	-	-	-				E-DPGM = 1
4. Is the next master for this assignment?		-	-	-	Y	Y	Y	N				K-ASG# = B-ASG#
5. Is there a change for the master?		-	-	-	Y	Y	N	N				E-FLD1 = K-FLD1
6. Is the change a delete?		-	-	-	Y	N	N	N				E-DPGM = 1
A C T I O N S												
1. Move entire master to serve as base.					Y	Y	Y					K → L
2 Post corresponding change fields to master.			Y			Y						E→L (ASG#, UJ#, PROG, DESC, ESTM, DUEV, DEPT, BILC
3. Post start-up constants to master.			Y									Blanks to STAT, ESTV Zero to HRTD, VTTD, K10S, K20S, K30S, K40S, K50S
4. Set-up and write delete error message.		Y										E-ASG# →AF39 E-UJ# → AF48 Write AF
5. Set-up and write deleting message.					Y							E-ASG# →AG19 E-UJ# → AG29
6. Advance control to next change record.		Y	Y		Y	Y						GIV E
7. Advance control to next master record.					Y	Y	Y					GIV K
8. Transfer to this Table 01-B1		Y										TR 01-B1
9. TRansfer to Table 01-B3			Y			Y						TR 01-B3
10. TRansfer to Table 01-C1				Y				Y				TR 01-C1
1 TRansfer to Table 01-B2					Y							TR 01-B2
12. TRansfer to Table 01-B4							Y					TR 01-B4
13.												

DATA PROCESSING SERVICE
TABLE LAYOUT

TITLE Finish deleting on Prog-Syst. Delete.

Job. No.:
Name: Earl Althoff
Project: D.P.S. Billing
Table No: 01-B2

CONDITIONS	FREQUENCY / RULE NO.	1	2	3	4	5	6	7	8	9	10	CONDITION ABBREVIATION
1. Are there any more Prog-Syst. changes?		Y	N	N								E-FLD1 = L-FLD-1
2. Is there a current time record?		-	Y	N								C-FLD1 = L-FLD1
3.												
4.												
5.												
6.												
ACTIONS												
1. Set-up and write message AC.		Y										E-UJ#→AC30 Write AC
2. Set corresponding to delete message.			Y									Action 6 of table 01-A3
3. Set rest of corresponding and write delete message.			Y									Action 7 of table 01-A3
4. Advance controls to next Prog-Syst. Change.		Y										GIV E
5. Advance Controls to next time record.			Y									GIV K
6. TRansfer to Table 01-B2		Y										TR 01-B2
7. TRansfer to Table 01-B1			Y	Y								TR 01-B1
8.												
9.												
10.												
11.												
12.												
13.												

D A T A   P R O C E S S I N G   S E R V I C E

ELEMENTS DEFINITION

SET # ____ PAGE # ____

Job No.: ____

Name: ____

Project: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

15-DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____

DIGIT TITLE ____ 4-DIGIT ABBREV. ____

DEFINITION: ____