COMPOSE / PRODUCE: a user-oriented report generator capability within the SDC time-shared data management system

by WILLIAM D. WILLIAMS*
Shell Oil Company
New York, New York

and

PHILIP R. BARTRAM
System Development Corporation
Santa Monica, California

INTRODUCTION

At present there are numerous report program generators, most of which are quite similar in purpose and in method of operation. Although they generally relieve the programmer from concern with detailed program logic, they nevertheless require him to have intimate knowledge of the characteristics of his data base, the kinds of entries he wants extracted, the manipulations to be performed, and the detailed format of the desired report. Furthermore, as report program generators have become more generalized (and thus more flexible), they have also become more complex, requiring the user to furnish complicated specification statements in coded form. This, of course, demands an increased amount of training and experience on the part of the programmer, and makes his source programs more subject to error.

In an attempt to overcome some of these difficulties, System Development Corporation is developing a user-oriented report program generator as part of the Time-Shared Data Management System (TDMS). As shown in Figure 1, the report generator function within TDMS is a two-phase program: the first phase, called "COMPOSE," is used to design the report on-line; the second phase, called "PRODUCE," actually produces the report.

TDMS is a general-purpose file management system operating under the SDC time-sharing executive

*Research Associate at System Development Corporation
functions to be provided by TDMS include the ability to describe and load a data base, maintain the data base, and retrieve data in response to human query.

This paper discusses the design features of on-line report program generators, and gives several examples of the operation of COMPOSE and PRODUCE. A more detailed discussion of the logical structure of a TDMS data base was given by Emory W. Franks at the Spring Joint Computer Conference, Boston, Massachusetts, April 26, 1966.*

Report generation

For purposes of discussion, “report” is defined as a display of information, the volume of which is such that most of the time the end product will be produced on an off-line printer. There are essentially two steps that are necessary to produce a final report: report description and report generation. In an on-line, time-shared environment, these two steps are so distinct that it is convenient to treat them as separate entities. When a user is describing his report, there is a great deal of on-line dialogue that must take place between him and the system; but once the description is complete, the actual generation of the report is a production-like job that requires little or no user interaction.

Thus, in the design of the TDMS report generator, these two functions were kept separate. The flow of information for COMPOSE and PRODUCE is shown in Figures 2 and 3. Functions performed by COMPOSE and PRODUCE are listed below:

COMPOSE
- Obtains a data base name on-line from the user.
- Creates a report description, interactively, from on-line statements by the user.
- Saves the report description under a file name given by the user; this will be its means of later communication with PRODUCE.

PRODUCE
- Obtains a report file name on-line from the user.
- Queries the user concerning any incomplete information from the COMPOSE phase.
- Obtains the report description and the related data base from the system.
- Compiles a report program.
- Generates a report by operating the compiled program.

Design criteria for a report generator

The main limiting factors in the design of report generators in the past have been the machine configuration the designers had to work with, and the logical structure of their data files. The report generator discussed in this paper is designed to operate under the control of a time-sharing executive using IBM S/360 computers (Model 50H or larger). File organization in TDMS makes optimum use of random access mass memory, permitting rapid retrieval and the maintenance of hierarchical relationships among file entries. Thus the designers of COMPOSE and PRODUCE were able to take advantage of facilities that were not available to many previous designers of report generators.

The main design goal of most report generators is to provide the capability for rapid, economical retrieval and presentation of any data from a data base. However, many report generating programs—although economical of machine time—largely ignore the specific needs of the user. Since the ultimate user of the reported information is concerned mainly with receiving the right amount of information, at the right time, and in a usable form, it seems natural to assume that the more directly involved the user becomes in the whole report generation process, the greater the
probability that his needs will be met. Furthermore, in order to accommodate the wide variety of users who have access to a time-shared system, it is necessary to provide a means of communication with the system that is simple and natural for the majority of users.

These considerations led the designers of the COMPOSE/PRODUCE report generator to design a system that involves the user—at least in the COMPOSE phase—to a greater extent than has been thought possible before. The following subsections describe how this plan was implemented.

**User-oriented system**

Apparently, the most desirable form of communication between a user and his data base is natural English; research is being done by several organizations in that regard. For example, an SDC-developed forerunner of TDMS, called TSS-LUCID, uses a command language which is a restricted subset of natural English. TSS-LUCID met with such favorable user response that the language of COMPOSE has been designed along the same lines. Within TDMS, the language specifications are such that they are consistent over all operations of the system from data description to retrieval.

User concern with the logical procedure of generating a report is kept to a minimum in COMPOSE. The user describes his report in a random order of nonprocedural statements. COMPOSE interprets these statements and builds the necessary report description for the PRODUCE phase of the operation.

After routine initial dialogue (naming files, etc.), the user of COMPOSE employs four basic types of descriptive statements corresponding to the "command" words shown in Table I. Group I statements determine selection, order of retrieval, and data reduction to be done. Examples of Group I statements are shown in Figure 4. Group II statements describe the contents of the report itself; examples of Group II statements are shown in Figure 5. Group III and IV statements give the user the opportunity to receive (on-line) a picture of how his report will be formatted. He may call for one of two types of review at any time. He also has the ability to add, delete, change or reformat his report, as is shown in Figure 6.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT TYPE</td>
<td>DATA BASE SELECTION</td>
<td>REPORT DESCRIPTION</td>
<td>REPORT REVIEW</td>
<td>FORMAT CONTROL</td>
</tr>
<tr>
<td>STATEMENTS</td>
<td>QUALIFY</td>
<td>TITLE</td>
<td>REVIEW</td>
<td>MASK</td>
</tr>
<tr>
<td></td>
<td>SORT</td>
<td>HEADING</td>
<td>PROOF</td>
<td>FEED</td>
</tr>
<tr>
<td></td>
<td>DERIVE</td>
<td>CONTENT</td>
<td></td>
<td>SPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECAP</td>
<td></td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>REPRINT</td>
</tr>
</tbody>
</table>

**Table 1—Basic compose descriptive statements**

**QUALIFY GAME WHERE FIELD EQ HOME**

**SORT BY TEAM, PLAYER**

**DERIVE PCT = (HIT/AB) * 100**

**Figure 4—Examples of data base selection**

**TITLE IS SEASON PERFORMANCE 1966**

**HEADING IS CITY, TEAM, SEASON HISTORY**

**CONTENT IS GAME, OPPONENT, FIELD, GATE**

**RECAP ON PLAYER = SUM HITS, SUM RUNS, MAX RBI**

**RECAP ON TEAM = TOTALS, SUM TMRUNS, SUM UPRUNS**

**Figure 5—Examples of Report Description**

**REVIEW REPORT**

**PROOF T1 . . . T4**

**FEED PAGE AFTER RECAP ON PLAYER**

**LIMIT OPPONENT = H9**

**PUT T1 . . . T4 AFTER OVERFLOW**

**Figure 6—Examples of Report Review and Format Control**

**Flexible use of TDMS data bases**

Without describing in detail the logical structure of a TDMS data base, suffice it to say that initially the user describes what he considers to be a logical entry. This entry may consist of any number of names,
dates, and numbers. Values which occur only once per entry are said to be at level 0; all others are defined as members of branches of a logical tree structure. These branches are called “repeating groups” and allow up to 16 levels of subdivision. Figure 7 shows a simple illustration of an entry for a baseball team season history.

If a user were to request a report from this data base, he might make a statement as follows:

CONTENT IS TEAM, COUNT WON,
COUNT LOST

At report-generation time, PRODUCE will loop through each entry and print the won-lost record of each team. PRODUCE is said to be iterating on a cluster of data values, the initial definition of a cluster being a logical entry. COMPOSE allows the user to be much more flexible than this. By the use of an operator known as “Qualify,” he may define his own cluster:

QUALIFY OPPONENT

This now causes the iteration to take place on each occurrence of “Opponent” and all related values both up and down the logical tree. “Qualify” not only allows the user to define his own cluster, but gives him the opportunity to subset his data base:

QUALIFY GAME WHERE FIELD EQ HOME

The iteration will take place on each occurrence of “Game,” but only on those games played at home.

Once a cluster has been defined, the data base, or its qualified subset, may be sorted on any values in that cluster and in any order. Although “Game” was originally defined as being minor in logical relation to “Team,” TDMS allows the user to make “Team” or “Opponent” (or other element names on those respective levels) minor in relation to “Game,” e.g.:

SORT BY GATE, OPPONENT, TEAM

Interaction with the user
The ability of the system to interact on-line with the user is of enormous value. COMPOSE attempts to achieve the following in its dialogue with the user:

• Prompt the user about rules concerning his data base or a particular facet of COMPOSE.
• Alert him concerning incomplete information.

Data reduction capability
COMPOSE gives the user ability to do data reduction in two ways. The first involves an element name from the data base, modified by one of the following: Sum, Average, Count, Maximum, Minimum, Sigma. The range over which these operators have effect is under complete control of the user.

The second form of data reduction involves the derived variable, which may be any arithmetic expression, including one or more related element names from the data base or additional derived variables. Figure 8 shows a simple example of data reduction and the use of dialogue between the system and the user. When COMPOSE encounters an element tag that does not appear in the data base, it requests a definition. If the definition is a valid form of data reduction, COMPOSE will request a format for the derived variable. (In the following figures, the system’s response is underscored; the user’s input is not.)

CONTENT IS TEAM, WINS, LOSSES, PERCENTAGE

DERIVE WINS =: COUNT RESULT
DERIVE LOSSES =: (162-WINS)
MASK LOSSES =: 999
DERIVE PERCENTAGE =: (WINS/162)
MASK PERCENTAGE =: .999

User control over format
COMPOSE has access to certain tables in a user’s data base that contain information regarding size, format, etc., of all the data fields. Using this information, and setting up arbitrary standards such as centering of title lines, alignment of heading and content lines, etc., COMPOSE attempts to format the report as described by the user, although the user does not...
have to worry about field or line position or about the order in which he entered his descriptive command statements. He may get an excellent idea of what his report will look like by use of one of the report review commands. They are:

- REVIEW, in which he merely receives an ordered monologue of what he has said.
- PROOF, in which a sample output of the report is shown.

The user may call for one of these reviews at any time during COMPOSE. If he wishes to alter the format of one or more fields or lines, he may do so by using SPACE (horizontal) or FEED (vertical).

Figure 9 shows what the user would receive as an on-line response for the given content statement, and an edit statement he might make.

Changing, updating and completing a report

COMPOSE allows the user to easily change any one of his statements either during his initial use of COMPOSE or at a later date. He may also call on a previously described report and use it as a basis for composing a new report without destroying his original. COMPOSE has also given the user the ability to leave certain decisions unresolved, to be completed later at the time of actual report generation (PRODUCE). This has been done by use of 'K preceding a word. For example:

\[
\text{CONTENT IS TEAM, OPPONENT, GATE} \\
\text{PROOF} \\
\text{SPACE 10 AFTER OPPONENT IN CONTENT IS TEAM} \\
\text{Figure 9 — On-line content statement response} \\
\text{QUALIFY PLAYER WHERE SALARY GR 'K SAL} \\
\text{By leaving the “Salary” limit open until the time of PRODUCE, this one report description can be used for any number of reports with different “Salary” criteria.} \\
\text{Sample uses of COMPOSE} \\
\text{Using the data base described in Figure 7, two simple problems, report descriptions and sample output are shown.} \\
\text{Problem 1} \\
\text{Generate a report showing the record of home attendance for each team, ordered by opponent. The report is to show the home team, game number, and attendance. Show the average attendance for the season, broken down by opponent; also show the average attendance for the season for all opponents. The record for each home team is to begin on a new page. Figure 10 shows the user statements, and Figure 11 shows a sample output.} \\
\text{Figure 10 — User report description for problem 1} \\
\text{Problem 2} \\
\text{Generate a report showing the performance of each pitcher, broken down by the opponents he has faced, and for each opponent give the number of innings he pitched, his won-lost record and earned-run average. Show each pitcher’s season record for the same. Begin each player’s report on a new page. Figure 12 shows the user statements, and Figure 13 shows a sample output.} \\
\text{SUMMARY AND CONCLUSION} \\
\text{In order to provide a report generation capability within a data management system that operates under time-sharing, it is necessary to design a report program generator that provides sufficient power to meet the user’s needs, but that is not so complex that the nonprogrammer user will be deterred. The COMPOSE/PRODUCE program being developed at SDC is a step toward the achievement of this goal. It allows the user to describe his report on-line, to do data reduction, and to control output format. It takes advantage of the advanced file structure of a TDMS data base, permitting great flexibility in the selection and representation of various reports from a data base. The COMPOSE portion of the program accepts command statements in an English-like language and in any sequence, in order not to deter the user who is more familiar with his data base than he is with the mechanics of file manipulation and report generation. Work is continuing at SDC to make user communication with large, structured files of data as simple and logical as possible, while retaining the power made available to the user through large-scale computing equipment.}
### HOME ATTENDANCE 1966

<table>
<thead>
<tr>
<th>HOME TEAM</th>
<th>OPPONENT</th>
<th>GAME NO.</th>
<th>ATTENDANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOS ANGELES</td>
<td>ATLANTA</td>
<td>4</td>
<td>23468</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>20182</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>26437</td>
</tr>
<tr>
<td></td>
<td></td>
<td>51</td>
<td>31628</td>
</tr>
<tr>
<td></td>
<td></td>
<td>52</td>
<td>28727</td>
</tr>
<tr>
<td></td>
<td></td>
<td>98</td>
<td>19605</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99</td>
<td>17342</td>
</tr>
<tr>
<td></td>
<td></td>
<td>143</td>
<td>24471</td>
</tr>
<tr>
<td></td>
<td></td>
<td>144</td>
<td>25389</td>
</tr>
<tr>
<td>LOS ANGELES</td>
<td>CHICAGO</td>
<td>11</td>
<td>16237</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>15331</td>
</tr>
</tbody>
</table>

**AVERAGE ATTENDANCE AGAINST ATLANTA**

<table>
<thead>
<tr>
<th>HOME TEAM</th>
<th>OPPONENT</th>
<th>GAME NO.</th>
<th>ATTENDANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOS ANGELES</td>
<td>CHICAGO</td>
<td>11</td>
<td>16237</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>15331</td>
</tr>
</tbody>
</table>

---

**Figure 11 — A portion of final report for problem 1**

**Title:** Pitching Performance 1966

**Qualify Player where Position EQ PITCHER**

**Sort by Team, Player, Opponent**

**Heading is Team, Pitcher, Opponent, Innings, Won, Lost, ERA**

**Recap on Opponent = City, Player, Opponent, Sum Inings, Sum Won, Sum Lost, ERA**

**Derive ERA = \( \frac{9 \times (\text{Sum ERUNS})}{\text{Sum Inings}} \)**

**Mask ERA = 0.99**

**Recap on Player = Sum Won, Sum Lost, AVG ERA**

**Feed page after Recap on Player**

---

**Figure 12 — A portion of final report for problem 2**

**Pitching Performance 1966**

<table>
<thead>
<tr>
<th>TEAM</th>
<th>PITCHER</th>
<th>OPPONENT</th>
<th>INNINGS</th>
<th>WON</th>
<th>LOST</th>
<th>ERA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOS ANGELES</td>
<td>DRYSDALE</td>
<td>ATLANTA</td>
<td>96</td>
<td>3</td>
<td>3</td>
<td>3.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHICAGO</td>
<td>84</td>
<td>2</td>
<td>4</td>
<td>3.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CINCINNATI</td>
<td>103</td>
<td>2</td>
<td>1</td>
<td>2.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOUSTON</td>
<td>114</td>
<td>3</td>
<td>1</td>
<td>1.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEW YORK</td>
<td>124</td>
<td>2</td>
<td>0</td>
<td>2.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHILADELPHA</td>
<td>83</td>
<td>1</td>
<td>4</td>
<td>3.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PITTSBURGH</td>
<td>78</td>
<td>2</td>
<td>2</td>
<td>2.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAN FRANCISCO</td>
<td>97</td>
<td>0</td>
<td>4</td>
<td>3.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ST. LOUIS</td>
<td>108</td>
<td>1</td>
<td>2</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 13 — A portion of final report for problem 2**