Header 记录详细说明了对象级别信息，例如操作系统级别、创建日期和时间。

每组报表行记录前有一个表描述记录。

报告记录是一个或多个指定页眉、明细、页脚、中断、汇总和最终总计报告行的记录。

值记录用于设置不同的非表格报告选项，例如报告是否应在打印中断总数后开始新页面。 值记录由 V、不同的四位格式选项编号、选项选择的长度和选定的选项字或编号组成。

End 记录仅包含 E 字符。 E 记录后面的所有信息都将被忽略。

As you can easily see, form source is cryptic. This is definitely one area where the Query Management Programmer's Guide is helpful. A series of single-digit alpha codes specifies a record type in position one; a series of four-digit numeric codes are used to define a report format option. The alpha record codes are:

H -- Header Record

T -- Table Description Record

R -- Report Row Record

V -- Value Record

E -- End Record

The Header record details object level information such as the operating system level, and creation date and time. There is one Table Description record before each group of Report Row records. Report records are one or more records that specify header, detail, footer, break, summary, and final total report lines. Value records are used to set distinct nontabular report options, such as if the report should start a new page after a break total printed. Value records consist of the V, the distinct four-digit format option number, the length of the option selection, and the selected option word or number. The End record consists only of the E character. All information following the E record is ignored.

The Table Description records, at first, seem to consist of an array of random three- and four-digit numbers that mean nothing. A closer inspection will show that the four-digit number's first two digits specify the Table type and the second two digits specify the option. 3 identifies most of the options that can be specified in a QM/Form source Table Record. These four- digit options allow you to code a dynamic columnar coding format for the subsequent Report Row Record. The first four digit code is the Table Record type. For instance, 1110 defines the Field Table Record. The second number, three digits long, defines the number of Report Records that the Table Description record's columnar format is being designed for. The third number, also three digits, defines the number of four digit/three digit option sets in this Table Record. The three-digit numbers that follow the four-digit numbers further define each four-digit number.

The Table Description records, at first, seem to consist of an array of random three- and four-digit numbers that mean nothing. A closer inspection will show that the four-digit number's first two digits specify the Table type and the second two digits specify the option. Figure 3 identifies most of the options that can be specified in a QM/Form source Table Record. These four- digit options allow you to code a dynamic columnar coding format for the subsequent Report Row Record. The first four digit code is the Table Record type. For instance, 1110 defines the Field Table Record. The second number, three digits long, defines the number of Report Records that the Table Description record's columnar format is being designed for. The third number, also three digits, defines the number of four digit/three digit option sets in this Table Record. The three-digit numbers that follow the four-digit numbers further define each four-digit number.

As you review the QM/Form in 2 remember that all of a Table's option need not be defined. But then, unselected options could not be used in the following Report Row Record like the C-Spec's dropped AN/ORs. Also, options that were not defined in the T-record or options that were not indicated in the Report Row record will default to the SAA standard. For instance, a field's width would default to the database's field size.

As you review the QM/Form in Figure 2 remember that all of a Table's option need not be defined. But then, unselected options could not be used in the following Report Row Record like the C-Spec's dropped AN/ORs. Also, options that were not defined in the T-record or options that were not indicated in the Report Row record will default to the SAA standard. For instance, a field's width would default to the database's field size.

The Report Row Record's Usage option allows you to define a wide spectrum of field usage definitions. The usage codes -- AVG, COUNT, FIRST, LAST, MAX, MIN, SUM can be used to specify reporting on break and final lines. Usage codes BREAK1 through BREAK6 specify fields on which to report subtotals. Finally, the OMIT usage code can be used to disregard completely a field on the report. The OMIT usage is handy when several QM/Forms are created for use with one QM/Query.

Field Report records also allow several other control options. Column headings are the most obvious. Field headings can be up to 62 characters in length and use embedded underscores '\_' to define multi-line headings. Width can be set, usually to override calculated field lengths. The sequence of field placement on the report can be overridden to be other than the QM/Query SQL SELECT field placement. Numeric edit codes can also be set if the QM default does not suit you. These strange codes -- E, D, I, J, K, L, P -- are the new set of SAA edit codes (see figure 4). A number immediately following these numeric edits overrides the default decimal precision for the Field Record. Unlike the C edit default which would simply truncate character data, the CW character edit will put any characters in a field that exceed the column width in the next report line. The CT edit may be use for mixed DBCS and single-byte character edit. Finally, the indentation between output fields can be set.

CALLABLE INTERFACE

The Callable Interface (CI) is a method of using Query Management with C, COBOL, and RPG programs. This interface allows application programs to execute QM commands. Don't get too excited here -- the term interface in CI means only that there is communication between the QM commands and the calling program. Data pulled from a QM/Query is not made available to the application program. The communications area was set up to pass return codes and completion messages back to the HLL from QM.

The RPG interface is through a CALL to the API DSQCIR which has three parameters. The first parameter is a data structure name which describes the CI communications area. The second variable is a four-byte binary value that indicates the length of the third parameter, a CI command string. This command string is used to run a query, print a query, or run any of the other commands available in QM. For more information please refer to the Query Management Programmer's Guide.

Interactive HLL programs using QM's CI run comparatively slow when executing a QM/Query. However, a QM/Query, once run, remains in a quiescent state basically like an OPNQRYF does, and lists are much faster the second time a request is run.

HAPPY SQLING

Overall, I would have to say that QM allows greater flexibility in report formatting over AS/400 query. A QM user would require an inherent knowledge of their relational database. Run-time QM is slow, but no slower than AS/400 query. A QM/SQL statement can be keyed in a minute versus the time-consuming process of the AS/400 Query menu-driven interface -- especially when several files are joined. Before QM I would write AS/400 query specifications on paper for my operator to create later. Now, using QM, I can enter the QM/SQL source statement in a couple minutes myself. Also, QM would be excellent for use as a prototyping tool for a proposed database system's screen inquiries and reports.

SQL is here to stay, and for those of us that do not have the standard SQL system, we can still become proficient in SQL using Query Management while providing a valuable service for our current employers.