Interfacing SYSTEM 2000® DBMS with Fourth Generation Languages

Gary A. Egan
GCI Systems Inc.

What is SYSTEM 2000?

SYSTEM 2000® is the Data Base Management System (DBMS) marketed by SAS Institute Inc. It has been in existence since the early 1970's. As with all DBMS, SYSTEM 2000 contains a number of separate components. Included in these components are a query language (Quest), a report writer and a procedural language interface (PLEX).

What is a Fourth Generation Language?

A Fourth Generation Language (4GL) is a software product that is meant to replace the procedural languages used in writing programs and developing systems.

Some examples of procedural languages are COBOL, FORTRAN, C and PL/1.

Every 4GL has its own language syntax which must be learned to use the 4GL.

There are two major classifications of 4GL. The first type actually generates a procedural language program. This generated program is then compiled. However, all changes to the program must be made via the 4GL and a new program must be generated and compiled.

The second type of 4GL does not generate a procedural language program. Instead, the 4GL interprets each command at execution time and does the appropriate action. This type of 4GL is common in on-line applications when there is screen handling and user interaction.

What is a Fifth Generation Language?

The terms Third Generation Language, Fourth Generation Language and Fifth Generation Language are difficult to define exactly because they mean different things to different people. Five "experts" in the field may give five different definitions.

In this paper, a Fifth Generation Language is defined to be a software product that is able to "learn" using Artificial Intelligence techniques. The language developer need not code for all possibilities because the 5GL system is able to expand its knowledge and store it for future reference.

In order to "learn", the 5GL interacts with the user. If the users asks a question that contains words or syntax not already known to the 5GL, the 5GL will prompt the user with a question or series of questions to attempt to understand what it is that the user desires. When the 5GL "understands" the new words or syntax, this knowledge is recorded in its "knowledge bank" for future use.

Why develop a 4GL/5GL Interface?

An obvious question is why bother to design and develop an interface between SYSTEM 2000 and other software products such as a 4GL or a 5GL.

SYSTEM 2000 is an excellent DBMS and is used in hundreds of companies world-wide. Thousands of data bases have been built and thousands of programs have been written to access those data bases. The amount of time and money invested is enormous.

SYSTEM 2000 has a query language and a report writer to access the data bases in a non-procedural manner. These tools are quite good and satisfy many of the needs of their users. However, every company has a variety of software products purchased from different vendors. Some are screen painters, report generators, 4GL and 5GL.

However, very few software products can access a SYSTEM 2000 data base. Therefore, in order to use these products to act upon data stored in a SYSTEM 2000 data base, the data first needs to be extracted or down-loaded from the data base. This often requires

a special program to be written for each request. This is not really an acceptable approach.

As a result, SYSTEM 2000 is often not used because the data is not readily available to other products. However, if software existed that allowed SYSTEM 2000 to interface with a variety of these products, more companies would make greater use of SYSTEM 2000 and they would have access to its many excellent capabilities.

Cross-reference Directory

The Cross-reference Directory (XREF) is needed to map the various names that a person might use to identify a data item to the name the data item is given in the data base.

The XREF Directory would contain, at a minimum, the name of the item that is found in a query, the data base where the item is found and the name (and number) of the item as it is defined in the data base.

Of course, additional information could also be stored in the XREF Directory. For example, it could indicate the data format and whether the item is a key. In addition, occurrence frequencies and performance information could be maintained so that the data could be retrieved in the most efficient manner.

The XREF Directory would contain only those data base items which are available to the user. This is one quite easy method to control data access.

A single data base item can be called many different names. For example, the data item Social Security Number found in the Personnel data base could be known as SSN, SS Number, Soc Sec Num and so on. In this case, the XREF Directory would contain 3 entries, each one pointing to the same data base item.

The XREF Directory should be sorted to optimize the search.

Example

Mr. Stephen A. Success, the purchasing agent for Acme Manufacturing Company, needs to purchase additional parts to meet the deadline for building power generators. He needs to learn which parts are involved in constructing the generator. He does know the part number of the generator. It is 12345. Thus, his query would be something like this:

> Tell me the part numbers of the parts needed to make part number 12345.

The query is immediately parsed by the 4GL/5GL. Superfluous words are removed. The query is reduced to the following:

> Tell part numbers to make part number 12345.

Tell is the verb. Part numbers is the object. To make is a qualifier meaning that a set relationship exists. Part number 12345 is the search criteria. The only query item name in this query is part number.

When the query is parsed, examined and understood, the 4GL/5GL must reformat the query and present it to the interface in the consistent format that the interface expects.

Now the interface takes over. First, it must determine which data items are needed, where they are located and how to retrieve them. The XREF Directory must contain the mapping from the query item name to the data base item name.

If the query item name is not in the XREF Directory, an error results. A more sophisticated system would prompt

the user for the correct mapping. Then this mapping could be added to the XREF Directory.

Once all the data items are mapped, the interface then knows where the data is located and how to retrieve the data. That is, the interface determines which access paths to use and in which sequence to use them. The XREF Directory should contain the necessary information so that the interface retrieves the data efficiently.

Now the interface constructs the proper SYSTEM 2000 commands and passes them to the DBMS to retrieve the data. As the data is retrieved, the interface is responsible for formatting the data so it can be used by the 4GL/5GL.

The part of the interface which communicates directly with the 4GL/5GL would need to be developed specifically for that language. The remaining parts of the interface are independent of the 4GL/5GL. They could be called "generic".

Note in this example there are two important points. First, it is assumed that the SYSTEM 2000 DBMS has access to all the data needed to satisfy the query. Second, the type of SYSTEM 2000 commands needed to retrieve the data are not explained.

These two points are very important in deciding which SYSTEM 2000 facility to use: Quest or PLEX.

Quest has the restriction that only one data base can be accessed at a time. PLEX has no such limit.

Quest commands are entirely self-contained. You execute the command and receive the data. The next command has no relationship to the previous command. You do not traverse through a data base using Quest. Quest is non-procedural.

PLEX is an acronym for Procedural Language EXtension. Thus, by definition it is procedural. PLEX is designed for more sophisticated logic and complexity. Using PLEX commands, you can position anywhere in the data base and traverse throughout the data base retrieving any records required.

Interface using PLEX

One design would be to use PLEX calls in the interface. To SYSTEM 2000, the interface would appear to be a procedural language program written and compiled to perform a specific function. In fact, the interface builds and executes each call dynamically, as required.

There are advantages using this approach. There would be complete flexibility in formatting the output data. Fields could be character or numeric, signed or unsigned, in compressed format or display format. The fields could be combined into records. Different record formats are possible. Multiple output files could be produced. Another significant advantage is that any number of data bases can be accessed. The requested data can be retrieved from many data bases. The data could even be distributed at different locations. If the interface is a procedural language program using PLEX, data could also be retrieved from files other than SYSTEM 2000 data bases. This could reduce the need to perform file downloading and reformatting.

One disadvantage is that it is more difficult to design and implement the interface when PLEX is used. Logic must exist to traverse up, down and across the data base. The interface must be able to link or connect data bases at execution time. Data formatting routines must exist.

An interface utilizing PLEX is best suited for creating extract files and for interfacing with compiled 4GL programs and software products from

other vendors. It would be more efficient that Quest for retrieving large amounts of data.

However, this design is not optimal for interactive queries such as those from an interpretive 4GL or 5GL.

Interface using Quest

This approach is definitely more simple and direct. All data could be accessed using only one retrieval command - PRINT. Since each command is self contained, the interface would not require logic to traverse the data base. The interface would not need logic to determine where and how the data is stored. Let SYSTEM 2000 handle these problems. Let SYSTEM 2000 do the data formatting.

The one major drawback in using this technique is that Quest can access only one SYSTEM 2000 data base at a time. Therefore, the user must ensure that all required data is contained in one data base when the query is processed. There are three separate options to satisfy this requirement.

Quest - Option 1

This option is the easiest to implement. The data base designer must ensure that all required data is in a single data base. An error occurs if the XREF Directory reveals that data needed to satisfy the query comes from multiple data bases.

This approach means that, somehow, the data base designer knows the changing needs of all the users. Of course, this is not practical. Neither the data base designer nor the user will be satisfied.

Quest - Option 2

Since Option 1 cannot handle those queries that must access more than one data base, the next progression is to combine the data into one data base before the query begins. An extra step is added to the beginning of the session.

The interface asks the user to name all the data items that will be accessed during the session. Using the XREF Directory, the interface determines where the data items are located. All necessary data bases are read and the data is loaded into a single data base. This approach is similar to some functions of a relational DBMS. The SYSTEM 2000 COLLECT statement could be used.

When completed, the interface proceeds to process the queries.

Since all data is gathered prior to the queries, the response time for each query is at a minimum. This is especially true if the resulting data base is small. Also, this new data base can be kept for future use.

This option also has its disadvantages. The user must know in advance which data is needed during the session. If he omits some data items, he must start over. As a result, most people would ask for more data than they actually need just "to be sure". This would waste CPU time and disk space. The user would wait unnecessarily while "too much" data is being gathered.

Quest - Option 3

In this option, the data is collected as needed. When the interface determines that it must access multiple data bases to satisfy the query, it invokes a function to collect the data into a single data base. Then the query continues.

This approach is most responsive to the user's changing needs. No unwanted data is retrieved.

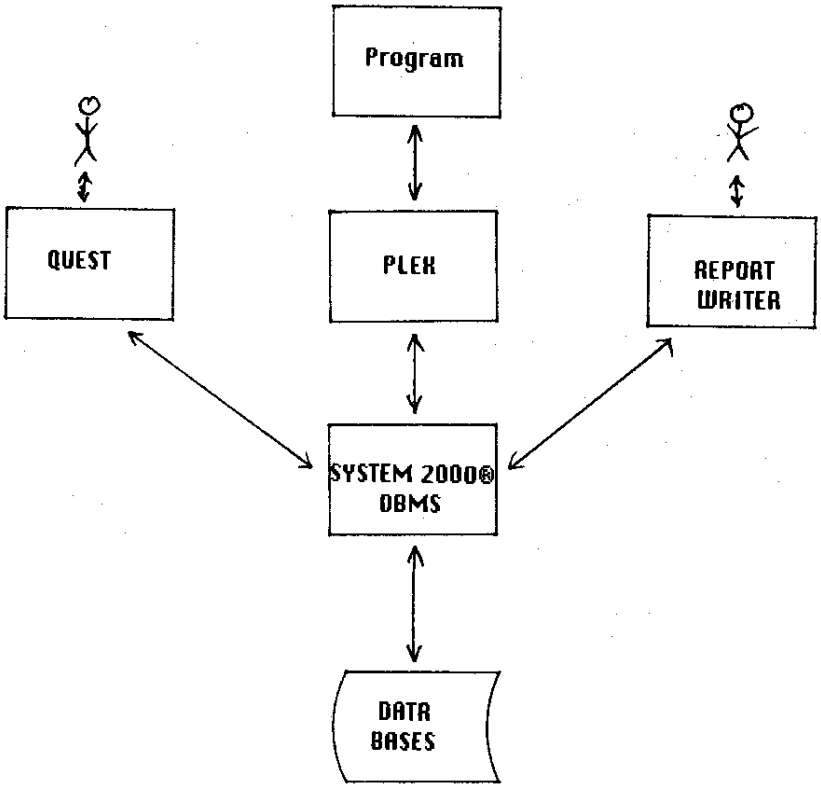
Since data is obtained as needed to satisfy the query, severe spikes can occur in response times. It may take minutes, or longer, to retrieve all the data needed for the query. The user may not be able to understand how a simple query can be so slow. Some type of smart look-ahead logic may be needed to predict what the user may ask for and do the work to collect the data while the user is thinking or keying.

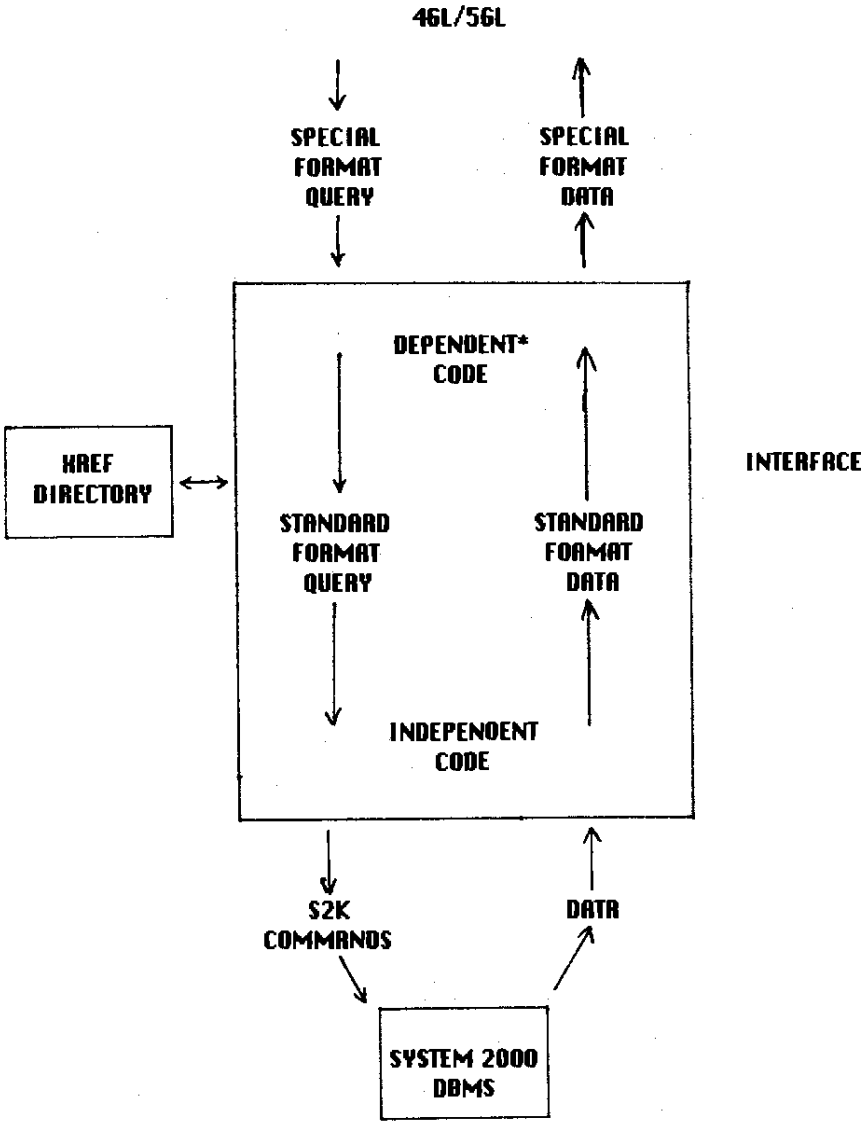
Conclusion

This paper has demonstrated several approaches to developing an interface between SYSTEM 2000 and other software products such as Fourth Generation Languages (4GL) and Fifth Generation Languages (5GL). Currently, this type of interface does not exist.

In addition, this paper has explained why such an interface is needed. It would, among other things, extend the life of SYSTEM 2000 applications and give companies the use of the many excellent capabilities of SYSTEM 2000.

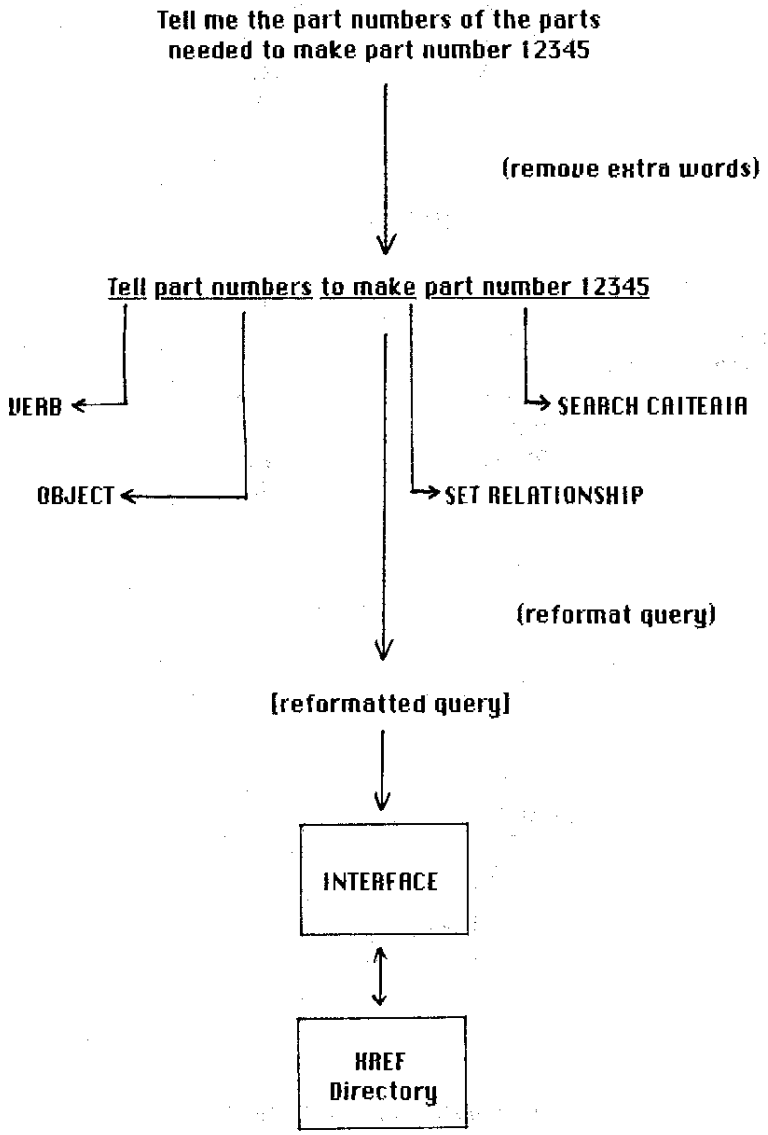
Now we must await the development of this extremely useful product.

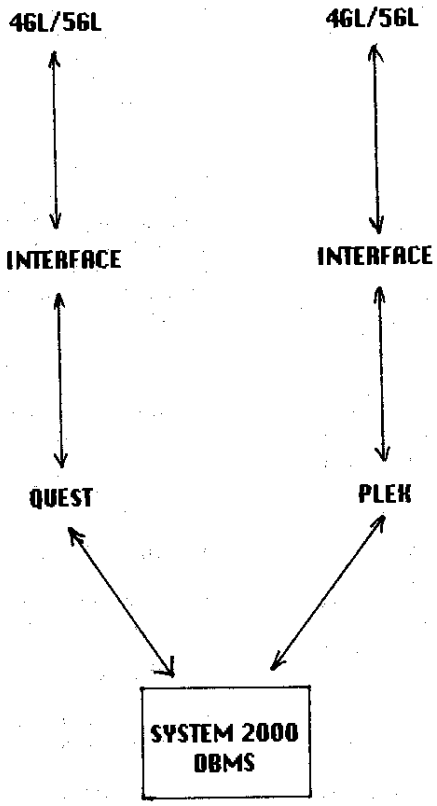

Program
QUEST
PLEX
REPORT WRITER
SYSTEM 2000® DBMS
DATA BASES



* DEPENDENT CODE IS TAILORED SPECIFICALLY TO THE 4GL/5GL

EXAMPLE


Tell me the part numbers of the parts needed to make part number 12345
(remove extra words)
Tell part numbers to make part number 12345
VERB
OBJECT
SET RELATIONSHIP
SEARCH CRITERIA
(reformat query)
[reformatted query]
INTERFACE
KREF Directory


4GL/5GL
4GL/5GL
INTERFACE
INTERFACE
QUEST
PLEX
SYSTEM 2000
DBMS