Getting the Users into the Game: Version 3.X

A Flight of Fantasy?

Art Brooks,
Director
Administrative Data Processing
University of Missouri – Rolla

Institution Information:
The University of Missouri – Rolla, founded in 1871 as the Missouri School of Mines and Metallurgy, is one of four campuses that constitute the University of Missouri system. While through its history engineering and science has been the focal point of its educational offerings, the campus also grants bachelor level degrees in several liberal arts areas. UMR offers bachelor, master and doctoral degrees in twenty-eight disciplines in three colleges and schools; the College of Arts & Science, the School of Mines & Metallurgy, and the School of Engineering. Although its student population has consistently hovered around 5,000 in population, the campus has a history of ranking among the nation’s leaders in bachelor degrees granted in engineering, and its graduates can be found employed around the world.

Paper abstract:
In today’s Information Technology environment, the programming backlog has continued to increase in dramatic proportions. That growth is to such a degree that adding more programmers is simply not the answer even if they could be acquired. This paper will present a concept that focuses on drawing the users into the report-producing arena. Fundamentally, the approach redefines the application architecture in a perspective that reduces the technical aspects of the relational tables in favor of table definitions to which laymen can better relate. Simply stated the application tables are recreated in ‘event oriented’, ‘functional tables’. This approach dramatically increases the number of tables defined and will cause relational purists to raise objections. The paper will discuss the concept, its initial implementation and the experiences to date.
Getting the Users into the Game: Version 3.X
A Flight of Fantasy?

The paper being presented is a description of an attempt to address a plaguing reporting dilemma and is not a guaranteed solution to this escalating IT problem. At the root of the definable crisis is an issue of supply and demand coupled with unrealistic user demands and an innate fear of technology on the part of typical computer users. With the dramatic increase in complexity of our administrative applications, this plight is only going to intensify. It would clearly seem our only viable option to address this issue is to get the user community into the game.

Dealing with a shortage of IT personnel and a surplus of reporting requirements takes on the perspective of a ‘Never Ending Story’. There are days when as a manager one wonders if ‘the powers to be’ believe IT directors are people who are students of Siegfried and Roy. Over the years the staffing has basically remained constant while the demands for service have dramatically accelerated. Thankfully, the reporting tools and database platforms available to us today are vastly superior to those of a decade ago. Only with those enhancements has any progress been made at all, if progress is what we have accomplished.

Upon entering the last decade of this century there was a considerable push within the academic community to address computer illiteracy among the clientele being served. Naively, many of us believed that increase in the knowledge base would provide dividends for the programming staff. It was anticipated the unreasonable demands from computer-timid staff would be tempered through a better understanding of what our computers could reasonably do and the effort it takes to bring a user’s desire to fruition. There was also a glimmer of hope that computer literate users could accomplish some of the more basic requests themselves, thereby positively impacting the demands on the computing staff.

With the end of the decade at hand it is shocking to realize that reduction in computer illiteracy actually made our situation worse. Instead of appreciating what it takes to accomplish a project, the users have come to believe anything is possible with computers. The ‘just push a button’ syndrome of several decades ago is in reality many times worse. Thanks to increased reporting requirements from external agencies and the attempt to deal with unpopular staffing adjustments, the user community has turned more and more to the computer. If that isn’t enough of a load increase, the users see unrealistic ‘solve all of your problems’ solutions on television, hear about them at user conferences and are enticed by smooth talking salesmen. Then, one must add the campus ‘dreamers’ who conjure up vague or science fiction appearing computer solutions to their office problems, and want someone else to make their dreams come true. All of this creates a load on IT staffs that has reached depressing as well as distressing proportions.

The University of Missouri - Rolla IT staff first addressed this load problem in 1986 with a commitment to abandon the traditional programming tools and databases to go with newer techniques. That re-tooling focused on SQL programs accessing relational reporting tables downloaded from the production files. Today, that set of tables is called a ‘data warehouse’. The result of that shift in techniques was a significant improvement in programmer productivity. It also provided the users a tool by which they could do their own reporting rather than waiting for a programmer to do it for them. Although it held great promise and did have a commendable success for programmers, as few as two or three users actually took on any portion of their report writing.

By the mid-nineties the programming requirements had continued to increase unabated, which prompted the UMR programming staff to once more look at new solutions, solutions that would be more effective at drawing an interest
from the users to attempt their own report writing. The solution implemented on the campus was the installation of InfoMaker, a reporting tool from PowerSoft. That tool was cost affordable, had significant output formatting capabilities, and had a graphical option. Since the user would not have to master SQL syntax and learn computer keywords, it was believed this could provide the needed relief from the accelerating reporting requirements. With that hope in mind and with the users in agreement, over one hundred installations of that reporting tool were installed on user desktop systems during a three-year period of time.

The net effect of that approach saw perhaps a couple of dozen individuals regularly running reports written by someone else. There were also a few more than a half dozen additional users writing reports themselves. The remainder of the installations have been used very sparingly or not at all by the users on whose desktop the software was installed. For the most part this lack of usage can undoubtedly be attributed to some users having decided the product took more of their time than they felt they had to spend while others concluded it was more difficult than they had anticipated. In essence, they found the reality that creating reports is much more than ‘just pushing a button’. It is simply easier to have someone else do the work for them.

Unwilling to accept that limited success and feeling it imperative to empower the users, the campus application support staff met once again in the fall of 1998. That group openly discussed the attempts of the past and the probable reasons for the meager successes with the users. It was acknowledged the characteristic in report writing which would most likely make user empowerment unsuccessful was the requirement for the report creator to have a basic understanding of the database from which they were going to attempt to write a report. The second factor was the need to have an understanding of SQL basics; such elementary functions as table joins, where statements, table keys and ‘if’ statements. All of this is assumed to be a basic skill by a trained programmer, but can be overwhelming for a naive user.

After considerable discussion, the challenge set by this group was to see if it was possible to reduce the technical requirements of report writing to a level so basic that in its pure form the user would only have to concentrate on formatting the appearance of the final product. The report design feature of InfoMaker is so similar to word processing and desktop publishing that it was believed the users could make the adjustment if their primary effort could be reduced to report design. That perspective led the group to focus on reducing complex systems to their most simple form. The degree of computerese required had to be eliminated to the greatest extent possible.

To achieve that simplification, the group defined an approach which focused on creating reporting tables that were designed for a specific report only. The concept of identifying ‘event oriented’, ‘functional’ tables was defined. In theory this approach would mean a separate table for every report desired. Although it was recognized this would demand a significant increase in disk space, in today’s computer hardware market disk drives are inexpensive. The group felt that cost in overhead would be offset by an improvement in processor efficiency. This is the final product of that group’s effort:

*****

A New Perspective in Reporting from the Data Warehouse at UMR

Preface
This is a proposal for a change in direction in reporting from relational databases at UMR. Traditionally, UMR has attempted to provide campus users with the ability to produce their own reports by giving them access to downloaded SQL tables. In acting in a responsible manner, the tables have been defined with efficiency as a primary requirement. This has meant the data has been organized into groups according to the relationship of the information involved. In addition there has been an attempt to limit the extent of data duplicity, allowing it only where it seemed
necessary. For a trained programmer this data perspective seems proper and has been fairly productive in producing requested reports as needed by the campus.

Over the course of several years InfoMaker has been installed on user desktops across the campus with the intention of empowering the users to create their own reports, an approach that has met with very limited success. It is very clear one reason for this limited use is a lack of time on the part of the users. They are simply too busy to have the time to understand the logic of a given database structure, organization, naming conventions and coding schemes, then to comprehend the complexity of creating an acceptable report using InfoMaker. Only recently has it been realized a contributing factor in this limited success has been with the organization of the tables. Data oriented tables are not necessarily obvious organizations for the normal users. After considerable discussion it has been decided a new approach needs to be attempted; one which presents the data in the perspective the users can best relate.

The Proposal
With this document it is being proposed a new direction in table presentation be implemented. That direction must be EVENT oriented and not DATA oriented. Users must be able to identify collections of data by a function they need to perform and not by logical data associations. In other words there needs to be tables created that are defined for a specific use; a collection of tables that simplify the query and reduce the number of joins required. From this perspective the ultimate query would be one which did a ‘select all rows’ on a single table. There would be no specific selection criteria and no tables to join.

In the traditional approach to programming, the program selects the data from the data source then produces the desired report in essentially a one step process. The basic concept being proposed in this document is to pre-select as succinctly as possible in the creation of the relational table. This approach, while a two step process, reduces, perhaps even eliminates the selection required when producing the final report. It allows the user to focus simply on the appearance of the final product without having to worry with what data to select and how to select it. As long as the table is succinctly defined for the user, this approach would reduce the incidents of individuals producing incorrect reports because of their lack of knowledge of the data characteristics.

This document proposes the creation of FUNCTIONAL tables that are created for a specified EVENT. It must be understood in creating these tables, as with other relational methodologies, there will be physically resident tables as well as logically resident tables. The rule must be enforced that no physical table will be produced if a logical table can be defined. It is also critical that all tables are succinctly defined and those definitions made readily available to the user community.

Acceptable practices dictate that if the table to be developed is merely a reorganization of existing tables or a different perspective of the tables, then a VIEW must be created. However, if the data to be presented must be manipulated in order to create the final form, then a PHYSICAL table must be created. Perhaps the best example of a required physical table is with the creation of address labels. With such requests, frequently duplicates are not desired, the type of salutation for the recipients can vary, the address type can vary from person to person and there can be blank rows in the address lines. In order to produce a final report the data would have to be manipulated in order to establish a presentable form. Therefore a physical table would have to be created that has the data pre-selected and preformatted with a specific event in mind.

The philosophy being proposed violates the primary rule in relational databases of having data singularly occurring. In reality it more closely approximates a maximizing of data redundancy rather than restricting it. This obviously will
significantly increase the amount of disk space required to store the data. However, it is anticipated this design will reduce the CPU requirements, improve the response time and make the reporting queries more efficient while providing the users with greater capability. It will make query production easier for the trained programmer and will remove some of the programming load by more effectively empowering the users.

It is important to understand the number of tables in existence in this approach is irrelevant. The computer professional must also understand the basic concept is to pre-select the data in creating the tables and not select the data in creating the report. Although it is recognized there still will be some selection required at reporting time, the important perspective is to significantly reduce the selection required by the user and most importantly to present the data in the form a user understands. This means the data must be organized in such a manner the user can focus on preparing a mailing for The Alumnus Magazine, or for a Parent’s Day activity or a faculty and staff mailing. They do not have to deal with ‘enrollment indicators’, ‘jurisdictions’, ‘release of information’, ‘benefit categories’ and the like. All of those requirements are pre-selected for them.

The basic design of this approach utilizes three primary types of tables; Functional, Selection and Status. Functional tables are multiply occurring within any given application. It would be anticipated Selection tables could be singularly occurring with a given application while the Status table is singularly existing for the campus. All reporting is performed from the Functional table while the purpose of the Selection table is to provide options to further delineate the final report. As its name would imply, the Status table is intended to provide the user and the system with the status of each table.

**Functional Tables:** The fundamental principle in this proposed approach lies with the creation of Functional tables. Functional tables very simply are subsets of an existing application or applications that have been created to serve a specific function. Conceivably, a Functional table can be a collection of data from multiple applications if the event or function requires it. They do not have to be homogeneous in nature. Because they have a specific purpose for their existence, most Functional tables are not intended to be permanent nor repeatedly refreshed. The tables are populated with data that has been created from pre-selection routines according to the definition of the Information Manger. For many requests a single Functional table that is properly defined can be the sole source for producing the desired report. It is this fundamental perspective which creates an increase disk requirement while reducing the level of complexity for the users and potentially reducing the processing requirements.

**Selection Tables:** While it would be convenient if all reporting could be handled by Functional tables, it is realized that possibility is unreasonable. Therefore, an integral part of this proposal is the formation of ‘Selection’ tables, a collection of data that is anticipated to be commonly used in creating reports from a given application. Depending upon the perspective taken in defining the table, Selection tables could be singular occurring in a given application and therefore would be regularly refreshed since these serve the need to create reports from multiple functional tables. This set of tables would allow the user to utilize the data created for a specific function (event) and utilize it for a different business function. It would require the joining of two tables, but eliminates the application complexity for the user and significantly reduces the number of table joins required for the report. In order to make the data more understandable for the user, liberties may be taken with the data in the Selection table. The purpose of this alteration is to eliminate application native codes which may be meaningless to the casual users and which can cause erroneous reports to be generated. Where appropriate, multiple data items can be merged into a single item, if the combination makes the data easier to use or more understandable for the user.

**Status Table:** In order to keep the system manageable, the Functional tables should not be allowed to remain populated with data for an extended period of time without being refreshed or emptied. In order to manage that issue,
a Status table would be created that would monitor the currency of each table. That table would consist of: table name, table create date, last refresh date, ‘archiving’ days and auto-refresh flag. Prior to creating the table, the Information Manager requesting the table creation must specify the number of days the table may be populated before it is emptied and note if the table is to be auto-refreshed. Each weekend a routine would need to be run that will review the refresh dates and the ‘archive’ days to determine if action needs to be taken on a table. When that number of ‘archiving’ days arrives, the table will be emptied of data, but the table will remain defined. If the table is flagged for auto-refresh, the table would be refreshed on the weekend after the ‘archiving day’ arrives. It will be the responsibility of the Information Manager to run a new table refresh when it is deemed necessary for tables that are not identified for auto-refresh.

Authorization Table: As a part of maintenance of the Event oriented approach, an Authorization table will be created. The purpose of this table is to identify the tables to which each individual has been granted access as well as to provide a simple method by which it can be determined what users have access to each table. This table will consist of a userid, a column that denotes the Functional/Selection table/view, authorization date and expiration date.

Implementation

The “generic label” table concept: In considering this approach it has been determined a single table format could be identified for the campus; a format which seemed applicable irrespective of the system. With the various label requests from the campus there are identifiable consistencies. Every label has a salutation (person addressed), and multiple lines of address. Since the length of each line of address can vary from application to application as can the number of lines identified, it was decided to define a table that used the maximum column length and maximum column occurrence. While every system has multiple lines for the address, all of them have occurrences where lines of addresses can be blank. Such incidents require the programmer to embed a series of ‘if statements’ in every label program to move lines of addresses up if blank lines are encountered. This process is complex enough that some users simply can not handle it while other users do not wish to deal with it.

Although it is accepted all labels have a salutation, the content and its length can vary. Some labels have simply a person’s name in the first line of the label. Other routines have a name in the first line and a title or company in the second line. Still other routines may have ‘To the Family of’ in the first line and the student’s name in the second line. In order to generically handle this, it was concluded to design a table that simply had multiple lines of contiguous (non-blank) information. In creating the table it does not matter what any line contains nor how many lines contain data as long as the blank lines occur at the end of the table. What is important is to name the table with nomenclature that indicates the table contents and to inform the user how many lines per label the final product will produce. In addition each table will have to have a profile which succinctly defines the contents of the table. These profiles will have to be readily available for the user community. In using these tables the user will also have to know the standard label used on this campus is ‘three-up’, thirty-five characters wide and five lines long using Times Roman font with twelve points.

Generic Table Definition:

Refresh date/time - to provide the user with the ability to determine the currency of the data in the table, a date/time stamp will be included in each table. This will require the user to initially test the report including the refresh date/time column prior to running a final report, if they wish to determine how recent the data had been refreshed. While this is an extra step, it may be more useable from the user’s perspective than looking at the Status table itself. If the user determines the data is not current enough for their report, the user will need to request the appropriate Information Manager to refresh the table.
Line 1-10 - the address lines of this table have been defined by the application that has the most number of address lines and the longest address lines. That system is the Advance system. The table will have salutations, and address types pre-selected with all blank lines occurring consecutively in the last group of address line columns.

Selection item - it is anticipated the Information Managers will be able to identify a small subset of selection items that would commonly be used for reports involving this specific table. The intention of including a few selection items in the functional table is to make the query easier for the user to write and to reduce the number of queries that require table joins. These items would be segments of a given application that the Information Manager would consider likely frequent selection criteria for the data contained in the table. This additional data could be such items as gender, enrolled term, year graduated or type of employee. As with the Selection table contents, some data items may be altered from their native form if the new presentation makes the data easier to use.

Sort item - it would be conceivable the tables could be presorted in zip code order, but a selected set of attributes could be included in the table to allow for additional sort capabilities. These columns have been included in the table definition for that reason. Specifically, the addressee’s name would be expected to be one of the sort items. If a user wishes to use alphabetic as a sort in producing a report, the salutation will not be in a format that will produce an acceptable sort.

Age of data - this is the number of days since the data in this row was last updated.

Row identifier - this is the item of data that succinctly identifies the row and allows the table to be used to join with other tables. Clearly if the row identifier is empty the row will not be selected in any report that includes a join.

An Initial Template: After considerable discussion, the table format for labels was defined as follows:

<table>
<thead>
<tr>
<th>Refresh date/time</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 2</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 3</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 4</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 5</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 6</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 7</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 8</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 9</td>
<td>60 char</td>
</tr>
<tr>
<td>Line 10</td>
<td>60 char</td>
</tr>
<tr>
<td>Sort attribute 1</td>
<td>60 char</td>
</tr>
<tr>
<td>Sort attribute 2</td>
<td>60 char</td>
</tr>
<tr>
<td>Sort attribute 3</td>
<td>60 char</td>
</tr>
<tr>
<td>Sort attribute 4</td>
<td>60 char</td>
</tr>
<tr>
<td>Sort attribute 5</td>
<td>60 char</td>
</tr>
<tr>
<td>Selection attribute 1</td>
<td>60 char</td>
</tr>
<tr>
<td>Selection attribute 2</td>
<td>60 char</td>
</tr>
<tr>
<td>Selection attribute 3</td>
<td>60 char</td>
</tr>
<tr>
<td>Selection attribute 4</td>
<td>60 char</td>
</tr>
<tr>
<td>Selection attribute 5</td>
<td>60 char</td>
</tr>
<tr>
<td>Age of data</td>
<td>03 char</td>
</tr>
<tr>
<td>Row Identifier</td>
<td>09 char</td>
</tr>
</tbody>
</table>

Reporting Accuracy: Inherent in distributing reporting capabilities to the general campus community is the concern a well intentioned user could produce a query which appears correct, but is in fact very bogus. By pres-selecting the data and populating functional, event focused tables, the potential for this occurrence is greatly reduced. This does not prevent them from misusing the data, but does restrict the data accessed as well as the content that is presented. It greatly reduces the chances of a user misunderstanding the data and producing an entirely erroneous report. In database terms, knowing what constitutes an ‘enrolled student’ or a ‘current employee’ are not as obvious as it might seem to the casual user. The use of Functional and Selection tables eliminates the need to know such definitions.

Maintenance: Maintenance of the Functional tables needs to be in the hands of the Information Managers. Procedures must be put in place to empower the Information Managers to update the defined Functional and Selection tables, thereby allowing the frequency and the currency of the tables to be at the discretion of the
individuals most knowledgeable of the systems and the data. There would not be an anticipated schedule of refreshing the Functional tables. Such activities are totally at the discretion of the Information Managers. It should be the responsibility of the Information Managers to inform the users of the status of the Functional tables as well as to define the contents and profile.

**Table Definitions, Creations and Documentation:** Clearly it is the responsibility of the campus Information Technology staff to create the tables, to develop routines to empower the Information Managers to refresh the tables and to provide technical assistance wherever needed. It is also the responsibility of Information Technology to determine which cases justify the creation of new physical tables and which ones require the creation of new views. The criteria for data selection and the table definitions is the responsibility of the combined efforts of the Information Managers and Information Technology. Due to the effort required to write the documentation for the new tables, that responsibility can also be a shared effort between the Information Managers and the IT. However, it must be a requirement that NO NEW TABLE CAN BE MADE AVAILABLE until the documentation has been made publicly available.

**Security:** In a traditional SQL environment users are either granted access to entire tables or views of table(s) which are created for specific user(s). This proposed approach retains the same concept except it would focus more on granting access to entire tables rather than creating views of segments of existing tables. It also allows the Information Manager to focus data distribution on events or functions rather than having to extrapolate views from anticipated needs of the users. Security in effect could become simpler for the Information Manager and less of a problem for the general user community. As with all administrative systems, the determination of security is totally the responsibility of the Information Managers. The IT staff can only grant user access as defined by the Information Manager responsible for the specific application. No user will be granted access to any table without prior approval from the Information Manager.

**Data Warehouse Organization:** Currently, there exists a data warehouse at UM and one at UMR. One implementation of this approach would be to establish the UM data warehouse as the resident or permanent platform for warehouse reporting. In this scenario the UMR data warehouse would be abandoned in its present form as it relates to core system applications, leaving the campus unique applications in tact. However, the locally resident core system applications tables would be replaced with Functional and Selection tables that are derived from the UM data warehouse on an ‘as needed’ basis. This approach would eliminate the current data warehouse redundancy and any confusion as to which warehouse to use while retaining the required interaction between applications needed by the campus.

**Examples:** Following the concept proposed, Functional tables such as these could be created:

| The Alumnus Magazine | All Staff       | All enrolled Students |
| Golden Shillelagh    | BlueKey (student directory) | Parent’s Day |
| Homecoming           | Accepted Undergraduates | All Faculty |

With such event-oriented tables, requests for reports for that specific event could conceivably be prepared with no selection required. This also means it can be simple for a campus user to generate their own report. Take as an example a campus user who has a need to do a mailing that exactly matches the format of the mailing for The Alumnus Magazine, except they want to only include individuals who live in a specific Alumni section. In this instance, the ’Alumnus Magazine’ table could be used and joined with either the Selection table, if the section data is included, or join with the table in the Advance system which includes the section information.

Given the need for faculty/staff mailings, one table could be created which has both faculty and staff included with a selection item that defines the type of individual. One could also define two separate tables identified as Faculty or
Staff. The decision to create one table or two would depend upon the anticipated frequency of use and the ease of using the tables for the users. What is most important is such items as determining employment status, ‘release of information’, salutations and address type are determined in the creation of the table and not left to the discretion of the creator of the final report.

The Office of Institutional Studies and the Registrar’s produce reports from tables noted as ‘IPED’. Could the use of Functional tables make this process easier? In order to eliminate writing new queries each semester they have needed to have a ‘global’ IPED table created which holds ALL IPED reports for ALL semesters. Would it be more efficient to create ‘Fall IPED’, ‘Winter IPED’ and ‘Summer IPED’ tables. Since they also prepare multi-year reports, would it be reasonable to create a ‘last five years table’ and a ‘last ten years table’? It would require changing the term range each fall in refreshing the tables, but would prevent them from having to specify a term range in their report. This would also speed up the report as the ‘global’ table is getting very large.

**Documentation Format:** the format for the user documentation would be as follows:
1. Table Name - event significant, limited to eighteen characters, non-blank.
2. Purpose of the table.
3. Criteria for the table creation.
4. Archiving days.(number of days before table is emptied)
5. Whether the table is auto refreshed.
6. Address Line definition - must identify the contents of address lines, length of each line and the maximum lines containing data.
7. Selection items - must identify the contents of the five selection items
8. Sort items - must identify the contents of the five sort items.

**Future Applications:** While the focus on this approach is for InfoMaker report writing, the ultimate method of report execution could well be from the Web. Although there are programming solutions available to launch traditional SQL queries from the Web, this proposed platform should make the Web responses quicker and less processor intensive. In reviewing the Web alternatives, the possibility of having a menu driven report request procedure should be seriously considered. Additionally, it needs to be reviewed as to whether the Web would be a more viable way of distributing queries/reports to users. (making available existing reports for downloads to user PCs) There are numerous possibilities, however, the potential for Web utilization is a matter that needs to be considered after this approach is deemed acceptable and placed into production.

**Summary**

In summary this proposal presents a new approach in presenting and defining the data warehouse for the campus users. The focus in this approach is to present the data in an image that is most understandable for the users, that being data that is assembled according to the needs of specific occasions. Fundamentally, this approach is Event oriented, utilizing ‘short-lived’ tables that are designed and created with a particular function in mind. Introducing the concept of Functional tables with Selection tables as companions, this proposal has the purpose of making both the trained programmer and the casual user more efficient and accurate in their creation of SQL reports. While acknowledging this will dramatically increase the disk space requirements, the Event oriented perspective should improve server performance and drastically reduce the potential for ‘runaway’ or inefficient queries.

In implementing this approach it is intended the creation of new queries/reports will be significantly simpler through the elimination of required table joins. The anticipated benefits of this data organization is empowering the users, improving the accuracy of reports, simplifying security, provide greater control to the Information Managers and improve the processing throughput while creating a platform from which other improvements can be made.

*****
Implementation and Experiences to date

In the spring of 1999, UMR implemented a new administrative system for a department, replacing a system that had over ten years of effort in producing user-defined reports. With the new system providing a replacement for few of the existing reports, a shortage of programming staff, and the users expecting to continue with their daily activities, there was a need to consider alternative programming procedures. Since the new system consisted of three hundred relational tables, it was determined the system had to be simplified if any progress was to be made with the reporting requirements. It was decided the new ‘event oriented’ approach provided the best opportunity to generate reports in the shortest period of time.

To begin the process the most common requested reports were reviewed and a handful of simplified relational tables were built. Due to the urgency of the matter these initial tables were designed to address the needs of a collection of similar reports rather than following the pure definition of one table for one report. These tables, by the definition of the new concept, are Selection tables, being an aggregation of numerous application tables. When placed into production, they provided a welcomed simplification of application complexity and produced a significant improvement in report development time and reporting consistency.

Among the earliest tables defined were address tables which are built with all blank lines removed, thereby eliminating the need to imbed numerous ‘if statements’ in the final report queries. As with the other tables this has meant a reduction in the requirement of technical expertise and data idiosyncrasies on the part of the report creator. This solution also allowed for the preparation of a series of reports that would have been extremely difficult to accomplish with traditional SQL approaches. In one instance, the user wanted a single report with a single set of columns to be created, drawing comparable data from four different tables. With traditional SQL reporting approaches the report would have had one occasion of each column for each table involved in the join. By using Perl to pre-select the data and placing appropriate data in common columns, the report was effectively completed using InfoMaker to do the final reporting. (Twenty-five separate reports were eventually created from this table.)

At this time the initial urgency to produce reports has begun to ease slightly. There have been a few true Functional tables defined and placed into production. As defined in the proposal document the tables were created with all of the table joins and complex data manipulations in the pre-select, thereby dramatically simplifying the final report writing. Those experiences have reinforced the belief in the benefits of this approach to report writing. Enthused by this beginning, the UMR programming staff is beginning to evaluate the initial results of this approach. In the next iteration these Selection tables will be significantly supplemented by Functional tables with the intent to reduce the reporting requirements to their true simplest form. The results of the initial effort are encouraging, but the desire is to so reduce the technical knowledge as to make it feasible for a user to create some portion of their own reporting needs. Even if that goal is not realized, the simpler tables improve the effectiveness of the programming staff. It also means that adjustments in report selection for a single table can actually benefit numerous queries without the need to modify each of the reporting queries.

To date, the author of this paper has been the predominant benefactor of this approach. There have been feelings of elation when new reports have been created with the need for only a single select statement and without a single table join. This is a vast improvement from some queries in other applications that have a multitude of selects and as many as eight or nine table joins. While a scant handful of queries have a dozen select statements, most of the queries have only two or three selects. In the instances where the query has only a few selects, those select statements have tended to be user prompts. This has significantly improved the development time while heightening the report writer’s elation.
Due to the urgency of the situation the critical maintenance portion of this approach has not been initiated. As a result we have had to resort to doing automatic nightly refreshes of the defined tables. This aspect of the concept is felt to be critical in the long-term success of this approach. Without a logical and manageable approach to table maintenance the concept will suffocate itself over time. Hopefully, when the Y2K push is over the maintenance feature can be created and placed into production.

The only real negatives experienced with this approach thus far has to do with the user’s lack of knowledge of the functional table characteristics and the occasional delay in creating a new functional table. The report creator must have total blind faith that what is in the table is defined properly for the report being prepared. That can make an experienced programmer very apprehensive regarding the results of the final product. Questions asked in regards to the table’s selection criteria have to be referred to the creator of the table. It does mean that in testing the report if the results are determined to be incomplete or incorrect, it has very rarely been the query that has needed to be changed, it has been the script that created the table. Therefore, errors found in similar reports have been fixed by changing one script rather than having to change each individual query.

The other problem encountered has been the delay in creating or modifying tables. This approach takes a greater amount of planning which inhibits its ability to react quickly to changes in requirements. Obviously, this was a greater problem in the beginning when few tables had been created and no reporting models were clearly defined. Such technical issues are inherent with data warehouses or with reporting servers where an exact copy of the original system is not created. The copy of the live data can add consistency to reporting requirements and insert a measure of security, but they also have a certain time delay in their responsiveness.

As stated earlier, the intent in taking this approach was two-fold. Of greatest importance was to find a way for user’s to create their own reports, thereby easing the load on the programming staff. Of secondary concern was to provide a productivity tool for the programmers. As with our earlier experiences, the second objective was accomplished without question. Unfortunately, the primary objective still appears to be elusive. With this latest attempt at empowering the users, the Information Manager responsible for the new administrative system has had InfoMaker installed on her desktop for some time. She also was the first user to have the newest version of that software installed on her desktop. At this point her only use of the new tables has been to run reports created by one of the CIS staff. Prior to getting the latest version of the software, on numerous occasions she made the offer to help address the load of reports her office needed. Thus far the closest she has come to using the power she has been given is to change the sort order of one report.

There is no question this user is very busy, nor is there any doubt she was sincere in wanting to create some of her own reports. However, the net results follow the trend of the majority of the users who were given such opportunities in the past. Unless they have a natural interest in what they conclude is programming or they have specific duties stating this as a requirement, most users will find many very legitimate reasons for not involving themselves in the business of creating their own reports. This aversion has been shown with considerable more frequency than the welcome embracing of the capability.

GETTING THE USERS INTO THE GAME. A FLIGHT OF FANTASY? Perhaps it is, but it may also be a phenomenon that simply takes time to establish a foothold. This latest approach holds the greatest promise for us.
Approach Comparisons

Traditional Approach

Application

Database

Query & Select

Full selection
Report formatting

Final Report

Requires full data knowledge
Requires adequate query knowledge
More flexible

Event Oriented Approach

Application

Database

Preselect

Partial/no selection
Report formatting

Final Report

Preselect routine run upon demand by Information Manager

Requires minimal query knowledge
Requires minimal data knowledge
Queries easier to write
Query complexity reduced
Security enhanced
Reduces errors
Provides
EVENT ORIENTED REPORTING

DEFINED TABLES

Purpose -- to provide the campus user community with more efficient, easy to use, easy to understand SQL tables from which they can generate their own reports. These tables are designed to satisfy the reporting needs for a specific event. They are designed to be refreshed upon demand only.

Purpose -- to provide the users with a reduced set of selection criteria from which reports can be created without having to join with multiple tables or to have a significant knowledge of the system from which they need to create the report.

Purpose -- to provide a view of all functional and selection tables and their status. It is from this table that it is determined when functional and selection tables are to be emptied.

Purpose -- to provide a mechanism by which user access can be managed and authorizations can be easily traced. This table contains the userid and the tables to which that userid has been granted access.
EVENT ORIENTED REPORTING
CONCEPTUAL VIEW
Address Labels

**Functional Table**

- Refresh date/time
- Line 1
- •
- •
- Line 10
- Selection item 1
- •
- Selection item 5
- Sort item 1
- •
- Sort item 5
- Age of data
- Row Identifier

- May contain ANY address information
- Salutation may vary from individual to individual
- Address is PREFERRED address, type may vary with individuals

Selection items anticipated for this collection of data

Sort items anticipated
- Name (L,F,M) definitely

Commonly occurring items for selection

**Application**

**Database**

**PRESELECT**

**Selection Table**