Selection of Database Management Software

Brian Westrich, University of Minnesota

Abstract

For organizations involved in computerized data management, selection of database management software (DBMS) is an important decision with far-reaching implications. A method is presented for effective selection of a DBMS that meets organizational needs. The method involves determining the complexity of required data processing and the level of organizational resources, choosing an appropriate DBMS genre, developing DBMS selection criteria, and choosing a DBMS based on these criteria. A DBMS selection process recently completed at the University of Minnesota Nutrition Coordinating Center (NCC), which led to the selection of a DBMS consisting of PowerBuilder and DEC rdb, will be described to illustrate this method of DBMS selection. The intended audience is the computer literate scientist who has little or no experience in developing, integrating, or installing software.

Introduction

For the purposes of this paper, database management software (DBMS) is defined as software that automates the collection, storage, retrieval, and presentation of computerized data. For organizations involved in computerized data management, such as organizations working with food and nutrient data, selection of database management software is an important decision with far-reaching implications. In the remainder of this paper, a method will be described for selecting a DBMS that will meet the needs of an organization.

Methods

The recommended steps in the DBMS selection process include 1. choosing the DBMS tool genre based on required level of processing complexity and available organizational resources; and 2. formulating and using selection criteria to choose the best DBMS tool from this genre. Of these two steps, the first is by far the most important, and yet is often the one most neglected by those selecting DBMS tools. In the remainder of the paper, each of these steps will be described more fully.

Complexity of data processing tasks

The selection process begins with an assessment of the complexity of the overall data management goals of the organization. The simplest data management task is to merely store data for later use. As one moves to greater levels of complexity, one may wish to structure data, relate data, or perform complex high volume data processing (the levels of complexity listed here are not intended to be a complete list of all data processing tasks, but instead a list of the tasks most representative of a given level of complexity). In this section, each of these levels of complexity will be more completely described.

Storage

Data storage simply implies the ability to save data for later use. Storing data electronically offers several potential benefits over storing data in a paper file. A basic benefit to electronic storage is that

calculations may be performed on these data without having to enter the data into a computer since the data is already in a computer. Another benefit is that electronically stored data are more readily searched than data stored only on paper.

Structuring

Data that is stored may also need to be structured. Figure 1 shows an example of structured data. In this figure, commercial product label information is broken up into different "pieces" (or fields), such as product name, ingredient listing, nutrient values, and other information. The advantage of such structuring is that one can more readily perform operations that are specific to a particular field. Examples of such field-specific operations include retrieval of all product names in the database, or retrieval of records for all products with a calorie content greater than a particular amount. In database terminology, the grid of data pictured in Figure 1 is referred to as a "table".

Figure 1: Example of structured data

Product Name	Ingredients	Serving size	Calories	Fat (grams)
Kellogg's Frosted Mini-Wheats	whole wheat, sugar, sorbitol, gelatin,	55 grams	190	1.0
Oak Grove Lime Sherbet	whey, water, sugar, corn sweeteners, ...	1/2 cup	120	1

Relating

Different types of structured data may need to have relationships defined between them. For example, product label information includes a list of one or more product ingredients. Rather than store all of the ingredient names in a single field, it may be desirable to create a new table which contains one row for each ingredient, as shown in Figure 2. Such a solution would allow, for example, the ability to store additional ingredient information, such as the estimated amounts of the ingredients.

If products and ingredients are stored in different tables, one needs the ability to define a relationship between these two tables. Such a relationship consists of the identification of one or more fields that are common to the two tables. For the example in Figure 2, the common field would be the product name.[1]

Figure 2: Example of two related data tables

(Product table)

Product Name	Serving size	Calories	Fat (grams)
Kellogg's Frosted Mini-Wheats	55 grams	190	1.0
Oak Grove Lime Sherbet	1/2 cup	120	1

(Ingredient table)

[1] Arbitrary identification numbers are usually preferred over text descriptions in such situations because of computing efficiency and data integrity purposes. Specifically, numbers can be processed by computers much faster than text, and also take up less space, and if the text names are modified the relationship between the two tables may be compromised.

Product Name	Number	Name	Estimated amount (grams)
Kellogg's Frosted Mini-Wheats	1	whole wheat	23
Kellogg's Frosted Mini-Wheats	2	sugar	12
Kellogg's Frosted Mini-Wheats	3	sorbitol	8
Kellogg's Frosted Mini-Wheats	4	gelatin	4
Kellogg's Frosted Mini-Wheats			
Oak Grove Lime Sherbet	1	whey	50
Oak Grove Lime Sherbet	2	water	30
Oak Grove Lime Sherbet	3	sugar	15
Oak Grove Lime Sherbet	4	corn sweeteners	12
Oak Grove Lime Sherbet	...	...	...

Complex high volume data processing

There may also be a need to perform extremely complex processing tasks, as well as process high volumes of data. An example of complex processing tasks is the management of time-related databases. Time-related databases distinguish between changes due to changes in the food marketplace and changes due to better nutrient data, and thus allow nutrient calculations to be comparable over time. Management of time-related databases requires the use of highly sophisticated database management techniques.

High volume processing usually also requires the ability for multiple users to simultaneously enter, browse, and update data; the ability to automatically keep a record of all data transactions made for data integrity and auditing purposes; the ability to ensure that user errors or hardware failures do not lead to corrupt data; the ability to perform data backups at the same time that databases are being modified; and the ability to readily upgrade to more powerful hardware such as UNIX workstations and multiprocessor machines as processing needs increase (scalability).

DBMS genres

There are several genres (or families) of DBMS tools that correspond to each of the levels of processing complexity that were previously discussed. These include word processors, spreadsheets, personal DBMS tools, industrial DBMS tools, and third generation DBMS tools. Word processor tools are the best choice when one wishes to merely store data in an electronic format. For such tools, no preparation in terms of data structuring or software development is needed before data entry can occur, and data entered can be readily searched.

At the next level of complexity, where one also wishes to structure one's data, an electronic spreadsheet is the tool of choice. Spreadsheets present the user with a grid within which data can be entered, and thus allow the data to be structured with minimal effort. The ability of spreadsheets in data management is often underestimated. Almost from the beginning, spreadsheets were designed with the intent that they would be data management tools. The "Lotus 123" software package was so named because it was designed to accomplish three functions, one of which was database management. Spreadsheets continue to be unparalleled in their flexibility at setting up ad-hoc data management applications, which makes them ideal tools for prototyping more complex applications. Recent additions such as the Data Forms feature of Microsoft Excel (which allows "instant" creation of data entry screens)

continue to add to the power of spreadsheets as data management tools. In short, one should not underestimate the ability of spreadsheets to perform data management tasks.

Personal DBMS tools allow one to relate data, and thus are more capable than spreadsheets for handling tasks that involve multiple types of structured data. Using personal DBMS tools, multiple data tables can be created and related to each other. Data entry forms and data queries that exploit these relations can also be readily created. Examples of personal DBMS tools include XBase packages such as DBase III, FoxPro, and Clipper, as well as other packages such as R:Base, Paradox, and Microsoft Access.

Industrial DBMS tools provide all the features of the previous genres, but also provide the types of features historically found on mainframe computers which allow complex high volume data processing. Such features include the ability to log all database transactions for historical purposes, the ability of multiple users to simultaneously access and update databases, and the ability to perform more complex data processing than do personal DBMS tools. One example of such processing is an SQL sub-select query. SQL sub-select queries greatly aid in the management of time-related databases, but are not currently supported by most personal DBMS tools.

One trend in industrial DBMS tools is to employ a computing architecture referred to as "Client/Server". In a Client/Server architecture, processing tasks are divided between a computer called a "server" which runs software that receives requests for data and provides data, and one or more computers called "clients", which run software that handles screen displays, and requests data from the server. Only those tasks needing central management (data storage, data integrity) must be performed by the server. Other tasks (user interface, data presentation) can be delegated (or "offloaded") to the clients. This division of processing labor allows the processing of large volumes of data using more cost-effective hardware (microcomputers and workstations versus minicomputers and mainframes). It is common to use the term "Client/Server DBMS" to refer to industrial DBMS's, but this terminology is not used in the current paper since the distinguishing characteristic of an industrial DBMS is not its Client/Server architecture but its ability to perform complex high volume data processing. Another reason why the term industrial DBMS is preferable over Client/Server DBMS is that when one uses a Client/Server architecture, one can use different genres of DBMS tools on the client side. For example, one could access one's data on an industrial database server not only through an industrial DBMS client software, but also through a word processor, spreadsheet, or personal DBMS[2].

The last genre of database tools to be discussed are DBMS tools based on relatively "low level" computer languages such as C or Pascal. Low level computer languages are harder to understand and are also less concise than are the languages used in other DBMS tool genres. Because of this, DBMS tools based on low level languages require highly skilled staff and are also much more labor intensive to use. To help counter this, third party libraries can be used to manage database management functions such as storage and retrieval of data. However, such tools do not usually provide users with the same level of facilities as do industrial and personal DBMS tools. For example, users of low level languages and third party libraries must still write custom programs to display data, as well as to do such database maintenance functions as creating, displaying, updating, and deleting database structures and the contents of these databases.

---

2 In some situations, the term DBMS is used to refer to the database server software (as opposed to the database client software), though such usage is not adopted in the current paper.

Though DBMS tools based on low level languages are more labor intensive and require more highly skilled staff than do other DBMS tools, they are inherently more flexible than other DBMS tool genres (in fact, most other DBMS tools are written using these low level languages). Thus the use of low level languages may be justifiable in certain specific circumstances. For example, some dietary data collection and processing software packages have specific requirements with regard to system performance and hardware platforms that only a low level language may be able to satisfy. Software developed by the University of Minnesota Nutrition Coordinating Center for the NHANES III study was designed to conduct a dietary interview using a hierarchical, time-related food database with suitable response times to allow one to conduct dietary interviews using 80286 based microcomputers. At the time that this software was implemented, such processing was only possible using a low level computer language; thus this software was implemented using the C programming language as well as a third party DBMS library. As hardware performance and capabilities of fourth generation languages continue to increase, for the most demanding applications there will be a continued and inevitable movement away from low level language DBMS tools and towards other tools that provide sufficient flexibility at greatly reduced development cost.

Genre selection

Table 1 shows, for each level of processing complexity previously discussed, the level of organizational resources (personnel and hardware) and the DBMS tool genre that is needed to support this level of complexity. It is worth emphasizing that one's organizational resources must be sufficient to support the level of complexity of data processing that one plans to perform. For example, if one needs to relate data, one needs a part-time data manager or consultant to manage the DBMS tools as well as to provide overall guidance in developing databases and database applications. A common error is to attempt to use a database genre that requires more resources than are available. This error leads to inefficiency due to lack of knowledge of, or technical support for, the database tools. Such a situation can negatively impact on the organization and lead to an eventual mistrust of the DBMS tools that were initially chosen, rather than a recognition of the true source of this difficulty, which is a mismatch between organizational resources and the DBMS tool. Thus, it is important to verify adequate organizational resources before committing to the performance of a particular level of data management complexity. Therefore, to select the appropriate tool genre for an organization, one should first establish the desired level of task complexity and also verify that the required organizational resources are available.

Table 1: Appropriate processing complexity and organizational resources for a given tool genre

Complexity	Minimum required resources	Genre
Storing	Paper	Non-automated
Electronic storing	Computer.	Word processor
Structuring	Semi-skilled staff.	Spreadsheet
Relating	Part-time computer staff	Personal DBMS
Complex, high volumn	Full-time computer staff; network	Industrial DBMS
Highly specific	Highly skilled programmers. Network of high performance workstations.	Low level language

The genres in Table 1 are guidelines as opposed to absolute rules. For example, many word processors have the ability to structure data through use of data tables, not all personal DBMS tools allow one to relate data, and it is sometimes difficult to draw the line between personal and industrial DBMS software. But despite such exceptions, these genres are useful categories for illustrating the major differences between available DBMS tools and for aiding in the selection of DBMS software.

Selection criteria

The next step after choosing the DBMS genre is to develop a list of selection criteria that are specific to one's organization. Everest (1991) has developed a generic list of database features for this purpose.

Available tools in genre

Once the tool genre has been decided, one can immediately focus one's tool search to those tools in the specific genre. Because of the large number of available DBMS tools, such focus can make the DBMS selection process much more manageable. While assembling this list of tools, one often learns of additional DBMS features that were omitted from the initial selection criteria. This information can be used to further update the selection criteria as needed.

Tool selection

The final step, DBMS tool selection, uses the selection criteria to select the DBMS tool of choice. Quantitative methods may be used to accomplish this selection. Such methods assign weights to each of the selection criteria, score each package in relation to each criteria, and use these scores and weights to calculate an overall score for each product. The product with the highest score is then chosen. However, because of the large amount of time and effort needed to thoroughly evaluate a DBMS tool (during which the DBMS tool market may change!), and because the genre selection process has already ensured that any tool chosen will match the desired level of processing complexity and the amount of available organizational resources, it may be more realistic to find a tool that does what one needs to do, rather than to find the "best" tool. Thus, a less rigorous approach may be preferable to the quantitative approach.

Hands-on evaluation of DBMS tools is often helpful in the final stages of DBMS tool selection. Such evaluation can be facilitated by vendors who are willing to supply evaluation copies of DBMS tools.

Example application of method

In the Summer of 1993, the University of Minnesota Nutrition Coordinating Center (NCC) selected a set of DBMS tools to meet its current and future organizational needs. The mission of NCC is to develop state of the art tools for diet assessment. Since effective management of large and complex databases is the central aspect of this activity, complex high volume data processing is a necessity. NCC has a full time programming staff, a dedicated network administrator, and the complete cooperation of the manager of the computing resources of NCC's parent organization, which includes a minicomputer running industrial DBMS server software. Thus, the industrial DBMS genre was the genre of choice for NCC.

The NCC client software selection criteria included (in order of importance) flexibility, multi-developer support, productivity, performance, training, import / export of data, ad hoc data manipulation, portability, vendor experience and viability, connectivity, documentation, end user tools, and price. The server software selection criteria included (in order of importance) functionality (for example, ability to manage time-related databases), connectivity (ability to use a variety of types of client software), data integrity (recovery from hardware failures), performance (speed), scalability, documentation, training, and price. Note that for both client and server, software price was the least important criterion. This reflects the fact that software costs are a small fraction of the costs of hardware and development personnel, and these two latter costs were already taken into consideration during selection of the database genre.

NCC selected DEC rdb for database server software, and PowerSoft's PowerBuilder for database client software. However, other organizations using the DBMS selection method described here will likely select different tools due to differing organizational needs and environment. PowerBuilder was selected primarily because of its ability to produce database applications that utilize a graphical user interface, because of its ability to use object oriented development techniques to improve developer productivity, and because of its ability to work with a wide variety of database servers. DEC rdb was selected because of its ability to work with PowerBuilder, as well as its availability (including maintenance and support from full-time systems personnel) on a minicomputer of NCC's parent organization, and the ease with which PowerBuilder applications can be ported to another database server in the unlikely event that sufficient computer processing power cannot be allocated to NCC from the organizational minicomputer. Another possible benefit of the PowerBuilder rdb combination is that the mechanism used for communication between client and server is the Open Database Connectivity standard (ODBC), a standard also used by the EuroNIMS Food Information Management System (EuroNIMS, 1994).

NCC plans to migrate all of its databases and database applications to the PowerBuilder / rdb environment. In doing so, the advantages of increased data integration will be realized, as well as increased long-term application development productivity through the use of the object-oriented application development capabilities of PowerBuilder.

A key advantage of following a methodological and well-documented selection process is that decisions emerging from such a process are readily defensible in the future, and can be revisited as technology progresses, as organizational needs evolve, and as available resources change. For example, a DBMS tool that is a competitor to PowerBuilder (ObjectView) recently issued marketing materials claiming their product to be far superior to PowerBuilder. In checking the documentation for NCC's DBMS selection process, it was found that when NCC selected PowerBuilder (in the Summer of 1993), the available version of ObjectView (2.0) had been reviewed in several articles and found to be less mature than PowerBuilder (Rayl, 1992; Anon, 1992; Anon, 1993). This suggests that the choice of PowerBuilder was a reasonable one at the time. Furthermore, it is also possible that the next version of PowerBuilder may again surpass ObjectView, resulting in a "leapfrogging" of the two products. Due to the learning

curve of industrial DBMS tools, it is clearly inappropriate to switch to a different tool every few months. Therefore, the decision to choose PowerBuilder, and to stay with it in the near term future, can be readily justified.

Summary

Effective DBMS selection is critical to organizational success. The most important step in DBMS selection is identification of the DBMS tool genre that best matches one's required processing complexity and available organizational resources. Genre selection quickly eliminates many DBMS tools from consideration, thus making the DBMS selection process more manageable. After the appropriate genre has been selected, one should formulate selection criteria that are specific to one's organizational needs, identify tools in the DBMS genre, and use the selection criteria to select an appropriate DBMS tool.

Conclusion

As computer technology continues to rapidly progress, those confronted with decisions can facilitate the decision making process, as well as justify their decisions in hindsight, through using a clearly planned, documented selection process. Successful DBMS selection and use, and ultimately the success of the organization, is facilitated by the use of such a process.

Final note

Despite the focus here on the importance of selecting the best tool for a particular situation, one needs to keep in mind that most organizations have multiple data management needs. Just as an effective carpenter keeps more than one tool in her toolbox, an effective data manager usually needs more than one tool in her database management toolbox. Thus, all but the tiniest of organizations will benefit from using tools from more than one of the DBMS tool genres of word processor, spreadsheet, personal DBMS, industrial DBMS, and low level language DBMS tools.

Acknowledgments

Bernd Zwatschka and Scott Kiesling provided technical assistance in the DBMS evaluation process. John Klensin and John Vilandre provided helpful comments on the manuscript. This research was supported by the Nutrition Coordinating Center, Division of Epidemiology, School of Public Health, University of Minnesota.

References

Anon (1992) Windows front-end development tools, *Info World,* April 27, page 62.

Anon (1993) ObjectView version 2.0 (1993) *DBMS,* July, page 3.

Everest GC (1991) DBMS Feature List. Carlson School of Management, University of Minnesota, Minneapolis, MN.

EuroNIMS (1994) The EuroNIMS system architecture: from GUI to DBMS. EuroNIMS newsletter, vol. 1, no. 2. The Opas Centre, St. John's Innovation Center, Cambridge: EuroNIMS.

Khoshafian S, Chan A, Wong A, and Wong HKT (1993) Client/Server SQL Applications. San Mateo, CA: Morgan Kaufmann.

Marcelo K (1993) Client/Server: Serving the business: A case study of a company's trials and tribulations with client server technology. *Database Programming and Design* August, page 58.

Murray J (1993) Selecting a Data Base Management System: Part I - series introduction. *Data Management Review* January, page 26.

Murray J (1993) Selecting a Data Base Management System: Part II *Data Management Review* February, page 29.

Murray J (1993) Selecting a Data Base Management System: Part III *Data Management Review* March, page 30.

Murray J (1993) Selecting a Data Base Management System: Part IV - The plan. *Data Management Review* April, page 21.

Murray J (1993) Selecting a Data Base Management System: Part V - MIS staff concerns. *Data Management Review* May, page 22.

Murray J (1993) Selecting a Data Base Management System: Part VI - Contacting the vendors. *Data Management Review* June, page 28.

Murray J (1993) Selecting a Data Base Management System: Part VII - Contacting the vendors' customers. *Data Management Review* July, page 38.

Rayl EW (1992) What's new with ObjectView. *Data Based Advisor,* November, page 63.

Williams W (1993) The right stuff: Finding your Windows front end. *Data Based Advisor* April, page 61.

Zurek B (1993) Comparing development tools: Will the "real" front-end development tool please stand up? *DBMS* July 1993, page 20.