Database development and working with a programmer

By Anne Hedrich and Harlan Brewer

Careful planning plus a good programmer equal success

You have a great idea for collecting information that would help your library. The information would need to be organized, manipulated, and updated to be the most useful. The obvious thing to do is to create a database to do all of this. You look at the software available and peek at the instructions for programming. You realize that if you had a year's leave of absence and could take an appropriate programming class, or had a half year's leave and time to learn the programming from the instruction manual, you could do the programming yourself. Or, you could hire a programmer.

Perhaps you are doing a survey of library needs or use, and need a respondent tracking system, or a database of how people responded to your questions. Maybe you are creating a database of a local resource, or you want an inventory management system of some kind. In any of these situations, you may find yourself in need of a computer programmer to create a database that meets your needs. Programmers know the field, and are familiar with database creation, structure, and uses. They can save you lots of time and headaches. They can also be expensive. Here are some tips for working with a programmer that will help maximize their benefit to you and minimize the costs.

Assessing the needs of your project

It is important to determine whether or not you really need a programmed database for your project. Your needs might be answered by one of the software packages already on the market. For example, spreadsheets work well for keeping data organized, and they provide the ability to do calculations and statistical work with the information you have. If you will want to compare the data in different combinations and do other data manipulation, you may need something more. Check to see if a similar project has been done elsewhere. Literature searches, questions posted on appropriate Internet discussion groups, or checking with colleagues may all provide leads to other projects. Occasionally such a project has produced software that is for sale. Tweaking software that pretty much meets your needs takes a lot less work and money than starting from scratch.

If you have determined that the only way for you to make your unique project work is to create your own database, the next step is to set down in detail what you want to do. Write down what information you will put into the database, and what you want to do with it after it is in there. Will you be updating data? If so, will you be keeping the original information in the database? How do you want to display and/or report data? Will simple data displays do, or will you want a function that creates printed professional reports? These are all questions that you will want to answer before you start to create your database. Have a sheet of categories of information collected and the specific ways you will manipulate it (e.g., creating a list from one of the fields in each record). This will help with linking fields and organizing records in database creation.

Perhaps a word about the parts of databases would be useful here. The programmer works with software to create a database. This can be software that is available on the market and has a programming language of its own, or it can be creating a database on a computer that

Anne Hedrich is a reference librarian at Utah State University, e-mail: annhed@cc.usu.edu; Harlan Brewer is a Ph.D. student at Utah State University, e-mail: hbrewer@msn.com

understands common computer languages such as Fortran, Basic, or C. Data in databases are organized into fields, which are similar to fields on bibliographic databases used in libraries. Fields are combined to make a record, again similar to bibliographic databases familiar to librarians. Records may be organized into tables, or they may be the highest level of organization in the database. Putting fields together in different combinations (achieved by linking fields), or pulling out only records with a specified entry or type of entry in a field, are ways of combining data.

Planning the project

When planning a project, always have the end in mind. Plan how long the project should take, and set a timeline for the work. Decide what will determine when the project is completed—whether it is when all of the data are entered, or when reports have been submitted, or, in some cases, when a certain time period has passed. Completion of the project should not rely on the amount of funding. If you determine that the end of the project will be when all of the money for programming or data input has been spent, you will end up with half of a project.

Talk to people at institutions where similar projects have been done. This will give you a good idea of what to expect from the project as a whole, and from the database and the programmer in particular.

Determine the costs of the project. You will need to budget for a programmer for *x* hours, including time for database creation, database adjustments, determining database integrity, and troubleshooting when things don't work as you thought they would. Talking to people who have done similar projects or asking several local programmers are ways of getting an estimate of how many hours your particular project will require. Research the standard programming pay rate in your area. Consider your options. Student programmers will be cheaper by the hour, but you may end up with a product of less quality because you are probably working with someone with less experience. Additionally, student programmers might take longer to do the programming work, and end up costing almost as much as an experienced professional programmer. Volunteers may work well, but again, consider the experience of the person you select and the ramifications of that level of experience. Whomever you hire, make sure that he or she documents what he or she has done on your project. That way, if you ever need to work with someone else, you will not have to pay that person to spend time figuring out what the first programmer did. You will also need to pay someone to input data, either in cash, or in your time if you are the one to do the data entry. You may have paper, copying, and/or mailing costs (for example, if the data are to come from a survey). If you do not already own the software you plan on using, you will need to purchase it.

Picking a programmer

There are two primary things to consider when choosing a programmer. The first is the quality of work that you expect to receive. The second is whether or not you can communicate well with your programmer.

When picking a programmer, look at the experience of the candidates. How long have they been programming? What educational credentials do they have: do they have a bachelor's degree in computer science, or are they relying on the Basic programming they did in high school math class in the 1970s? Have they worked with the software that you have chosen? Ask them how long they have worked with the software, not only as a developer but as a user too. From being a user, they have a better sense of what the software can do and can better counsel you to develop it efficiently. Do they have recommendations for other software? Get an idea of their experience by asking what other databases they have designed. Ask them if they are familiar with database normalization and whether they have done this or not. (Normalization means that the data are consistent throughout the organization of the database. It generally ensures that you have minimal data redundancy, in addition to consistent record updates and consistent record deletions.) Ask for references and check with other clients to see if they were satisfied with the programming work.

The other area of consideration is communication. Can the programmer explain things to you so that you understand him or her? Do not try to cover up your lack of understanding of computers by nodding a lot and saying "mmm" at what seem to be appropriate moments. You are partners in database development, and you must understand what is going on, at least at a rudimentary level, to make sure that the product matches your need. Asking for clarification also gives you a better chance to assess if you really can or can't understand what the programmer is talking about. Does the programmer understand you? Does he or she ask questions to make sure he or she is clear about what you want? And, finally, you might want to consider if you feel that there is something beyond communication there. Would you like to meet with this person on a weekly or biweekly basis for a period of time? After all, you may be working with this person for years!

Working with a programmer

All aspects of your working relationship should be detailed before work begins. Set a pay rate and schedule. A tentative timeline should be mutually understood, as well as what will determine the end of the project. Plan when you will meet about the progress of the database development. Plan how you will contact the programmer with questions or requests outside of the regular meetings. Many of these activities come under the rubric of general management. As with any situation, heeding practical management strategies will pay off in less confusion and headaches later.

Plan for the following phases in your project: meetings to convey your needs to the programmer, database normalization, database development with feedback, test run, data entry, another test run, final adjustments, and any ongoing adjustments if the project is a continuous one with, say, annual updates. Realize that you may be in one of the phases of the project longer than you thought you would be.

You, as project manager, will want to check to make sure the information is entered correctly. Database integrity must also be determined (are the data going into all of the fields you need them to?) The programmer can help you set up and run tests to check the database.

Constructing a database is time consuming, but there are times when nothing but such a resource will do (compare using an online catalog with using a card catalog!). A key to the successful creation of a useful tool is finding a good programmer to work with. And primary to a successful relationship with a programmer is good communication. Careful planning will help ensure the success of your project. Finally, realize that you cannot plan everything, and that the unexpected might come up. Flexibility is important to keep a balanced approach and your sanity. Good luck, and have some fun! ■