

Home / Learning / Library / Informal PC-based databases

Informal PC-based databases

Share to    

Article | Lessons Learned | July 1998

PM Network

Oehlberg, Tim

How to cite this article:

Oehlberg, T. (1998). Informal PC-based databases. *PM Network*, 12(8), 35–38.

Reprints and Permissions >

Formal databases like payroll or sales and contract are structured to perform specific functions, but informal databases have great potential to provide critical information to people throughout an organization. This article overviews database theory and terminology and proposes that more effort be expended in standardizing the development and documentation of organizations' PC-based, informal databases. The pros and cons of hierarchical versus relational databases are discussed, and an outline for designing an informal database is provided. Informal databases are valuable corporate capital and can be leveraged by those companies that help develop and document this important informational resource.

Is your department or group at the mercy of the one person who understands the database? Here's the knowledge you need to keep things running smoothly.
by Timothy Oehlberg



INFORMATION IS THE CORE of project management: we gather, assess, and package it to improve the accuracy of decisions. Today, databases are the most common tool used to manipulate information. They can be divided into two categories: formal databases and informal databases. Formal databases—such as payroll, sales and contacts—are developed using a structured process that performs a specific function.

Informal databases, on the other hand, are developed to provide information at the local level—in a department or group, for example. These databases may produce information critical to managing an organization or project, yet knowledge concerning the database design and structure often resides within the mind of a single individual. Given the employee turnover rate in the current job market, it's surprising that businesses do not expend more effort in standardizing the development and documentation of their informal databases.

Offer from our training partner

PMP® Certification Training

\$998

- ✓ Money Back Guarantee
- ✓ 35 Contact Hours
- ✓ Flexible Class Schedules
- ✓ FREE On-Demand Course



Get PMP Certification Training

Advertisement

Offer from our training partner

PMP EXAM PREP

Live Online Classes

\$998 save \$500

- ✓ Money Back Guarantee
- ✓ Flexible Class Schedules
- ✓ 35 Contact Hours



View Course

PMtraining

Advertisement

Database Basics

Informal databases have characteristics that distinguish them from formal databases. For starters, they perform a unique function specific to the local organization or project, such as tracking personnel training or documenting the work breakdown structure dictionary. A single individual or small group using commercial software, such as Access, Paradox, or Dbase IV, usually develops informal databases. The development and implementation of an informal database is not treated as a distinct activity or project by the organization, but rather is blended into routine operations.

These characteristics highlight the primary advantages of informal databases: they are tailored to the specific needs of the work group; they are inexpensive to develop; and they are designed and produced in relatively short periods of time. The goal of developing a standardized design and documentation framework must be to maintain these advantages, while minimizing the disadvantages of limited design knowledge by the users and a general lack of documentation. The first step in developing a standardized approach is to review the basics of the database.

I define a database as an integrated collection of data, organized to serve a number of applications. Applications such as preparing a report, performing an arithmetic function (such as subtotalling sales figures), or linking information to other databases (such as linking payroll data and production data) are all due to the integration and organization of data.

Information that describes a specific attribute is *data*. Examples include the duration of a project activity, the color of your eyes, or your average speed on a trip. Data that describes an attribute class is called a *field*. Examples would be original duration, eye color, and average speed. Note that, while data is specific to a given person or situation, fields are general categories. Fields are grouped together to form *records*, and records are grouped together to form *tables*. Groups of these tables are combined to form *databases*. The Exhibit 1 depicts the relationships between these terms.

Defining Our Terms: Data, Fields, Records and Tables

Name	Dept.	Title	
R. Johnson	Engineering	Engineering Manager	
A. Roape	Engineering	Lead Electrical	Record
		Field	

Data

Exhibit 1. Tables, like the one pictured here, express the contents of a database in organized form. Within a table, a record (shown in blue) contains one or more categories or fields (shown in green). Fields are made up of related pieces of specific information, or data (example shown in red).

A Hierarchical Database Structure

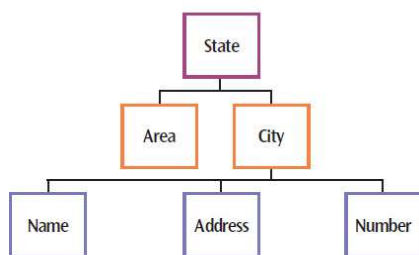


Exhibit 2. The hierarchical database model organizes records using a tree-like structure. This model is used for large databases that have a consistent format and require a relatively rapid response time.

Example of a Reference Table

Value	Description
Activity Code Length = 1	
A	Environmental Project
B	Groundwater Remediation
C	Interim Storage Facility

Exhibit 3. This example, from Primavera's P3 for Windows software, enables the user to set a reference table in which a value of either A, B, or C is used, as opposed to typing the entire description. **Relational** databases often consist of several tables like this one, each with a defined relationship to another.

Databases can be structured in several ways; the most prevalent are the *hierarchical* and the *relational* models. These models influence the behavior of the database by affecting flexibility, size, and efficiency. The hierarchical model, also known as a flat file database, uses a tree structure as the basis of organizing data. Some examples of hierarchic databases would be the phone book, a work breakdown structure, or a restaurant menu. Using the phone book example, if one of the fields was city of residence then each record would have a name in the city field. These systems are useful for organizing large amounts of data that

require rapid responses (such as a phone listing) and that use a consistent format. An example of the hierarchical model is illustrated in Exhibit 2.

The hierarchical model has several advantages: it is easy to develop; reports and other functions are straightforward; and it has a relatively fast response time. Its main disadvantage is that making system changes is very time consuming. Using our phone book example, suppose that one of the listed cities changed names. Each record would have to be accessed and updated and, even though this could be accomplished using a batch program, the amount of effort is considerably higher than with the **relational** database model. Other disadvantages associated with this design include a higher chance of input errors, the large amount of storage space needed, and a lack of operational flexibility.

The **relational** database model uses a series of references between tables. These references are created as a relationship between groups of records. A good example of a **relational** database is the activity codes used in Primavera scheduling software. Primavera has a pull-down menu that allows the user to develop a dictionary for a custom code value. The user defines the desired length of the code value (one character, two characters, and so on), the labels to be used for each code value, and the equivalent meaning of the value. Exhibit 3 shows an example of an activity code dictionary.

Once the dictionary table is set up, the scheduler can use the reference value of either A, B, or C to identify the project name associated with each activity, as opposed to typing the entire project name. **Relational** databases often consist of several tables like the one in Exhibit 3, each with a defined relationship to another. Exhibit 4 depicts a relationship map for a Microsoft Access database.

The tables can be combined into new tables, which are called *queries*. Queries represent the real power behind **relational** databases, because they allow the data in separate tables to be combined, organized, and used to generate new data. Thus, new types of information can be developed from the information already contained in your database.

For example, let's suppose that a manager wanted to find out the number of times that each customer has contacted a specific employee. A new query would be created that listed each employee name, and due to the **relational** references, would count the number of times a customer was associated with the employee. These reports can be prepared rapidly, by anybody in the organization who learns to use the software.

Global changes to the **relational** system can be easily completed, and the amount of computer storage used is comparatively less than hierarchical models. However, the **relational** model is slower to respond than a hierarchical system, especially when large numbers of records are being maintained.

The design and implementation process that follows can be used for either type of database.

Designing an Informal Database

The design process for informal databases uses a team approach to develop, test, and implement the new database. The extent to which each of the following tasks are performed is dependent upon factors such as the proposed size of the database, the number of users, and the impact of the database on productivity. For the single user, the design process may be as simple as diagramming the relationships on notebook paper. Large databases may require that all of the tasks described below are performed, and that substantial effort is expended toward documentation. It is the role of the project manager to determine the appropriate level of design discipline.

Establish a Database Design Team. The database design team should represent all the groups that will be impacted by the creation of the database and who therefore have a stake in the final product. At minimum, the team should include a manager, a programmer, and an end user. A kickoff meeting should be held to determine team member functions and assign tasks. Design decisions should be made by consensus.

Develop Database Outline. The database outline is developed to determine the main and auxiliary purposes of the database. Small projects may combine this step with developing the detailed design; on larger projects, several meetings may be required to reach consensus on the database outline. The final product of this step is an outline of your final scope document.

Develop a Detailed Design. This design is an iterative process that consists of the team members agreeing on the details of the database. The detailed design should include the scope of work and a data flow diagram.

Create Initial Version of Database. This step is performed almost entirely by the programmer, and consists of converting the detailed design into a product. Component functionality is tested as it is developed, and some initial trials may be conducted using small data sets. Larger data sets are used when the end users perform the first functional tests.

Perform Functional Tests With End Users. The functional tests should be conducted in as close to real conditions as possible. The purpose of this step is to determine if the database meets the design criteria and if these criteria were adequate. Documentation of the functional tests is invaluable to the programmer, and can be accomplished by having the users keep a journal while testing the database. The journal should include information such as initial impressions, ease of use, and any error messages that occur. The journal should be given to the programmer, with a summarized list of corrections, comments, and modifications.

Custom, internally developed databases will increase as software and hardware become more powerful. This trend, coupled to the increased mobility of the workforce, forces project managers to formalize the database development process or risk losing valuable corporate knowledge when the “database guy” takes off for greener pastures.

Reader Service Number 5027

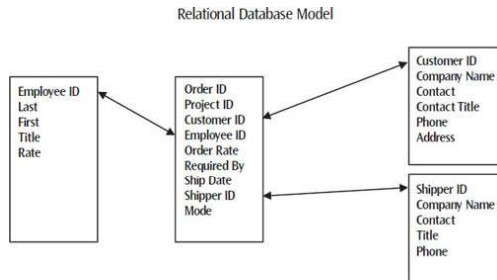


Exhibit 4. The relationship map associates records from various data tables to each other via common fields. In the example shown, a record only has to indicate a specific data point, such as customer ID, to allow the creation of reports that use all of the associated information. This saves time in terms of data entry and maintenance.

While this list does not encompass all of the attributes necessary in developing an efficient database, I believe it contains the major components. A checklist such as this is a useful metric to compare against your database design.

✓ **Stable and Reliable.** The most important characteristic of a good database design is reliability and stability. A reliable database will produce the same answer for the same set of inputs over time. A stable database will remain operational instead of locking up your computer. Stability and reliability can be designed into the database by using software standards and by keeping the design functions as simple as possible.

✓ **Easy to Use.** Databases are designed for the end user, not the programmer. The interface should be intuitive to the end user and perform the desired functions. This is easily accomplished by including end users in the design process.

✓ **Produces the desired Information.** The purpose of designing a database is to produce a desired product. The primary purpose of the database can easily get lost, especially when auxiliary functions are added on during the design process. A good method to ensure that the desired information is produced is to label the main function of the database on the scope of work, and to review the scope periodically.

✓ **Has Adequate and Complete Documentation.** For internal database development, this is probably the most common mistake made. Usually, this is due to the demands of the schedule and lack of resources to complete this task. To correct it, include documentation requirements in the scope document, and assign the function to an individual other than the programmer (she or he has enough to do).

✓ **Has Technical Support.** Even though the database is properly tested and installed, there are always issues and questions that arise. The lack of technical support can cause delays, reduce confidence in the database, and increase the frustration level in the office. The implementation phase should include establishing a technical contact to address questions and comments, before the organization is impacted. This technical contact can be one of the end users, the programmer (although they really are busy), or a contract service.

✓ **Performs a Unique Function.** If you have two databases performing similar function, you may have wasted some of your resources. Prior to developing a new database, explore the possibility of modifying an existing database.

Incorporate Test Results/Revise as Required. The programmer should have a clear picture of the reason behind any changes or modifications to the database design. This understanding can be achieved by using an informal test journal, interviewing the end users, and repeating some of the functional tests. The changes should be incorporated, and tested until the programmer and the end users are satisfied with the results.

Issue/Install Final Version. The final version of the database should be issued and installed on the end users' computers. If the installation is not performed by

a technically proficient individual, a detailed installation procedure should be attached.

Complete Support Activities. Support activities include ensuring adequate compliance with your company's QA/QC standards, developing training and procedures for end users, establishing a technical support system, and adequately documenting and filing your design data.

AN EMPLOYEE WHO DEVELOPS either a formal or an informal database possesses knowledge that is part of the corporate capital. The process outlined here focuses on documenting this knowledge, while still maintaining a nonrestrictive environment. The use of custom, internally developed databases will increase as software and hardware become more powerful. This trend, when coupled to the increased mobility of the workforce, will force project managers to formalize the database development process or risk losing valuable corporate knowledge. ■

Timothy Oehlberg is a project manager who consults for Project Time & Cost (PT&C) Inc., developing and implementing project management systems and business integration solutions. He is a member of PMI, the Institute of Management Accountants, and is a Certified Cost Consultant. He can be contacted at 303/641-6404 or oehlberg@ix.netcom.com.

This material has been reproduced with the permission of the copyright owner. Unauthorized reproduction of this material is strictly prohibited. For permission to reproduce this material, please contact PMI.

PM Network • August 1998

Like what you just read?
Log in or register for a free PMI account to get access to even more articles like this one.

Login or Register

Related Content

Article

Moment of Truth
PM Network queries the project management community about lessons learned.
[Read Now](#)

Article

Momento da verdade
PM Network consulta a comunidade de gerenciamento de projetos sobre as lições aprendidas.
[Read Now](#)

Article

El momento de la verdad
PM Network consulta a la comunidad de gestión de proyectos sobre las lecciones aprendidas.
[Read Now](#)

Article

Major Knowledge Diffusion Paths of Megaproject Management
By Wu, Hengqin | Xue, Xiaolong | Zhao, Zebin | Wang, Zeyu | Shen,...
[Read Now](#)

Article

Building Project Management Capabilities with Knowledge Networks
By Algeo, Chivonne | Linger, Harry | Pugh, Katrina | Asif, Zaheeruddin...

Feedback

10 Lessons of the Most Influential Projects

By Prashara, Sunil

There are many lessons to be found in the Most Influential Projects. We've pulled out 10 of...

Read Now

Read Now

Offer from our training partner

PMP® Exam Prep Classes

- ✔ Money Back Guarantee
- ✔ 35 Contact Hours
- ✔ Flexible Schedules

PMtraining

\$898

On-Demand Course

~~\$348~~ **FREE**
With any Live Class

View Classes

Advertisement

Quick Links

- Report PDUs [↗](#)
- Online Courses
- PMBOK® Guide
- Webinars [↗](#)
- Events
- Store
- eLearning Gifts

Certifications

- Project Management Professional (PMP)®
- Certified Associate in Project Management (CAPM)®
- PMI Agile Certified Practitioner (PMI-ACP)®
- Compare Certifications

Membership

- Become a Member
- Local Chapters
- Membership FAQs

Community

- Latest from the Community [↗](#)
- Discussions [↗](#)
- Templates [↗](#)
- Blogs [↗](#)
- Volunteering

Partner with PMI

- Enterprise and Commercial Partners
- Training Partners

Organization

- Our Mission & Vision
- Our Purpose
- Our Leadership
- The PMI Blog
- What is Project Management?
- What is a Project Manager?
- Press & Media
- Careers

Support

- Contact Us
- Store Help

PMI®

Stay Connected

[f](#) [X](#) [@](#) [v](#) [in](#) [i](#)

[Accessibility](#)

[Privacy](#)

[Sitemap](#)

[Terms of use](#)

[Purchasing Terms](#)

[Advertising & Sponsorship](#)

© 2025 Project Management Institute

Feedback