



Breaking the Rules to Improve Performance, Part 1—Expanding the Concept of a Keywords Subfile

By: Walt Nelson

Introduction

In the database development community in general, and in the 4D development community in particular, we have certain “rules” that we observe and live by. Most of these rules are considered “good practice” in designing relational databases, and they stemmed from the ongoing research by experts in the field, such as E.F. Codd and C.J. Date. Some database designers regard those rules as unbreakable Commandments.

There are times, however, when you need to open your mind to possible approaches that achieve the desired results by *breaking* the relational rules. One such time is when you are having performance problems with a database—especially one that is used in a multi-platform environment.

Now that 4D Server supports multiple operating systems and multiple hardware platforms, performance will become more and more of an issue. If you write a database that is used by several different types of machines over a Wide Area Network (WAN), you will have to be much more concerned about performance than you were when the database was used in one building, on one platform, over direct a Ethernet connection. Performance that was “acceptable” or even “outstanding” over a local area network, can become “unacceptable” or even “unbearable” over a wide area network or through a dial-up modem connection.

If you have experienced performance problems with an application—and if you are willing to open your mind to “unconventional” solutions, then this technical note should benefit you.

Three Rules That We Will Break to Increase Performance

There are many rules that we 4D developers have learned to live by over the years. There are three rules in particular, that can adversely affect the performance of an application in a WAN or a dial-up situation.

- Don't use subfiles



4th Dimension

Technical Note 96-26

- Don't use 4D's built-in relational lookups; write your own.
- Don't duplicate data—normalize

In this technical note, we will talk about performance-enhancing ways to break rule #1 (subfiles) and rule #3 (de-normalized data). In a companion technical note this month, we will talk about performance-enhancing ways to break rule #2 (4D's built-in relational lookups) and rule #3 (de-normalized data). When we discuss each rule, we will first examine why the rule exists and then we will understand why breaking the rule, in certain cases, can give you geometric improvements in performance.

Rule #1: Don't Use Subfiles

The theoretical problem with subfiles is that, according to Codd and Date's description of Relational Theory, subfiles do not fit into the relational model. Subfiles are more hierarchical than relational, and so they have a major data access limitation: a subrecord can only be accessed through its parent record. This makes it difficult—or at least cumbersome—to Report, Import and Export data into and out of subfiles.

In the example database accompanying this technical note, we have shown you a way to increase performance many-fold by taking advantage of the hierarchical "weakness" of subfiles.

Rule #3: Don't duplicate data; normalize.

To *normalize* data is to design the physical structure of the database so that, with the exception of primary and secondary keys, no piece of data is located in more than one place. For example, if you have a [Companies] file and a [Contacts] file, normalization rules dictate that you only store the Company Name in one place: the [Companies] file. In theory and in practice, there are very good reasons for normalizing data. Normalization simplifies data maintenance and reduces storage requirements.

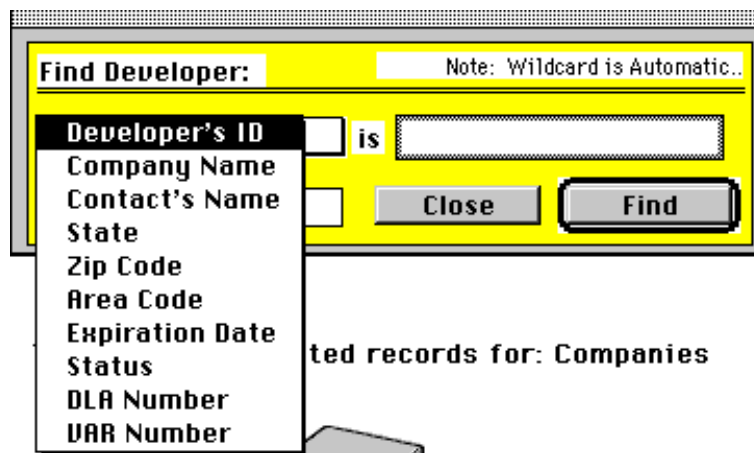
Multiple-field Searches Using Subfiles and Duplicated Data

A standard developer technique is to create a "Quick Search" dialog that enables the user to search indexed fields. Here is an example :



4th Dimension

Technical Note 96-26



The above dialog is a “smart” quick search. The pop-up menu changes depending on the file the user is searching. Many 4D developers use this interface to allow the user to search an indexed field.

There are some problems with this interface in a WAN or a dial-up connection, however:

Problem #1: The user has to understand the structure of the data: “How are customer names stored? Do I search for the Company name, or the last name, or last name first name, or just the first name?”

Problem #2: The user has to use the mouse to choose from a pop-up or to click on a check box or radio button. This can become a major irritation for people who like to use keyboard shortcuts.

Problem #3: A new user or an occasional user will not know certain “inside information” about the data. For example: the company that everyone calls “New York Brokerage” may have a “The” at the beginning of the company name. The real name of the company might be “The New York Brokerage Company.” The user will not be able to find this record by searching for “New York...”

Problem #4: If the user is unsure of the *spelling* of the customer's name, the user may have to do several search operations to get a list that contains that customer.

Problem #5: If the user needs to search by **either** one field **or** another field (e.g., Contact name or Company name), or if the user needs to search several fields at the same time, you have to expand the dialog and expand the underlying code to accommodate this need.



4th Dimension

Technical Note 96-26

Search Field:

- Company
- Last Name
- First Name
- Zip Code
- Keyword(s)

Search for:

Find Contact records that match:
the above conditions. Any of All of

All of these problems can exist in any 4D application; however, when the user is connecting via a 56kb line over a WAN or—even worse—via a dial-up modem, at 28.8kb, 19.2 or 14.4kb, the problem is magnified many-fold. For example, a single-field indexed search might take seven (7) seconds to return a result set of records to a local user running a Pentium or a Power Mac client over Ethernet. That same search, performed by a dial-up user running a Powerbook 180 or a laptop 486 via a modem connection, might take seventy (70) seconds—or longer—to return the same result to the user.

If the user has incorrect or incomplete information, making it necessary to perform several search operations to find the correct record, then this one search could turn into a ten-minute ordeal. If that happens a few times, you will have a problem with your remote users; they will become more and more dissatisfied with the application's performance. Eventually, they will probably complain to the executive who is writing the checks!

To maintain good relations with that customer, you **must** find a way to reduce those search times, and to increase the chances that the desired record will be found in the first batch of records returned in the first search.

A Solution: Expanding the Concept of a Keywords Subfile

For many years, 4D developers have used the Subfile technology in 4th Dimension to store Keywords that categorize records. For example, a [Companies] file, may have a subfile that stores keywords such as "Customer," "Vendor," "Supplier." Although this same keyword information could certainly be stored in a separate file, most developers use the Subfile technology. This technical note shows you a way to expand this Keywords concept and accelerate the Quick Search process for a remote user.

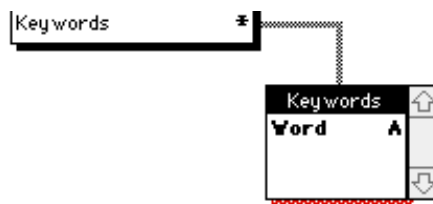
Here are the steps:

Step #1: Decide what fields the user will need to search.

For example, in a [Call History] file the user may need to search by: Contact ID#, Contact First Name, Contact Last Name, Company Name, Follow-up Status, Call Summary Keywords, Sales Rep Name, Tech Support Rep Name and Product Name.

Step #2: Create a [Call History]Keywords subfile with one field: Keyword.

Make the data type Alpha and set the length to an appropriate number. In the accompanying example, we used a length of 30. Index this field.



Step #3: Write a procedure that receives a string and divides (“parses”) the string into separate words.

You will use this procedure when creating keywords, and when performing searches. See the example procedure *KeywordParse* at the end of this technical note.

Step #4: Write another procedure that receives each of the words, checks to see if a Keyword subrecord already exists.

If not, then the procedure creates the subrecord. See the example procedure *KeywordSubCreate* at the end of this tech note.

Step #5: In the data entry layout for that file, call the parsing code and the subrecord creation code in the script of each of your target objects.

In the attached example, whenever any of the following fields is modified, the application calls the keyword comparison/creation routines:

- [Call History]Summary
- [Call History]Company_Name
- [Call History]Contact_Last_Nm
- [Call History]Call_Category
- [Call History]Company_ID
- [Call History]Contact_1st_Nm
- [Call History]Case_Number
- [Call History]TS_Engr_Name



Advantages of the Extended Keyword Search Engine

This Extended Keyword Search technology has several advantages over the traditional "Quick Search" user interface. Let's discuss a few of these advantages.

Advantage #1: The user doesn't need to know anything about the structure of the data.

The user only needs to know one or more pieces of information that the user is searching for.

Advantage #2: The user does not have to use the mouse to choose from a pop-up menu or to click on a check box.

When the user types in a search string and tabs out of the field, the search automatically begins.

Advantage #3: The user does not have to move the cursor to several different enterable variables.

Instead, the user can type a string of keywords into one enterable variable; the system is smart enough to find matching records whether the match is a Company name, Contact name, Summary Keywords, Product name, Customer ID or any combination thereof.

Advantage #4: A new user or an occasional user does not need to know "inside information" about the data.

For example, "The New York Brokerage Company" will appear in the list if the user enters any word of the name in the search variable vFind: New, or York, or Brokerage, or Company, or any combination.

Advantage #5: If the user is unsure of the *spelling* of the customer's name, the user can enter several possible variations of the spelling.

The search engine will find all records that match any of the possible spellings.

Advantage #6: If the user needs to search by either one field or another field, you do not have to expand the dialog and expand the underlying code to accommodate this need.

This type of search is already supported.



4th Dimension

Technical Note 96-26

Advantage #7: You can enhance the search engine for your power users by giving them the ability to enter special characters that will tell the search engine to perform in special ways.

- In the example database, the user can specify an **and** search by typing an Ampersand (&) as the first character of the search variable *vFind*. If the Ampersand is present, the search will only return those records that contain all keywords in the search string. In this way, the user can return a shorter list.

For example, the user could find all call history records pertaining to John Smith of Allied Steel Corporation by typing the following search string:

& John Smith Allied Steel

- The example database also gives special meaning to the Underscore (_) character. When that character is present, it means the search engine will treat the words as a phrase and search for that phrase.

For example, the user can find all call history records pertaining to Cleveland Container Corporation by entering the following phrase in the variable *vFind*:

Cleveland_Container

Advantage #8: You can enable the user to add user-defined Keywords that have meaning to that user.

In this way, the user can have a personal method of retrieving groups of [Call History] records that have special meanings to that user. See the example on the layout [Call History].Input.

Advantage #9: Because so many search operations use the same index, [Call History]

Keywords'Word, that index will always be in 4D or 4D Server's data cache memory.

This means that keyword searches will always take place at the fastest possible speeds.

Taken together as a group, the above nine advantages mean that a search task over a WAN or a 28.8kb modem connection that used to require several attempts and ten minutes to return the desired set of records, can be reduced to one attempt and ninety seconds over the same connection.



Some Cautions About Using Subfiles

Although subfiles are an ideal solution for the problem presented in this technical note, you should be aware of some cautions when using subfiles. Keep in mind that, when the user accesses a record, 4D sends the entire record with all subrecords to the client workstation. If your records become too large, this will increase the amount of time that it takes to transmit the record over a WAN or dial-up connection. Here are some cautions that you should keep in mind when using this technique:

- Limit your subfile to one field, the Keyword field.
- Limit the length of the keyword field. We suggest 30 characters or less.
- Limit the number of keyword subrecords per parent record. We suggest a maximum of twenty-five to thirty subrecords.

Summary

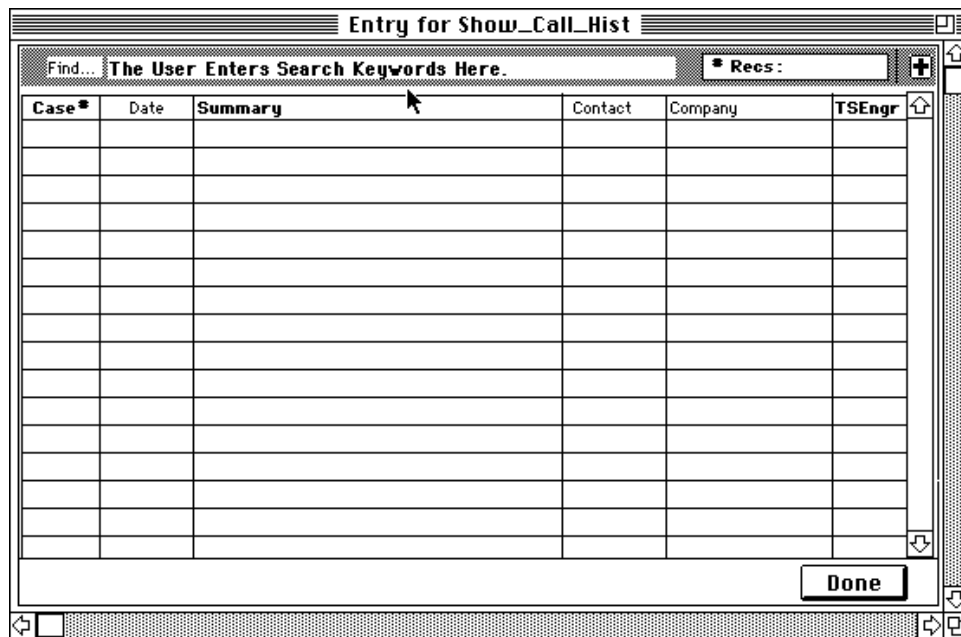
By breaking the rules regarding Subfiles and de-normalized data, you can significantly improve the search performance and power of your 4D applications.

The Expanded Keyword Search technique is a powerful tool that you can use to enable the user to simultaneously search on several fields—and to perform the search quickly and easily. To successfully use this search engine, the user does not have to understand anything about the relational structure of the database. This simplifies the training process for new users. At the same time, this search engine offers a great deal of flexibility to the power user.

Appendix: About The Example Database

The example database *Performance Rules* contains examples of the performance-enhancing techniques covered in Technical Note 96-31 and 96-32. When you launch the database, you are in User mode. To test the Expanded Keyword example:

1. Go to the **File** menu and select **Choose File/Layout**.
2. Select the file [Show_Call_Hist].
3. Click the **Choose** button.
4. Hold down the **Command** key and type `n` to get to the Included Output Layout of Call History.



5. Experiment with searching for keywords such as "crash," "print," and "Walt"
6. Experiment with the special characters Ampersand (&) and Underscore (_).



4th Dimension

Technical Note 96-26

To understand how this technique works, study the following global procedures:

CallHistContLook

KeywordCreate

KeywordParse

KeywordSearch

TextParseWords