

SOPHIEs made easy: Part 5



*Dr Robert
Trehanne-
Jones*

GP, Torquay

This is the fifth part of our journey through the SOPHIE database questions. This month we shall be looking at the detail of \$LOOK and \$FETCH. The examples in this issue will use four-byte Read codes.

As a brief reminder, the so-called database questions are the ones which require no input from the user. By using a database question in your guideline you will be asking the software to look into the electronic patient record and to retrieve some element of their registration or clinical details.

\$LOOK

The \$LOOK question simply scans back through the database to look for the presence (or otherwise) of a Read code, and takes action accordingly. Simple, or what? In fact we are going to only spend the briefest time on \$LOOK for this precise reason—its simplicity means that, in my view, it has very limited use.

Let's see how the thing works in practice. You should know the score by now; click on the LOOK icon on the palette and then drag the box open with your mouse and you'll get something like *Fig. 1*.

Now double click on the icon in the top left hand corner and you get *Fig. 2*.

The field at the top can be filled with a short user-friendly rubric (that's 'a few words', between you and me) to indicate to the end-user what the SOPHIE is trying to do at this point. The field at the bottom is used to complete the code or chapter that you are looking for.

Here's an example; suppose you were to use this question to establish whether your patient was a diabetic or not, and to take action accordingly. In the top field you might type 'Checking for diabetes.....' And in the Read code field you might type 'C2' (without the inverted commas). Here's something important to note; if you are using four-byte Read codes and you were to type in 'C2..' in this field (note the full stops, to give a complete four-byte code) you would only look for the chapter heading code for diabetics. By using only the two characters 'C2' you not only look for the chapter heading but for all subsidiary codes within that chapter as well.

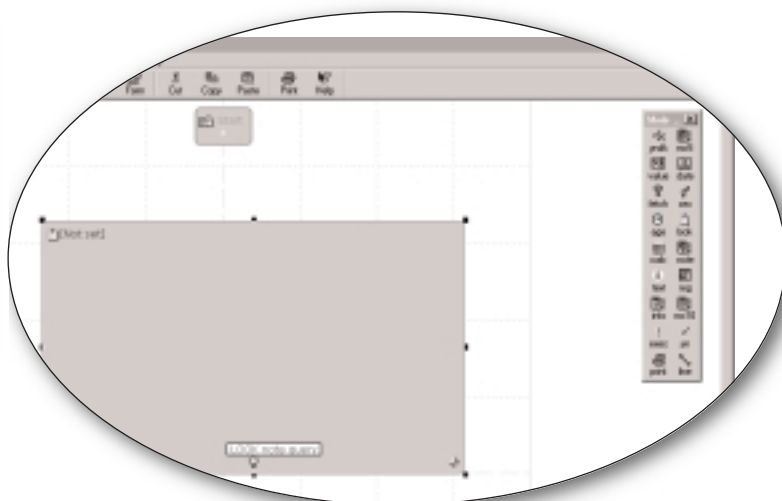


Fig. 1

The exit options are the same as for any other SOPHIE question. If the search finds the code then it exits down the 'Yes' pathway and if the code is not found then it exits down the 'No' pathway.

And that's it! Believe it or not that's all there is to the \$LOOK question—it just says 'if a code is in the patient record somewhere, then do this, and if the code isn't there, then do that'.

\$FETCH

If you've understood \$LOOK then you're halfway to understanding \$FETCH, which is just \$LOOK with bells on. It doesn't take long to realise that you might want to do a little more than just find a code—you might want to refine your search, perhaps to look for the most recent incidence of a code in the record, or look only within a specific time frame. \$FETCH allows you to do all this and much more.

Users with long memories will remember a question called \$GET which appeared in the first version of SOPHIE back in 1988. The superbly crafted \$FETCH option rendered \$GET completely obsolete, while \$LOOK only just managed to hold on in very limited circumstances.

Let's click on the \$FETCH icon, open the box, and see what we get after we have double-clicked on the icon at the top left hand corner once again (*Fig. 3*).

You will see that by default you are looking at the tab labelled 'Note details'; there is a second tab labelled 'Validation' which we shall come to in a minute. This first tab requires you to define the code or codes you are looking for, in very much the same way as you did with \$LOOK. The 'Prompt' field at the top is where you type your user-friendly string, such as 'Checking for smear result' and the Read code field is used in exactly the same way as before. '4K2' will find all smear results in that chapter (assuming your surgery uses this chapter of codes for recording smear results!) while using '4K2..' in this field would look only for the chapter heading code and would therefore be unlikely to generate the desired result.

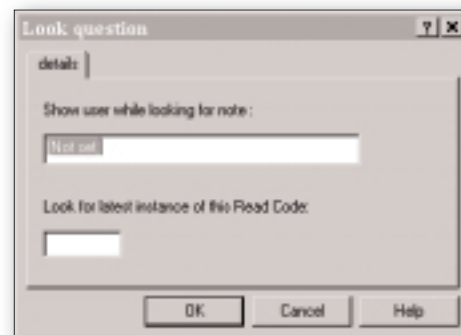


Fig. 2

The field labelled 'Which note to get' defaults to '1' to indicate the most recent occurrence of that code in the patient record. By using the figure '2' in this field you can look for the last-but-one occurrence, and so on. Why might you want to look for any code other than the most recent, I hear you ask? Well, it is my common practice, when designing a diabetic monitoring SOPHIE, for instance, to have three \$FETCH questions in quick succession, all identical apart from this field. By labelling this field 3, 2 and 1 in that order I can get the SOPHIE to flash up the three most recent blood sugars, for instance, at the top of the screen, to give the user a brief snapshot of the degree of diabetic control for this patient. You might do the same thing with HbA1c results, and I'm sure you can think of other examples for other disease management plans.

Now this is where it starts to get interesting! It doesn't take a rocket scientist to work out that a number of the codes you will \$FETCH into your SOPHIE have numeric values attached to them—the blood sugar values just quoted are an obvious example. What if you could save that value to use later on in your SOPHIE, for example as part of calculation question? Well, the good news is that you can, and here's how.

The field labelled 'Variable to hold value' defaults to the letter 'F' to indicate that this question will take any numeric attached to the Read code

you're looking for and save it as variable F for use later on in the SOPHIE. In fact you can save it as virtually any letter of the alphabet you choose, but there are some important exceptions. Some of these are displayed when you click the down arrow to the right of the box to open up the full alphabetical list, such as 'A' which is always used to mean 'Age', and 'S' and 'D' which are always used to mean 'Systolic' and 'Diastolic' blood pressure. It messes things up considerably if you use A, S or D for any other variable! There are a couple of others which, while not being absolutely forbidden, are used by so many SOPHIE developers as to be almost sacrosanct—these include 'H' for 'Height' and 'W' for 'Weight'.

Apart from these you can use any other letter you like to save as your variable, but do remember that if your SOPHIE starts to get complicated, with a number of variables, then it makes very good sense to make a note to yourself of which letters of the alphabet you've used to save what variables. You can either use a piece of paper for this purpose, or the built-in authoring notes screen which is available at the bottom of the View menu in the graphical editor.

Still with me so far? Good, because there's a tekky warning on the next paragraph! It concerns the one remaining field on this tab called 'Which value to get'. Vintage users will recall that in the very early days of System 5, in versions up to and including 5.4, it was possible to have more than one numeric attached to a particular Read code. In these instances it is possible to use this field to determine which of those numerics you actually select to save as your variable. We're talking about some very old records here, and the chances of you wanting to use this option are about as close to zero as you can get. I've never used it, but I'm including it for the sake of completeness.

OK, so you can remove the cold towel from around your head, as we move into the 'Validation' tab (Fig. 4).

This is the area where we determine the time frame where we want the search to take place, and with a little bit of intuition you can see that the values default to a time frame looking back for a period of three months. Click on the 'Date units' window arrow to see that it also offers an unsurprising choice of days, weeks or years. The numeric fields and radio buttons to the right of this box allow you to determine the start and end points for your search—the default values start at 'three months before today' and finish at 'zero before today' although it won't matter which radio button you have clicked to determine finish, because 'zero after today' means exactly the same thing.

The only rule here is that the 'From' date must be chronologically earlier than the 'To' date so that if you wanted to look between one and two years ago you must set up the fields to read 'From two years before today to one year before today'.

It's time for another important tip, where many developers have come unstuck over the years, and that is the

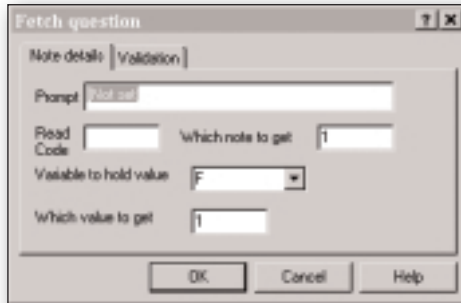


Fig. 3

relationship between the time frame endpoints and the 'Which note to get' field. A note entry must fulfil both selection criteria to be displayed and used for decisions. You cannot say, for instance, 'I want to look at a certain time frame and then see how many codes there are in that frame, and then pick out the last-but-one of those codes—it doesn't work like that. What you can do is tell the SOPHIE to find the last-but-one occurrence of the code, and

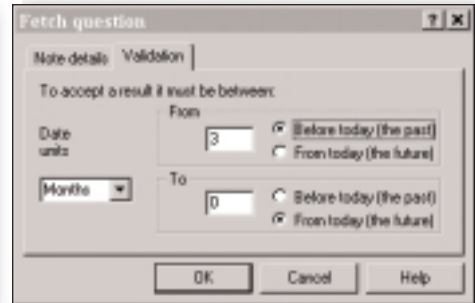


Fig. 4

determine what happens (Fig. 5).

Let's finish with a puzzle—there is a glaring problem with the simple scheme I have outlined here. Can anyone spot it? I'll have a word with the Editor and see if we can come up with a prize, even if it's only a free drink at conference!

Next time we'll be looking at the \$CALC question before moving on to the questions which require patient input.

- | | |
|------------|--|
| 1. \$AGE | Less than 25 → GOTO Q.2
More than 25 → GOTO Q.3 |
| 2. \$FETCH | Find height within last 12 months
Yes - calculate patient's BMI using values
No - prompt user to measure patient's height. |
| 3. \$FETCH | Find height within last 10 years
Yes - calculate patient's BMI using values
No - prompt user to measure patient's height. |

Fig. 5 A schematic SOPHIE—but what's wrong with it?

then if that occurrence does occur within your time frame markers then it will count as a hit.

So much for the heavy theory, so let's have some practical examples. Many users have a simple SOPHIE on their systems which uses a combination of height and weight to work out the BMI (using a \$CALC question, which will be described in the next issue) and code it accordingly. You stand the patient on the scales, weigh them, bang in the 22A code, and hook the SOPHIE, which will almost certainly use a \$FETCH question to look for the patient's height.

What are the important factors in taking a height code and working with it? Most humans stop growing in their early twenties, so for patients over a certain age you might use the time frame limits in your \$FETCH question to find a 229 height code from some time ago and use it in your calculation. The same is not true for adolescents, where a time frame of a year might be appropriate. Already your simple SOPHIE is beginning to grow, as SOPHIEs do very easily, in my experience!

You will need to begin with an \$AGE question to determine which of several \$FETCH questions the SOPHIE uses to find a height code, and the major difference between each \$FETCH will be the time frame.

Let's use a schematic diagram to

System 6000—velocity coding

Velocity coding sorts the list of Read codes according to frequency of use, which means that afterwards you don't have to scroll down the pick list to find common items. It saves ages. Set it up as follows:-

1. Log in to the UNIX server as root.
2. Type: `/medapps/s6000/utills/db/velcode` and press <Return>
3. The system asks if you wish to run velocity coding—answer 'Y' if you are sure.
4. You are now asked how you wish to run Velocity Coding:
 - A) Now (All users must be logged off the system already)
 - B) Automatically Scheduled for this coming Sunday morning. Select either A or B as desired.
 See the Notes and Problems section of the electronic System 6000 Clinician help file for more details and warnings.

Premiere—Tab key

Use the Tab key to cycle round the various fields that can be highlighted, such as Date, etc.