file placeholders for Postscript and bitmapped fonts found in *1-2-3* version 2.3, even though the program supports Bitstream's font-scaling technology exclusively. Get past this annoyance and the fact that *1-2-3 for Home* includes only the Swiss (Helvetica equivalent) typeface—not even Dutch (Times Roman) is included—and you'll find a well-integrated product that uses just two megabytes of disk space (plus 800 kilobytes for the installation library, and 500K for the entire set of SmartSheets). *1-2-3 for Home* needs at least 350K of standard memory to run, and can use expanded memory to store worksheets as you work on them.

Remember to back up your diskettes before installing them (surprisingly, Lotus' policy of writing to the first one in the set is documented, although at the back of the manual), and don't attempt to use the database functions (remember, the matrix is only large enough for 500-record databases), and *1-2-3 for Home* could turn out to be the ticket you've been seeking to the spreadsheet party.

1-2-3 for Home
Lotus Development Corp.
55 Cambridge Parkway
Cambridge MA 02139-9168
617-577-8500
$149

**Approach**
Approach Software Corporation

If you've grown tired of waiting for a viable *Windows*-based database program, we have good news. With the release of *Approach*, your wait is over. You can now create reliable, relational applications with very little effort, and as a bonus, the data type you use is a matter of preference—*Approach* supports *dBASE* (III and IV), *Paradox*, and *Oracle* tables. It's no professional development environment, but it's very, very good, nonetheless. Check out *Approach*.

It's important that you be comfortable with the “non-professional” point. Although you may very well enjoy the simple process involved in laying out *Approach* applications, programmability is limited to whatever formulas you can embed with *Approach*'s calculated fields. This means that there is no such thing as tweaking code that behaves in a manner not quite to your liking, and no command-based, ad-hoc data query. You can't even order *Approach* to index data; the process is carried out automatically when you join two or more databases relationally. *Approach* is designed for form-based application building, and if you want a traditional programming language, will disappoint.

Looking at *Approach*, you get a good feel for why nobody else has yet delivered a decent database program for *Windows*; it's a tough product to design. *Approach* attacks the dilemma by simplifying the issue. You need to learn how to design a data structure, of course, but once you've mastered that art, almost everything you do in *Approach* is easy. By current *Windows* standards, *Approach* presents itself to you in the form of a clean screen, letting you place the elements you want included in data forms wherever they suit you, and fettered only by a single row of logically-grouped buttons that change as you switch between design and browse modes.

You can use fields, graphic images (as both data and design elements), graphic primitives, and controls such as check boxes and radio buttons to dress up forms, and once you've fashioned your interfaces, *Approach* the data retrieval tool is navigated with nothing more than VCR-action buttons, one to indicate your desire to initiate a search, and three that add, commit, and delete records (see figures 1 and 2). Both designing forms and using them afterward to formulate queries is very easy. As strong as *Approach* is here, though, there are a few glaring layout inadequacies; the unalterable snap-

to grid is always invisible, users who opt to display the ruler will find that it automatically turns itself off every time they leave design mode and go to browse, and when you drag an object