Simple net.db-ing

Sven O. Rimmelspacher

Implementing applications with net.db isn't a tough job at all. In fact, this program may help you eliminate some of those programming chores you've been putting off for no good reason. Here's a practical example.

When people talk about net.db, they often say something like, "Yes, it's a good product, but we and our customers don't have any catalogs or products that we want to show on the Internet, and therefore we have never used it." But when you look at it from another side and think about what net.db can do for you as a system that supports your development issues, you'll find some nice ideas that are easy to implement and can really help you. In this article I'll give you two examples of how we use net.db and how it helps us in the development processes.

How it started

Recently I came to the conclusion that a knowledge base could help in our development cycle. When I thought about how much time I had to accomplish this task, I decided to try a net.db application. (Also, I have never used net.db before and I needed to learn it.) It's simple, can be created in minutes (or at least hours), and can contain everything we need—even business graphics. So I started to create a simple database structure that would fit our needs. It looked like the one in **Listing 1**.

Listing 1. The first table design for the knowledge base.

```
create table knowbase
(
  inputdate datetime not null,    // date/time when entered
  id char(8) not null,            // who has entered it
  category smallint not null,     // in which category fits this entry
  symptom char(254) not null,     // what is the symptom/error/question
  solution char(254)              // what is the solution
);
```

Listing 2. The final table design for the knowledge base.

```
create table knowbase
(
  serial integer not null,        // unique key for each row
  inputdate datetime not null,    // date/time when entered
  id char(8) not null,            // who has entered it
  category smallint not null,     // in which category fits this entry
  symptom char(254) not null,     // what is the symptom/error/question
  solution char(254),             // what is the solution

  primary key (serial)            // definition of primary key
);
create unique index knowbase_1 on knowbase (serial);
```

Some minor problems

I created the table in our SQLBase database in the intranet. Then I started net.db and the page wizard, which led to the error message, "SYSADM.knowbase: This table has no primary key—no page will be created." All right; net.db needs a primary key in order to "wizard" on the table. When I looked at the data structure, I knew I was in trouble because identifying a unique row would lead to a primary key that would be too large (in other words, more than 255 bytes as it is defined in SQLBase). So I had to add an artificial key, which I called serial. For a simple solution to creating a unique key for each new row, I used the sysdbsequence feature of SQLBase. Finally, I had my correct database table design, as shown in **Listing 2**.

More than one table

This time the page wizard was successful and it created my new book.

The ID field should hold the user ID, so we can ask the one who has entered the data if we have further questions. The idea of the category field was to provide the possibility to categorize the data in groups like "error", "question", "feature", "wish", "idea", and so on (these categories should be taken from another table declared below). Finally, the symptom and solution columns were meant to be used for the actual information of the entry.

Searching

The first page I wanted to create was the search page. The default fields were the serial and ID fields. Neither was what I wanted, so I deleted them and created a field for the symptom

column where the user could search. The second search criteria that seemed to make sense was the category; here I wanted to use the category texts (see above) and store the category number, both from the knowcat table. I had no idea how to do this, so I just started the Component Assistant, which led me through this issue. It asked the following:

1. Choose a component from the list? I picked Auto Combo Box.

2. The name, title and data source column? For the column I chose knowbase.category.

3. Which Datasource table should fetch the content of the ComboBox from? I selected sysadm.knowcat.

4. Where to populate the combo box from? It needed to know which value should be written to the data source column and which one should be displayed. So I selected knowcat.category and knowcat.cattext, respectively.

After that I previewed the search page. It contained a combo box with all my values from the knowcat table and an additional [all] entry. Wow! I only had to adjust the position and then I was done!

Browsing

The next step was the browse table. Previewing wasn't possible because no data was found in the table, so I inserted an entry manually. After that the browse table still couldn't be previewed, since net.db immediately shows the detail screen if it finds only one entry for a selection. After inserting a second entry I could finally preview the browse table. It also contained the serial and id columns, which I didn't want; so I removed them and added the category, symptom, and solution columns.

Once again I had the problem that I wanted to display the category texts from the knowcat table; but this time the Component Assistant couldn't create a combo box, since we had just a table. A little browsing through the items in the left pane of the net.db designer showed a Table Joins Wizard. OK. Let's see what this one could do for us . . .

1. I wanted to create a new join. I selected the knowcat table in the Joins to table combo box.

2. The alias isn't important; I just used c.

3. On the next page I had to select which columns make up the join. I selected category from both tables.

Listing 3. The category table for the knowledge base.

```
create table knowcat
(
  category smallint not null,     // the category key
  cattext char(32),               // the text for this category

  primary key (category)          // definition of primary key
);
create unique index knowcat_1 on knowcat (category);

insert into knowcat values ( 1, 'Error' );
insert into knowcat values ( 2, 'Question' );
insert into knowcat values ( 3, 'Wish' );
insert into knowcat values ( 4, 'Idea' );
insert into knowcat values ( 5, 'Feature' );
```

After selecting the properties from the category column and c.cattext (which now existed) in the datasource column, I previewed the browse table again. Now it showed the text instead of the numbers.

Details

My next requirement: to work on the detail screen, which is used for viewing details of a data set and entering new data.

The serial number used for each entry shouldn't be visible, because we can fill it on a new entry automatically. This can be done on the Properties | Advanced page for this field, where you can find a field, "Default text values or SQL sequence." As I've mentioned, we can use the sysdbsequence feature of SQLBase here, so I entered, "SQL:SELECT sysdbsequence.nextval FROM knowbase." This will get the next sequence number from SQLBase each time a new row is to be entered.

Once again I had to create the category combo box, but this time I already knew what to do; it was actually exactly the same as for the search screen.

The last thing was the automatic fill of the date field with the current date when a new data set is created. Once again a SQLBase function helped me here. I just entered, "SQL:SELECT @now FROM knowbase," in the default text field of the properties for this field.

Finally

Now everything was ready to start. I created a link to this page (the URL for this link can be found in the advanced properties of your page), started it, and entered my first data sets. Everything worked fine. I was amazed at how fast and easy this all was. I had worked only for about an hour on this application. I used the rest of the day for designing some header, background, and footer files for these pages and also for a simple description on how to use this stuff, which actually needed much more time than the application itself.

Additional features for this application could include adding a graphical evaluation, say, a GROUP BY category to find out how the entries are categorized or even an image field where you could add screenshots of errors or rapid prototyping ideas.

What next?

After this successful first use of net.db I created other such applications; I think there's one more that could be of interest to you—the SDK constants application.

Often in the newsgroups people ask for the value of an SDK constant. It's sometimes hard to find these values, especially if you don't have a development system, like Microsoft Developer Studio, that comes with them. So I extracted all constants (I found 4,161 of them) from the C header files and put them into a single file in the format:

```
Constant=Value
```

Then I wrote a very simple application that reads this file, splits the lines in a constant and value part, and inserts these tokens in a database table that has the structure:

Listing 4. The table for the SDK constants application.

```
create table sdkconst
(
   name char(64) not null,
   value char(64),
   primary key (name)
);
create unique index sdkconst_1 on sdkconst (name);
```

This time I knew what I had to do. Five minutes later I completed a net.db application that enables the user to browse or select the SDK constants. Since Explorer is usually always running on my desktop, I have immediate access to these values.

Think about it

These applications aren't very sophisticated, but they do show you what you can do with net.db besides the catalog stuff. If you think about what net.db can do for you in your development issues, you'll discover several other applications that really can help you reduce time for jobs that are small and annoying (and therefore not done even if they should be). **CP**

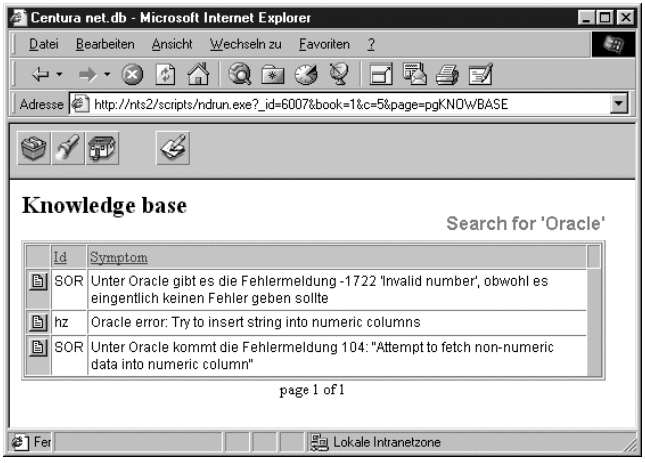

Figure 1. Sample selection in the net.db knowledge base application.

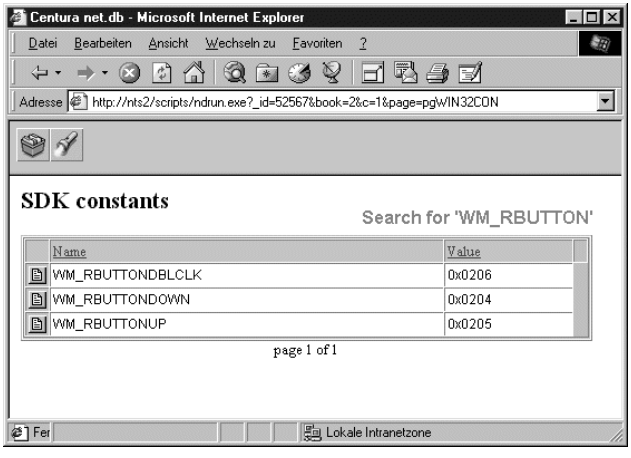

Figure 2. A sample from the SDK application.

Sven O. Rimmelspacher is a senior developer and project manager at Pickert & Partner GmbH, a company that has its own custom quality management system. He is also the author of DocSal, a documentation tool for SAL code. He can be reached at sven@rimpi.de.