SQLWindows® Solo

Quick Start

20-2135-1001

GUPTA™

Trademarks

Copyright

SQLWindows Solo Quick Start

20-2135-1001

August 1994

Authors: Susan Wilson, Denise Tindell, Sonya Pelia, Patricia Wright, Smokey Cormier, Vik Chaudhary. With Technical Evangelist Rob Martin.

Contents

The Quickest Route to Client/Server

You have in your hands a guide for the quickest route to client/server productivity: SQLWindows Solo. SQLWindows Solo is the development environment of Gupta's SQLWindows 5, with a few minor modifications. The database is limited to a maximum of 5MB, and network SQL connectivity is not included.

SQLWindows 5 is the only quick and powerful client/server development system. SQLWindows 5 includes QuickObjects, powerful pre-built objects that make even novice programmers immediately productive and slash the time it takes to create functional applications.

This Quick Start guide will get you up and running with SQLWindows Solo in no time at all. To make your start even easier, we recommend a "test drive," which is the purpose of chapter 1.

Test Driving SQLWindows Solo

Once you've completed the test drive by building a working SQLWindows application, we invite you to move on to chapter 2, Creating Object Classes, where you will learn how to create re-usable, extendable object classes based on objects you created in chapter 1. You will see that SQLWindows is truly an object-oriented programming environment, and that SQLWindows delivers on the promise of re-usable code, full object-oriented extensibility, and multiple inheritance.

Chapter 3, *Roadmaps*, helps you explore other regions of the SQLWindows development environment. We explain SQLWindows Designer, taking a closer look at all the tools available in SQLWindows 5. There is complete information on all of the QuickObjects, intelligent, pre-defined objects that provide you with a fast and easy way to create client/server applications. Object-oriented programming concepts and advantages are explained in detail, and the SQLWindows Team Programming features (available in SQLWindows 5 Corporate Edition) that help you keep track of large scale projects involving teams of developers are fully described.

Gupta SQLWindows 5 puts you on the fast track to development with QuickObjects and gives you the power and productivity to deliver high-performance client/server applications. As part of the Gupta Client/Server suite, which includes the SQLBase database engine and SQLNetwork connectivity products, SQLWindows is your best choice for getting started quickly with client/server development -- and the only choice for getting client/server done.

Chapter 1

Test Driving SQLWindows Solo

Hello and welcome to the SQLWindows Solo test drive. I'm your navigator, Susan. I'm here to help you get behind the wheel and to point out the powerful features of SQLWindows Solo as you do the driving. Every adventure has a story behind it. You are the lead character of this story, embarking on the most important journey of discovery in your development career. You are test-driving SQLWindows Solo, the quick and powerful client/server application development system for Windows.

SQLWindows Solo includes a powerful new architecture, QuickObjects. QuickObjects are tables, windows, push buttons and other smart objects that you use to build applications by pointing, clicking, dragging, and dropping. With QuickObjects, you can create many complete applications without writing any code.

QuickObjects are all the application objects you need to save the day for the character in distress in the test drive story. Our cousin Chris works for a Hawaiian shirt company that has an urgent need for a sales order and history application. Chris just called me and the conversation went like this:

Chris: "Susan, my order application isn't running very well on the new LAN that was installed. It's very slow, and sometimes just hangs."

Susan: "That's not good."

Chris: "I know. I need an order application that was made to work on a client/server PC LAN. You've been telling me about the power and performance of SQLWindows. Didn't you say it was created from the beginning for PC LANs and client/server?"

Susan: "Yes, SQLWindows was created from the ground up with the easy power and performance to get client/server done."

Chris: "I really need help today. Can you build a true client/server application that will manage my orders?"

Susan: "No problem, I'm taking a test drive right now with someone who wants to get to know the SQLWindows 5 client/server application development environment. With QuickObjects, our new person can get the application done in short order, without writing any code, by pointing, clicking, dragging, and dropping QuickObjects."

Chris: "I knew you had some powerful client/server software at Gupta. Call me when it's finished."

Now you know the story. Let's get started.

As you work through the test drive, we introduce features of SQLWindows like the tool bar, an MDI window, and the tool palette. These features are discussed in more detail in Chapter 3, *Roadmaps*, and are also briefly defined in the *Glossary*.

Install SQLWindows Solo

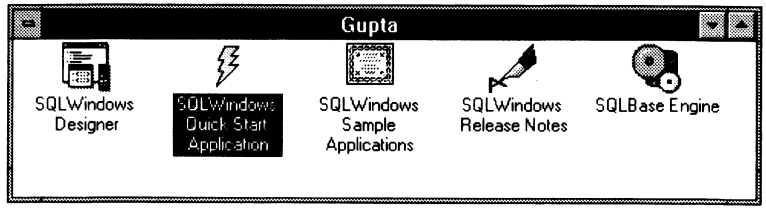
❑ To install SQLWindows Solo from disk.

1. Start Microsoft Windows.
2. Put Disk 1 in your floppy disk drive. Normally, this is Drive A, but may be different for you.
3. Select the **Run** command from the **File** menu in the Program Manager in Microsoft Windows.
4. Type: **`A:\SETUP`**
5. Click **OK** or press **Enter**.

❑ To install SQLWindows Solo from CD-ROM.

1. Start Microsoft Windows.
2. Put the CD in your CD-ROM player. Normally, this is Drive D, but may be different for you.
3. Select the **Run** command from the **File** menu in the Program Manager in Microsoft Windows.
4. Type: **`D:\SETUP`**.
5. Click **OK** or press **Enter**.
6. Click the **Quick Installation** push button in SQLWindows Solo Setup to install Solo. (Click the **Deploy** push button if you want to create your own database with SQLBase and ODBC database access).

   SQLWindows builds the Gupta Program Group for you when the install program finishes. The program group looks like this:

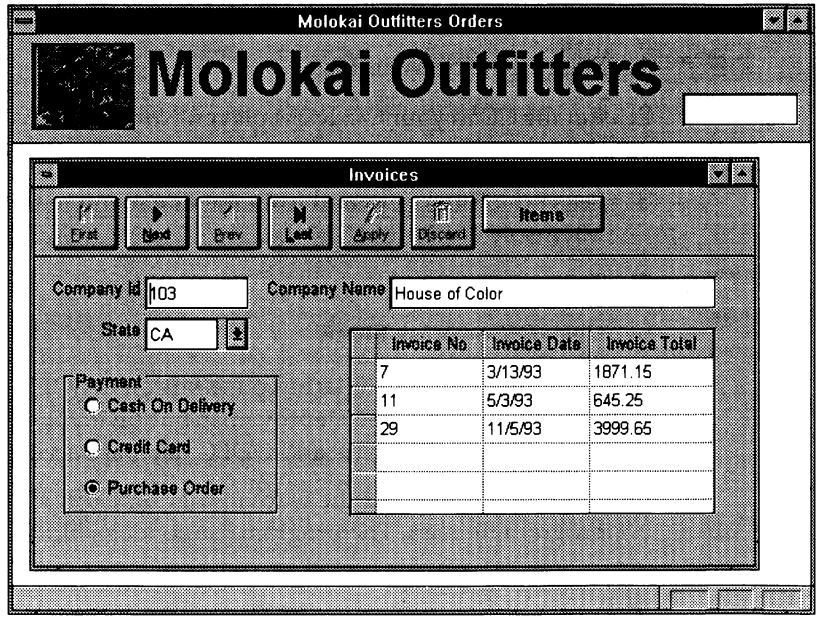


Take a look at the Quick Start Application

You're ready to start using SQLWindows Solo! First, let's take a look at a finished version of the application you build in this test drive.

1. Double-click on the SQLWindows Quick Start Application icon that looks like this:

The Molokai Outfitters Orders application opens. You guessed it. Our cousin Chris works for Molokai Outfitters in Hawaii.

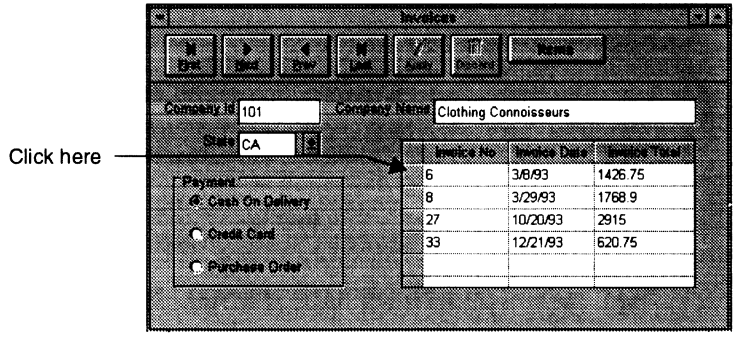


This is the order application you are going to build.

2. Click the **Next** push button. Use the **First**, **Next**, **Prev**, and **Last** push buttons to scroll through the available data. The table on the lower right shows the active orders for the company name on the screen.

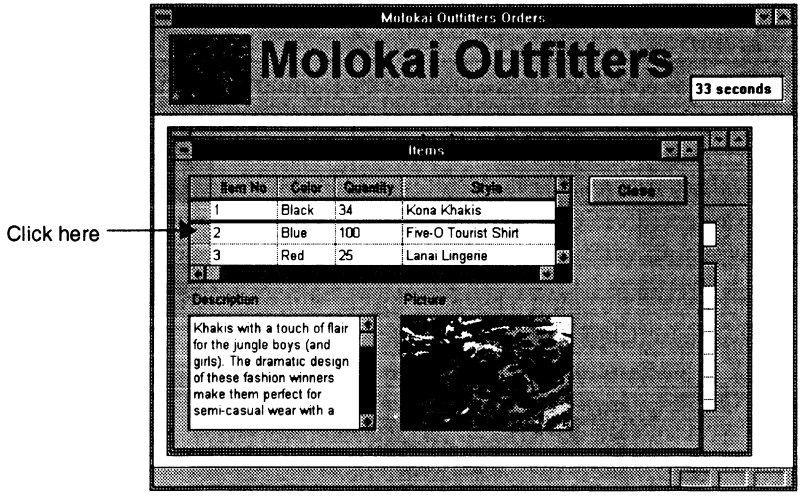
3. Click **Next** until you display the **Clothing Connoisseurs** company name in the Company Name field above the table.

4. Click the first invoice row in the table to highlight.



5. Click the **Items** push button to open the Items window.

   This screen is designed for entering and updating order details.

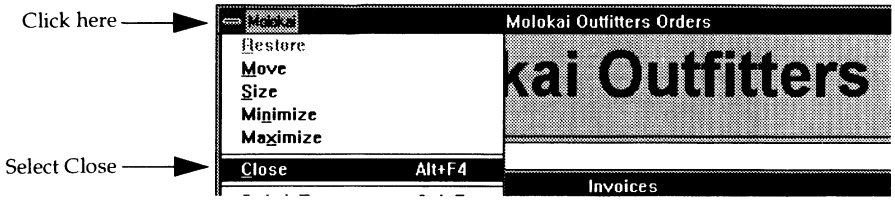


6. Click on the second line item. The picture and description of the garment change.

7. Click the **Close** push button to return to the Invoices window.

   Now, you can see the kind of application you can build quickly with SQLWindows. You know where the test drive leads. Let me show you how we'll reach our destination. Close this sample application and start building your own version.

8. Select **Close** from the System menu in the title bar of the Molokai Outfitters Orders window.



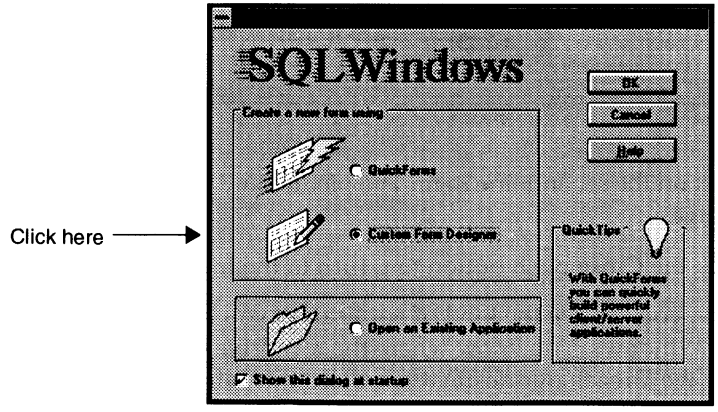
Build your main application window

You're ready to start building a fully functional SQLWindows Solo application. In this first section, you start SQLWindows Solo and build the main window.

1. Double-click on the SQLWindows Designer icon in the Gupta program group.

   The SQLWindows banner window and the SQLWindows QuickForms window open. QuickForms is a new, powerful feature of SQLWindows that automatically builds a complete form for you in seconds. Just specify all the fields and tables you want to build, and which database you want to access, and SQLWindows builds the form graphically right before your eyes, and writes the code for you behind the scenes.

2. But we are building a custom application today. So, click **Custom Form Designer**. Click **OK**.



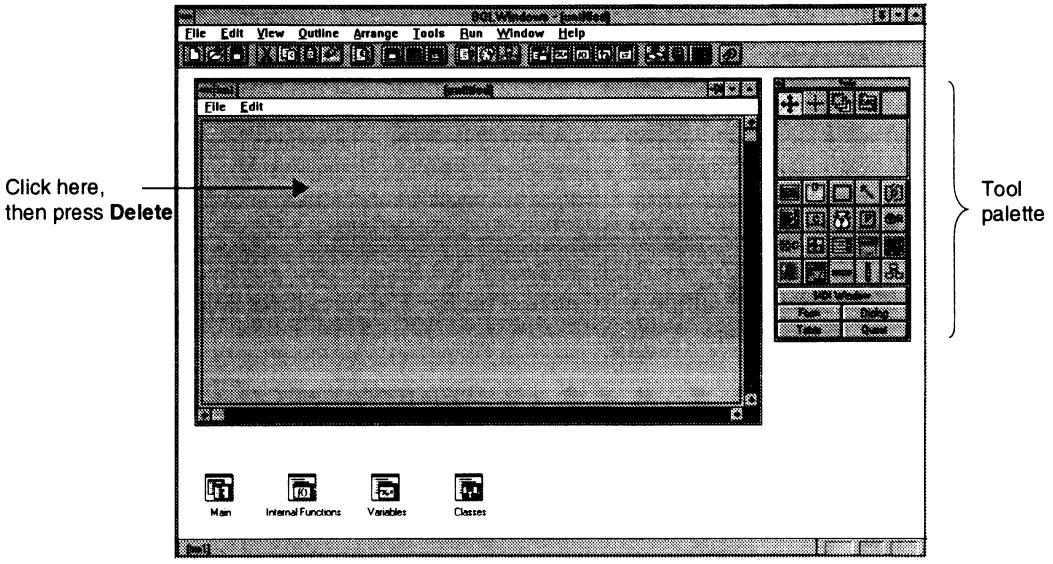
An empty form window and the SQLWindows tool palette appear. SQLWindows anticipates your development requirements and

automatically provides a form window for you to work with. However, in this application, we want to start with an MDI window.

A form window can contain myriad objects that perform regular application functions, such as retrieve, display, and update data. An MDI window is an expandable window or workspace that you use as a frame to contain flexible forms and top-level table windows.

So, let's delete the default form window SQLWindows automatically provides, and create an MDI window.

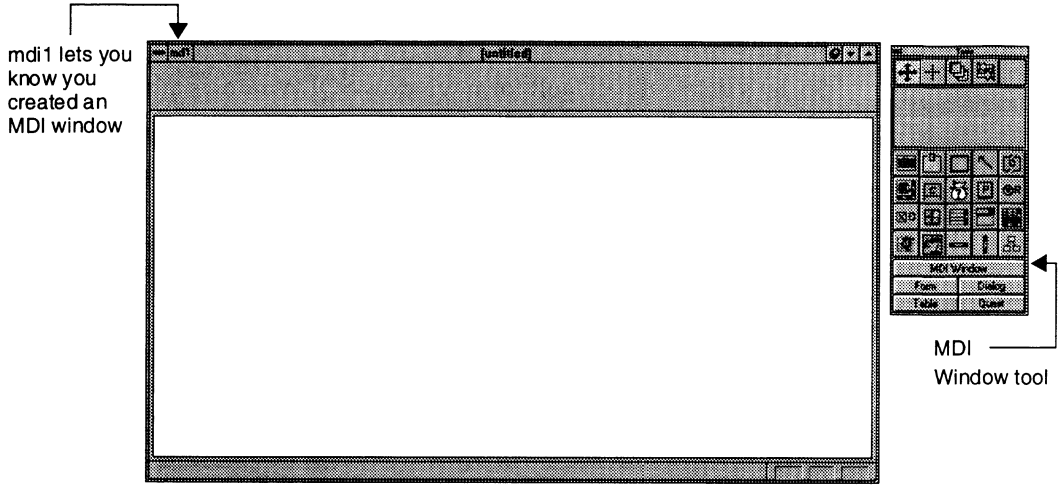
3. Delete the empty form window by clicking on it and pressing the **Delete** key.



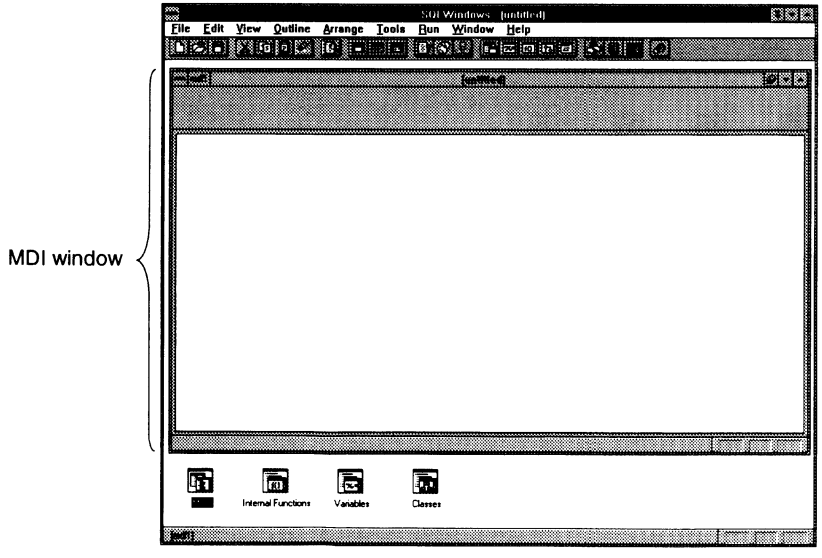
Tip: To move the tool palette, click the bar at the top of the palette and hold down the mouse button, then drag the tool bar to a new location and let go. Press the **F4** key to bring back the tool palette if it is not visible.

4. On the tool palette, click on the MDI Window tool.

   This action creates the MDI window for you. The MDI window is where you will build the various forms for your application. This picture shows you what the untitled MDI window looks like:

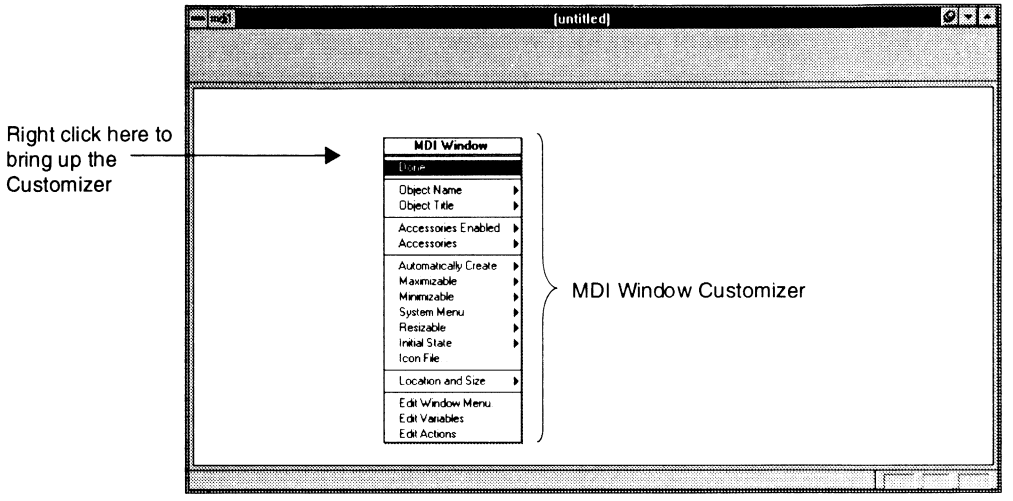


5. Resize the window to make it fill the SQLWindows Solo workspace, as you see here.



To resize the window, click on it. Then, move the cursor to the edge until it becomes a small arrow. Hold down the mouse button and pull. Now you can enlarge the window to the approximate size shown in the picture.

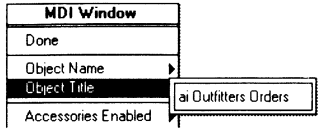
6. Place the cursor over the center of the MDI window and click the **right** mouse button to bring up the MDI Window Customizer menu.



Tip: The SQLWindows Customizer is a menu, unique to each object, where you can specify attributes of an object, such as name, color, and actions to perform. To bring up the Customizer menu for any object, place the cursor over the object and click the right mouse button.

7. Select **Object Title** from the MDI Window Customizer menu.

8. Type **`Molokai Outfitters Orders`** and press **Enter**.



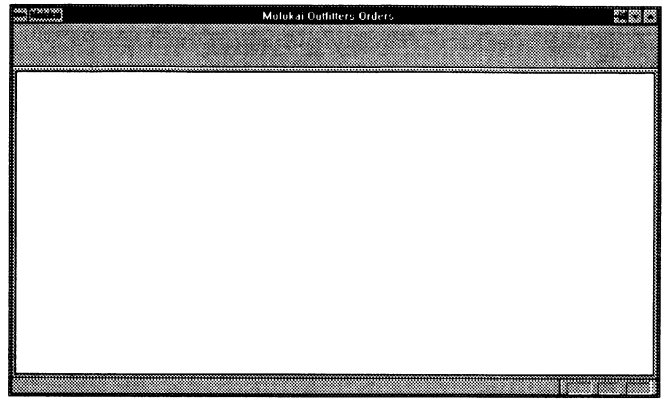
Tip: It is important to type the name exactly as shown, with the first letter of each word capitalized. We will call this object by name from other objects in the application. Please use the exact name(s) provided in this tutorial for all of the windows and objects.

9. Select **Object Name** and type **`Molokai`** and press **Enter**.

10. Click **Done** in the MDI Window Customizer.

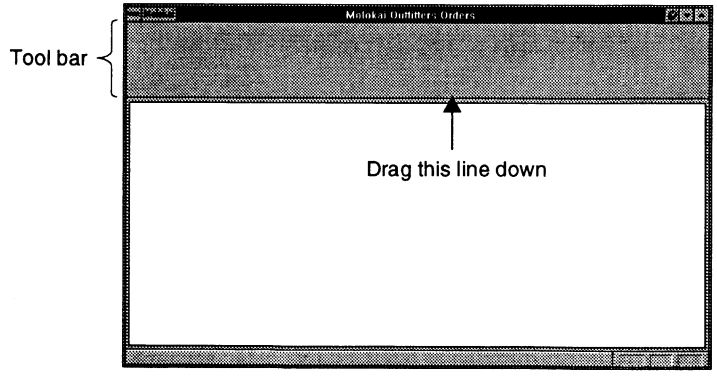
Tip: If you make a mistake and need to remove an object from the application as you build it, you can delete it at any time. Just select the object, then press the **Delete** key.

The object you have just named looks like this. Note the changes to the title bar of your MDI window:

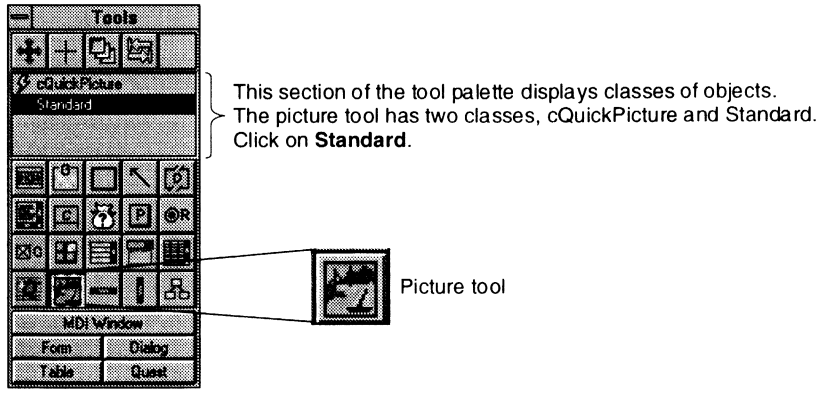


Now, let's add the company logo and banner to the tool bar. First, make the tool bar big enough to fit the objects you want to build.

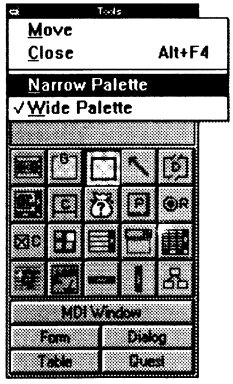
1. Click on the tool bar area.
2. Put your cursor on the bottom line of the tool bar, hold the left mouse button down and drag downward. Now, you have enough room for your objects.



3. Click on the **Picture** tool in the tool palette. Click the **Standard** class in the tool palette.

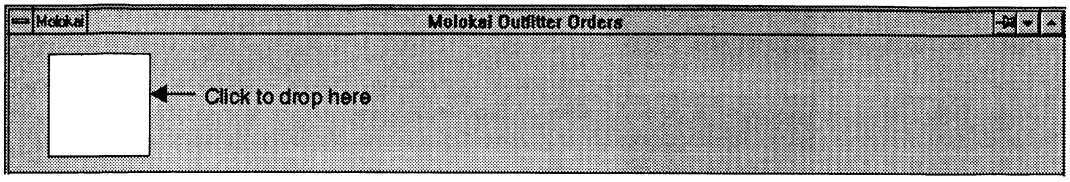


Tip: This tool palette is shown in the **Wide Palette** view. You can change the view by clicking the **Control** menu of the tool bar and setting it to the **Narrow Palette** view.



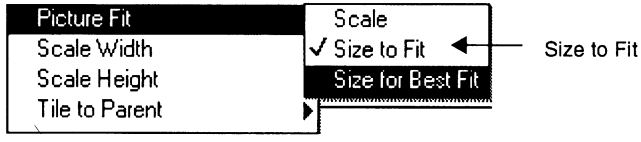
4. Move the cursor over to the left side of the tool bar.

5. Click to drop the picture object on the tool bar. Resize it to look like the object in the following picture. To resize it, hold down the mouse button and drag the object's edge before letting go.

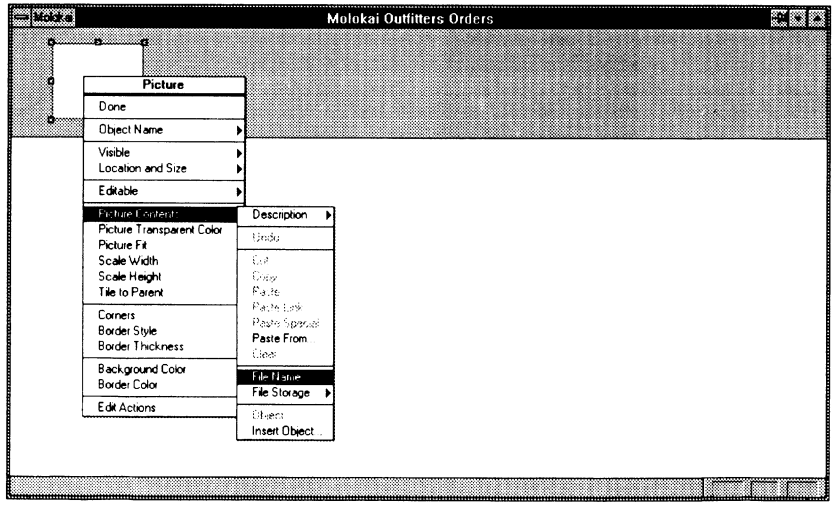


Now, let's populate your picture object with a picture. We use the Customizer.

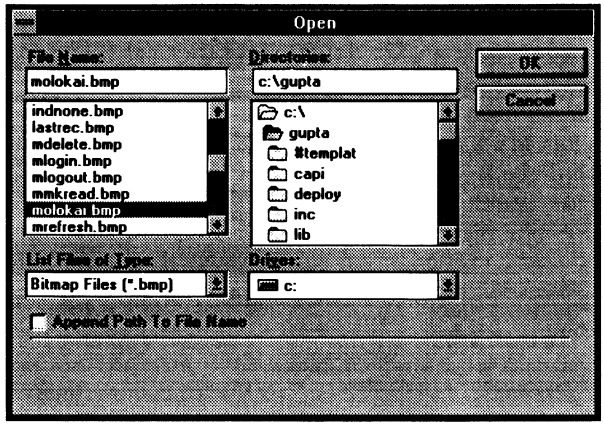
6. Click on the picture object to select it.
7. Right mouse click on the picture object to bring up the Customizer menu.
8. Select **Picture Fit**, then **Size to Fit**.



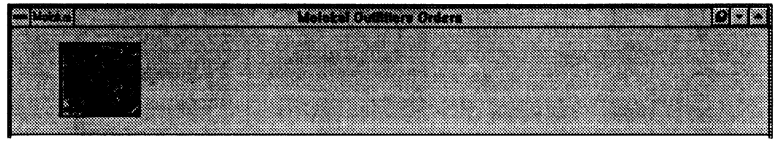
9. From the Customizer, select **Picture Contents**, then **File Name**.



The **Open** dialog box appears. Here, you choose the company logo bitmap to display in the picture object.

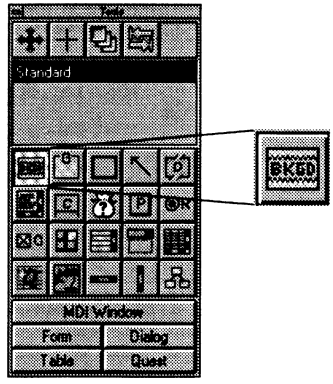


10. In the **File Name** list box, scroll down to find the file named **molokai.bmp**. Click on it to display it in the **File Name** data field. Click **OK**.



The picture appears in your picture object. Now, let's add the text object for the company name.

☞ 1. From the tool palette, select the Background Text tool (BKGD).



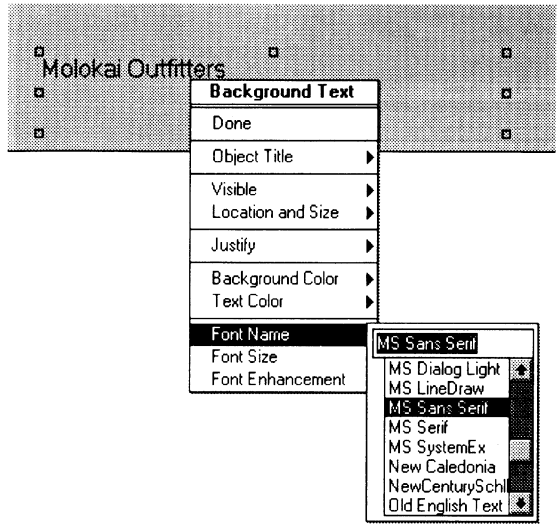
Background Text tool

2. Drop the background text object onto the tool bar to the right of the logo bitmap. This creates a "window" where you can type the banner text.

3. Type **`Molokai Outfitters`** and press **Enter**.

4. Resize the Molokai Outfitters text object so you can see the entire string. Do this by clicking on it and dragging the handles.

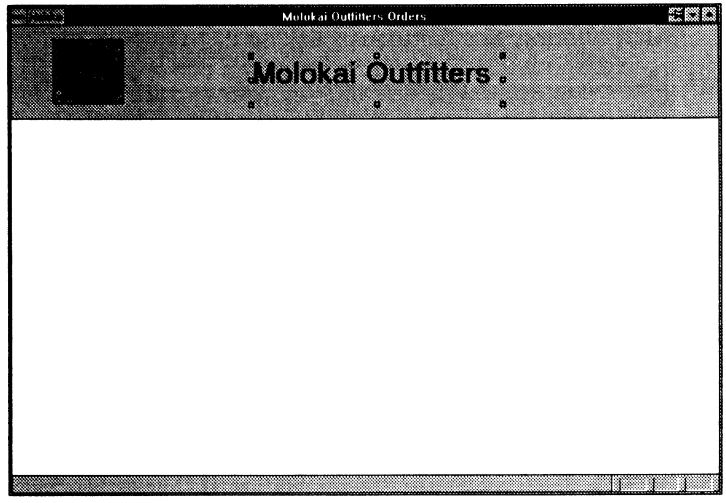
5. Right mouse click on the Molokai Outfitters text object to bring up the Customizer menu. From the Customizer, select **Font Name**. In the list box, select **MS Sans Serif**. Set **Font Size** to **24** points. Set **Font Enhancement** to **Bold**. Set the **Text Color** to **Red**.



Tip: When you assign several attributes at once from the Customizer, you can keep the Customizer open until you have made your last choice, then click **Done**.

6. Click **Done** in the Customizer menu.

7. Resize the text object (click on it, then pull the small boxes, called **handles**) so the tool bar looks like this:

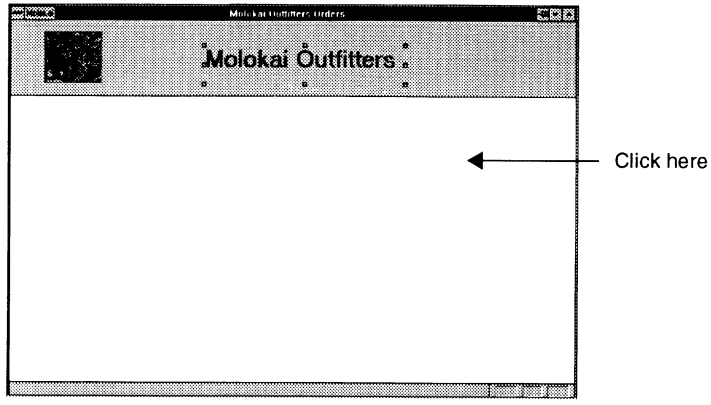


Summary

You just created your main application window, called Molokai Outfitters Orders, and added the Molokai Outfitters banner consisting of their company logo and name. Next, let's create the form window where you can display company details and invoice information.

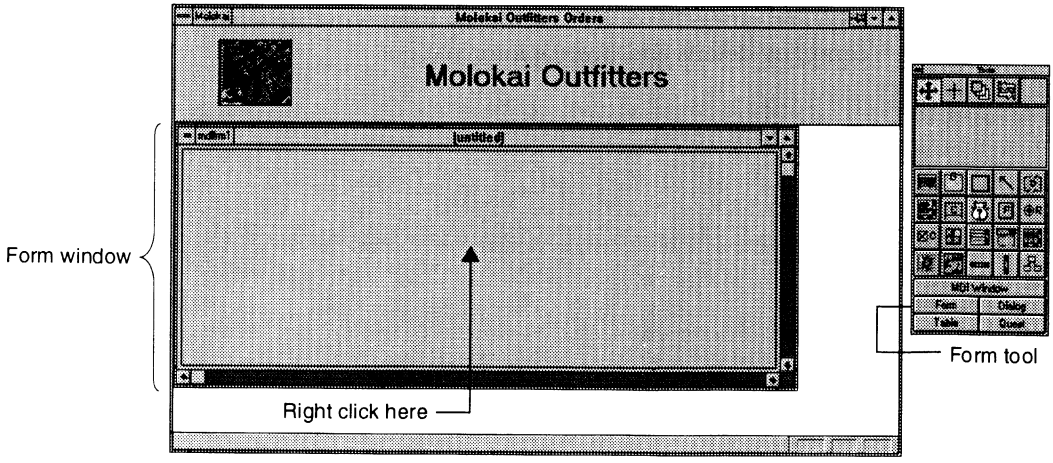
Create the Invoices form window

Before you create the Invoices form, make sure the Molokai Outfitters Orders MDI window has the focus: click on it.



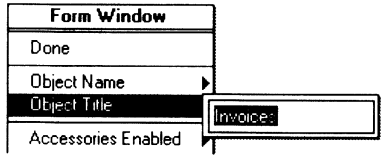
☞ 1. Click **Form** on the tool palette. This creates a child window in the MDI window.

2. Resize the form to make it larger. The size in this picture is approximately the size you want:



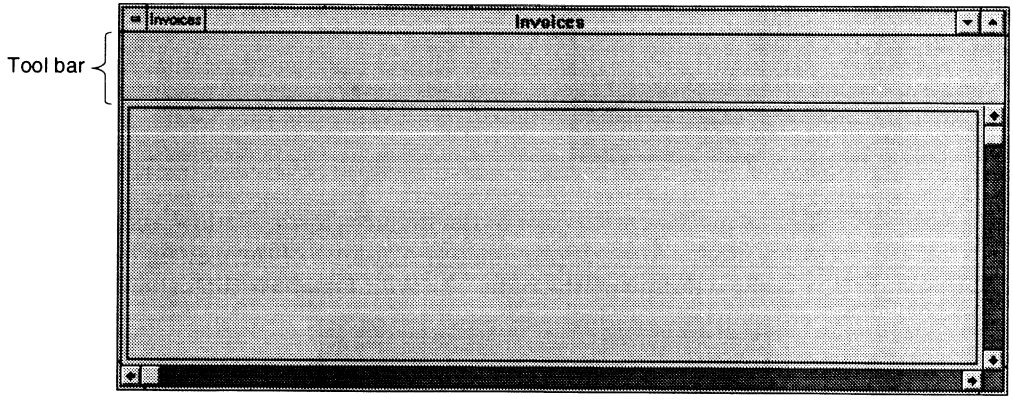
3. Right mouse click on the form window to bring up the Customizer menu.

4. From the Customizer menu, select **Object Title** and type **`Invoices`** and press **Enter**.



5. Select **Object Name** and type **`Invoices`** and press **Enter**.
6. Select **Accessories Enabled** and change to **Yes**. This action creates your tool bar.

7. Click **Done.** The form window looks like this:

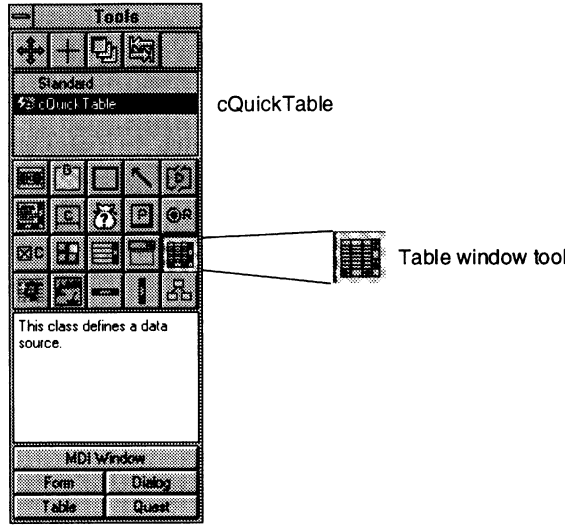


Now you have built the Invoices form. Next, add a data source object (a table) that links your form to the database so you can retrieve data.

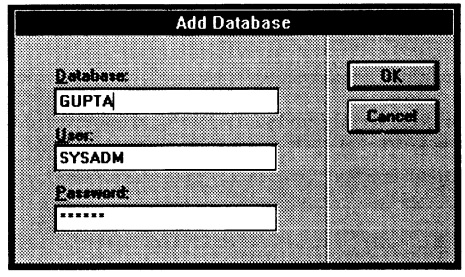
Build the data source for Company information

1. Click on the Table window tool in the tool palette.

   The tool palette expands to display information and the options for your choice. Select the **cQuickTable** data source class in the list box at the top of the tool palette.



2. Move the cursor over to the Invoices form window and click to drop it onto the form, approximately a quarter of the way up from the bottom of the window on the right. The **Add Database** dialog box appears:

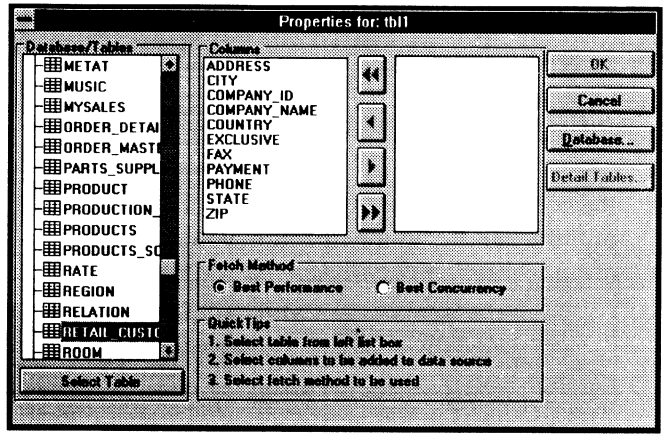


3. In the **Database** field, type **GUPTA**. In the **User** field, type **SYSADM**. In the **Password** field, type **SYSADM** (asterisks appear).

   This dialog box only appears the first time you try to access the database while in SQLWindows. From now on, it will only ask you for user name

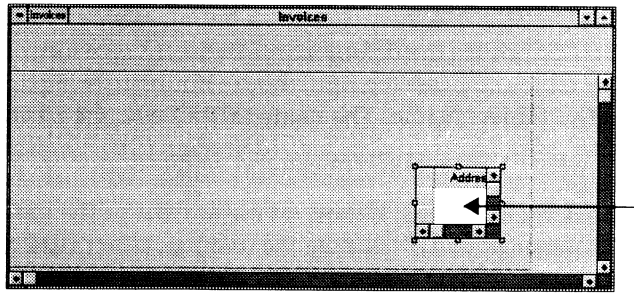
and password if you leave SQLWindows and return, or if you want to choose a different database.

4. Click **OK**. The **Properties** dialog box appears.



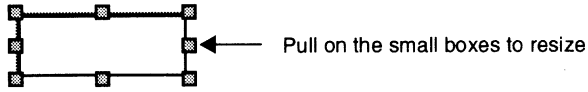
5. In the Database/Tables list box on the left, scroll down to the RETAIL_CUSTOMER table (just typing 'r' four times on your keyboard also takes you to this table) and double-click on it. All the column names appear in the Columns box in the middle, ready for you to select them for your table.

6. Click the double-right arrow. Click **OK**. The form window looks like this.



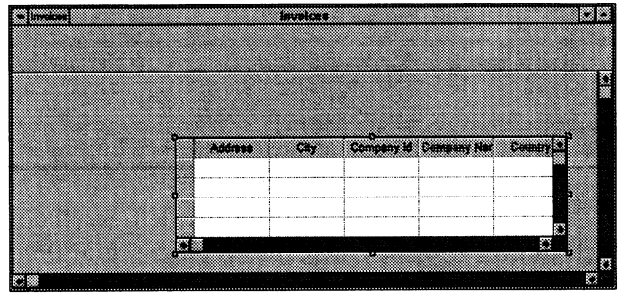
7. Click on the table to select it so you can resize it.

Tip: To resize an object in a window, select the object you want to resize. To select it, move the cursor over the window and click. A selected object looks like this:



Then, move the cursor to any one of the small boxes. The cursor becomes a small, two-way arrow. Hold down the mouse button and move the cursor to resize.

8. Resize the table to look like this (click on the table, then use the cursor to pull the handles in the direction you choose):



SQLWindows gives this object the default name **tblRETAIL_CUSTOMER1**. The 1 at the end of the name indicates this is the first instance of this table. This table connects the Invoices form to the GUPTA database.

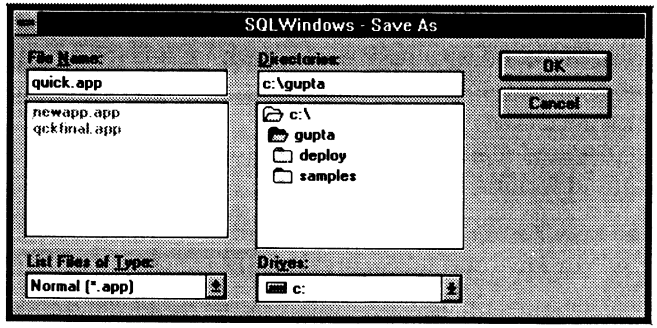
You do not see the default name in the Invoice form. Place the cursor over the table and press the right mouse button to bring up the Customizer and select **Object Name**. The name **tblRETAIL_CUSTOMER1** is displayed.

Guess what? You're ready to run the application.

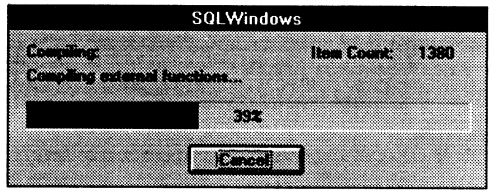
1. Click the **Run** push button on the tool bar at the top of the SQLWindows Designer. If you do not see the tool bar, select **User Mode** from the **Run** menu.

2. SQLWindows asks if you want to save your changes. Click **Yes**.

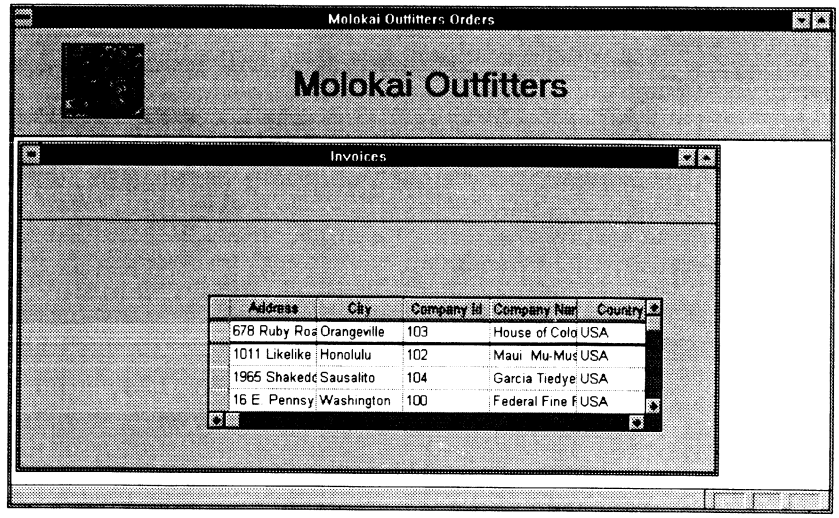
3. Save your application as QUICK.APP. Type **QUICK.APP** in the file name box and press **OK**.



The application starts compiling. When it is compiling, SQLWindows tracks it like this:



Your working application looks like this:

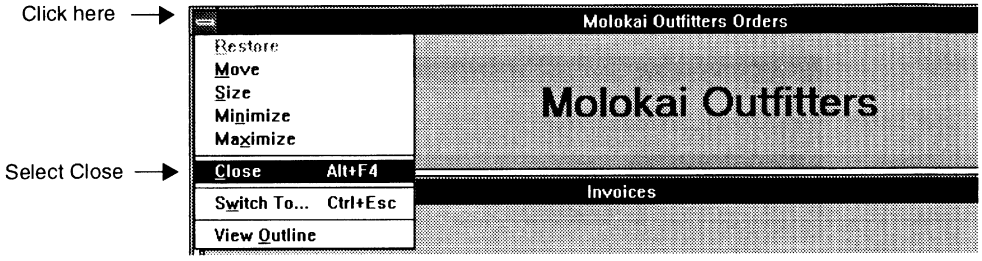


Summary

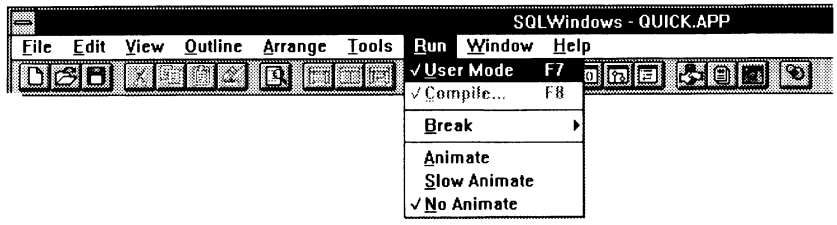
You have now created a data source on your form. All the code required was created behind-the-scenes for you. All you did was point, click, drag, and drop. The code is complete and works as it is to bring the Customer information into the form. The window is embedded exactly where you placed it in the form and works without any additional coding steps to display and manage your customer data. You accomplished all of this without writing any code!

You now have a choice. You can continue with the next topic, or you can quit SQLWindows and take a break.

To continue with the next topic, select **Close** from the System menu in the title bar of the Molokai Outfitters Orders window.



Tip: You can always toggle back and forth between Design Mode and Run Mode in SQLWindows by selecting **User Mode** from the **Run** menu.



To quit, select **Exit** from the SQLWindows **File** menu. To return to SQLWindows and continue this test drive:

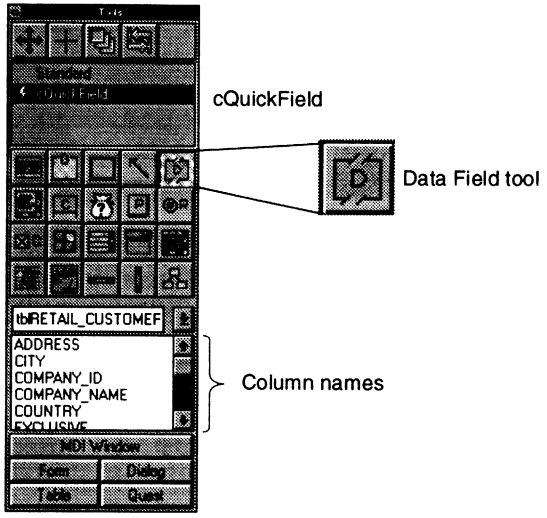
1. Double-click on the SQLWindows Designer icon in the Gupta program group.
2. Select **Open an Existing Application** from the window that appears.
3. Specify **`QUICK.APP`**.

SQLWindows returns you to in Design Mode.

Add customer data using QuickObjects

You just saved and ran your application. Now you add QuickObjects (data fields, push buttons, and a combo box) to the Invoices form. You use these objects to view information in the database.

☞ 1. From the tool palette, click on the Data Field tool.

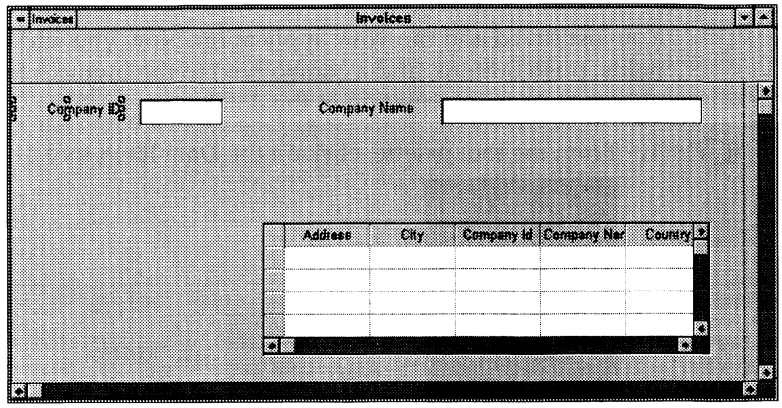


2. Select **cQuickField** from the list box in the tool palette by clicking on it.
3. Select, in this order, these column names from the list box on the lower part of the tool palette: COMPANY_ID, COMPANY_NAME.

Tip: To multi-select from a list box, click on the first selection, then hold the control key (**Ctrl** on your keyboard) while clicking additional items.

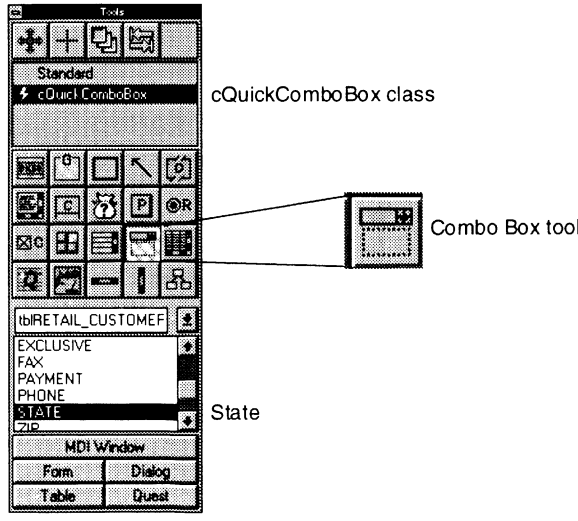
---

4. Drop the data fields onto your Invoices form by moving the cursor over the center of the form and clicking. Arrange them so that the fields are lined up like this:



Now you are experiencing the power of QuickObjects! You just created instant data-aware fields for your customer information with mouse clicks. Think back to the last programming environment you used and the amount of code you had to write. Recall the work involved to map the form data sources to the supporting database.

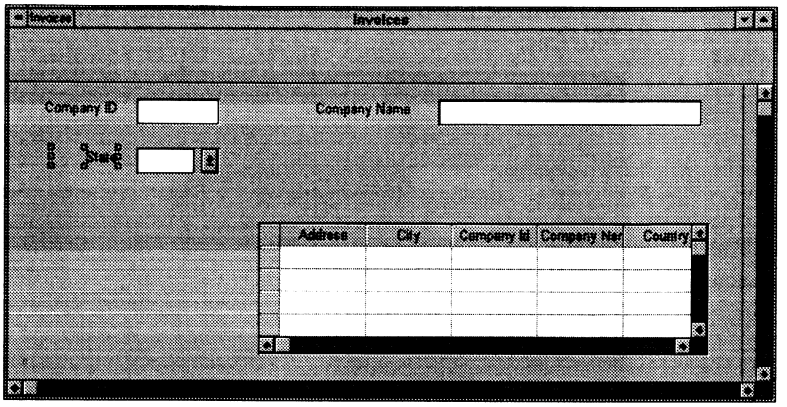
5. Now, click on the Combo Box tool on the tool palette.



6. Click the **cQuickComboBox** class that appears at the top of the tool palette.

7. Click on the **STATE** data item in the list box that appears at the bottom of the tool palette. Move the cursor over to the form and click to "drop" the combo box onto the form under **Company ID**.

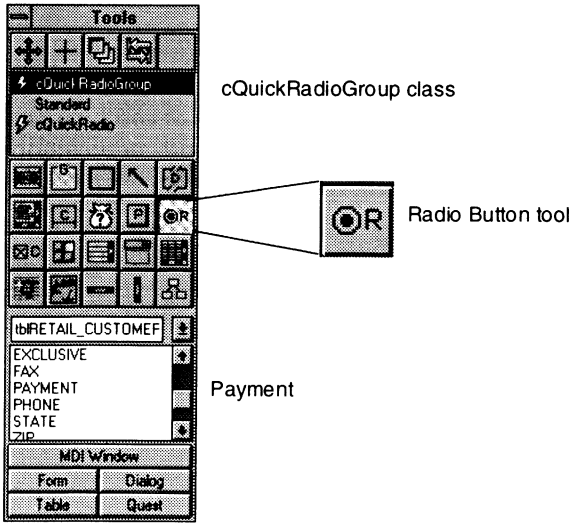
   Your form looks like this:



   Creating the STATE combo box with QuickObjects turns perhaps 30 minutes of experienced programming time into just three mouse clicks.

Now let's use another kind of tool, the Radio Button.

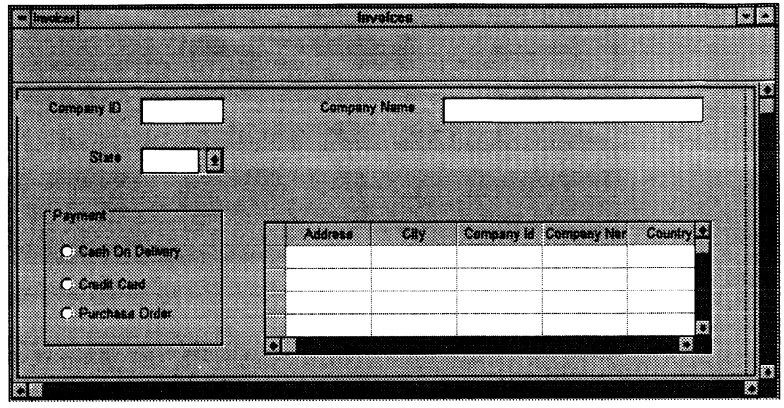
8. Click on the Radio Button tool.



   This tool has two cQuickObject classes available. One is for a single radio button, the other is for a group of radio buttons. The cQuickRadioGroup

class creates the names of the radio buttons automatically with information from the table.

9. Click on the **cQuickRadioGroup** class at the top of the palette. Click **PAYMENT** in the list box at the bottom of the palette. Move the cursor to the form below the State combo box. Click to drop. The form now looks like this:



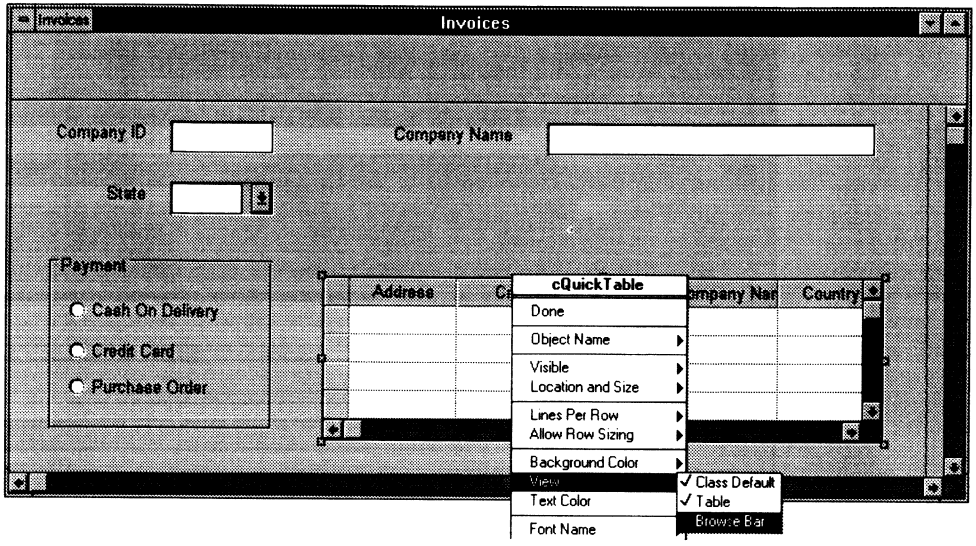
With QuickObjects it's easy to create a labeled group of radio buttons.

QuickObjects help you by reading the database and then using the correct names for you. You see how easy it is to be a quick and powerful programmer with SQLWindows.

Look where we are. The company information is on the form in just a few mouse clicks. You did not write one lick of code. You created complex visual objects that determined their elements and composition from the database. It was easy.

Now, we want to change the view of the table to a browse bar. We want to do this because we're ready to run the application again, and we'll want to scroll through the data using a browse bar connected to the database.

10. Click on the table window object to select it (click in the white area). Right mouse click to bring up the Customizer menu. From the Customizer, select **View** and set to **Browse Bar**.



11. Click **Done**. This table will now behave as a fully functional browse bar in Run Mode.

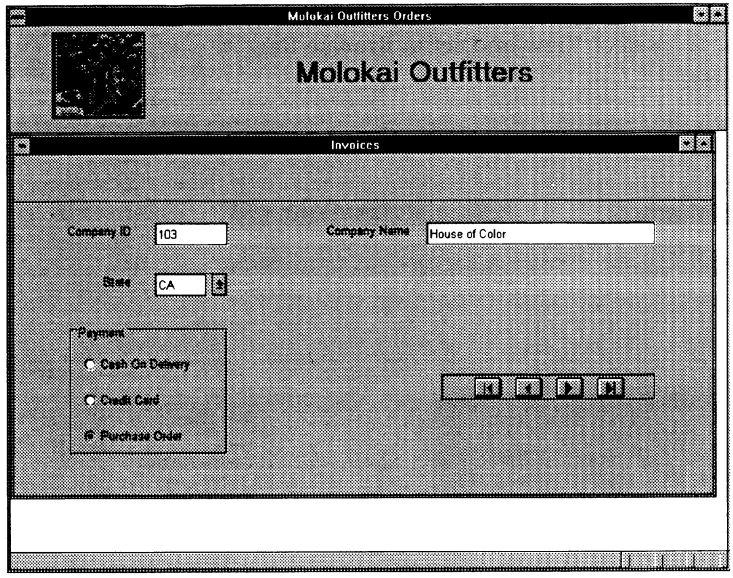
Tip: When a table is in **Table** view, click on various line items in the table to display corresponding data in data fields in Run Mode. When a table is in **Browse Bar** view, use the left and right arrows to display corresponding data in the data fields in Run Mode. You can toggle back and forth in Design Mode using the View option on the Customizer menu.

Let's run the application again.

12. Click the **Run** push button on the tool bar at the top of the SQLWindows Designer, or select **User Mode** from the **Run** menu.

13. SQLWindows asks if you want to save your changes. Click **Yes**.

Your working application looks like this:

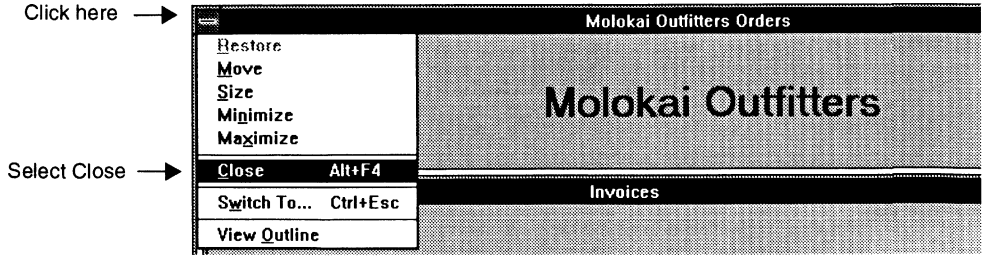


Notice how easy it is to test your prototypes, running your application on the fly.

Navigate through the data. Click the right arrow on the browse bar first. Then use the arrows to move through the information for various companies. Let the incredible power of the SQLWindows programming environment sink in.

Let's return to the SQLWindows Designer and keep building.

1. Click on the top left corner of the Molokai Outfitters Orders form to drop down the System menu for the form.

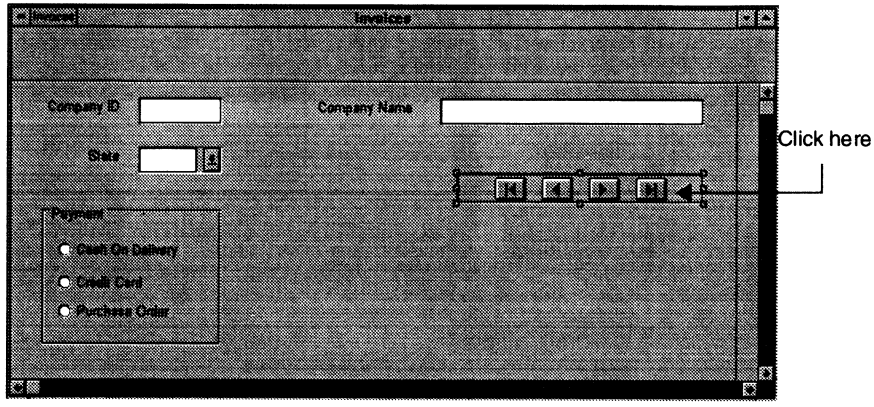


2. Click **Close**. SQLWindows returns you to the SQLWindows Designer.

Build a table to display invoice information

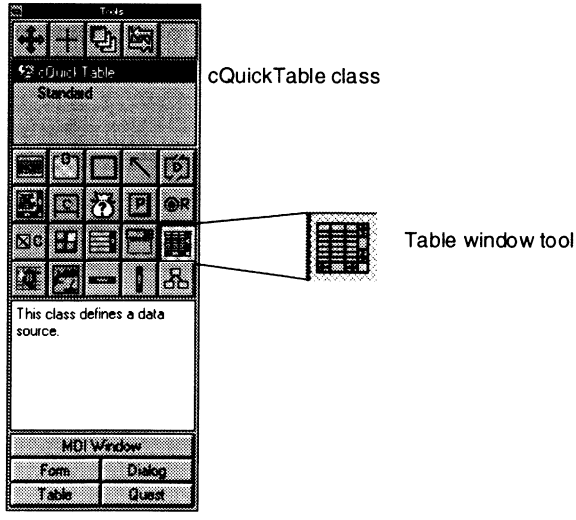
Before building the next table, let's move the browse bar up to make room.

1. Click on the browse bar to select it. Hold the mouse button down and "drag" it up to make room for the next table.



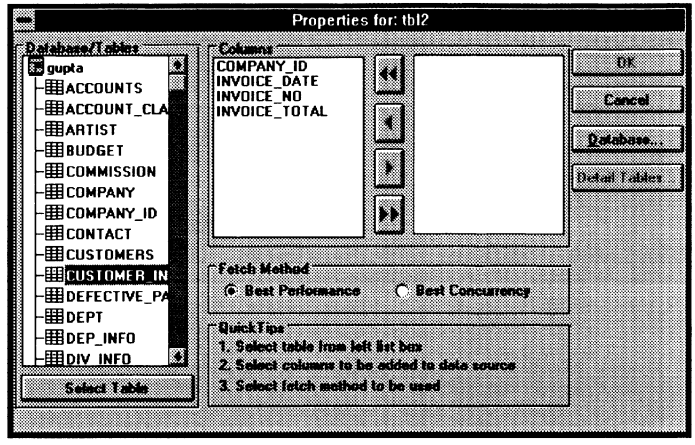
2. Click on the Table window tool in the tool palette.

   The tool palette expands to display information and the options for your choice. Select the **cQuickTable** class in the list box in the tool palette.



3. Move the cursor over to the Invoices form and click to drop this table in the lower right corner. When you click to drop it, hold down the mouse button and drag to the right a couple of inches. This resizes it.

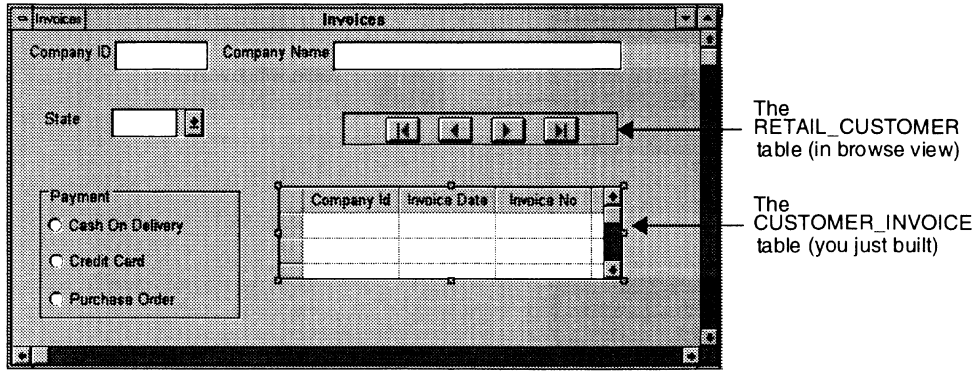
   The **Properties** dialog box appears.



4. In the Database/Tables list box on the left, double-click on the **CUSTOMER_INVOICE** table. All the column names appear in the Columns box in the middle.

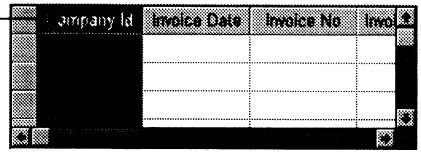
5. Click the double-right arrow to move all the column names to the right box. You are selecting these columns for the table on the Invoices form. Click **OK**.

   The form looks like this:



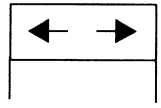
Now you want to "hide" the Company ID column in this table because that information displays already in the Company ID field above. To hide the column, place your cursor over the column heading and left click to select the column.

Click here



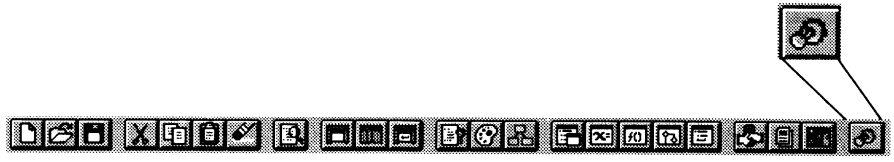
6. Right click on the column heading (not in the column itself) to bring up the Customizer for that column. Select **Visible**, then set to **No**. The column disappears. Click **Done**.

7. Make the Invoice No column the first visible column on the left. To do this, select the column heading. Move the cursor until it changes to the following shape in the column heading. Then, click on the column. Now, drag the entire column to the left.



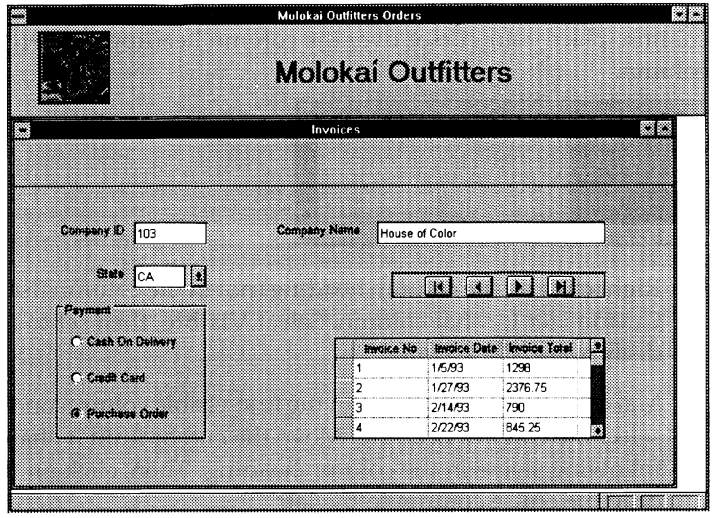
Let's run the application.

8. Click the **Run** push button on the tool bar at the top of the SQLWindows Designer, or select **User Mode** from the **Run** menu.



9. SQLWindows asks if you want to save your changes. Click **Yes**.

Your application looks like this:

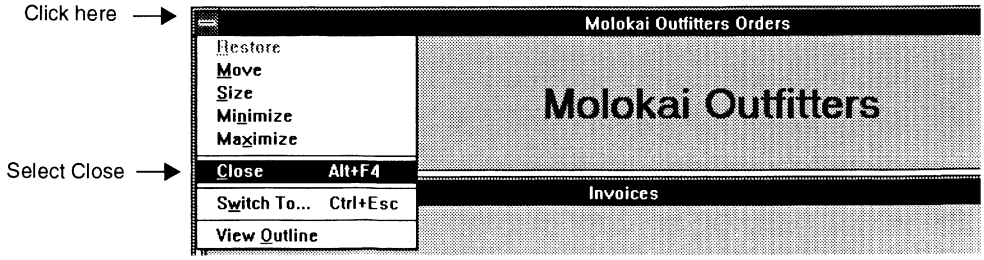


Note that the customer information appears for the first record that displays. However, as you scroll through the various records using the browse bar, the invoice information does not change. This is because your detail table, Customer Invoice, is not linked to the master table, Retail Customer. We'll do that in the next section.

Summary

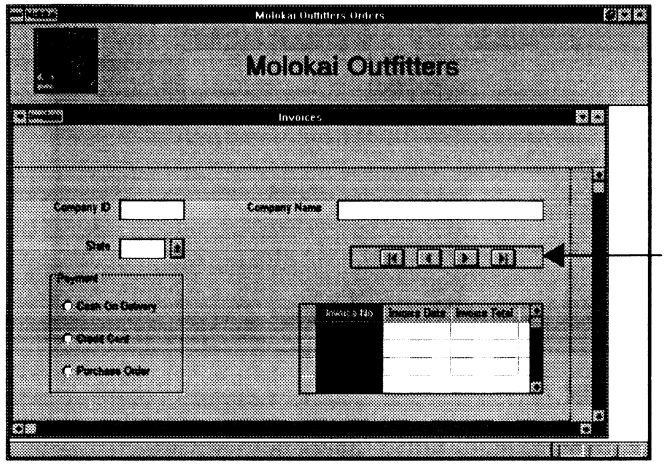
See how easy it is to create a window on your form to display a table? It's easy to customize the object to hide columns, change the order of columns, do whatever you need to finish your application. Just imagine how long it could take to "get it right" in other programming environments for Windows.

Let's close the application again and return to the SQLWindows Designer to keep building. Select **Close** from the System menu in the application's title bar.

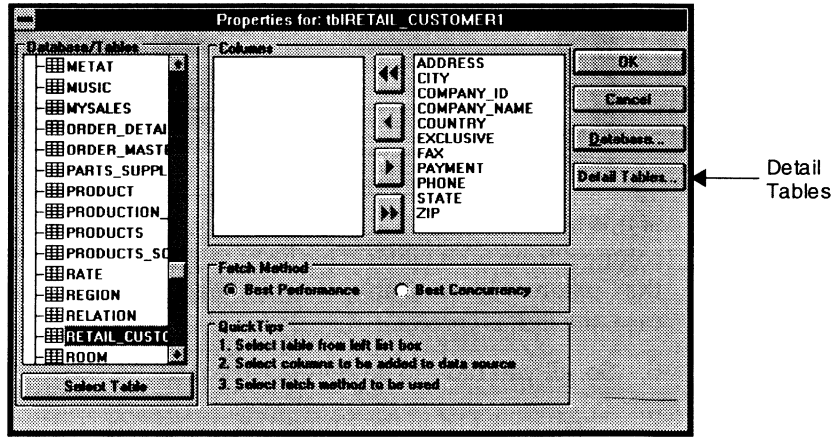


Linking the master and detail tables

Your detail table, Customer Invoice, is not linked to the master table, Retail Customer. Here is how the application looks:

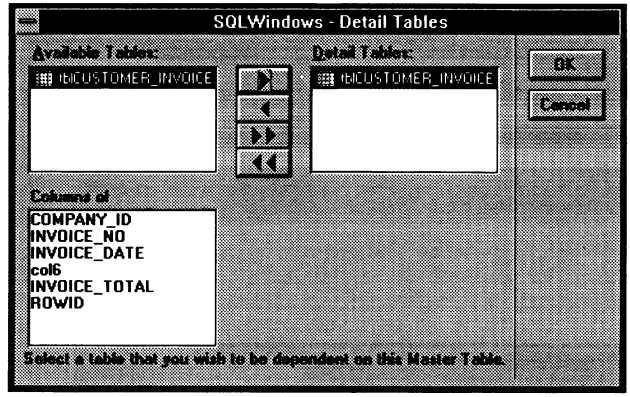


1. Right mouse click on the browse bar. This is the browse bar for the Retail Customer table, which is our master table. The cQuickTable Customizer menu appears.
2. Select **Quick Table**.
3. The **Properties** dialog box opens. Click the **Detail Tables** push button.



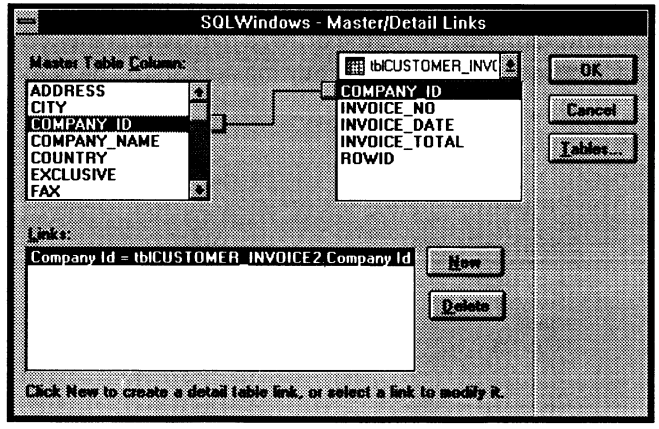
The **Detail Tables** dialog box appears.

4. tblCUSTOMER_INVOICE2 appears in the **Available Tables** box on the left. That's your detail table. Click the right arrow to select it. All column names appear in the **Columns of** box.



5. Click **OK**.

   The **Master/Detail Links** dialog box appears.



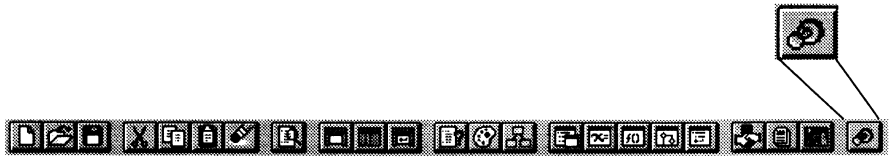
SQLWindows makes the most likely link for you, using the Company ID columns from both tables. This links the Retail Customer table information to the Customer Invoice table. You can change the link if you need to, but in this case, the link is correct.

6. So, all you have to do now is click **OK**!

7. Then, click **OK** in the **Properties** dialog box.

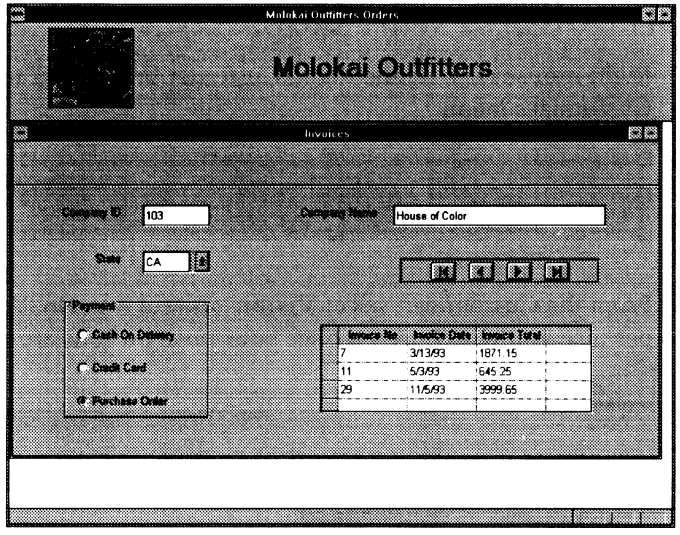
Now you can check on the progress you have made in your application.

8. Click the **Run** push button in the SQLWindows tool bar, or select **User Mode** from the **Run** menu.



9. When SQLWindows asks you if you want to save your changes, click **Yes**.

The application looks like this. Use the browse bar, and you can watch the information in all the other fields change.

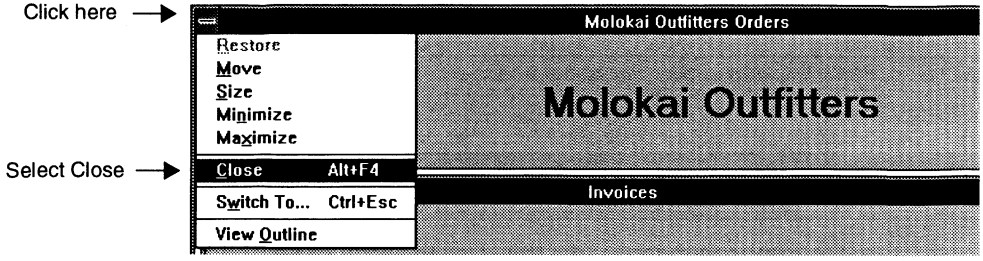


Summary

Now we're cruising! That's all it takes to link the two tables on your form. SQLWindows is designed to make power programming easy. You linked the two tables without writing any code. Reflect on how easy this is. By now you are already getting used to this "no code" method of programming Windows applications. Whether you are an experienced Windows and client/server programmer, or you are just starting out, you can appreciate how fast your applications can be up and running with SQLWindows.

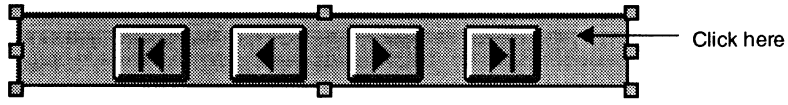
Now you will hide the standard browse bar for the Retail Customer table and build push buttons for browsing through the Retail Customer table, retrieving a customer's invoices, and applying edits made to the information on the form.

1. Select **Close** from the System menu in the applications's title bar to return to the SQLWindows Designer.

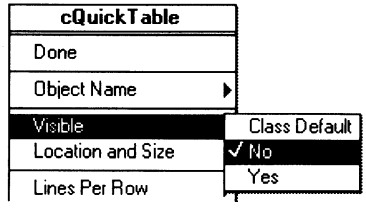


Create push buttons with QuickObjects

1. Right mouse click on the Retail Customer browse bar to bring up the Customizer menu.



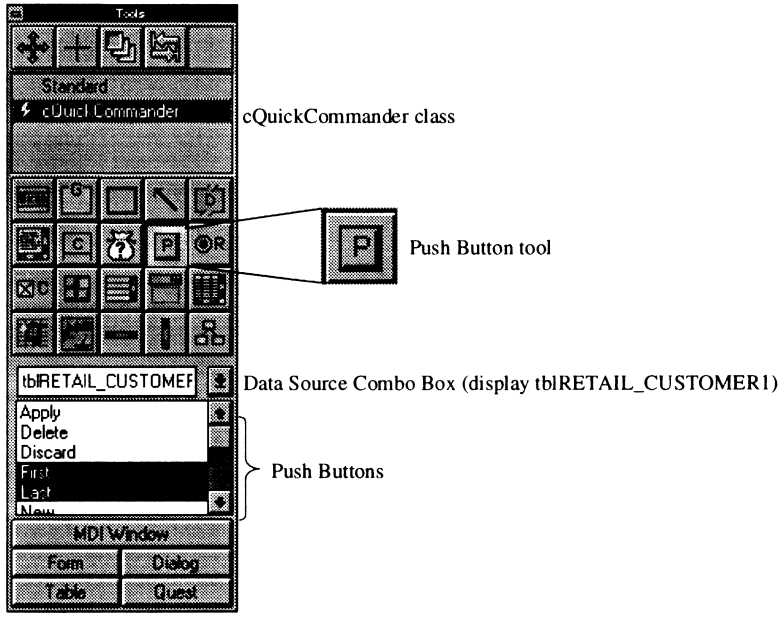
2. From the Customizer, select **Visible** and set it to **No**. You set this to **No** because you won't need it once your push buttons are in place.



3. Click **Done**. Now let's create some push buttons.

1. Select the Push Button tool on the tool palette. Select the **cQuickCommander** class.

Note: Make sure **tblRetail_Customer1** is highlighted in the data source combo box on the tool palette. Click the **down** arrow next to the **Data Source** combo box to display available tables.



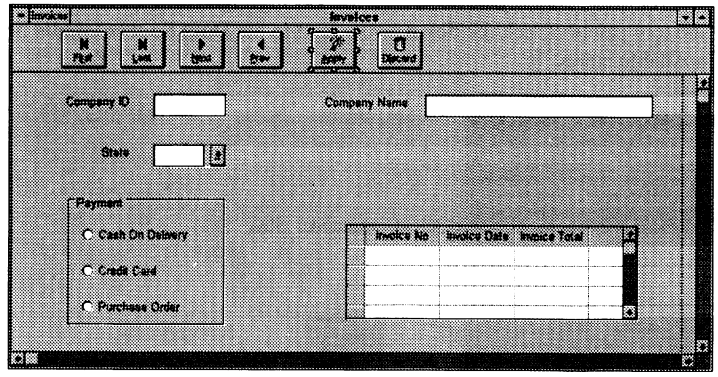
2. Click on the **First** list item (push button) in the list box at the bottom of the tool palette. Hold down the **Ctrl** key and click the **Last**, **Next**, and **Prev** push buttons on the list.
3. Move the mouse arrow to the tool bar at the top of the form and click the mouse to drop the push buttons onto the form.

Now create the **Apply** and **Discard** push buttons in the same way.

Note: Make sure **tblRetail_Customer1** is selected in the data source combo box on the tool palette.

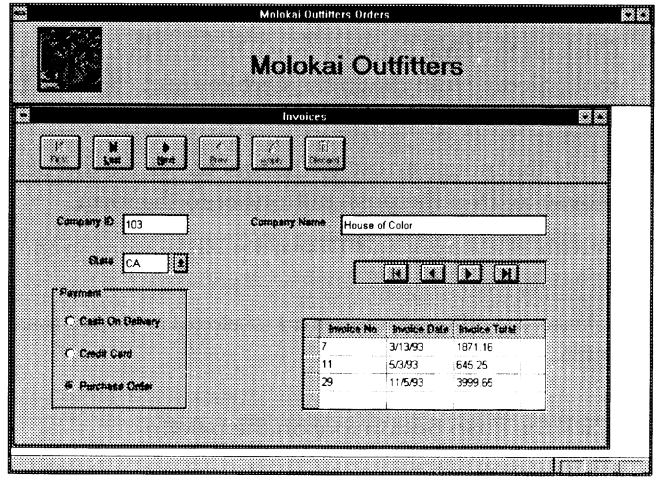
4. Click the Push Button tool on the palette.
5. Click the **cQuickCommander** class at the top of the palette.
6. Hold down the **Ctrl** key and click the **Apply** and **Discard** push buttons. Place the mouse cursor next to the **Prev** push button on the form and click to drop.

Your form now looks like this:



☞ 1. Let's run the application again. Click the **Run** push button or select **User Mode** from the **Run** menu.

2. Save the changes you have made to the application. It looks like this.



3. Try out the **Next**, **Prev**, **First**, and **Last** push buttons.

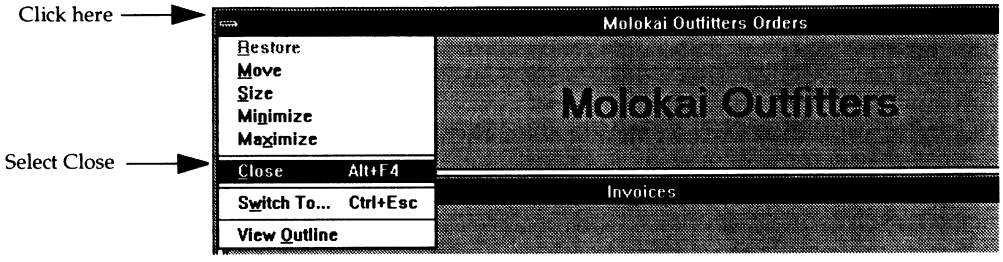
   The **Apply** push button applies changes to the database. The **Discard** push button discards changes you make before applying them to the

database. When you change information in a table, SQLWindows puts a check mark next to the line item.

	Invoice No	Invoice Date	Invoice Total	
✓	7	3/13/93	1871.16	
	11	5/3/93	645.25	
	29	11/5/93	3999.65	

Let's return to the SQLWindows Designer and keep building.

4. Select **Close** from the System menu in the application's title bar to return to the SQLWindows Designer.

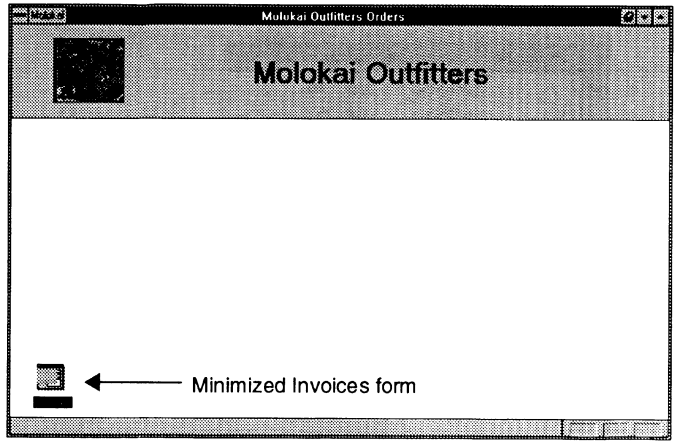


Create the Items form

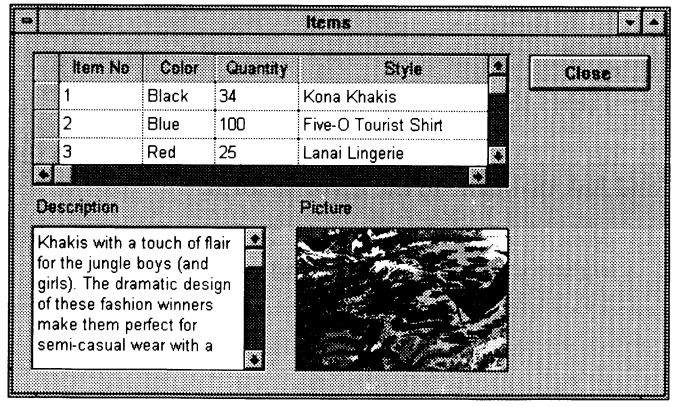
In this section, you create the new form and give it a title, name, add a table, and link that table to the Invoices form.

1. Minimize the Invoices form. To do this, click the down arrow in the upper right corner of the form window:

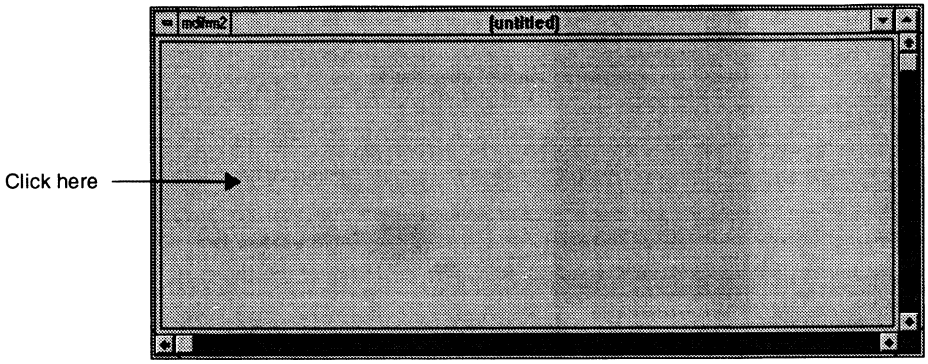
Your workspace looks like this:



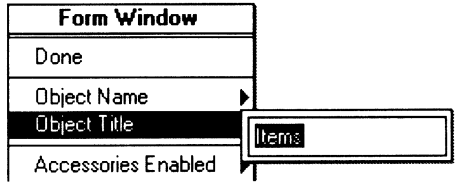
Now we are going to add a detail form for viewing invoice items. The form will display order line items, a description of the shirt being ordered, and a picture of the fabric. Here is how the finished form will look:



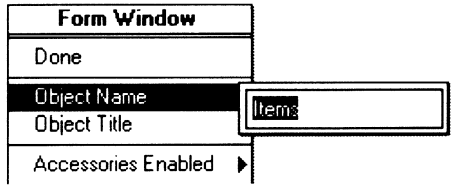
2. Click on the **Form** tool in the tool palette to create your new form. The new, untitled form window opens and looks like this:



3. Right click on the form window to bring up the Customizer menu. Select **Object Title** and type **`Items`** and press **Enter**.



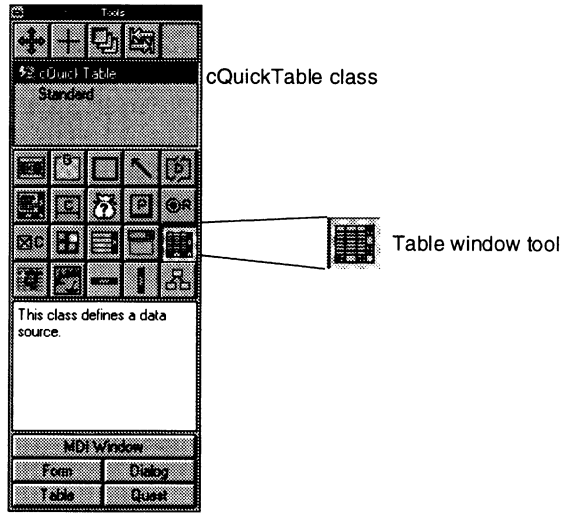
4. On the Customizer menu, select **Object Name** and type **`Items`** and press **Enter**.



5. Click **Done**.

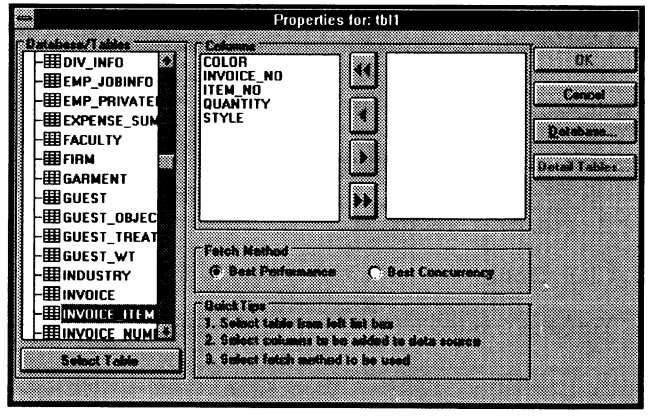
You just created a new form and named it. Now, let's add a table.

1. Click on the Table window tool in the tool palette. Select the **cQuickTable** class.



2. Drop the table object onto the **Items** form at the top left. The **Properties** dialog box appears.

3. Double-click on the **INVOICE_ITEM** table (type "i" until you see the table highlighted).



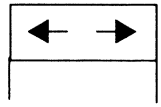
4. The column names appear in the Columns box. Click the double-right arrow to move all the column names to the right box. Click **OK**.

   Now, this table is connected to the database, but not linked to the Invoices form. First, let's arrange the columns in the order we prefer. Do not skip

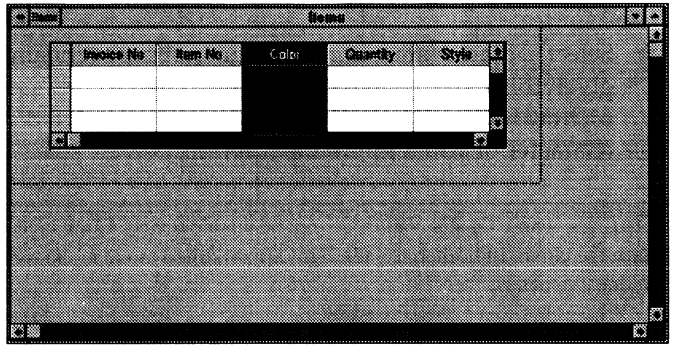
this step, as it determines the location of the columns we want to link later on.

5. Resize the table to show all columns. Click in the white area, then drag the right edge to the right.

6. Move the Color column to the right of the Item No column. Click on the column heading until you see the double arrow cursor.



7. Drag to the right over the Invoice No and Item No columns. Your form now looks like this:

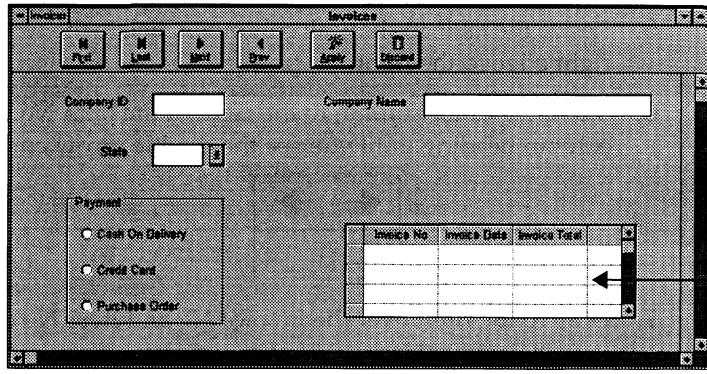


To ensure that this table displays information about the invoice you selected on the Invoices form, you need to link the Invoice Items table on this form to the Customer Invoices table on the Invoices form.

To do this link, we have to give SQLWindows very specific information about which columns, in which tables, on which forms, we want to link. So that you have the information you need, let's go check the Object Name of the Customer Invoices table.

☞ 1. Double-click on the **Invoices** icon to open the Invoices form. If you cannot see the icon, minimize the Items form first.

2. Click on the **Customer Invoices** table to select it, then right click to bring up the Customizer.



3. Select **Object Name**, and write down the exact name SQLWindows has given this table. Mine says **tblCUSTOMER_INVOICE2**. But, depending on how many tables you have created and deleted up to now, the last character of yours might be different.

Tip: The whole Object Name may not be visible at first. Click in the Object Name box to place the cursor in it, then use the right arrow key to move to the right and display any hidden characters.

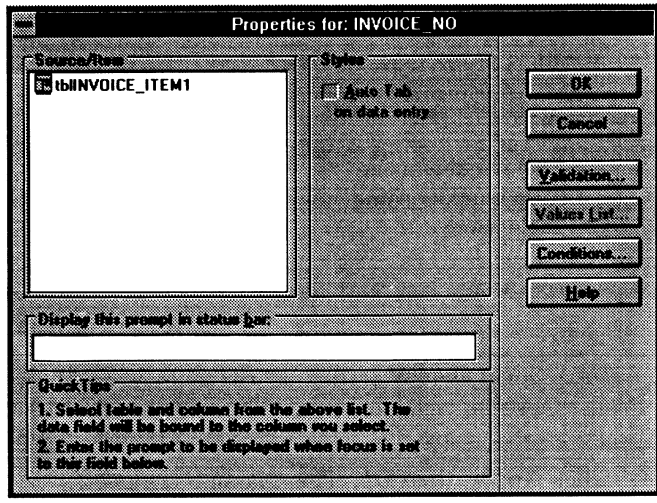
---

4. Click **Done**, then minimize the Invoices form again. (If you minimized the Items form, double-click on its icon to open it again.)

Now, let's create that link between the two forms. We are going to link the Invoice No column in the Customer Invoice table (on the Invoices form) with the Invoice No column in the Invoice Items table (on the Items form).

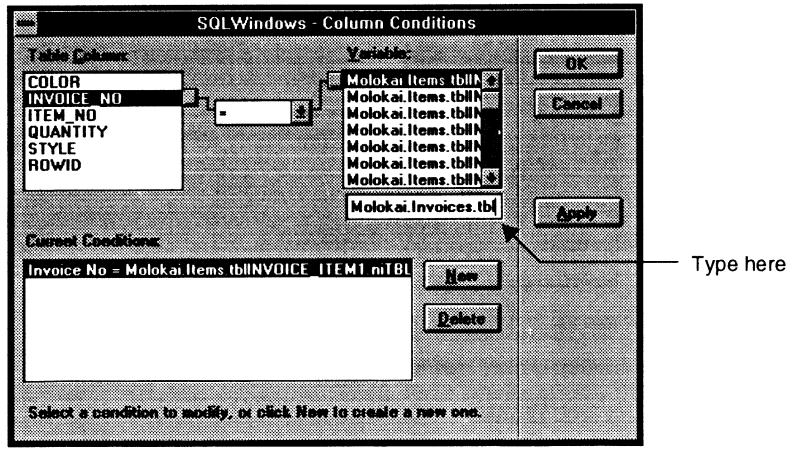
☞ 1. On the Items form, click on the heading of the **Invoice No** column in the Invoice Items table to select it. Then, right click to bring up the **cQuickColumn** Customizer.

2. Select **Quick Column**. The **Properties** dialog box appears.



3. Click **Conditions**.

   The **Column Conditions** dialog box appears.



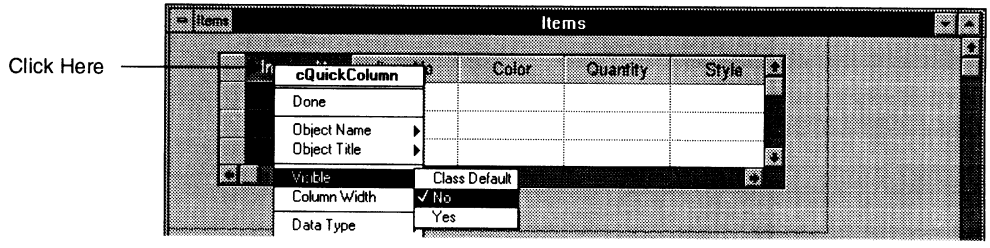
4. Click **New**. Select **INVOICE_NO** in the Table Column box on the left. Remember, SQLWindows made what it thought was the most likely link, but you want to specify a different one. In the **Variable** data field (under the list on the right), double-click to select the existing text, then press the

Delete *key* to clear the box. (Do *not* click the Delete push button under Current Conditions.) Then, type:

```
Molokai.Invoices.tblCUSTOMER_INVOICE2#2
   A       B              C            D
```

A = MDI window name	= Molokai
B = Form window name	= Invoices
C = Table name	= tblCUSTOMER_INVOICE2 (the number 2 may vary depending on the number of tables you've created - enter this name as you wrote it down in step 3 of the previous procedure)
D = Column number	= #2 (the number 2 may vary depending on how you arranged your columns)

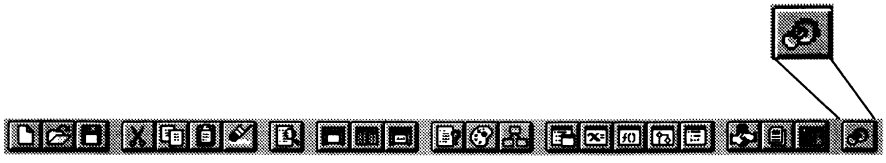
5. Click **Apply**, then **OK**. This dialog box closes.
6. Click **OK** in the **Properties** dialog box. The two tables are linked!
7. Now, hide the Invoice No column. You don't need to show it because that information is already displayed on the Invoice form. Move the cursor over the Invoice No column title and click to select it.



8. Right mouse click to bring up the Customizer for this column. From the Customizer, select **Visible** and set it to **No**.
9. Click **Done**.

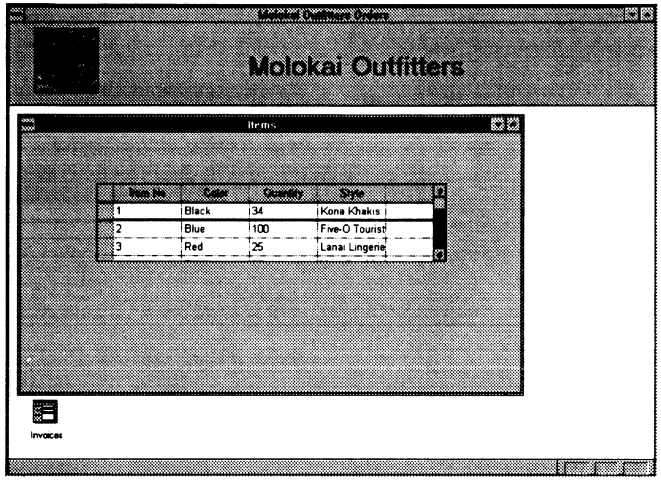
Let's run the application.

1. Click the **Run** icon on the tool bar or select **User Mode** from the **Run** menu.



2. SQLWindows asks to save your changes. Click **Yes**.

   Your application looks like this:



3. Slect **Close** from the System menu in the Molokai Outfitters Orders window to return to the SQLWindow Designer.

Summary

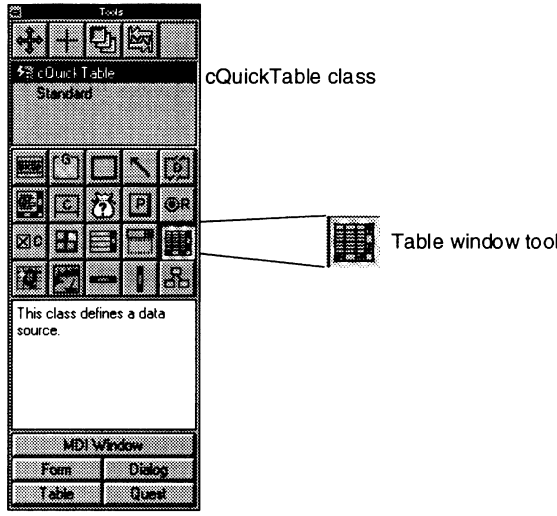
SQLWindows is easy to use, and powerful. You just added a detail form, using the power of the MDI form and the easy power of QuickObjects. You linked the Items form to the Invoices form without writing any code. I know you're excited now about using SQLWindows as your application development system.

Now you're going to finish the Items form by adding windows that display information about each shirt style and a picture of the fabric.

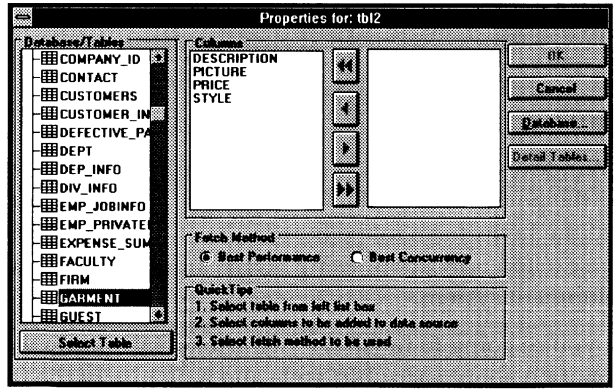
Create the Garment table

Now let's add Garment table information.

1. Click on the Table window tool in the tool palette. Select the **cQuickTable** class.



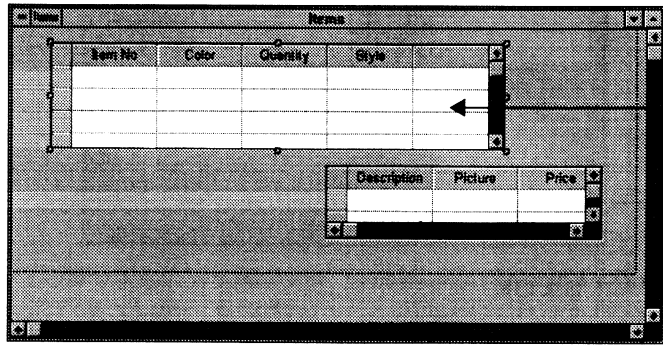
2. Drop the table object onto the Items form below the other table. In the **Properties** dialog box, double-click on the GARMENT table (type "g" until you see the table).



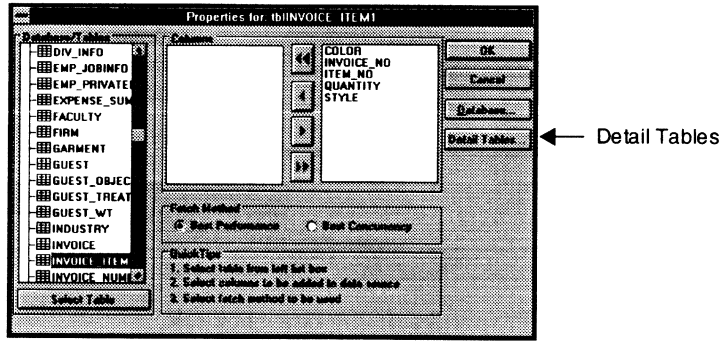
3. Click the double-right arrow to move all the column names to the right box. Click **OK**.

Now you have another data source (**Garment**). You need to link the **Garment** data source to its master, the **tblINVOICE_ITEM1** table.

4. Click on the data source that shows the INVOICE_ITEM table information.

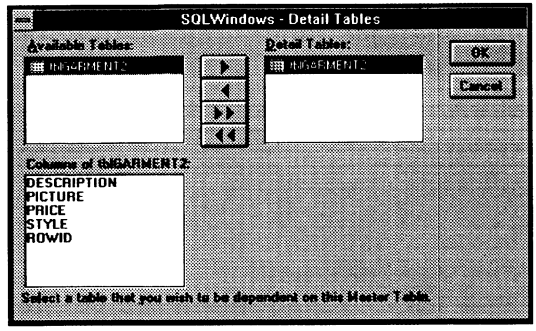


5. Right mouse click to bring up the Customizer. From the Customizer, select **Quick Table**. In the dialog box that appears, click **Detail Tables**.



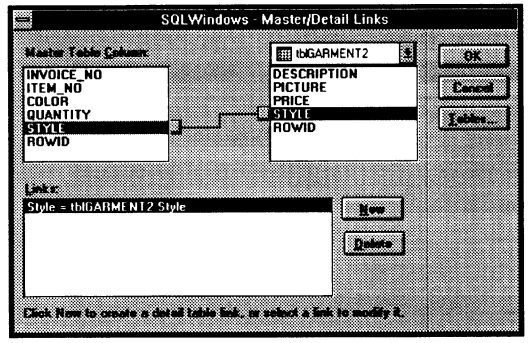
The **Detail Tables** dialog box appears.

6. tblGARMENT2 is already selected for you in the left box. Double-click on it to move it to the right box. The column names appear in the lower box.



7. Click **OK**.

8. The **Master/Detail Links** dialog box opens, showing the link between **Item** and **Garment**. Click **OK**.



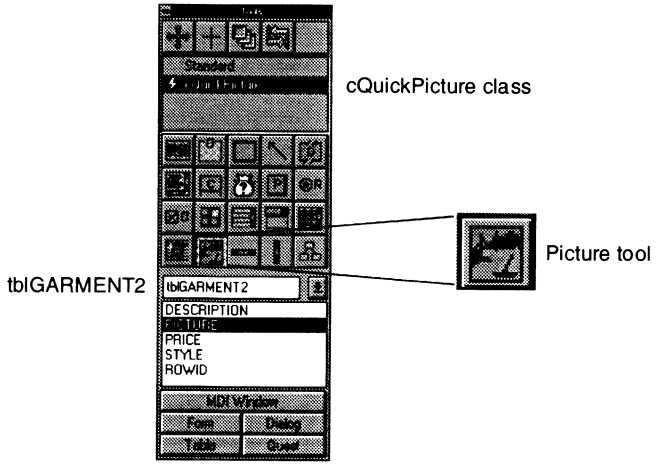
9. In the **Properties** dialog box, click **OK**.

Create the picture object for the shirt fabric

Now that you've linked the tables, you don't need the Garment table to be visible.

☞ 1. Right mouse click on the **Garment** table to bring up the Customizer menu. From the Customizer, select the **Visible** property and select **No**. Click **Done**.

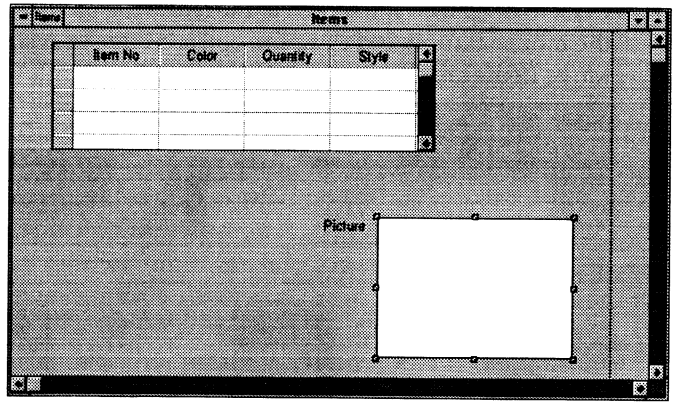
2. Click on the **Picture** tool in the tool palette. Choose the **cQuickPicture** class from the list box in the tool palette.



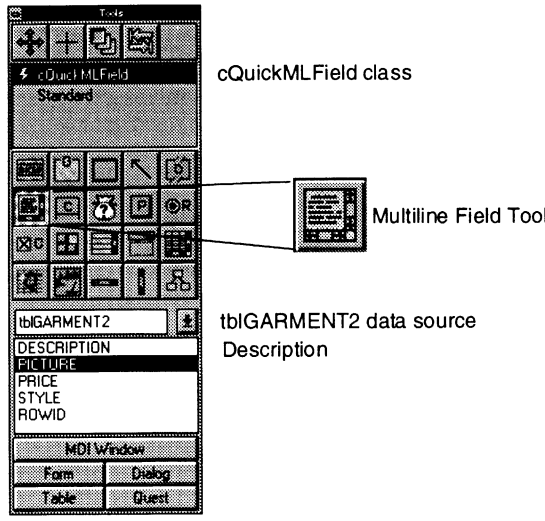
   The **Garment** data source (tblGARMENT2) is already highlighted in the list box in the tool palette.

3. Select the data item in the **Garment** data source named **PICTURE**.

4. Drop it on the screen to the right and below the **Items** Table object.



5. Click on the Multiline Field tool in the tool palette.

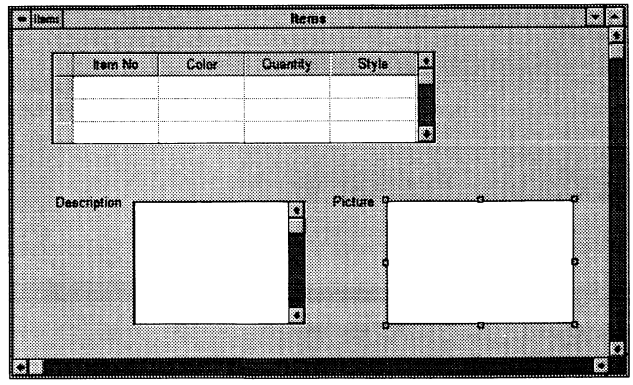


6. Choose the **cQuickMLField** class from the list box in the tool palette.

   The **tblGARMENT2** data source is already displayed on the tool palette.

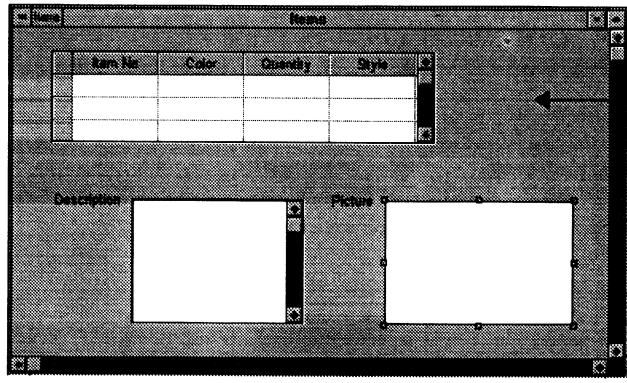
7. Select the data item in **tblGARMENT2** named **DESCRIPTION**.

8. Click to drop it on the screen to the left of the picture object, but hold down the mouse button and drag the object's edge to resize it before letting go. Move items around as necessary until your window looks like this:



   The Items window should not be automatically created when you start your application. You only want to see this window when you want to view detail invoice information.

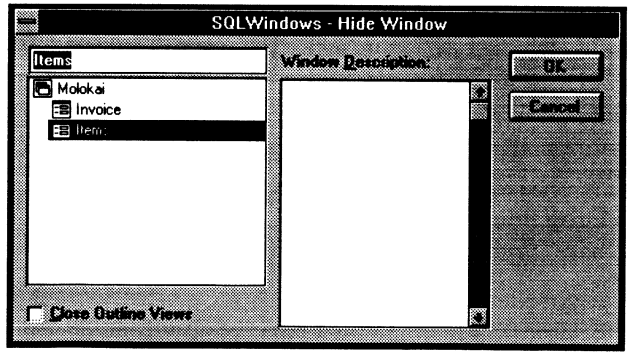
9. Right click on the Items form window to bring up the Customizer.



10. From the Customizer, change **Automatically Create** to **No**.

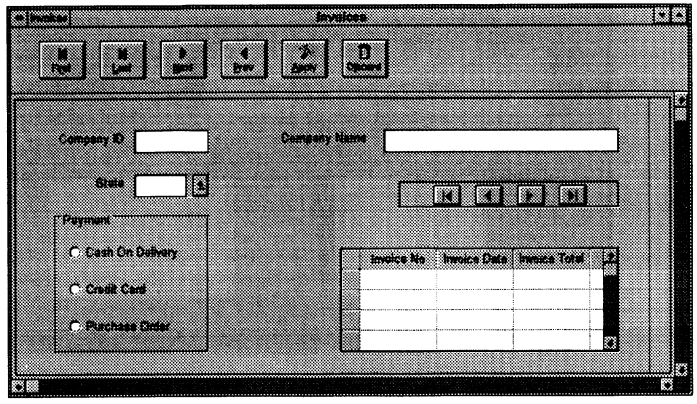
11. Click **Done**.

Hide the second form you created, named **Items**. From the **View** menu, select **Hide Window**. Make sure **Items** is highlighted. Click **OK**.

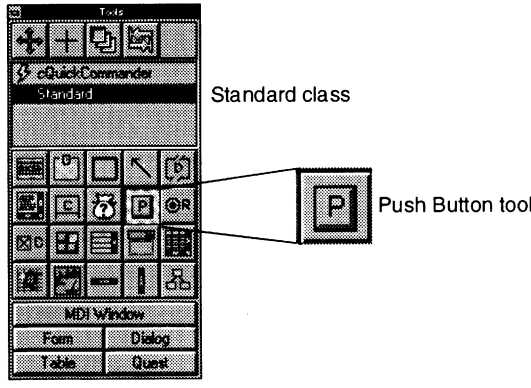


From now on, the Items form window only appears when you ask for it to be displayed. So, we need to add a push button to display the Items form.

1. Double-click on the icon of the Invoices form. We are back at the first form you created, named Invoices.

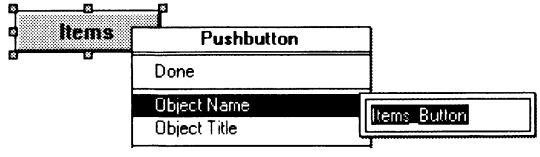


2. Select the Push Button tool and the **Standard** class from the tool palette.



3. Drop it to the right of the **Discard** push button on the Invoices form tool bar. Type `Items`. The title of the push button changes. Press **Enter**.

4. Right click on the **Items** push button to bring up the Customizer. From the Customizer, change the Object Name to `Items_Button`. Click **Done**.

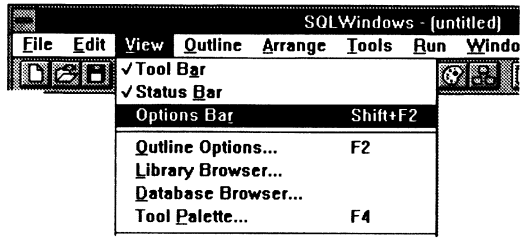


I know I've been a noisy champion of the no code, easy-to-use QuickObjects that make SQLWindows so powerful when building applications. Of course, any application project presents some obstacles. SQLWindows has the power to hurdle them. You'll see how effortless it is to code with SQLWindows.

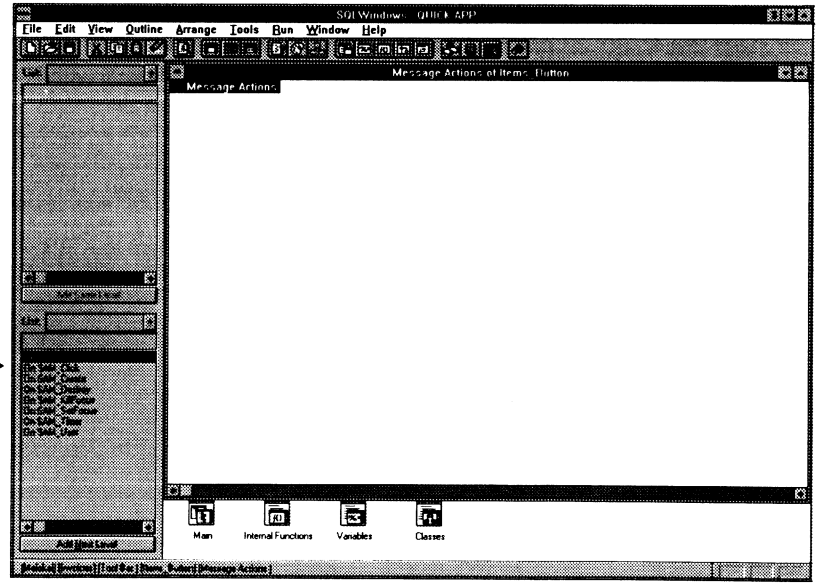
Code the Items push button

When you click on the **Items** push button, you want the application to bring up the Items form.

1. Right click on the **Items** push button to bring up the Customizer. From the Customizer, select **Edit Actions**.
2. Select **Options Bar** from the **View** menu in SQLWindows. You can also press **F2** to bring up the Outline Options bar.



The Outline Options bar appears on the left side of the workspace.



On SAM_Click →

The Message Actions section lets you specify actions that you want executed when the window receives a message. The message serves as the flag that sets off the action. Each message is associated with an event. For example, clicking a push button is an event.

In this case, SAM_Click (the SAM prefix stands for *SQLWindows Application Message*) is sent to the Message Actions section of the **Items** push button.

3. Find **On SAM_Click** and double-click on it.

   ✦ On SAM_Click

   appears in your outline. The Outline Options bar changes to show you the commands you can use with **On SAM_Click**.

4. Double-click on **Call** in the Outline Options bar.

   ✦ Call

   appears in your outline. The Outline Options bar displays the available functions you can use with the Call.

5. Scroll through the upper list box of the Outline Options bar to find SalCreateWindow. (You can also type part of SalCreateWindow to find it.)

6. Double-click on **SalCreateWindow**.

   ✦ Call SalCreateWindow (Template, Window_Handle)

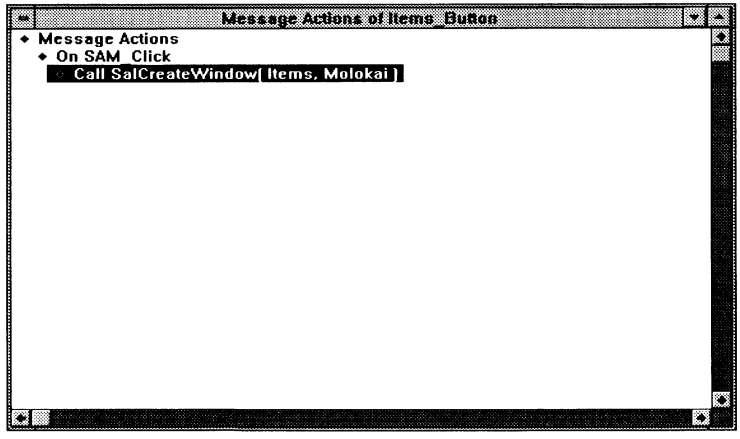
   appears in your outline.

7. **Template** and **Window_Handle** are highlighted. Type `Items, Molokai` over them. It should look like this:

   ✦ Call SalCreateWindow (Items, Molokai)

8. Press **Enter**.

You're finished! Your outline window now looks like this:



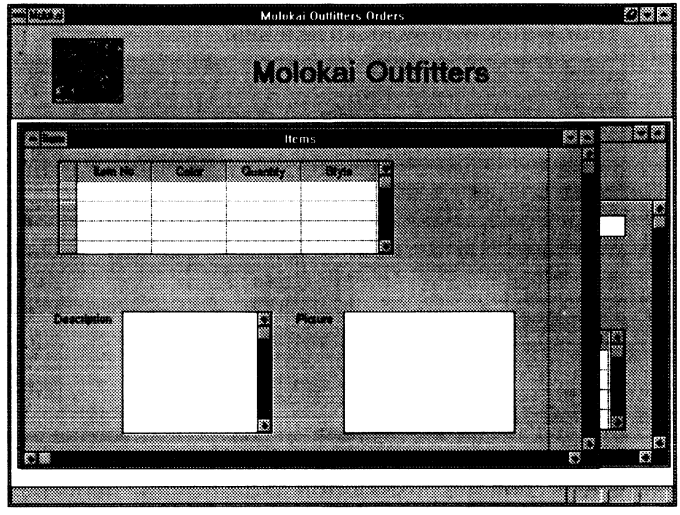
From now on, the **Items** push button brings up the Items form whenever you click it in the application.

Add a Close push button

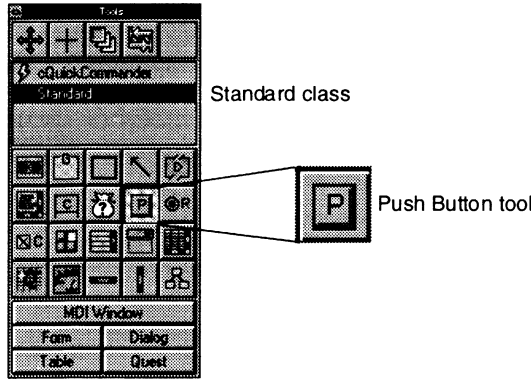
Next, you need a push button to close the window when you're finished browsing the detail information it displays. So, let's create a **Close** push button.

First, bring up the Items form.

☞ 1. Select **Show Window** from the **View** menu and highlight **Items**. Click **OK**.



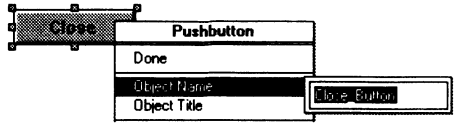
2. Select the Push Button tool and the **Standard** class from the tool palette.



3. Drop the push button in the upper right corner of the Items form.

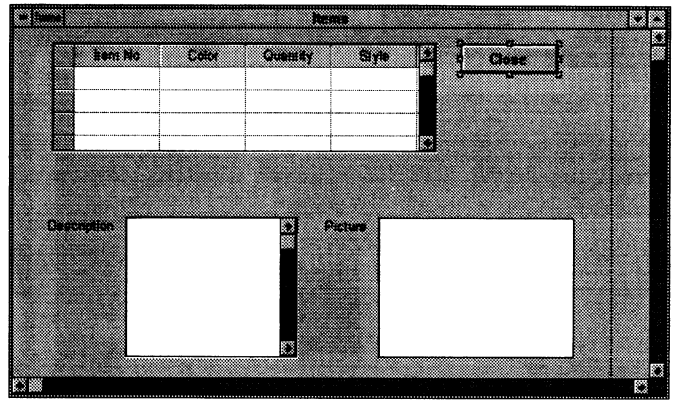
4. Type **`Close`** to give the push button a title. This text replaces the word **`(untitled)`** on the push button as you type. Press **Enter**.

5. Right click on the push button to bring up the Customizer menu. Select **Object Name** and type `Close_Button` and press **Enter**.



6. Click **Done**.

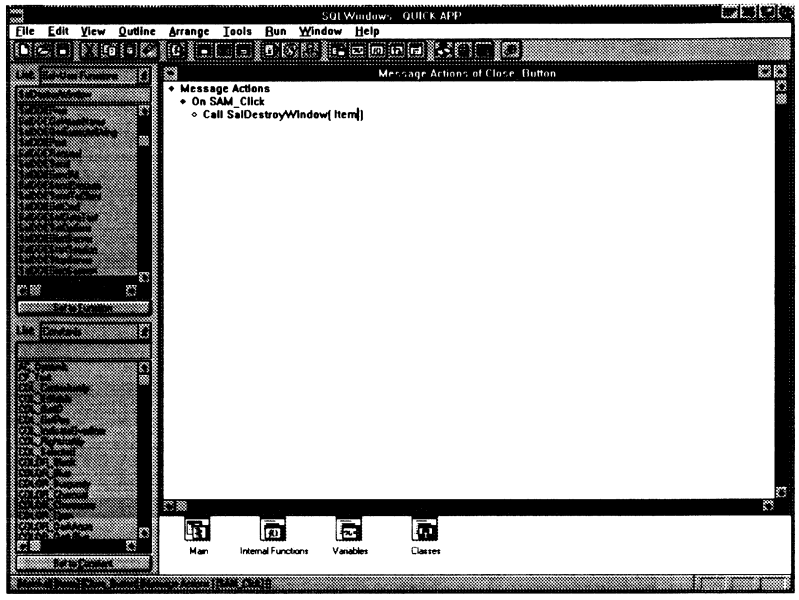
   Your window now looks like this:



Now, let's code the push button to destroy the Items form when clicked.

☞ 1. Click on the **Close** push button. Then, right click on the Close push button to bring up the Customizer. Select **Edit Actions**.

2. Press **F2** to bring up the Outline Options bar, if it is not already up.



3. Find **On SAM_Click** and double-click on it.

   ✧ On SAM_Click

   appears in your outline. The Outline Options bar changes to show you the available commands you can use with On SAM_Click.

4. Double-click on **Call** in the Outline Options bar.

   ✧ Call

   appears in your outline. The Outline Options bar displays the available functions you can use with the Call.

5. Scroll down in the upper list box of the Outliner until you see **SalDestroyWindow**. Double-click on **SalDestroyWindow**.

   ✦ Call SalDestroyWindow (Window_Handle)

   appears in your outline.

6. Window_Handle is already highlighted. Type `Items` over it.

   It should look like this:

   ✦ Call SalDestroyWindow (Items)

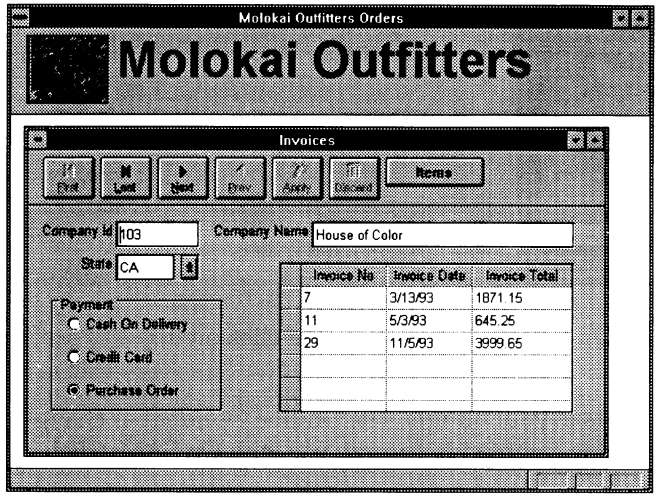
7. Press **Enter**.

You're finished! At this point, save and run your application.

1. Click on the **Run** icon, or select **User Mode** from the **Run** menu.

2. SQLWindows asks to save your changes. Click **Yes**.

Your window looks like this. Click on the **Items** push button to open the Items form. Click on the **Close** push button to return to this form.



Summary

I've been telling you how quick and powerful SQLWindows with QuickObjects is. Now you see for yourself. The application is done. It is a complete multiple window application for order management. You created the entire application without writing any code yourself. The small amount of custom code you did create was accomplished using the Outline Options tool that virtually writes the code for you. The rest was done with QuickObjects! Now get on the modem and wire the completed application over to cousin Chris in Hawaii.

Thanks for test-driving SQLWindows. I know you're convinced it's the application development system with the easy power to get client/server done.

Chapter 2

Creating Object Classes

Object-Oriented Programming: from buzzword to reality

You probably didn't realize it, but you're already an Object-Oriented Programming (OOP) programmer. You used objects and classes all along in Chapter 1. You used OOP when you selected the push button tool from the palette. You chose between a "Standard" class and a "cQuickCommander" class. In fact, the push button you added to the form was a working instance of the class you chose. Now that you're an OOP programmer, we're going to show you more of what you can do.

In this chapter, we are going to replace the existing push button with a better push button, something developers do all the time. The steps are as follows:

- Create a push button class from a push button object you used in Chapter 1.
- Add new functionality (a data field to show the elapsed time as you run the application).
- Create a totally new class, a timer class, with the new functionality.
- Use multiple inheritance to create a third class.

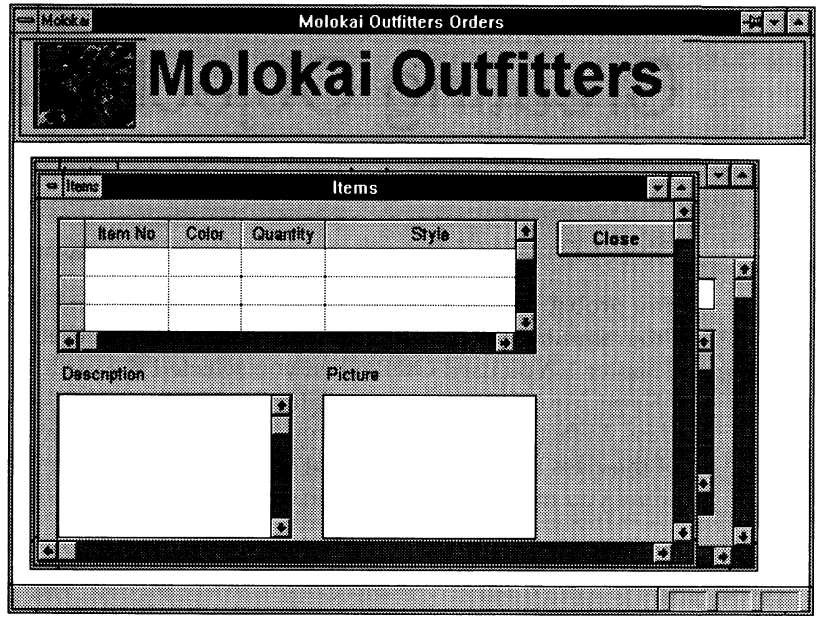
By the end of this chapter we will have given new meaning to the word "re-usability." You will know how to use the class editor to findwork you (or some other programmer) did earlier, (yesterday, or last year.) Through it all you'll see how SQLWindows comes with tools and a way of an organizing code that makes it easy to re-use functionality you have already created, to add new objects that do more neat things, and to use and combine your code to meet the ever changing needs of application development.

For a description of object-oriented programming concepts, see the Object Oriented Programming section of Chapter 3, *Roadmaps*.

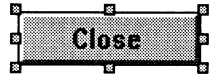
Re-using an object and building on it

First, let's re-use an object you created in chapter one to create a new class called **cClose_Button**.

1. Start SQLWindows and open the QUICK.APP file you created in Chapter 1. When you open the application, the **Items** form window is shown on top of the **Invoices** window.

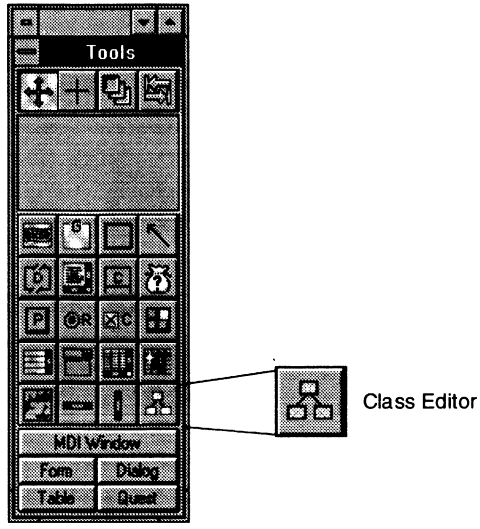


2. Click the **Close** push button on your **Items** form window.

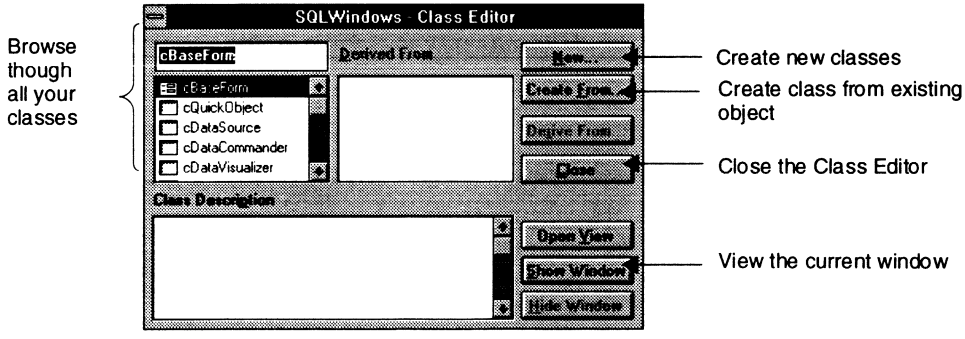


3. Press **F4** to bring up the tool palette, if necessary.

4. Click **Class Editor** on the tool palette.

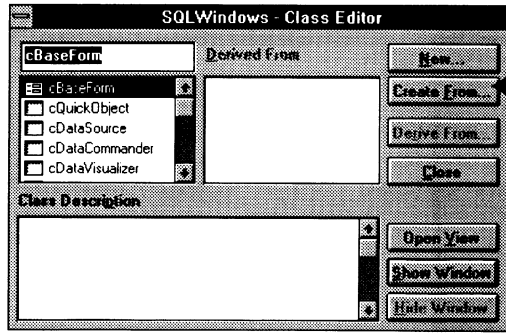


The **Class Editor** dialog box opens:

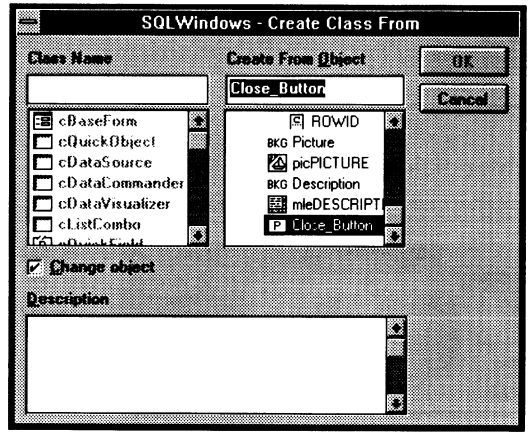


The Class Editor lets you browse through all classes you used to make objects in your application. You can also create new ones. You can click **Show Window** to view the appearance and properties of a class. It also lets you add to the appearance of objects that can be based upon a class. By providing a visual way to manage class definitions, the SQLWindows Class Editor helps you easily create re-usable classes and class libraries that can be leveraged by the rest of your team, even if the original authors created the objects years ago, or are no longer working with you.

5. Click the **Create From** push button in the **Class Editor** dialog box.



The **Create Class From** dialog box opens.



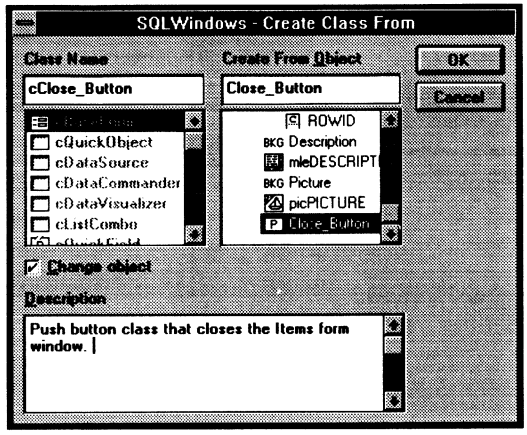
Now you re-use code to create a new push button class, called **cClose_Button**. You create this push button class from the **Close_Button** object in your application. The reason you are creating a new class is so we can add functionality to the push button to make it better. You want to leave the Standard default class of push button unchanged.

The **Close_Button** object is already selected in the **Create From Object** list box.

6. Type **cClose_Button** for the Class Name. **cClose_Button** appears in the text field below Class Name.

7. Enter **`Push button class that closes the Items form window.`** for the description. The description is a convenient way to keep track of the changes you make to the classes you create. Actually, you are writing documentation on the fly. These comments will be helpful when you or the next programmer picks up where you left off, months, or even years later.

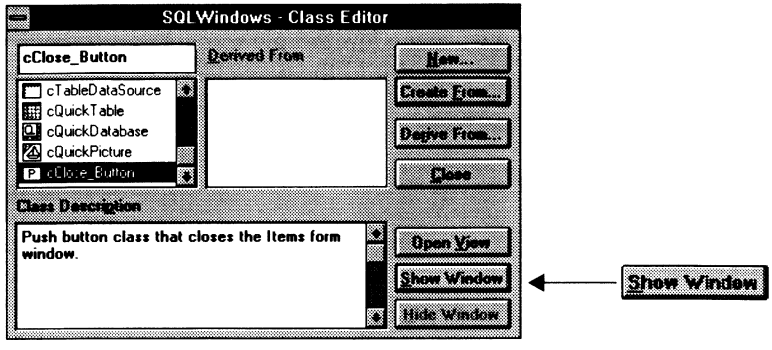
   After you have made your changes, the **Create Class From** dialog box looks like this:



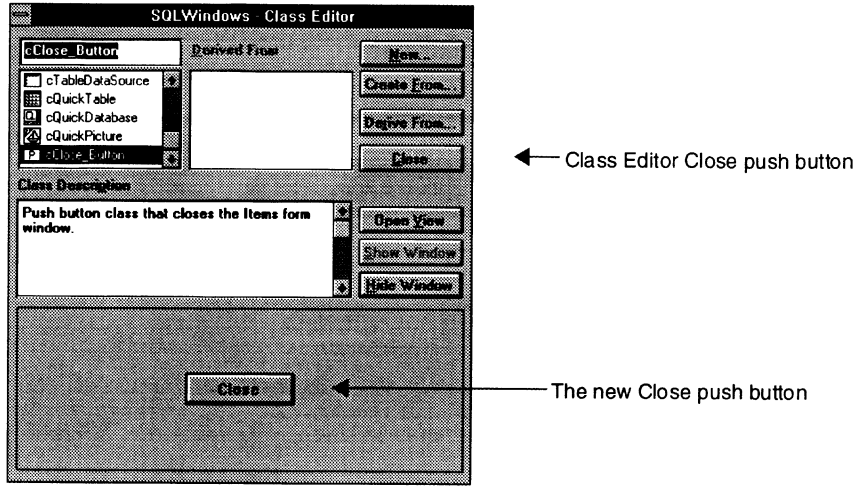
Tip: Make sure the Change object check box is checked. This changes the **Close_Button** push button to an instance of the class (**cClose_Button**) you are creating.

8. Click **OK**. The application returns to the **Class Editor** dialog box.

   In the **Class Editor** dialog box, click **Show Window**.



The next picture shows the **Class Editor** dialog box with the new push button, **Close**. You created a push button class, **cClose_Button**, from an existing object, **Close_Button**. You did this in just a few easy steps, but what you created is very powerful and flexible. When you create classes, you also create the foundation for other classes and the ability to re-use code.



9. Close the **Class Editor** by pressing the Class Editor Close push button.

Summary

You have created a push button class based on an object you made in Chapter 1, an object we decided in Chapter 2 needed to be more robust. You changed the original object from the standard default push button class to one you defined called **cClose_Button**. You used the data and behavior of an existing object to create a new class. In this way, your new class inherits the properties of the object with just a few simple steps. Everything is visual. You can see your progress as you build the new class.

Use the new push button

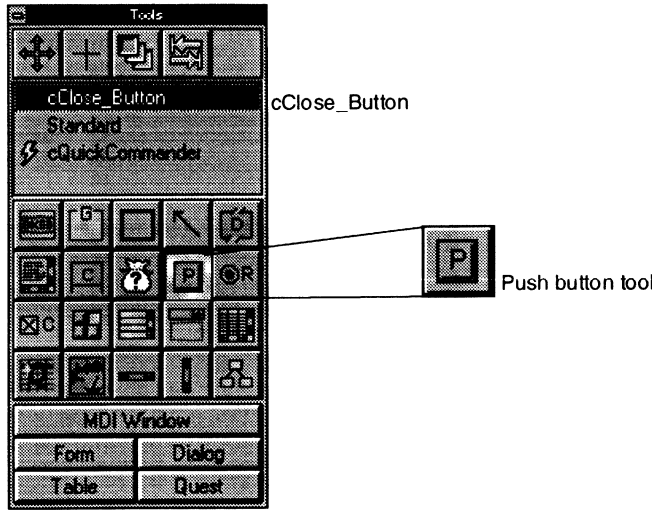
Now, let's experiment with the new push button. Try out the results and see how well integrated this new class is.

1. Open the **Items** window, if it is not already open.

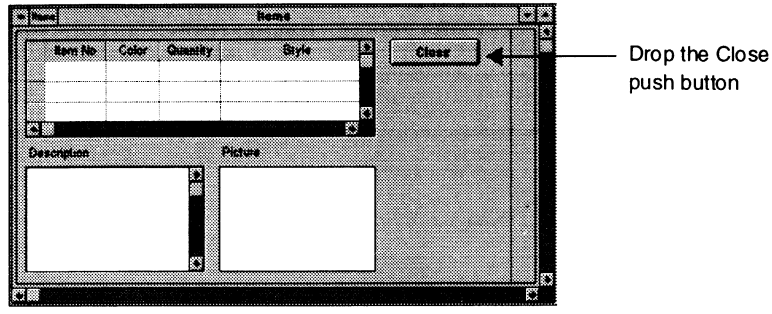
2. Select the **Close** push button in the Items window. Press the **Delete** key to remove the item. Remember the code is still in the application library as our new class.

3. Select the push button tool from the tool palette. The mouse pointer turns into an image of a small P when you move the pointer off the tool palette.

4. Select the **cClose_Button** class from the tool palette.



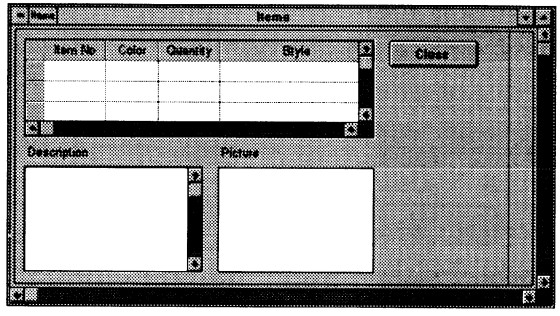
5. Drop the push button in the same place where the previous **Close** push button was located.



The substitution is now complete.

6. Click the **Run** push button on the tool bar at the top of the SQLWindows Designer, or select **User Mode** from the **Run** menu.

7. SQLWindows asks if you want to save your changes. Click **Yes**.
8. Click the **Items** push button on the Invoices form.
9. Click the **Close** push button to close the Items window. The **Close** push button works exactly as it did before.



10. Select **Close** from the System menu to return to the SQLWindows Designer.

Summary

The swap of one object for another object is complete, but we haven't yet added any new functionality. We've laid the foundation for re-using the functionality you created in chapter one. So far the **Close** push button works just like its previous version, having copied its behavior. You have just used inheritance.

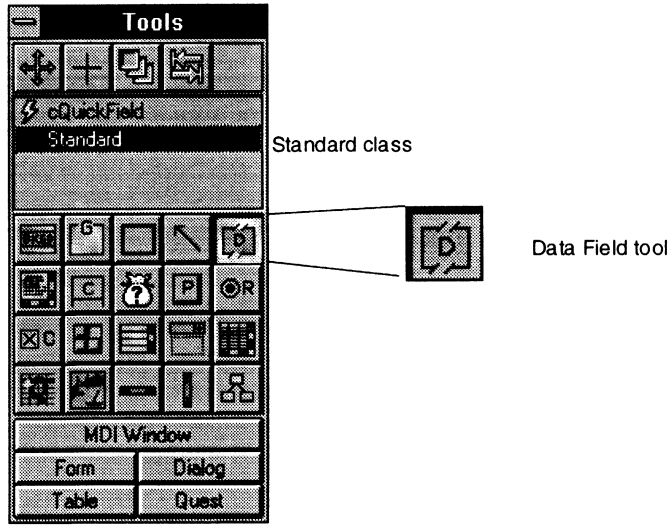
Create a new function and use it to make a new class

Now that you understand the basics of class creation, let's go on to another example. SQLWindows doesn't come with a timer class, so we will create one from scratch (and show you in the process how easy it is to create all of the classes you'll need in your business, or in your third party library.)

We want the push button to have a timer attached to it. Instead of just adding to the existing push button (you may have other needs for the simple push button without the timer attached), let's create a separate timer class and then hook them together.

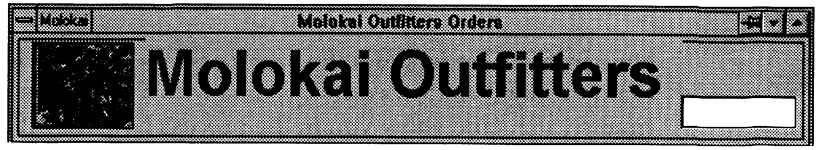
First you create a standard data field. This field will show the clock running while you are viewing the invoice details in the Items window. This data field is called **Time Elapsed**.

1. Click the **Data Field** tool in the tool palette. The mouse pointer turns into an image of a small D.
2. The tool palette list box shows the **cQuickField** and **Standard** classes. Choose the **Standard** class.

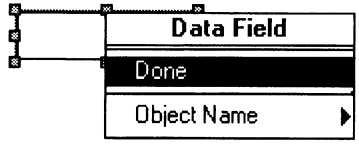


3. Move the cursor over the Molokai Outfitters form and drop the data field on to the window's tool bar.

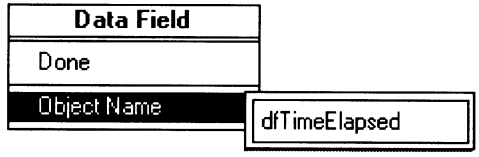
The drawing shows the new, untitled data field located to the right of **Molokai Outfitters** on the tool bar.



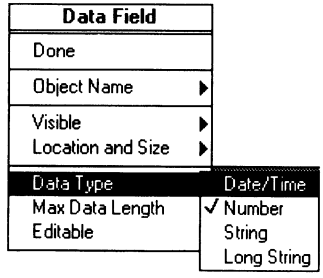
4. Place the cursor over the new data field and click the right mouse button to bring up the Customizer.



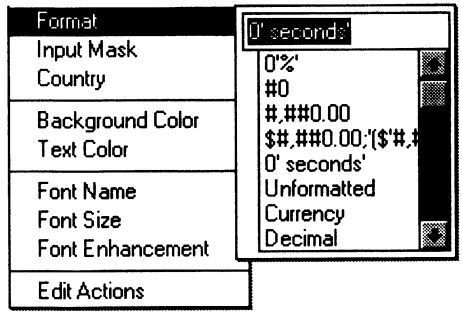
5. Change the object name to `dfTimeElapsed` and press **Enter**.



6. Change the data type to **Number** from the Customizer. When you change the option to Number, a check mark appears to verify your choice.



7. Set the format to 0' seconds'. Several formats are offered. However, you need to type in 0' seconds'.



8. Click **Done** in the Customizer.

Summary

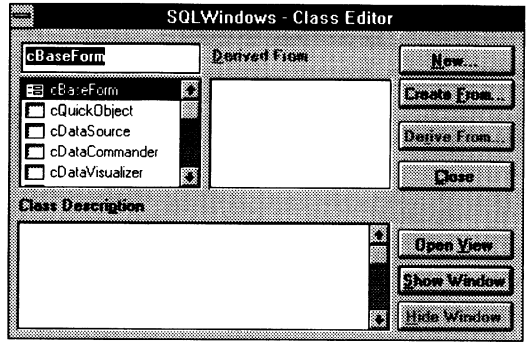
Now you have a new data field, called **dfTimeElapsed**, placed on the application's tool bar. This new functionality is present in the application, but the timer is not yet complete. You might think of this as prototyping, being able to put something into the application that isn't done yet. So, let's add more behavior to it.

Create the timer push button class

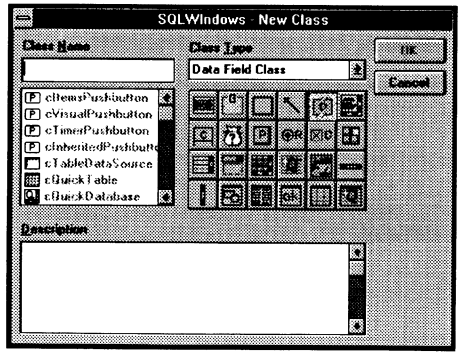
The timer data field needs to increment every second to be able to show how long you've been doing a task in the Items form. In this section, you create the timer push button class. Then, in the next section, you will add more functionality and tie the push button to the timer class.

☞ 1. Select **Class Editor** from the tool palette.

The **Class Editor** window opens:

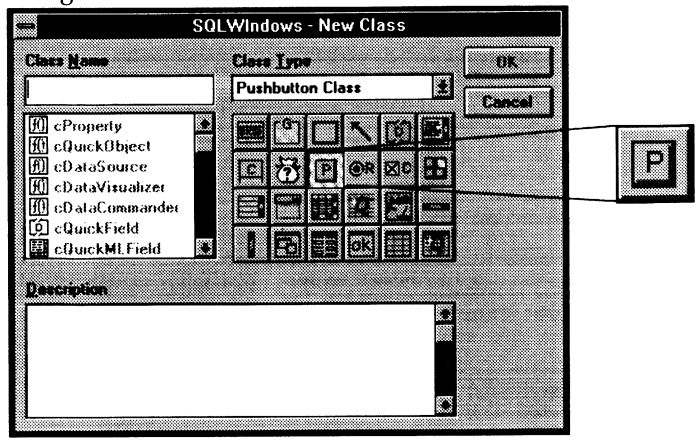


2. Click the **New** push button. The **New Class** dialog box opens.



The **Class Type** displays or defaults to the last type you created which was **Data Field Class**.

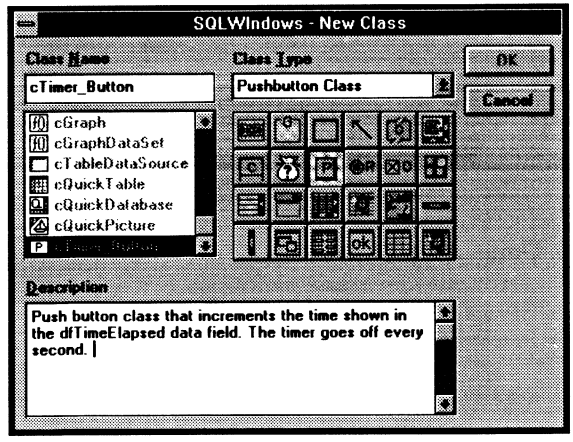
3. Click the **Push Button** tool in the New Class tool palette. The **Class Type** changes to **Pushbutton Class**.



4. Type **`cTimer_Button`** for the **Class Name**.

5. Type **`Push button class that increments the time shown in the dfTimeElapsed data field. The timer goes off every second.`** for the description.

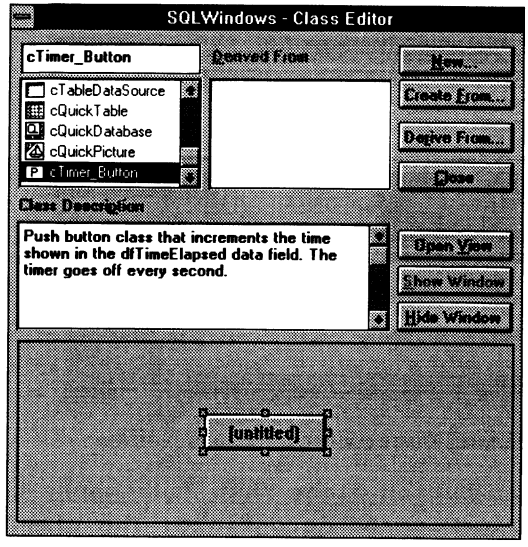
   After you make all your entries, the **New Class** dialog box looks like this:



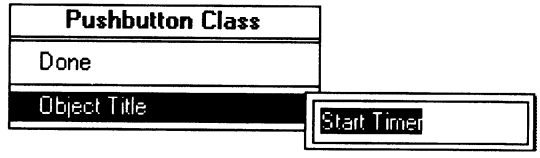
6. Click **OK**. The Class Editor opens.

   Now you need to give the push button class the right title.

1. Click the **Show Window** push button in the **Class Editor** dialog box. The **Class Editor** dialog box opens to show the new, untitled push button class.

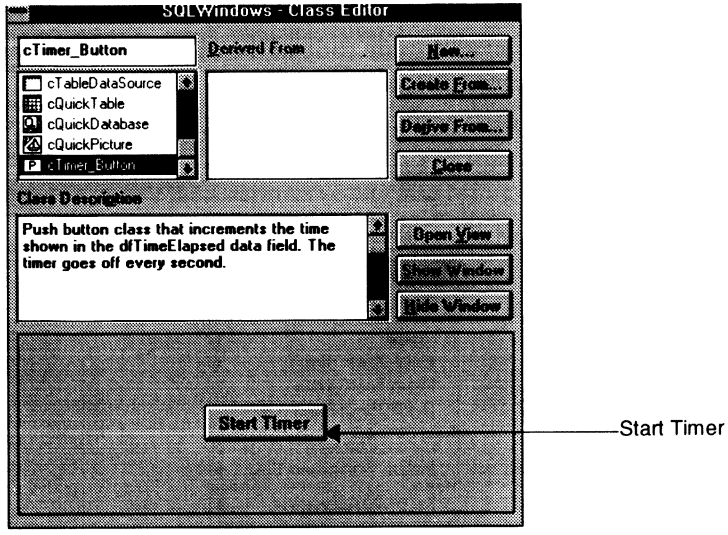


2. Place the cursor over the untitled push button and right mouse click to open the Customizer.
3. Choose **Object Title** and enter `Start Timer`.



4. Click **Done** in the Customizer.

Your finished window looks like this:



As you can see, **cTimer_Button** has changed to **Start Timer**.

5. Resize the **Start Timer** push button to an appropriate size if necessary.

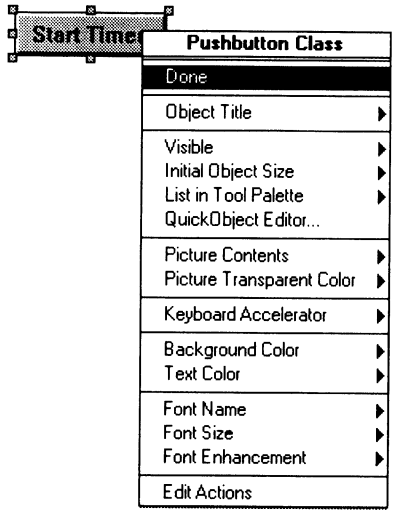
Summary

You have created a new class, **cTimer_Button**. When you run the application, the timer will be set off every second to show the amount of time elapsed during use of the Items window. You complete this functionality by adding the code in the next section, *Add functionality to the timer push button class*.

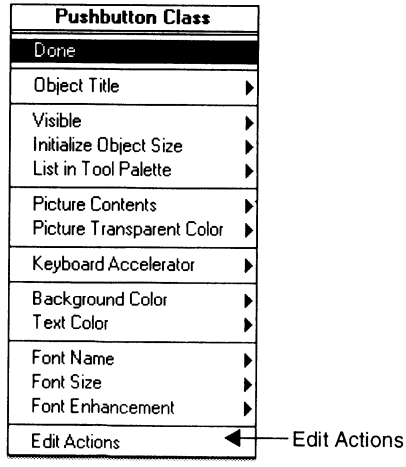
Add functionality to the timer push button class

Now you add some code to the cTimer_Button class to set off the system timer each time you run the application.

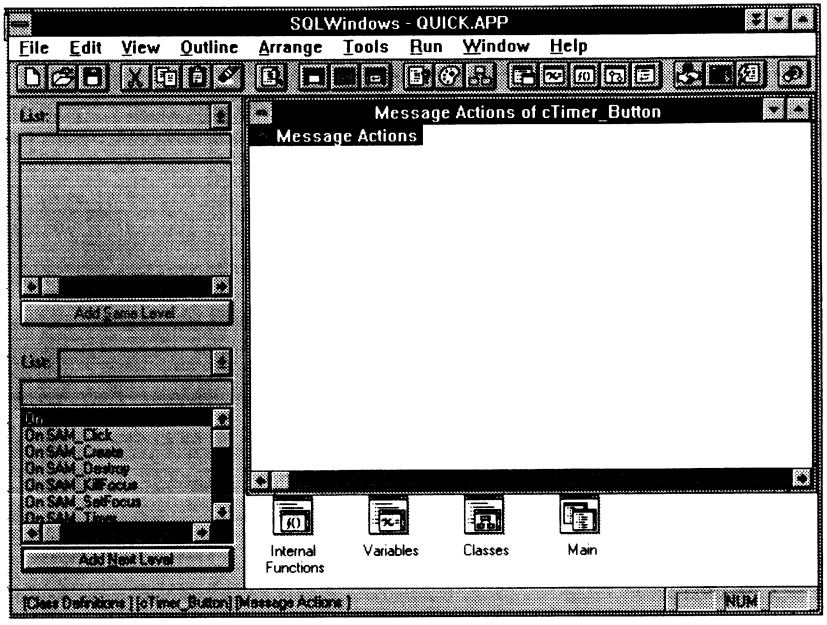
1. Place the cursor over the **Start Timer** push button and click the right mouse button to bring up the Customizer.



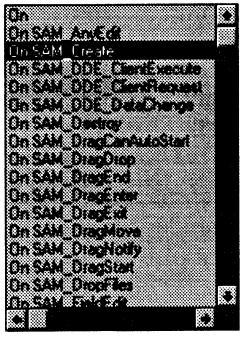
2. From the Customizer, click **Edit Actions** to enter code for the cTimer_Button class.



3. Press **F2** to bring up the **Outline Options** bar if not already up.



4. Scroll through the Outline Options bar to find **On SAM_Create** and double-click it.



   ✦ On SAM_Create

   appears in your outline. The Outline Options bar changes to show you the available commands you can use with On SAM_Create.

5. Double-click **Set** in the Outline Options bar.

   ✧ Set

   appears in your outline. The Outline Options bar displays the available functions you can use with Set.

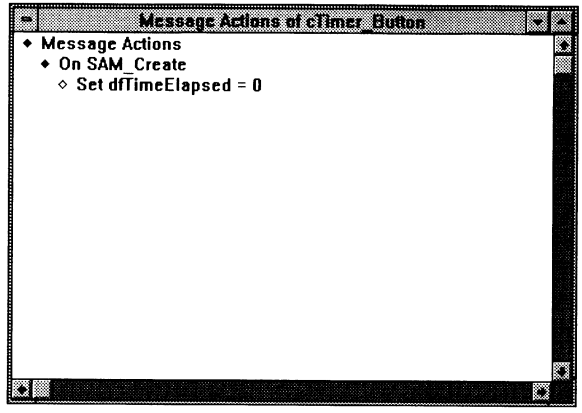
6. Enter `dfTimeElapsed = 0`. Press **Enter**.

   The code now looks like this:

   ✦ On SAM_Create
   ✧ Set dfTimeElapsed = 0

   This initializes the data field to 0 seconds when you use the Items part of the Molokai Outfitters application.

   At this point, the Message Actions section of your outline for this data field looks like this:



7. Double-click **Call** in the Outline Options bar.

   ✧ Call

   appears in your outline. The Outline Options bar displays the available functions you can use with the Call.

8. Double-click **SalTimerSet** in the top part of the Outline Options bar.

   ✧ SalTimerSet (Window_Handle, Number, Number)

   appears in your outline. `Window Handle, Number, Number` is already highlighted.

9. Enter `hWndItem, 1, 1000` by typing over the Window Handle section so that this part of the code looks like this:

   ✦ On SAM_Create
   ✧ Set dfTimeElapsed = 0
   ✧ Call SalTimerSet(hWndItem, 1, 1000)

   hWndItem is a system variable that refers to the cTimer_Button itself. The second parameter is an ID for the timer. The third parameter is the timer interval in milliseconds.

10. Click **On SAM_Create**.
11. Scroll through the Outline Options bar to find **On SAM_Timer** and double-click it.

    ✦ On SAM_Timer

    appears in your outline. The Outline Options bar changes to show you the available commands you can use with On SAM_Timer.
12. Double-click **Set** in the Outline Options bar.

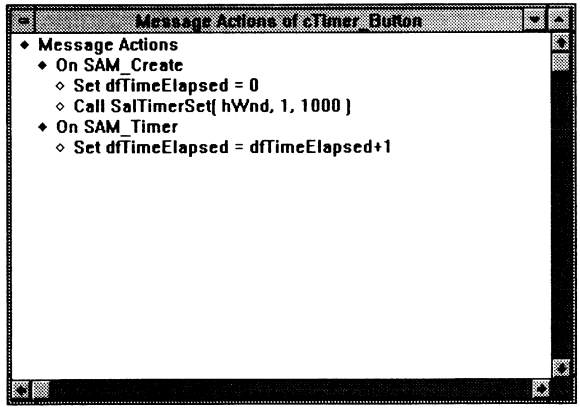
    ✧ Set

    appears in your outline.
13. Now, enter `dfTimeElapsed = dfTimeElapsed +1`

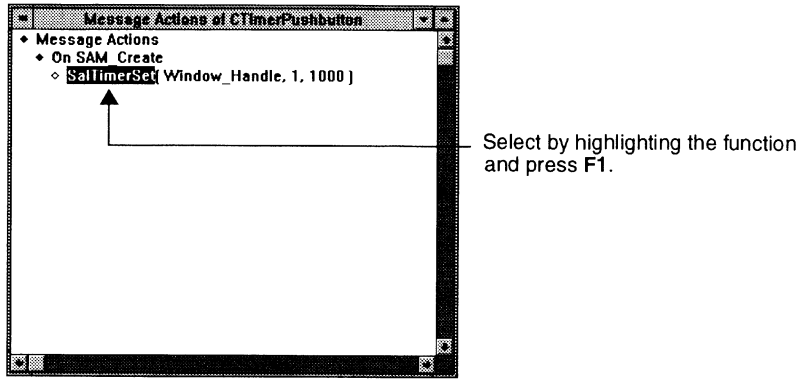
    This part of the code looks like this:

    ✦ On SAM_Timer
    ✧ Set dfTimeElapsed = dfTimeElapsed +1

    Your final, finished code looks like this:



Tip: You can read online information on the functions you use. Highlight the function you need to know about and press **F1**. The following example shows what to select for SalTimerSet.



14. Bring the **Items** window to the top by clicking anywhere in it.

Summary

Now the cTimer_Button push button has a timer designed to go off every second.

Specifically, the code activates a timer every 1000 milliseconds, i.e., 1 second. When the timer is activated, it sends a SAM_Timer message to the push button. hWndItem is a system variable that refers to the current object, i.e. the push button.

Most importantly, you have refined the behavior of the objects you are working with. Some lights should be going off for you about OOP programming, about inheritance, about step by step iteration, about making something, enhancing it, remaking it again and again and again.

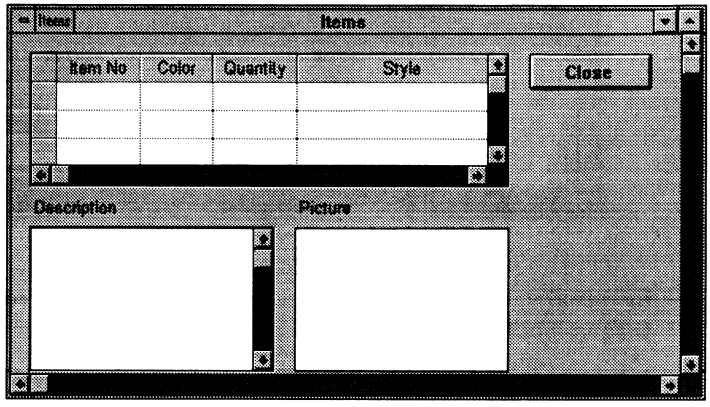
This is the work flow of the Quick Objects programmer. You always know where to find your previous work. It is presented in clear outline form. You should be convinced by now that reusability of code means returning to previously done work and giving it more pizazz any time you want.

From a professional point of view, we want you to change how you feel about application development. We want you to get a feel for the flow and rhythm of how you will build applications with SQLWindows. We hope to give you some ideas on how this can help your business and serve your clients.

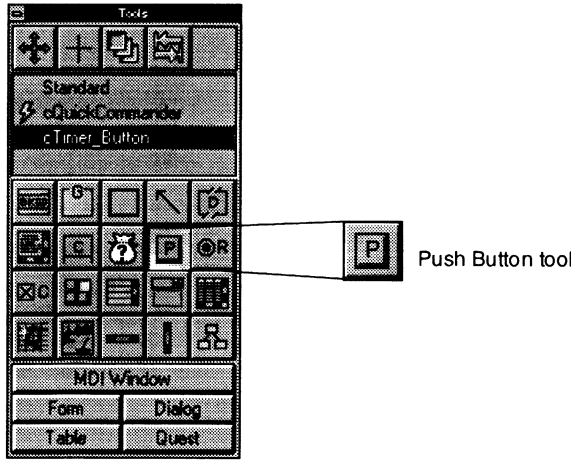
Use the timer push button class

Now let's use the timer class.

1. Resize the **Items** window to make room for the **Start Timer** push button. To resize the window, click on it. Then, move the cursor to the edge until it becomes a small arrow. Hold down the mouse button and pull.

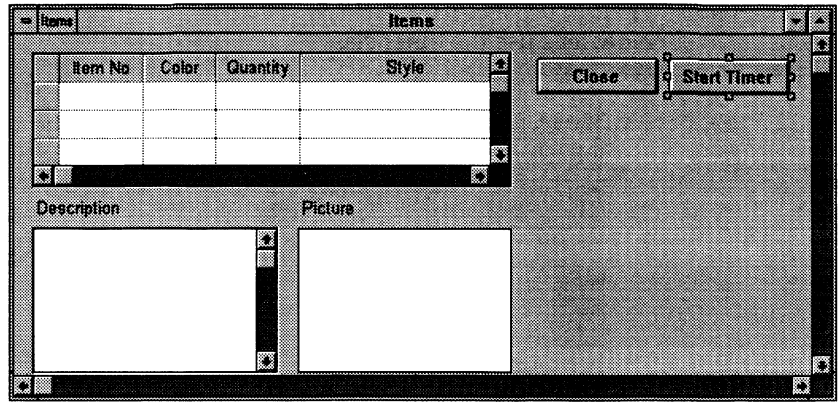


2. From the tool palette, choose the Push Button tool.



3. Select the **cTimer_Button** class.

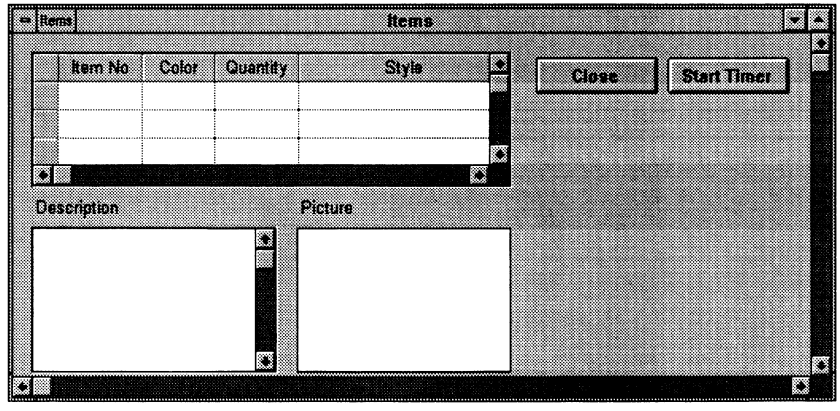
4. Drop it on the form to the right of the **Close** push button.



You created and dropped the Start Timer push button on the Items form. Eventually, you will see the results of the code you wrote when you created the push button class.

Now we derive a *new* class, which combines the properties of the two classes you just created. This is called *multiple inheritance*.

Right now, the application looks like this:

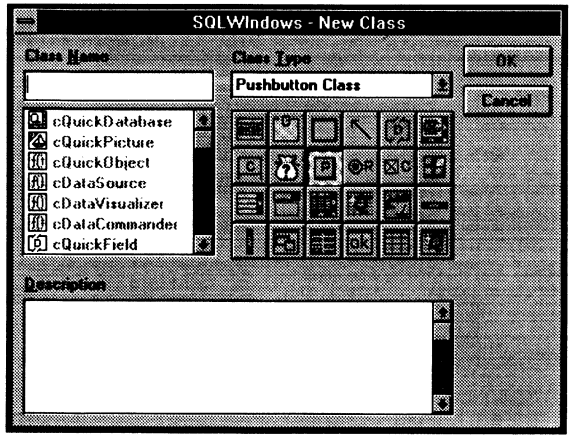


1. Delete the two push buttons you created: the **Close** push button and the **Start Timer** push button. Click each push button and press the **Delete** key.

2. Choose the **Class Editor** tool from the tool palette. The **Class Editor** window opens.

3. Click the **New** push button. The **New Class** window opens.

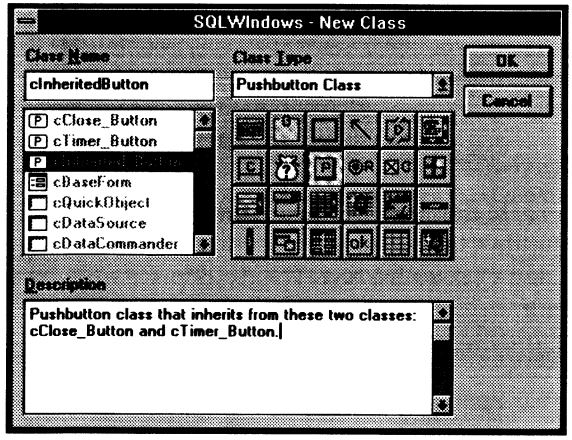
4. Select the **Push Button** tool from the tool palette.



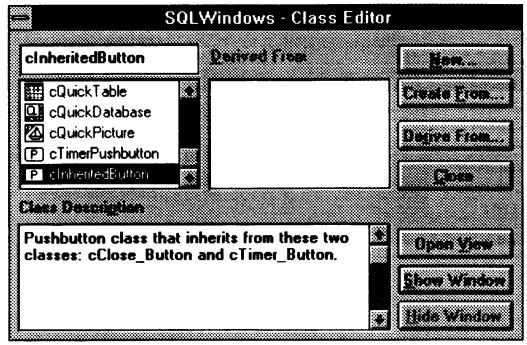
5. Name the new class **cInherited_Button**.

6. For the description, enter: **`Pushbutton class that inherits from these two classes: cClose_Button cTimer_Button.`**

   The following picture shows that you have named the new class **cInherited_Button**, that the class type is **Pushbutton Class** and that this class inherits from two other classes, **cClose_Button**, and **cTimer_Button**.



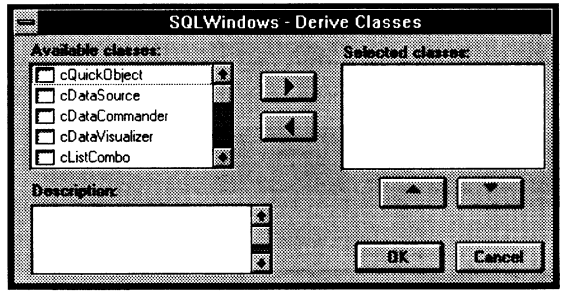
7. Click **OK** in the **New Class** window. You return to the **Class Editor** window.



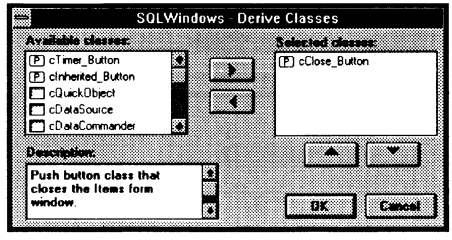
Now we derive the properties from the two classes we just used, cClose_Button and cTimer_Button.

☞ 1. Click the **Derive From** push button in the **Class Editor** window.

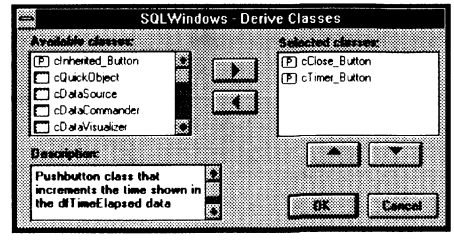
The **Derive Classes** window opens.



2. Choose, in this order, **cClose_Button**, and **cTimer_Button** from the **Available Classes** list box. As you *double-click* each class, it appears in the **Selected Classes** list box. The description appears for each class as you select it.

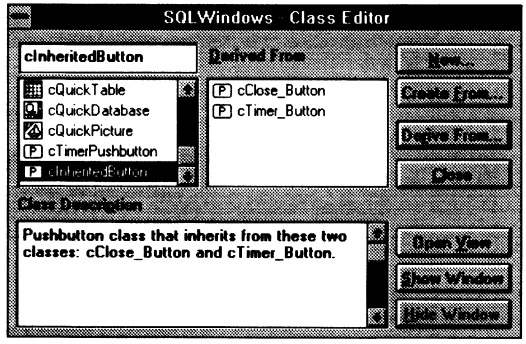


Choose cClose_Button



Choose cTimer_Button

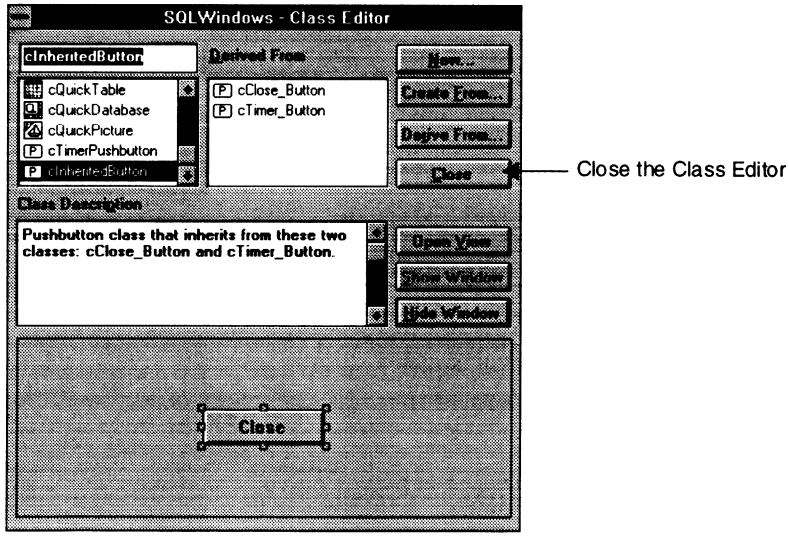
3. Click **OK**. You return to the **Class Editor** window.



Show Window 4. Click **Show Window**.

5. If necessary, resize the derived push button class so you can see the icon and text.

   Your window should look like this:



Close 6. Close the **Class Editor** by pressing the **Close** button.

Summary

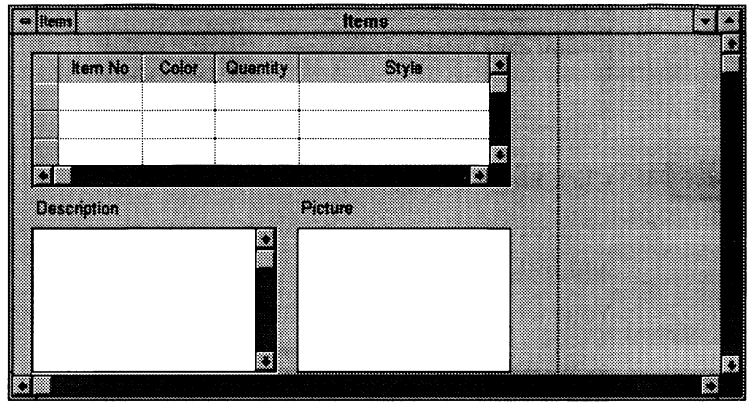
You have created a class that derives its properties from two different classes. This sets up the class to be used as a timing device in the application. Once

again, the power and flexibility of SQLWindows object-oriented programming has been proven. Programming was never this easy before!

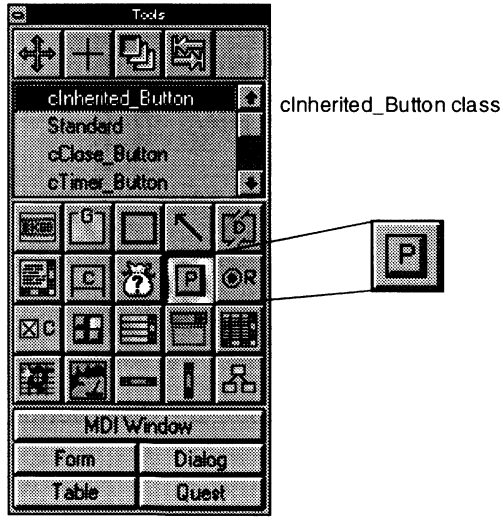
Use the derived class

Now we experience using the derived class, **Close**.

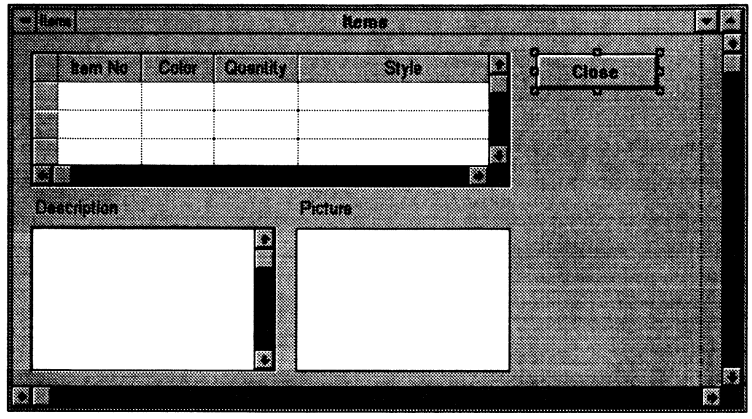
1. Open the **Items** window.



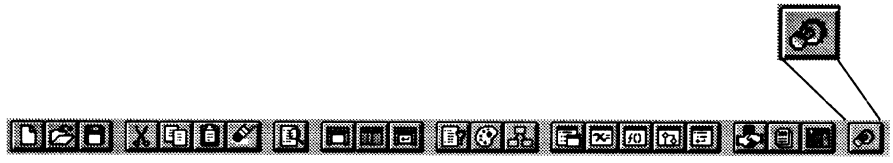
2. Choose the push button tool from the tool palette. Drop this push button from the **cInherited_Button** class.



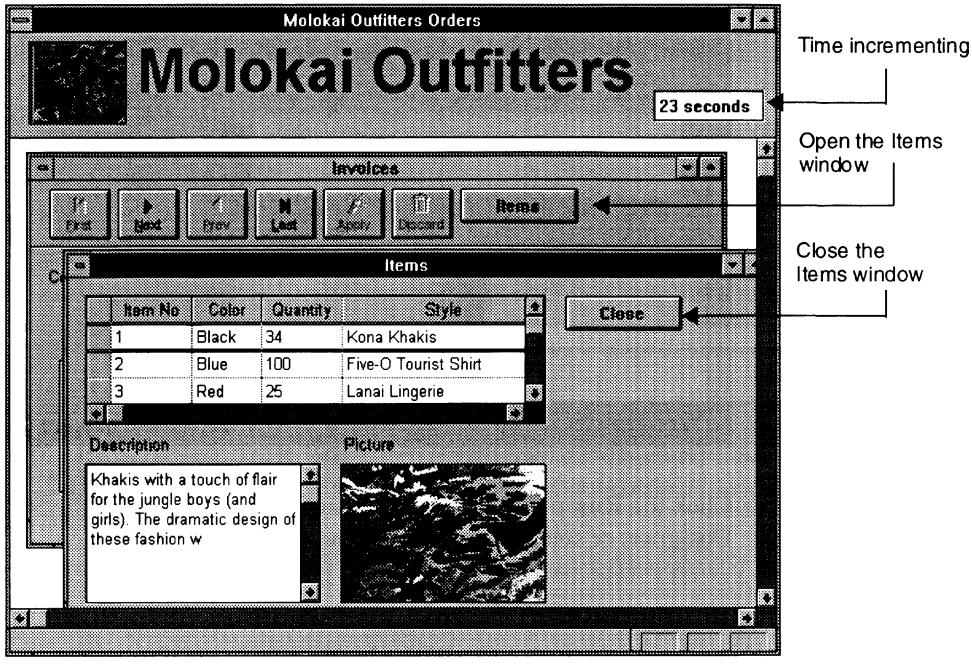
3. Place the new **Close** push button in the window where the previous **Close** push button was before.



☞ 1. Click the **Run** push button on the tool bar at the top of the SQLWindows Designer, or select **User Mode** from the **Run** menu.



SQLWindows asks if you want to save your changes. Click **Yes**.



Press the **Items** push button in the **Invoice** window to open the **Items** window. This way, you can easily find out details of any Invoice.You'll see the time incrementing in the data field in the tool bar of the Molokai Outfitters application. Then, when you want to close the Items window, just press the **Close** push button.

Now, let's use the new **Close** push button. You can close the Items window and toggle back and forth between the Items window and the rest of the application.

Congratulations!

You have just added significant Object Oriented Programming (OOP) functionality to the Molokai application. In the process, you have learned a lot about OOP. (And you thought OOP was hard.) Let's reflect for a moment.

Object Oriented Programming (OOP).

OOP is a new approach to creating applications, extending applications, and maintaining applications over time. For some programmers, it represents a whole new way of thinking about programming, about how programmers relate to other programmers, about how code created at one point in time (even during prototyping) relates to later development and long-term application maintenance. In the end OOP is a strategy that aims to increase the quality of our systems development efforts. The word "reusability" takes on strategic importance.

Creating Class Libraries.

In these two chapters, you have created the beginnings of your own OOP library of functions. There is a lot more to learn and there are hundreds of new classes to be built, but you have experienced the basic ebb and flow of the processes by which you will approach the building of your own libraries. Whether this library is used to deposit your corporate business practices, or the library is used as a start of your own third party library, other developers can now acquire, and re-use, and inherit from your proven code base.

Browsing Class Libraries.

Creating code is one thing. Organizing code so that it can be located and built on is something else altogether. SQLWindows organizes application objects and code as an Outliner. The object Class Editor assures you that you always know where to look for work you or somebody else has previously done. The SQLWindows Class Editor organizes all your programming efforts, presents your user forms as outlines, actually takes you to specific details in the code, and even helps you create documentation that stays with the classes at every level.

Using Class Libraries to Extend Functionality.

You see how SQLWindows comes with tools and an organization that invites you to call up and build on foundation classes and objects without worrying that you will "break" the applications when you elect to add something new. We saw this when we walked through the creation of a new-and-improved class and then used the new class to replace an existing object in the application. What you were doing was creating an abstract class from a

previously built object and then created a new class that inherited properties from more than one class.

In the end, Object Oriented Programming (OOP) with SQLWindows is about making it easier to be successful at the business of making and re-making applications. We have attempted in two brief chapters to outline the 'Gupta way' of making your job as a professional developer easier, more organized, and more profitable.

Chapter 3

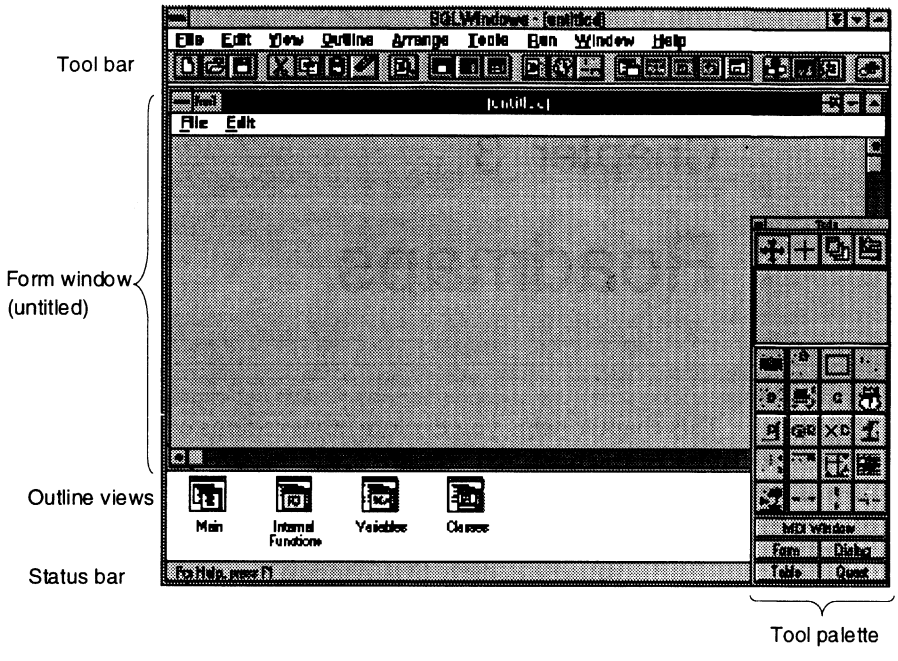
Roadmaps

This chapter helps you explore other regions of SQLWindows. Now that you've taken the SQLWindows Solo test drive, and created classes of re-usable objects, this chapter gives you a stack of roadmaps to take with you:

- **Exploring the SQLWindows Designer**. Take a closer look at the tools you use to drive SQLWindows.
- **What are QuickObjects?** Get familiar with QuickObjects, the intelligent, pre-defined objects that provide you with a fast and easy way to create client/server applications.
- **Exploring Object-Oriented Programming.** Familiarize yourself with some basic concepts and advantages of object-oriented programming.
- **Explore SQLWindows' Team Programming**. Take a look at features that help you keep track of large projects involving teams of developers.

Exploring the SQLWindows Designer

The SQlWindows Designer is a highly graphical and easy to use environment:



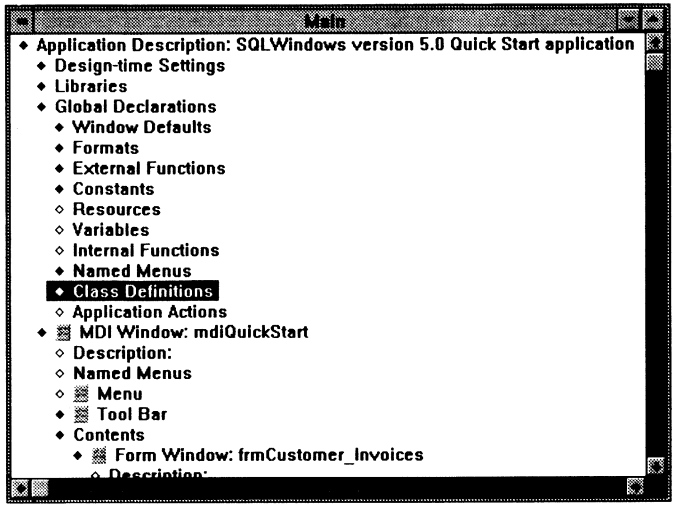
The tool bar, a Form window, Outline views, status bar, and the tool palette make up the SQLWindows Designer environment.

- The **tool bar** provides icons that help you build and edit your application quickly and easily.
- The **Form window** is where you graphically build the user interface of your application without having to write code.
- When you build an application in the Form window, SQLWindows automatically adds the appropriate code to your outline for you. **Outline views** enable you to see textual descriptions of different sections of your application code as you build.
- The **status bar** shows the setting of the Num Lock, Scroll Lock, and Caps Lock keys, and provides other helpful information.
- You use the **tool palette** to add graphical objects like push buttons and data fields to the Form window.

SQLWindows Outliner shows your application code as you build

The SQLWindows Outliner shows your application code as you build the application graphically in a visual, tree-like outline.

The Outliner looks like this:



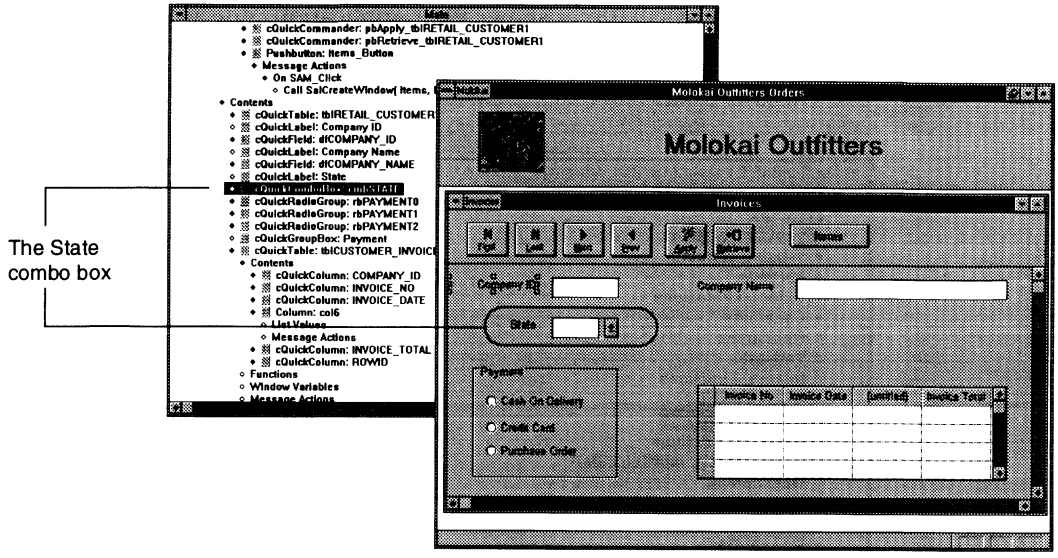
You can collapse or expand the entire application outline or specific sections in order to view higher or lower levels of detail. The Outline window is unique to SQLWindows. It makes it very easy for you to examine the entire content of your application at any level of detail. As your project grows larger, the Outliner is of invaluable assistance in navigating through project detail. You can even print out the entire application outline for off-line review.

The Outliner corresponds to your application

The Form window and the Outliner are two representations of the same application. The Form window shows you the application as the end user sees it.

Objects in your form window and the same objects in the Outliner are directly related to each other. This association lets you easily find your place in the outline. You can click any object in the Form window, and you are

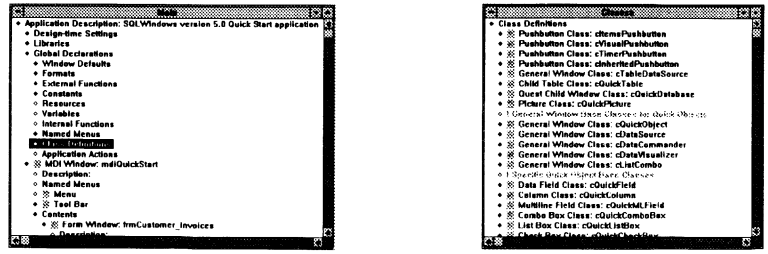
automatically positioned on the object in the Outline.



When you select an object in your application, SQLWindows highlights the corresponding code for that object in the Outliner.

Outline views take you directly to sections of the outline

The Outline views provide you with views of different parts of your application. While you build an application, you can click on each view to display that section of the outline of your application. The larger the application, the more invaluable these views become for managing your application. Here are examples of two of the outline views:



The Main view shows your entire outline, and highlights the part of the code you are working on. The Classes view shows the Class Definitions section of your outline.

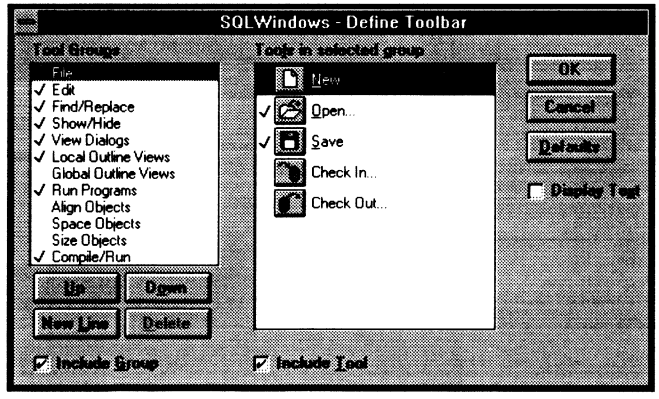
Tool bars perform actions quickly for you

Tool bars provide icons that help you build and edit your application quickly and easily. They also provide quick access to various components of SQLWindows. This picture shows you a typical SQLWindows tool bar.

Create
Open
Save
Cut
Copy
Paste
Undo
Find
Select Window
Hide Window
Show Current Window
Activate outline options
Activate tool palette
Create/edit new class
Create a new view
View variables
View functions
View message
View window
TeamWindows
ReportWindows
Quest
Run

Each icon represents an action you can perform quickly with SQLWindows.

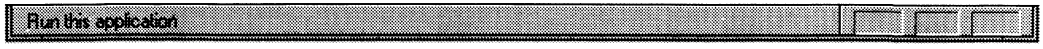
You can customize the tool bar, quickly and easily. Select **Preferences, Toolbar** from the **File** menu in SQLWindows.



In this dialog, highlight the icon group you want in the Tool Groups box on the left. The tools in that group display on the right. **Include Group** includes the group on your tool bar. **Include Tool** selects or deselects each tool in the group for your tool bar. The **Up** and **Down** buttons position the group on the tool bar. The **New Line** button starts a new tool bar. You can even **Display Text** under each icon to help you remember its function.

Status bar keeps you informed

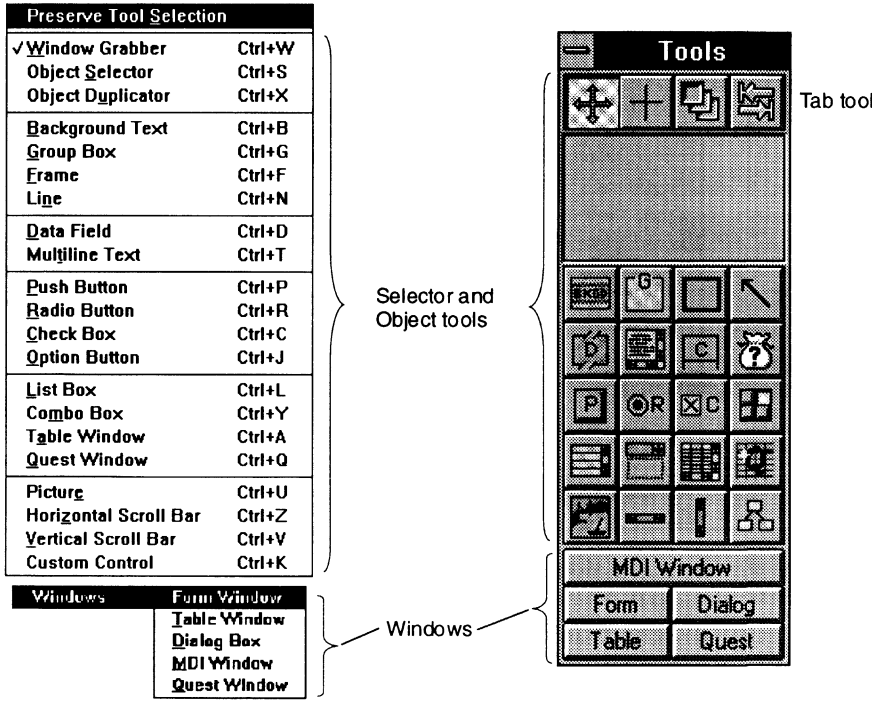
SQLWindows has a Status bar at the very bottom that shows the setting of the Num Lock, Scroll Lock and Caps Lock keys. The status bar also displays a message that shows the current menu pick or tool bar pick. You can turn the status bar on and off by selecting or de-selecting the status bar menu item on the View menu.



Tool palette keeps tools handy for you to use

The floating Tool Palette appears when you start SQLWindows. The tool palette contains icons representing a rich choice of graphical objects you can easily add to an application as part of your interface design.

You can select tools from the tool palette or from the **Preserve Tool Selection** and **Window** menus in SQLWindows.



Menu items on the left correspond in order to each tool in the palette on the right, except for the Tab tool in the upper right corner of the palette.

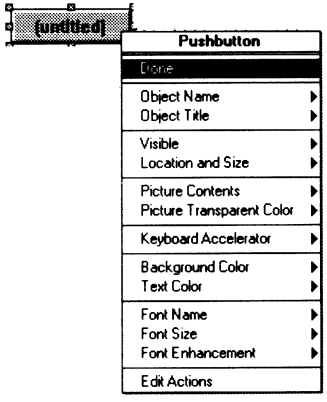
The tool palette displays when you start SQLWindows. To display the tool palette at other times, press **F4** or select **Tool Palette** from the **View** menu.

Using the tool palette to add objects to an application is easy:

1. Click the icon for a data field, push button, picture, or other object. These graphical entities are called objects.
2. Move to the Form window, click, and the object appears. SQLWindows "drops" the object onto the form. This technique is called "Drag and Drop".

Customizer lets you name and modify objects

When you first create applications in SQLWindows, it is similar to laying out shapes on a canvas. In SQLWindows, the shapes are the objects that you work with. The Customizer lets you modify objects by giving them attributes such as a name, a title, a color, and so on.



Customizer is unique for each object, tailored for the ways in which you use an object. This Customizer is for a push button object.

To display the Customizer, place the mouse pointer over the object and click the right mouse button.

You can set attributes for any graphical object you drop on your Form window. The Customizer gives the developer complete control over the look and feel of graphical objects.

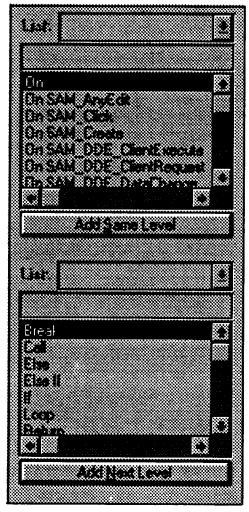
Adding application code is easy with SAL

The logic in a SQLWindows application is written in SAL, the **S**QLWindows **A**pplication **L**anguage. SAL is powerful, simple to use and easy to learn. Developers who have previously used COBOL or BASIC can quickly grasp SAL.

The Sal code in your application is stored as part of the application Outline. This means that the benefits of a unique, collapsible outliner are also available to all application code.

Outline Options bar displays coding options as you work

From the **View** menu, select **Options Bar** to display the Outline Options bar. As you add functionality to your application, this tool lists coding options that are appropriate to the section of the Outliner you are working in.



Depending on your location in the application Outliner, the Outline Options bar provides you with a context-sensitive list of suggestions, in the upper and lower halves, for the next line of code to write.

The Outline Options bar has a unique, context-sensitive, fill-in-the-blanks approach to coding which is exceptionally useful to new users of SQLWindows. Use of the Outline Options bar is helpful but optional. With SQLWindows, you can either enter the code directly or use the Outline Options bar for "fill-in-the-blanks" coding.

Online Help provides quick look-up for information

You can get context-sensitive help at any time by pressing **F1**.

What are QuickObjects?

QuickObjects are intelligent, predefined objects that provide you with an easy point-and-click alternative to writing code when you create client/server applications. The purpose of QuickObjects is to speed up development time and to reduce the amount of testing necessary by providing components with proven functionality.

The QuickObject architecture reduces the time needed for new client/server developers and at the same time, does not compromise the flexibility that experienced developers need to create large-scale applications. The QuickObject architecture does this by creating an intuitive architecture that is powerful and extensible.

Each QuickObject component is independent, yet the interaction between the components is specific. For instance, the cQuickField QuickObject will always interact the same (*specific interactions*) with any available data source (*independence*). This allows QuickObjects to be used in the same easy-to-learn-and-use manner, whether they are used in their standard or extended form, or whether they are supplied by Gupta or created by a third party.

QuickObjects save you time

There are many benefits to using QuickObjects:

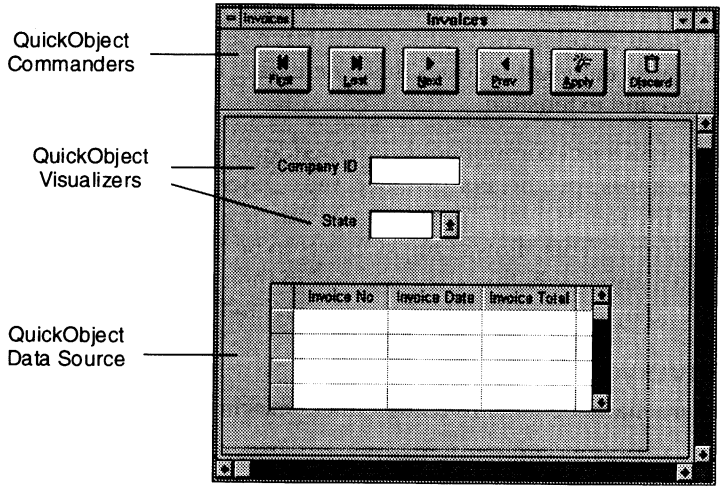
- You can confidently use the "no programming" approach with QuickObjects. For example, in order to create a data field that shows you the value of a column table, you would have to write many lines of SQL code. With QuickObjects, there is no need to write any code.
- QuickObjects are based on object-oriented concepts and are fully extensible. For example, you can take a QuickObjects class such as a Fetch All Records push button and derive your own specialized Fetch Records for customers who pay by credit card.
- You can use QuickObjects right away to build your own application, for your own specific purposes. You can extend the use of QuickObjects by manipulating them to create new classes with user customized behavior.

Data sources, visualizers, and commanders are ready to use

There are three types of QuickObjects, which correspond to the three major components of a client/server architecture:

- Data Source
- Visualizers
- Commanders

This picture shows examples of each on a SQLWindows form.



Data Sources link your application to databases and email

The first type of QuickObject is the Data Source. This is a very intelligent object that controls all aspects of the interaction between the client application and the data. This interaction includes:

- Transparent access and interaction with the server data.
- Passing data to the graphical objects in the application.
- Receiving any changes to the data made by the user.
- Specifying the user commands appropriate for the data.
- Controlling the enabling and disabling of user commands, based on the state of the data.

For instance, a QuickObject data source knows how and when to establish an initial connection with the data it accesses, how to navigate through the data in response to user commands, and what user commands are appropriate for

the accessed data. Most of the intelligence in the application is located in the QuickObject data sources.

QuickObject data sources work with traditional data sources, such as relational tables or views, or with non-traditional sources of data, such as email systems.

Typical data source objects are:

Database QuickObject This is an object that interacts with structured databases. The databases may be relational (such as SQLBase, Oracle, and Sybase) or personal databases (such as dBase and Paradox) via ODBC.	This QuickObject can be contructed using these tools from the tool palette:  cQuickDatabase  cQuickTable
Email QuickObject This is an object that provides you with an easy interface to your electronic mail system. Supported mail systems are VIM (Lotus cc:Mail, Lotus Notes), MAPI (Microsoft Mail) and MHS.	This QuickObject can be contructed using these tools from the tool palette:  cQuickEMail Note: You need to include the Email QuickObject library QCKMAIL.APL in your application.
Lotus Notes QuickObject This is an object that provides you with an easy interface to documents and forms in Lotus Notes databases.	This QuickObject is available in other editions of SQLWindows.

Data source QuickObjects provide a quick and easy connection to the data that you use in selected tables and windows. This form of QuickObjects allows you to define the data access you want to connect to and work with on specific windows. You make your data source choice from the tool palette or from the dialog box in the Customizer. Data sources are not necessarily visual and can be the connection for the Visualizer QuickObject and commands invoked by the Commander QuickObjects.

Visualizers display data instantly

Visualizer QuickObjects control the presentation of the server data to the user and any changes the user might make to the data.

Examples of QuickObject visualizers are data fields, list boxes, combo boxes, and radio buttons. You associate a visualizer with a particular data source at design time. This simplifies your development effort, since a QuickObject visualizer does not change with different QuickObject data sources, regardless of the type of data the QuickObject data source is accessing.

The following chart lists the QuickObject visualizers that come with SQLWindows 5.0 and the tool you use to create them.

The following chart lists the QuickObject visualizers that come with SQLWindows 5.0 and the tool you use to create each one.

DataField QuickObject	 cQuickField
Multiline Field QuickObject	 cQuickMLField
Radio Button QuickObject	 cQuickRadio
Radio Group QuickObject	 cQuickRadioGroup
Checkbox QuickObject	 cQuickCheckBox
Listbox QuickObject	 cQuickListBox
Combobox QuickObject	 cQuickComboBox
Picture QuickObject	 cQuickPicture
Business Graphics QuickObject Visualizer that displays a graph from a data source.	 cQuickGraph

Commanders perform actions

The final type of QuickObject is the Commander. Commander QuickObjects direct user commands to the QuickObject data source. A QuickObject commander controls how the QuickObject data source accesses and modifies its associated data. The visual representation of a commander is generally a push button, but can also be a picture.

An application might contain push buttons for navigation through database records, such as *First* and *Last* push buttons. The icon used for the commander push button is predefined and dictated by the command that the commander represents. (For instance, the Discard command is predefined with the trash can icon.) This simplifies the development effort, since a single type of QuickObject commander can handle any and all commands for a particular QuickObject data source.

You can easily create your own QuickObjects

You can also create your own QuickObjects from existing ones. You create a new QuickObject class and have it inherit from an existing one. In this way, you can expand your choice of QuickObjects and create objects that have re-usable behavior tailored to your specific needs.

Any object created from a QuickObject can be customized via standard SQLWindows programming features. This enables you to use the power of a QuickObject and then customize it to meet your business criteria.

SQLWindows sample applications


SQLWindows Sample Applications

You can explore *Creating your own QuickObjects* in the SQLWindows Sample applications. The sample application shows how the SQLWindows QuickObjects framework is used to create QuickObjects from SQLWindows classes.

Exploring Object-Oriented Programming

SQLWindows provides a full implementation of an object-oriented programming (OOP) language. Object-oriented programs consist of independent software objects that have specific jobs and each object is responsible for managing how its job is done. Unlike traditionally structured programs, the various objects in OOP are autonomous. In general, to get data from an object, you send a message to the object or call some function defined in the object.

Additionally with OOP, you create code that is:

- **More reusable**. If you create classes that hide their implementation, you will be able to reuse the code in the same program and other programs.
- **More reliable and maintainable**. Code that is reused results in more reliable and maintainable programs because the code has already been tested. This also saves you time.
- **More extensible**. Your code is made more extensible by using the override behavior in derived classes and the late-binding features.

SQLWindows implements these concepts of OOP:

- Objects
- Classes
- Inheritance
- Late-binding (polymorphism)

Let's talk about them.

Objects bring data and behavior together

An object is data and behavior bundled together. The data part of an object is like a record or structure in other languages. The behavior of an object takes the form of functions that access the object's data.

In OOP, you write code in such a way that you do not rely on the implementation of an object. This is called *information hiding*. The goal in information hiding is to make an object as independent from another object as possible.

Classes group objects by type

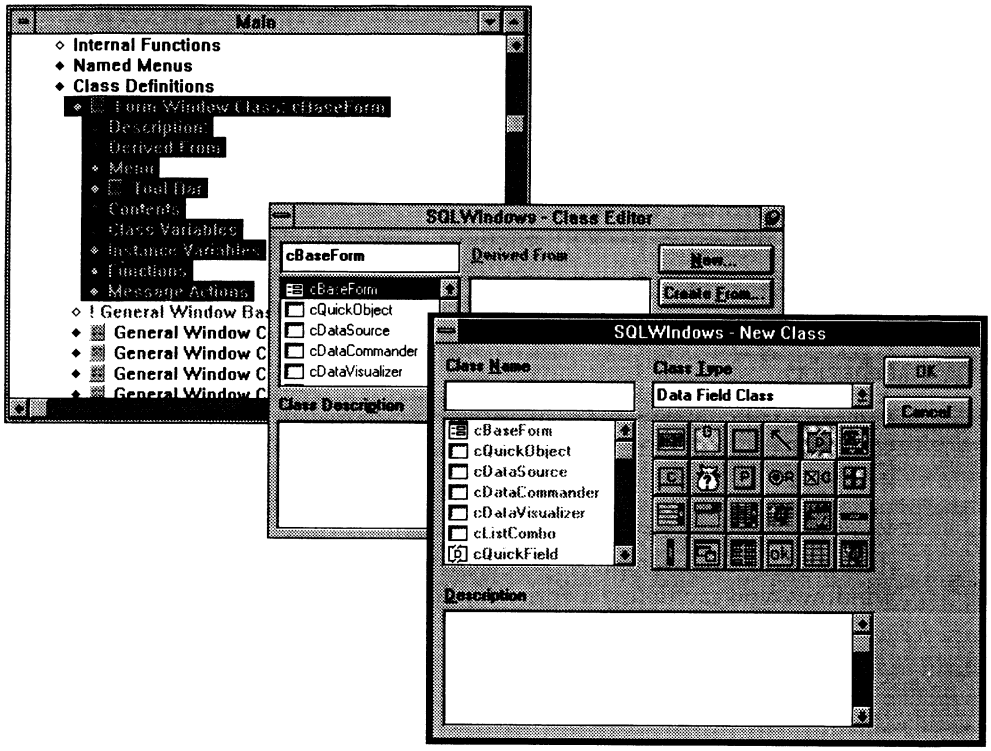
To use the full power of OOP, you must work with *classes*. A class is the OOP term for type. In procedural languages, each variable has a type; similarly, in OOP, each object is a member of a class. The OOP classes are templates or blueprints used to create similar objects with the same properties (data and behavior).

You can easily define your own classes that represent visual objects. SQLWindows also provides you with QuickObjects (discussed in this chapter) that have been built using the SQLWindows OOP features.

You define new classes using the SQLWindows **Class Editor** and **New Class** dialog boxes. For example, if you want to have a number of similar form windows, you can define a Form Window class as follows:

```
Form Window Class: MyPurchaseOrderFormClass
```

The following example shows **Class Editor** and **New Class** dialog boxes open for defining a new class in the Main program outline. You access the **Class Editor** dialog box by selecting the **Class Editor** option in the **Tools** menu.

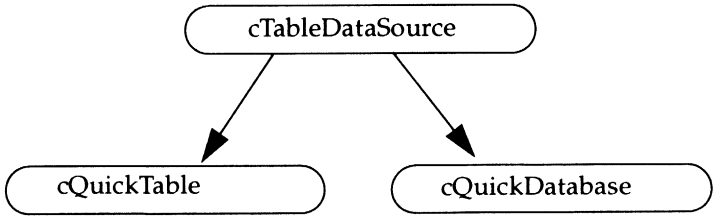


Your class definition (dependent on the type of class or object) can include the following sections: Description, Derived From, Menu, Tool Bar, Class Variables, Instance Variables, Functions, and Message Actions.

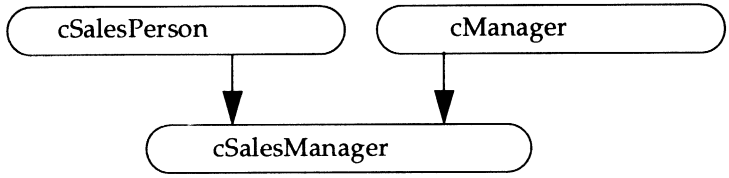
Inheritance makes building new classes easy

Inheritance is the ability to define a class in terms of another class. We say the new class is derived from or inherits from the older (base) class. The derived class automatically includes all the data and functions of the base class.

For example, if cQuickTable and cQuickDatabase are two classes derived from the base class cTableDataSource, then both cQuickTable and cQuickDatabase inherit all the features of cTableDataSource.



SQLWindows also supports *multiple inheritance*, which allows you to derive a class from more than one class. For example, in your application you might have one class, called cSalesPerson, for the sales people and another class, called cManager, for the managers in a company. You can derive a third class for sales managers called cSalesManager. This derived class inherits the properties of both cSalesPerson and cManager.



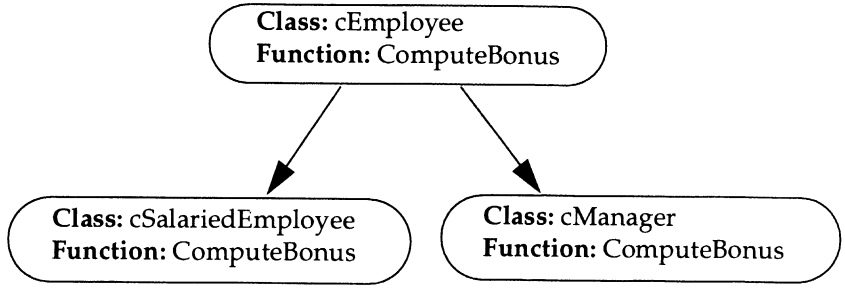
Late-binding (polymorphism) makes your programs flexible

With OOP, you can call a member function using a variable that can, at various times in the execution of the program, refer to different objects. At run time, the system determines the class of the object and thus which member function to call. This feature is called *late-binding* or *polymorphism*.

For example:

```
call hWnd.cEmployee..ComputeBonus()
```

In this example, the cSalariedEmployee and cManager classes are derived from the cEmployee base class and all three classes have the function ComputeBonus.



At different times in the program execution, `hWnd` might refer to either a cSalariedEmployee object or a cManager object and the correct ComputeBonus is called.

The pseudo code for the above example, using traditional programming, might look like this:

```
if a manager
   call MgrComputeBonus()
else if a salariedemployee
   call SalaryComputeBonus()
...
...
```

One of the advantages of using late-binding is extensible code since you can add new classes without needing to change many areas of code. In the above OOP example, you can derive additional classes from the class cEmployee and the call to `hWnd.cEmployee..ComputeBonus` will automatically handle the new classes.

OOP - a better approach

Using OOP, you create a paradigm of autonomous objects interacting versus the procedural structures in traditional programming. The OOP model uses the same concepts during the analysis and solution stages rather than synthesizing a design for the business problem resulting in various data structures and procedures. For example, with OOP, if you need to discuss a Purchase Order as a part of the business problem, you can have a Purchase Order object in your design and work with Purchase Order objects.

Team Programming with SQLWindows

The team programming feature of the SQLWindows Corporate Edition brings increased efficiency to your entire team. Team programming makes it easy to manage teams of various sizes, share code, and standardize programs. When you're ready for enterprise-wide development, obtain the SQLWindows Corporate Edition.

Teamwork gets the job done

When client/server projects grow beyond the pilot stage, you need to apply the power of teamwork. Team programming provides your team with integrated tools to manage multiple projects, allocate programming skills, and work simultaneously on many parts of a complex application. Also, team programming lets you copy similar code *quickly and easily* and then modify it.

Team programming combines the security and power of mainframe-class software with the flexibility of today's graphical tools to make your development team more productive.

With team programming you can:

- Manage multiple projects easily.

  Today's Information Service organizations often juggle many projects to keep up with application demand. You can maintain full control over resource allocation and tracking by assigning team members project stages, tasks, and access privileges with team programming. Additionally, team programming lets managers monitor the development process.

- Plan for changes to your application.

  With team programming's impact analysis you will see the potential effects of changes to applications or the database. Team programming tracks all changes and warns you if they affect other screens in the application—or even other applications.

- Protect your code while you make changes.

  SQLWindows and its team programming component support every step of the client/server development process, making team programming a reality for your development group.

  Team programming and the Gupta Open Repository provide project management and integrated version control previously available only in minicomputer and mainframe-based development systems.

Check-in/check-out facilities assure you that only one programmer at a time can modify a piece of code, and team programming tracks all check-in/check-out activities. SQLWindows supports version control through an easy-to-use interface to the popular PVCS program.

Gupta Open Repository provides multi-user storage

At the heart of team programming is the Gupta Open Repository—a centralized, multiuser storage area for application source code and other development information.

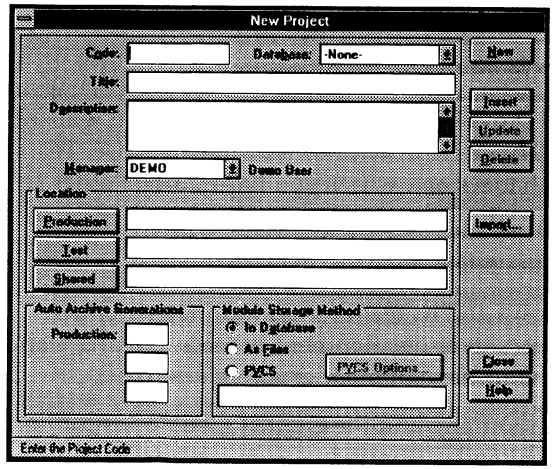
The Gupta Open Repository contains an extended storage area of information about the application database and holds project-related information and files, including a current copy of each project module. A project module is a project-related file such as a SQLWindows application, a Microsoft Word document, an Excel spreadsheet, or a Paintbrush graphic. The Repository contains information about the contents of the screens within each SQLWindows application.

Using SQLWindows team programming features

On the following pages are examples of how you might use some of the features in team programming.

Creating a new project

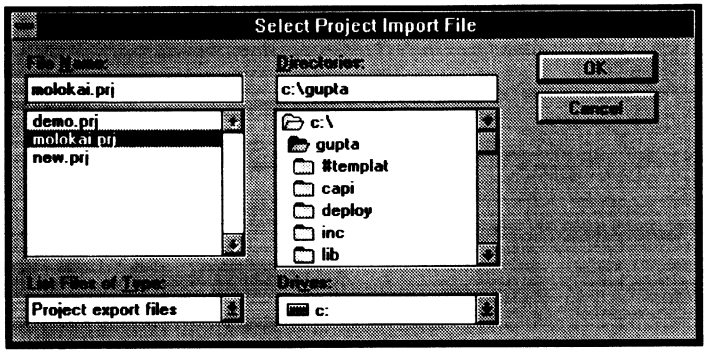
You can create a new project to manage applications. Either define it, or import a copy of an already existing project, then modify it. You can specify where to store your new project and how many archive copies to create.



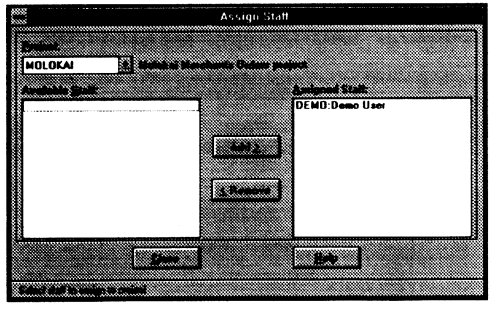
You can choose to store the project modules in one of three ways:

In Database	As long varchar entries in the Repository database.
As Files	As special files in your Workspace path.
PVCS	As PVCS archives in the Project path; in certain circumstances, this allows high compression.

Once you specify **New Project** information, you can **Import** files into the project using the **Select Project Import File** window. From this window you can choose whatever file you want to copy and use for your new project.



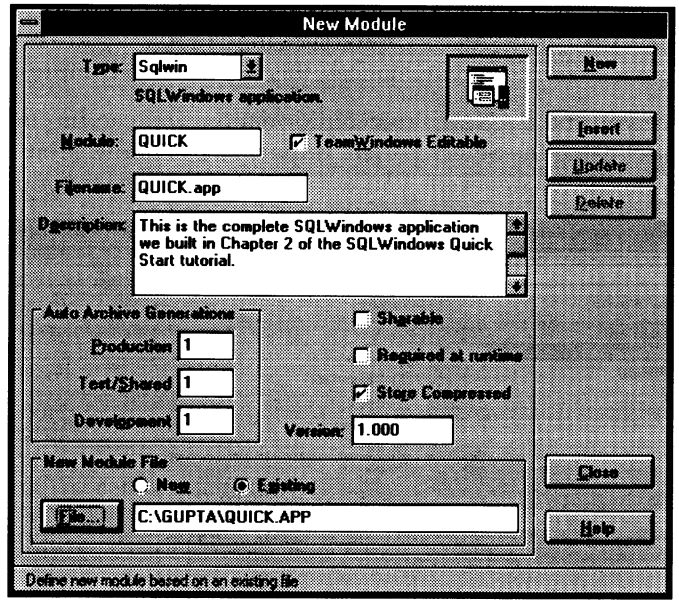
You can add team members to a project using the **Assign Staff** dialog box.



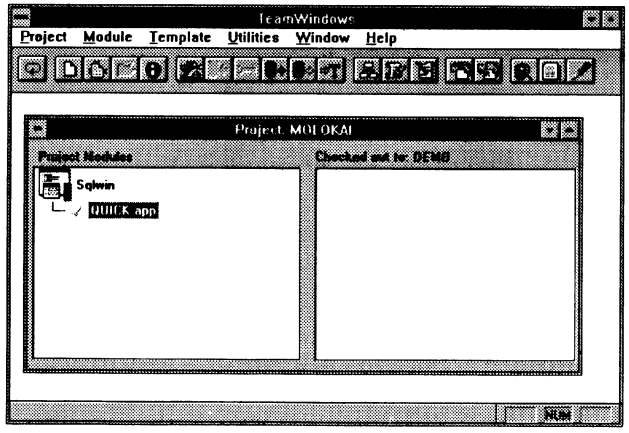
Adding an application to a project

Adding an existing application to a project is very easy using the **New Module** dialog box. This automatically associates the new module with your

project. Just fill in the information about the new module and the existing application.

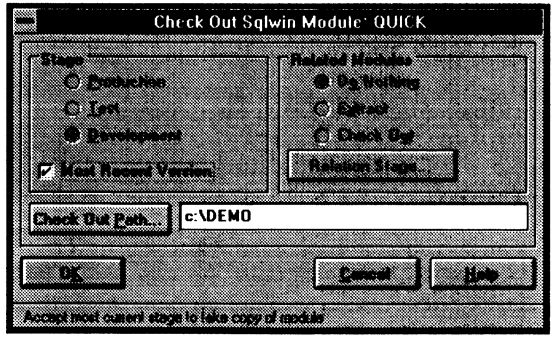


The application is associated with your project in the Repository:



To make changes to the application, you need to check out the module to your workstation. This is easy to do. In the project window, shown above, just drag an application module from the left hand side of the folder to the **Checked out** box on the right hand side.

After you check out the file, you should see the **Check Out Module** dialog box:

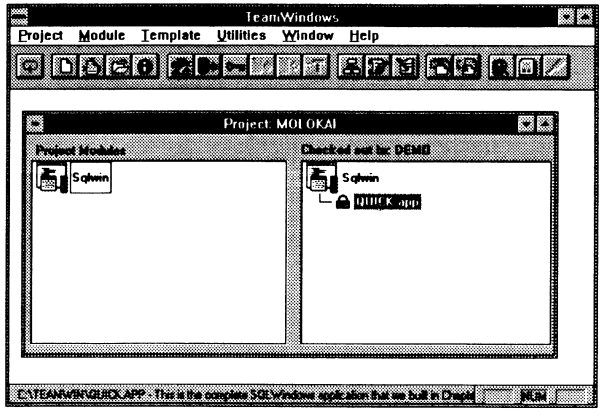


The Repository is capable of tracking related modules. For example, if a report file and two bitmap files are related to **QUICK.APP**, this relationship is tracked. This dependency management feature helps you stay in control of large client/server projects.

Once you check a file out, other team members have read-only access to it — they cannot modify it. All changes you make to the application are protected until you check it back in to the Repository. If necessary, you can always retrieve an archived copy of your application.

The Repository also lets you track non-Gupta modules. For example, you could store a project schedule with Microsoft Project in the Repository and access it directly with a single keystroke. This feature of the Repository makes it much easier for you to control all of the information associated with a particular project. It also helps you monitor the progress of your project.

After you have checked out the module, your window might look like this:

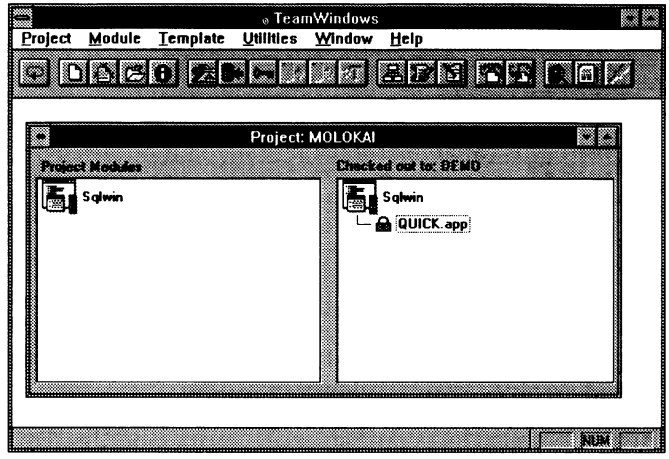


While the module is checked out, you can edit and integrate it with other files or modules, then you can run an impact analysis on your changes.

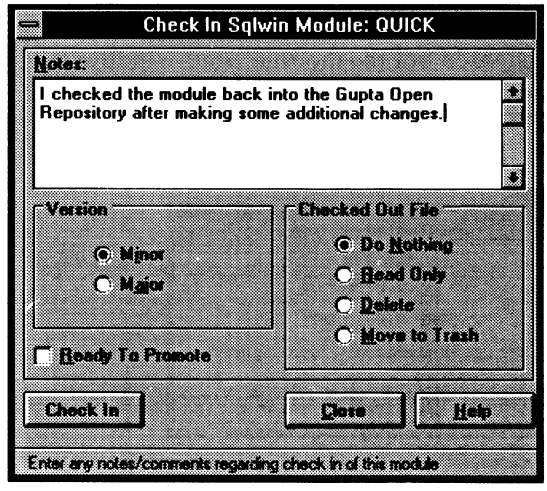
To edit a module using SQLWindows, press **F6** or click **SQLWindows** on the **Module** menu. This launches the module in SQLWindows.

Checking modules back into the Repository

To return the application to the Repository, just drag it back to the left hand side of the project folder.



After you have checked in the application, the **Check In Module** dialog box helps you record it:



You can enter some notes that indicate what changes you made to this module, if any. Also, you can specify whether the changes you made qualify as a minor or major version change. The Repository automatically increments the version number based on whether you select Minor or Major. With team programming, version control is accurate and easy to maintain.

Once the module has been checked in to the Repository, you may check it out or copy it as many times as you like. You can delete the left over local copy right away or you can move it to a trash directory for deletion later. Other team members can also check it out and use it.

Chapter 4

Where to?

Where are all the answers? We've tried to anticipate your questions as well as your business requirements. In this chapter, we answer:

- **Where do I go from here?** SQLWindows includes the features and functionality you're looking for. We'll help you decide which package is the perfect one for you.
- **Where can I learn more about specific features and functions?** We provide a complete set of documentation and online help that can answer your detailed questions and demonstrate the features you want to use.
- **How can I complete my evaluation?** We've put together an evaluation checklist you can use to make sure you're getting all the features you need and more. Compare SQLWindows to any other product!
- **Can I get help installing?** If you're having trouble installing the software, we'll tell you how to get help over the phone.
- **Where can I get help, or discuss my needs, ideas, and plans with other users?** We'll show you how to join the GUPTA Forum on CompuServe and plug into the information highway with Gupta.

Where do I go from here?

Now that you've driven SQLWindows Solo, where do you go from here? For those of you who want to create single-user desktop applications for desktop deployment, you can keep going with SQLWindows Solo. It includes a free 5MB SQLBase database engine that you may deploy either with your SQLWindows Solo applications, or with applications you created using other SQLWindows editions. SQLWindows Solo provides for unlimited deployment of your applications against desktop databases such as dBase, Paradox, and Clipper with ODBC connectivity.

When you move to creating fully networked applications that need to run against an unlimited size single-user SQLBase database, networked SQLBase databases, or other popular networked relational databases via ODBC connectivity, choose the SQLWindows Starter Edition as your next step. SQLWindows Starter Edition includes Lotus Notes integration, with point, click, drag and drop programming to add Lotus Notes functionality to your application.

When it's time to develop and deploy applications across the workgroup, you will want the SQLWindows Network Edition. SQLWindows Network Edition provides high-performance native connectivity routers for most corporate databases, including Oracle, SQLServer, Informix, Ingres, and AS/400.

When you and your development peers require an all-out team development and enterprise deployment environment, it's time to move to the SQLWindows Corporate Edition. The powerful tools and features of the corporate edition include team programming facilities, a repository database for managing application modules and version control, high-performance runtime execution with the exclusive SQLWindows 4GL-to-C compiler, database and network performance tuning tools, state-of-the-art graphical database administration tools, and an interface to CASE tools via Gupta TIE (Tools Integration for the Enterprise).

The following table shows you what's included with each package.

	Corporate	*Network*	*Starter*	*SQLWindows Solo*
SQLWindows with Quick Objects	✓	✓	✓	✓
Quest data management tool	✓	✓	✓	✓
SQLBase development engine *	✓	✓	✓	✓
Free deployment of 5MB SQLBase engine				✓
OBDC connectivity to personal databases	✓	✓	✓	✓
QuickObjects for e-mail systems	✓	✓	✓	✓
QuickObjects for Lotus Notes	✓	✓	✓	
Visual Toolchest class library	✓	✓	✓	
ODBC connectivity to SQL databases	✓	✓	✓	
SQLRouters for Oracle, SQLServer, Informix, Ingres, and AS/400 - for **development**	✓	✓	✓	
SQLRouters for Oracle, SQLServer, Informix, Ingres, and AS/400 - for **deployment**	✓	✓		
SQLWindows Compiler	✓			
SQL Console application tuning facility	✓			
TeamWindows, Gupta Open Repository	✓			
Interface to CASE tools via Gupta TIE	✓			

* SQLBase engine in SQLWindows Solo Edition stores up to 5 MB. You can deploy this engine with any SQLWindows application.

Where can I get answers?

Want to know more about everything? Press **F1** to bring up online help. With SQLWindows **Solo**, these books and online documentation are ready to help.

IF YOU WANT TO ...	READ
Start quickly building SQLWindows applications.	SQLWindows Solo Quick Start & online help
Develop advanced SQLWindows applications.	*Power Programming with SQLWindows*

In addition to the documentation and help provided with the SQLWindows Solo, SQLWindows **Network** and **Starter** Editions come with these books and online documentation.

IF YOU WANT TO ...	READ
Start quickly building your database and accessing information.	*Quest Quick Start* & online help
Build powerful forms, queries, and reports.	*Quest User's Guide*
Refer to descriptions of SQLWindows features.	*SQLWindows Developer's Reference*
Look up syntax, descriptions, and examples of SQLWindows functions.	*SQLWindows Function Reference* & online help
Generate reports from SQLWindows applications.	*ReportWindows User's Guide* & online help
Take advantage of a powerful library of reusable objects in your applications.	*Visual Toolchest Class Library* & online help
Create applications to read, update, and display information contained in Lotus Notes databases.	QuickObjects for Lotus Notes & online help
Enter SQL commands in a Windows environment and perform database management functions.	*SQLBase SQL Reference*
Manage your SQLBase database, and design databases and applications.	*SQLBase Database Administrator's Guide*
Learn about Gupta connectivity information and trouble-shooting.	*SQLNetwork Guide to Gupta Connectivity*

IF YOU WANT TO ...	READ
Take advantage of our technical services.	 *Technical Services* booklet

The SQLWindows **Corporate** Edition comes with this additional documentation and online help.

IF YOU WANT TO ...	READ
Manage projects, maintain source code, and develop in a multi-user development environment.	  *SQLWindows Team Programming User's Guide* and online help
Improve runtime performance and resource usage of SQLWindows applications.	 Compiler online help
Manage database tasks with an easy-to-use graphical interface.	 *SQLConsole In This Package*

The Gupta Forum on CompuServe is a place where you can get answers.

IF YOU ARE A COMPUSERVE SUBSCRIBER...		TYPE
And you want to learn about Gupta products and communicate with other Gupta users.		GO GUPTA (See the section later in this chapter on the GUPTA Forum on CompuServe.)

In the United States, Fast Facts provides additional information.

IF YOU WANT TO ...		CALL
Get some of the latest information on SQLWindows.	 FAX	Gupta Corporation's Fast Facts. Call **415/617-4600** to access a catalog of available documents. (Only in the United States and Canada.)

How can I complete my evaluation?

Congratulations on having built your first application with SQLWindows Solo and QuickObjects! You have just entered the promising world of object-oriented client/server computing. As you discovered, SQLWindows is easy to learn and fun to use. But don't stop now... the power of SQLWindows will keep you going. Browse the on-line documentation to learn more!

In case you need more information, we have listed below over 100 important product requirements, all met by SQLWindows. SQLWindows is available in many editions... the full power of all those editions is described below. Just in case you plan to compare SQLWindows with other products, walk through this list carefully... many vendors show you glitzy demos but fail to provide the power to get the job done!

As you have discovered in using the SQLWindows Solo, and reviewing the accompanying on-line documentation, SQLWindows offers easy to learn features to help you get started quickly and plenty of power to help you get the job done.

Ease of Use	***SQLWindows***		
Quickly build applications without coding	✓		
Build applications for all database tasks -- Query, Update, Insert, Delete -- without coding	✓		
Build master-detail links without coding	✓		
Build multi-screen applications without coding	✓		
Create menus without coding	✓		
Create graphs without coding	✓		
Create reports without coding	✓		
Email-enable an application without coding	✓		
Integrate queries and reports built with end-user tools	✓		
Integrated, context-sensitive on-line help	✓		

Coding Environment	***SQLWindows***		
Tight integration between design and coding environments	✓		
View code in your application at any level of detail	✓		
Coding assistant to help you write code	✓		
Manage code associated with a single graphical object or all code in your application at once	✓		
Easily print out all code associated with your application to review off-line	✓		
Integrate with C, C++, COBOL code	✓		
Partition application logic across client and server	✓		
Translate applications into international languages	✓		

Object Oriented Programming	***SQLWindows***		
Ready-to-use objects that encapsulate basic functionality for quickly getting started with OOP	✓		
Take standard system objects and fully extend or customize them to suit your business needs	✓		
Programmer-defined business objects that can be customized when instantiated by other developers	✓		
Convert existing non-object-oriented objects into true object-oriented abstract classes	✓		
Support for graphical and non-graphical, i.e. functional, objects and classes	✓		
Single inheritance to reuse classes with modifications	✓		

Object Oriented Programming	SQLWindows		
Multiple inheritance to easily combine graphical and non-graphical classes for flexible reuse	✓		
Class editor or browser to create and review classes	✓		
Consistent access mechanism for standard classes and user-defined classes	✓		
Create hidden classes which other developers can not access	✓		
Support for late binding and polymorphism	✓		
Integrated class library with ready to use objects	✓		
Third-party class libraries to speed up development	✓		

Compilation & Debugging	SQLWindows		
Global compiler to compile and cross-reference entire application	✓		
Incremental compiler to take less time to compile during subsequent compilations	✓		
Animated debugger to watch the execution path of your application to help understand application behavior	✓		
Debug entire application without having to load debugger with code fragments one step at a time	✓		
Set break points anywhere in code	✓		
Use dynamic breakpoints for flexibility	✓		
Evaluate variables or expressions	✓		
Patch-in code on the fly during debugging	✓		
Watch message passing among objects	✓		
Monitor stack usage	✓		

High Performance	***SQLWindows***		
Convert 4GL code to fast C code	✓		
Tuning facility to help monitor and optimize client, server and network performance in integrated manner	✓		
Early binding to speed run-time performance	✓		
User-controlled memory caching for GUI speed	✓		
Background fill of data windows for fast response time	✓		

Team Programming	***SQLWindows***		
Native version control through integrated repository	✓		
External version control through Intersolv PVCS	✓		
Use of valuable PVCS functionality like difference analysis and version labeling	✓		
Facilities to track inter-dependent modules	✓		
Ability to archive older versions	✓		
Single integrated repository to manage data definitions, source code and all other information	✓		
Extensible repository for accommodating all module types, including bitmaps, icons, documentation etc.	✓		
Easily access non-Gupta modules like Microsoft Project schedules or Microsoft Word requirements specifications	✓		
Project management for multiple users across life cycle	✓		
Project reporting and systems documentation	✓		

Team Programming	SQLWindows		
Impact analysis to predict impact of database changes on application code to simplify application maintenance	✓		
Coding standards to promote consistency in coding	✓		
Security using username and password control	✓		
Privilege levels to control which users access which objects resulting in fewer conflicts and more security	✓		
Audit trail and check in/ check out log facilities	✓		
On-line team communication facilities	✓		
Templates to promote reusable object frameworks	✓		

Graphing and Reporting	SQLWindows		
Display wide variety of graph types- line, bar, pie, etc.	✓		
Customize graph with labels, legends, etc.	✓		
3-D rotation capability for attractive data presentation	✓		
Change graph type at run-time to get different views	✓		
Customize graph with labels, colors, etc. at run-time	✓		
Quick default reports based on query data	✓		
Banded report designer for full control over reports	✓		
Include images in reports	✓		
Two-pass reports	✓		
Mailing label reports	✓		

Graphing and Reporting	***SQLWindows***		
Form letter reports	✓		
Matrix and cross-tab reports	✓		
Reusable report templates	✓		
Link report to multiple data sets	✓		
Formula editor to build sophisticated reports	✓		

Database Access	***SQLWindows***		
Access popular databases through high-performance custom connectivity	✓		
Access databases through ODBC	✓		
Access more than one database simultaneously from same application	✓		
Take full advantage of back-end functionality like array binding or deferred parsing	✓		
Multiple SQL isolation levels to handle all database locking models, including row, page, table level locks	✓		
Support of dynamic SQL for maximum flexibility	✓		
Fully support stored procedures. Retrieve both single result value and multiple result set rows	✓		
Maintain bi-directional scrollable cursors and front-end result sets to minimize re-querying and network traffic	✓		
Cursor context preservation across commits	✓		
Support of named cursors for greater application control	✓		

Integration with Microsoft Windows	SQLWindows		
DDE support	✓		
OLE support	✓		
Embed any OLE object in applications	✓		
MDI support	✓		

Open Integration with Other Systems	SQLWindows		
Comprehensive strategy to address open integration	✓		
External version control tools	✓		
CASE tools	✓		
Testing tools	✓		
Email systems	✓		
Lotus Notes	✓		
TP Monitors	✓		

Installation Help

In the United States and Canada, SQLWindows is packaged with thirty days of free installation assistance. If you have difficulty installing the software, call us at 1-800-704-8782 and our Technical Service engineers will assist you. Outside the U.S. and Canada arrangements vary by country. Please contact your local Gupta sales or support office for more information.

If you encounter non-installation problems or questions, you can visit Gupta's CompuServe Forum for assistance.

Outside the United States and Canada, use these numbers:

Gupta Asia	323 0178
	+65 323 0178 (from outside Singapore)
	323 0195 (FAX)
Gupta France	(1) 46 94 99 66 (within France)
	+ 33 1 46 94 99 66 (from outside France)
	(1) 46 20 04 30 (FAX)
Gupta Germany	089 74 81 21 45 (within Germany)
	+ 49 89 74 81 21 45 (from outside Germany)
	089 78 49 17 9 (FAX)
Gupta Northern Europe	3465 52100
	+31 3465 52100(from outside Holland)
	3465 67680 (FAX)
Gupta UK	0628 478 400(within U.K.)
	+ 44 628 478 400(from outside U.K.)
	0628 481 076 (FAX)
Gupta US	(415) 321 4484
	+1 415 321 4484 (from outside USA)
	(415) 617 4680 FAX

The Gupta Forum on CompuServe

The Gupta Forum on CompuServe is the ideal place to get assistance while you explore Gupta products. In the Forum you can join an active Gupta Developer community dedicated to sharing their experience with Gupta products and to making people like you successful with them. Many developers spend time in the Gupta Forum daily. Even though they have access to traditional phone-based technical support, they choose to come to the Gupta Forum first with their questions because they can quickly get access to a large developer community facing similar challenges and a large body of already-discovered knowledge. In addition, Gupta staff members are on-line daily facilitating the exchange of knowledge and the smooth operation of the Forum.

What you can expect

The Gupta Forum's strength lies in its membership. Gupta Forum members are on the cutting edge of Gupta implementations worldwide. They know what it takes to "get client-server done" with Gupta products. Gupta Forum members also know that by actively participating in the Gupta Forum, they advance the state of the art and remain close to emerging trends.

The Gupta Forum's TeamAssist volunteer organization looks over the message areas daily and contributes valuable TeamTip documents and TEAMTOOL utilities for all.

Come join our active and constructive members. Start a message "thread" by creating a forum message. Or, scan the topics being discussed in each section and contribute to a thread in progress by replying to an existing message.

How to get the most from our forum

Once you know how to create and reply to messages, you can get help on any of the forum's other services, including file downloading and live conferencing. The brief examples that follow should have you joining the Gupta Forum, posting and checking for messages, and even downloading files in no time.

Free Software and Account Setup

To ensure that your initial cruise into the Gupta Forum is a smooth one, SQLWindows Solo includes a free copy of CompuServe's access software, WinCIM. We have also arranged for CompuServe to waive the account sign-up fees and apply a $15.00 credit towards your travels on CompuServe. The WinCIM software includes an on-line account signup utility.

As part of this chapter we explain how to install the WinCIM software and sign up for your new account. Please be aware that most account sign-ups go through two phases: temporary password and permanent password.

Tip: While using your temporary password, you'll only be able to read and download from the Gupta Forum. You won't be able to ask questions or contribute files until you have your permanent password.

Credit card sign-ups are the fastest and take about one week or so to go from temporary to permanent. Consider signing up before you have a question. We look forward to seeing you up on the Gupta Forum!

Setting up WINCIM

The following has been adapted from CompuServe's Quick Start Guide for the CompuServe Information Manager for Windows (WINCIM).

Before you start to set up WinCIM, be sure to read the Service Agreement Terms (see inside back cover of *Membership Guide for Windows*).

Tip: If you encounter problems setting up WinCIM, call CompuServe Customer Service at one of the phone numbers listed inside the Membership Guide.

1. Insert the **WinCIM Installation** diskette into your floppy disk drive.
2. Start up Microsoft Windows.
3. Choose **Run** from the Windows Program Manager File menu.
4. Type: **A:SETUP** and then click on the **OK** button.

   This assumes your floppy disk drive is called Drive A. If it has a different designation, use that instead.

The Setup program takes over and prompts you to identify the drive and directory where you want to install WinCIM. All related files will be placed into subdirectories of the directory you specify.

5. To use the default of C:\CSERVE, click OK. To use a different drive and/or directory, type them in and click OK.

   We strongly recommend using the default directory. C:\CSERVE is the 'common directory' for CompuServe telecommunications software.

The Setup program prompts you to indicate whether or not to copy the Signup files.

6. If you are not yet a CompuServe member, choose **Copy the Signup Files.** If you are a member, choose **Do Not Copy the Signup Files.**

   You are a CompuServe member only after you have signed up, connected to CompuServe, and received a new Password and a Permanent User ID number.

The Setup program copies and decompresses the WinCIM files, displaying notices that indicate its progress. When the setup process has been successfully completed, you will see the CompuServe Program Group window.

7. If you chose **Copy the Signup Files** in Step 6, you will be asked if you want to sign up for CompuServe membership now. To sign up, click the Yes button, and then proceed to the section below titled *Signing Up for CompuServe Membership.*

   If you do not wish to sign up now, you can do so later by double-clicking the Signup icon in the CompuServe Program Group window.

   - If you used WinCIM's Signup software to sign up for CompuServe membership, your Session Settings are automatically filled out for you.
   - If your C:\CSERVE directory already contained DOSCIM (the CompuServe Information Manager for MS-DOS), WinCIM will use the DOSCIM Session Settings. Therefore, you do not need to fill them out.

8. To start up WinCIM, double-click the WinCIM icon. If you have your permanent User ID number and the correct CompuServe password, you can connect to CompuServe and start using the services.

9. If you were already a CompuServe member before you obtained the WinCIM software, and you did not have DOSCIM in C:\CSERVE, you must fill out the Session Settings dialog (see *Session Settings* below) to provide WinCIM with the information needed to communicate with CompuServe.

DOSCIM Compatibility

You can use WinCIM with Version 2.1 (or greater) of DOSCIM. The two CIMs can share the 'common' C:\CSERVE directory, and both will use the same:

- Session Settings
- Modem Settings

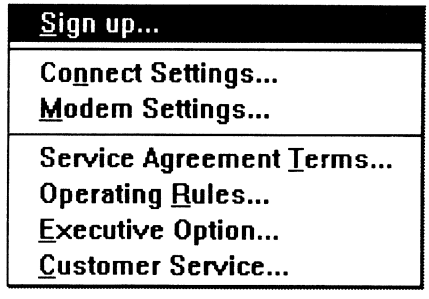
- Filing Cabinet documents
- In-Basket messages
- Out-Basket messages
- Address Book
- Favorite Places list
- Quotes ticker symbols

Signing Up for CompuServe Membership

Unless you are already a CompuServe member, you must sign up for membership before you can begin using CompuServe.

You are a CompuServe member only after you have signed up, connected to CompuServe, and received a new Password and a Permanent User ID number.

In Steps 6 and 7 of the setup procedure (see *Setting Up WinCIM* above) you copied the WinCIM Signup files and started the Signup software. As a result, you see the signup menu.



When you're ready to proceed, choose **Sign up**. The Signup software takes you through a series of dialogs where you supply all the information CompuServe needs before you can become a member. Which dialogs you see depend on what country you're in and what billing method you select.

You will need the Agreement Number and Serial Number which accompany the software. Depending on your billing method, you may also need:

- Credit card account number and expiration date
- Checking account information for direct debit (only available in the United States and some European countries)
- Business account bank and trade references

If you are signing up outside the United States, you will have to supply a telephone access number, either by typing it in or by selecting it from a list of phone numbers for your country.

Once the signup software has all the required information, it will call CompuServe to register you as a member. Depending on your billing method, you will most likely receive your permanent User ID number and a temporary password.

Be sure to write down the User ID number and password when you receive them. You are now a CompuServe member.

Exit the Signup program.

To start using WinCIM, double-click the WinCIM icon in your CompuServe Program Group window.

You can now connect to CompuServe and start using the services. However, there will be some restrictions on your membership until CompuServe mails you your permanent password (usually within two weeks).

Long-Distance Telephone Access (North American Members Only)

In the United States and Canada, if there is no local telephone access number for CompuServe, the Signup software will fill in a long-distance number. However, you may be able to find a more cost-effective phone number online, by choosing **Go** from the Services menu and then filling in PHONES as the service name.

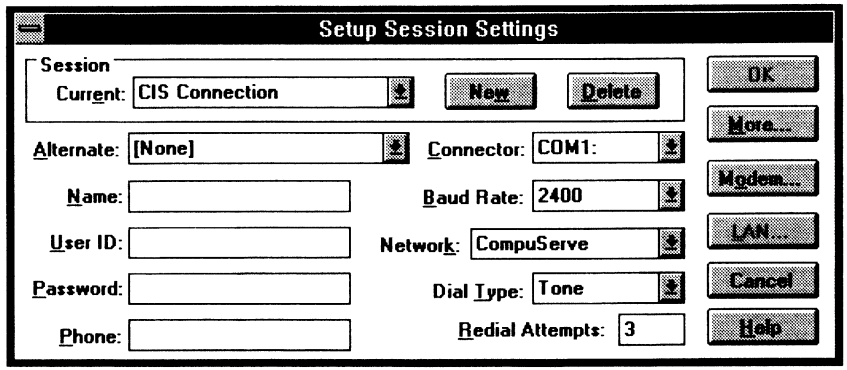
WATS Line

One way to access CompuServe without a local telephone number is through the CompuServe U.S. WATS line (800 access number). You can use the WATS line as soon as you receive your permanent CompuServe password.

Session Settings

If you signed up for CompuServe membership when you installed WinCIM, or if you installed WinCIM into a common directory that already contained any other version of CIM, WinCIM already has the information needed to connect your computer to CompuServe. Otherwise, you must provide this information by choosing **Session Settings** from the initial desktop's Special

pull-down menu. (You can also use this command to change the information any time you like.) This takes you to the Setup Session Settings dialog:



The Session: box is for selecting, creating and deleting sets of session settings. You can create as many sets as you need, for different phone numbers when you travel with your computer, different people using the same computer, and so on. To create a set, click New. Once you have created a set of session settings, you can select it from the Current: pop-up menu, and then use it, edit it or click Delete to delete it.

Every available set of session settings will be listed in the pop-up menu for the Current: textbox. In our example, there is as yet no session settings to list. There must be at least one set, or WinCIM won't be able to connect you to CompuServe.

To create your first set of session settings, fill in the blank boxes and edit the default information as needed, and then click the OK button. WinCIM automatically assigns the name "CIS Connection" to the first set of session settings you specify.

The Alternate: textbooks identifies a set of session settings that WinCIM will use if it is unable to establish a connection with the Current settings. The Alternate: pop-up menu lists all the same sets of settings as the Current: pop-up menu, plus [None], which means not to try any other settings if the Current settings fail.

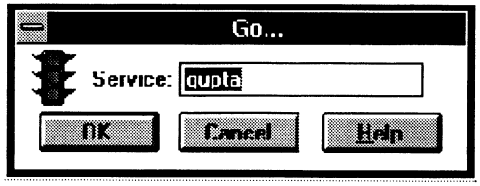
Once you select Alternate Settings, they will remain linked to the Current settings until you select a different set. You can edit either set without affecting the linking.

For additional information about Session Settings, press F1 for online help.

Joining and Using the Forum

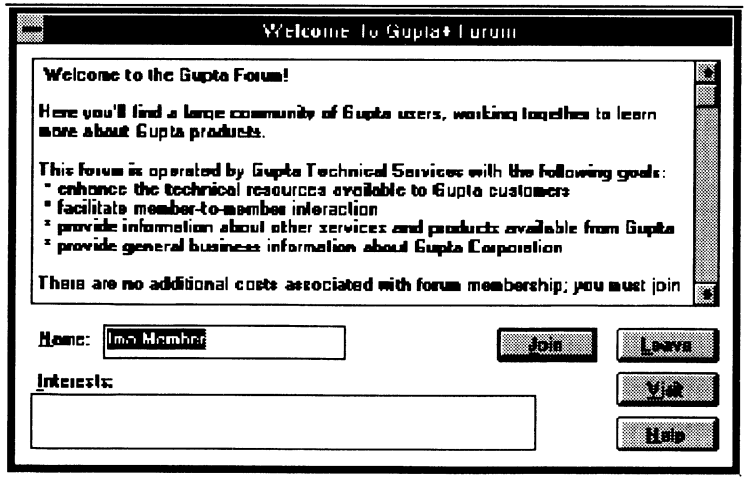
Choose **GO** from the services menu and specify **GUPTA** as the service name to go to.

We assume you have successfully obtained an account and password and are able to go on-line by checking the Basic Services icon on the Services window. If you need any help with account setup or getting on-line for the first time, please see page 3 of Compuserve's *Membership Guide for Windows* for phone numbers to call for assistance.



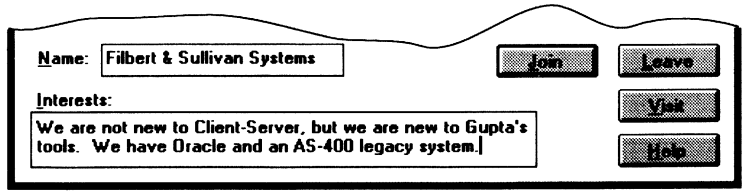
Tip: There is no charge for "Joining" a forum beyond your usual connect-time charges. As a "visitor" you won't have access to much of the forum, but still pay connect-time charges, so be sure to join.

The first time you GO to the Gupta Forum, you'll be asked if you want to join.



☞ 1. The **Name** field is automatically filled in with the **Name** field in the Session Settings entry you used to call CompuServe. You are welcome to change the name at this time, but please choose either your full **Firstname Lastname** or **Company Name**. To keep the name as it is, use **Tab**, your

mouse or **Alt-I** to move to the Interests box. Below we've changed Ima Member to Filbert and Sullivan Systems.



2. The Interests box is your chance to tell other members about your interests and what sort of experience you would like to share. This is optional information, but we encourage you to include at least a few words here. In the example above, we mention our background and the types of databases (Oracle and AS-400) we'll be working with.

3. Click on **Join** (or just press **Enter**).

   Congratulations! You are now a member of the Gupta Forum on CompuServe. Turn to pages 16 and 17 of CompuServe's *Membership Guide for Windows* for a handy reference as you explore the forum and read the sections below.

Changing your name or member interests

To change your member interests any time after you have joined, choose **GO** and type **GUPTA**. Once inside the Gupta Forum, pull down the **Special** menu and select **Change Member Interests**.

To change your Name any time after you have joined, send a message to *sysop* or **sysop* explaining who you "were" and who you would like to "be."

Asking a question: Message sections

To ask a question, simply post a message in the right product section, with a subject that clearly and specifically identifies the topic. We'll show you the detailed steps below.

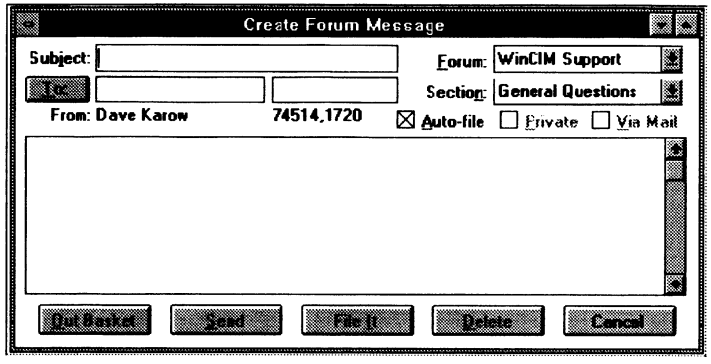
Should I be on-line or not? We recommend that you always create off-line, since it saves you money. Just as you would not call a distant fax machine before creating the document you plan to send, there is no reason to connect to CompuServe before composing your forum messages.

Tip: To create messages off-line, you must first go on-line and "join" the Gupta Forum so that WinCIM knows the forum and section names. See the section "Joining the Forum and creating a member entry" for more information.

Creating Forum Messages

While off-line or in Basic Services, pull down the **Mail** menu and select **Create Forum Message**. The same dialog appears when you are on-line in the Gupta Forum and pull down the **Messages** menu and select **Create Message**.

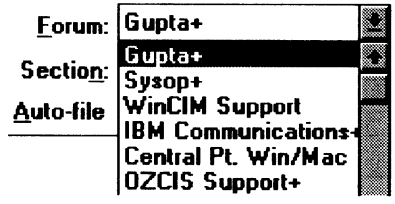
Below is the Create Forum Messages dialog as it might first appear:



Before you start typing your message. There are a five things you need to take care of before you write your message. Doing them first ensures that you'll post in the right forum and section.

1. Select the **Gupta+** forum.

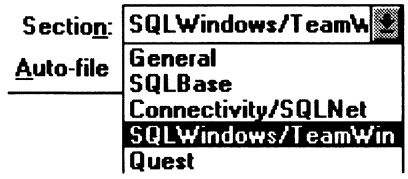
   If you are on-line, this is the default. Still, it's a good habit to make sure **Gupta+** is selected here.



2. Select the appropriate message Section.

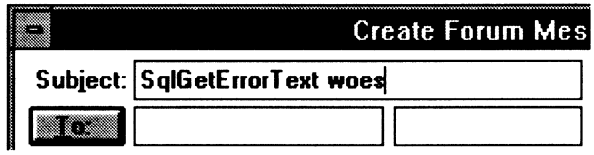
   Select the forum messaging *section* most closely related to your specific question (in this case, we'll say your question is about error handling in

SQLWindows and the section to choose is therefore SQLWindows/TeamWin).



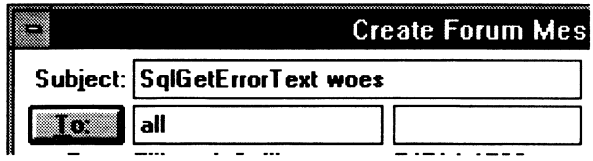
3. Choose an effective Subject.

   Choose your subject carefully. Be as specific as you can. You only have 24 characters to work with, so you need to be creative! Just remember, you want to attract other members who have the knowledge you seek. Here we're having trouble getting the output we want from the SqlGetErrorText function, so we add the word "woes" to the end of the Subject.



4. Choose the **To** address. (Usually **all**).

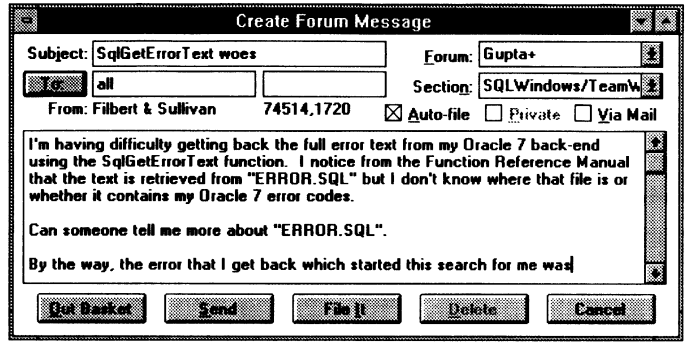
   All messages are public, and can be seen by any member who chooses to follow that discussion. The **To** field only controls who gets a "message waiting" indication until they read the message.



The convention is to post your message to "all" and leave the ID# field blank unless you know that a certain individual is a subject matter expert on the topic at hand and a regular forum contributor. Please do not address messages to "Tech Support" or the like. As you spend more time in the Gupta Forum and get to know who the active participants are in each section, you will develop a list of useful names and ID#'s to start threads to. The Gupta TeamAssist TopicRoster (HELP4U.ZIP) lists Gupta Forum TeamAssist volunteers who specialize in various topics. You can wait to read that file after

you have posted your first few messages, or you can jump ahead to the File Library section on page 8 to find out how to downloadHELP4U.ZIP from the General Library.

Writing the message text. You should keep your message focused to a single topic. If you have several questions, post several messages, each with the appropriate **Subject** and in the related **Section**. This increases the chance that each of the different questions evolve into a full discussion and will not get skipped over. Use a friendly, constructive tone whenever possible. Remember, you are asking your fellow software professionals to contribute to your success.

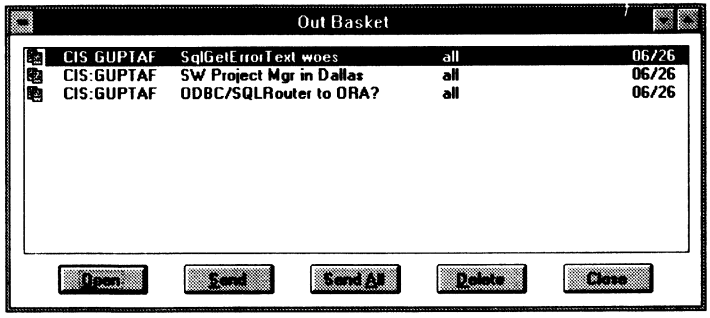


Now we have the message completed and we're ready to put it in our **Out Basket** or **Send** it.

Posting Your Message

By pressing **Out Basket**, you place the forum message in WinCIM's outgoing mail/forum messages folder. This is a handy way to prepare several messages off-line and then go on-line to post them all at once at lower cost. To post your messages, go into the forum and choose **Out Basket** from the **Messages** menu. This opens your **Out Basket**, where you can highlight the message(s) and then

click **Send** or **Send All**. You must be online and in the Gupta Forum to send from your out basket.



Sending immediately. If you prepared the message on-line, you click the **Send** button in the message's dialog. If you are not in the Gupta Forum, WinCIM asks if you want to go there to post the message.

Tip: Versions of WinCIM prior to 1.3 have been known to create multiple Forum Database entries for the same forum and ask you if you want to "Go" to the forum even though you are already there. Simply respond **Yes** and let WinCIM take you out and back into the forum. To clear up the duplicate Forum Database entries: go off-line and be sure that you have no messages in your out-box. Pull down the **Special** menu, select **Forum Database**... and delete each of the multiple **Gupta+** entries.

Checking for replies

There are two types of replies you are interested in: replies directed to you, and replies directed to other in the same topic or "thread".

Checking replies to you. If someone replies directly to one of your messages, you see the darkened **Message Waiting** indicator the next time you GO GUPTA.

Simply click the **Message Waiting** button and the following **Waiting Messages** dialog box appears:

Using Browse and Search Messages to find replies to others. When your thread has two or more participants, replies can be posted to someone other than yourself. You should check back once or twice each day to see if there has been discussion among other participants on the thread.

Use the **Browse Messages** button, or the **Search Messages** button, to find the thread and look for other answers. See the following section for more information.

Reading Messages

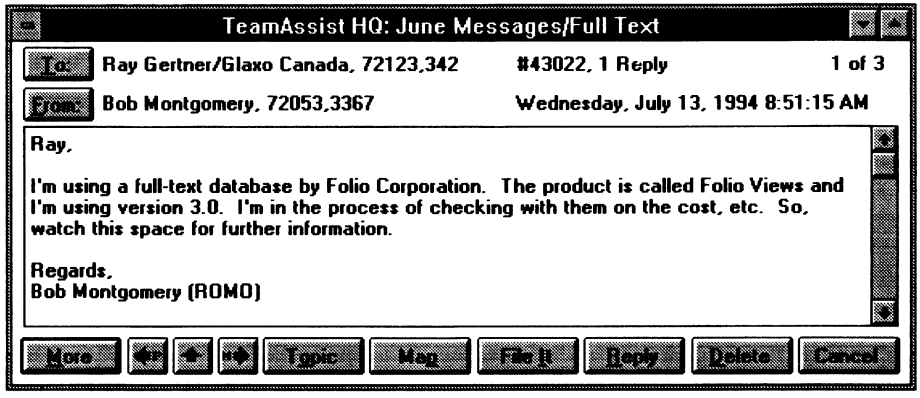
Messages may be read using any one of the following procedures:

Messages, Search + type criteria for search + select a **topic** from the resulting message topic list and press **Get**.

Messages, Browse + choose a **section** from the message sections dialog + choose a **topic** from the message topic list and press **Get**.

Messages, Get Waiting + choose a **topic** from the message topic list and press **Get.**

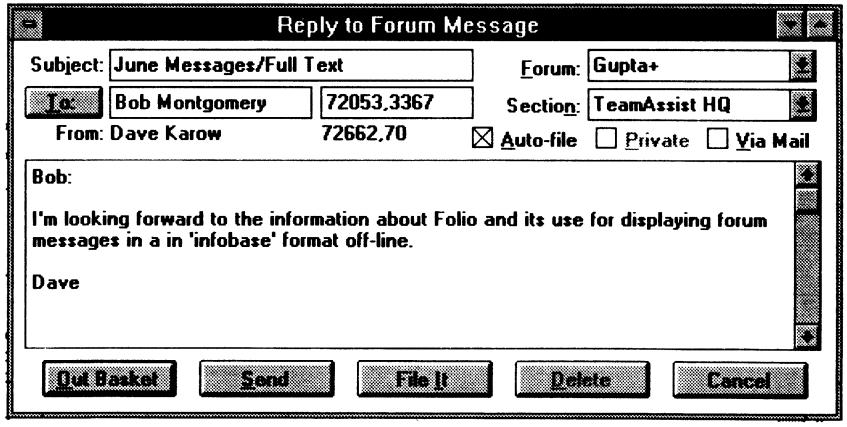
The resulting **Forum Message** dialog looks like this:



To learn about all of the buttons at the bottom of the **Forum Message** dialog, use WinCIM's on-line help and search for **Forum Message**.

Replying to a message

To reply to an existing message, simply press the reply button, type your response and press **Send** or **Outbox** being sure not to alter the **Subject, Forum, To:** or **Section** fields. Leaving these fields alone ensures that your message will be attached to the topic or "thread"as a reply. Changing any of them causes your message to start a new thread (a logically separate topic).



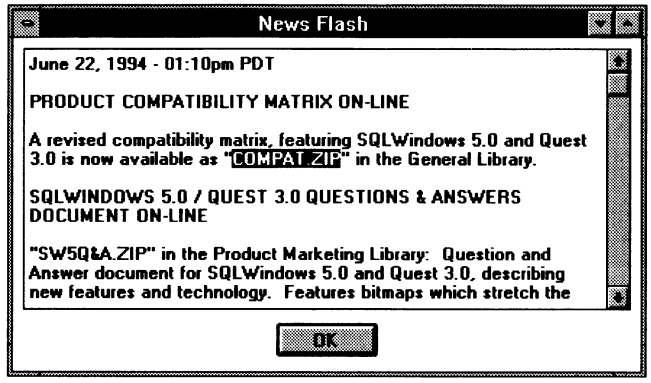
Self-Service Information: File Libraries

Any machine readable file can be stored in the Gupta Forum's library. Files are uploaded (or "contributed" as the WinCIM dialogs call it) by any forum member, including but certainly not limited to Gupta staff. The forum sysops

preview all uploads, check executables for viruses, edit the title, keywords and description that accompany each file as needed and then release the files to the public. These released files are the ones you see when you browse or search the forum's libraries.

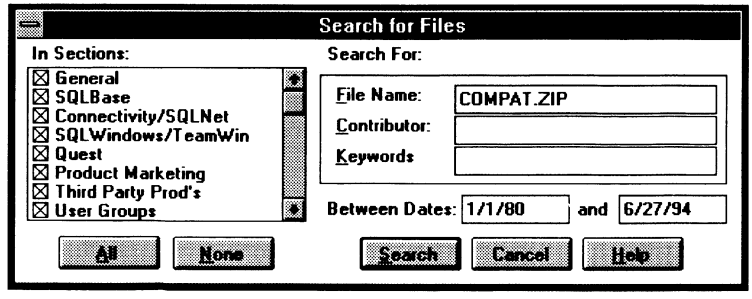
The Search For Files dialog

The fastest way to find a file in a CompuServe forum library is to search by the file name. This process is very fast, requires no scrolling through long browse lists, and looks something like the following. First, we notice in the News Flash that the file COMPAT.ZIP is available:

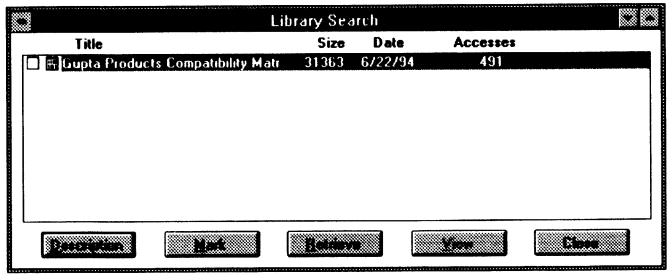


For convenience and accuracy, we highlight the file name and use **Ctrl-C** to copy it to the clipboard. Next, we pull down the **Library** menu and select **Search**, or we click the **Library Search** icon on the forum toolbox:

The defaults make the next step easy. Your cursor is now in **File Name** and all sections will be selected. Just press **Ctrl-V** to paste into the search dialog.

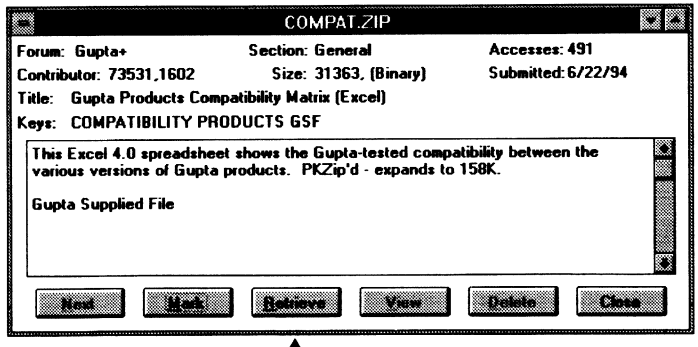


Simply press <**Enter**> or click on **Search** to start the search. Library search results appear in a browse window similar to the one below. The format of the display in library browse windows is controlled in **Special**, **Preferences**, **Forums**.... You may choose to display either file name or title, along with file size, modification date, accesses (download count) and contributor ID#.



Press Retrieve

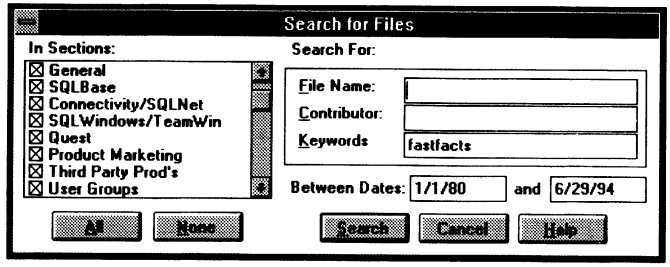
Press **Description** to see the full information kept on-line for each file. Press **Retrieve** from either of the dialog boxes shown above and immediately following to download the file to your computer.



Press Retrieve

Use keywords to view subsets of files

Another powerful way to search is to employ keywords. Try repeating the above search with the file name cleared out and by typing the keyword `fastfacts` in the **Keywords** field.



Currently, the following keywords are in use in the Gupta forum:

Keyword Used	Document
DEVCON94	1994 Developer's Conference Materials
FASTFACTS	Fast Facts documents
FORUMTIP	Success tips for forum use
MESSAGES	Archived Forum Messages
PRESSRELEASE	Gupta/3rd Party Press Releases
TEAMTIP	TeamAssist Tech Tips
TEAMTOOL	TeamAssist Utilities
TRAINING	Gupta Training Information

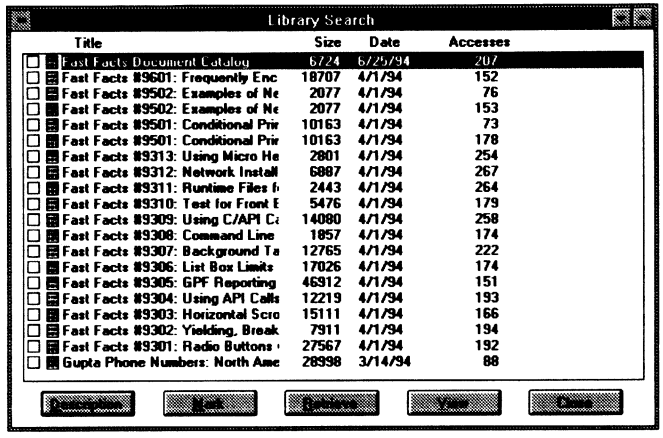
Files to get you started:

Keyword Used	Document
AUTOP.INF	All about forum autopilots and navigators
HELP4U.ZIP	TeamAssist Roster

Keyword Used	Document
MEMBER.ZIP	Gupta Forum Member Handbook

These files are "must haves" for success in the Gupta Forum. Download and read them at your earliest convenience.

This gives you a sub-list of just those files labeled with the FASTFACTS keyword:



Thank you for using Gupta's products!

Sales and Support for SQLWindows Solo

Gupta offers a variety of support options to help get the most from SQLWindows.

If you have any questions about your product, first refer to the printed product documentation, or consult the on-line help. For Technical Support issues, you can visit the Gupta CompuServe Forum for assistance. Alternatively, if you are having installation difficulties you may contact Gupta technical support.

Gupta offers various programs to purchase additional support, upgrade to higher edition of the SQLWindows product, and obtain additional information.

Outside the United States, contact Gupta at the Gupta subsidiary office that serves your area.

Sales Information for SQLWindows Solo

For non-technical support issues, please use these numbers.

From United States and Canada:

Gupta Direct (800) 444-8782

- Sales Information
- Purchase Products and Support
- Renew Support

Fast Facts (415) 617-4600

- Maintenance Release Information
- Current Release Product Listing
- Technical Hints and Tips

Education Information (415) 617-4619

- Education Information
- Order Maintenance Releases
- Product Licensing Information
- Order Product Upgrades

Gupta Partner Recruitment (800) 564-8782

- VARs, VADs, SIs, Consultants/Distributors

Gupta Partner Support (415) 617-3960

Customer Service (415) 617-4700

- Product Shipment Status / Inquiries
- Invoice Inquiries

- Post Sales Account Inquiries
- Product Licensing Information

Outside the United States:

- Gupta Asia +011 852 848 9188
- Gupta France +33 1 46 94 99 66
- Gupta Germany +49 89 7481210
- Gupta Northern Europe +31 3465 52100
- Gupta UK +44 628 478 333

Technical Support for SQLWindows Solo

In the United States and Canada, SQLWindows Solo is packaged with thirty days of free installation assistance. If you have installation difficulties with SQLWindows Solo, please call the appropriate number below.

For all other Technical Support issues, please visit the section on the Gupta CompuServe Forum. Alternatively you may purchase support from Gupta by calling one of the Sales numbers listed below.

Gupta Technical Support Centers:

Gupta US (800) 70 GUPTA

Gupta Asia 323 0178

+65 323 0178 (from outside Singapore)

323 0195 (Fax)

Gupta France

(1) 46 94 99 66 (within France)

+33 1 46 94 99 66 (from outside France)

(1) 46 20 04 30 (Fax)

Gupta Germany

089 74 81 21 45 (within Germany)

+49 89 74 81 21 45 (from outside Germany)

089 78 49 17 9 (Fax)

Gupta Northern Europe

3465 52100

+31 3465 52100 (from outside Holland)

3465 67680 (Fax)

Gupta UK

0628 478 400 (within the UK)

+44 628 478 400 (from outside the UK)

0628 481 076 (Fax)

The SQLWindows support line is staffed by expert technicians who know the product inside and out, and are ready to help you make the most effective use of Gupta's SQLWindows and other Gupta products.

User Groups for SQLWindows Solo

User Groups are a great way to meet other users and exchange ideas, solutions and techniques when developing with Gupta products. There are Gupta User Groups in major areas around the world.

To obtain the latest publication of user groups in the United States and Canada, please call our Fast Facts line.

United States (415) 617-4600 (Fax)

Enter document number - **9902**

The document will then be faxed to you.

For information on user groups in other countries, contact the Gupta subsidiary office in your area.

Glossary

abstract class: An abstract class has no instances and is created for the sole purpose of organizing a class hierarchy or defining methods and variables that apply to lower-level classes. See *instance*.

archive: An organized collection of data. Also, a place where data is kept.

argument: See *parameter*.

application: A SQLWindows program written for a user that applies to the user's work.

Application Actions: A section in the outline that contains procedural code that executes when the application receives messages.

application window: A form window, table window, or MDI window.

attribute: A distinct feature assigned to an object.

backend: See *database server*.

background text: A field that contains static text.

base class: The class from which a given class is derived. A derived class inherits its base class' data structure and behavior. Also called superclass, ancestor, and parent class. Contrast with *derived class*. See also *class*, *inheritance* and *object-oriented programming*.

behavior: The set of operations that an object can perform on its data.

bitmap: A series of bits where one or more bits correspond to each display pixel. For a monochrome bitmap, 1 bit corresponds to 1 pixel. In a gray-scale or color bitmap, multiple bits correspond to each pixel to represent shades of gray or color.

browse: A mode where a user queries a database without necessarily making additions or changes. In a browsing application, a user needs to examine data before deciding what to do with it. A browsing application lets the user scroll forward and backward through data.

buffer: A memory area that holds data during input/output operations.

CASE: The acronym for Computer Aided Software Engineering. CASE consists of software tools that automate or support the process of designing and programming software systems.

case sensitive: A condition in which data must be entered in a specific lowercase, uppercase, or mixed-case format.

cell: The intersection of a row and a column in a table window.

character: A letter, digit, or special character (such as punctuation mark) that represents data.

check box: A box that represents an option. When checked, the option is on; when not checked the option is off.

class: A template that specifies the data and behavior of an object. Objects are created at run time and are instances of a class, while classes are a static description of a set of objects. You create classes in hierarchies, and *inheritance* lets you pass the data and behavior in one class down the hierarchy. See also *base class, derived class, inheritance, object,* and *object-oriented programming.*

class function: A function that you write in a class definition. A class function is the implementation of an object's behavior. An object's functions are shared by all objects in the same class. Also called method or member function.

class variable: Stores data that is shared by all objects in a class. Contrast with *instance variable.*

client: A computer that accesses shared resources on other computers running as servers on the network. Also called front-end or requester. See also: *server* and *database server.*

collapse: Hiding lower outline levels.

column: In a table window, a complete vertical line of cells. See also *table window* and *row.*

In a database, a data value that describes one characteristic of an entity. A column is the smallest unit of data that can be referred to in a row. A column contains one unit of data in a row of a table. A column has a name and a data type. Sometimes called field or attribute. See also *database* and *row.*

Commander: A QuickObject used to manipulate data which is always linked to a data source. See *Data Source* and *Visualizer.*

combo box: A child object that contains a data field and a list box. The list box contains scrollable, pre-defined choices the user selects to fill a data field. See *data field.*

contents: The objects in form windows, table windows, dialog boxes, MDI windows, and tool bars.

cursor: A work space in memory that is used for processing a SQL command. This work space contains the return code, number of rows, error position, number of select list items, number of program variables, rollback flag, and the command result. A cursor is part of a *Sql Handle*.

Cursor is also a name for a mouse pointer.

Customizer: A list of attributes used to define the behavior of a selected object. The customizer is invoked by dragging the window grabber over the object and double-clicking on it.

data field: A one-line data entry/output field.

Data Source: A QuickObject used to define the data access connection. A data source is used to define the data for specific windows in an application. See *Commander* and *Visualizer*.

data type: One of the standard forms of data that SQLWindows can store and manipulate.

database: A collection of interrelated or independent pieces of information stored together without unnecessary redundancy. Client applications can read and write a database.

database server: A DBMS that a user interacts with through a client application on a different computer. Also called *backend*. See also *engine*.

DBMS (database management system): A software system that manages the creation, organization, and modification of a database and access to data stored within it. A DBMS provides centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency, and security.

derived class: A class defined from an existing class (its base class). A derived class inherits its base class' data structure and behavior and it can change (override) the inherited data structure and behavior. It can also add its own data structure and behavior. All of the derived class' data structure and behavior-changed, new, and inherited-is inherited by its own derived classes. Also called descendant, subclass, and child class. Contrast with *base class*. See also *inheritance* and *object-oriented programming*.

design window: A top-level window at designtime. You use the *Window Editor* to add child objects to a design window.

designtime: An environment in which you can create and change an application. Contrast with *runtime*.

dialog box: A single-function window that displays data and messages and accepts input. In general, a dialog box requests or provides information to the user. See also *modal dialog box*, *modeless dialog box*, and *system modal dialog box*.

diamond: The symbol that represents the beginning of an item in the outline.

disabled: A menu item or menu that cannot be chosen; the menu item or menu title appears dimmed or gray.

embed: To insert an object completely within a OLE client application. An embedded object contains a presentation format (bitmap or MetaFile), a data structure that identifies the server, and the native data provided by the server. A user can edit an embedded object directly in the client application. Editing an embedded object starts the server and sends the native data back to that server. See also *link*, *object*, and *OLE*.

encapsulation: This term has two related meanings:

- Combining data and procedures together in a class or object.
- Making the behavior of a class or object visible while hiding the details of its implementation.

Also called data hiding or information hiding. See also *class* and *object-oriented programming*.

engine: A DBMS that a user interacts with through a client application on the same computer. See *database server*.

expand: Displaying lower outline levels.

fetch: To retrieve one or more rows of data from a database.

File Handle: A data type that identifies an open file.

form window: A top-level window used for data entry and display.

format: The appearance of data. Currency, percentage, decimal, date, time, invisible, numbers, and unformatted are examples.

Fourth-generation language (4GL): A type of computer language that accepts system requirements as input and generates a program to meet those requirements as output. This type of language is used for clearly defined procedures such as the generation of menus, forms, and reports.

frontend: See *client*.

function: A routine that performs a task that needs to be done several times but at different places in an application. SQLWindows has 5 types of functions: built-in *system functions*, *internal functions*, *window functions*, *external functions*, and *class functions*.

global: A variable, function, or object known in all parts of an application. Constants are always global. See also *local* and *scope.*

grid: A pattern used to align objects in a design window.

group box: A box that labels a set of related child objects.

group separator: Separates two contiguous sets of radio buttons.

handle: A number that identifies a window, a database connection, or an open file. An application gets a handle by calling a Sal* or Sql* function. The application then uses the handle in other functions to refer to the window, database connection, or file. An application does not know the actual value of the handle.

hierarchy: The relationship between classes. See also *base class, derived class, inheritance,* and *object-oriented programming.*

hWndForm: A variable that contains the handle of the parent.

hWndItem: A variable that contains the handle of a child object.

icon: An image that represents a minimized application or window.

icon file: A file that contains a window icon, specified in .ico format.

impact analysis: A report that shows the impact of a global change to an application or group of applications. For example, the impact of a database change on every screen, in all modules in a project.

instance: An object that belongs to a particular class. For example, *America* is an instance of the class *country.*

instance variable: A variable that contains data that is private to a specific object in a class. Also called field, member, and slot. Contrast with *class variable.*

inheritance: Arranging classes in a hierarchy so that classes can share the data and behavior of other classes without duplicating the code. A derived class automatically includes the data and behavior of one or more base classes. Derived classes can add their own data and behavior and can redefine inherited data and behavior. A derived class can inherit from one base class (single inheritance) or more than one base class (multiple inheritance). See also *base class, class, derived class,* and *object-oriented programming.*

interface: The external view of an object which hides the code that implements its behavior.

library: A collection of SQLWindows objects such as form windows, dialog boxes, or class definitions that are shared by more than one application.

link: In OLE, to create a reference to an object in an OLE client application whose native data is stored in another file maintained by an OLE server for that object. The client application contains only a presentation format such as an icon or bitmap and a data structure that identifies the linked file. The user can edit a linked object directly in the client application. When the object changes in the server application, the changes appear (automatically or on demand) in the client application. See also *embed*, *object*, and *OLE*.

In programming, to connect programs compiled or assembled at separate times so they can be executed together.

list box: A read-only object that displays a list of items.

local: A variable, function, or object that is known only in a single part of an application. See also *global* and *scope*.

master/detail form: The master table contains key information. The detail table stores detailed information. The master table forms the base from which all details are extracted.

maximize: To expand a window so it fills the entire screen.

MDI (Multiple Document Interface): A user interface model created by Microsoft.

MDI window: A workspace used to manage the placement of form windows and top-level table windows.

MDI child window: An object created inside an MDI window, which is its parent window. The parent window owns the child window.

menu: A list of choices from which you can select an action. A menu appears when you click the menu title in the menu bar.

menu item: A choice in a menu or menu bar.

message: The way that objects interact with each other. An object (the sender) sends a message to another object (the receiver) to make a request or notify it of something. The receiver can ignore it or take some action.

Messages make it easy to reuse code. The internals of an object can change while the messages remain the same. Code changes are then highly localized.

message actions: A section in the outline where you code actions that are responses to messages.

minimize: To collapse a window into an icon.

modal dialog box: A dialog box that suspends the application until the user closes the dialog box.

modeless dialog box: A dialog box that does not stop processing within other windows.

module: A set of programs, broken down into components, commonly called modules. Each module has its own set of procedures and data. In general, modules are designed to be as independent of each other as possible.

mouse pointer: A graphic symbol that shows the location of the mouse on the screen. The mouse pointer is usually an arrow, but can change to other shapes during some tasks. Also called *cursor.*

multiuser: The ability of a database server to provide its services to more than one user at a time.

multiline field: A multiple line field for data entry and display.

multiple inheritance: A class or object that inherits properties from more than one class.

null: A value that means the absence of data. Null is not considered equivalent to zero or to blank. The value of null is not considered to be greater than, less than, or equivalent to another value, including a null value.

object: A *window object* is a visual element on the screen such as a table window, push button, or menu.

An *OLE object* is a piece of information such as a drawing, chart, or sound that is linked with or embedded in another application. See also *embed, link,* and *OLE.*

An *OOP object* is a representation of a software entity such as a user-defined window or a user-defined variable. An object has the data structure and behavior specified by its class (which is its blueprint). Also called instance, occurrence, or instantiation. See also *class, encapsulation,* and *object-oriented programming.*

object-oriented programming (OOP): Programming that uses objects which combine data and behavior. Objects use the data structure and behavior of their class. See also *class, encapsulation, inheritance, object,* and *polymorphism.*

OLE (Object Linking and Embedding): A method of sharing information between different Windows applications. By linking and embedding objects, you can combine different types of information in a single application. A SQLWindows application can act as an OLE client. See also *link, embed,* and *object.*

operator: A symbol or word that represents an operation to be performed on the values on either side of it. Examples of operators are: arithmetic (+, -, *, /), relation (=.!=, >, <, >=, <=), and logical (AND, OR, NOT).

outline: The statements that define the objects and procedural logic in an application.

Outline Options bar: The part of the Outline window that lists options that are appropriate for adding to the current outline section.

outline items: Each line of text in an outline.

Outline window: The window that displays the application outline. You use the Outline window to write and test an application with SQLWindows.

parameter: A value for a function that defines the data or controls how the function executes. Also called argument or operand.

polymorphism: The ability for different objects to respond to the same request in different ways. For example, both a push button and a scroll bar can respond to a paint message, but the actions they take are different. The sender does not need to know the type of object that is responding to the message. Also called overloading. See also *OOP.*

populate: To fill a table window with data from a source such as a database, array, or file

popup menu: See menu.

profile: A set of format specifications.

push button: An object that invokes an action when the user clicks it.

query: A request for information from a database, optionally based on specific conditions. For example, a request to list all customers whose balance is greater than $1000. You give queries with the SQL SELECT command.

QuickForms: An automatic or manual feature for building forms from Data Sources.

QuickObject: Pre-defined objects, defined in a class library, for use in SQLWindows applications. A QuickObject is a data source, a visualizer or a commander. See *data source, visualizer*, and *commander.*

QuestWindow: An object that lets the user build Quest applications from a query or table activity.

radio button: A circle that represents an option that can be on or off. In a group of radio buttons, only one can be on at a time.

Repository: A centralized, multi-user database that holds a data dictionary of information about the application database. It also holds project-related information, files, and miscellaneous information.

row: In a table window, a complete horizontal line of cells in a table window.

In a database, a set of related columns that describe a specific entity. For example, a row could contain a name, address, and telephone number. Sometimes called record or tuple. See also *table* and *column.*

runtime: The time during which a user executes a program.

SAL (SQLWindows Application Language): A procedural language for writing actions that execute at runtime.

scope: The part of an outline where the name of a variable, function, or object is known. See also *local* and *global*.

scroll bar: A rectangle with an arrow on each end and a square box. A window can have either a vertical or horizontal scroll bar, or both. A horizontal scroll bar runs along the bottom of the window, a vertical scroll bar runs along the right side of the window.

scroll box: The box in a scroll bar. The scroll box indicates the position relative to the entire window's contents. Also called thumb or elevator box.

section: A parent item and all its children.

server: A computer on a network that provides services to client applications. See also: *client* and *database server*. The term server is also used to refer to applications in DDE, ReportWindows, and OLE.

single inheritance: A mechanism where a class can make use of the methods and variables defined in the class above it on that branch of the class hierarchy.

single-user: A database server that can only provide its services to one user at a time.

standard class: A built-in SQLWindows object. See *QuickObject*.

status bar: A bar at the bottom of the designer window that shows the setting of the Num Lock, Scroll Lock, and Caps Lock keys, as well as other information such as the current menu pick or tool bar pick.

structured development: A top-down approach to program design in which a program is consistently broken down into components. Each of the components is decomposed into a sub-component. This type of development involves separate teams of programmers, who each work on various components, which are assembled into a complete program at the end of the implementation phase of development. Contrast *object-oriented programming*.

subclass: A class that is a special case of another class. For example, *deer* is a special case of *Mammal*. See *derived class*.

With the inheritance process, all the subclasses for a given class use the methods and variables of that class. Large groups of objects are programmed only once, in the higher-level class, and the subclass is used only to add or modify behavior as required for special case classes.

sizing pointer: A pointer you use to change the size of an object.

SQL (Structured Query Language): A standard set of commands used to manage information stored in a database. These commands let users retrieve, add, update, or delete data. There are four types of SQL commands: Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL), and Data Control Language (DCL). Pronounced *ess-que-ell* or *sequel.*

Sql Handle: A data type that identifies a connection to a database. See also *cursor.*

SQLWindows: A graphical SQL application development system for Microsoft Windows.

statement: A unit in the application outline that specifies an action for the computer to perform.

string: A sequence of characters treated as a unit.

system function: A built-in SQLWindows function that performs an often-needed task.

system modal dialog box: A dialog box that suspends all Window applications until the user closes the dialog box.

table: The basic data storage structure in a relational database. A table is a two-dimensional arrangement of columns and rows. Each row contains a like set of data items (columns). Also called relation.

table window: An object that displays data in a tabular format (columns and rows). A table window can be a top-level or child window.

template: The definition of an object created at run time.

tool bar: A rectangular area at the top of the window where objects are placed. The objects represent the functions of an application.

Tool palette: The moveable Tool palette contains icons that represent graphical objects like push buttons and data fields. These icons are used to add objects to an application during designtime. See *designtime.*

top-level window: A form window, table window (that is not a child), or dialog box.

user mode: A designtime mode in SQLWindows when you compile and run an application.

value: Data assigned to a constant or a variable.

variable: A named item that can be any of a given set of values. Contrast with *constant.*

Visualizer: A QuickObject that displays and interacts with the data from a data source. Examples of Visualizers are data fields, check boxes, radio buttons, and business graphics. See also *Commander* and *Data Source.*

window: A rectangular area on the screen where an application receives input from the mouse or keyboard and displays output.

Window Editor: SQLWindows' drawing functions. You use the Window Editor to add child objects to a *design window.*

window function: A function that you write in a top-level window (form window, dialog box, or table window) or in an MDI window. The scope of a window function is in actions in the window.

Window Grabber: A mouse pointer that moves an object.

Window Handle: A data type that identifies a single instance of a particular window.

Windows: A graphical user interface from Microsoft that runs under DOS. In Windows, commands are organized in lists called menus. Icons (small pictures) on the screen represent applications. A user selects a menu item or an icon by pointing to it with a mouse and clicking.

Applications run in windows that can be resized and relocated. A user can run two or more applications at the same time and can switch between them. A user can run multiple copies of the same application at the same time.