A Reviewer's Guide to Oracle Developer/2000 Release 2.0 August 1997

A Reviewer’s Summary

Developer/2000 is Oracle Corporation's major offering in the application development arena. Working in tandem with the Network Computing Architecture (NCA), Oracle7, Oracle8, and Designer/2000 as well as many third-party products, Developer/2000 provides a complete development environment for building form, report, and graphics applications. This paper introduces the Developer/2000 architecture and shows how the product relates to other parts of the Network Computing Architecture. It details the major new features of Developer/2000 release 2.0, then gives you a guided tour of the new product to build a simple application.

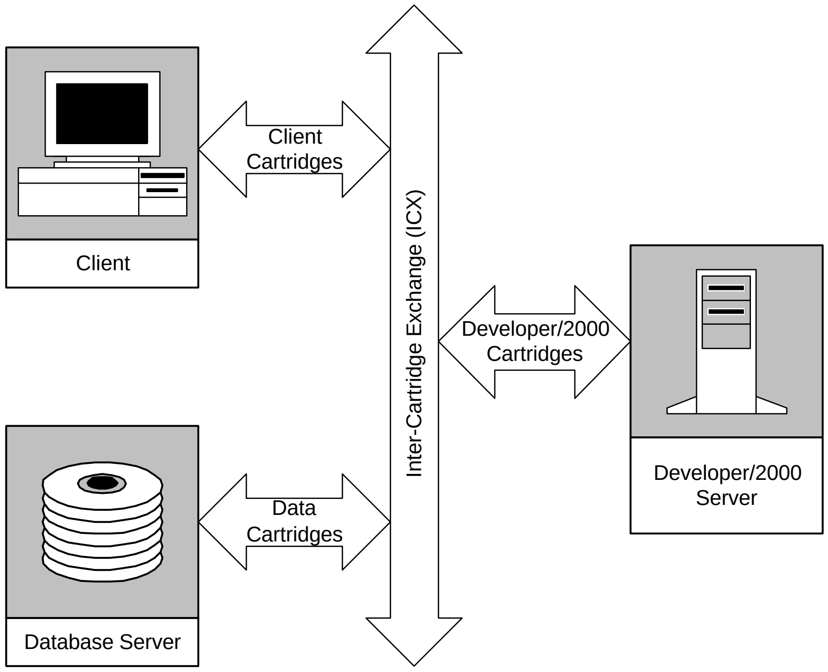
The Developer/2000 Architecture

Developer/2000 is Oracle Corporation's primary application builder. It comprises several different major systems that work together through open standards to give you the ability to rapidly develop corporate network and Web applications that combine all the advantages of client/server computing with the flexibility, low cost, and ease of use of the Web.

Redefining the Architecture of Computing—NCA and the Web

ActiveX, DCOM, CORBA, Java, JDBC, JSQL, HTTP, HTML, IIOP--the commitment to technology an application developer must make has changed its face and its many names. What was client/server computing just a little while ago has grown into a hydra-headed labor of Hercules as multi-tier application architectures merge with the World Wide Web.

Oracle Corporation's Network Computing Architecture (NCA) is leading the way to taming this monster. NCA cartridges are objects that communicate with clients, servers, and each other through the NCA's Inter-cartridge Exchange (ICX) software bus. This technology uses the open standards of HTTP, HTML, CORBA, and Java to provide a comprehensive open architecture that works with standard client browsers, web servers, and database servers.



Developer/2000 provides cartridges for its three major components: Forms, Reports, and Graphics. Each of these components has a complete development environment. With the Procedure Builder for server-side debugging and the Query Builder for managing queries, the tools integrate with one another seamlessly to give you a complete solution to your application development needs. Tied together with the Project Builder, a new visual project environment, these tools combine into a powerful system for building applications rapidly.

If you combine Developer/2000 with Designer/2000, Oracle's modeling and application generation tool, you can integrate your analysis, design, build, test, and deployment processes into a single start-to-finish environment. Developer/2000 fully integrates with Designer/2000, making use of its extensive generation and repository capabilities.

The Form component of Developer/2000 builds event-oriented applications that respond to client input wherever they might be on the network. The application works through a combination of interacting objects and triggers that fire when specific events occur. It also provides a runtime debugger you can use to interactively debug your applications using the same technology as the Procedure Builder.

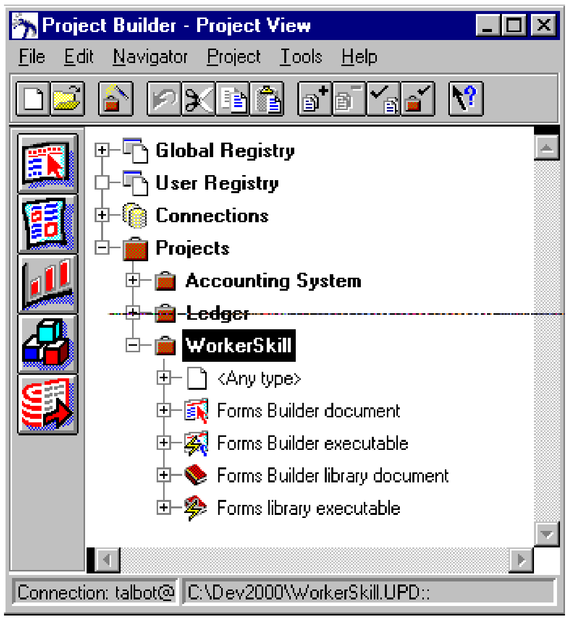
The Report component of Developer/2000 builds a comprehensive array of report styles that execute on a report located anywhere on your network. You can run reports synchronously or asynchronously. You can even build interactive, drill-down reports that you can display in a report viewer for the client.

The Graphics component, while it can display charts in a standalone viewer, integrates with the Form and Report tools to let you embed a comprehensive array of chart styles and other graphics in your applications. You can build the charts using data in your form or report, or you can get the data from the server. You can build interactive, drill-down charts using just a few property settings, then display these active charts in your applications. You can add timers to your charts to update them automatically.

The Project Builder

The Project Builder is Developer/2000's tool for launching the various Builders, version control, and system building.

The Project Wizard helps you to create new projects by entering basic information about the project.



The Project Builder Launcher lets you launch any tool by clicking on its icons. It comes configured with the five main Developer/2000 tools: the Form Builder, the Report Builder, the Graphics Builder, the Procedure Builder, and the Query Builder. You can add any tool you like to the Launcher tool bar to extend your development environment with third-party tools.

The Procedure Builder tool bar and menu system give you access to all the capabilities of the Builder:

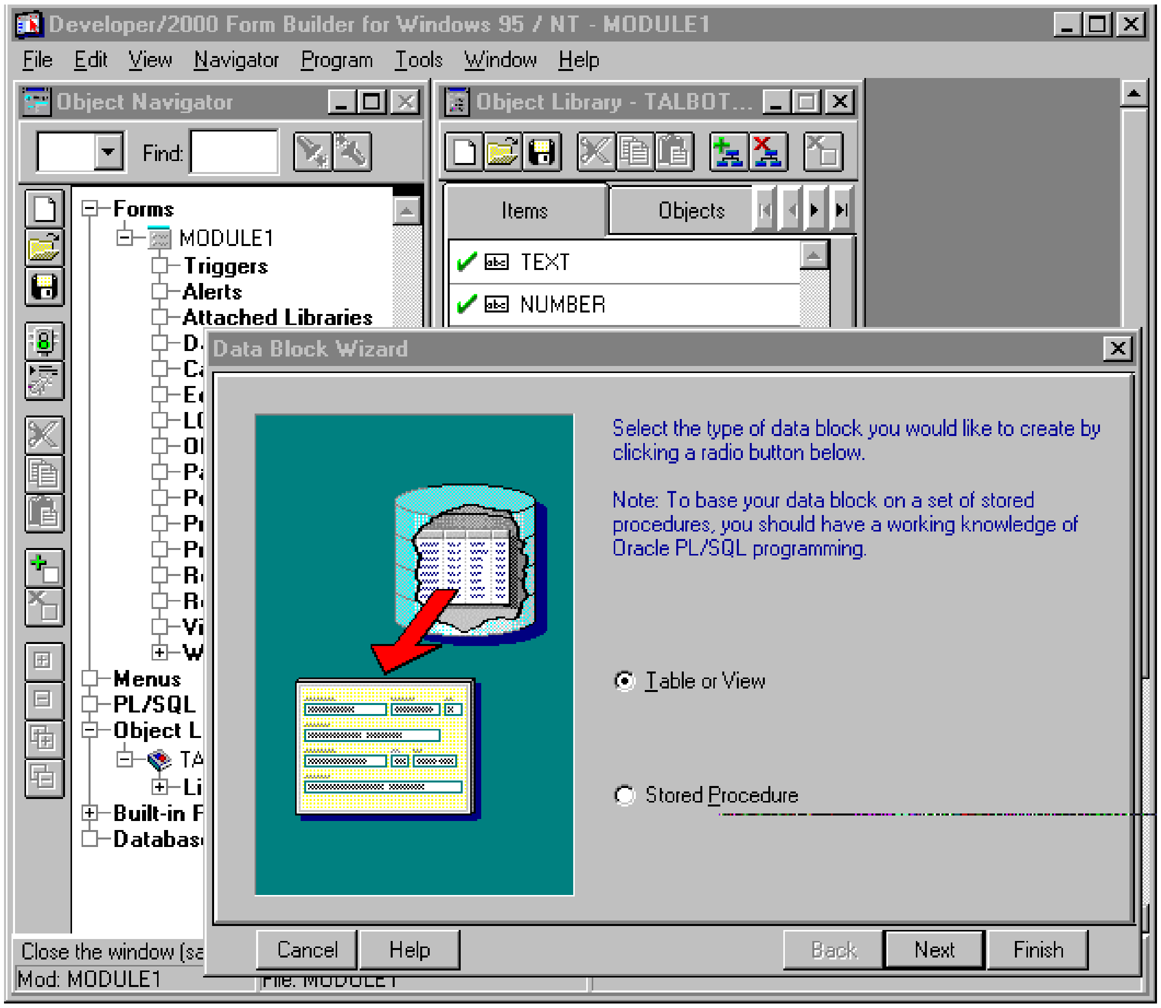
- Importing and exporting projects
- Adding and removing files from projects
- Compiling all files, selected files, or files that have changed (incremental)
- Version control (checking in files, checking out files, setting options)
- Delivering the project

The Project Builder hierarchy is a set of objects, including the Registry, each of which has a set of properties. By adding new objects to the hierarchy, and by changing the properties using the simple scripting language the Builder provides, you can extend the Builder to manage any kind of process or file. For example, you can add a DOC file to integrate design documents into the Builder, or you can change the tool you use to deliver the project (by default, you use PKZIP).

The Module Builders

The Form, Report, and Graphics Builders are the three main development tools you use to build your applications. Each contains an Object Navigator with a hierarchy of modules and objects plus a set of editors for the various components of the modules. All the builders give you access to PL/SQL libraries and to the server-side schema objects in your database (users, tables, views, and stored program units).

The Form Builder lets you build forms, menus, PL/SQL libraries, and form object libraries. The Data Block Wizard lets you create and modify data blocks from tables or views in the database schema. The Layout Wizard lets you create and modify form canvases and windows (including stacked and tabbed canvases) using the data blocks you've created. The Layout Editor lets you edit the canvas layout down to the pixel, and the PL/SQL editor lets you edit and compile PL/SQL code for your report triggers. The Property Palette provides an easy-to-use way to change object properties, which lets you have extremely fine control over the objects you create. The Menu Editor lets you build menus, items, and the PL/SQL code they execute. The Report Wizard lets you integrate reports into your application, while the Chart Wizard lets you integrate charts. The Object Library window lets you drag and drop objects into object library repositories for reuse across your applications.

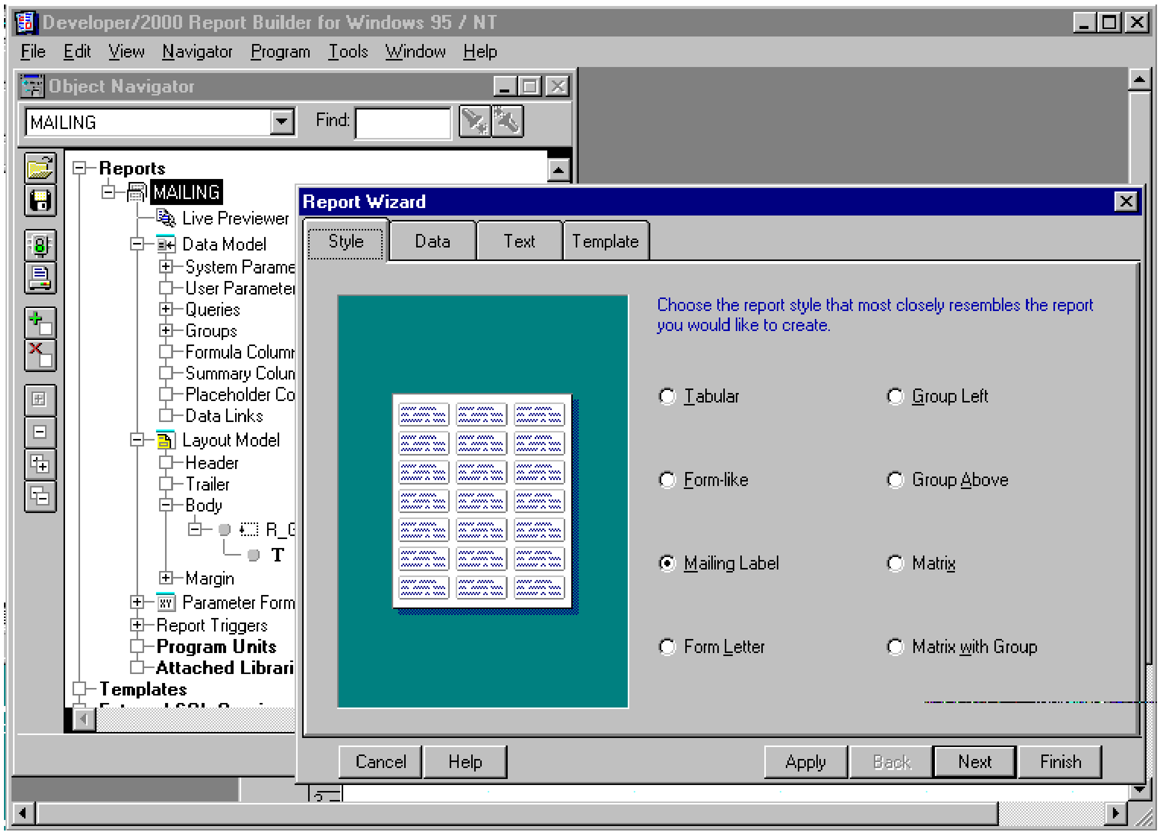


Select the type of data block you would like to create by clicking a radio button below.

Note: To base your data block on a set of stored procedures, you should have a working knowledge of Oracle PL/SQL programming.

[✓ Table or View] Stored Procedure

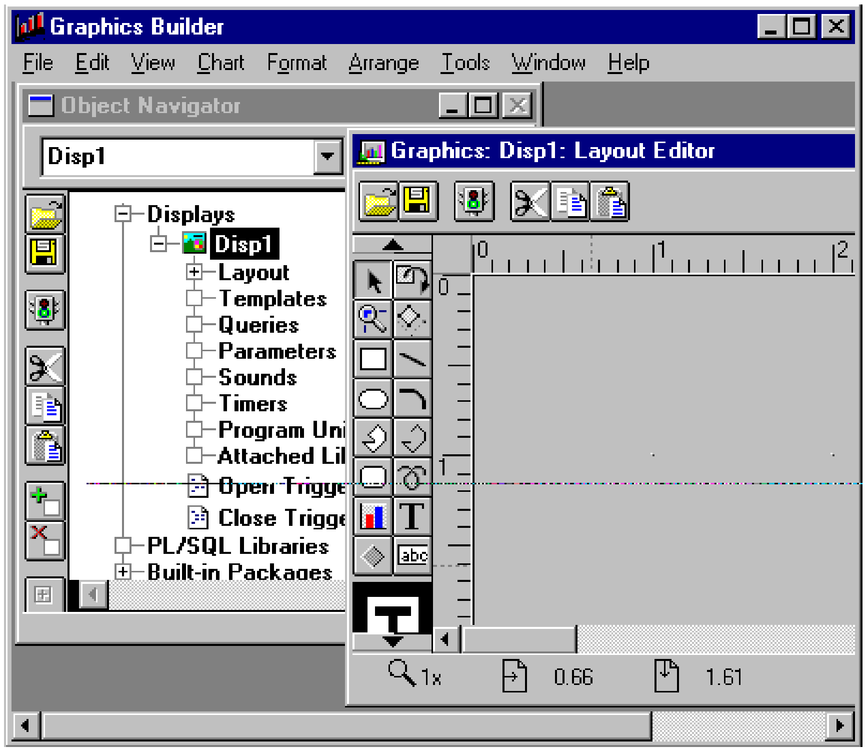
The Report Builder lets you build reports and report templates as well as PL/SQL libraries for use in reports. The Report Wizard takes you through the entire process of building or changing even moderately complex reports (tabular, grouped, mailing label, form, and matrix/crosstab reports are all a part of the Wizard). The Chart Wizard lets you integrate charts from the Graphics component into your reports. The Live Previewer displays the report using real data from the server and lets you arrange it to your heart's content. The Report Editor gives you a graphical environment for building the report's data model, layout, and parameter form. You can integrate multiple queries, data links, groups, and summary columns in the data model, and you can edit in fine detail the structure and formatting of the report layout. The Web Wizard lets you build HTML and Adobe Acrobat report structures, including Acrobat bookmarks and hypertext links.



Choose the report style that most closely resembles the report you would like to create.

( ) Tabular
( ) Group Left
( ) Form-like
( ) Group Above
[✓ Mailing Label]
( ) Matrix
( ) Form Letter
( ) Matrix with Group

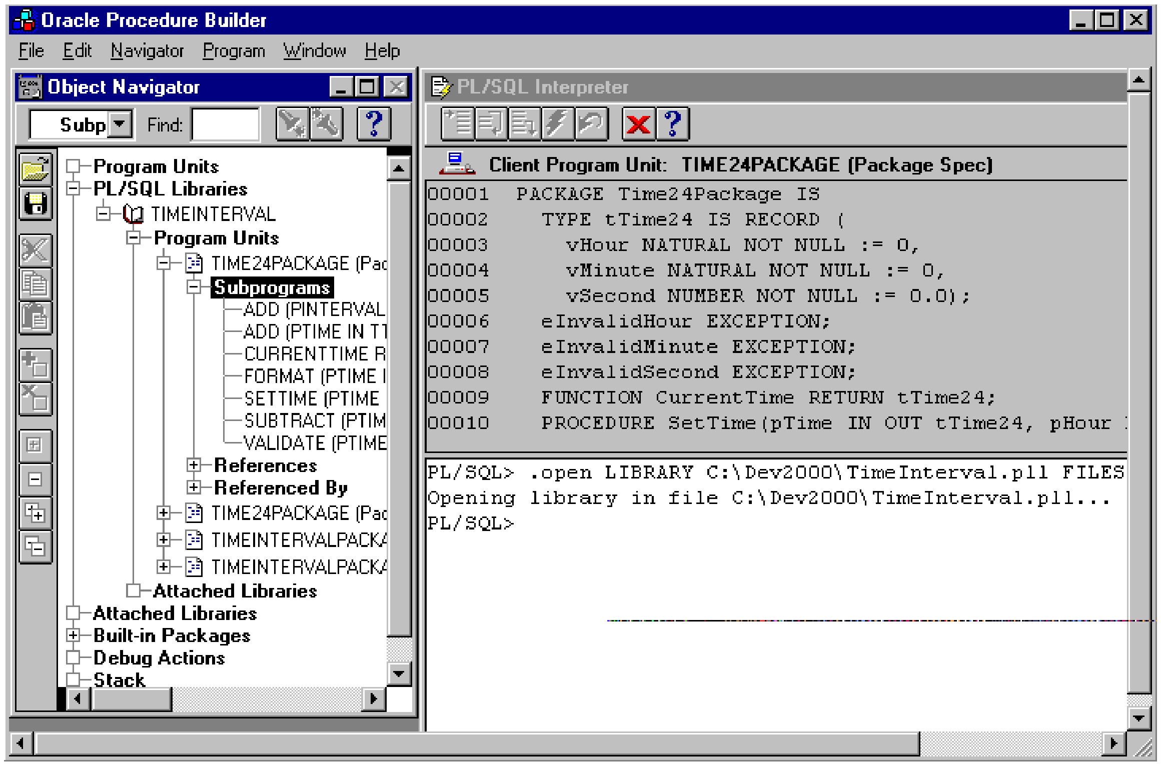
The Graphics Builder lets you build charts and chart templates as well as PL/SQL libraries for use in those charts. The Chart Genie is a wizard that takes you through the steps of creating and modifying your data model and chart structure. The Layout Editor displays the charts you build using real data from the server, then lets you modify the properties of the chart objects.



The Procedure Builder and Query Builder

The Procedure Builder and Query Builder are auxiliary tools that complement the three main builders. These tools help you to build and debug PL/SQL and SQL code that you use in your applications.

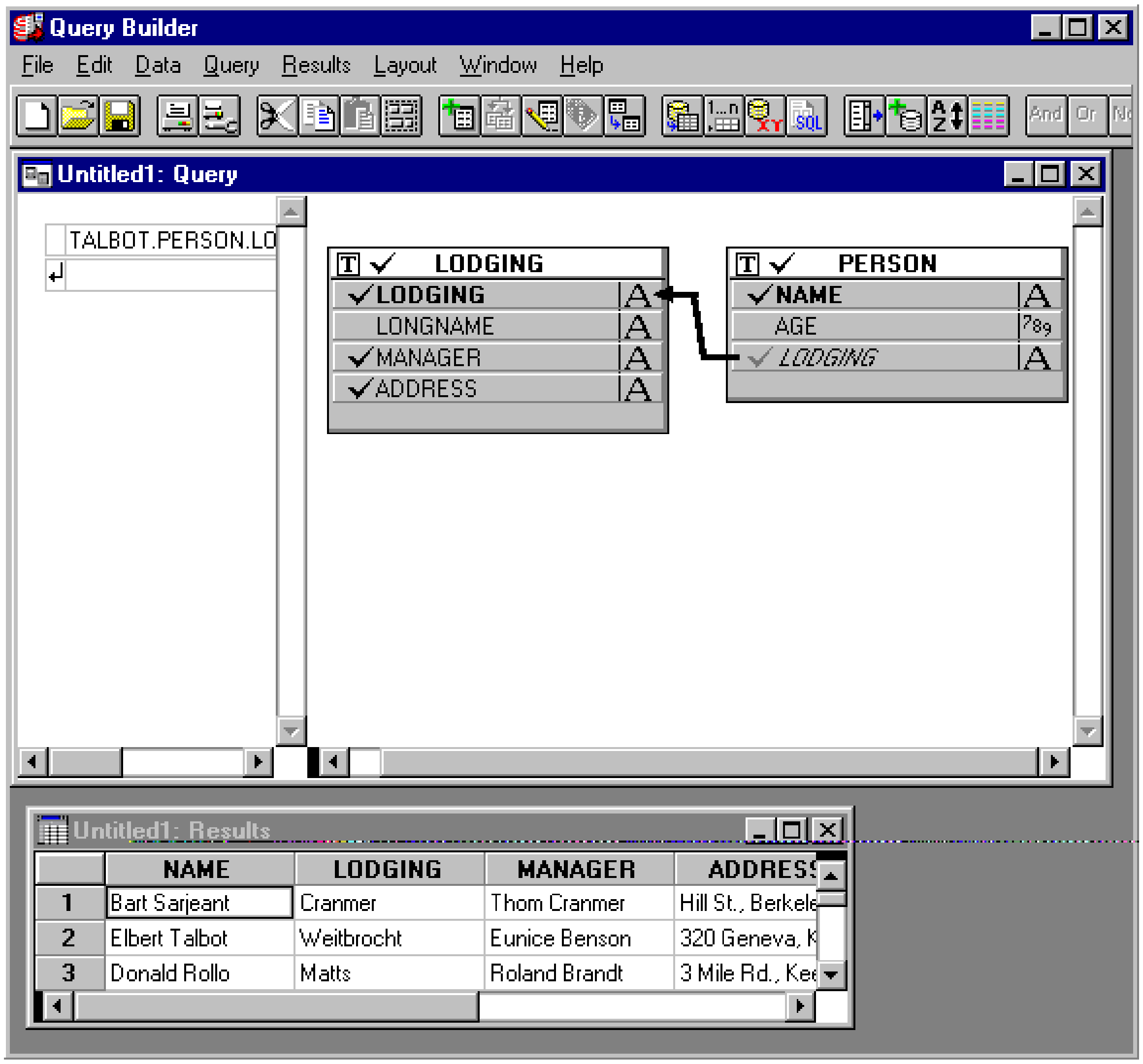
The Procedure Builder is a standalone PL/SQL interpreter and debugger that lets you build, compile, run, and debug both client-side and server-side PL/SQL program units. The Procedure Builder integrates the process of building PL/SQL code that is independent of any of the three main components of Developer/2000. You can move such code between client-side libraries and the database server by dragging and dropping, letting you easily structure your multi-tier PL/SQL architecture. The PL/SQL Interpreter lets you run and debug code wherever it resides, whether on the client or the server. The Program Unit Editor lets you enter and compile PL/SQL procedures, functions, and packages.



```
PACKAGE Time24Package IS
  TYPE tTime24 IS RECORD (
    vHour NATURAL NOT NULL := 0,
    vMinute NATURAL NOT NULL := 0,
    vSecond NUMBER NOT NULL := 0.0);
  eInvalidHour EXCEPTION;
  eInvalidMinute EXCEPTION;
  eInvalidSecond EXCEPTION;
  FUNCTION CurrentTime RETURN tTime24;
  PROCEDURE SetTime(pTime IN OUT tTime24, pHour
```

```
PL/SQL> .open LIBRARY C:\Dev2000\TimeInterval.pll FILES
Opening library in file C:\Dev2000\TimeInterval.pll...
PL/SQL>
```

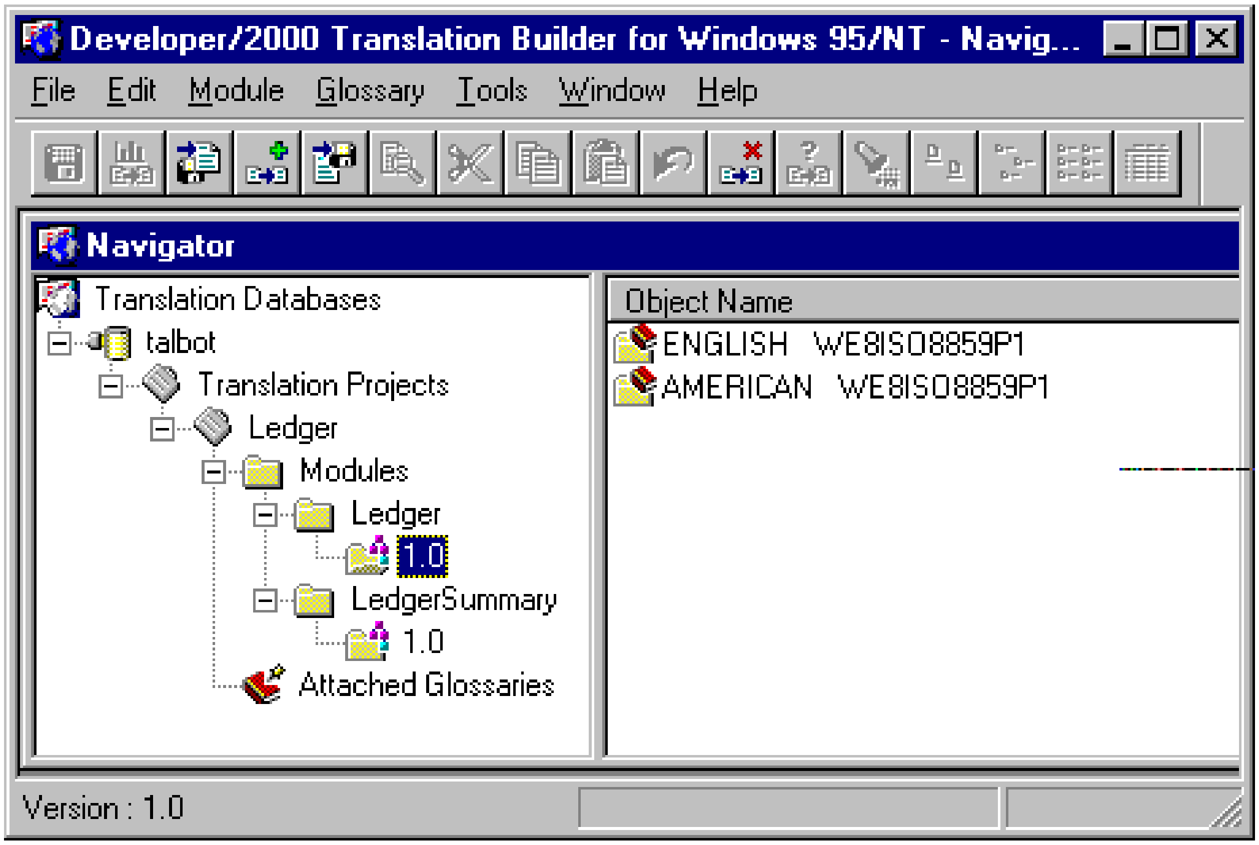
The Query Builder lets you create SQL query modules in a highly usable graphical environment. The tool accesses the server data dictionary to display and link tables and columns. Using these, you create the SQL select list, FROM clause, WHERE clause, GROUP BY clause, and ORDER BY clause of the SELECT statement. You can inspect and modify the resulting SQL in the SQL window, then execute the query and display the results in a spreadsheet-style Results window. You can perform basic formatting and grouping in this window as well for informal reports. You can run the Query Builder from the Report Builder to build the report data model, or you can save the SQL statement in a file for use in any tool.



	NAME	LODGING	MANAGER	ADDRESS
1	Bart Sarjeant	Cranmer	Thom Cranmer	Hill St., Berkele
2	Elbert Talbot	Weitbrocht	Eunice Benson	320 Geneva, K
3	Donald Rollo	Matts	Roland Brandt	3 Mile Rd., Kee

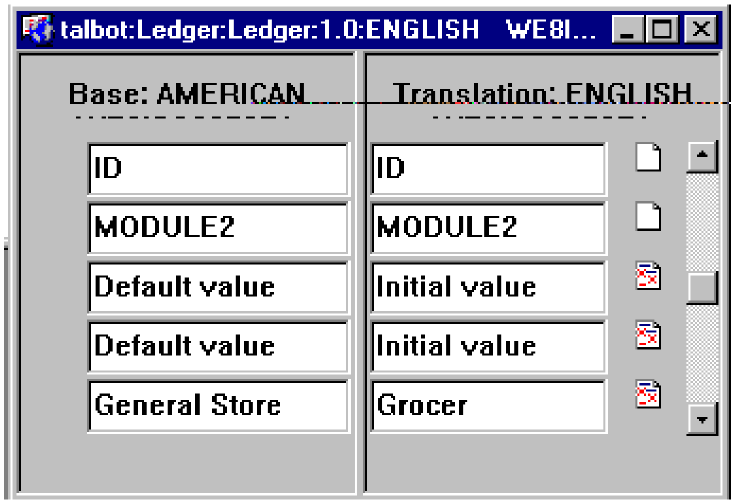
The Translation Builder

The Translation Builder translates all the string resources in a Developer/2000 application from a base language into multiple target languages using a set of database tables you create on your server. The Translation Builder works through a set of database tables you install in a central user and make available through public synonyms.



Object Name	
ENGLISH	WE8ISO8859P1
AMERICAN	WE8ISO8859P1

The Translation Builder provides a separate project manager that lets you manage a set of modules that you tie together into a single application as a single translation project. Each project has a name and the base language and character set, usually taken from the current setting of the NLS_LANG environment variable, which you'll find in the System Registry with all the other Oracle variables.



Base: AMERICAN	Translation: ENGLISH
ID	ID
MODULE2	MODULE2
Default value	Initial value
Default value	Initial value
General Store	Grocer

The first step in creating a translation is to import the base version into your project. The initial translation is the default project language and character set, but you can change these as required. You also have the ability to specify versions of files so you can maintain several revisions of your translations.

You display the translations in the left pane of the Translation Builder Navigator and double-click the one you want to translate to display the Translation Editor with its side-by-side listing of the strings. You just type in the translation on the right.

The final step is to export the translated strings back into the Developer/2000 module. You use the Export Strings facility to write all the translations into the single module file, which contains all the translations. When you run with NLS_LANG set appropriately, you see the translated strings rather than the base strings. This powerful facility greatly improves your ability to deploy applications on a global basis.

Running Applications on Multiple Platforms

Client/Server Applications

Developer/2000 realizes the promise of multi-platform, client/server computing. Using a graphical user interface standard to the client platform, you can access a database server running anywhere on a network, whether on a mainframe, a minicomputer, or a file server. You can develop client applications that run under Microsoft Windows, the various flavors of UNIX, and the Macintosh without changing any application code, just recompiling.

Thin-Client Web Applications and the Multi-Tier Architecture

You can also build applications that run on any thin client that supports a Web browser capable of running Java 1.1. The implications of this architecture, however, go beyond just supporting a new GUI platform. You are now running in a multi-tier architecture rather than in a client/server architecture.

A multi-tier architecture consists of a thin client, one or more application servers, and one or more database servers, all integrated across a network.

The Developer/2000 Forms Cartridge is a modified version of the standard Forms runtime engine that runs on an application server. The Forms Cartridge communicates with the other cartridges and with the thin client and database server to perform most of the processing in an application. The Reports Cartridge is a multithreaded application server that can schedule and generate thousands of concurrent reports in batch print, HTML, or PDF formats. You can schedule and execute multiple reports to multiple destinations with a single networked reports cartridge. The Graphics Cartridge offers a similar application-server capability to provide dynamic charts and graphics to your applications on the Web.

The Forms Cartridge downloads a small Java applet through a Web server to the thin-client browser when the client starts up a Developer/2000 application through a Web page. The cartridge connects to the database server, then interprets thin-client events and sends user-interface instructions back to the client. It communicates with the other cartridges to generate reports and graphics. In the future, you will be able to use NCA to communicate with any application cartridge from your Developer/2000 application.

Integrating with Designer/2000

Designer/2000 is Oracle Corporation's application modeling tool. Using the Function and Entity-Relationship Editors of Designer/2000, you model your business functions and database requirements. You then generate your database schema and your module definitions (forms and reports) from the requirements models in the Designer/2000 repository using the Application Design Wizard, which supports full model-driven generation capabilities. Alternatively, you can reverse engineer your existing Developer/2000 forms and reports into module definitions.

Designer/2000's Module Diagrammer then lets you edit the model-generated application prototype into a complete application. It lets you specify a complete mapping of module objects to database server objects and most of the form or report properties from Developer/2000. You then use model-driven generation to create a Developer/2000 form or report module from the Designer/2000 repository. You can also generate HTML or Visual Basic applications from models using this approach.

Developer/2000 thus integrates with Designer/2000 to provide round-trip engineering of your applications.

The true power of Designer/2000 comes from its preferences and templates. These are reusable, configurable settings that you can automatically apply to your modules to set up most of the properties of a Developer/2000 application. A *preference* is a setting that the generator uses to create or format some element of the module. There are over 400 forms preferences and over 200 reports preferences. Forms preferences categories include, for example:

- Coding Style
- Layout (Button, Checkbox, Page, Poplist, and myriad others)
- Roles
- List of Values
- Commenting

Reports preference categories include, for example:

- Coding Style
- Layout Field Style
- Layout Frame
- Report Level Objects
- SQL Generation

The Preferences navigator provides a kind of object navigator for preferences along with its Property Palette. Although the format is slightly different from the Developer/2000 Object Navigator, the logic is much the same.

A *template* is a form or report that contains specially named objects. The template acts as a kind of automated object library for the generators. When you generate a module from a model using model-driven generation, the generator reads the template to set various aspects of the module:

- *Boilerplate:* Text such as the module name used as boilerplate in the form or report
- *Visual attributes:* Named visual attributes to associate with an item
- *Attached libraries:* Standard sets of attached libraries
- *Canvases:* Content, stacked, toolbar, and tab canvases with standard settings
- *Code:* Module program units
- *Specialty items:* Preprogrammed features such as a query mode indicator or special trigger setups that represent solutions to common problems or needs
- *Buttons:* Buttons that execute most of the standard key actions, such as Next_Record or Commit_Form
- *Coordinate system:* The generated module uses the coordinate system settings in the template form

Designer/2000 ships with a basic set of templates that can get you started. Judicious and consistent use of templates can dramatically improve your productivity because you don't have to recreate all the standard settings in every module you create. Designer/2000 adds to the template capabilities of Developer/2000 to produce even better results than using Developer/2000 templates directly with the Form and Report Builders. Compared to using just a fourth generation language such as Developer/2000 alone, using Designer/2000 can result in a 40 percent average productivity gain in the design and build phases of your application projects.

In the near future, Designer/2000 will be able to generate application components from its models for all NCA tiers. It currently generates database server tier objects (tables, columns, indexes, key constraints) for Oracle7 and other database managers. You will be able to generate Oracle Web Application Server cartridges, C++ cartridges, and Developer/2000 cartridges, with Java in a future release. For the thin-client tier, Designer/2000 generates Developer/2000 applications along with Microsoft Visual Basic applications, again with Java applications in a future release.

Integrating Third-Party Software

The Oracle Alliance: Version Control and Testing

The Oracle Alliance Program creates, fosters and expands relationships with leading information technology companies to deliver best-of-breed solutions to customers around the world. By leveraging Oracle's world class products and services, Alliance members build scaleable solutions that incorporate the most up-to-date technology available.

You can find all the members of the Oracle Alliance program on the Oracle Alliance web site at www.alliance.oracle.com. You can use the search tools to find Alliance members in the Open Tools Initiative program with tools that apply to the Developer/2000 Oracle technology.

Developer/2000 ships with integration software for certain configuration management tools from Alliance members PureAtria/Rational Corporation (ClearCase), I (PVCS), and Starbase Corporation (StarBase).

The two automated testing tools that work with Developer/2000 are SQA Manager (SQA/Rational Corporation) and WinRunner and Xrunner (Mercury Interactive). You should seriously consider using these tools for automated regression testing of your applications.

ODBC and Third-Party DBMSs

The Open Data Base Connectivity (ODBC) standard is a de facto, standard application programming interface (API) for database managers. This API lets you write applications that you can move from one DBMS to another without rewriting or even recompiling your code. To use Developer/2000 with a DBMS other than Oracle7, you need to know how it can access that DBMS through ODBC.

The Oracle Open Client Adapter (OCA) is an interface to ODBC that lets you make ODBC calls instead of Oracle7 calls in Developer/2000. The OCA supports your use of any ODBC-compliant data source. Oracle supplies scripts that install views similar to the Oracle data dictionary tables that Developer/2000 uses. The OCA transparently manages the incompatibilities between database managers so that you need to change as little as possible to run an application against a database manager other than Oracle.

You can use the ODBC drivers that Oracle supplies with Developer/2000 to get excellent performance, not just for Oracle databases but for third-party database servers as well. A tip: in reviewing the ODBC interface for Developer/2000, you should compare performance of different kinds of server through ODBC drivers, not through native interfaces. Otherwise, you will get very misleading benchmark numbers.

If you are converting an application to use the OCA, here are several changes you must make to your form object properties:

- Set the Primary Key property of primary key column items in data blocks to Yes.
- Quote any object names that the data source stores in mixed case to enable Developer/2000 to use the names.
- Set the Update Changed Columns data block property to Yes if you get errors about not being able to update primary key columns.
- Adjust the Records Fetched data block property if you are trying to fetch more records than the data source supports at once.

ActiveX

Microsoft Corporation developed Object Linking and Embedding (OLE) to let separate systems cooperate in presenting a single compound document that links to other documents or embeds the documents directly. For example, a Form module can contain objects such as word processor documents or spreadsheets. OLE2 extended the model to use the Common Object Model (COM) to provide automation, which lets documents operate other documents. For example, Microsoft Excel can act as a toolbox of commands relating to spreadsheets for other applications.

With its 32-bit operating systems, Microsoft introduced ActiveX, its replacement for OLE on 32-bit platforms. ActiveX controls, which were at first called OLE controls or OCXs, are standalone software components that send events to clients and display a user interface through COM technology. Third-party developers have made available a wide variety of ActiveX controls, so you can avoid the often significant amount of effort you must spend to develop your own controls.

In Developer/2000, the ActiveX Control item creates a container for an ActiveX control. You should use ActiveX controls in data blocks with the Single Record property set to Yes, meaning the data block contains only a single record. This lets Developer/2000 initialize the data block and hence the control when the form starts up.

Each ActiveX control has a set of properties, methods, and events. Properties define the physical and logical characteristics of the control. Methods are actions that the ActiveX control can perform. Events notify the application of a change in the ActiveX control. A Developer/2000 application writes values to and from control properties, calls methods to get and set properties, and intercepts and acts on events.

To aid in your use of ActiveX controls, the Form Builder imports the methods and events in PL/SQL. To import ActiveX Control methods and events, you take the following steps:

1. Run the OLE Importer and select an OLE class from the dialog. This displays the methods and events for the class.
2. Ctrl-click on the desired methods and events packages, then check the Methods check box to import the method or check the Properties check box to import the get and set accessors for properties.
3. Click OK.

Developer/2000 then creates PL/SQL packages that correspond to the selected methods under the Program Units node in the Object Navigator. To use them, just call any of the methods in any of the packages from triggers or other program units in your application.

When an ActiveX control fires an event, there are two ways Forms Runtime can deal with the event: directly call the appropriate restricted procedure provided by any of the events packages available under Program Units node, or call the On-Dispatch-Event trigger. Form Builder exposes these events in the events package that the OLE Importer creates. Each event corresponds to a PL/SQL procedure in the events package. When the control raises an event, the Developer/2000 runtime system automatically executes the corresponding procedure, in which you code the PL/SQL that you want to run when the event occurs.

Using ActiveX controls is an excellent way to get the benefits of software reuse in Developer/2000. Your interface is thus not limited to the features that Oracle provides but rather by the inventive minds of third-party developers.

Major New Features of Developer/2000 Release 2.0

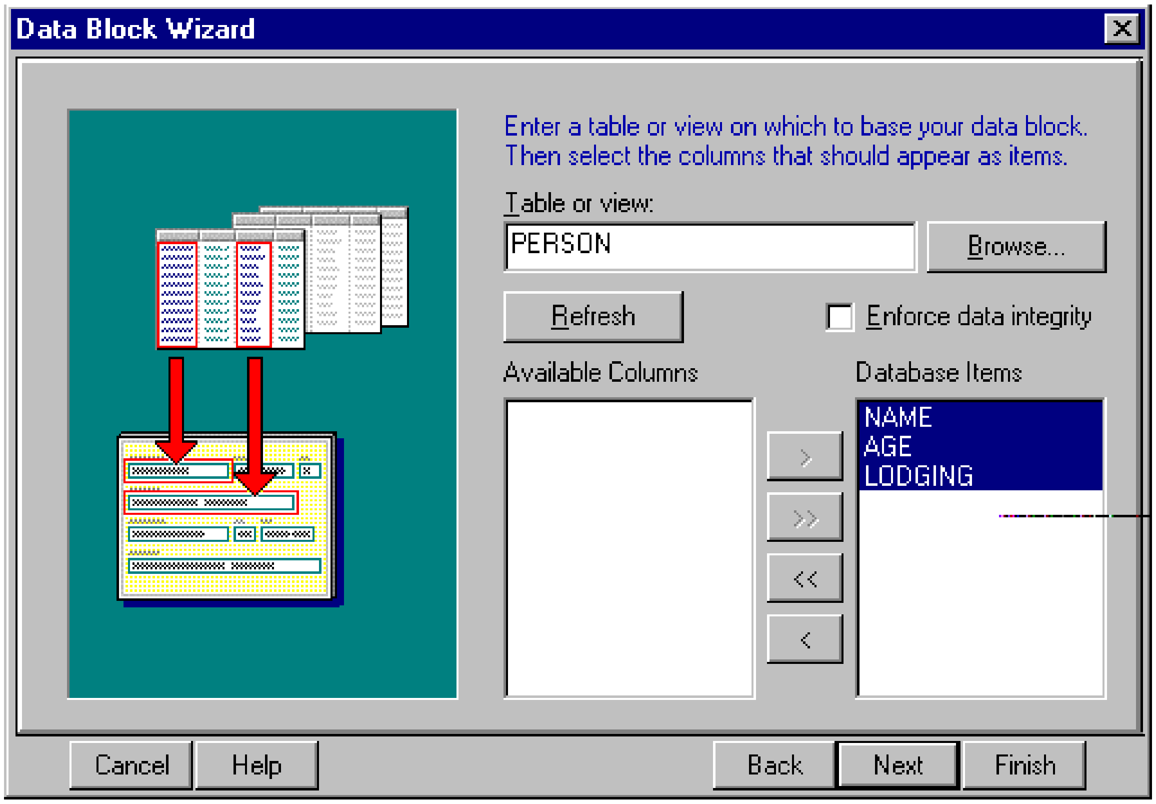
Developer/2000 release 2.0 adds some major new features to the system that dramatically enhance the ease and productivity with which you can develop applications. The new Developer/2000 reentrant wizards make creating and changing the basic objects in your applications much easier, and the new reuse features let you reuse objects much more directly than before.

Wizards

The Forms and Reports components of Developer/2000 have new, reentrant wizards that take most of the pain out of creating and modifying your applications.

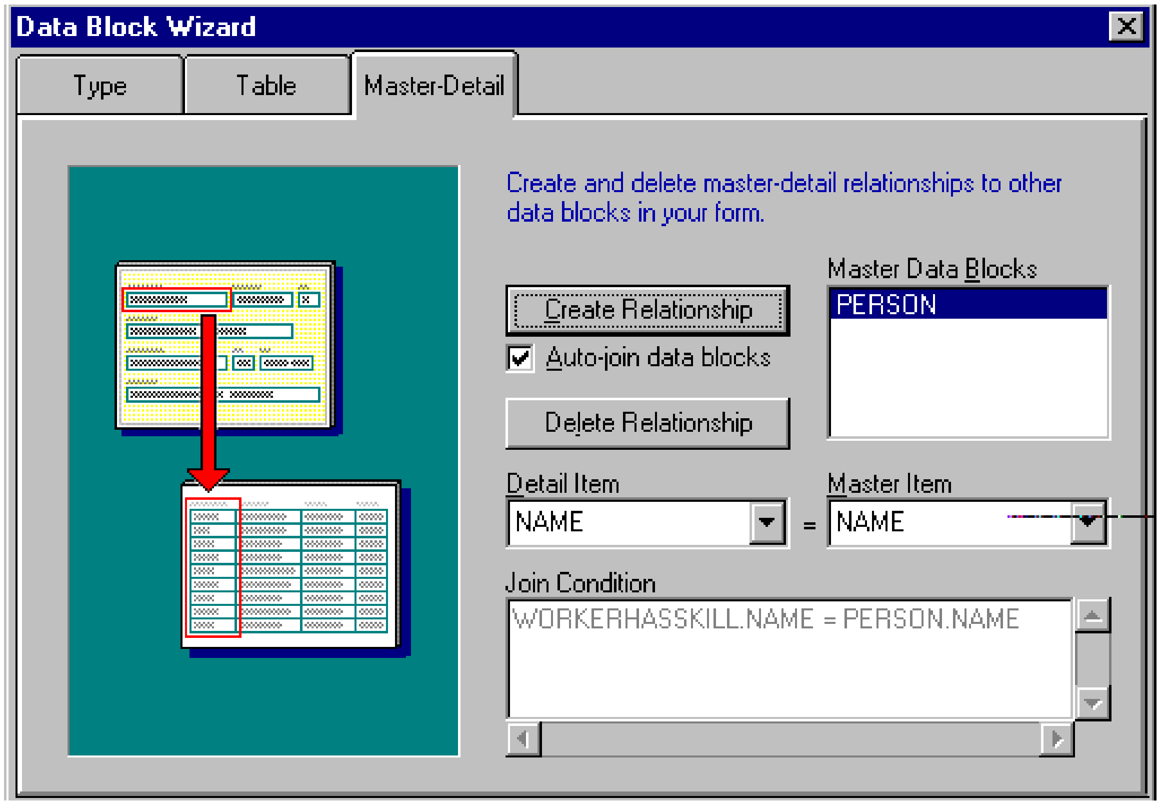
The Data Block Wizard

The data block is a primary component in a form. Data blocks represent the mapping of the form to the data in the database. The reentrant Data Block Wizard automates the process of creating and modifying the mapping by giving you access to the table definitions on the server. You just pick a table or view, and the wizard does the rest.



Release 2 adds the capability to base a data block on a package of stored procedures rather than on a base table or view. This means that you can encapsulate access to the database in the procedures to guarantee that no application can evade the server's business rules by accessing the tables directly. The Data Block Wizard lets you enter the names of the procedures and automatically creates the data items from the record type you define as part of the package. It takes care of all the details outside of actually creating the package on the server.

The Data Block Wizard also lets you set up master-detail relationships between two data blocks. Usually, you just specify the name of the master block when you create the detail block, and the wizard figures out the relationship between the data items, but you can add the relationship at any point by running the Wizard again. The wizard then adds the trigger code that your application needs to manage the master-detail blocks.



Create and delete master-detail relationships to other data blocks in your form.

Master Data Blocks: PERSON

Auto-join data blocks [✓]

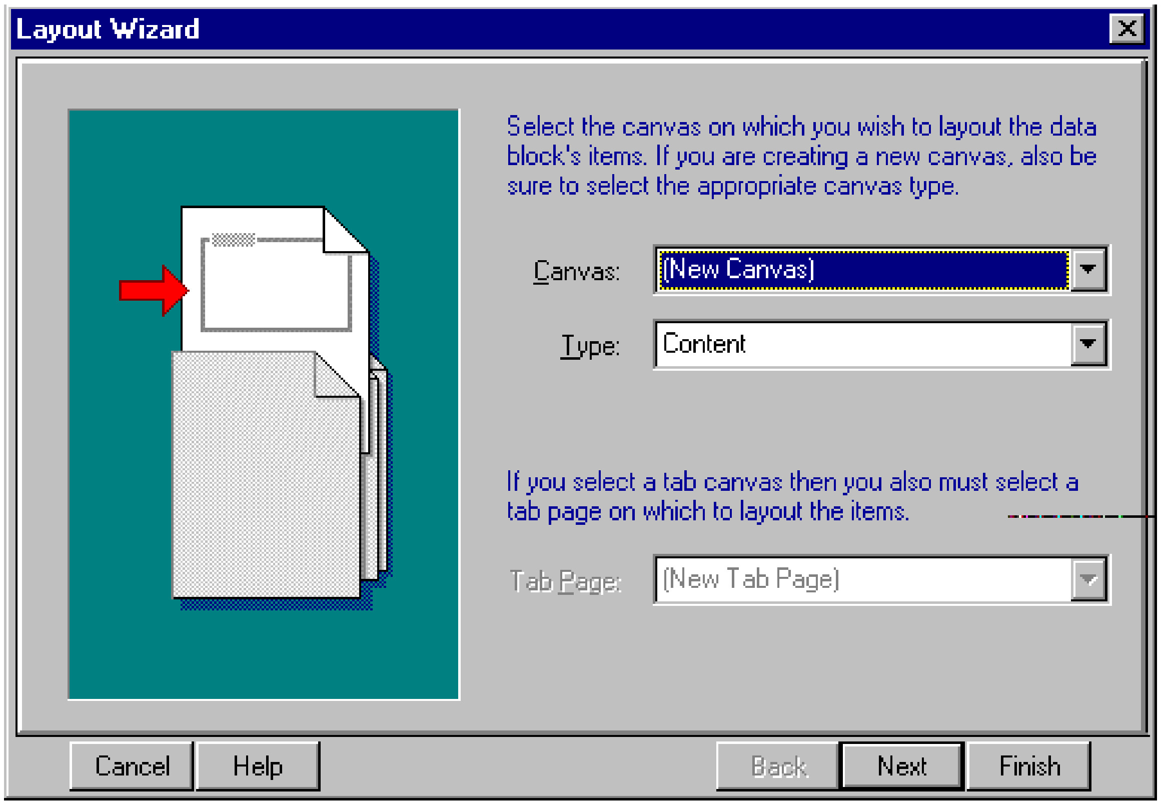
Detail Item: NAME = Master Item: NAME

Join Condition: WORKERHASSKILL.NAME = PERSON.NAME

The release 2 Data Block Wizard lets you modify the data block as well as create it. You just select the block, then run the wizard. You can add or delete items, change the master-detail relationship, and so on.

The Layout Wizard and Tabbed Canvases

Once you've defined your data blocks, you lay out the items on canvases with the reentrant Layout Wizard. A canvas is a rectangular object on which you place items to display. You then display the canvas in a window. You can add stacked canvases on top of the main content canvas to hide and remove data items on demand. Release 2 also adds the tabbed canvas, which lets you create several tabs that the user can control.



Layout Wizard

Select the canvas on which you wish to layout the data block's items. If you are creating a new canvas, also be sure to select the appropriate canvas type.

Canvas: (New Canvas)

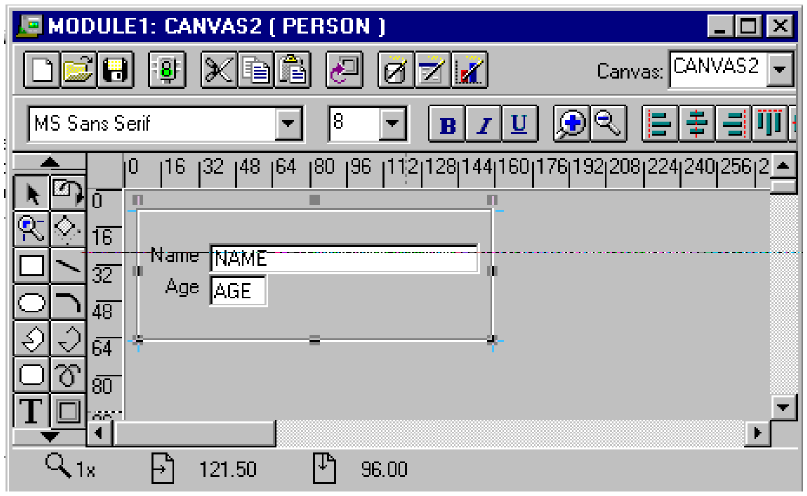
Type: Content

If you select a tab canvas then you also must select a tab page on which to layout the items.

Tab Page: (New Tab Page)

The reentrant Layout Wizard has screens for the following tasks:

- Specifying the canvas on which to place the items in the block (with a choice of content, stacked, or tabbed canvas styles)
- Specifying which items to display on the canvas and their data types
- Specifying the prompts and sizes of the items on the canvas
- Whether to build a form or a tabular display
- Whether to use a form style or a tabular grid-like style for layout
- Row details (number of rows to display, space between rows, frame title, scroll bars)



MODULE1: CANVAS2 (PERSON)

Canvas: CANVAS2

MS Sans Serif 8

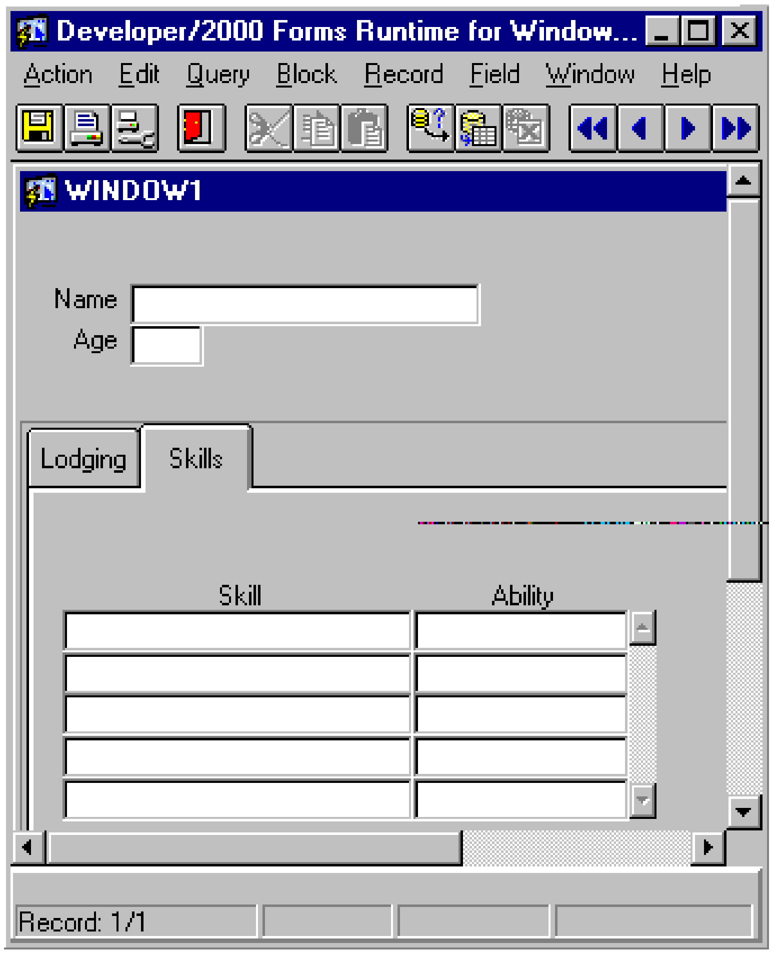
Name NAME

Age AGE

1x 121.50 96.00

Once you've set up the layout, you can use the Layout Editor to modify the arrangement of items. Developer/2000 creates a frame object around the items from a single data block. This frame controls the automatic wizard layout of the items. By selecting it, then running the Layout Wizard again, you can modify the layout and regenerate it without manually changing the items in the Layout Editor.

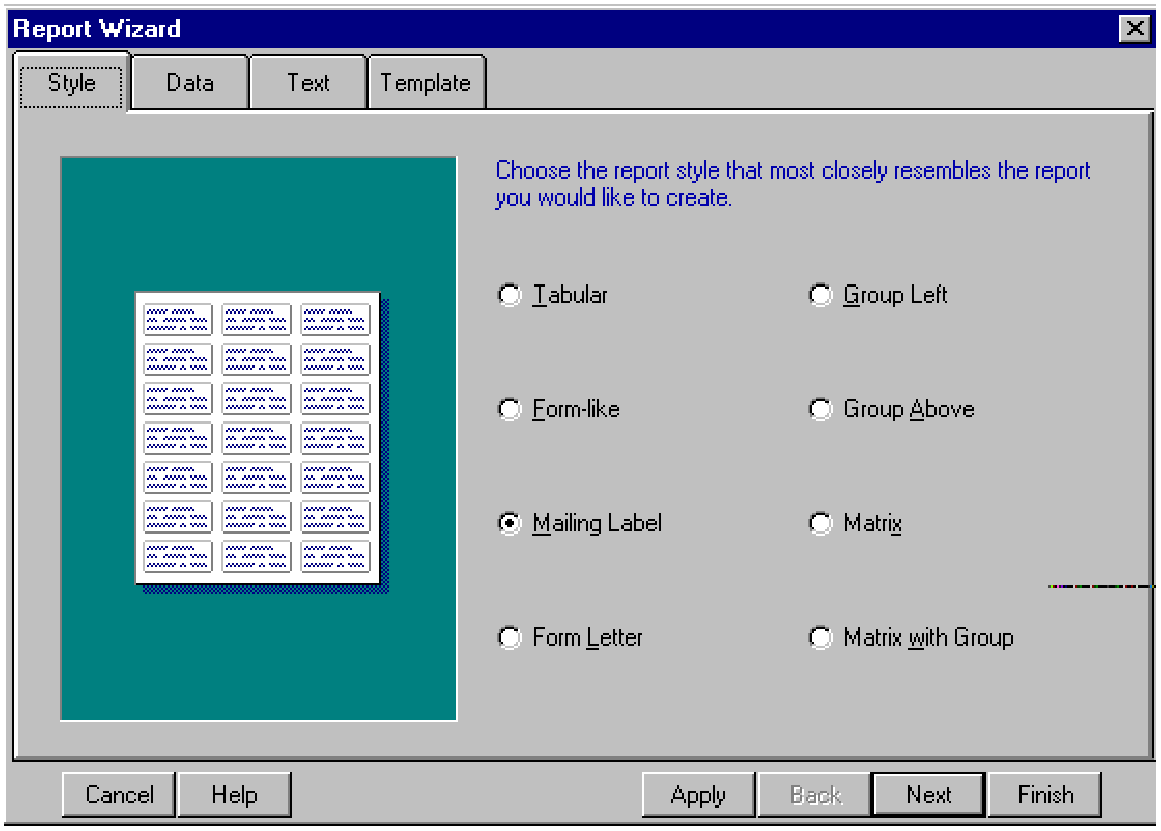
Release 2 adds a long-anticipated feature: tabbed canvases, as this complete application shows. Tab controls are all the rage in user interface design now, and with release 2, you can organize your data blocks in a series of tabs. This lets you expose only a few fields to the user at once, resulting in a much easier to use interface.



The Report Wizard, the Live Previewer, and the Report Object

The reentrant Report Wizard lets you create several different varieties of report in the Report Builder:

- *Tabular:* A report laid out in columns and rows
- *Group Left:* A tabular report with one or more break groups on the left
- *Group Above:* A tabular report that subdivides the rows into groups down the page
- *Form-like:* A report laid out in fields like a form
- *Form Letter:* A textual layout with embedded fields that you can use to produce many copies of the same text with different field values
- *Mailing Label:* A repeating set of field groups
- *Matrix:* A crosstab report
- *Matrix with Group:*A matrix report with a break group



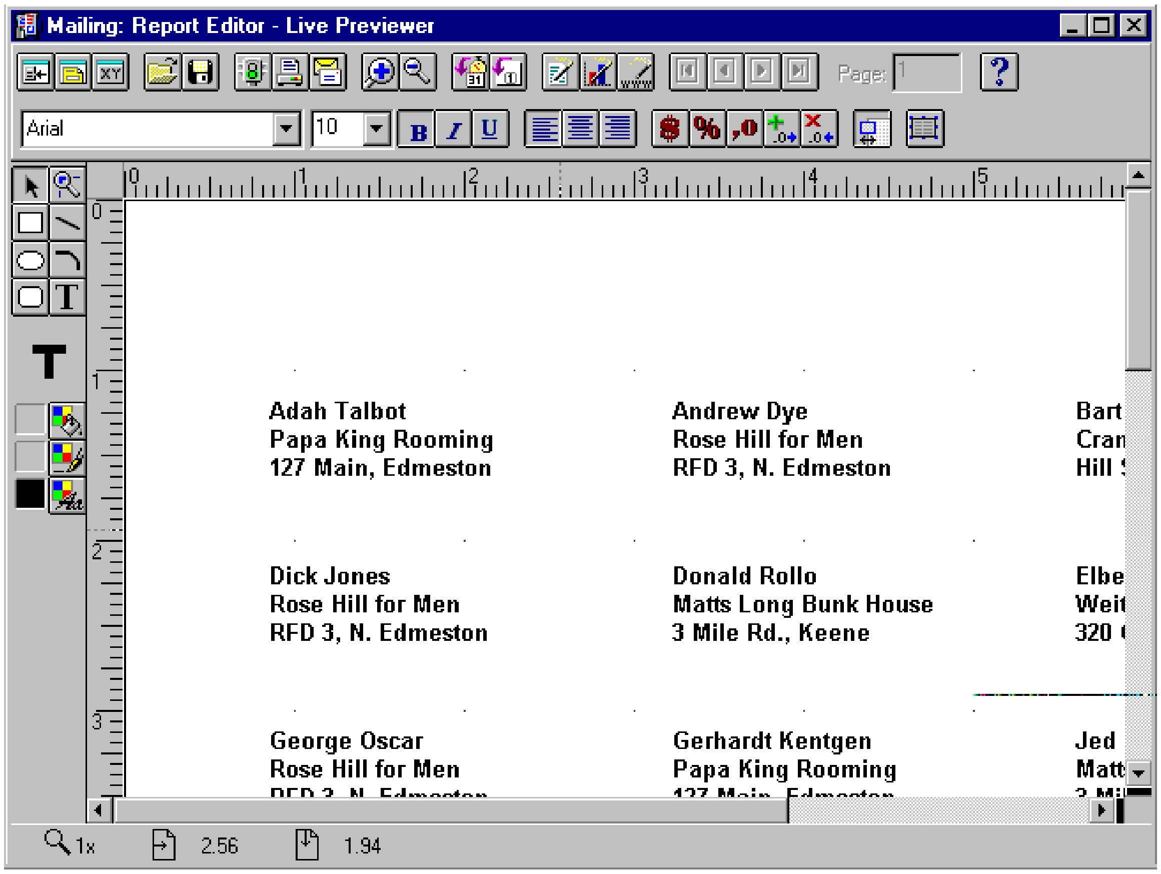
Choose the report style that most closely resembles the report you would like to create.

( ) Tabular ( ) Group Left
( ) Form-like ( ) Group Above
(●) Mailing Label ( ) Matrix
( ) Form Letter ( ) Matrix with Group

The Report Wizard screens take you through the entire process of building a report:

- Choosing the report style
- Creating the data model with one or more SQL queries, possibly by using the Query Builder
- Selecting the data model items to display as fields
- Specifying the labels and sizes of the fields
- Selecting the data model items to group by
- Selecting the data model items to total as grouped or grand totals and the aggregate function to use
- Specifying the text and sequence of items in a mailing label or form letter
- Specifying the rows, columns, and cells of the matrix
- Specifying the template to use to generate a report in a standard format

Once you've built the report, the Report Builder runs the report and displays the result in the Live Previewer. You can use the Previewer to edit most of the layout aspects of the report—sizing and repositioning fields, modifying the formatting and layout characteristics of a field, and so on. You use the Layout Editor to add new fields or to modify the frames that surround fields, and you use the Data Model Editor to modify the data model. You can always run the reentrant Report Wizard again to modify the generated report.



Adah Talbot
Papa King Rooming
127 Main, Edmeston

Andrew Dye
Rose Hill for Men
RFD 3, N. Edmeston

Bart
Cran
Hill

Dick Jones
Rose Hill for Men
RFD 3, N. Edmeston

Donald Rollo
Matts Long Bunk House
3 Mile Rd., Keene

Elbe
Weit
320

George Oscar
Rose Hill for Men

Gerhardt Kentgen
Papa King Rooming

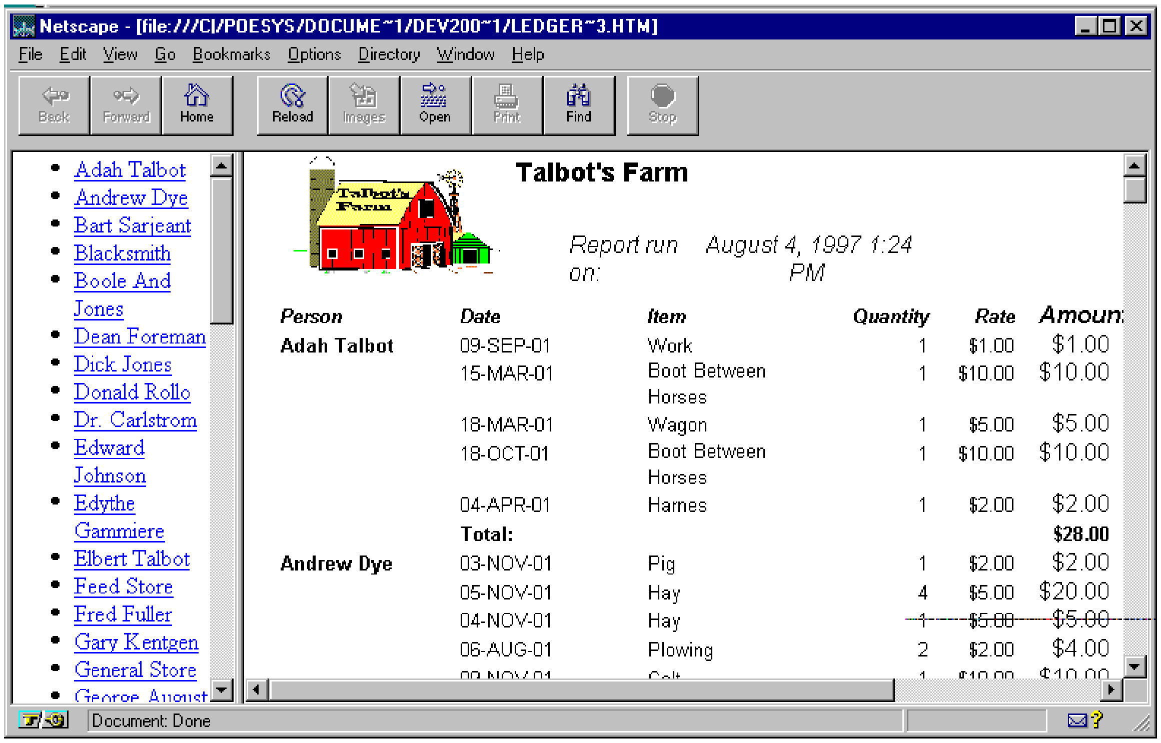
Jed
Matt

Release 2 of Developer/2000 adds the report object to the set of objects in a form module. The report object links the form module to a report module you produce with the Report Builder. When you create the report object, the Form Builder prompts you to either create a new form or to link to one you've already created in a file on disk. To create a new report, the Form Builder runs the Report Builder, which in turn runs the Report Wizard to create the report. You can also specify whether to get the data for the report from the database or from a block in your form application. You run the report by calling the built-in procedure Run_Report_Object in a menu item or trigger with the name of the report object.

The Web Wizard

The Report Builder has a reentrant Web Wizard that lets you structure your report objects for display in one of two World Wide Web formats, HTML or Adobe Acrobat. HTML is a form of markup language that you use to build static pages for display in Web browsers. Adobe Acrobat uses the PostScript-derived Page Definition Format (PDF) to encode documents. Using the Acrobat Reader or the Acrobat plug-in for your Web browser, you can display the document exactly as if you were looking at a printed copy of it.

Producing an HTML report is easy using the Web Wizard. You can create bookmarks from break groups in your report. This provides a handy hyper-linked table of contents to the side of the document in the browser.


Netscape - [file:///C|/POESYS/DOCUME~1/DEV200~1/LEDGER~3.HTM]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Document: Done

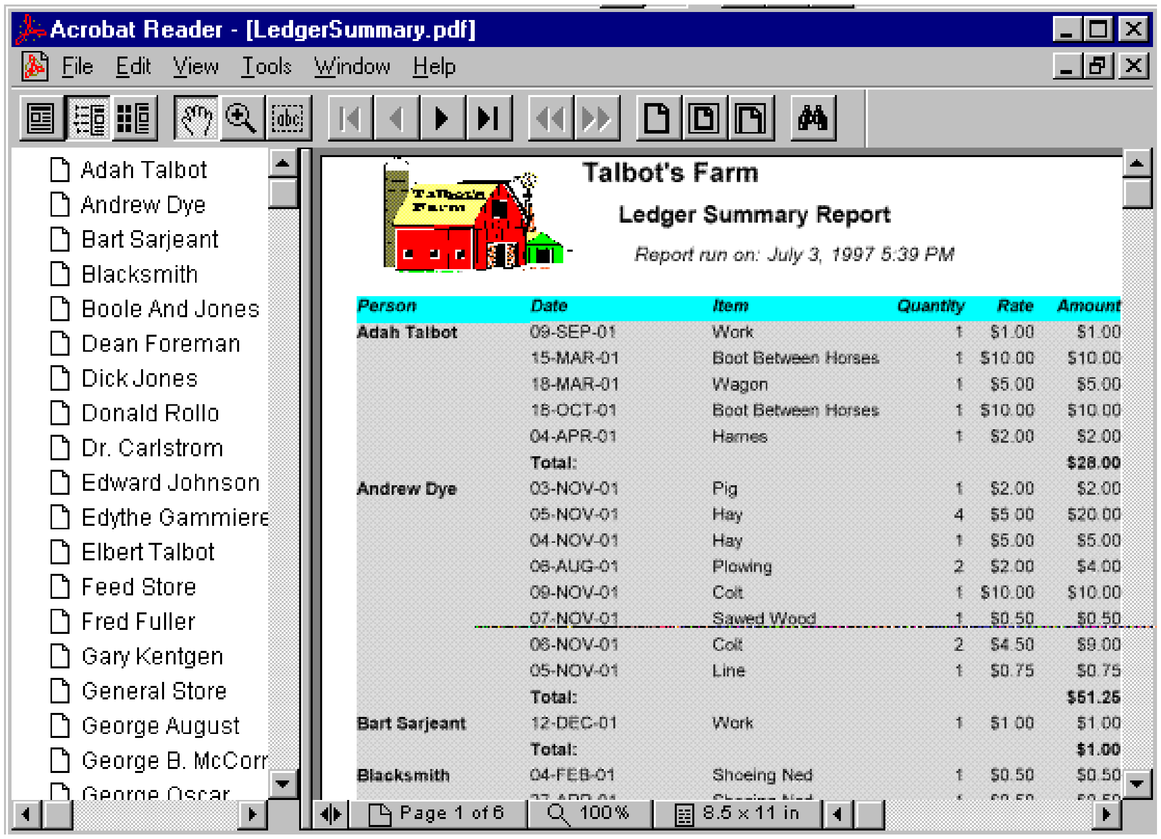
- Adah Talbot
- Andrew Dye
- Bart Sarjeant
- Blacksmith
- Boole And Jones
- Dean Foreman
- Dick Jones
- Donald Rollo
- Dr. Carlstrom
- Edward Johnson
- Edythe Gammiere
- Elbert Talbot
- Feed Store
- Fred Fuller
- Gary Kentgen
- General Store

Talbot's Farm

Report run on: *August 4, 1997 1:24 PM*

Person	*Date*	*Item*	*Quantity*	*Rate*	*Amoun*
Adah Talbot	09-SEP-01	Work	1	$1.00	$1.00
	15-MAR-01	Boot Between Horses	1	$10.00	$10.00
	18-MAR-01	Wagon	1	$5.00	$5.00
	18-OCT-01	Boot Between Horses	1	$10.00	$10.00
	04-APR-01	Harnes	1	$2.00	$2.00
	Total:				**$28.00**
Andrew Dye	03-NOV-01	Pig	1	$2.00	$2.00
	05-NOV-01	Hay	4	$5.00	$20.00
	04-NOV-01	Hay	1	$5.00	$5.00
	06-AUG-01	Plowing	2	$2.00	$4.00

Producing an Adobe Acrobat document is just as easy and gives you access to the full array of graphic effects available with Reports.


Acrobat Reader - [LedgerSummary.pdf]
File Edit View Tools Window Help
Page 1 of 6
100%
8.5 x 11 in

- Adah Talbot
- Andrew Dye
- Bart Sarjeant
- Blacksmith
- Boole And Jones
- Dean Foreman
- Dick Jones
- Donald Rollo
- Dr. Carlstrom
- Edward Johnson
- Edythe Gammiere
- Elbert Talbot
- Feed Store
- Fred Fuller
- Gary Kentgen
- General Store
- George August
- George B. McCorr

Talbot's Farm

Ledger Summary Report

Report run on: July 3, 1997 5:39 PM

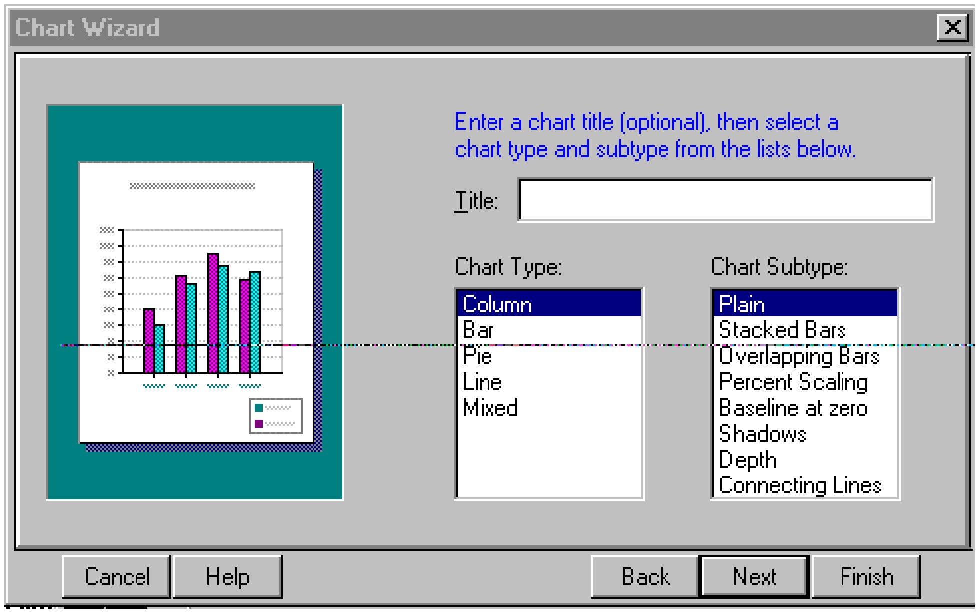
Person	*Date*	*Item*	*Quantity*	*Rate*	*Amount*
Adah Talbot	09-SEP-01	Work	1	$1.00	$1.00
	15-MAR-01	Boot Between Horses	1	$10.00	$10.00
	18-MAR-01	Wagon	1	$5.00	$5.00
	18-OCT-01	Boot Between Horses	1	$10.00	$10.00
	04-APR-01	Harnes	1	$2.00	$2.00
	Total:				**$28.00**
Andrew Dye	03-NOV-01	Pig	1	$2.00	$2.00
	05-NOV-01	Hay	4	$5.00	$20.00
	04-NOV-01	Hay	1	$5.00	$5.00
	06-AUG-01	Plowing	2	$2.00	$4.00
	09-NOV-01	Colt	1	$10.00	$10.00
	07-NOV-01	Sawed Wood	1	$0.50	$0.50
	06-NOV-01	Colt	2	$4.50	$9.00
	05-NOV-01	Line	1	$0.75	$0.75
	Total:				**$51.25**
Bart Sarjeant	12-DEC-01	Work	1	$1.00	$1.00
	Total:				**$1.00**
Blacksmith	04-FEB-01	Shoeing Ned	1	$0.50	$0.50

The Chart Wizard

The Graphics Builder uses the Chart Genie for creating charts. Both the Form Builder and the Report Builder have a reentrant Chart Wizard for creating charts that lets you create the chart and embed it in the right place in your form or report. When you create a chart item in a form or report, the Builder asks whether to use a stored display module you've created with the Graphics component or to create one using the Chart Wizard.

The Chart Wizard lets you set up or modify the basic characteristics of a chart:

- The type of chart (line, bar, pie, and so on) and its subtype
- The data to use on the X axis
- The data to use on the Y axis
- The file in which to store the display module you create



Enter a chart title (optional), then select a chart type and subtype from the lists below.

Title:

Chart Type: Column (selected), Bar, Pie, Line, Mixed

Chart Subtype: Plain (selected), Stacked Bars, Overlapping Bars, Percent Scaling, Baseline at zero, Shadows, Depth, Connecting Lines

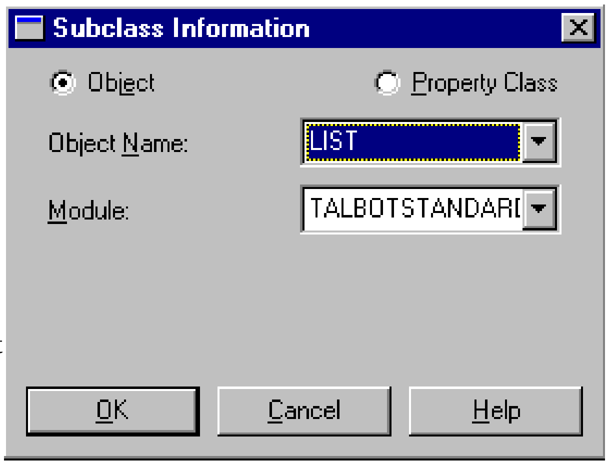
When you finish the creation process, you have a chart object in the form or report module that links the form or report to the display module. In most cases, the integration information controls when the chart gets initialized and displays information, but you can use built-in subprograms to manage the display as well.

Object-Based Reuse

Subclassing

Release 2 introduces subclassing, the ability to inherit some or all the properties of an object of the same type. For example, you can base a text item on another text item, or a canvas on another canvas. The *subclass* is the object making the reference, and the *parent class* is the object to which the subclass refers and from which it inherits properties.

The Subclass Information property is common to most objects in a form module. Clicking on the More icon for this property displays the Subclass Information dialog box, which lets you choose between subclassing an object and subclassing a property class through a radio button.



[✓ Object] Property Class

Object Name: LIST

Module: TALBOTSTANDARI

The drop-down list of objects contains only those objects of the same type as the subclass. If you clicked on the Subclass Information property for a list item, you see only list item objects to subclass. If you clicked on a canvas, you see only canvases.

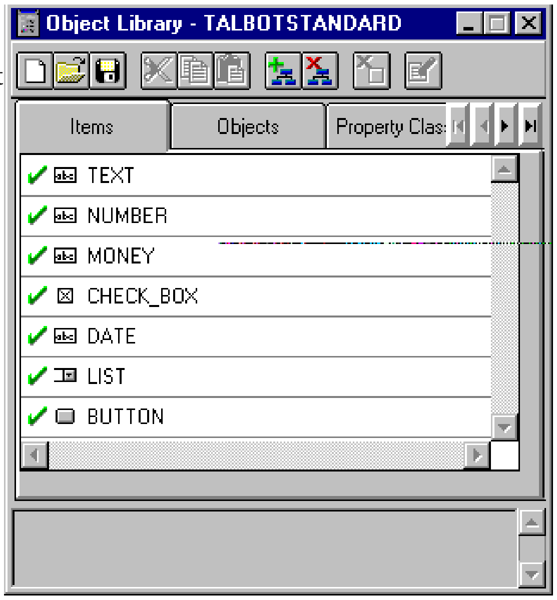
There are four kinds of properties in the subclass:

- *Default:* the Form Builder sets the property from its internal defaults; the Property Palette displays a small circle next to the property name
- *Set:* the property comes from you or the Form Builder assigning a specific value (for example, the Layout Wizard can set the Height and Width properties of most objects, overriding any inherited property value); the Property Palette displays a small rectangle next to the property name
- *Inherited:* the property comes from the same property in the subclassed object or its parent objects; the Property Palette displays a right-angled arrow next to the property name
- *Overridden:* the property, originally inherited from a parent class, now has a value you or the Form Builder assigned, breaking the inheritance link; the Property Inspector displays the Inherited arrow with a red X on it indicating a broken link

The subclass inherits only properties you have explicitly set in the parent class or in its parents. Any other property gets the Form Builder default (a circle rather than an inheritance arrow). As with property classes, you can build entire hierarchies of parent classes, adding and overriding properties down the hierarchy.

The Object Library and SmartClasses

Release 2 does not stop at subclassing but adds a new feature, the object library, to make subclassing even more useful. An object library is a container for form objects. You can drag any number of individual objects such as blocks, items, canvases, windows, or property classes into a library for reuse in other modules.



Like the PL/SQL library, the object library corresponds to a separate file (or database object, if you choose to store it on the database server). When you double-click on the new object library icon, Developer/2000 opens the Object Library editor and displays two empty tab pages. It creates these tab page objects under the Library Tabs subheading under the new library in the Object Navigator.

Organizing your object libraries is totally up to you. There are no constraints on putting objects on any tab page you choose.

You need to decide what the purpose of your library is. For example, Talbot's Farm uses a standard object library that all its applications share. This makes items, canvases, windows, and other objects consistent in formatting and basic property settings. Such a standard library contains a broad array of small, reusable objects organized into type-based tab pages. For example, you could have one tab page per type of object (one for blocks, one for items, one for property classes, and so on). Another kind of library might package a block and its items, or several blocks sharing a master-detail relationship, for reuse in several applications.

To copy objects into the library, just drag them onto the open tab page. This copies the object onto the open tab page. The lower box in the Object Library Editor displays the Comment property of the object. You can set the comment in the original object through the standard Property Inspector, or you can use the Comment tool in the Object Library Editor to create a comment in the copy in the Object Library.

To use objects in an object library, you open the library in the usual way using the File Open tool, then drag and drop the objects into your forms. You get the same alert asking whether to copy or subclass the object that you do dragging objects between forms. Subclassing an object from an object library lets you inherit any changes to the library over time. The Form Builder remembers what object library you last opened and opens that library every time you start up the Builder.

Alternatively, you can create objects in your form, then subclass them from the object library. You can do this in two ways.

First, you can use the Subclass Information property of the new object to select a parent object from the object library. For example, if you created a text item in a block and wanted to make it a subclass of the NUMBER object in the TalbotStandard.OLB object library, you click on the property's More button and see the Subclass Information dialog box. You set the radio button to Object, choose the Standard module (that is, the object library), and then choose the NUMBER object from that library.

A second way to subclass from a library uses the new SmartClass feature. A SmartClass is a class in a library that you choose to make available through a pop-up menu. You can make any object in your library a SmartClass. Open the object library and select the object you wish to make into a SmartClass in the Object Library Editor. Now choose the Object→SmartClass menu item from the main menu of the Form Builder. You see a green check mark appear next to the object in the Object Library Editor.

Select an object in the Object Navigator of the same type as the SmartClass object. Right-click on the object and choose the SmartClasses menu item. You now see a submenu that lists all the SmartClasses in open libraries of the same type of object as the object on which you right-clicked the mouse. This example shows three text item objects in the object library, a general text item, a number subclass, and a date subclass.

Choosing one of the SmartClasses subclasses the object from the SmartClass in the object library. This arrangement, like SmartTriggers, gives you a context-dependent menu that greatly speeds up subclassing from your object library.

A Guided Tour of Developer/2000

To show you how this all works, this section develops a very simple example of a Human Resources application that uses tabbed canvases and runs a simple mailing-label report.

Building an Integrated Human Resources Application

A small farming business, Talbot's Farm, needs to build a human resources application. The application must present each worker's name and address and must list the skills and abilities of each worker.

The Oracle7 CREATE SCHEMA statement for the Talbot user looks like this:

```
CREATE SCHEMA AUTHORIZATION Talbot
CREATE TABLE Lodging (
Lodging       VARCHAR(15) PRIMARY KEY, /* short name for lodging */
LongName      VARCHAR(40),             /* complete name */
Manager       VARCHAR(25),             /* manager's name */
Address       VARCHAR(30)              /* address of the lodging */
)
CREATE TABLE Skill (
Skill         VARCHAR(25) PRIMARY KEY, /* name of a capability */
Description   VARCHAR(80)              /* description of the skill */
)
CREATE TABLE Person (
Name          VARCHAR(25) PRIMARY KEY,       /* worker's name */
Age           INTEGER,                       /* age in years */
Lodging       VARCHAR(15) REFERENCES Lodging /* ref to Lodging */
)
```

```
CREATE TABLE WorkerHasSkill (
Name          VARCHAR(25) REFERENCES Person, /* worker's name */
Skill         VARCHAR(25) REFERENCES Skill,  /* capability name */
Ability       VARCHAR(15),                   /* proficiency */
PRIMARY KEY (Name, Skill)
)
;
```

To create the schema, find the files TABLES.SQL and INSERTS.SQL in the zip file, **‘revguide.zip’,** accompanying this paper. You must first create a user for the schema. For example, in SQL*Plus, while logged on as a DBA user such as SYSTEM, you can create the Talbot user with this command:

GRANT CONNECT, RESOURCE TO Talbot IDENTIFIED BY george;

This creates the Talbot user, grants necessary system privileges, and sets the password to "george". After creating the user, logon to the user in SQL*Plus and run the TABLES script:

```
@tables
```

After successfully creating the schema, you can load the example data:

```
@inserts
```

Building the Tabbed Form

To build the form, you start the Form Builder and enter the Data Block Wizard to build your first block. To build the main part of the HR application, you go through these steps:

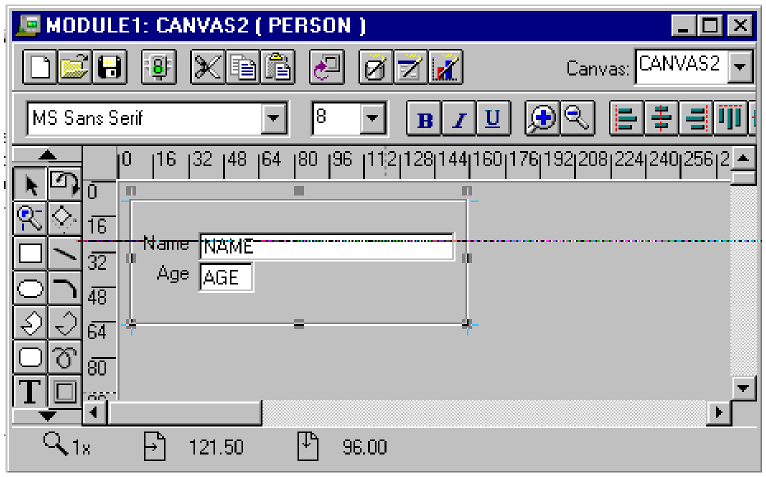
1. Create the Person master block to display the name and age of the worker.
2. Create the Lodging detail block to display the address of the worker in a tab page and create the master-detail relationship manually.
3. Create the WorkerHasSkill detail block to display in a tab page the skills that a worker possesses, and their ability with respect to each skill. Use the master-detail screen in the Wizard to create the relationship to Person.

   Note: The complete HR application is available with the code that accompanies this paper (HR.FMB and HR.FMX).

Building the Person Block

To create the Person block, you click on the Browse button in the data block screen of the Wizard. The Builder prompts you for an Oracle user name and password, then connects you to the database and displays all the tables and views of the user. In this case, you pick the Person table and click OK. You then choose which columns to include in the data block; in this case, you click on >> to select all the columns, including Lodging. Clicking on Finish then brings up the Layout Wizard.

In the canvas screen, select the <New canvas> choice to create a new canvas, the main content canvas for the application. In the display fields screen, select the Name and Age items for display, leaving the Lodging item out. You will use the Lodging item to establish the master-detail relationship later. Make the block a form block rather than a tabular one, displaying one worker at a time. Leave everything else as it is and click on Finish. You now see the Layout Editor with the new canvas and its items.



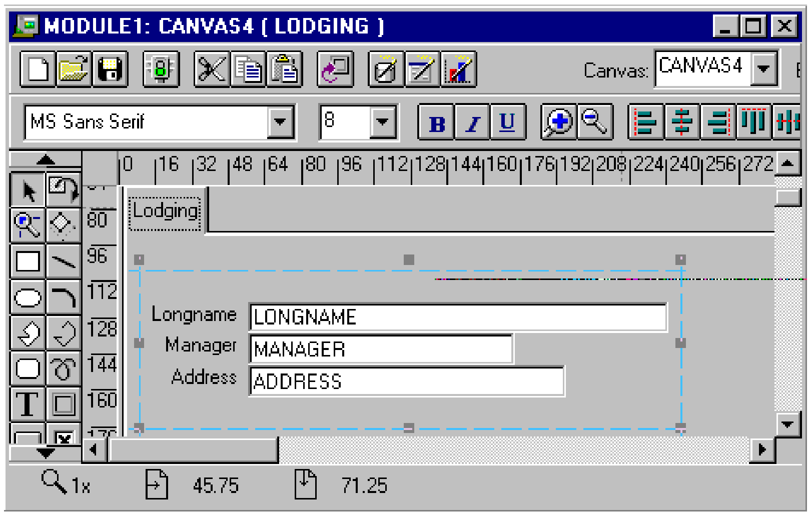
In this case, you don't want the frame to show, so select it and use the Line Color tool in the Layout Editor to turn off the line (the No Line choice at the bottom of the pop-up color menu).

Building the Lodging Block

To create the Lodging block, you select the Person block or the Data Block heading and click on the Create tool in the tool palette to the side of the Object Navigator. You then choose to create the block using the Data Block Wizard. You go through the table selection process again, this time choosing the Lodging table and all its columns.

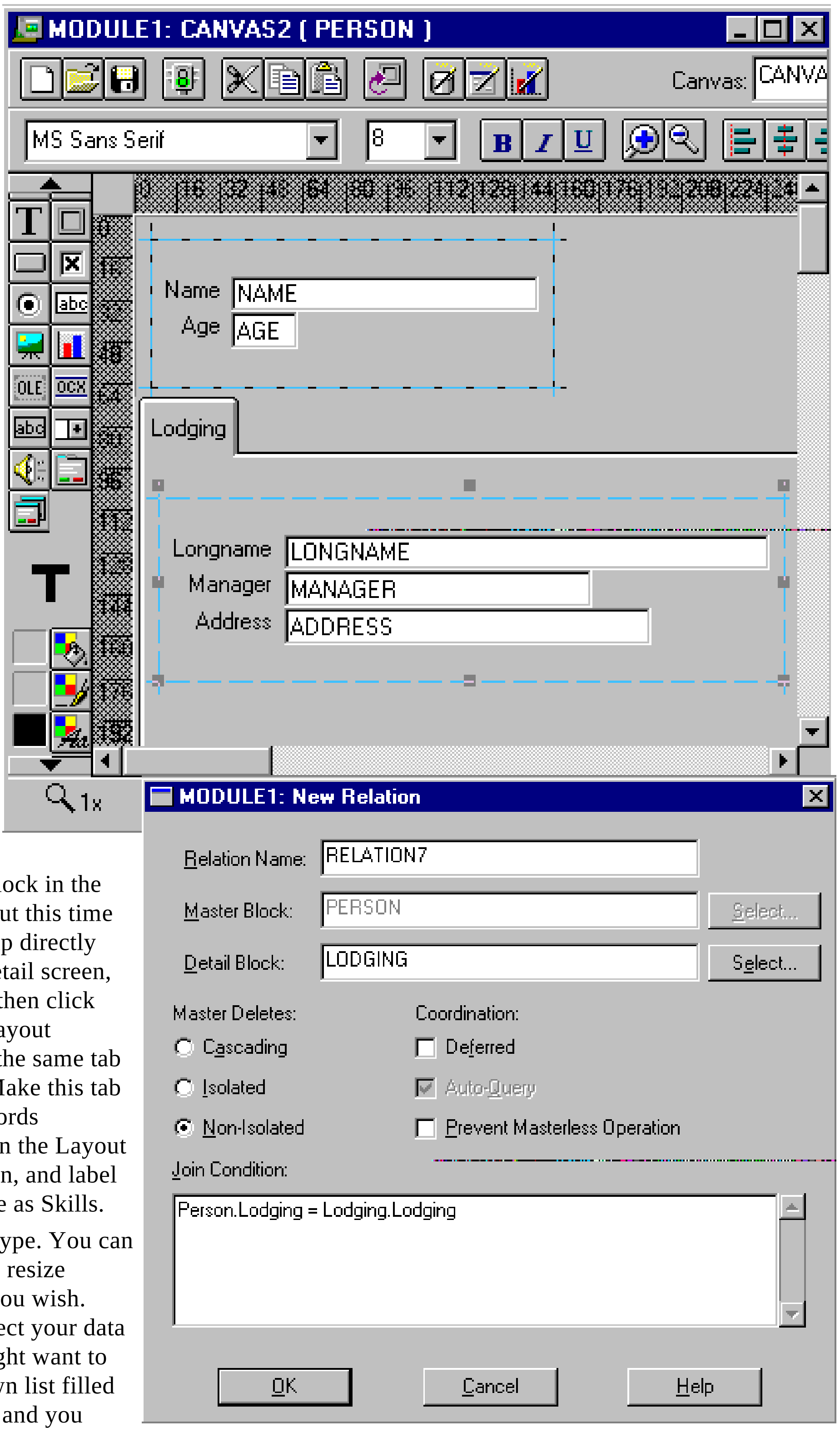
You cannot use the master-detail screen at this point because the Wizard does not recognize the relationship between Person and Lodging due to its unusual nature (many-to-one instead of one-to-many). You'll establish the relationship later using a Relation object.

In the Layout Wizard, select a new tab page as the canvas, then choose to display the LongName, Manager, and Address items (not Lodging). Create the tab page as a form, since there is only going to be a single lodging for each worker. Click on finish to create the tab page. Again use the Layout Editor to make the frame invisible, then set the tab page label by editing it in the Property Palette. Select the tab page object in the Layout Editor by clicking on it, then choose Tools→Property Palette. Find the Label property and change it to Lodging. The result looks like this in the Layout Editor:



You now must position the tab page correctly with respect to the content page for the Person data block. In the Layout Editor with the Person block canvas displayed, choose the View→Stacked Views menu item, then choose the new canvas from the resulting dialog. You will now see the tab page displayed on top of the Person canvas. You can now drag it and resize it to fit as you wish.

Now, establish the master-detail relationship. Create a new relation object under the Person block using the Create tool after selecting the Relations heading. In the New Relation dialog, select the Lodging block as the detail block, then enter the relationship between the Lodging.Lodging item and the Person.Lodging item. Clicking on OK creates the relationship and the several triggers that implement it.



MODULE1: CANVAS2 (PERSON)

Canvas: CANVA

MS Sans Serif 8

Name NAME

Age AGE

Lodging

Longname LONGNAME

Manager MANAGER

Address ADDRESS

MODULE1: New Relation

Relation Name: RELATION7

Master Block: PERSON

Detail Block: LODGING

Master Deletes: ( ) Cascading ( ) Isolated [✓ Non-Isolated]

Coordination: ☐ Deferred ☑ Auto-Query ☐ Prevent Masterless Operation

Join Condition:

Person.Lodging = Lodging.Lodging

Building the Skill Block

Now create the WorkerHasSkill block in the same way as the Lodging block, but this time create the master-detail relationship directly using the Wizard. In the master-detail screen, click on Create Relationship, and then click OK to create the relation. In the Layout Wizard, create a new tab page on the same tab canvas as the Lodging tab page. Make this tab page a tabular one with 5 skill records showing, and give it a scroll bar. In the Layout Editor, turn off the frame line again, and label the tab page in the Property Palette as Skills.

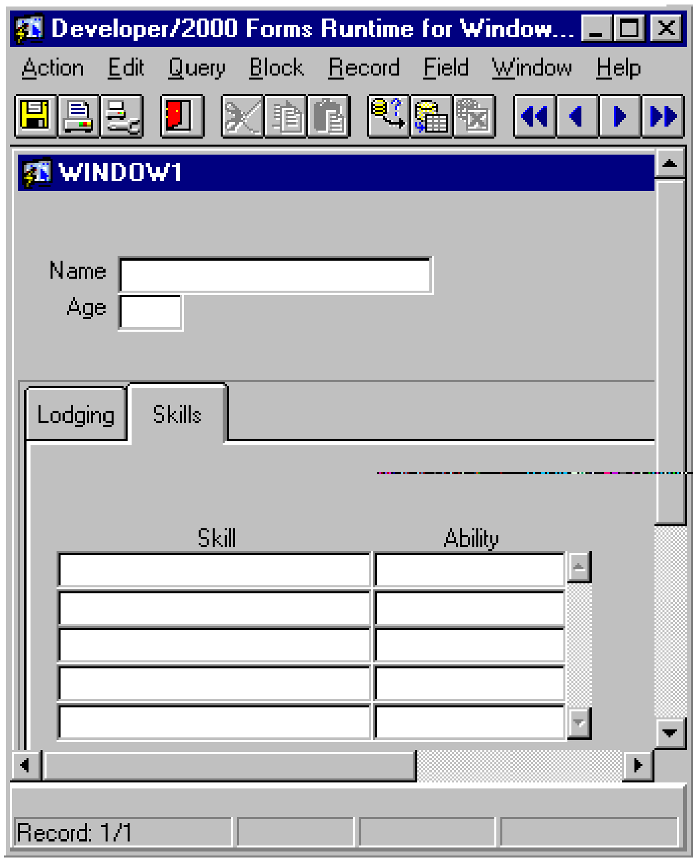
You now have a basic form prototype. You can rearrange the items on their pages, resize them, and set up the canvases as you wish. You can also edit the items to reflect your data entry needs. For example, you might want to set up the Skill item as a drop-down list filled in by a query from the Skill table, and you might want to add in a way to set the Lodging for the worker, such as a button that displays a list of available lodgings in a dialog. If you just compile and run the application as it is, it looks like this:

Adding a Report

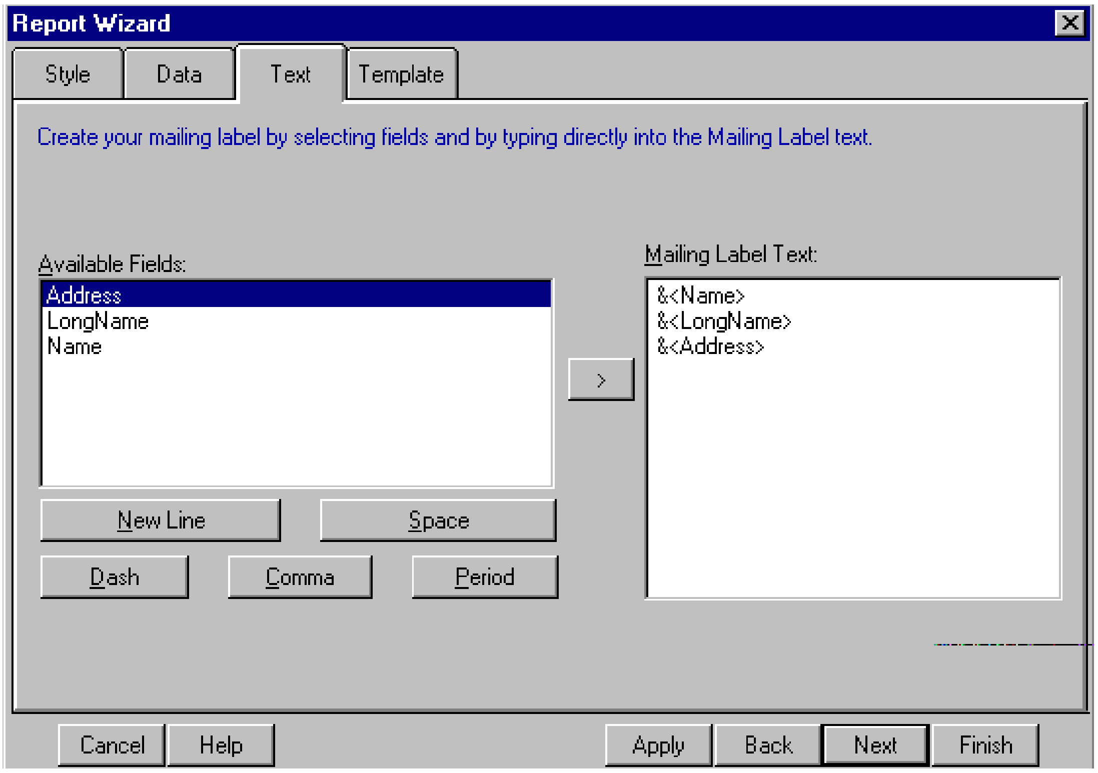
The HR department at Talbot's Farm needs to produce mailing labels for sending materials to all workers. Talbot's can use this report to generate labels for paychecks or other mailings, for example. A mailing-label report prints its repeating records without any headings in identically spaced chunks down a page. When you print the report, you generally do so on special adhesive mailing-label paper from which you peel off the labels after printing.

First, create a new report using the Report Wizard. The query for the report is slightly complex because it involves two joined tables instead of a single one. This gives you the name from the Person table and the LongName and Address of the lodging for the person from the Lodging table. The SQL also orders the result by the person's name.

```
SELECT Name, LongName, Address
  FROM Person, Lodging
 WHERE Person.Lodging =
Lodging.Lodging
 ORDER BY 1
```


Developer/2000 Forms Runtime for Window...
Action Edit Query Block Record Field Window Help
WINDOW1
Name
Age
Lodging
Skills
Skill
Ability
Record: 1/1

The next screen has special features for mailing label construction. This Text screen lets you position the fields in a prototype mailing label.



Create your mailing label by selecting fields and by typing directly into the Mailing Label text.

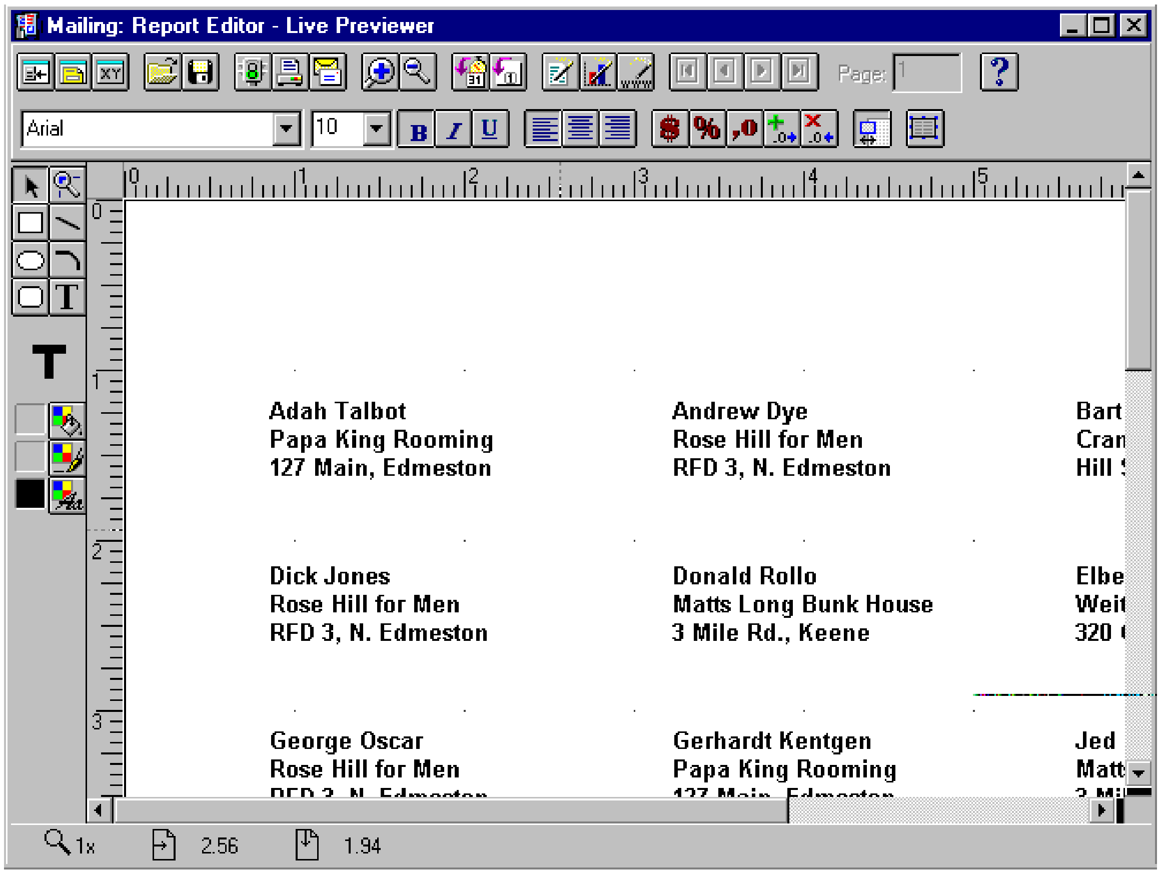
```
&<Name>
&<LongName>
&<Address>
```

It provides special buttons for the various punctuation marks that usually separate items in a mailing label (new lines, spaces, commas, dashes, and periods). To create the Talbot mailing label, for example, you select Name, click on the > button, click on the New Line button, and repeat for the Long Name and Address fields. The only other available screen in the wizard is the template screen. If you have a template set up for your mailing label reports, you can use it; otherwise, use the Draft template or no template.

You can create your own custom menu and create a menu item in it that runs the report, or you can create a button in a window or on a toolbar that runs the report. First, you create a report object in the Form Builder and link the new report (LABELS.RDF) to the object. If you want to print the labels, set the Destination Type property of the report object to Printer; otherwise, set it to Preview. You then code a call to Run_Report_Object in the menu item code or the trigger body. The PL/SQL code looks like this:

```
Run_Report_Object('LABELS');
```

When you choose the menu item or click on the button, the code executes and the report runs:



Adah Talbot
Papa King Rooming
127 Main, Edmeston

Andrew Dye
Rose Hill for Men
RFD 3, N. Edmeston

Bart
Cran
Hill

Dick Jones
Rose Hill for Men
RFD 3, N. Edmeston

Donald Rollo
Matts Long Bunk House
3 Mile Rd., Keene

Elbe
Weit
320

George Oscar
Rose Hill for Men

Gerhardt Kentgen
Papa King Rooming

Jed
Matt

References

Robert J. Muller's *The Developer/2000 Handbook, Second Edition* published by Oracle Press in 1997 is the only comprehensive book on Developer/2000 release 2. Some of the examples and material from that book appear in this Reviewer's Guide by permission of Oracle Press.

Steve Illingworth's article "Getting Started: NCA and Oracle Tools," in *Oracle Magazine* (July/August 1997, pages 79-84) gives a useful overview of the Oracle tools such as Developer/2000 in relation to the World Wide Web and Oracle's NCA.