

Oracle Designer/2000 Visual Basic Generator Technical Overview

VERSION 1.0: OCTOBER, 1995

Part C10428

October 1995

Author: Mike Gwyer

Contributors:

Copyright © 1995 Oracle Corporation.

All rights reserved. Printed in the U.S.A.

This document is provided for informational purposes only and the information herein is subject to change without notice. Please report any errors herein to Oracle Corporation. Oracle Corporation does not provide any warrants covering and specifically disclaims any liability in connection with this document.

Table of Contents

1. INTRODUCTION.....	1
2. COMPONENTS OF A GENERATED APPLICATION.....	3
2.1 <i>Projects, MAKs and .EXEs</i>	3
2.2 <i>Zones and Items</i>	3
2.3 <i>Validation and Application logic</i>	8
3. GENERATING APPLICATIONS USING THE VISUAL BASIC GENERATOR.....	10
3.1 <i>The database design</i>	10
3.2 <i>The module design</i>	11
3.3 <i>User Preferences</i>	13
3.4 <i>Templates</i>	16
3.5 <i>Visual Basic Library Forms and Code</i>	17
3.6 <i>Incorporating Third-Party Controls</i>	17
4.0 SUMMARY.....	19

1. Introduction

Microsoft's Visual Basic has become a widely used development tool for developers building client-server applications. This wide use has happened despite a number of shortcomings that make the building of scaleable client-server applications in Visual Basic difficult.

Unlike development tools like Oracle Forms 4.5, Visual Basic has limited in-built database integration and while products like Oracle Objects for OLE make database access easier, there is still a requirement for a lot of developer code to integrate Visual Basic Forms with the underlying server.

Another problem with building applications using Visual Basic is the lack of support for teamworking. This problem applies to many of the GUI tools available today, and whilst developers enjoy the flexibility and power available to them with these tools, the management of these developments is very difficult. The definition of site standards for look and feel is a necessity and the enforcement of these standards can be a major issue when multiple developers are building parts of an application that need to be integrated. A less attractive aspect of Visual Basic development is the lack of design documentation - it may be quick and easy to build applications, but this lack of documentation makes these applications slow and difficult to maintain. The limited scope of this documentation makes sharing design information throughout development teams very difficult. The modelling and generation of Visual Basic applications from Designer/2000 maintains comprehensive design documentation of the application for impact analysis, as well as enforcing a common look and feel based on a series of UI rules stored as User Preferences within the Repository.

The Oracle Designer/2000 Visual Basic Generator brings all the benefits of modelling and generation to those users building 2nd generation client-server applications in Visual Basic, in the same way that the Forms 4.5 and Reports 2.5 Generators empower those sites building applications using Oracle Developer/2000.

The Oracle Designer/2000 Visual Basic Generator is a program that generates fully-functional, data-oriented Visual Basic applications, based on module and database design specifications recorded in the Oracle Designer/2000 Repository. During generation, the rules that define the required look and feel are read from the repository, and an application is built that will share an interface style with other components generated by Designer/2000. Each generated application comprises one or more windows, each of which can contain a number of zones through which data can be queried or manipulated. These generated applications can be loaded into the Visual Basic design environment from where runtime executables can be built, or further refinements can be made.

The generated applications use the Data Access Objects of Oracle Objects for OLE to provide tight integration with an Oracle7 server.

2. Components of a generated application.

2.1 Projects, .MAKs and .EXEs

The generation of Visual Basic forms is driven from module definitions defined graphically through Designer/2000. In Designer/2000 terminology, a module is a set of forms that allow the user to perform a specific task, such as Order Entry or Query Customer Details.

The Visual Basic Generator allows you to build projects containing one or more modules, with full navigational support between these modules. At generate time you can choose to create a new application or add to an existing one. The generator produces the .MAK file used by the Visual Basic Project to control which forms (.FRMs) make up this application, as well as generating the forms which make up the windows and zones. The .FRM files contain the generated Visual Basic code that comprises the calls to Oracle Objects for OLE to provide data access and manipulation.

It is possible to load these generated projects into Visual Basic and then, if necessary, add additional application logic prior to building a .EXE for the entire application.

2.2 Zones and Items

The Visual Basic Generator builds application consisting of a number of forms or windows, each displaying one or more zones.

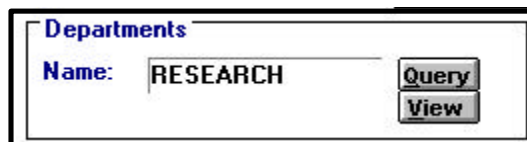
Each individual zone is laid out according to one of four zone layout styles, and these styles determine the overall layout of a particular zone in the generated Visual Basic application. It is possible to use a mixture of these zone styles within a single application.

The four zone layout styles are:

- Context
- Form
- List
- List/Form

Context

A context zone displays a query-only view of a single record. The record to be shown is chosen from a list of values (LOV) dialog box which is automatically called on startup, or may be called by hitting the Query button. As a context zone contains no enterable fields, the Visual Basic Generator does not generate any hot-key inter-field navigation.



Context zone styles do not allow data to be edited directly on the main window. For each such zone, a property sheet dialog is generated, which allows new rows to be created and existing records to be updated or viewed. The dialog is displayed when the user clicks on the zone's View or New buttons. Property sheet dialogs also display a set of controls that are mapped to a single row of data . The property sheet dialog below shows the properties of a row from the

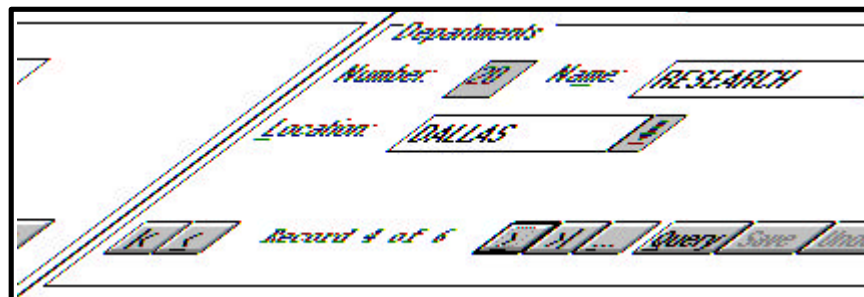
EMPLOYEES table. Each enterable field in a the property sheet has hot-key navigation provided by the Visual Basic Generator. In the example shown below Alt+n will navigate the User directly to the Number field, while Alt+m will go to the Name field.



The screenshot shows a window titled "Employees (BLAKE)". It contains several input fields: "Number" with the value "7698", "Name" with "BLAKE", "Job" with "MANAGER", "Manager" with "KING" and a dropdown arrow, "Hire date" with "01/05/81", "Basic" with "2850", and "Comm" which is empty. At the bottom, there are four buttons: "OK", "Cancel", "Undo", and "Apply Now".

Form

A Form zone displays a single record, the properties of which can be updated or changed. A form zone is similar to the interface provided by a 'data control' in Visual Basic. Browse buttons allow movement between records or directly to a specific record. The Save and Undo buttons allow the User to commit any changes made to the database, or to rollback the most recent changes.

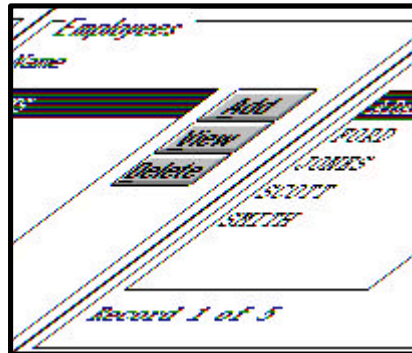


The screenshot shows a form zone for a table named "Departments". It has three fields: "Number" with the value "20", "Name" with "RESEARCH", and "Location" with "DALLAS". Below the fields are several navigation buttons: "Previous", "Next", "First", "Last", "Query", "Save", and "Undo". A status bar at the bottom indicates "Record 4 of 6".

Selecting the Query button in a form zone displays a query dialog which is a modal window used to allow the entry of search criteria to be applied when executing a query against a particular zone. They display a set of controls that are mapped to columns from that zone. You specify which columns should be displayed in a query dialog through the Module Data Diagrammer. If no columns are selected for a query dialog then the generated application would perform an unrestricted query.

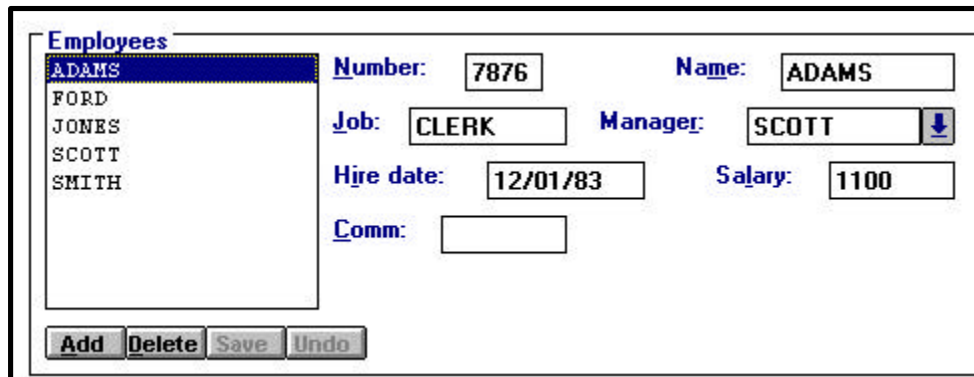
List

A listbox displays a set of queried records. As with context zones, a listbox is query-only, and a separate property sheet dialog is called via buttons to edit the record or add a new record. Selecting a row and hitting the Delete button will remove the record from the database. Foreign Key Constraint Cascade rules defined in the server can then control any secondary deletions that need to be performed (for example you can define that the deletion of a department should force the deletion of all the employees in that department).



List/Form

A List/Form zone is similar to the Form style, except that a listbox displays the set of queried records alongside a single-record view of the currently selected record. As the user navigates through the list of Employee names, the detailed properties displayed in the Form region change to reflect the currently selected record in the listbox. Again a series of buttons control the writing or cancellation of any pending changes through to the server. The example below shows a Listform zone based on the Employees table - the list region displays the name of the set of employees, while the form region shows all the properties of the currently selected employee. The definition of the Module in the Designer/2000 repository controls which columns should be shown in the list region.

A screenshot of a List/Form zone interface titled "Employees". On the left, a listbox contains the names of five employees: ADAMS, FORD, JONES, SCOTT, and SMITH. The name "ADAMS" is highlighted. To the right of the listbox, there are several form fields for the selected employee: "Number:" with the value "7876", "Name:" with the value "ADAMS", "Job:" with the value "CLERK", "Manager:" with the value "SCOTT" and a dropdown arrow, "Hire date:" with the value "12/01/83", "Salary:" with the value "1100", and "Comm:" with an empty field. At the bottom of the interface, there are four buttons: "Add", "Delete", "Save", and "Undo".

Control Groups and Stacking Areas

It is possible to group together certain fields within a zone, thus providing granular control over the layout of the generated application. The mechanism used for defining this grouping of data items on screen is the same as that used by the Oracle Forms Generator, thus ensuring consistency between the layout of both Visual Basic and Oracle Forms Applications.

The grouping of fields in Visual Basic applications is of particular interest when used in conjunction with stacking areas. A stacking area comprises a number of stacked frames, each of which may contain a control group or a detail zone. The Visual Basic Generator adds a radio group with a button for each of the stacked frames, whereby the user can navigate between the stacked frames selecting which group of items should be displayed. In the example below, the generated application always shows the Number, Name, Job, Manager and Hiredate of an employee, but the User also has access to the Salary Details, Phone Numbers and Salary History of each employee and may select which of these groups of information they wish to display at any one time by checking the appropriate radio button. Each of these frames may contain more properties of the employee record (e.g. Salary, Commission) or a detail zone showing records from another table that has a foreign key link to the Employee Record (e.g. their last 5 salaries and the dates they changed).

The screenshot shows an Oracle Forms application window titled "Employees". The form contains the following fields and controls:

- Number:** 7782
- Name:** CLARK
- Job:** MANAGER
- Manager:** KING (with a dropdown arrow)
- Hire date:** 09/06/81
- Radio Button Group:** Three radio buttons are present: "Salary Details" (unselected), "Phone Numbers" (selected), and "Salary History" (unselected).
- Home:** 021 234123
- Car:** (empty field)
- Office:** 043 897665
- Extension:** 243

At the bottom of the window, there is a navigation bar with the following elements:

- Navigation buttons: back, forward, search.
- Status bar: Record 2 of 8
- Action buttons: Query, Add, Delete, Save, Undo.

The prompts, datatypes and field sizes for all of the data items displayed in the zones are derived from the definitions held in the Designer/2000 Repository. It is possible to set 'standard' or default values for prompts and fields which will be used by the Forms, Visual Basic and Power Objects Generators when producing applications that display these items. The fonts and styles used for the displayed items are set through the template visual basic application. Hint text for all fields and buttons is generated from the hint text associated with the underlying column in the Designer/2000 repository. This ensures consistent hint text throughout the whole Visual Basic application, as well as consistency across any Oracle Forms 4.5 or Oracle Power Objects applications generated from the same definitions. The display of this hint text can be controlled at run time of the Visual Basic form through a number of options included by the Generator on the standard menu for all generated applications. The user can choose to have no hint text displayed, or to have the text appear after a short or long delay. The data manipulation buttons included on the generated form are derived from the allowed operations defined in the Designer/2000 repository for the module on

which the form is based - for example a form based on a module definition which did not allow the creation of data would not contain an Add button.

2.3 Validation and Application logic

Generated applications are designed to run against an Oracle7 database server, using the Data Access Objects of Oracle Objects for OLE.

For each zone in an application, a dynaset is created at form startup through which single-record Insert, Update, Delete, and Query operations are performed. After Insert or Update operations, the dynaset is automatically refreshed so that data derived or defaulted by server-side triggers is displayed to the user. If required, a second dynaset is created for querying associated lookup data.

Also, for each zone, a dynaset is created to fetch the zone's set of records. Here, only the primary key plus some descriptive data is fetched for each record; the full details are then queried upon navigation to a specific record. The descriptive data allows the user to navigate directly to the required record without having to browse records one at a time.

If any server errors are encountered, the number and text of the error are examined and, if anticipated, are converted to a more user-orientated message (definable in the design Repository). If the error is not anticipated, then a Server Error dialog is displayed.

The business rules and application logic can be defined within Designer/2000 and use all the features of an Oracle7 server. Primary Key, Unique Key, Foreign Key and Check Constraints can be defined in the server to enforce the business rules associated with the underlying data model. The Visual Basic applications built using the Designer/2000 generator can also intercept application errors returned from database triggers or from any PL/SQL packages, procedure and functions called from database triggers. This allows the developer to implement complex application logic in PL/SQL on the server, raising application errors containing meaningful error messages that will be presented to the end user.

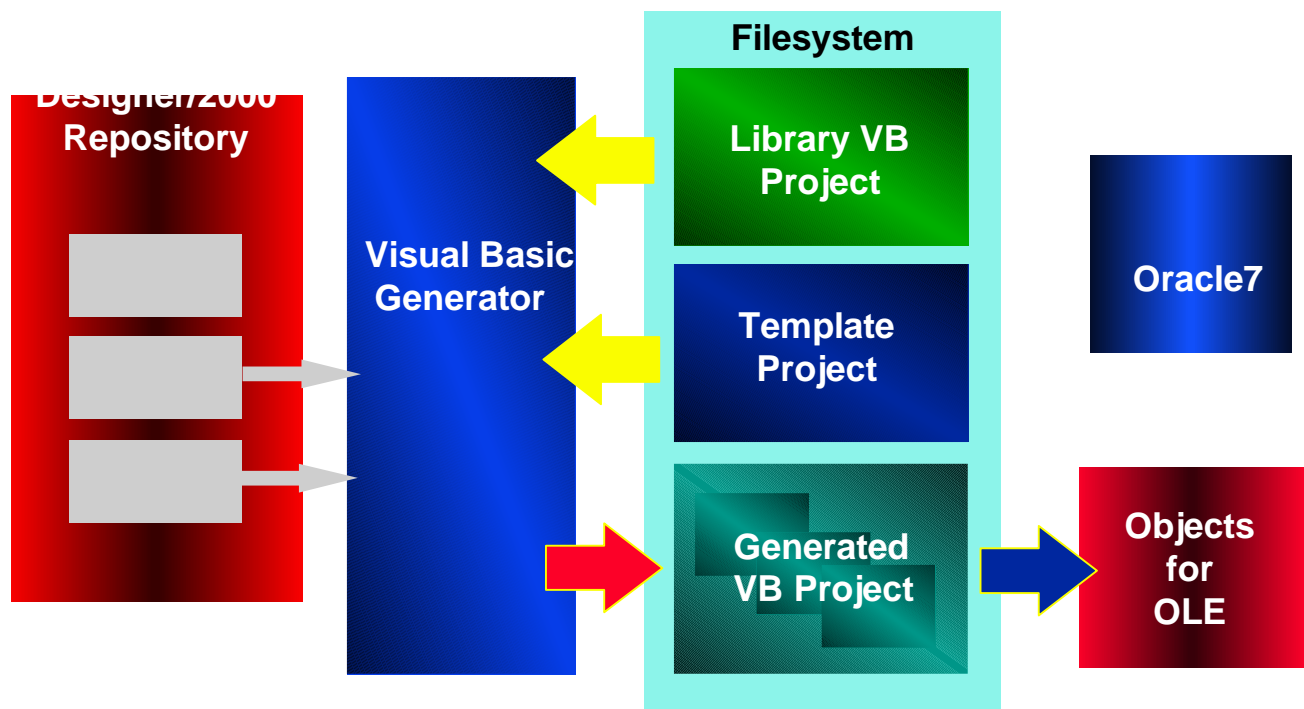
The database constraint definitions, database triggers and PL/SQL objects are all stored within the Designer/2000 repository, thus ensuring that full impact analysis and application integration is possible for both client and server application components.

The generated message handling is the same as that implemented in the Oracle Forms Generator, thus providing a consistent interface across applications generated and deployed into a heterogeneous environment.

3. Generating applications using the Visual Basic Generator.

The Visual Basic Generator can be invoked from the Module Data Diagrammer, the Module Structure Diagrammer or the Repository Object Navigator and, upon invocation, displays a dialog allowing you to control whether to generate the module as a new Visual Basic project or to add it to an existing project. If you wish to allow navigation between modules it is necessary to generate them into the same project. The dialog also allows you to identify the module as the startup form for the project - this tells the Generator that the module will be the first screen that appears when the application is run, and, moreover, code will be added to end the Visual Basic session when that main form is unloaded.

The Visual Basic Generator takes a number of inputs which govern the appearance and database integration of the resultant application. These inputs are the database design, module design, user preferences and template project.



The Data Diagrammer within Designer/2000 allows the creation of Tables, Views, Snapshots and their columns, and the various key constraints that can exist between these objects. All constraints can be flagged as being implemented in the Client code, in the server or both, although the Visual Basic Generator does not enforce any constraints marked as Client side - it does however intercept any constraint violation error messages and report the user friendly text associated with that constraint in the Designer/2000 repository.

The Module Logic Navigator within Designer/2000 allows the definition of Database Triggers, PL/SQL Packages, Procedures and Functions. Database triggers may call other PL/SQL procedures, which allows the building of complex applications based on the event model supported by the database triggers. Application logic implementing any complex business rules, secondary updates or derivation can be stored in the server and invoked whenever data is entered, updated or deleted from a table. Database triggers can fire either on a per row or per transaction level and, in the case of Update triggers, specified only to fire when a specific column, or set of columns, is updated.

Any application error raised by these server-based logic components will be recognised and reported back by the generated Visual Basic application - the mechanism of this application error handling is consistent with the behaviour of the Oracle Forms 4.5 Generator as well as the Oracle Power Objects generator.

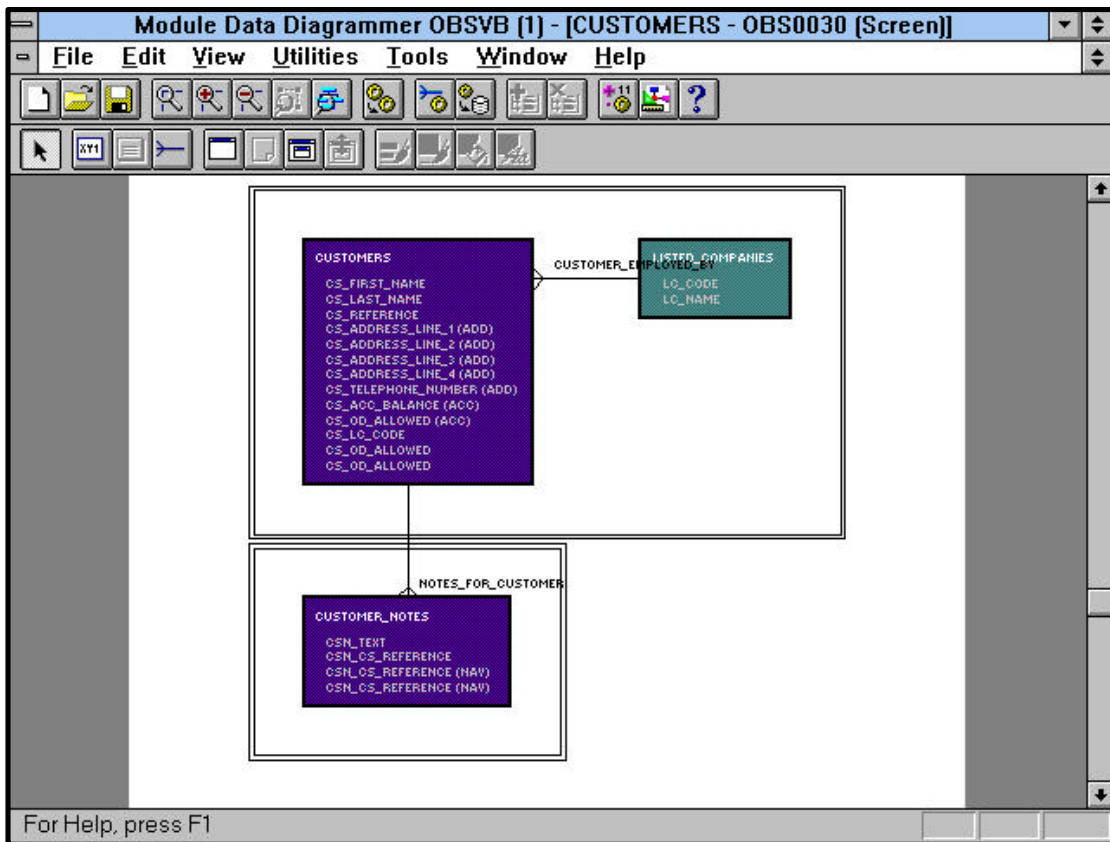
3.2 The module design.

The main input to the generation process is a module design specification recorded through the Module Data Diagrammer component of Designer/2000. This records the tables and columns used by the module, the links between them, and detailed information on how the module uses the data.

A generated Visual Basic application comprises one or more Visual Basic forms (or windows), and each generated form has one or more zones. The mapping of a single module to one or more Visual Basic windows is defined visually, using the Module Data Diagrammer.

A zone is the interface through which database records are queried, updated, created, and deleted. It is based on a specific table (its 'base' table), but may also use other tables to provide lookup information ('lookup' tables). For example, an Employee zone might be based on the EMPLOYEES table, but might also use the DEPARTMENTS table to provide lookup information on the department location and cost centre. Any table may be used multiple times in any one module, and lookup table usages may be chained to multiple levels. These references to tables are summarised automatically into Module Table Usages and Module Column Usages to enable full impact analysis of the Designer/2000 Repository, identifying which modules use which tables through the Matrix Diagrammer or through a number of reports included in the Designer/2000 product.

The Module Data Diagrammer allows you to select the tables, views and snapshots to display in the generated application, as well as the columns for each of these objects. When a number of tables are to be used you can define which Foreign Key constraints should be used to link these Tables usages together. It is possible to group together a base table usage and its associated look-up usages, and force the generator to place these items on the same zone.



In the above figure we are looking at a module that will contain two zones - the first is based on the Customers table with some information looked up from the ListedCompanies table using the CUSTOMER_EMPLOYED_BY foreign key to perform the join between the two tables. The second zone is based on the CustomerNotes table and will display a number of lines of text associated with the Customer.

For each zone it is possible to specify whether the end user should be allowed to Insert, Update or Delete data through that zone, and refine this definition further to specify the behaviour for individual columns.

When creating a module definition in Designer/2000, the headings and display characteristics will inherit the default values defined against the underlying table or column, but all of these characteristics may be overridden through the Module Data Diagrammer or the Repository Object Navigator.

Zones may be inter-related via master-detail relationships; these relationships may occur within a single form, or may span different forms. Links between module definitions (a 'module network') may be set up using the Module Structure Diagrammer, such that a zone may be related to another zone in another form sourced from another module definition - in this case the Visual Basic Generator would incorporate a button into the first Form to allow navigation to the first zone of the related module. The Generator also produces code to co-ordinate the way that master and detail zones interact; for example, a detail zone is disabled unless a valid master record is present.

3.3 User Preferences

User preferences are options that influence the appearance and behaviour of generated Visual Basic modules. Every preference has a default setting that is used by the Generator if the preference has not been set explicitly, but you may set these Preferences using the Preference Navigator component of Designer/2000. The Visual Basic Generator allows preferences to be set at application level, thus facilitating the enforcing of site standards, or module level to override a standard setting for specific unique modules.

Preference settings can also be grouped into named preference sets. This facilitates the development and completion of a site standard through the evolution and development of one module (called MASTER, say) and then allows you to generate the rest of the modules in the application using MASTER's preferences. This also facilitates the migration of sets of preferences between different development repositories, thus ensuring a consistent look and feel between application developed and generated from different repositories.

User preferences form one component of the on-line style guide that ensures all generated applications adhere to a consistent look and feel - this removes the onus from individual developers for producing applications that will integrate seamlessly.

The Visual Basic Generator preferences are grouped into seven categories:

- Controls
- DBA
- End User Interface
- Layout Control
- Layout Group.
- Layout Zone
- Templates

Controls

These preferences define the default mapping from the Control Style defined for a detailed column usage to the type of control to be used in the generated application. In the Module Data Diagrammer you can specify that a column should be displayed as a Checkbox, in the template you can define more than one CheckBox style control and then User preferences allow you to map the generic control type Checkbox to one of these specific controls in the template. Using these preferences it is possible to control the display control type for columns whose display datatype is in the following list:

- Checkbox
- Character Text
- ComboBox
- Date Text
- Integer Number Text
- ListBox
- Radio Group
- Real Number Text

It is also possible to specify the location of the control definition file to be used when generating the application. The control definition file contains information that defines the capabilities and characteristics of each type of control used in generated applications, particularly those additional controls beyond the standard set shipped with Visual Basic.

DBA

These preferences set the options for referencing database objects. These preferences allow the developer to specify how the generated application should interact with the CODE_CONTROLS table for getting its unique sequence numbers for new records, and also how the valid values for columns should be stored in the reference tables.

End User Interface

These preferences govern various aspects of the user interface; specifically they allow control over:

- Whether to generate hint text for Data Controls
- Whether to generate hint text for Buttons
- The maximum number of rows to auto-query
- Whether to generate Mnemonic Access Characters to Controls
- Whether to generate Mnemonic Access Characters to Data Entry Controls
- Whether to re-use characters when assigning mnemonics
- Whether to generate modal Property Sheet Dialogs

Layout Control

These preferences control the way the Generator lays out individual controls; for example, the gap between a textbox and its caption. With these preferences you can define:

- What character to use as the default caption separator
- How large to make the gap between a control and the next caption
- Whether to allow captions above data controls
- How large to make the gap between a caption and the next control
- What marker should be used to identify where a prompt should be split around the control (for example the caption 'Distance(miles)' would cause the VBG to create the following Distance (miles)
- What marker should be used to identify where a prompt should be split over multiple lines
- Alignment of list of values button
- Gap between the list of values control and button
- Radio group button orientation
- Inter button spacing and multi-line format of horizontal radio groups
- Inter button spacing of vertical radio groups
- Default length for Text controls based on character, date and numeric datatypes.

Layout Group

These preferences determine how the controls in a generated control group are placed. Using these user preferences it is possible to

- Specify the spacing between items in a horizontally oriented group
- Specify the spacing between items in a vertically oriented group

- Specify the orientation of a group
- Specify the number of controls to be placed on a line in a horizontally oriented group

Layout Zone

These preferences control the way the Generator places controls within a zone, property sheet dialog, or query dialog. Using these User Preferences allow you to control the layout within zones, and control the following components:

- Specify how controls are centred in a Zone
- Specify the number of controls that are placed on each line in a zone that wraps onto more than one line
- Specify the amount of vertical space used to separate lines of controls in a zone where the record wraps onto more than one line.
- Specify whether or not the Generator uses tab stops for positioning controls and captions in a multi-line row
- Specify the tab stops that are used to display controls and their captions.
- Specify whether or not the Generator uses the space available below short controls that are adjacent to tall controls.

Templates

These preferences control the options that give the location of the template and library objects, and that define the use of template menus. The following preferences control the following aspects of the templates:

- Location of the Library (standard inclusions) project
- Specify the name of the menu control that indicates where menu items will be placed to locate module network entries
- Specify the name of the menu that indicates where navigation items involved in zone co-ordination will be placed.
- Location of the template project.

3.4 Templates

Templates determine the look-and-feel of the forms, zones, dialogs, frames, and controls that make up the generated Visual Basic application. By documenting the site standards for all the look and feel of Visual Basic applications in the template and preferences, the style guide is defined on-line and applied by the Generator, with no need for individual developers to get involved in changing the appearance of their applications.

A set of standard templates is shipped with the product as a Visual Basic makefile. This contains a set of template forms, each responsible for defining the templates for a specific purpose. The forms are of four main types:

- Template Controls
- Template Windows
- Template Zones
- Template Groups

Template Controls

The single template controls form contains controls that define the appearance of generated controls. It is here that the developer can set fonts, sizes, colors, and so on. For each control type (say, TextBox), you may define up to four versions. These determine the look and feel when the control is used as:

- a mandatory control
- a query-only control
- an optional control
- a control within the query dialog

Template Windows

The template windows define the appearance of the generated windows, property sheet dialogs, and query dialogs. The size of the template form defines the size of the largest window to be generated and the items within the template define the margin size. It can also include controls to be copied in during generation.

Template Zones

The template define the appearance of generated zones. This includes:

- the relative position of the zone's controls to its buttons
- the orientation of the buttons
- the appearance and location of other controls, such as browse buttons
- the look and feel of the frame that the zone may be placed on

Template Groups

The template groups define the look and feel of the frames that enclose stacked and unstacked control groups, and of the radio group used to bring the stacking area to the front. These also define the look and feel of the zones that become part of the stacking area.

3.5 Visual Basic Library Forms and Code

Standard library forms, code, and controls are included with Visual Basic Generator. These elements are known as 'standard inclusions'. Standard inclusions are not generated, but simply copied over at generation time. They consist of the following:

- a set of routines dealing with database access
- a large number of general routines
- a set of global constants, where the user can modify end user message text if required
- the logon form, used as a front-end to generated applications, where the user enters logon and database connection information
- the server-error form, used to display detailed information about server errors
- an about box that provides information about the generated application
- a file containing global constants
- a file containing Oracle Objects for OLE global constants

3.6 Incorporating Third-Party Controls

A comprehensive set of controls is supplied with Visual Basic, but one of the strengths of Visual Basic development is the ability to reuse 3rd party controls in your own applications. There are many companies involved in the writing and release of VBX controls and these often provide a cheap and easy way to assemble applications. It is straightforward to incorporate these third party controls into generated applications by adding a new template control for it in the template controls form (which defines the appearance of generated controls) and describing its properties within the control definition file (which defines the characteristics and capabilities of the control type).

Once a third-party control has been defined, it can be used during generation by either setting up the user preferences so that all controls of a given type (say, CheckBox) use the new control or by setting the Template Control Name property of selected column usages so that only specific controls within specific modules use the new control.

Third-party controls may be used for buttons and frames, as well as for data entry controls.

4.0 Summary

The Oracle Designer/2000 Visual Basic Generator facilitates the building of Oracle7 based Visual Basic systems by combining the benefits of a repository-based teamworking environment with support for the modelling and generation of complete client-server applications.

The Visual Basic Generator supports a variety of layout options, and provides comprehensive support for inter-form and intra-form navigation, thus ensuring that applications can be generated which adhere to a variety of different styles and standards.

The module definitions captured in the Designer/2000 provide the comprehensive design documentation and impact analysis support that is essential for the successful management of client-server development.

The shared data definitions and user preferences help to enforce a common UI style and behaviour across any number of generated forms, thus enabling the integration of generated components into larger systems that share a common look and feel.

The use of Oracle Objects for OLE ensures that the generated Visual Basic applications benefit from tight integration with the Oracle7 server.