Oracle® WebDB™

Creating and Managing Components - Task Help

Release 2.2

October, 1999

Part No: A77057-01

Oracle WebDB Creating and Managing Components - Task Help, Release 2.2

Part No: A77057-01

Authors: Susan Barton, David Mathews, Carol Menzigian

Contributors: Haranadh Abburu, Mark Clark, Susan Léveillé, Thiagarajan Natarajan, Frank Rovitto, Todd Vender, Hui Zeng

Table of Contents

Send Us Your Comments

Oracle WebDB Creating and Managing Components - Task Help, Release 2.2

Part No: A77057-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the part number, chapter, section, and page number (if available). You can send comments to us by electronic mail to webdbdoc@us.oracle.com.

If you have any problems with the software, please contact your local Oracle World Wide Support Center.

About This Book

Oracle WebDB Creating and Managing Components - Task Help (Part No: A77057-01) is a hardcopy version of the online Task Help for the component building features of WebDB. Task Help contains step-by-step instructions for performing tasks in WebDB.

To display online Task Help in WebDB, click the large help button  located on the upper right of every WebDB page.

The contents of this book are also available in printable Adobe Acrobat PDF format on your WebDB product CD at:

`\doc\us\comptask.pdf`

The following table lists the other information units in the WebDB documentation library:

Information Unit	**Description**	**Location in \doc\us directory on WebDB product CD**
Release Notes (Part No: A77054-01)	Describes last minute changes to the product or documentation. Read the Release Notes before installing or using Oracle WebDB.	`relnotes.htm`
Quick Tour	Provides a high-level introduction to Oracle WebDB. Accessible by clicking "Quick Tour" in the WebDB or Site Builder task help Table of Contents.	`\QuickTour\wbstart.htm`

Oracle WebDB Tutorial Guide
(Part No: A77075-01)
	Provides step-by-step lessons that teaches you the basics of WebDB.	**PDF version:**
		`\Tutorial\tutorial.pdf`
		HTML version:
		`\Tutorial\html\toc.htm`

Information Unit	Description	Location in \doc\us directory on WebDB product CD

Oracle WebDB Installation Guide
(Part No: A77053-01)
	Provides complete Oracle WebDB 2.2 installation instructions for both Windows NT or Solaris environments. Also, provides upgrade and site migration details, WebDB Listener installation and configuration, *inter*Media Text set up information, and WebDB CGI executable and WebDB cartridge to Oracle Application Server setup information.	**PDF version:**
		`\Installation\installation.pdf`
		HTML version:
		`\Installation\html\toc.htm`

Oracle WebDB Creating and Managing Components - Field-Level Help (Part No: A77055-01)	Hardcopy-formatted version of the online Field-Level Help for the component building features of WebDB.	`compref.pdf`
Oracle WebDB Creating and Managing Sites - Task Help (Part No: A77056-01)	Hardcopy-formatted version of the online Task Help for the Web Site building features of WebDB.	`sitetask.pdf`
Oracle WebDB Creating and Managing Sites - Field-Level Help (Part No: A77058-01)	Hardcopy-formatted version of the online Field-Level Help for the Web Site building features of WebDB.	`siteref.pdf`

Setting Up WebDB

About WebDB configuration tasks

After installation of WebDB, the DBA and WebDB Developers should complete (in the order shown below) the following tasks before users begin creating WebDB components or sites:

- Configure the HTTP Listener
- Create WebDB User Accounts
- Assign WebDB roles
- Create Component and Object Schemas
- Grant object privileges to Component Schemas
- Provide WebDB-specific build and browse privileges to developers
- Create shared components in addition to the default set provided by WebDB

Configuring the Listener

The Listener is a lightweight Web server with a built-in PL/SQL gateway that helps build and deploy PL/SQL-based applications such as WebDB. During WebDB installation, the Listener and PL/SQL gateway are configured with settings collected as part of the install process and some other default settings. A user with the DBA role can run WebDB with this configuration, or update it. For example, the DBA may want to change:

- WebDB Data Access Descriptor (DAD) settings.
- the default home page.
- MIME types.
- directory mappings.
- other settings that affect Listener performance.

To change your current Listener and PL/SQL Gateway settings:

1. Click  at the bottom of any WebDB page.

2. Click **Listener Settings**. The Change PL/SQL Gateway Settings page displays. To change settings, type new ones in the entry fields on the page, then click **Apply**.

3. To change Listener settings, click the hypertext link, **Change Listener Settings** at the top of the page.

Notes

- Click the small help icon  to view information about entry fields on the Change PL/SQL Gateway Settings and Change Listener Settings pages.
- If your end user cannot open a file downloaded from a WebDB site, make sure that you add the correct mime type and associated file extension to the MIME Types table listing on this page; for example, the mime type can be application/excel with extension xls.
- See "Chapter 5, "Configuring the WebDB Listener and PL/SQL Gateway" of the *"Oracle WebDB Installation Guide"* for more information on using the Listener to

run multiple virtual hosts, serving static files, and running PL/SQL applications with the Listener. Also, for advanced users, see "Manually Editing the Listener and Gateway Configuration Settings."

About Data Access Descriptors (DADs)

A DAD is a set of values that specify how the PL/SQL gateway connects to the Oracle database server to fulfill an HTTP request. There are separate DADs for the instance of WebDB that you installed in the database and for each Web site you create using WebDB. For example, if you install WebDB and use it to create a Web site, the following DADs are automatically configured with default values that you specify at installation time:

- A DAD for the component building features of WebDB. The name for this DAD is the same as the WebDB User Name that was specified in the WebDB install wizard when WebDB was installed on Windows NT or Solaris.

- A DAD for the public user account created for the Web site. This is the portion of the Web site accessible to all end users. The DAD name for this account is the same name as the Owning Schema name that was specified in the Site Creation Wizard when the site was built; for example, `MYSITE`. The DAD is automatically configured with the user name and password you see on this page. This means that public users can access the WebDB site without having to log on.

- A DAD for the administration user account created for the Web site. This is the portion of the Web site accessible to the Site Administrator. The DAD name for this account is the same name as the owning schema plus the character `S`; for example `MYSITES`. This DAD is not configured with a user name and password. This means that users cannot access site administrative features without first logging on. Type in the user name and password shown on the last page of the Site Creation wizard to access site administrative features for the first time.

As you generate more Web sites at your location, additional DADs for the public and administrative user accounts for these sites are generated.

About WebDB user accounts

After installing WebDB, the DBA must create a user name and password for users who do not already have access to the Oracle database where WebDB is installed. New WebDB user accounts must be created in the WebDB User Manager.

In addition to setting up a user name and password, the DBA defines:

- The user's storage information. This includes the user's default and temporary tablespaces and an Oracle Profile.

- Whether the user will be a WebDB Developer who can create components, or a Component Schema that will own the components created by WebDB Developers. A user identified as a WebDB Developer is automatically granted the WEBDB_DEVELOPER role and can by default build components in his or her own schema.

A user name is the same as a schema name in WebDB. For example, if you create a WebDB Developer named SCOTT, this user is able by default to create WebDB components and database objects in the SCOTT schema.

About the DBA and WEBDB_DEVELOPER roles

The DBA must assign one of two WebDB-specific roles to every user. Users cannot perform any tasks in WebDB unless they are member of one of these roles. A WebDB role defines the actions the user can perform by controlling access to WebDB menus. The role also specifies the user's default privileges for browsing and building in schemas.

If you plan to build WebDB components or objects, you must have the WEBDB_DEVELOPER role. A user identified as a WebDB Developer in the User Manager is automatically granted the WEBDB_DEVELOPER role.

Users can also be granted the DBA role. Since this role grants all WebDB privileges, including the ability to create and drop users, only users who are familiar with the Oracle database should be granted this role. DBAs can grant the DBA role to other users.

WebDB roles and their associated privileges are listed below.

Role	Privileges granted to user
DBA	• Access to all WebDB menus. • All Oracle database administration privileges, including creating and dropping users, and assigning Oracle and WebDB-specific roles and privileges to other users. • Browse In any schema in the database for objects and components. • Create WebDB components (such as reports, charts, and menus) and objects (such as table, triggers, and synonyms) in their own schema and any schema in which they have been granted Build In privileges.
WEBDB_DEVELOPER	• Access to all WebDB menus except those under the Administration, Monitoring, and Site Building links on the home page. • Browse for database objects and components in their own schema and any schema in which they

have been granted Browse In privileges.

- Create WebDB components and objects in their own schema and any schema in which they have been granted Build In privileges.
  Note These privileges are automatically provided if the user is granted the WEBDB_DEVELOPER role in the User Manager. If the WEBDB_DEVELOPER role is granted to the user in the Role Manager, Build In privileges are not automatically granted.

The DBA creates roles and manages role members in the Role Manager. After assigning a role, the DBA can navigate to the User Manager and assign to a user Build In and Browse In privileges in addition to those associated with the role.

About Oracle roles

Oracle roles grant groups of users Oracle privileges such as SELECT, INSERT, UPDATE, DELETE, and EXECUTE on objects in the database. A DBA can create Oracle database roles and manage role members using the Role Manager and the Role tab of the User Manager.

Roles reduce the number of privileges that the DBA has to explicitly grant to each user. A DBA can grant the same privileges for a group of related users who are members of the same role.

If the object is being used to build a WebDB component, these privileges must be explicitly granted to the Component Schema through the WebDB Grant Manager, not through an Oracle role. Therefore, although the DBA can create roles to grant objects any of the above privileges, the DBA typically creates roles only to grant EXECUTE privileges on procedures. For example, the DBA can create a role that allows users to execute a component.

Assigning users to Oracle Roles


You must have the DBA role to assign a user to a role.

You typically use an Oracle role only to grant EXECUTE privileges that allow groups of users to run WebDB components. This is because you must explicitly grant SELECT, INSERT, UPDATE, DELETE, or EXECUTE privileges (using the Grant Manager) on objects used to build WebDB components. These privileges cannot be granted through a role.

For example, you might create a role that grants EXECUTE privileges on all procedures in the SCOTT schema, then assign that role to all users to whom you want to grant those privileges.

You follow different steps depending on whether you want to give one role to several users, or give several roles to a single user.

To assign one or more users to a single Oracle role:

1. Click  on the bottom of any WebDB page.

2. Click **Role Manager**.

3. In the **Find an Existing Role: Role** field, type the name of the role you want to assign to the user or role, then click **Find**. To search for existing roles, click  .

   After you click **Find**, the Manage Roles page displays current members of the role you selected display in the **Members** list box.

4. If the user or role is not already in the **Members** list box, type a user or role name in the **User/Role** field, then click **Add**. To search for users or roles, click  .

5. Repeat this step to continue adding user or roles to the list. The user or role name appears in the **Members** list box each time you click **Add**.

6. Click **Apply**.

7. (Optional) To remove a member from a role, select one or more members in the **Members** list box, then click **Remove**.

To assign one or more Oracle roles to a single user:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager.**

3. In the **Find an Existing User: User Name** field, type the name of the user to whom you want to assign a role. To search for users, click  .

4. Click **Find**.

5. Click the User Manager **Roles** tab.

   The **Is Member Of** list box displays all roles to which the user currently belongs.

6. To assign a new role to the user, type the name of the role in the **Role** field, then click **Add.** To search for roles, click  .

   The role appears in the **Is Member Of** list box each time you click **Add**.

7. Click **Apply**.

8. (Optional) To remove a member from a role, select one or more roles in the **Is Member Of** list box, then click **Remove**.

Notes

- WEDB_DEVELOPER and DBA are special kinds of roles. These roles provide access to WebDB menus and grant default Build In and Browse In privileges. You typically assign them in the User Manager.

- You can make a role a member of another role. For example, you can make a role called `Sales_Personnel` a member of another role called `Headquarters_Personnel`. This causes all members of the Sales_Personnel role to have the same privileges as the Headquarters_Personnel role, but not vice versa.

About Component and Object Schemas

Before permitting users with the WEBDB_DEVELOPER role to create components and objects, the DBA must first create special types of database user accounts called Component Schemas. These accounts are created in the User Manager just like regular WebDB user accounts. Component Schemas own the components built by WebDB Developers. By default, all WebDB Developers can build a component in their own schema. To build in a schema other than their own, developers must have Build In privileges in at least one other Component Schema.

After creating a Component Schema, the DBA must explicitly grant to it (using the WebDB Grant Manager) SELECT, INSERT, UPDATE, DELETE, and EXECUTE on any table, view, or procedure that will be used to build a component in the Component Schema. For example, to build a report in a Component Schema based on the SCOTT.EMP table, the Component Schema must have SELECT privileges on SCOTT.EMP.

If the object is located in the Component Schema itself, no object privileges are required. For example, the SCOTT schema automatically has privileges on the SCOTT.EMP table.

Any WebDB Developer who has Build In privileges in the Component Schema is automatically granted the object privileges of the Component Schema. For example, in order for a developer named JANE to build a report in the FRED Component Schema based on the EMP table located in the SCOTT schema:

- JANE must be granted Build In privileges in the FRED schema.
- The FRED schema must be explicitly granted SELECT privileges on the SCOTT.EMP table.

Note that JANE is not required to have explicit object privileges on SCOTT.EMP

Tips One easy way to manage the objects needed by Component Schemas is to place them in the Component Schemas themselves.

You can also set up your database with separate Component Schemas and "Object" Schemas. An Object Schema is simply a database schema that contains objects. It could contain all the objects you think are required for WebDB Developers to build components in a particular Component Schema.

It’s not necessary to set up a matching Object Schema for every Component Schema For example, the DBA could place objects needed by several Component Schemas into a single schema.

About Oracle database object privileges

An Oracle database object privilege is a permission granted to a user to perform some action on a database object. These object privileges include SELECT, INSERT, UPDATE, DELETE on tables and views and EXECUTE on procedures, functions, and packages. They can be granted using the WebDB Grant Manager or using Oracle commands. If the object is being used to build a component, these privileges must be explicitly granted using the Grant Manager, not through the use of a role.

If an object is located in the user's schema, no explicit grants are required.
The DBA typically grants Oracle database object privileges to WebDB Component Schemas. Any WebDB Developer who has Build In privileges in the Component Schema is automatically granted the object privileges of the Component Schema.

About Build In and Browse In Privileges

By default, users have Build In and Browse In privileges within their own schema, but are typically granted permission to browse other schemas, or build in other Component Schemas. After assigning the DBA or WEBDB_DEVELOPER role to a user, the DBA may want to assign Build In and Browse In privileges to the user in addition to those associated with the role:

- Build In privileges specify which Component Schemas will own the components built by the WebDB Developer. A Component Schema is any schema that has been designated as such in the User Manager.

- Browse In privileges specify which database schemas the developer can browse for objects such as tables, views, and procedures on which the component will be based. The developer can also browse for other objects such as functions or triggers that can be included in the code that creates the component.

  Browse In privileges provide the user the ability to view database objects using the Browse feature of WebDB. To build a component, the Component Schema the user is building in must be granted explicit privileges on the database object such as SELECT, INSERT, UPDATE, DELETE, and EXECUTE.

Users given Build In privileges in a schema automatically receive Browse In privileges in that schema.

Adding to the default set of shared components

The DBA should identify developers who will be responsible for creating any shared components required to build charts, reports, and other WebDB components. Shared components include:

- Links that create hypertext jumps between WebDB components.
- Lists of Values that add selectable lists to entry fields in forms and component parameter entry forms.
- User interface templates that control the look and feel of components.
- JavaScripts that perform field- and form-level validation on components.
- Color, image, and font definitions that identify the colors, images, and fonts used in the components.

WebDB includes a default set of shared components during installation. You can search for shared components after installation using the Find an Existing Component Page.

Getting Around in WebDB

Changing your password

Any WebDB user can change their own password.

To change your password:

1. Click  on the bottom of any WebDB page.
2. Click **Change Your Password.**
3. In the **New Password** field, type the new password.
4. In the **Confirm Password** field, retype the password.
5. Click **Apply.**

   A page appears indicating whether you successfully changed the password.

About user connection information

You can view information about your current WebDB session by clicking the  button in the upper left corner of any WebDB page. User connection information is information about your current WebDB session, and is based on the user ID and password you used to log into WebDB.

Connection information includes the following:

- Your user ID.
- WebDB roles currently granted to you; for example, `WEBDB_DEVELOPER`.
- Schemas you can build and browse in.
- Your current language preference for WebDB menu text; for example, `(en)` English or `(sp)` Spanish.
- The name of the server running WebDB and the schema where WebDB was installed: for example, `webdb.mycompany.com/webdb/`
- The IP address of the server and the connection protocol; for example, `HTTP/1.00`.
- The Web browser you are currently using to view WebDB; for example, `Mozilla`.

You can also log off your current WebDB session or change your password from this page.

Logging off

To log off your current WebDB session:

1. Click  at the top of any WebDB page.

2. Click **Log off**.

Notes

- If you close your browser, you are automatically logged off WebDB.

- If you use your browser's Back button to navigate to a non-WebDB page, you are not logged off WebDB. You must follow the steps shown above to log off.

Navigating in WebDB

Using the navigation toolbar

At the bottom of every WebDB page is a navigation bar that allows you to quickly link to other WebDB menus and pages.

Place your mouse over any button on the navigation bar to display a small pop-up that shows the link destination for the button. Click the button to link to the destination.

Using the Frames view

WebDB provides a hierarchical tree that you can use to navigate to WebDB menus and pages. To use the tree, click  on any WebDB page to navigate to the WebDB home page, then click  . The home page divides into two frames. The left frame contains an expandable tree hierarchy of WebDB menus and pages. Click a node in the tree to navigate to a menu or page.

Using the link history

WebDB menus contain a history of the links that you made to navigate to the current menu or page. The history appears near the bottom of the menu above the navigation bar. For example,

Back: User Interface Components, Build, Oracle WebDB

might appear at the bottom of a User Interface Component building menu. You can click on any of the links in the history to navigate to a previous page.

Searching for WebDB menu text

You can use the **Find Menu Options** field on the home page to search for text in any WebDB menu or menu description. This feature is convenient for quickly navigating to pages or menus, or locating a page or menu when you are unsure of the navigation path.

To search for text in a WebDB menu or menu description:

1. Click  in the upper right corner of any WebDB page.

2. In the **Find Menu Options** field, type the text you want to find. For example, type `object` to find menus and pages with the word "object" in them.

3. Click **Find Menu Options**.

   A page appears displaying hypertext links to all menus and pages matching your search criteria.

4. Click a link to display a menu or page.

Browsing the Database

About browsing

WebDB browsing provides a graphical view of the objects in your database, such as tables, packages, and procedures. You can browse the database using object names, object types, the schemas owning the objects, or any combination of these search criteria. For example, you can search for all tables owned by the SCOTT schema. You can also search for a table named EMP in the SCOTT schema, or tables named EMP in all schemas.

Once you locate an object, you can click it to perform different actions depending on the type of object. For example, you can:

- query and update table and view contents.
- execute database procedures and functions.
- view the contents of packages.
- view information about triggers, synonyms, or indexes.

To browse an object, you must have Browse in privileges in the schema that owns it. Browse In privileges only allow you to browse an object. To perform other actions on the object, such as use it to build a WebDB component, the schema where you are building the component must be granted explicit privileges on the object.

Notes

- The WEBDB_DEVELOPER role automatically grants privileges to Browse In your own schema.

- WebDB uses the terms **component** and **object** to distinguish between the dynamically generated HTML Web pages and content that WebDB creates, and the Oracle database objects on which these components are based. Use the Browse

Objects page to search for objects. Use the Find Component page to search for components.

- WebDB returns a maximum of 1,000 objects in a search for any object type. If a search returns more than 1,000 objects, you need to narrow your browse criteria.

Browsing database objects

To browse the database for objects:

1. Click  at the bottom of any WebDB page.

2. In the **Schema** field, type the name of the schema you want to browse. If you aren't sure of the schema, click  to the right of the field to view a pop-up list of schemas. You can type search criteria in the pop-up list to refine your search.

   If you leave this field blank, WebDB searches all schemas you have privileges to browse.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list, choose an object type.

4. In the **Object Name** field, type the name of the object you want to find.

   If you aren't sure of the object's name, you can use a wildcard to search for it. For example, typing `%EMP%` might locate a table named `Employee` and a function named `Calc_Employee_Salary`.

5. Click **Browse**. A page displays all objects matching the search criteria you specified.

   Note If you didn't specify search criteria in the **Schema** field, a page containing schemas may display. Click a schema on the page to display its objects.

Notes

- You can search for database objects by specifying search criteria in the **Schema**, **Type**, **Object Name** entry fields or any combination of these entry fields.

- To narrow your browse search criteria, specify values in all three entry fields.

- You can use the % wildcard in the **Schema** and **Object Name** fields. For example, typing `SCO%` in the **Schema** field might locate the schemas named `SCOTT` and `SCOUT`.

Viewing object information

WebDB provides information about objects stored in the database, including the object's owner, type, and dependencies on other objects.

To view object information:

1. Click  at the bottom of any WebDB page.

2. In the **Schema** field, choose the schema that owns the object. To search for schemas, click  to the right of the field.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list, choose an object type; for example, **Tables** if you want to view information about a table object.

   (Optional) In the **Object Name** field, type the name of the object. You can use the % wildcard if you aren't sure of the name: for example, E% to search for the EMP table.

4. Click **Browse**.

5. Click an object.

6. Click **Show Object information**.

   A page displays the object information.

Querying Tables and Views

About tables and views

Tables and views are two types of database objects supported by WebDB. Once you have located a table or view using one of the WebDB browsing methods, you can query and update data within the table or view using a Query by Example form.

In addition, you can use the Query by Example form to:

- Choose which table or view columns to display.
- View unique values within table or view columns.
- Use logical operators to select data within the columns.
- Choose a format and method for displaying query results.

Note

- When end users display views, updateable columns are displayed in black, non-updateable columns are displayed in blue, and required (mandatory) columns are displayed in red.

Logical operators in table and view queries

You can use the following logical operators to query tables and views. These operators work in Query by Example forms and component parameter entry forms:

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
%	wildcard

Querying tables and views

To query the contents of a table or view:

1. Click  at the bottom of any WebDB page.
2. In the **Schema** field, choose the schema that owns the table or view. If you aren't sure of the schema, click  to the right of the field to view a pop-up list of schemas. You can type search criteria in the pop-up list to refine your search.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list, choose `Table & Views`.
4. (Optional) In the **Object Name** field, type the name of the table or view you want to query.
5. Click **Browse**.
6. Click the table or view you want to query.

   A Query by Example form for the table or view displays. The Query by Example form contains entry fields corresponding to each table or view column you want to query.

7. Type or select your query criteria in one or more of the table or view column entry fields.
8. Check the check box next to each column you want to display in your query results.
9. (Optional) Specify **Order by** and other query options on the Query by Example form. If you have a question about an option, click the small  button to view

help.

10. Click **Query**. A page displays a table containing the columns you selected and all table rows matching your search criteria.

Notes

- To view all rows in the table or view, leave all table column entry fields in the Query by Example form blank or deselected, then click **Query**.
- Click **Reset** to clear values from all column entry fields.
- You can use the % wildcard in the **Schema** and **Object Name** fields. For example, typing SCO% in the **Schema** field might locate the schemas named SCOTT and SCOUT.

Updating tables

To update a table row:

1. Click  at the bottom of any WebDB page.

2. In the **Type** list page, choose `Tables`.

3. In the **Schema** field, choose the schema that owns the table. To search for schemas, click  to the right of the field to view a pop-up list of schemas. You can type search criteria in the pop-up list to refine your search.

   Note Only schemas that you are allowed to browse in are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

4. In the **Object Name** field, type the name of the table you want to update.

5. Click **Browse**.

6. Click the table you want to query.

   A Query by Example form for the table displays. The Query by Example form contains entry fields corresponding to each table column you want to update.

7. Specify query criteria in one or more of the entry fields for the table rows you want to update. For example, to update records for sales clerks in a table containing employee data, you might enter `Sales` in the Department entry field and `Clerk` in the Job Title entry field.

8. Check the check box next to each column you want to display in your query results.

9. (Optional) Specify **Order by** and other query options on the Query by Example form. If you have a question about an option, click the small  button to view help.

10. Click **Query and Update**.

    A page displays a report of all table rows matching your search criteria.

11. Click **Update** in the **Action** column for the table row you want to update.

    A form displays the column names and current values for the table row you are updating.

12. Type over or reselect the values in the column entry fields and click **Update** to update the row with the values you select.

To insert a new row into a table:

1. Follow steps 1-6 above.

2. Type the values you want to insert in the column entry fields.

3. Click **Insert New Row**.

To delete a row from a table:

1. Follow steps 1-11 above.

2. Click **Delete.**

Notes

- Red text displayed in the Query by Example form indicates a table column that does not accept null values. Leaving columns blank that do not accept null values causes an error.

- When end users display views, updateable columns are displayed in black, non-updateable columns are displayed in blue, and required (mandatory) columns are displayed in red.

- You can insert a new row, delete a row, or update a row only if you have been granted the required database object privileges on the table.

Viewing distinct table column values

To view a report of all the distinct values in a table or view column:

1. Click  at the bottom of any WebDB page.
2. In the **Schema** field, choose the schema that owns the table or view.
3. In the **Type** list, choose `Table & Views`.
4. (Optional) In the **Object Name** field, type the name of the table or view.
5. Click **Browse**.
6. Click a table or view.

   The Query by Example form for the table or view displays.
7. Click the datatype icon next to the table or view column whose values you want to view. For example, click  next to a column whose values have the VARCHAR datatype.
8. Click **Report Distinct Values**.
9. (Optional) Click a value to view the table row or rows that contain the value.

Browsing Procedures, Functions, and Packages

About packages, procedures, and functions

Packages, procedures, and functions are types of Oracle database objects supported by WebDB. Procedures are groups of SQL and PL/SQL statements containing business rules and application logic. Functions are procedures that return values. A package groups functions, procedures, and other PL/SQL commands.

Once you have located a package, procedure, or function using one of the WebDB browsing methods, you can click the object to perform additional actions:

- Browse the contents of packages.
- View information about packages such as the package spec and body, and dependencies.
- Execute any functions or procedures that the package contains.

Browsing package contents

To browse the functions and packages contained within a package:

1. Click  at the bottom of any WebDB page.

2. In the **Schema** field, choose the schema that owns the package. To search for schemas, click  to the right of the field.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list, choose **Packages, Procs & Functions**.

4. (Optional) In the **Object Name** field, type the name of the package whose contents you want to view.

5. Click **Browse**.

6. Click the package whose contents you want to display.

7. A page displays the functions  and procedures  contained within the package.

8. (Optional) Click a function or procedure to execute it.

Viewing the package spec and body

The package spec contains the list of functions, procedures, variables, constants, cursors, and exceptions contained within the package. The package body contains the PL/SQL code implementing the specification.

To view a package's spec and body:

1. Click  at the bottom of any WebDB page.

2. In the **Schema** field, choose the schema that owns the package. To search for schemas, click  to the right of the field.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list, choose **Packages, Procs & Functions**.

4. In the **Object Name** field, type the name of the package whose contents you want to view.

5. Click **Browse**.

6. Click the package whose spec and body you want to view.

7. Click **Show: Object Information**. A page displays the package spec and body.

   Note You may need to scroll down to bottom of the browser window to view the spec and body.

Executing procedures and functions

To execute a procedure or function stored in the database:

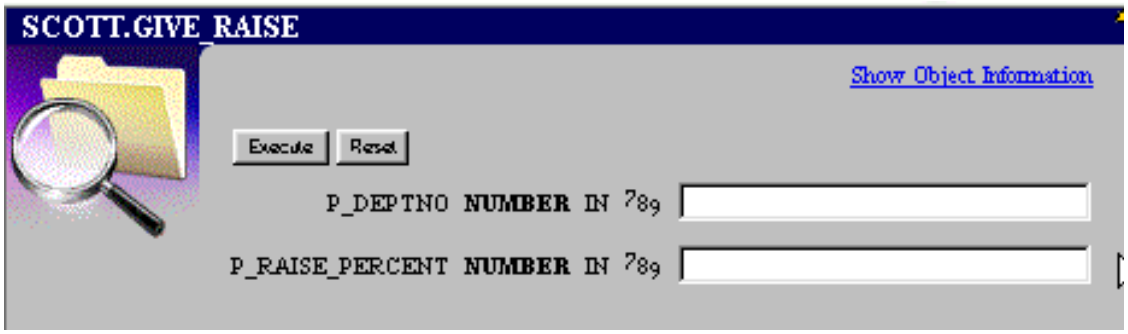
1. Click  at the bottom of any WebDB page.

2. In the **Schema** field, choose the schema that owns the procedure. If you don't know the name of the schema, click  to the right of the field to view a pop-up list of schemas. You can type search criteria in the pop-up list to refine your search.

   Note Only schemas that you are allowed to browse are listed. To browse another schema, contact your DBA to obtain Browse In privileges for the schema.

3. In the **Type** list box, choose `Procedures` (or `Functions`).
4. (Optional) In the **Object Name** field, type the name of the procedure or function.
5. Click **Browse**.
6. Click the procedure or function you want to execute.

   A form displays entry fields for each argument in the procedure or function. The following example shows the form for the GIVE_RAISE procedure, stored in the SCOTT schema. The procedure grants a raise to all members of a specified department contained in the SCOTT.EMP table (or any table that uses the same column names as SCOTT.EMP). The form contains entry fields for two arguments: `P_DEPTNO` and `P_RAISE_PERCENT`.



Tip If you are unsure about what the procedure is supposed to do when it is executed, click **Show Object Information**. A page displays the procedure's source code, plus other information such as its owner, status, and when it was last updated.

7. Type each argument you want to pass to the procedure or function in the appropriate entry fields. In the above example, you might type 10 for the department (`P_DEPTNO` entry field) and 7 for the raise percentage (`P_RAISE_PERCENT` entry field).

8. Click **Execute**. If the procedure or function executes successfully, you will see a  success icon.

Searching for WebDB Components

Searching for components

To find a WebDB component such as a chart, report, or form:

1. Click  on the bottom of any WebDB page.

2. In the **Schema** list, choose the schema that owns the component you want to find.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to build in this schema.

3. (Optional) In the **Name Contains** field, type one or more consecutive characters of the component's name.

   The **Name Contains** field is not case sensitive.

4. (Optional) If you know the component's type (for example, chart, form, report), check the check box next to its type in the **UI Components** check box group and uncheck all other check boxes. You can also choose a component **Status** to narrow your search. For example, you can choose **Archive** to search for old versions of the component, and **Production** to search for the most recent version.

5. Click **Find**. A list of components that match your search criteria appears at the bottom of the page.

   The  icon indicates components created within the last 7 days.

6. Click a name in the **Component Name** column to perform actions such as executing, editing, or managing the component.

Viewing information about components

To view information about a component such as its owner, creation date, status and stored attributes:

1. Click  on the bottom of any WebDB page.

2. In the **Schema** list, choose the schema that owns the component you want to find.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to build in this schema.

3. (Optional) In the **Name Contains** field, type one or more characters of the component's name.

4. (Optional) If you know the component's type (for example, chart, form, report), check the check box next to its type in the **UI Components** check box group and uncheck all other check boxes. You can also choose a **Component Status** to narrow your search. For example, you can choose **Archive** to search for old versions of the component, and **Production** to search for the most recent version.

5. Click **Find**. A list of components that match your search criteria appear at the bottom of the page.

6. Click a name in the **Component Name** column. The Manage Component page for the component appears.

7. Click **Manage**.

8. Click **About**. A page displays stored attributes and other information about the component.

Building Components and Objects

About component build wizards

WebDB provides build wizards that guide you step by step through the process of creating a component. For most build wizards, these steps include choosing:

- The schema that will own the finished component.
- A database object such as a table, view, or procedure on which the component will be based.
- The data that will display in the component.
- A display format for the data.
- A look and feel for the component.

Each wizard step contains buttons that you can use to navigate through the steps in the wizard. You should use these buttons to navigate through the wizard, not your Web browser's Forward and Back buttons. Click  to display the next step and  to display the previous step. A step is complete only if you click  to navigate to the next step or the last page of the wizard. Your selections are saved as you navigate from step to step.

As you complete the steps, WebDB automatically builds the SQL statement that selects data used in the component. You don't have to complete every step in the wizard. Each build wizard contains  (fast finish) buttons that allow you to create the component based on the information already collected by the wizard. The buttons appear in the build wizard steps once the wizard has collected enough information to create the component. For some wizards, the button appears after only a few steps; for others, after more.

Clicking the  button, or the **OK** button on the last wizard page finishes the component and stores it in the database as a PL/SQL packaged procedure. You now have the option of running the component you created as is, or editing it. You edit a component using a collection of tabbed pages that allow you to reselect options you originally selected in the wizard. If you clicked the  button, WebDB created the component using the default values for options on all the wizard steps you didn't complete to build the component. You can now go back and change any of these values using the tabbed edit pages.

Once you are ready for other users to run the component, you only need to provide them with the URL that contains it and privileges to execute it.

What are schemas?

A schema is a named collection of components and database objects. A schema name is the same as the user name.

Component Schemas own the components built by WebDB Developers. Object schemas typically own the objects, for example, tables and views, used to build the components.

When you build a component or object in WebDB, you must specify the name of a Component Schema that will own the finished component or object. A WebDB user with the DBA role must grant you Build In privileges to build components in this schema.

When you identify to a WebDB wizard the Component Schema you are planning to build in, you are given access to all the objects to which the schema has been granted privileges. For example, if you are building a component in the FRED Component Schema, and FRED has been given SELECT, INSERT, UPDATE and DELETE privileges on the SCOTT.EMP table, you are given the same privileges on SCOTT.EMP as FRED.

You can search the database for objects within each database schema you have privileges to Browse In. You can search for components within each schema you have privileges to Build In.

What are building privileges?

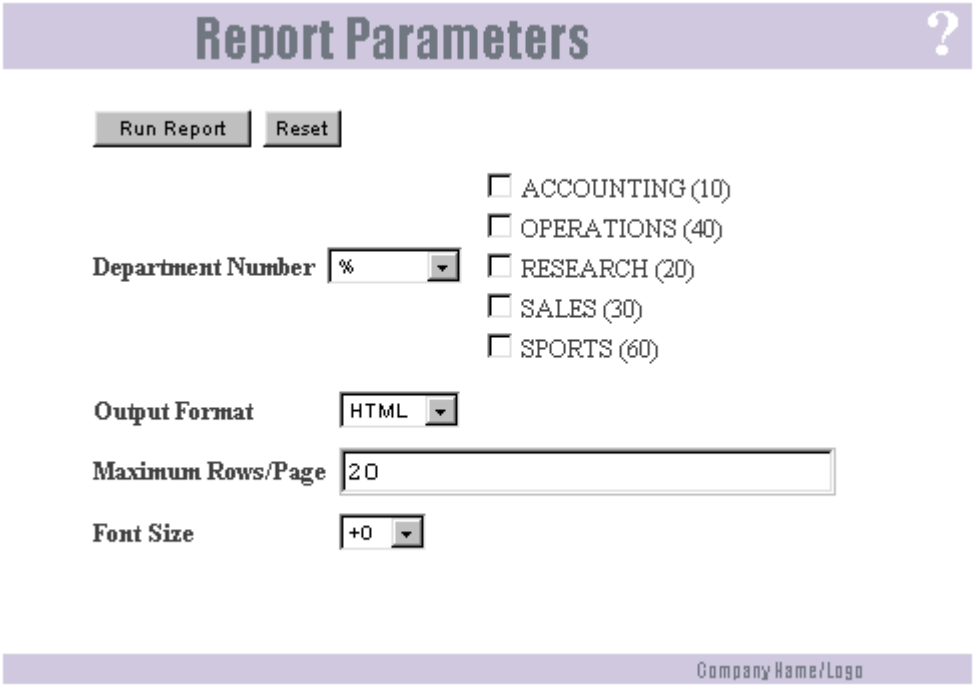
Before you can build a WebDB component, you must have *Build In* privileges for the Component Schema that will own the completed component. Build In privileges allow you to create a component within the specified database schema. In addition, you inherit any object privileges currently granted to the schema. For example, if you are building a component in the FRED Component Schema, and FRED has been given SELECT, INSERT, UPDATE and DELETE privileges on the SCOTT.EMP table, you are given the same privileges on SCOTT.EMP as FRED.

WebDB provides a report that allows you to view your current Build In and Browse In privileges.

A user with the DBA role can grant or revoke Build In privileges using the Privileges tab of the User Manager.

What are parameter entry forms?

Parameter entry forms prompt end users for values that are used to display WebDB components. For example, you could build a report based on the SCOTT.EMP table that displays information for all employees in an organization. As shown below, the component's parameter entry form might allow an end user to choose which departments to display in the report: Accounting, Operations, Research, and so on.



You can create parameter entry forms for charts, reports, calendars, and hierarchies. The build wizards for these components have a Parameter Entry Form Display Options page that you can use to specify which parameters you want to add to the parameter entry form. For each column in the table or view on which you base the component, you can add a parameter entry field that enables the end user to choose which column data to display.

If you code your own SQL query to create a component, you specify bind variables in the SQL query that builds your WebDB component. WebDB adds a field to the parameter entry form for each database table column that you associate with a bind variable.

As shown in the above graphic, you can add other options to the parameter entry form; for example, **Output Format**, **Maximum Rows/Page**, and **Font Size**. The Parameter Entry Form Display Options page lets you choose which options to expose to the end user of the parameter entry form.

You can choose to generate a parameter entry form as part of the package that is created when you complete the wizard steps for building a WebDB component. For example, if you include a parameter entry form when you build a chart named Employee, WebDB creates a package containing two stored procedures: `Employee Chart.Run Chart Parm` and `Employee Chart.Run Chart`. Running the first stored procedure displays the parameter entry form. Running the second displays the component.

End users can run also the parameter entry form by clicking the **Parameters** option on the Manage Component page.

Building components

To build a WebDB component:

1. On the bottom of any WebDB page, click the button that corresponds to the type of component you want to create. For example, click  to create a form.

2. The Building page for the component type you selected appears. Click the **Create** button.

   Note On some Building pages, you must choose a component subtype before clicking **Create**. On the Form Building page, for example, you have the option of creating a Master-Detail form, a Query by Example form, a form based on a table, or a form based on a procedure.

3. The first page of the component build wizard appears. Follow the instructions shown on each page of the build wizard until the last page. You can also click a fast finish button  on some pages to finish the component without completing the entire wizard.

Editing components

 To edit a component, you must have Build In privileges in the schema that owns the component.

To edit a component:

1. At the bottom of any WebDB page, click the type of the component you want to edit. For example, click  to edit a report.

2. In the **Find in Schema:** list, choose the schema that owns the component, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Browse In privileges in the schema. You must ask a user with the DBA role for privileges to browse this schema.

3. Click the name of the component in the **Component Name** column of the component's Building page.

4. Click  . The Edit tabbed dialog for the component displays.

   Note Clicking  opens the Edit tabbed dialog for the most recent component version. If you want to edit an archived version of the component, click one of the archive versions in the **Versions(s) Status** field on this page. When you save an archived version, it becomes the most recent component version.

5. Click one or more tabs to edit the values in the entry fields on the corresponding tab page. To access help about the entry fields on the page, click the small  button.

6. When you finish editing entry field values on all tabs you want to change, click **Finish**.

The new version of the component displays as a production version in the **Version(s) Status** field of the Manage Component page. The version of the component you opened to edit now displays as an archived version.

7. (Optional) If you decide you don't want to save your edits, click **Cancel**.

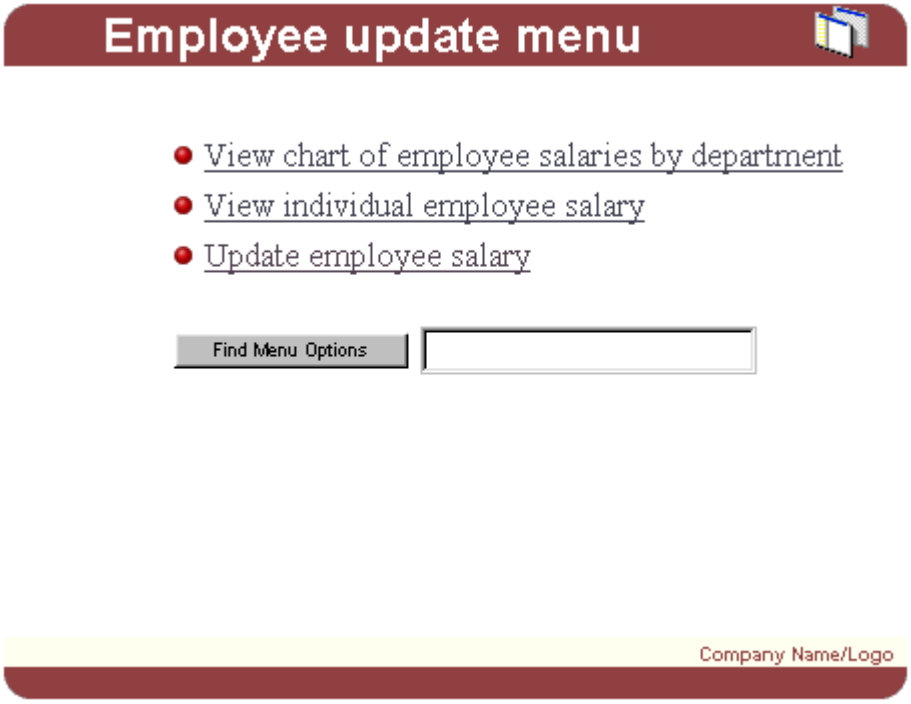
Notes

- As you switch from tab page to tab page, WebDB keeps track of any changes you make in the tabbed pages. Click **Finish** only after you've made changes on all the tabs pages you want to make.

- You edit all components except menus using tab pages. You edit menus using a hierarchical representation of your menu and any menus related to it.

Creating a Web application using WebDB components

After you create components in WebDB, you can link them together to form an entire Web application. One way to do this is to create a WebDB menu that displays hypertext links to one or more components; for example:



- `View chart of average employee salaries by department.`
  Links to a WebDB chart with this information.

- `View individual employee salary.`
  Links to parameter entry form for a report. An entry field on the form asks the end user to enter an employee name to view a report of the employee's salary history.

- `Update employee salary.`
  Links to a submenu containing separate options for updating salary, updating a commission percentage, or awarding a bonus.

After you create the menu and its links, you can restrict user access to components and submenus using roles.

In addition to menus, you can use WebDB Links to tie components together. Links add hypertext jumps between components. For example, the salary history report described above could contain hypertext links from the employee name in the report to a form for updating the employee's salary.

About setting the look and feel of a component

Each component build wizard contains a step that allows you to select options that determine the look and feel of component-specific features. For example, you can set colors and fonts for report headings, label fonts for boxes that appear in a hierarchy, and the width and color of bars on a chart.

In addition, the build wizard allows you to select a UI template that sets an overall look and feel for the page on which the component appears. UI templates control page background elements such as page titles, images that appear in the title, and background colors and images. A single UI template can be applied to any WebDB component.

WebDB templates set look and feel options in addition to those you set in the component build wizard. The template does not override any wizard options you select.

Using functions as default values for components

You can use functions to generate default values in the entry fields for forms and component parameter entry forms. You can specify these on any page of a component build wizard that contains an option for specifying a default value in the finished form or parameter entry form.

For example, you can specify a default on the Column Formatting and Validation page of the Forms (from Tables) component build wizard. In the **Display & Validation: Default Value** field on this page, you could type `#owner.package-name.nextval` for the EMPNO column of the SCOTT.EMP table where:

- # is a required character that you must type before specifying a function in an entry field on a component build wizard page.
- `owner` is the schema that owns the function. You must always prefix all user-defined functions with the owning schema. If you don't own the function, make sure you have privileges to execute it from its owner.
- `package-name` is the package containing the function.
- `nextval` is the name of a user-defined function.

The name.nextval function could display the next employee number from the EMPNO column each time the form is displayed.

You could specify a default value in a report's parameter entry form using the Column Conditions page of the Report Build wizard. On this page, specify

- `DEPTNO` as the **Column** for a report based on SCOTT.EMP
- = as the **Condition**
- `#owner.get_default_dept` as the **Value**

The report's parameter entry form could display an entry field with a default value that matches the next department from the EMPNO column each time it displays.

About components you can build

You use a build wizard to build a WebDB component such as a chart, form, or report. WebDB build wizards create HTML Web pages with content based on data stored in the Oracle database. The build wizard produces a PL/SQL stored procedure that is stored in the database. When executed, the stored procedure dynamically renders the HTML and JavaScript code that displays the component.

Some components display database data in a graphical format on a Web page. The build wizard for the component helps you select which data to display using a SQL SELECT statement. For example, you can create a report that displays on a Web page the first and last names of all employees in a department's database table along with their employee and department ID numbers.

Other WebDB components provide interfaces that allow users to change data in database objects. A Query by Example form, for example, can insert or update the employee first name, last name, and IDs in the same table you used to create the report.

These are the WebDB components you can build:

	Report	Displays data you select from the database table or view in a tabular format.
	Chart	Displays data you select from the database table or view as a bar chart.
	Calendar	Displays data you select from the database table or view as a calendar.
	Dynamic Page	Displays dynamically generated HTML content on a Web page.
	Hierarchy	Displays data you select from the database table or view as a graphical hierarchy of items containing up to three levels.

	Query by Example Form	Displays a form with entry fields for querying, deleting, updating, or inserting rows into a database table or view.
	Master-Detail (MD) Form	Displays a form that displays a master row and multiple detail rows within a single HTML page. The form contains fields for updating values in two database tables or views.

	Form (based on a table or procedure)	Displays a customized form that can be used as an interface for updating tables, executing stored procedures, and generating other customized forms.
	Menu	Displays an HTML-based menu containing hyperlinked options to other menus, WebDB components, or URLs.
	Frame Driver	Displays a Web page with two frames. End user queries in one frame control the contents of the other frame.

About calendars

You build a calendar using the calendar build wizard. The wizard takes the results of the SQL query that you supply and displays it in calendar format. The calendar shows data according to dates you specify using the SQL query. For example, you can create a calendar that displays the names of employees on the date each is due for a performance evaluation.

You can create hypertext links from values within the calendar to other WebDB components. In the previous example, you could add a hyperlink from each employee name to a form that allows the end user to update information about the employee such as employee id number and salary.

You can specify bind variables in your SQL query to create a parameter entry form that prompts end users for values to display in the chart.

The calendar build wizard provides options for setting the look and feel of the finished calendar using a UI template. For example, you can control calendar text fonts and background colors.

To view an example of a finished calendar:

1. Click  at the bottom of any WebDB page
2. Choose `SCOTT` in the **Schema** list on the Find an Existing Component page.

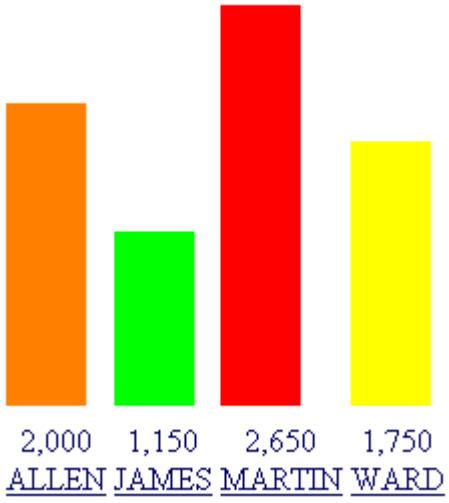
   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.
3. Type `EXAMPLE_CAL` in the **Name Contains** field and click **Find**.
4. Click **EXAMPLE_CAL** in the **Component Name** list at the bottom of the page.
5. Click **Run**.

About charts

You build a chart using one of two build wizards:

Chart from Query Wizard	Guides you through all steps for creating a chart, including creating the SQL query that selects the data displayed in the chart. If you are unfamiliar with SQL or are new to WebDB, you may want to create your chart using this wizard.
Chart from SQL Query	Allows you to write your own SQL query that selects the data displayed in the chart. After you write the query, the wizard allows you to set the same display, text, parameter entry form, and PL/SQL options that you can set in the Chart from Query Wizard.

The wizards displays the results of the SQL query in a bar chart. The example below shows a chart based on the SCOTT.EMP table that displays employee names as labels and chart bars representing each employee's salary.



Charts are based on at least two table or view columns:

- Values in the **Label** column identify the bars on the chart; for example, ALLEN, JAMES, MARTIN, etc.
- Those in the **Value** column calculate the size of the bars on the chart relative to one another. Value columns must always contain numeric data.

You can also specify a **Link** column. Values in this column create hypertext links from the chart's labels to other WebDB components or URLs. The above example contains links from the employee name labels to another component; for example, a form that allows an end user to update the employee's salary.

Charts are an effective way of displaying aggregate data. You can use **Group By** and **Summary Options** in the wizard to sum the column values returned by your query; for example, the minimum, maximum, or average value; or the number of values in a column.

To create a parameter entry form that prompts end users for values to display in the chart, use a bind variable in your SQL query or options on the Parameter Entry Form Display Options page of the Chart from SQL Query wizard.

You can set options to control the size and color of chart bars, label fonts, and background colors. The final chart can be displayed on a Web page or downloaded in SYLK format to Excel. You can also provide end users of the chart with the option to execute charts based on large amounts of data in batch mode rather than real-time.

To view examples of charts:

1. Click  at the bottom of any WebDB page.

2. Choose `SCOTT` in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. To view an example of a chart created using the **Chart from Query Wizard** option, type `EXMPLE_WIZ_CHART` in the **Name Contains** field.

To view an example of a chart created using the **Chart from SQL Query** option, type `EXAMPLE_SQL_CHART`.

4. Click **Find**.

5. Click the name of the chart in the **Component Name** list at the bottom of the page.

6. Click **Run**.

About dynamic pages

You build a dynamic page using the dynamic page build wizard. The wizard allows you to create Web pages that display dynamic content. You use the wizard to specify any HTML code that creates a Web page. You can type your own HTML or copy it from another Web page editor. After this, you add code enclosed in <ORACLE> </ORACLE> tags, whose executed results display on the page. The code can be a SQL statement or a PL/SQL block.

When an end user requests the Web page, its content is generated from current data in the database, based on the code you specified between the <ORACLE> </ORACLE> tags. The code automatically executes every time an end user requests the page.

For example, you can create a page that automatically executes a SQL SELECT statement on a database table, then displays the query results in the page. You can also set up pages to automatically execute functions and procedures and display the results on the page.

To view an example of a dynamic page:

1. Click  at the bottom of any WebDB page.

2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. Type `EXAMPLE_DYNAMIC_PAGE` in the **Name Contains** field and click **Find**.

4. Click **EXAMPLE_DYNAMIC_PAGE** in the **Component Name** list at the bottom of the page.

5. Click **Run**.

About forms based on tables, views, and procedures

You build a form using the forms build wizard. The wizard allows you to create custom forms that allow end users to insert, update, or delete data contained in tables or views. User access to table or view columns can be restricted by your choice to include on the form entry fields for updating some table columns, but not others.

You can also design a form to call a procedure, and pass to it parameters selected by the user on the form. This is especially useful for complex stored procedures that accept many parameters.

Using the wizard, you can create forms with multiple entry fields, control the size of the entry fields and their default values, and include selectable Lists of Values. The wizard provides options for performing JavaScript validation on user-specified values entered in any text field on the form. You can also choose the order in which fields representing table columns display on the form.

Other options allow you to select an overall look and feel for the form based on a UI template, as well as set the appearance of individual form elements such as labels and headings.

To view examples of forms:

1. Click  at the bottom of any WebDB page.

2. Choose `SCOTT` in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. To view an example of a form based on a table, type `EXAMPLE_FORM` in the **Name Contains** field.

   To view an example of a form based on a procedure, type `EXAMPLE_SP_FORM`.

4. Click **Find**.

5. Click the name of the form in the **Component Name** list at the bottom of the page.

6. Click **Run**.

About Master-Detail forms

You build a master-detail form using the forms build wizard. The Master-Detail form wizard creates a form that displays a master row and multiple detail rows within a single HTML page. With this form, users can query, insert, update, and delete values from two tables or views. The tables or views are joined together in the wizard by a SQL JOIN condition.

The wizard creates three pages:

- A Master Row Finder page that allows the end user to query the master table for a master row
- A Master Row Finder Results page that displays the query results. The end user can click a hypertext link to update a master row and its associated detail rows.
- The Master-Detail form, containing the selected master row and related detail rows.

Values in the master row determine which detail rows are displayed for updating. For example, the master row could display column values from the SCOTT.DEPT table. This table joins to SCOTT.EMP by a column in each table called DEPTID. Selecting a department ID in the master row displays in detail rows all Employees having the same Department ID.

The end user of the Master-Detail form can update or delete values in the master row. The wizard provides an option to allow cascading deletes, so that deleting a master row also deletes detail rows. The same Master-Detail form can be used to insert new detail rows, or update and delete existing detail rows.

The wizard allows you to choose a different look and feel for the Master Finder page and Master-Detail form. In addition, you can specify different formatting for master and detail rows on the Master Detail form. For example, you can choose the order in which master and detail table rows are displayed on the form, add a selectable List of Values, and perform JavaScript validation on user-specified values in any entry field on the form.

To view an example of a Master-Detail form:

1. Click  at the bottom of any WebDB page.

2. Choose `SCOTT` in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. Type `EXAMPLE_MD_FORM` in the **Name Contains** field and click **Find**.

4. Click **Example_MD_Form** in the **Component Name** list at the bottom of the page.
5. Click **Run**.

About Query by Example forms

You build a Query by Example form using the forms build wizard. The wizard creates a form that allows end users to query or insert values into a database table or view. You can create a query-only form, or a form that allows end users to both query and update the table or view.

The Query by Example form contains entry fields that correspond to the columns in the database table or view on which the form is built. You can restrict user access to table columns by choosing to include in the form entry fields for updating some table columns, but not others. You can also specify default values and present to the user a selectable List of Values for each entry field.

Other Query by Example options allow end users of the form to:

- Sum column values
- Use logical operators to select data within the columns
- Choose the format and method for displaying the query results
- Run the report in batch mode or real-time

You create a Query by Example form to allow end users to query, update, and delete table and view rows. If you want end users to query but not update table or view data, create a parameter entry form for a report. The report wizard provides options and instructions that help you do this. End users of the parameter entry form can query but not update the data in the table or view on which the report is based.

To view an example of a Query by Example form:

1. Click  at the bottom of any WebDB page.

2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. Type `EXAMPLE_QBE` in the **Name Contains** field and click **Find**.

4. Click **EXAMPLE_QBE** in the **Component Name** list at the bottom of the page.

5. Click **Run**.

About frame drivers

You build a frame driver using the frame driver build wizard. The wizard allows you to create Web pages with two frames. You can add to one frame (driving frame) a SQL query that drives the contents of a second frame (target frame).

The contents of the target frame can be:

- HTML code or text
- PL/SQL code
- A URL

For example, you could create one frame that contains a form with a list of all names in the SCOTT.EMP table. When a user selects a name from the list, the other frame might display a form for updating information in the SCOTT.EMP table about the selected employee; for instance, salary, department, or location.

To view an example of a frame driver:

1. Click  at the bottom of any WebDB page.
2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.
3. Type `EXAMPLE_FRAME` in the **Name Contains** field and click **Find**.
4. Click **Example_Frame** in the **Component Name** list at the bottom of the page.
5. Click **Run**.

About hierarchies

You build a hierarchy using the hierarchy build wizard. The wizard creates a graphical hierarchy that displays data selected from tables or views. For example, you can create a hierarchy that displays an employee organization chart, or the hierarchical relationship between menus in a Web site.

The table or view on which the hierarchy is based must be self-referencing; that is, at least two columns in the table must share a recursive relationship. The SCOTT.EMP table has a recursive relationship between values in the MGR column and those in the EMPNO column. An employee's manager is shown in the table by the manager's employee number. To create a hierarchy based on this table, you identify to the build wizard MGR as a Foreign Key column and EMPNO as the Parent Key column.

The hierarchy can contain up to three levels. End users click buttons in the finished hierarchy to navigate up and down levels. For example, you can create an organization chart that allows end users to drill down to sales personnel and drill up to managers or farther up to vice presidents. As the end user drills up the hierarchy, new data from the table appear on the topmost or parent level.

You can add hypertext links to the hierarchy. For example, employee names in an organization chart could be hyperlinked to reports that provide additional information about the employee such as employee ID or salary.

You can create a parameter entry form for the hierarchy that allows end users to specify values that will be used to display the hierarchy. For each entry field on the form, you can add a selectable List of Values, and perform JavaScript validation on user-specified values.

The wizard allows you to set an overall look and feel for the hierarchy using a UI template, as well as set the appearance of individual component elements using options in the hierarchy wizard. For example, you can control text font and colors as well as background colors that call attention to text on different levels.

To view an example of a hierarchy:

1. Click  at the bottom of any WebDB page.

2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. Type `EXAMPLE_HIERARCHY` in the **Name Contains** field and click **Find**.

4. Click **EXAMPLE_HIERARCHY** in the **Component Name** list at the bottom of the page.

5. Click **Run**.

About menus

You build a menu using the menu build wizard. The wizard creates HTML-based menus viewable from any Web browser. The menus contain hypertext links to submenus and links to WebDB components and URLs. The menu, its submenus and any links to WebDB components or URLs can be secured at the role level to prevent access by unauthorized users.

You can choose display on a single menu up to five levels of a menu hierarchy, with each level indented on the menu to the right. Descriptive text can be added to links, and hyperlinks to submenus that appear on menus. You can set a overall different look and feel for the menu and its submenus based on a UI template. Or, you set a different look and feel on a submenu by submenu basis

To view an example of a menu:

1. Click  at the bottom of any WebDB page

2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges for this schema. Ask your DBA for privileges to build in this schema.

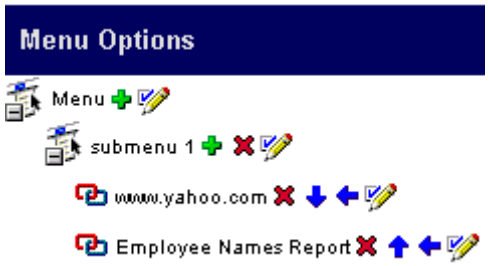
3. Type `EXAMPLE_MENU` in the **Name Contains** field and click **Find**.

4. Click **Example_Menu** in the **Component Name** list at the bottom of the page.

5. Click **Run**.

Building menus

WebDB provides a new way to build and edit menus. As shown in the graphic below, the Menu wizard now contains a Menu Options page that allows you to build a menu using a hierarchical representation of the menu's parts. The left frame of this page represents the hierarchy, as shown below.



There are three basic parts in a WebDB menu hierarchy:

- The topmost level of the hierarchy is called the *menu*. There is only one menu per hierarchy.
- A *submenu* is simply a menu called by another menu or submenu. There is no limit to the number of submenus in the hierarchy. A submenu appears as a hypertext link on its parent menu.
- *Menu items* also appear on menus and submenus. Instead of linking to a submenu, menu items jump to a WebDB component or another URL.

  Menu items appear as  icons in the hierarchy, whereas menus and submenus appear as folders ().

By selecting options based on the above hierarchy, you can create a menu similar to this:

When you edit an item in the hierarchy, the right frame of the Menu Options page displays a different set of options depending on whether you are editing a menu, submenu, or menu item.

To create a new menu:

1. Click  on the bottom of any WebDB page.

2. Click **Create**.

3. In the **Schema** list, choose the schema that will own the finished menu.

   Only schemas that you are allowed to build in display in the list. If a schema you want to build in is not listed, you must obtain Build In privileges for the schema from your DBA.

4. In the **Name** field, type a name for the stored procedure that will contain the menu. **Note** The **Name** will not appear on the menu itself. You can choose a name that appears on a banner in the menu on the next page of this wizard.

5. Click  .

   The Menu Options Page displays two frames. On the left side of the frame is the menu hierarchy. Since you have just begun creating the menu, there should be only one item listed in the hierarchy, a menu with the default name of `My Menu`.

6. (Optional)To change the name of `My Menu`, click  . The right frame of the page displays default options for `My Menu`.

   Type a new name in the **Name** field in the right frame. If you click  again, `My Menu` updates with the new name in the hierarchy on the left. The **Name** also appears in a banner at the top of the finished menu.

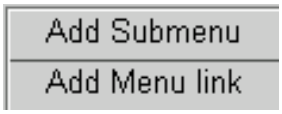
7. (Optional) You can choose other options in the right frame, then click  to apply them to `My Menu`. For example, you may want to restrict the number of users who can view the menu using the **Role Security** option, or choose a look and feel for the menu using the **Template** option. If you have questions about these or any other options on this page, click the small help  button in the upper right corner of the page.

To add a submenu to a menu:

8. Follow steps 1-7 above.

9. Click  next to the menu. The menu always appears on the topmost level of the menu hierarchy in the left frame of the Menu Options page.

   A pop-up menu asks you to create either a new submenu or menu item.



10. Click **Add Submenu.** The new submenu appears under the menu or submenu in the hierarchy.

11. (Optional) To choose options for the submenu, add a submenu or menu item below it, or perform other actions, click the folder  next to the submenu you just created.

    A series of buttons appears to the right of the submenu that indicates the actions you can perform on it.

Button	Allows you to:
	Insert a submenu below the current menu or submenu, **OR** add a menu item that appears on the current submenu. Clicking this icon opens a dialog box in which you to choose whether to add the submenu or menu item.
	Delete the current menu, submenu, or menu item. **Note** You cannot delete a menu (the top level of the hierarchy).
	Change the order in which the submenu or menu item appears on a menu or submenu.
	Move the submenu or menu item under a new parent in the hierarchy.

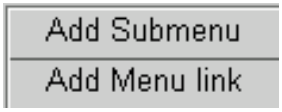
Click  to choose options for the submenu.

The right frame of the Menu Options page reloads with a different set of options depending on whether you are editing a menu, submenu or menu item. Any options you choose will be saved when you finish the wizard, or choose options for another menu or menu item in the hierarchy.

Click the small help button in the upper right corner of the page for information about any option on this page.

To add a menu item to a menu or submenu:

12. Click next to a menu or submenu in the hierarchy. A pop-up menu asks you to create either a new submenu or menu item.



13. Choose **Add Menu Link**. The new menu item appears under the menu or submenu in the hierarchy.

14. Click next to the menu item you just created. The right frame of this page reloads with menu item options.

15. In the **Link** field, specify either:

    - A link to URL.

      Type its location in the Link field; for example, `http://us.oracle.com`

    - A link to a WebDB component

      Type the name of the package containing the component in the **Link** field. For example, you can type `SCHEMA.COMPONENT.SHOW` where SCHEMA is the name of the schema that owns the component, COMPONENT is the component name, and SHOW is the procedure used to display the component. You can also specify SHOW_PARMS to display the parameter entry form for the component.

      OR

      Click next to the **Link** field to choose from a list of links to components that have been created in the Link Manage.

16. (Optional) You can choose other link options in the right frame, then click  to apply them to the link. If you have questions about any options on this page, click the small help  button in the upper right corner of the page.

To finish the menu:

17. Click  .

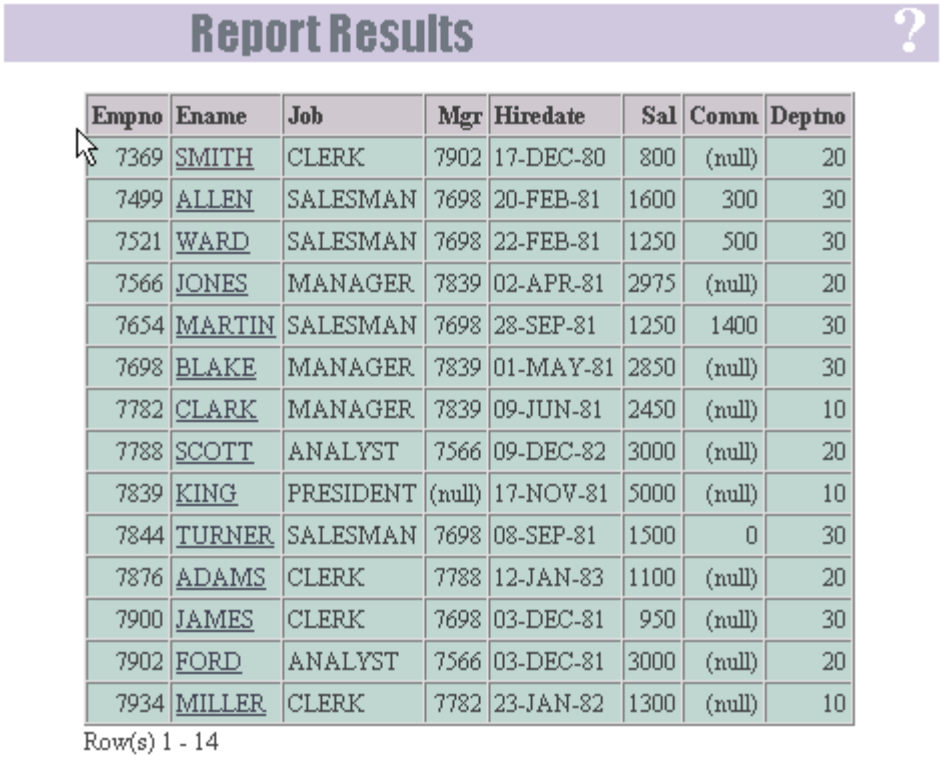
About reports

You build a report using one of two build wizards:

Report from Query Wizard	Guides you through all steps for creating a report, including creating the SQL query that selects the data displayed in the report. If you are unfamiliar with SQL or are new to WebDB, you may want to create your report using this wizard.
Report from SQL Query Wizard	Allows you to write your own SQL query that selects the data displayed in the report. After you write the query, the wizard allows you to set the same display, text, parameter entry form, and PL/SQL options that you can set in the Report from Query Wizard.

The report build wizard takes the results of a SQL query and displays it in a tabular format. Report headings correspond to the database table or view columns specified in the SQL query. You create a report to allow end users to query a table or view, but not update any of its data. If you want to allow updates of the table or view, create a Query by Example form.

The following report was created using several columns of the SCOTT.EMP table.



Report Results

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	(null)	20
7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100	(null)	20
7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10

Row(s) 1 - 14

You can specify options in the Report from Query wizard or a bind variable in the Report from SQL Query Wizard to create a parameter entry form. The parameter entry form prompts end users for values to display in the chart. For each entry field on the form, you can add a selectable List of Values, and perform JavaScript validation on user-specified values.

You can create hypertext links from values displayed in the report to other WebDB components or URLs. The above example contains links from the employee names to another component; for example, a form that allows an end user to update the employee's salary.

The wizards allow you to specify other options that:

- Limit data displayed in the report
- Specify JOIN BY conditions between two tables or views
- Format data in the report
- Format the report's parameter entry form
- Specify PL/SQL code that executes at various points in the report

To view examples of reports:

1. Click  at the bottom of any WebDB page.

2. Choose **SCOTT** in the **Schema** list on the Find an Existing Component page.

   Note If SCOTT is not listed, you don't have Build In privileges. Ask your DBA for privileges to build in this schema.

3. To view an example of a chart created using the **Report from Query Wizard** option, Type `EXAMPLE_WIZ_REPORT` in the **Name Contains** field.

   To view an example of a chart created using the **Report from SQL Query** option, type `EXAMPLE_SQL_REPORT`.

4. Click **Find**.

5. Click the name of the report in the **Component Name** list at the bottom of the page.

6. Click **Run**.

Writing SQL queries

About writing SQL queries in WebDB

WebDB uses a SQL query to select the table or view data displayed in a component. You have the choice of writing your own SQL query or allowing a build wizard to generate it for you when you build charts and reports. If you build a calendar or frame driver, you must code a SQL query.

Charts, calendars, and hierarchies all require WebDB-specific syntax in the SQL statement that creates these components. For example, you must identify in the chart SQL statement a table or view column that supplies labels for the chart. You also identify a column that contains numeric data to determine the size of chart bars. The wizards for building charts and other components contain instructions about how to specify any special syntax.

You can specify bind variables in the SQL query that allow the WebDB component to accept user input from a parameter entry form. The end user can then choose which data to display in the component. You can also specify in the SQL query hypertext links from column values displayed in the component to other components or Web pages. For example, you can link employee names on a department chart to individual reports containing information about each employee on the chart.

About joining tables

The Report and Master-Detail build wizards each provide a step that allows you to join columns from two tables. The step adds a JOIN condition in the WHERE clause of the SQL statement used to build the component.

If you want to use join data from multiple tables to create components other than Master-Detail forms or reports, you can edit the SQL query that the wizard uses to build the component. Type or add the JOIN condition for the tables in the SQL field.

Follow the same procedure for joining data from multiple tables owned by different schemas.

What are bind variables?

You use a bind variable to create an entry field in a parameter entry form for a component. Each bind variable corresponds to a column in the table or view on which the component is based. Users can enter values in the parameter entry fields to select the column data to display in the component.

You can add bind variables to a hand-coded SQL query when creating charts, reports, calendars, frame drivers, and dynamic pages. A bind variable appears in a SQL query as an alphanumeric string preceded by a colon (:var1, :var2, :var3,…). For example, the following SQL query creates parameters for SALARY and DEPT columns of the SCOTT.EMP table:

```
select ename, sal, dept

from scott.emp

where sal = :salary and deptno like :dept
```

Specifying this SQL query creates a parameter entry form with two text entry fields. The first allows the user to select a salary, the second a department number. When the user clicks a button on the parameter entry form to create the component, WebDB uses the values that the user typed in the entry fields.

You don't need to know SQL to specify bind variables. You can identify columns that will accept parameters in the **Column** field on the Parameter Entry Form Display Options page of several component build wizards.

If you specify a bind variable for a table or view column, you can also specify a List of Values for the column. Instead of prompting the user of the parameter entry form with a simple entry field to type values, a group of possible values for the entry field displays. If you specify a List of Values for the DEPT column shown in the previous example, the parameter entry form might contain a Department list with the selections Accounting, Operations, Research, and Sales. The user chooses one of these values in the list to specify which column values display in the component.

Calendar SQL query examples

The format for a calendar query is:

```
SELECT
the_date,
the_name,
the_name_link,
the_target
from schema.object
```

where:	Description
the_date (required)	Displays text on the calendar on the dates contained in this table or view column. Values in this column must have the DATE datatype.
the_name (required)	Displays cell text from this table or view column according to the dates in the_date column
the_name_link (optional)	Specifies a link from the values in the_name to another WebDB component or URL. If you don't want to display links from calendar names, specify `null` for this column. **Hint** You can create a link using the WebDB link wizard and copy to your SQL query, as shown in the example below. To copy a link, follow the steps for testing links .
the_date_link (optional)	Specifies a link from the values in the_date column to another WebDB component or URL. If you don't want to display links from calendar dates, specify `null` for this column.

the_target_frame (optional)	Specifies the URL of a frame in a Web page. Specify this column if you want to link to a specific frame in a URL. If you don't want to link to a target frame, specify `null` for this column.

Example Calendar SQL query with link to another component:

The following query creates a calendar that displays employee last names on the dates they were hired. Clicking an employee name displays the Example_RPT in the SCOTT schema. A section of the calendar is shown below.

```
select
EMP.HIREDATE the_date,
EMP.ENAME    the_name,
'SCOTT.EXAMPLE_WIZ_RPT.show?p_arg_names=emp.ename
&p_arg_values='||ename||'&p_arg_names=_emp_ename_con
d&'
the_name_link,
null         the_date_link,
null         the_target
from SCOTT.EMP
```

November 1981

Monday	Tuesday	Wednesday	Thursday	Friday
02	03	04	05	06
09	10	11	12	13
16	17 » KING	18	19	20
23	24	25	26	27
30				

December 1981

Monday	Tuesday	Wednesday	Thursday	Friday
	01	02	03 » JAMES » FORD	04
07	08	09	10	11
14	15	16	17	18
21	22	23	24	25
28	29	30	31	

Chart SQL query examples

The format for a chart query is:

```
SELECT
    the_link,
    the_name,
    the_data
from schema.object
```

where:	Description
the_link (optional)	Specifies a link from the values in the_name to another WebDB component or URL. If you don't want to display links from calendar names, specify `null` for this column. **Hint** You can create a link using the WebDB link wizard and copy to your SQL query, as shown in the example below. To copy a link, follow the steps for testing a link .
the_name (required)	Data from this table or view column appear as labels in the chart.
the_data (required)	Data from this column is used to calculate the size of the chart's bars. The column you choose must always contain numeric data.

Notes

- You don't have to specify the column names (the_link, the_name, the_data in your SQL query. For example, you can specify:

  ```
  SELECT null, ename, sal from SCOTT.EMP
  ```

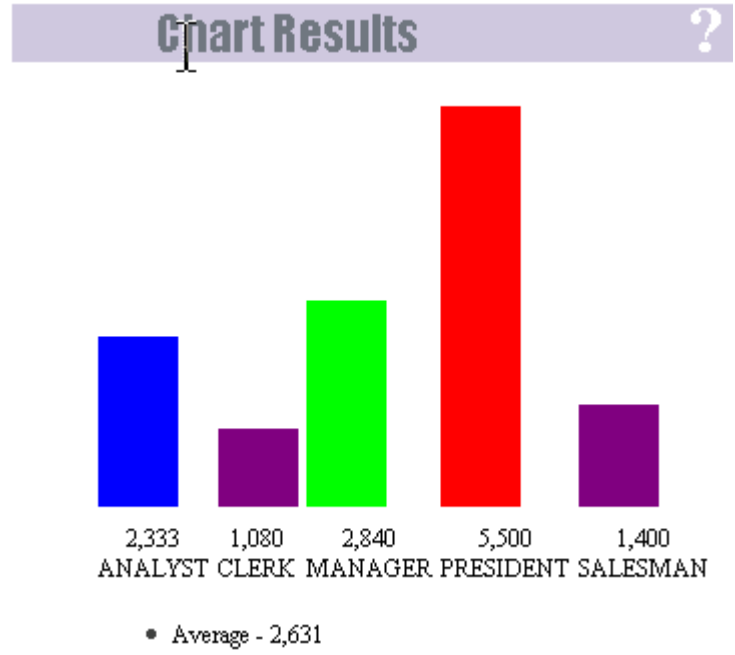
- Example SQL query using an expression (sysdate - hiredate). This chart shows the number of days from the employee's date of hire to today.

```
SELECT
null                the_link,
ENAME               the_name,
sysdate-hiredate    the_data,
from SCOTT.EMP
group by JOB
```

Example SQL query using GROUP by expression

The chart created by this query is shown below.

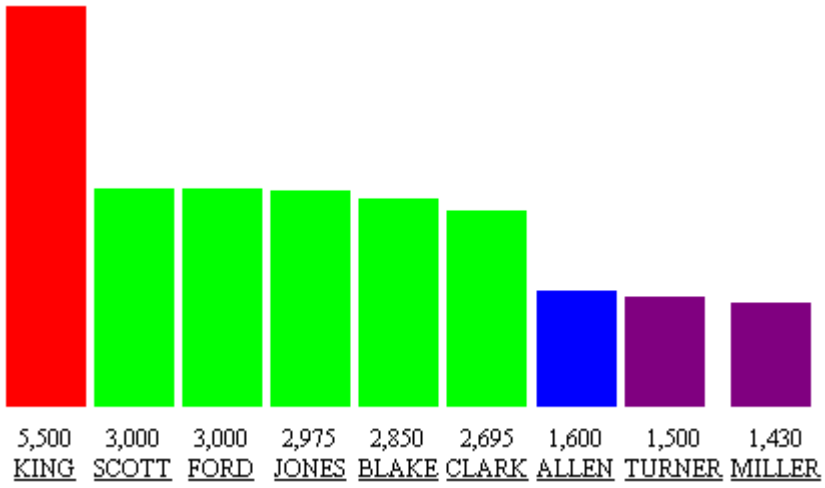
```
select
null          the_link,
JOB           the_name,
avg(SAL)      the_data,
from SCOTT.EMP
group by JOB
```



Example Chart SQL query with link to another component:

The following query creates a chart that displays employee last names along the bottom of the chart and their salaries. Clicking an employee name displays a report containing additional information from the SCOTT.EMP table. A section of the chart is shown below.

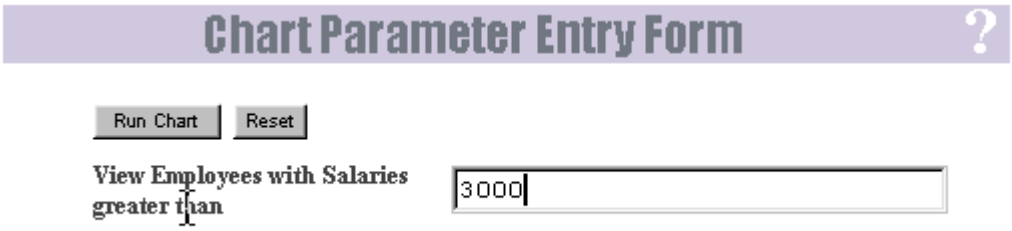
```
select
'SCOTT.EXAMPLE_WIZ_RPT.show?p_arg_names=emp.ename
&p_arg_values='||ename||'&p_arg_names=_emp_ename_con
d&'
the_link,
ENAME        the_name,
SAL          the_data
from SCOTT.EMP
order by SAL desc
```



Example Chart SQL query that uses a bind variable to create an entry field in a parameter entry form:

The following SQL query creates a simple parameter entry form for a chart. :SALARY is a bind variable that adds an entry field to the form. End users of the form can choose a value from the SAL column of SCOTT.EMP for displaying the chart. The parameter entry form is shown below.

```
select
  null  the_link,
  ENAME the_name,
  SAL   the_data,
from SCOTT.EMP
where SAL > :SALARY
Order by SAL desc
```



Dynamic pages SQL query examples

You can specify this HTML on the Dynamic Page Content page of the Dynamic Page wizard.

Example using function that dynamically generates the current date:

```
<html>
<body>
<h1>hello</h1>
It’s <oracle>
begin
htp.p(to_char(sysdate,’Day HH24:MI’))
end;
</oracle> is your data accessible?
</body>
</html>
```

Example using SELECT statement:

```
<html>
<body>
<h1>Employees in SCOTT.EMP</h1>
<oracle>select * from scott.emp
</oracle>
</body>
</html>
```

SQL-based Report examples

Example SQL-based Report query with a link to another component:

The following query creates a report that displays all employees from the SCOTT.EMP table. Values in the employee ID number column (EMPNO) link to Example_Form in the SCOTT schema. Note that the link is specified within an HTML anchor and that the anchor precedes the EMPNO column in the SQL query.

Hint You can create a link using the WebDB link wizard and copy it to your SQL query, as shown in the example below.

```
select
   '<A
REF="'||'SCOTT.EXAMPLE_FORM.show?p_arg_names=EMPNO&a
mp;
   p_arg_values='||empno||'"> '||empno||'</A>'
   empno,
   ename,
   deptno,
   mgr,
   sal
from SCOTT.EMP
```

This is the report created by the above query.

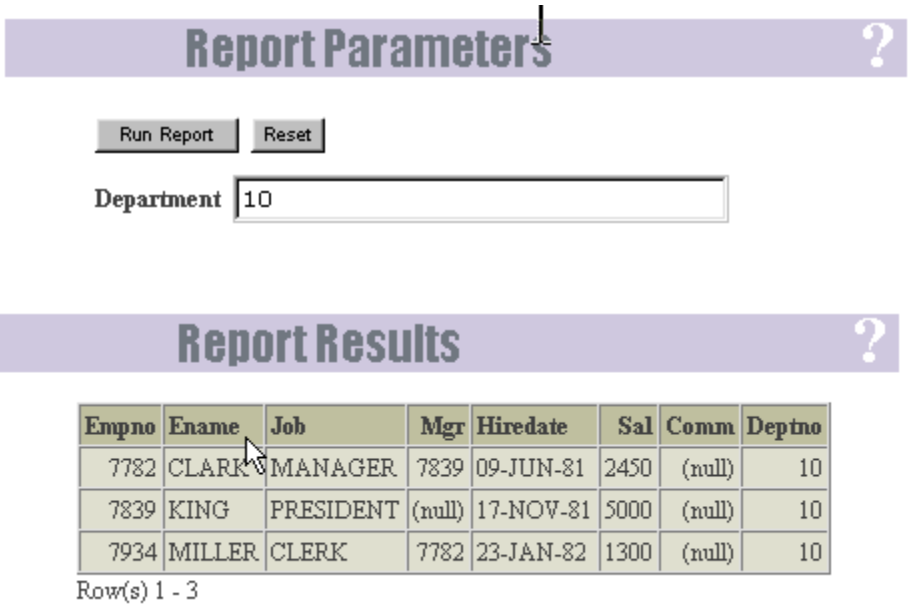
Empno	Ename	Deptno	Mgr	Sal
7369	SMITH	20	7902	800
7499	ALLEN	30	7698	1600
7521	WARD	30	7698	1250
7566	JONES	20	7839	2975
7654	MARTIN	30	7698	1250
7698	BLAKE	30	7839	2850
7782	CLARK	10	7839	2450
7788	SCOTT	20	7566	3000
7839	KING	10	(null)	5000
7844	TURNER	30	7698	1500
7876	ADAMS	20	7788	1100
7900	JAMES	30	7698	950
7902	FORD	20	7566	3000
7934	MILLER	10	7782	1300

Row(s) 1 - 14

Example Report SQL query that uses a bind variable to create an entry field in a parameter entry form:

The following SQL query creates a simple parameter entry form for a report. `:JOBS` is a bind variable that adds an entry field to the form. End users of the form can choose a value from the SAL column of SCOTT.EMP for displaying the chart. The parameter entry form and the report are shown below.

```
select deptno, ename, hiredate, job, sal, comm
from scott.emp
where job in (:jobs) or :jobs is null
```



Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10

Row(s) 1 - 3

After you code the SQL query, you can specify other options using pages in the Report from SQL Query wizard; for example, a List of Values for the **Department** entry field on the parameter entry form.

Managing WebDB components

About component execute privileges

Every WebDB build wizard contains a step that asks you to identify the database user account, also known as the schema, that will own the component built by the wizard. For example, if you log into WebDB as FRED and build a component in the FRED schema, you own the component.

By default, the schema that owns the component is granted privileges to execute, manage, edit, or drop the component from the database. If you have Build In privileges in this schema, you also have these privileges. For example, if you log in as SCOTT and have Build In privileges in FRED, you have the same privileges as FRED on any components owned by the FRED schema. In addition, you can grant execute privileges on a component owned by FRED to another user schema.

Use the Manage Component page to assign execute privileges that allow other schemas or roles to execute the component.

About editing components

You can edit a component if it is stored in your schema or in a Component Schema in which you have Build In privileges. If you don't have these privileges, you must ask a user with the DBA role to grant them to you.

You edit components using a collection of tabbed pages in a dialog box. Each tabbed page corresponds to a step in the wizard that created the component. Entry fields on each tabbed page contain the values that were specified during creation of the component, or by the user who last edited the component and saves the changes. In general, you can change any component build option except the table or procedure on which the component is based.

Components are locked while you edit them, preventing other users from making changes to the component. The component remains locked until you click the Finish or Cancel button on the Edit dialog box.

After you edit a component, it becomes the most recent production version. The version of the component you opened to edit is saved in the database as an archived version. The next time you edit the component, you can open the most recent production version or an older archived version.

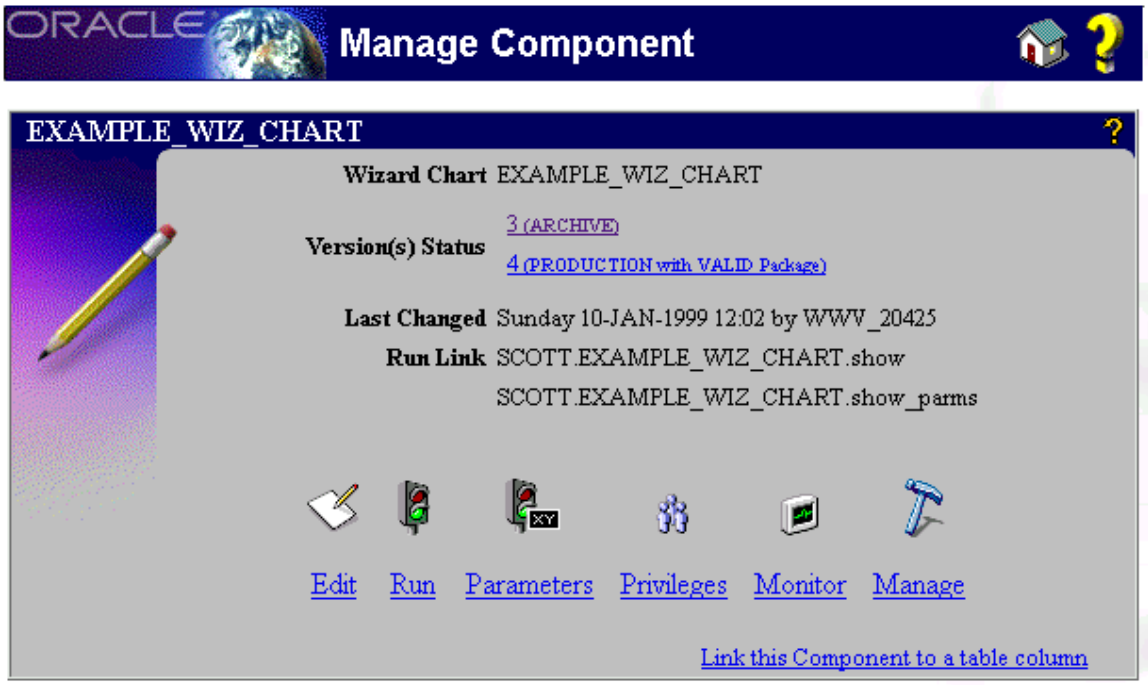
About component version status

WebDB allows you to store multiple versions of the same component in the database. For example, every time you edit and save a component, the previous version is automatically saved in the database as an archived version. The next time you edit the component, you can open the most current version or any archived version of the component.

You can delete archived component versions from the database at any time.

The most recent version of a component is called the production version. The production version can be valid or invalid, depending on whether the database package containing the component will run without errors. If a component has a version status of `(Production with INVALID package)`, you must edit the component to fix the errors before it will run.

You can view a component's version status on the Manage Component page, as shown below.


ORACLE
Manage Component
EXAMPLE_WIZ_CHART
Wizard Chart EXAMPLE_WIZ_CHART
Version(s) Status
3 (ARCHIVE)
4 (PRODUCTION with VALID Package)
Last Changed Sunday 10-JAN-1999 12:02 by WWV_20425
Run Link SCOTT.EXAMPLE_WIZ_CHART.show
SCOTT.EXAMPLE_WIZ_CHART.show_parms
Edit
Run
Parameters
Privileges
Monitor
Manage
Link this Component to a table column

Granting execute privileges on components

To grant privileges to run a WebDB component to other WebDB users:

1. At the bottom of any WebDB page, click the type of component on which you want to grant privileges. For example, click  to grant run privileges on a report.

2. In the **Find in Schema:** list, choose the schema that owns the component, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to build in this schema.

3. Click the name of the component in the **Component Name** column of the component's Building page.

4. Click  (Privileges).

   Note Don't click  on the navigation bar at the bottom of the page.

5. Review the names in the **Existing Grant: Grantee** column to ensure the user doesn't already have privileges to run the component.

6. Type the name of the user to whom you want to grant privileges in the **User/Role** field, then click **Grant Execute Privilege**.

   Click  to the right of the **User/Role** field to search for users and roles.

7. The user or role to whom you granted privileges appears in the **Existing Grant: Grantee** column, with a check mark next to it.

8. (Optional) To revoke an execute privilege from a user or role, uncheck the check box next to the user or role and click **Revoke**.

Viewing component versions

To view all versions of a component stored in the database:

1. Click  on the bottom of any WebDB page.

2. In the **Schema** list, choose the schema that owns the component you want to find.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to Build In this schema.

3. (Optional) Type one or more characters of the component's name in the **Name Contains** field.

4. (Optional) If you know the component's type (for example, chart, form, report), check the check box next to its type in the **UI Components** check box group and uncheck all other check boxes.

5. In the **Status** list box, choose **All status codes**.

   Choose another status code if you want to narrow your search.

6. Click **Find**. A list of components that match your search criteria appears at the bottom of the page.

7. (Optional) Click a name in the **Component Name** column. The Manage Component page displays all versions of the component in the **Version(s) Status** field.

8. (Optional) Click a component version to edit it.

Note

- If you click the edit button  , you automatically edit the most recent component version.

To delete a version of a component:

1. Follow steps 1-7 above.
2. Click  , then  .
3. Check the check box next to one or more versions of the component, then click **Yes**.

Note

- A component version status of (EDIT) indicates that another user is currently editing the component, and all versions of the component are locked. You cannot edit the component while it is locked.

Copying and renaming components

To copy or rename a component, you must have Build In privileges in the schema that owns the component. In addition, if you want to copy the component into a different schema, you must have Build In privileges in that schema.

To create a new copy of a component:

1. At the bottom of any WebDB page, click the type of component whose status you want to view. For example, click  to copy a form.

2. In the **Find in Schema** list, choose the schema that owns the component, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to browse this schema.

3. Click the name of the component in the **Component Name** column of the component's Building page.

4. Click  , then  .

5. In the **New Schema** list box, choose the schema that will own the new copy of the component.

   Note Only schemas in which you are allowed to build are listed in the box. If you want another schema to own the component copy, contact your DBA to obtain Build In privileges for that schema.

6. Type a new name for the component in the **New Component** field, then click **Copy.**

To rename a component:

1. Follow steps 1-3 above.
2. Click  , then  .
3. Type a new name for the component in the **New Component** field, then click **Rename**.

Note

- If you are copying a component to the same schema as the original, the **New Component Name** must differ from the **Current Component Name** on the Copy Component page.

Exporting components

You can export components to another instance of WebDB installed on a remote database. To do so, you copy SQL INSERT statements provided by WebDB and use these to insert components into tables associated with another instance of WebDB. You must have Build In privileges in the schema where the component is currently located and the target schema in the remote database.

To export a WebDB component from its current database to another database:

1. At the bottom of any WebDB page, click the type of component you want to export.

   For example, click  to export a dynamic page.

2. In the **Find in Schema** list, choose the schema that owns the component, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to Build In this schema.

3. Click the name of the component in the **Component Name** column of the component's Building page.

4. Click  , then  .

5. A page displays a SQL script that exports the component into a remote database.

6. Use options in your Web browser to copy and paste the SQL script code to an ASCII text file.

7. Save the file and export it to a schema in the remote database.

8. Run the SQL script using SQL*Plus in the remote database.

Viewing and changing component locks

 To view a lock on a component, you must have Build In privileges in the schema that owns the component. You must have the DBA role to change a lock.

A component lock prevents other WebDB Developers from overwriting your changes while you are editing a component. No other developer can edit the component when it is locked. Locked components display a version status of (EDIT) on the Manage Component page.

If you want to edit a component that is currently locked, you can view the name of the developer who has locked it. A user with the DBA role can override the lock.

To view the name of the user who has locked a component:

1. At the bottom of any WebDB page, click the type of component whose status you want to view. For example, click  to see if a developer has locked a form.

2. In the **Find in Schema** list, choose the schema that owns the component, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to Build In this schema.

3. Click the name of the component in the **Component Name** column of the component's Building page.

4. Click  , then **Show locks on this component**. A report displays showing all versions of the component, WebDB Developers who currently have locked any versions of the component, for how long, and other information about the component's versions.

To unlock a component:

1. Follow steps 1-4 above.

2. Click **Unlock** next to the component version in the **Action** column of the report.

Running (displaying) components


To run a component, you obtain execute privileges from its owner.

To run a component without parameters:

1. At the bottom of any WebDB page, click the type of component you want to run.

   For example, click  to run a chart.

2. In the **Find in Schema** list, choose the schema that owns the component you want to run, then click **Find**.

   Note If you are trying to find a component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to Build In this schema.

3. Click the name of the component you want to run in the **Component Name** column of the component's Building page.

4. Click  .

To run a component with parameters:

1. Follow steps 1-4 above.
2. Click  .

   The parameter entry form for the component displays.

3. Specify the values in the parameter entry form that you want to use to display the component.

4. Click the Run button. The text on the run button may differ depending on the options that the component's developer specified when creating it.

Notes

- If an error occurs when you run the component, check the **Version(s) Status** field of the Manage Component page to ensure there is a valid version of the component. If a valid version exists, the **Versions** field contains the text `(PRODUCTION with VALID PACKAGE)`. The Manage Component page is the same page that contains the  button.

- When you run a component, you may notice that a  help button displays on the same page as the component. The help button is part of the template that was used by the developer to create the component. It is not part of the W enable end users to scroll through table or view rows returned by the query one row at a time.
- WebDB help system. Clicking the button displays help text if the developer created help text for the component (using the Text Options page of the component build wizard). If the developer did not create help text, clicking  will not link to a page containing help text.
- After the component or parameter entry form displays, use your Web browser's Back button to return to WebDB.

Running components in batch mode

What are batch jobs?

WebDB Developers can add to component parameter entry forms options that allow end users to generate the component in batch mode. These include the parameter entry forms for charts, reports, forms based on procedures, Query by Example forms, and calendars. Batch processing is useful if the component is based on a large amount of data, or if you anticipate that the component may display many rows of data. In addition, you may want to execute a component in batch mode to save the results in the database as so that other users can view them.

When you execute a job in batch mode, WebDB displays a page indicating that the job was submitted to the batch queue, and a number that identifies the job; for example:

Your Job (2433) has been submitted to the batch queue.

Use this number to identify all jobs currently executing or in the queue waiting to be executed. If you decide to terminate an executing job or remove it from the queue, you need to specify the job number.

When a job has finished executing, it is stored in the database as stored results. The **Batch Results Manager** allows you to track jobs in the batch queue as well as search the database for stored results.

The end user who originally executed the component becomes the stored results owner. Stored results owners can use options in the **Batch Results Manager** to:

- Change stored results names.
- Change the stored results expiration date. This is the date when the stored results are automatically dropped from the database.
- Designate a public or private viewing status for the results. Public status means all WebDB users can access the stored results in the Batch Results Manager and view them. Only the owner can view stored results having a private status.
- Delete stored results.

Note

- The DBA may need to adjust some parameters in the `init.ora` file for the Oracle database in order to set up batch processing.

Setting init.ora parameters for batch jobs

This topic is intended for DBAs who manage the Oracle database in which WebDB is installed.

To enable WebDB end users to execute jobs in batch mode, the DBA should review the parameters in the init.ora file for the Oracle database. Here are some suggested settings:

init.ora parameter and setting	**Specifies**
job_queue_processes=2	Two background processes.
job_queue_intervals=60	The processes wake up every 60 seconds
job_queue_keep_connections =TRUE	Sleep, don't disconnect

If these parameters aren't set correctly, batch jobs may remain in the batch queue but not run.

For more information, refer to your Oracle documentation.

Stopping submitted batch jobs

When you execute a job in batch mode, WebDB displays a page indicating that the job has been submitted to the batch queue, and a number that identifies the job. Use this number to stop a job you have submitted to the batch machine queue.

To stop a currently executing batch job or one in the queue waiting for processing:

1. Click  on the bottom of any WebDB page.

2. Click **Batch Results,** then **Batch Results Manager.**

   The Stored Results page appears. All batch jobs that are currently queued or executing display in the **Queued Requests** list on the page.

3. Click **Remove** next to the batch job you want to stop in the **Queued Requests: Action** column.

Viewing stored results

To view the stored results of a component that has been executed in batch mode:

1. Click  on the bottom of any WebDB page.

2. Click **Batch Results,** then **Batch Results Manager**.

3. In the **Find Stored Results: User** field, type the name of the user who owns the stored results. The user who originally executed the component in batch mode is the stored results owner.

   Note You can view other users' stored results only if the user has designated them PUBLIC.

4. In the **Program** field, type the name of the stored results program name. The program name is the same name as the procedure used to display the component; for example, `MY_CHART.SHOW_PARMS`. WebDB uses the program name to identify stored results.

5. Click **Find.**

   A page appears displaying all stored results that match your search criteria.

6. In the **Action** column, click **View** next to the stored results you want to display.

Notes

- You can use the % wildcard in the **User** and **Program Name** fields.

- You can specify search criteria in the **User** field, **Program Name** field, or both.

Making stored results public

Owners can change stored results properties such as their name, the date when they expire, and whether other WebDB users can view them. You are the owner of stored results if you created them by successfully executing a component in batch mode.

To change the stored results program name, expiration date, or public designations of stored results:

1. Click  on the bottom of any WebDB page.

2. Click **Batch Results**, then **Batch Results Manager.**

3. In the **User** field, type your user name (the user name you use to log into WebDB).

   Note You can view other users' stored results if they have been designated public. To do so, type the user's name in the **User** field.

4. In the **Program Name** field, type the name of the stored results program name. The program name is the same name as the procedure used to display the component; for example, `MY_CHART.SHOW_PARMS`. WebDB uses the program name to identify stored results.

5. Click **Find.**

   A page appears displaying all stored results that match your search criteria.

6. In the **Action** column, click **Edit** next to the stored results you want to display.

7. (Optional) In the **Program Name** field, type a new program name for the stored results.

8. (Optional) In the **Document Expires In** field, type a new current expiration date (in days from the current date) for the stored results.

9. (Optional) In the **Is Public** field, choose whether the stored results are public (viewable by any WebDB user) or private (viewable only by you).

10. Click **Apply** to update properties of the stored results based on the options you specified on this page.

Notes

- You can use the % wildcard in the **User** and **Program Name** fields.
- You can specify search criteria in the **User** field, **Program Name** field, or both.

Building Oracle database objects

About Oracle database objects you can build

WebDB provides wizards for creating Oracle database objects. You can also build objects using standard Oracle database commands.

These are the types of Oracle database objects you can create in WebDB:

Database object type	Button	Description
function		Creates a standalone stored function containing a set of PL/SQL statements.
index		Creates an index on one or more table columns.
package		Creates a specification for a stored package and a package body.
procedure		Creates a standalone stored procedure containing a set of PL/SQL statements.
sequence		Creates a database object that is used to generate unique integers.
synonym		Creates an alternate name for a function, table, view, sequence, procedure, package, snapshot, or another synonym.
table		Creates a database table.
trigger		Creates a stored procedure associated with a table. The trigger automatically executes when a specified SQL statement is issued against the table.
view		Creates a logical table based on one or more tables.

Building database objects

To build a database object such as a table, procedure, or trigger:

1. Click  on the bottom of any WebDB page.
2. Click a button corresponding to the type of object you want to create. For example, click  to create a table.
3. The first page of the object build wizard appears. Follow the instructions shown on each page of the build wizard until the last page.

Building Shared Components

What are shared components?

Shared components are building blocks that can be used by you and other developers to create WebDB components. Shared components includes links, Lists of Values, JavaScripts, and look and feel elements that define the appearance of a component. Each component build wizard allows you to optionally add one or more shared components to the component. These shared components become part of the final component created by the wizard.

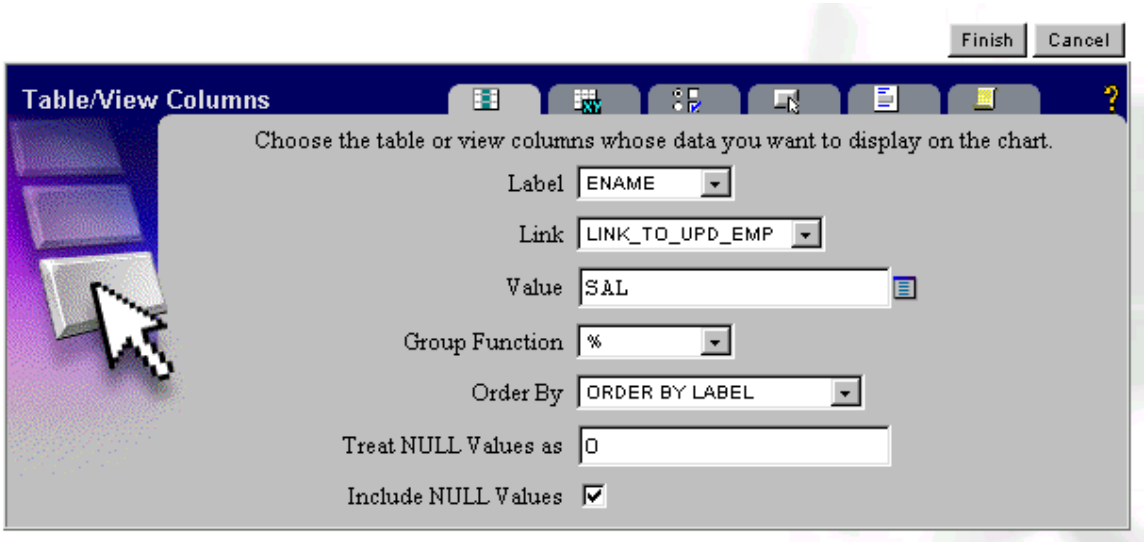
You can add these types of shared components to WebDB components:

Shared component:	Enables you to:
Links	Add hypertext jumps between WebDB components and other components, parameter entry forms, and URLs.
Lists of Values (LOV)	Add selectable dynamic values to entry fields in components and parameter entry forms. A List of Values is always associated with a table or view column. It displays values from the column in an entry field on a component or parameter entry form. Any component built using the column can use the LOV.A single List of Values can be displayed in several formats such as fields, radio buttons, or check boxes.
JavaScripts	Perform field- or form-level validation of entry fields in the component.
UI templates	Set the look and feel of the Web page on which a component appears.

Colors	Set the background color of a component, and other component elements such as report headings and chart bars.
Fonts	Set the font for text that appears in components, such as labels and headings.
Images	Add graphic image files to a component or its background.

Any WebDB Developer can create a shared component. A single shared component can be used by multiple component developers. Color, font, and image definitions, and UI templates automatically become available to other developers.

Developers who create links, Lists of Values, and JavaScripts typically build them in a schema that other WebDB Developers have Build In privileges in. These shared components then become available to WebDB Developers building components in the schema, typically through a pop-up or list. For example, the Table/View columns page of the Chart build wizard has a **Link** list that allows a WebDB developer to choose available links for the component.



WebDB includes a default set of shared components that are added to the database during installation.

Searching for shared components

To find a shared component such as a link, List of Values, JavaScript, or UI template:

1. Click  on the bottom of any WebDB page.

2. In the **Schema** list, choose the schema that owns the shared component you want to find.

   Note If you are trying to find a shared component in a schema that is not listed in the box, you currently don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to Build In this schema.

3. (Optional) In the **Name Contains** field, type one or more characters of the shared component's name.

4. (Optional) If you know the component's type (for example, link, List of Values, etc.), check the check box next to its type in the **Shared Components** check box group and uncheck all other check boxes, including those in the UI components check box group.

5. You can also choose a **Component Status** to narrow your search. For example, you can choose **ARCHIVE** to search for old versions of the shared component, and **PRODUCTION** to search for the most recent version.

6. Click **Find**. A list of shared components that match your search criteria appear at the bottom of the page.

   The  icon indicates components created within the last 7 days.

7. Click a name in the **Component Name** column to edit the shared component.

Managing color definitions

A color definition is an association between a color name and its associated value. You can specify any name for a color, and any color value (or its hexadecimal equivalent) supported by your Web browser. The color names you define are used in fonts, page backgrounds, and other elements of WebDB components.

To add a color definition:

1. Click  at the bottom of any WebDB page.
2. Click **Colors**.
3. In the **Color Name** field, type the name you want to give to the color. You can identify a color by any name you choose; for example, `My_Blue_Color`.
4. In the **Color Value** field, type a color value. You can specify any color value supported by your Web browser or its hexadecimal equivalent; for example, `Blue` or `#C0D9D9` for Light Blue. Hexadecimal values must be prefaced by the # character.
5. Click **Add Color**.
6. The new color name appears in the **Edit Color Definition: Name** list.

To edit an existing color definition:

1. Click  at the bottom of any WebDB page.
2. Click **Colors**.

3. In the **Edit Color Definition: Name** list, click the name of the color you want to edit.

4. (Optional) Edit the name displayed in the **Color Name** field. You can identify a color by any name you choose; for example, `My_Blue_Color`.

5. (Optional) Edit the value displayed in the **Color Value** field. You can specify any color value supported by your Web browser or its hexadecimal equivalent; for example, `Blue` or `#C0D9D9` for Light Blue. Hexadecimal values must be prefaced by the # character.

6. Click **Apply Changes**.

7. The new color definition appears in the **Edit Color Definition: Name** list of the Manage Colors page.

To delete an existing color definition:

1. Click  at the bottom of any WebDB page.
2. Click **Colors**.
3. Click the name of the color you want to edit in the **Edit Color Definition: Name** list.
4. Delete the current name in the **Color Name** field (that is, leave the **Color Name** field blank).
5. Click **Apply Changes**.

Managing font definitions

A font definition is simply the name of a font that is supported by your Web browser; for example, `Arial`, or `Times New Roman`. The fonts you define are used for text that appears in WebDB components.

To add a font definition:

1. Click  at the bottom of any WebDB page.
2. Click **Fonts**.
3. In the **Font Definition Name** field, type the name of the font you want to define; for example, `Arial`.

   You can specify any font name supported by a Web browser.
4. Click **Add**.

   The new font appears in the **Edit Font Definition: Font Definition** list of the Manage Fonts page.

To edit an existing font definition:

1. Click  at the bottom of any WebDB page.
2. Click **Fonts**.
3. In the **Edit Font Definition: Font Definition** list, click the name of the font definition you want to edit.
4. Change the name displayed in the **Font Name** field.

5. Click **Apply Changes**.

6. (Optional) Click your browser's Back button to view the new fontin the **Edit Font Definition: Font Definition** list of the Manage Fonts page.

To delete an existing font definition:

1. Click  at the bottom of any WebDB page.

2. Click **Fonts**.

3. In the **Edit Font Definition: Font Definition** list, click the name of the font definition you want to edit.

4. Delete the current name in the **Font Name** field (that is, leave the **Font Name** field blank).

5. Click **Apply Changes**.

Notes

- In order to be displayed by the end user's Web browser, the **Font Definition Name** must match the name of a font supported by the Web browser. If you specify a font that is not supported, the Web browser will display its default font.

- You can specify alternate fonts by separating them with commas in the **Font Definition Name** field; for example, `Times New Roman, Times`. In this example, if the user's Web browser doesn't support Times New Roman, it will display text using the Times font. If the Times font is not supported, the Web browser will display its default font.

Managing image definitions

An image definition is an association between an image name and the name of the file containing the image. You can specify any image name you choose. The image file must be located in the `/images/` logical directory on the same server where WebDB is located. If you place it in this directory, the image appears as a selection in any image lists displayed in WebDB; for example, in component build wizards and pages for creating UI templates.

To create an image definition:

1. Click  at the bottom of any WebDB page.

2. Click **Images**.

3. In the **Create a New Image Name: Image Name** field, type the name you want to use to identify the image to other users. You can specify any name you choose.

4. In the **File Name** field, type the name of the file containing the image. WebDB supports the .gif image file format.

   Note To specify a **File Name**, you must first store the image file in the /images/ logical directory.

5. Choose an image type from the **Image Type** list. Other users can specify this type when searching for image files.

6. Click **Add Image**. The new image definition appears in the **Select a Recently Edited Image** list at the bottom of the page.

To edit an image definition:

1. Click  at the bottom of any WebDB page.

2. Click **Images.**

3. If the image definition you want to edit appears in the **Select a Recently Edited Image** list, go to step 8.

4. In the **Find an Existing Image: Image Name Contains** field, type one or more letters in the name of the image you are trying to find.

5. (Optional) In the **File Name Contains** field, type one or more letters in the name of the file for the image.

   The **Image Name Contains** and **File Name Contains** fields are not case sensitive.

6. (Optional) In the **File Type** field, choose the image's type.

7. Click **Find Images**.

8. Click the image name to edit it.

   The Edit Images page appears.

9. (Optional) Edit the name displayed in the **Image Name** field. You can identify an image by any name you choose.

10. (Optional) Edit the file name displayed in the **File Name** field.

11. (Optional) Choose a new **Image Type**.

12. Click **Apply Changes** to update the image definition.

To delete an image definition:

1. Follow steps 1-8 above.

2. Delete the current name in the **Image Name** field (that is, leave the **Font Name** field blank).

3. Click **Apply Changes**.

Exporting shared components

WebDB stores shared components and information about them in database tables. These tables are automatically created during installation of WebDB. Each type of shared component has its own table. For example, a table called `WWV_USR_TABLE_LINK$` contains all links defined for the current instance of WebDB.

You can export shared components to another instance of WebDB (that is, WebDB installed under a different schema name) installed on the same database or installed on a remote database. To do so, you copy SQL INSERT statements provided by WebDB and use these to insert shared components into tables associated with another instance of WebDB. To export all links to a remote database where WebDB is installed, you'd use the SQL INSERT statement to insert the links into the table `WWV_USR_TABLE_LINK$` associated with the remote instance of WebDB.

You can export all shared components of a given type (for example, Links, Lists of Values, Colors, Images, etc.) or a single shared component.


You must have the DBA role to export all shared components of a given type. To export a single shared component, you must have Build In privileges in the schema that owns the Link, List of Values, or UI template.

To export all shared components of a given type:

1. Click  at the bottom of any WebDB page.

2. Click **Export Shared Components**.

3. Click the type of shared component you want to export. For example, click **Export Colors** to export color definitions.

   A page appears containing a SQL INSERT statement for each shared component that you can export.

4. Use your Web browser's copy and paste feature to copy the INSERT statements into an ASCII-formatted text file.

5. Using SQL*Plus, execute the file as the schema owner for the other instance of WebDB. For example, if the other instance of WebDB is installed under the schema name WebDB2, log on to the database as WebDB2 and execute the INSERT statements using SQL*PLUS.

To export a single Link, List of Values, or UI template:

1. Click  at the bottom of any WebDB page.

2. Click the type of shared component you want to export. For example, click **Links** to export a link.

3. If the shared components you want to export appear in the **Select a Recently Edited** (Link, List of Values, UI template) list, go to step 6.

4. In the (Link, List of Values, UI template) **Name Contains** field, type one or more letters in the name of the Link, List of Values, or UI template you are trying to find.

5. Click the Find button (**Find Link**, **Find LOV**, etc.).

6. Click **Export**.

   A page appears containing a SQL INSERT statement for the shared component.

7. Use your Web browser's copy and paste feature to copy the INSERT statements into an ASCII-formatted text file.

8. Using SQL*Plus, execute the file as the schema owner for the other instance of WebDB. For example, if the other instance of WebDB is installed under the schema name webdb2, log on to the database as `webdb2` and execute the INSERT statement using SQL*PLUS.

Building Links between Components

What are links?

Links are hypertext jumps between WebDB components and other components, component parameter entry forms, or any HTML page. For example, you can create a chart that displays the average salary for departments in an organization. An end user could click each department name to link to a report that displays all department employees along with their salaries.

If you create a link to a component, the component displays when an end user clicks the link. If you create one to a parameter entry form, the parameter entry form for a component displays. The end user can specify parameters in this form, then display the component using these parameters. If you create a link to an HTML page, the page displays.

Links can pass parameters to a target component. The parameters specify the initial content of the component when it displays. Before you build a link to a target component, you must ensure the target accepts parameters.

Any link you build becomes available to other WebDB Developers if you place it in a schema in which they have Build In privileges. It's a good idea to store a linkin the same Component Schema as its target component or parameter entry form. Other WebDB Developers can use the link to build WebDB components. The component build wizards contain steps that ask the developer to optionally choose a link for a column in the table or view on which the component is based.

Building links

A link creates an association between a WebDB component and another component, parameter entry form, or URL. After you create a link, other WebDB Developers can use it to create their components.

To build a link:

1. Click  at the bottom of any WebDB page.

2. Click **Links,** then **Create Link.**

3. In the **Schema** list, choose the schema that will own the finished link. Only Component Schemas in which you have Build In privileges are listed.

   Tip If you are building a link to a component or parameter entry form, choose the same schema that these are located in. Otherwise, choose a schema that WebDB Developers who will use the link have privileges to Build In.

4. In the **Link Name** field, type the name you want to give to the link.

   Tip Since this is the name that users of the link will use to identify it, you can choose a name that describes the link's function; for example, `LINK_TO_REPORT241` for a link to a report.

5. Click  .

6. In the **Link is pointing to** radio button group, choose the target for the link (a component, parameter entry form, or an HTML page).

   If the link is to a component or a component's parameter entry form, type the name of the component in the field at the bottom of the page, prefixed by the schema that owns it.

   For example, if the link is to a component called SCOTT_REPORT in the SCOTT schema, type `SCOTT.SCOTT_REPORT`. If the link is to the parameter entry form for SCOTT_REPORT, you still type `SCOTT.SCOTT_REPORT.`

Click  to search for component names.

7. If the link is to an HTML page, type its URL; for example, `http://www.mycompany.com`
8. Click  .
9. If you selected **WebDB Component Parameter Form** or **HTML Link** on the Link Target Type and Name page, you are finished with the link. Click **OK** on the Create Link page.
10. If you selected **WebDB Component** on the Link Target Type and Name page, click  .

11. If the WebDB Developer who created the target component specified columns that can accept parameters (either with a bind variable or using an option in the build wizard), they appear on the Link Target Input page. For example, if you chose to link to a component based on the SCOTT.EMP table, you might see ENAME, EMPNO, and DEPTNO on this page if they are parameters of the target component.

12. Specify values for each parameter on this page that will be passed to the target component when it displays. You can pass literal values or values in a column in the table or view on which the source component is based.

    For example, you can create a link to a report based on SCOTT.EMP that displays all employees in a particular department, then use the link you create in another component. The department number that displays corresponds to the value the end user chose when he clicked the link. For example, if the end user clicked 10, all employees in Department 10 display in the report.

    To do this, specify `=` as a **Condition,** `Column` as the **Value Type**, and `DEPTNO` as the **Value**.

    The Link Target Inputs page also displays options that control the appearance of the target component; for example, the maximum number of rows that can appear on a page of a report. The options correspond to options on the Display Options page of the component build wizard. Other options for a report can include **show_header** (enables you to show an HTML for the report when displaying it), **font_size** (enables you to specify the font size for text in the report), or **order_by_1** (enables you to specify a column to break the report on),

    For example, if the option is **show_header**, you could specify a **Value Type** of **Literal** and type `YES` or `NO` as the Value.

13. Specify options that will control the appearance of the component when it displays.

    If you have a question about an entry field on this page, click the small  button.

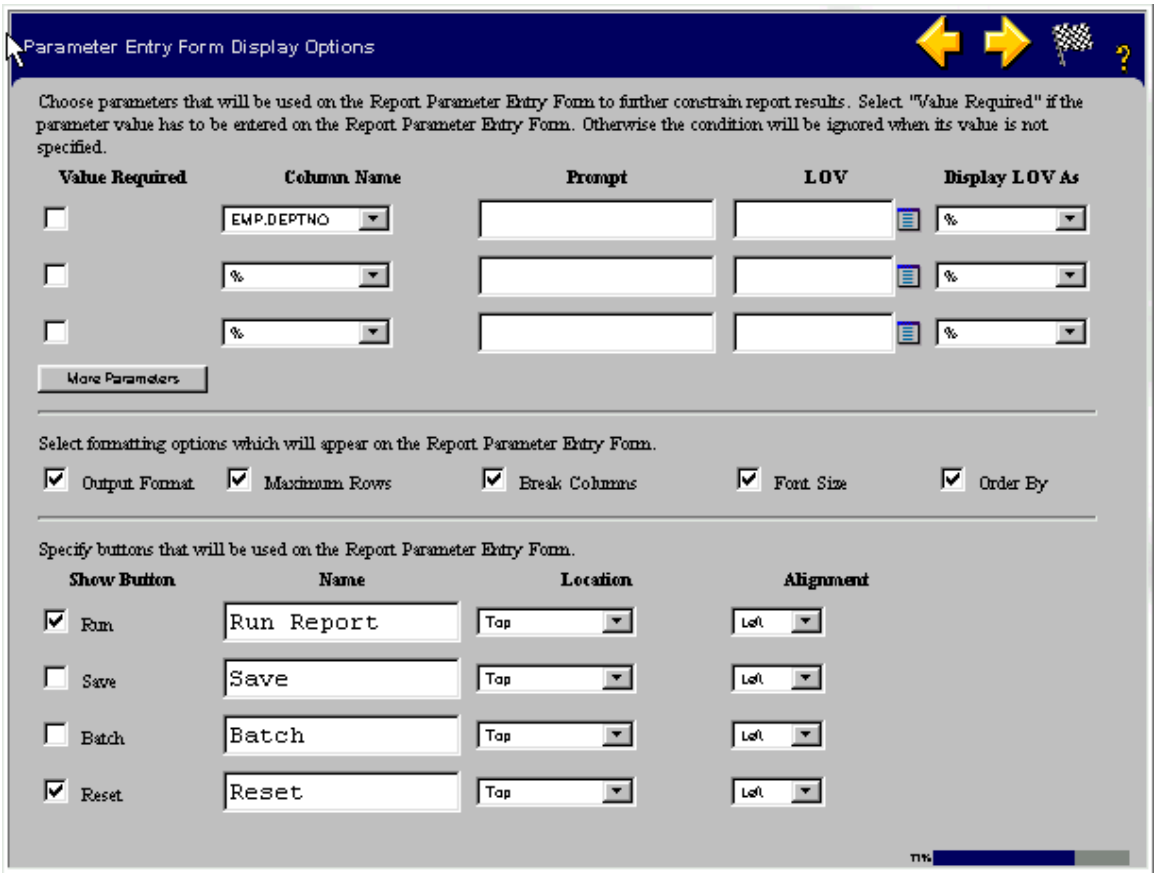
14. Click  .

    The Create Link page displays.

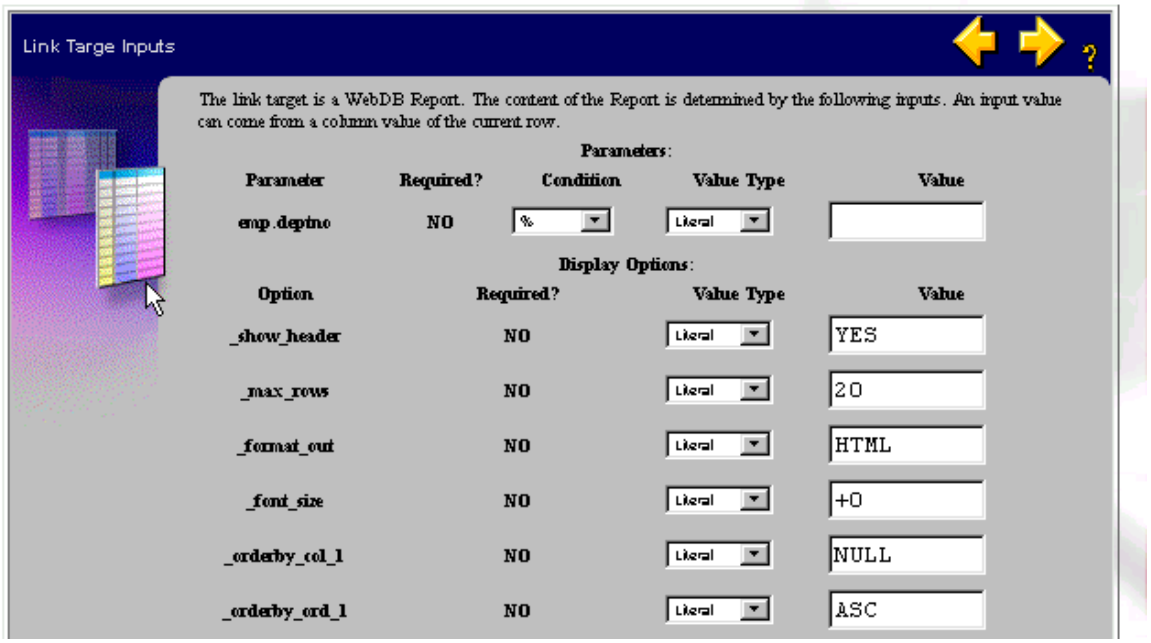
15. If you are satisfied with your choices in this wizard, click **OK**. The link will be stored in the database as a shared component in the Component Schema you specified on the Link Name page of this wizard.

Passing parameters between components

Before you build a link to a target component, you must ensure the target accepts parameters. You can create parameters using bind variables in the SQL query for the target component. Or, you can specify an option on the Parameter Entry Form Display Options page of a component build wizard. In the example below, the DEPTNO column of the SCOTT.EMP will accept a parameter because it has been selected in the **Column Name** list box.



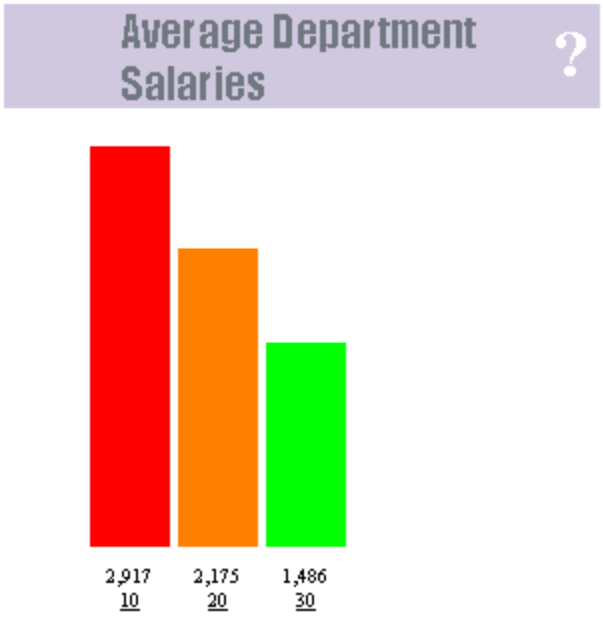
After you create the target component, you can create a link to it. The last page of the Link wizard lists all parameters specified for the target component. For example, if the target was the report built using the options shown above, the last page of the link wizard would look something like this:



The **Parameter** shown above is the same parameter that was specified on the Parameter Entry Form Display Options page of the report build wizard.

If you specify on the Link Target Inputs page = as the **Condition**, `Column` as the **Value Type**, and `DEPTNO` as the **Value,** the link will accept parameters corresponding to values in the DEPTNO column.

After you create the link, you can use it to build another component. For example, you could build a chart based on SCOTT.EMP that displays the average salary of employees in each department in the DEPTNO column. If you add the link to the DEPTNO column of the chart, clicking a department number in the chart would use the corresponding value in DEPTNO to display the report; for example:



Clicking 10 in the above chart would link to a report that displays employees in Department 10, as shown below:

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10

Row(s) 1 - 3

Testing links

You test a link to view its hypertext jumps and the link's destination. The destination will be a WebDB component, component parameter entry form, or URL.

Testing a link also allows you to view the link, an HTML anchor <A> tag that references the link, and an example SQL query that uses the anchor.

To test a link:

1. Click  at the bottom of any WebDB page.

2. Click **Links**.

3. If the link has been recently edited, go to step 6. Recently edited links appear in the **Select a Recently Edited Link: Link Name** column of the Manage Links page.

4. If the link hasn't been recently edited, type its name or a portion of its name in the **Find an Existing Link: Link Name Contains** field. To find a link named LINK_FROM_SALARY_CHART, for example, you could type `LINK_FROM_SALARY_CHART`, `salary` , or even `L`. The **Link Name Contains** field is not case sensitive.

5. Click **Find Link.** A page appears containing all links that match your search criteria. Click a link name to edit the link.

6. Click **Test**. The **Example Query Results** list displays all the hypertext jumps for the link.

7. Click each link to ensure the destination component and the data displayed in the component are correct.

   Hint You can copy the **Link** or the **Anchor** shown on this page into a SQL query.

Finding existing links

A link creates an association between a WebDB component and another component, parameter entry form, or HTML page. Once you find a link, you can edit the link or create a new link based on an existing one.

To find an existing link:

1. Click  at the bottom of any WebDB page.

2. Click **Links**.

3. If the link you are trying to find has been recently created or edited, it appears in the **Select a Recently Edited Link: Link Name** column of the Manage Links page. Click a **Link Name** to view the options that were used to create it, or to edit the link.

   If the link hasn't been recently created or edited, type its name or a portion of its name in the **Find an Existing Link: Link Name Contains** field. To find a link named LINK_TO_SALARY_CHART, for example, you could type `LINK_TO_SALARY_CHART`, `salary` , or even `L`.

   You can use the % wildcard in your search. The **Link Name Contains** field is not case sensitive.

4. Click **Find Link.**

   A page appears containing all links that match your search criteria. Click a **Link Name** to edit the link.

5. (Optional) Click **Delete** to delete the link from the database.

Editing links

A link creates an association between a WebDB component and another component, parameter entry form, or URL.

 Any change you make to an existing link impacts all WebDB components that currently use the link. Before editing a link, investigate which WebDB components currently use the link to assess the impact of the change.

To edit a link:

1. Click  at the bottom of any WebDB page.

2. Click **Links**.

3. If the link you are trying to find has been recently edited, it appears in the **Select a Recently Edited Link: Link Name** column of the Manage Links page. Click a **Link Name** to edit the link, or view the options that were used to create it.

4. If the link hasn't been recently edited, type its name or a portion of its name in the **Find an Existing Link: Link Name Contains** field. To find a link named LINK_FROM_EMPLOYEE_REPORT, for example, you could type `LINK_FROM_EMPLOYEE_REPORT`, `employee_report` , or even `rep`.

   You can use the % wildcard in your search. The **Link Name Contains** field is not case sensitive.

5. Click **Find Link.** A page appears containing all links that match your search criteria. Click a link name to edit the link.

   The Edit Links dialog box displays.

6. Click one or more tabs to edit the values in the entry fields on the corresponding tab page.

To access help about the entry fields on the page, click the small  button.

7. When you finish editing entry field values on all tabs you want to change, click **Finish**.

8. (Optional) If you decide you don't want to save your edits, click **Cancel**.

Exporting a link

To export an existing link to a remote database:

1. Click  at the bottom of any WebDB page.

2. Click **Links**.

3. If the link has been recently edited, go to step 6 . Recently edited links appear in the **Select a Recently Edited Link: Link Name** column of the Manage Links page.

4. If the link hasn't been recently edited, type its name or a portion of its name in the **Find an Existing Link: Link Name Contains** field. To find a link named LINK_FROM_SALARY_CHART, for example, you could type `LINK_FROM_SALARY_CHART`, `salary` , or even `L`. The **Link Name Contains** field is not case sensitive.

5. Click **Find Link.** A page appears containing all links that match your search criteria. Click a link name to edit the link.

6. Click **Export**. A page displays a SQL script that imports the link into a database.

7. Use options in your Web browser window to copy and paste the SQL script code to an ASCII text file.

8. Save the file and export it to a location in the remote database.

9. Run the SQL script using SQL*Plus in the remote database.

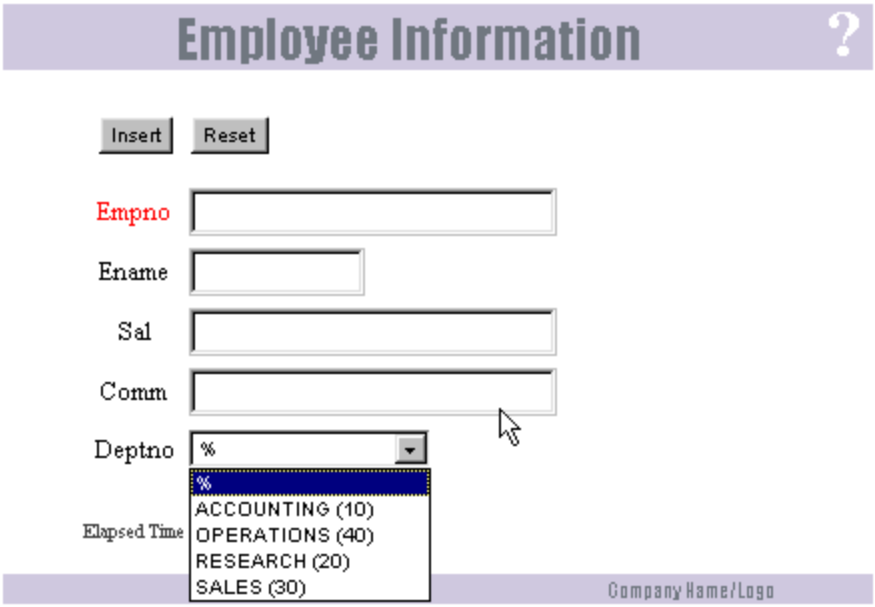
Creating Lists of Values

What are Lists of Values?

Lists of Values (LOVs) enable end users to choose entry field values in a form or component parameter entry form. WebDB Developers can use LOVs when developing components to pre-select the possible values in an entry field. It also makes the entry field easier to use, because the end user clicks the mouse to select a value rather than type it.

The component build wizards contain steps that ask a WebDB Developer to optionally choose an LOV for a column in the table or view on which the component is based. The wizards also allow the developer to choose a format for displaying the LOV; for example as a radio button group or check boxes.

For example, a WebDB Developer could create a form that allows end users to update the DEPTNO column in the SCOTT. EMP table. The developer could add a List of Values to this form that allows end users to select either Accounting, Operations, Research, or Sales, to update the DEPTNO column.



Any WebDB Developer can create an LOV and share it by placing it in a schema in which other WebDB Developers have Build In privileges. One way to do this is to place the LOV in a Component Schema. Any component built using the column or argument can use the LOV. The LOV described in the above example could be used in a department entry field in a form or a parameter entry form as long as each of these are based on SCOTT.EMP.

WebDB provides two methods for creating LOVs:

- A static LOV contains values that are hard coded when it is created.
- A dynamic LOV is dynamically generated from a SQL query of the table column. It contains all values in the column.

About using Lists of Values

Lists of Values (LOVs) enable end users to select from a list of possible values rather than type values in entry fields on forms and component parameter entry forms. Each LOV corresponds to a column in the table or view on which the component is based.

For example, you could create a parameter entry form for a chart that summarizes total monthly expenditures by department. The parameter entry form might contain two entry fields displaying Lists of Values:

- A list of departments allowing end users to choose either Sales, Marketing, Accounting, or Research.
- A list containing months of the year allowing end users to choose which monthly expenditures to display.

A WebDB Developer creates an LOV using the Create List of Values page. After the developer associates the LOV with a column in a table or view, any component based on data from that column can use the LOV. The developer typically makes the LOV available to other users by building it in a schema in which these users have Build In privileges. The users can then use it to build WebDB components.

To use a LOV, you associate it with a column name in the database table on which you are basing the component. When creating the chart in the above example, these would be the table columns containing department names and months. For each table column, you must add to your SQL query a bind variable. You don't need to know SQL to specify bind variables. The build wizard you use to create your module contains a step that asks you whether you want to use a bind variable for each column you include in your SQL query.

The build wizard asks you in another step to optionally choose an LOV for each table column that has a bind variable. A list displays all LOVs you have privileges to use. If you don't see an LOV you want to use in the list:

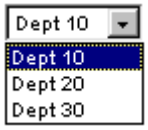
- The LOV has not been created for the table column name you specified. If so, you must create the LOV using the **List of Values Manager**.

- The LOV has been created, but you don’t have Build In privileges in the schema that owns the LOV. If so, you must request privileges to use the LOV from its creator or a user with the DBA role.

In another wizard step, you can select a format for presenting the LOV to an end user. For example, if you want your users to be able to select only one value from the list, you might choose to display radio buttons allowing only one selection. For multiple selections, you can choose check boxes or a list box allowing more than one selection.

Building a static list of values

Lists of Values (LOVs) allow end users to choose entry field values in a form or parameter entry form. A static LOV that contains any set of values you choose to add to the list. For example, you could create a list that allows end users to select numbers for a Department entry field. The list could display the values Dept 10, Dept 20, and Dept 30; or simply 10, 20, 30, whatever you choose.



You must associate the static LOV with a table or view column. Any component built using the column can use the LOV. The same values that you specify appear in the list no matter which component uses the LOV. If you want to create an LOV whose values are dynamically generated from a table of view column on which a component is based, create a dynamic List of Values .

To build a List of Values based on a static hardcoded list:

1. Click  at the bottom of any WebDB page.
2. Click **Lists of Values (LOVs)**.
3. Select the **Static, based on hard-coded values** radio button, then click **Create LOV**.
4. In the **Create this List of Values: Owning Schema** list, chose the schema that will own the finished List of Values.

   Note The list displays all schemas in which you have Build In privileges.

5. In the **Name** field, type a name for the LOV. Choose a name that describes the LOV's function; for example,`DEPARTMENTS_LOV`.

6. In the **Default Display Format** list, choose a default format for the LOV.

7. In the **Show Null Value** list, specify whether or not you want to display null values in the LOV.

8. In the **Display Value** column, type each value you want to include in the LOV. End users of the LOV will be able to select from this list.

9. For each **Display Value** you specify, type a **Return Value**. This value is passed as an argument to the component.

   In the **Display Order** column, type numbers beginning with 1 to set the order in which display values appear in the LOV. Type 1 for the first display value you want to appear at the top of the list, 2 for the second value, and so forth.

10. Click **Add LOV** to create the List of Values.

Building a dynamic List of Values

Lists of Values (LOVs) allow end users to choose entry field values in a component or parameter entry form. You can create a dynamic LOV whose values are based on the results of a SQL query that you supply. You create the SQL query specifying a Display column and a Value column using this format:

```
select display_column, value_column
from   table_name
```

- The display_column contains the values you want to display in the LOV.
- The value_column contains the values you want to pass to the component each time an end user clicks a display_column value. The display_column can be the same as the value_column.
- The table_name is the name of the table or view on which the LOV will be based. Any component built using the table can use the LOV.

If you want to create an LOV whose values are based on a hard-coded list rather than a SQL statement, create a static List of Values .

To create a dynamic List of Values:

1. Click  at the bottom of any WebDB page.
2. Click **Lists of Values (LOVs)**.
3. Select the **Dynamic, based on SQL Query** radio button, then click **Create LOV**.
4. In the **Create this List of Values: Owning Schema** list, choose the schema that will own the finished List of Values.

   Note The list displays all schemas in which you have Build In privileges.

5. In the **Name** field, type a name for the LOV. Choose a name that describes the LOV's function; for example, `DEPARTMENTS_LOV`.

6. In the **Default Display Format** list, choose a default format for the LOV.

7. In the **Show Null Value** list, specify whether or not you want to display null values in the LOV.

8. To define an LOV, write a select statement that displays a value, then a database value (this is what is passed to the stored procedure); for example:

```
select deptno, deptno from scott.emp
```

9. Click **Add LOV**.

Testing a List of Values

You test a List of Values (LOV) to ensure it contains the values you want to display to the end user. You can also see how the LOV appears when displayed in different formats, such as a list box or radio group.

To test a LOV:

1. Click  at the bottom of any WebDB page.

2. Click **Lists of Values (LOVs)**.

3. If the LOV was recently edited, it displays in the **Select a Recently Edited List of Values: LOV Name** list on this page. If so, go to step 7.

4. In the **Find an Existing List of Values: LOV Name Contains** field, type the name of the List of Values you want to find.

   The **LOV Name Contains** field is not case sensitive.

5. In the **LOV Type** list box, choose **Static** (for LOVs based on a hardcoded list) or **Dynamic** (for LOVs based on a SQL query). Choose **All Types** to search for both types.

6. Click **Find LOV**.

   The Edit Existing List of Values page displays all LOVs that match your search criteria.

7. Click a display format in the **Test As** list to display the LOV. For example, click **Radio** to display the LOV as a radio group, or **Combo** to display it as a combo box. Click **Export** to display a SQL script that allows you to export the LOV to a remote database.

Editing Lists of Values

Lists of Values (LOVs) allow end users to choose entry field values in a component or its parameter entry form. You can edit a List of Values to change the table or view column on which it is based, or the values contained in the list.

Any change you make to an existing LOV impacts all WebDB components that currently use the LOV. Before editing an LOV, investigate which WebDB components currently use the LOV to assess the impact of the change.

To edit a LOV:

1. Click  at the bottom of any WebDB page.

2. Click **Lists of Values (LOVs)**.

3. If the LOV you are trying to find has been recently edited, it appears in the **Select a Recently Edited LOV: LOV Name** column of the Manage Links page. Click an **LOV Name** to edit the LOV, or view the options that were used to create it.

4. If the LOV was not recently edited, type its name in the **Find an Existing List of Values: LOV Name Contains** field.

   The **LOV Name Contains** field is not case sensitive.

5. In the **Type** list, choose **Static** (LOV based on a hardcoded list) or **Dynamic** (LOV based on a SQL query). Choose **All Types** to search for both types of LOVs.

6. Click **Find LOV**.

   The Edit Existing List of Values page displays all LOVs that match your search criteria.

7. In the **LOV Name** column, click the name of the List of Values you want to edit.

The Edit List of Values page appears.

8. Change any options you want to change for the LOV and click **Apply Changes**.

   To access help about the entry fields on this page, click the small  button.

Creating User Interface Templates

About User Interface templates

User Interface (UI) templates control the look and feel of the Web page on which a WebDB component appears. By selecting a UI template when building a component, a WebDB Developer can automatically specify a page title, a title background, links to other Web pages, and background colors and images.

UI templates are good for standardizing the overall look and feel of groups of components. For example, you can design a UI template for your company's Web site that includes the company logo in the heading, the name of the company in the title, and a common background image. By ensuring every component uses the same UI template, you impose a standard appearance.

You can create these types of UI templates:

- Structured UI templates display the same image and text in the same location on every component that uses the template. For example, if a structured UI template contains a company logo and introductory text, the logo and text display in the same location in a chart, a report, or any other component that uses it. WebDB provides options that allow you to add the image and text without having to code any HTML yourself.

- Unstructured UI templates are based on HTML code. To create an unstructured UI template, you first write HTML code to create a Web page. You can also copy this code into WebDB from another source such as another Web page editor. You then edit the code to add substitution tags to the HTML. When the HTML code executes, the substitution tags embed components, titles, headings, and other elements into the Web page. For example, you can add a #BODY# tag that adds a component such as a chart or report to the original Web page background.

  Because you are writing your own HTML code, you can create a more elaborate and sophisticated unstructured template than you could if you created an unstructured template.

Building a dynamic List of Values

You typically build a UI template in a schema that is available for other WebDB Developers to Build In. The component build wizards contain a step that asks the developer to optionally specify a UI template to set the look and feel of the component.

Creating a structured UI template

To create a structured UI template:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Templates.**

   The Manage UI Templates page appears. Review the **Edit UI Template: Name** column on this page. You may find an existing UI template in the column that matches the one you plan to build, or one you can edit and save as a new UI template.

3. In the **Owning Schema** list, click the schema that you want to own the finished UI template. All WebDB Developers having Build In privileges in this schema will be able to use the UI template you create.

   If you don't see the schema where you want to store the UI template in the list, you don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to build in this schema.

4. Select the **Structured (specific images and text)** template type and click **Create Template**. The Create Structured UI Template page appears.

5. In the **Template Owner and Name: Owner** list, choose the schema you want to own the finished template.

   Note Only schemas you have privileges to build in are listed.

6. Type the name you want to give the template in the **Template Owner and Name: Name** field. Because this is the name a WebDB Developer will see when applying a UI template to a component during the build process, you should make the name as descriptive as possible. For example, if the template will be applied to all components created by the Accounting Group, you could name it `Accounting_Template`.

7. (Optional) Select the other options on this page you want to include in the UI template.

   If you have a question about an option, click the small  button. Leave an option blank if you do not want to include it.

8. (Optional) If you need to reset the options on this page to their default values, click **Reset**.

9. (Optional) Click **Test** to view the UI template based on the options you have selected.

   The lower frame of the page displays the UI template based on the options you selected in the upper frame. You can select or remove additional options, then press **Test** again to see how they look in the UI template.

10. When you finish selecting options, click **Save** in the upper page frame to save the UI template.

Creating an unstructured UI template

To create an unstructured UI template:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Templates**.

   The Manage UI Templates page appears. Review the **Edit UI Template: Name** column on this page. You may find an existing UI template in the column that matches the one you plan to build, or which you can edit and save as a new UI template.

3. In the **Owning Schema** list, click the schema that you want to own the finished UI template. All Web Developers having Build In privileges in this schema will be able to use the UI template you create.

   If you don't see a schema in the list where you want to store the UI template, you don't have Build In privileges in the schema. You must ask a user with the DBA role for privileges to build in this schema.

4. Select the **Unstructured (HTML code with substitution tags)** template type and click **Create Template**.

   The Define Unstructured Template page appears.

5. In the **Owner** list, choose the schema you want to own the finished template.

   Note Only schemas you have privileges to build in are listed.

6. In the **Name** field, type the name you want to give the template. Because this is the name a WebDB Developer will see when applying a UI template to a component during the build process, you should make the name as descriptive as possible. For example, if the template will be applied to all components created by the Accounting Group, you could name it `Accounting_Template`.

7. In the **Enter HTML code that creates a web page** field, type or paste the HTML code you want to use as the basis for your unstructured UI template. The HTML code you add to the field should create a Web page. You can paste HTML source code from another Web page or an editor.

8. Embed substitution tags in the HTML code in the location where you want the items associated with the tags to appear in the finished template.

   For example, you could embed a #BODY# substitution tag to the code. #BODY# adds a WebDB component such as a chart to the Web page when the HTML code executes. If the HTML source code divides a page into two frames, you can embed the #BODY# tag in different places in the code, causing the component to display in the left frame or right frame.

   Note A list of available substitution tags is shown at the bottom of the top frame.

9. (Optional) Select the other options on the Define Unstructured Template page that you want to include in the UI template. Leave an option blank if you do not want to include it.

   If you have a question about an option, click the small  button. Leave an option blank if you do not want to include it.

10. (Optional) If you need to reset the options on this page to their default values, click **Reset**.

11. (Optional) Click **Test** to view the UI template based on the options you have selected.

    The lower frame of the page displays the UI template based on your currently selected options. You can select or remove additional options, then press **Test** again to see how they look in the UI template.

12. When you finish selecting options, click **Save** in the upper page frame to save the UI template.

Creating JavaScripts

Guidelines for writing JavaScripts

WebDB allows you to create JavaScripts that perform validation on entry fields in forms. Field-level validation is performed when the end user causes the OnBlur condition to occur after entering a value in an entry field, for example, when tabbing to another entry field. Form level validation occurs after the user enters a value in an entry field and submits all values on the page, for example, when clicking an **OK** button.

Follow these guidelines when writing a field- or form-validation JavaScript application:

- All validation routines should be written as functions, and return either TRUE or FALSE values.
- The application should alert the end user with a message if the element (entry field) being validated contains an invalid value.
- The application should bring focus (position the cursor) to the element (the entry field) where the user entered the incorrect value flagged by the JavaScript application.

The following JavaScript validates that the user enters a numeric value into an entry field:

```
function isNumber(theElement)                    ← 1
{
if ( isNaN( Math.abs(theElement.value) ) )       ← 2
{
alert( "Value must be a number." );              ← 3

theElement.focus();                              ← 4

return false;
}
return true;
}
```

1) Identifies the name of the function and the entry field being validated.
2) Checks whether the absolute value of the entry field is a number. IsNaN ("Is Not a Number") is a JavaScript function.
3) If the value in the entry field is not a number, the user is alerted with the message, "Value must be a number."
4) The application brings focus to the entry field.

Other JavaScript functions that validate entry fields in components and parameter entry forms include:

JavaScript function	**Description**
isDate	Validates that the entered date is in the format: DD-MON-YY.
isInteger	Validates that the entered value is an integer value.
isAlphaNumeric	Validates that the entered value is alphanumeric.
isAlpha	Validates that the entered value is an alphabetical letter.
isEmailAddress	Validates that the entered e-mail address has the @ symbol included. This function does not validate if the e-mail address is true.

Creating JavaScripts

WebDB allows you to create JavaScripts that perform validation on entry fields in forms and component parameter entry forms.

To create a JavaScript:

1. Click  at the bottom of any WebDB page.

2. Click **JavaScripts**, then **Create JavaScript**.

3. In the **Owning Schema** list, choose the name of the schema that will own the finished JavaScript. Only schemas in which you have Build In privileges are listed.

   If you want other WebDB Developers to be able to use the JavaScript when creating components, choose a schema that these developers have Build In privileges.

4. In the **JavaScript Name** field, type a descriptive name for the JavaScript; for example, `NotNull` for a JavaScript that checks for null values in an entry field.

5. In the **Language** entry field, type the language in which the JavaScript will be written; for example, `JavaScript1.1` or `JavaScript1.2.`

6. Type or copy your JavaScript into the **Enter JavaScript** field.

7. Click **Add JavaScript**.

Editing JavaScripts

 Any change you make to an existing JavaScript impacts all WebDB components that currently use the JavaScript. Before editing a JavaScript, investigate which WebDB components currently use it to assess the impact of the change.

To edit a JavaScript:

1. Click  at the bottom of any WebDB page.

2. Click **JavaScripts**. The Manage JavaScripts page appears.

3. Check the JavaScripts displayed in the **Select a Recently Edited JavaScript: Name** column to see if the JavaScript is listed. If so, go to step 7.

4. If you know the schema that owns the JavaScript, choose the schema in the **Owning Schema** field. Otherwise, specify %.

5. Type the name of the JavaScript you want to find in the **JavaScript Name Like** field. You can also type a few letters in the name to perform a wildcard search.

6. Click **Find JavaScripts**. The Edit Existing JavaScripts page displays all JavaScripts that match your search criteria.

7. Click the name of the JavaScript you want to edit. The Edit JavaScript page appears.

8. Edit the JavaScript or change any other options and click **Apply Changes**.

Testing JavaScripts

To test a JavaScript:

1. Click  at the bottom of any WebDB page.

2. Click **JavaScripts**. The Manage JavaScripts page appears.

3. Check the JavaScripts displayed in the **Select a Recently Edited JavaScript: Name** column to see if the JavaScript is listed. If so, go to step 7.

4. Type the name of the JavaScript you want to find in the **JavaScript Name Like** field. You can also type a few letters in the name to perform a wildcard search.

5. If you know the schema that owns the JavaScript, choose the schema in the **Owning Schema** field. Otherwise, specify %.

6. Click **Find JavaScripts**.

   The Edit Existing JavaScripts page displays all JavaScripts that match your search criteria.

7. Click **Field** to test the JavaScript as a field-level validation application. Click **Form** to test it as a form-level validation application.

8. Type the value you are validating into the first entry field on the page.

   If you are testing a field-level JavaScript application, you must cause the OnBlur condition to occur; for example, by pressing the Enter or Return key or tabbing to the second entry field. After you cause the condition to occur, the JavaScript field-level validation application should run (for example, an error message could appear if you included this in your JavaScript).

9. If you are testing a form-level application, it should run after you type an invalid value in the entry field and click **Submit**.

Performing WebDB Administration

About user information

Users with the DBA role can display in the User Manager the following information about other WebDB users:

- Whether the user is a WebDB Developer or a Component Building Schema.
- Their database profile and default and temporary tablespaces.
- Their Oracle database roles.
- All database object privileges (SELECT, INSERT, UPDATE, DELETE, or EXECUTE) currently granted to the user. This includes EXECUTE privileges on packaged procedures containing WebDB components.
- All schemas in which the user has privileges to build or browse.

Updating user information

The User tab of the User Manager contains information about existing WebDB users, including:

- Whether the user is a WebDB Developer or a Component Building Schema.
- Roles that the user is a member of.
- Object privileges granted to the user.
- Build In and Browse In privileges granted to the user.
- The user's database tablespaces and profile.

To view or update this information:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager**.

3. In the **Find an Existing User: User Name** field, type the name of the user about whom you want to view information, then click **Find.**
   If you aren't sure of the user name, click  to the right of the field to view a pop-up list of user names. You can type search criteria in the pop-up list to refine your search.

   The User Manager page displays information about the user. You can update any of the information except the user's name. To do this, type new values in the entry fields on this page and click **Apply.** Or, navigate to other pages by clicking the page tabs or hypertext links.

   Click the small  help icon for information about fields on this page or any other page in the User Manager.

Creating a new user

 You must have the DBA role to create a new user.

To create a new user:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager**.

3. In the **Create a New User: User Name** field, type a name for the new user.

   In WebDB, user names are the same as database schema names. For example, if you create a user named SCOTT, the user will be able to create WebDB components and database objects in the SCOTT schema.

4. In the **Password** field, type a password for the user.

5. In the **Confirm Password** field, type the password again.

6. Choose a **Default Tablespace.** The default tablespace will be used to store any database objects created by the user.

7. Choose a **Temporary Tablespace** for the user**.** The temporary tablespace will be used for creation of temporary storage for operations such as sorting table rows.

8. Choose an **Oracle Profile** for the user. A profile is used to limit the amount of system and database resources that are available to the user.

9. Check the **WebDB Developer** check box to enable the user to create objects and components in their own schema or any schema in which you grant them Build In privileges. The user will be automatically assigned the WEBDB_DEVELOPER role.

   Check the **Component Building Schema** check box to treat this user as a schema in which other WebDB Developers can build objects and components. **Note** If you check this check box, this schema appears on the Build Privileges tab page of the User Manager. You must navigate to this tab page to grant WebDB Developers Build In privileges in order for other WebDB Developers to build objects and components in the schema.

10. Click **Create**.

    The User Manager tabbed dialog box appears if you successfully created the WebDB Developer or Component Building schema. You can use the pages to assign roles, object privileges, and Build In and Browse In privileges to the new user.

    The new user name also appears in the **Select Recently Created Users** list on the User Manager page if you were successful.

Note

- Refer to your Oracle documentation for more information about tablespaces and profiles.

Changing other users' passwords

To change a user's password other than your own:

1. Click  on the bottom of any WebDB page.
2. Click **User Manager**.
3. In the **Find an Existing User: User Name** field, type the name of the user, then click **Find**. To search for users, click  .
4. Type the user's new password in the**New Password** field.
5. Retype the password in the **Confirm Password** field.
6. Click **Apply.**

   A page appears indicating whether you successfully changed the password.

Managing Roles

Assigning the DBA or WEBDB_DEVELOPER role

 You must have the DBA role to assign a role to a user.

Two roles have special meaning in WebDB:

- WEDB_DEVELOPER
- DBA

These roles define the actions a user can perform by controlling access to WebDB menus. A user must have one of these roles to perform WebDB tasks. The roles also specify the user's default privileges for browsing and building in schemas.

There are three ways to assign the WEDB_DEVELOPER role:

- The User Manager contains a **WebDB Developer** check box that you can check when creating a new user. Checking this box enables the user to create objects and components in their own schema or any schema in which they have privileges, and automatically assigns the WEBDB_DEVELOPER role to the user.

- You can also check the **WebDB Developer** box when updating an existing user's information.

- You can assign the WEDB_DEVELOPER or DBA role to the user in the Role Manager as shown below. If you use this method, you must also explicitly grant Build In privileges that allow the user to build components in his or her own schema.

To assign a WebDB role to a user:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager.**

3. In the **Find an Existing User: User Name** field, type the name of the user to whom you want to assign a role. To search for users, click  .

4. Click **Find**.

5. Click the User Manager **Roles** tab.

   The **Is Member Of** list box displays all roles to which the user currently belongs.

6. Type either DBA or WEBDB_DEVELOPER in the **Role** field, then click **Add.**

   A new role appears in the **Is Member** Of list box each time you click **Add**.

7. Click **Apply**.

8. (Optional) To remove a user from a role, select one or more roles in the **Is Member Of** list box, then click **Remove**.

Creating new Oracle roles

You must have the DBA role to create a new role.

You typically create a role in WebDB only to grant EXECUTE privileges that allow groups of users to run WebDB components. This is because you must explicitly grant SELECT, INSERT, UPDATE, DELETE privileges to objects that will be used to build a WebDB component. These privileges cannot be granted through a role, but must be explicitly granted using the Grant Manager.

After you create a role, you assign users to it to grant the database object privileges associated with the role to the users.

To create a new role:

1. Click  at the bottom of any WebDB page.
2. Click **Role Manager**.
3. In the **Create a New Role: Roles** field, type the name of the new role; for example, `Sales_Department`. Blank characters and special characters such as % or & are not allowed. Type an underscore character (_) to add a space in a name.
4. Click **Create**.
5. In the **User/Role** field, type the name of the user or role you want to assign to the role, then click **Add**. The user or role appears in the **Is Member Of** list box each time you click **Add**.

   Note Roles can be members of other roles.
6. Click **Apply**.
7. (Optional) To remove a member from a role, select one or more roles in the **Is Member Of** list box, then click **Remove**.

Managing Build and Browse privileges

Granting privileges to Build In schemas

 You must have the DBA role to grant users Build In privileges.

Build In privileges allow an existing WebDB Developer to create components in a Component Schema.

To grant Build In privileges to a WebDB Developer:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager**.

3. In the **Find an Existing User: User Name** field, type the name of the user to whom you want to grant or modify Build In privileges. To search for users, click  .

4. Click the User Manager **Build Privileges** tab.

   The **Current Privileges** list box on the Build Privileges tab page displays all Component Schemas plus the schema for the WebDB Developer. The WebDB Developer's schema displays in the upper left corner of the page after the text **Build Privileges**.

   Selected (highlighted) schemas in the list box are those in which the WebDB Developer currently has Build In privileges.

5. To grant privileges to Build In a schema to the WebDB Developer, select (highlight) it in the Current Privileges list box, then click **Apply**.

6. To revoke a Build In privilege, deselect it and click **Apply**.

Notes

- Granting Build In privileges in a Component Schema automatically grants the schema's object privileges to the WebDB Developer.

- Granting Build In privileges in a schema automatically grants to the WebDB Developer the privilege to Browse In the same schema.

Granting privileges to Browse In schemas

 You must have the DBA role to grant users Browse In privileges.

Browse In privileges specify which database schemas the user can browse for objects such as tables, views, and procedures on which the component will be based.

To grant Browse In privileges:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager.**

3. In the **Find an Existing User: User Name** field, type the name of the user to whom you want to grant or modify Browse In privileges. To search for users, click  .

4. Click the User Manager **Browse Privileges** tab.

   The **Current Privileges: Schema** column on the Browse Privileges tab page displays all schemas in which the user currently has privileges to browse.

5. In the **Schema** field, type the name of the schema in which you want to allow the user to browse.

6. Click **Add**. The schema appears in the **Current Privileges: Schema** column.

7. Check the **Browse In** check box to grant privileges. Uncheck the check box to remove existing privileges.

8. Click **Apply**.

Notes

- Granting a user the privilege to build in a schema automatically grants the privilege to browse in the same schema.

- Browse In privileges only allow you to browse an object. To perform other actions on the object, such as use it to build a WebDB component, the schema where you are building the component must be granted explicit privileges on the object.

Viewing a report of build and browse privileges

Build In privileges allow an existing user to create components in a Component Schema. Browse In privileges specify which database schemas the user can browse for objects such as tables, views, and procedures on which the component will be based

To view a report of all users who have privileges to build or browse in database schemas:

1. Click  at the bottom of any WebDB page.

2. Click **Report WebDB Privileges.**

   A report displays all WebDB users who currently have build or browse privileges and the schemas in which the user has the privileges.

3. (Optional) Click a user name in the **WebDB User** list to update that user's build and browse privileges.

Notes

- Click **Browse In** to display all schemas in which users currently have browse privileges and the users who have them. Click **Build In** to display all schemas in which users currently have build privileges and the users who have them.
- Users who have privileges to build in a schema automatically have privileges to browse in the same schema.
- A user name is the same as the schema name in WebDB. For example, the report could display a user named SCOTT in the **WebDB User** list who has privileges to build and browse in his own schema, SCOTT.

Managing database object privileges

Granting database object privileges

 You must have the DBA role to grant or revoke privileges on database objects.

WebDB allows you to explicitly grant privileges such as SELECT, INSERT, UPDATE, DELETE, or EXECUTE on database objects. You follow a different set of steps depending on whether you want to grant privileges on a single database object to multiple users or roles, or grant privileges on multiple database object to a single user or role.

To grant privileges on a single database object to multiple users or roles:

1. Click  at the bottom of any WebDB page.

2. Click **Grant Manager**.

3. Specify search criteria in the **Find Database Objects: Schema**, **Object Type**, and **Name** entry fields to browse the database for the object on which you want to grant privileges.

   You can search for database objects by specifying search criteria in one, two, or three of these entry fields. For example, to search for tables owned by the SCOTT schema, type `SCOTT` in the **Schema** field, choose `Tables` in the **Object Type** list, and leave the **Name** field blank.

   You can use % wildcards in your search; for example, type `SCO%` to find SCOTT or SCOUT.

4. Click **Find**.

   A page appears displaying all objects matching your search criteria.

5. Click the object on which you want to assign privileges.

   The **User/Roles** list on the Grant Manager page displays all users and roles that have been granted SELECT, INSERT, UPDATE, DELETE, or EXECUTE privileges on the object you selected. The check boxes next to each user or role indicate which privileges have been granted.

6. If the user or role on which you want to grant privileges is not in the list, type a user or role name in the **User/Role** field, then click **Add**.

7. Check the check box under the privilege you want to grant to each user or role, then click **Apply**. To remove a privilege from a user or role, uncheck the check box, then click **Apply**.

To grant to a single user privileges on multiple database objects:

1. Click  at the bottom of any WebDB page.

2. Click **User Manager.**

3. In the **Find an Existing User: User Name** field, type the name of the user to whom you want to grant object privileges. To search for users, click  .

4. Click the Grants tab on the User Manager.

   The Grants tab page contains a list of all privileges granted to the user and the objects on which they are granted. The order in which object names display on this page is by object type (Sequences, Tables/Views, Procedures/Functions/Packages).

5. If the object on which you want to grant privileges is not in the list, type an object name in the **Object** field, then click **Add**. A new object appears in the list each time you click **Add**. To search for objects, click  .

6. Check the check box under each object privilege you want to grant to the user, then click **Apply**. To remove a privilege from a user, uncheck the check box, then click **Apply**.

Notes

- The Grant Manager page and User Manager **Grant** tab display check boxes based on the type of object on which you are granting privileges. For tables or views, the **Select**, **Insert**, **Update**, and **Delete** check boxes display. For procedures/function/packages, the **Execute** check box displays. For Sequences, the **Select** check box displays.

- You can add or remove several privileges at a time.

- If an object is being used to build a WebDB component, any required object privileges such as SELECT, INSERT, UPDATE, DELETE cannot be granted through a role. These object privileges must be explicitly granted to the Component Schema by following the steps on this help page.

Providing Security for Oracle Reports

What is Oracle Reports security?

Oracle Reports security ensures that a user can run an Oracle Reports report **only** if he or she has the authority to do so. Oracle Reports uses WebDB to provide this security. Access to the Oracle Reports report definition file (.RDF, .REP, .XML file), the Reports Servers on which the report can be processed, and the printers to which the report results can be output must be provided in WebDB. This certifies them for processing report requests in the secure environment. Adding report definition file, Reports Server, and printer access determines:

- Which reports, Reports Servers, and printers are available for processing in the secure environment
- Who has access to those reports, Reports Servers, and printers
- When those reports, Reports Servers, and printers are available
- Which parameter values a user is offered to control the output of reports

To run an Oracle Reports report in the secure environment, a user must log on using his or her WebDB user name and password. If the user is not authorized to run the specified report, the report does not run.

For more information, see the *Publishing Reports* manual.

What are Oracle Reports roles?

WebDB includes several roles that provide various levels of Oracle Reports administrative control. The following table describes these Oracle Reports roles in more detail:

Role	Description	Privileges
RW_ADMINISTRATOR	Sets up WebDB to enable Oracle Reports Security.	Access to all Oracle Reports Security features in WebDB. View detailed error messages. Access to all the CGI commands that are available to RW_DEVELOPER, RW_POWER_USER, and RW_BASIC_USER, plus: • killjob **Note** Only users with the RW_ADMINISTRATOR role have access to the WebDB **Oracle Reports Security** menu.
RW_DEVELOPER	Builds applications that contain Oracle Reports reports.	View detailed error messages. Access to all the CGI commands that are available to RW_POWER_USER and RW_BASIC_USER, plus:

		• showmap • showenv • showjobs • getjobid • parsequery
RW_POWER_USER	Customizes the URLs of Oracle Reports reports before running them.	View detailed error messages Access to all the CGI commands that are available to RW_BASIC_USER.
RW_BASIC_SER	Runs Oracle Reports reports.	View generic error messages. Access to the following CGI commands: • showmyjobs • killmyjobs • help

For information about how to add users to roles, see Assigning the DBA or WEBDB_DEVELOPER role.

Setting the authentication cookie domain

When a user successfully logs on to WebDB, a cookie containing authentication information is saved to his or her browser. When that user attempts to run an Oracle Reports report URL, the information in the cookie is used to provide the authentication information of the user to the Reports Server.

For the cookie to send the authentication information to the Reports Server where the report is sent, the cookie must know the domain in which to locate the Reports Server. This is done by setting the authentication cookie domain.

For example, if a Reports Server has a domain name of `my_webserver.west.my_company.com` and you set the authentication cookie domain to:

- my_company.com, the cookie can communicate with the Reports Server because it is located in a subordinate domain of the authentication cookie domain
- west.my_company.com, the cookie can communicate with the Reports Server because it is located in the authentication cookie domain
- sales.west.my_company.com, the cookie cannot communicate with the Reports Server because it is outside the authentication cookie domain
- east.my_company.com, the cookie cannot communicate with the Reports Server because it is outside the authentication cookie domain

Note

- The WebDB Listener must also be installed within the boundaries of the authentication cookie domain.

To set the authentication cookie domain:

1. Open the `wdbsvr.cfg` file in a text editor.

   This file is usually stored in your `[ORACLE_HOME]\listener\cfg` directory. For example, on Windows NT this might be `c:\orant\listener\cfg\wdbsvr.cfg`.
2. In the **[SERVER]** section of the `wdbsvr.cfg` file, set the **ORCookieName** configuration parameter to the name you want to use to identify the authentication cookie.
3. Set the **ORCookieDomain** configuration parameter to the authentication cookie domain. The default for the authentication cookie domain is the domain of the machine on which you are currently working.
4. Save your changes and close the configuration file.

What are availability calendars?

An availability calendar defines when an Oracle Reports report, Reports Server, or printer is available for use. There are two types of availability calendars:

- Simple availability calendars define a single availability rule (for example, a report might be available for processing every day from 12:00 a.m. to 10:00 p.m.).
- Combined availability calendars combine two or more availability calendars (for example, a report might be available on Monday **and** Friday from 12:00 a.m. to 10:00 p.m.).

Note You can only specify one availability calendar for each report, Reports Server, and printer. If the availability of a report, Reports Server, or printer cannot be defined by a single availability rule, you will need to create simple availability calendars for each applicable rule, and then combine those availability calendars to form a single combined availability calendar.

Examples

The following examples illustrate how you might use availability calendars to define the availability of your reports, Reports Servers, and printers.

Simple example: A report is available every day from 12:00 a.m. to 10:00 p.m.

Repeat Until example: A report is available every day from 12:00 a.m. to 10:00 p.m., until the end of the year.

Daily frequency example: A report is available from 12:00 a.m. to 10:00 p.m. every other day.

Simple combining example: A report is available every Monday **and** Friday from 12:00 a.m. to 10:00 p.m.

Repeat Monthly by Date example: A report is available every day from 12:00 a.m. to 10:00 p.m. However on the first day of every month the report is unavailable from 4:00 p.m. to 10:00 p.m. while records are updated.

Repeat Yearly example: A report is available every day from 12:00 a.m. to 10:00 p.m. However on Christmas Day the report is not available at all.

Monthly frequency example: A report is available every day from 12:00 a.m. to 10:00 p.m. However on Thanksgiving Day the report is not available at all.

Combining example: A report is available every day from 12:00 a.m. to 10:00 p.m. However on Christmas Day and Thanksgiving Day the report is not available at all.

Availability calendars: simple example

A report is available every day from 12:00 a.m. to 10:00 p.m.

You would create a simple availability calendar as follows:

Calendar Name		EVERY_DAY
Date/Time Availability	**Start**	May 17 1999 12:00 AM
	End	May 17 1999 10:00 PM
	Repeat	Daily
	Frequency	1
	Repeat Until	[Unchecked]

Availability calendars: Repeat Until example

A report is available every day from 12:00 a.m. to 10:00 p.m., until the end of the year.

You would create a simple availability calendar as follows:

Calendar Name		EVERY_DAY_1999
Date/Time Availability	**Start**	May 17 1999 12:00 AM
	End	May 17 1999 10:00 PM
	Repeat	Daily
	Frequency	1
	Repeat Until	Dec 31 1999 10:00 PM

Availability calendars: daily frequency example

A report is available from 12:00 a.m. to 10:00 p.m. every other day.

You would create a simple availability calendar as follows:

Note Setting **Frequency** to 2 means that the availability repeats every 2 days (that is, every other day) rather than every day.

Calendar Name		ALT_DAYS
Date/Time Availability	**Start**	May 17 1999 12:00 AM
	End	May 17 1999 10:00 PM
	Repeat	Daily
	Frequency	2
	Repeat Until	[Unchecked]

Availability calendars: simple combining example

A report is available every Monday **and** Friday from 12:00 a.m. to 10:00 p.m.

You would create:

- The following two simple availability calendars:

Calendar Name		MONDAY
Date/Time Availability	**Start**	May 17 1999 12:00 AM
	End	May 17 1999 10:00 PM
	Repeat	Weekly
	Frequency	1
	Repeat Until	[Unchecked]

Calendar Name		FRIDAY
Date/Time Availability	**Start**	May 21 1999 12:00 AM
	End	May 21 1999 10:00 PM
	Repeat	Weekly
	Frequency	1
	Repeat Until	[Unchecked]

- A combined availability calendar as follows:

Calendar Name	MONDAY_FRIDAY
Selected Availability Calendars	MONDAY
	FRIDAY
Excluded Availability Calendars	[None]

Availability calendars: Repeat Monthly by Date example

A report is available every day from 12:00 a.m. to 10:00 p.m. However on the first of every month the report is unavailable from 4:00 p.m. to 10:00 p.m. while records are updated.

You would create:

- The EVERY_DAY simple availability calendar from Availability calendars: simple example
- A second simple availability calendar as follows:

  Note In this case, you are specifying a period of time when the report is **not available**.

Calendar Name		UPDATE
Date/Time Availability	**Start**	Jun 1 1999 04:00 PM
	End	Jun 1 1999 10:00 PM
	Repeat	Monthly **by** Date
	Frequency	1
	Repeat Until	[Unchecked]

- A combined availability calendar as follows:

  Note You combine availability calendars together to identify which calendars determine when the report is available and not available. Then you identify which of those calendars specify when the report is **not available** by excluding them.

Calendar Name	REP_AVAILABLE
Selected Availability Calendars	EVERY_DAY
	UPDATE
Excluded Availability Calendars	UPDATE

Availability calendars: Repeat Yearly example

A report is available every day from 12:00 a.m. to 10:00 p.m. However on Christmas Day the report is not available at all.

You would create:

- The EVERY_DAY simple availability calendar from Availability calendars: simple example
- A second simple availability calendar as follows:

  Note In this case, you are specifying a period of time when the report is **not available**.

Calendar Name		CHRISTMAS
Date/Time Availability	**Start**	Dec 25 1999 12:00 AM
	End	Dec 26 1999 12:00 AM
	Repeat	Yearly
	Frequency	1
	Repeat Until	[Unchecked]

- A combined availability calendar as follows:

  Note You combine availability calendars together to identify which calendars determine when the report is available and not available. Then you identify which of those calendars specify when the report is **not available** by excluding them.

Calendar Name	WORKDAYS_EX_XMAS
Selected Availability Calendars	EVERY_DAY
	CHRISTMAS
Excluded Availability Calendars	CHRISTMAS

Availability calendars: monthly frequency example

A report is available every day from 12:00 a.m. to 10:00 p.m. However on Thanksgiving Day the report is not available at all.

You would create:

- The EVERY_DAY simple availability calendar from Availability calendars: simple example
- A second simple availability calendar as follows:

  Note You set **Repeat** to **Monthly by Day** because Thanksgiving is always on the fourth Thursday of November. You want the availability to repeat every fourth Thursday in November, not every November 25. You then set **Frequency** to **12** so that the availability repeats every 12 months rather than every month. Also note that in this case you are specifying a period of time when the report is **not available**.

Calendar Name		THANKSGIVING
Date/Time Availability	**Start**	Nov 25 1999 12:00 AM
	End	Nov 26 1999 12:00 AM
	Repeat	Monthly **by** Day
	Frequency	12
	Repeat Until	[Unchecked]

- A combined availability calendar as follows:

  Note You combine availability calendars together to identify which calendars determine when the report is available and not available. Then you identify which of those calendars specify when the report is **not available** by excluding them.

Calendar Name	WORKDAYS_EX_THANKS
Selected Availability Calendars	EVERY_DAY
	THANKSGIVING
Excluded Availability Calendars	THANKSGIVING

Availability calendars: combining example

A report is available every day from 12:00 a.m. to 10:00 p.m. However on Christmas Day and Thanksgiving Day the report is not available at all.

You would create:

- The EVERY_DAY simple availability calendar from Availability calendars: simple example
- The CHRISTMAS simple availability calendar from Availability example: Repeat Yearly example
- The THANKSGIVING simple availability calendar from Availability calendar: monthly frequency example
- A combined availability calendar as follows:

Calendar Name	WORKDAYS_EX_HOLS
Selected Availability Calendars	EVERY_DAY
	CHRISTMAS
	THANKSGIVING
Excluded Availability Calendars	THANKSGIVING
	CHRISTMAS

Alternative solution

You would create:

- The EVERY_DAY simple availability calendar from Availability calendars: simple example
- The CHRISTMAS simple availability calendar from Availability example: Repeat Yearly example
- The THANKSGIVING simple availability calendar from Availability calendar: monthly frequency example
- A combined availability calendar as follows:

Calendar Name	HOLIDAYS
Selected Availability Calendars	CHRISTMAS
	THANKSGIVING
Excluded Availability Calendars	[None]

- A second combined availability calendar as follows:

Calendar Name	WORKDAYS_EX_HOLS
Selected Availability Calendars	EVERY_DAY
	HOLIDAYS
Excluded Availability Calendars	HOLIDAYS

Creating a simple availability calendar

You must have the RW_ADMINISTRATOR role to create an availability calendar.

To create a simple availability calendar:

1 Click  at the bottom of any WebDB page.

2 Click **Oracle Reports Security**.

3 Click **Availability Calendars**.

4 Click **Simple Availability Calendar**, then click **Create**.

5 In the **Calendar Name** field, type a descriptive name for the availability calendar (for example, EVERY_DAY).

6 (Optional) In the **Description** field, type a description for the availability calendar if desired.

7 Click  .

8 In the **Start** fields, choose the start month, date, year, hour, minutes, and division of day (for example, May 17 1999 12:00 AM).

9 In the **End** fields, choose the end month, date, year, hour, minutes, and division of day (for example, May 21 1999 10:00 PM).

10 Click the appropriate radio button depending on how often you want the availability to repeat:

- Click **Occurs Only Once** if you do not want the availability to be repeated.

  Suppose a report is available from May 17 1999 through May 28 1999, after which it becomes obsolete and should no longer be available. You could create an availability calendar with a start date of May 17 1999, an end date of May 28 1999, and click **Occurs Only Once**.

- Click **Yearly** if you want to repeat the availability every year.

  Suppose a report is available on May 17 every year. You could create an availability calendar with a start and end date of May 17 1999 and click **Yearly**.

- Click **Monthly** if you want to repeat the availability every month.

  Suppose a report is available only on the first of every month. You could create an availability calendar with a start and end date of May 1 1999 and click **Monthly**.

  Note In this example, you also need to choose **by Date** to ensure that the availability repeats on the same date of every month (that is, June 1 1999, July 1 1999, and so on). If you choose **by Day**, the availability repeats on the same day of the week every month (that is, Saturday May 1 1999, Saturday June 5, 1999, Saturday July 3 1999, and so on).

- Click **Weekly** if you want to repeat the availability every week.

  Suppose a report is available every Monday. You could create an availability calendar with a start and end date of May 17 1999 and click **Weekly**.

- Click **Daily** if you want to repeat the availability every day.

  Suppose a report is available every day of the week (Monday through Sunday). You could create an availability calendar with a start and end date of May 17 1999 and click **Daily**.

11 If you click **Monthly**, choose whether to repeat the availability **by Date** or **by Day**.

- If you choose **by Date** the availability will repeat on the same date each month (for example, the first of every month: January 1, February 1, March 1, etc.)
- If you choose **by Day** the availability will repeat on the same day each month (for example, the first Monday of every month).

12 In the **Frequency** list, choose how regularly the availability repeats for the **Repeat** setting (Daily, Weekly, Monthly, Yearly).

Suppose you want a report to be available all day on every other Friday. You would create an availability calendar with a start and end date of May 21 1999, click **Weekly**, and set **Frequency** to 2.

13 If you want the availability to expire at a certain time, check **Repeat Until** and choose the expiration month, date, year, hour, minutes, and division of day.

Suppose you want to make a report available every day until the end of the year. You could create an availability calendar with a start and end date of May 17 1999, click **Daily**, check **Repeat Until**, and specify December 31 1999.

14 Click  .

15 The Summary page summarizes the choices you have made in the wizard.

Click **Show Calendar** for a visual representation of your availability calendar. You can view the calendar by day, week, month, or year.

16 Click  .

17 If you are satisfied with your choices in this wizard, click **OK**. The availability calendar will be created.

Note If you are not satisfied with the choices in the wizard and want to return to any of the previous pages, click  . Do not click your browser's Back button.

Note If a report is running when the availability ends, the report continues to run until the request has completed.

Creating a combined availability calendar

You must have the RW_ADMINISTRATOR role to create an availability calendar.

To create a combined availability calendar:

1. Click  at the bottom of any WebDB page.
2. Click **Oracle Reports Security**.
3. Click **Availability Calendars**.
4. Click **Combined Availability Calendar**, then click **Create**.
5. In the **Calendar Name** field, type a descriptive name for the availability calendar (for example, WORKDAYS).
6. (Optional) In the **Description** field, type a description for the availability calendar if desired.
7. Click  .
8. In the **Availability Calendars** list, choose the availability calendars that you want to combine.
9. Click  to move the chosen availability calendars to the **Selected Availability Calendars** list.

   Suppose a report is available on Monday and Friday. You could have two simple availability calendars: one for Monday and one for Friday. You could then combine the Monday and Friday availability calendars into a single combined availability calendar.

10. Click  .
11. In the **Availability Calendars** list, choose the availability calendars that define the time when the Oracle Reports report, Reports Server, or printer is **not available**.
12. Click  to move the chosen availability calendars to the **Excluded Availability Calendars** list.

    Suppose that none of your reports are available on Christmas Day. You could create two simple availability calendars: one for every day and one for Christmas Day, where the Christmas Day availability calendar has a start date and time of December 25 1999 12:00 a.m. and an end date and time of December 26 1999 12:00 a.m. (that is, all day on December 25 1999) repeated yearly. You could then create a combined availability calendar which combines the every day and Christmas Day calendars and then **excludes** the availability defined in the Christmas Day calendar.

13. Click  .
14. The Summary page summarizes the choices you have made in the wizard.

    Click **Show Calendar** for a visual representation of your availability calendar. You can view the calendar by day, week, month, or year.

15. Click  .

16 If you are satisfied with your choices in this wizard, click **OK**. The availability calendar will be created.

Note If you are not satisfied with the choices in the wizard and want to return to any of the previous pages, click  . Do not click your browser's Back button.

Restricting required parameters

Required parameters are the parameters needed to make a basic report request. The required parameters are:

- **Destination Types** The type of devices to which the report output can be sent
- **Destination Formats** The formats in which the report output can be presented
- **Reports Printers** The printers to which the report output can be sent (this applies only if the Destination Types includes Printer)

For security purposes, you restrict the values to which the required parameters can be set. For example, you may have provided access to many printers, but want the output of a particular report definition file to only be sent to a subset of those printers. In the Required Parameters page of the Report Definition File Access Wizard, you can choose the subset of printers to which the report output can be sent. This subset of printers can then be listed on the report's Runtime Parameter Form where the user can choose which one of those printers to send the report output to.

For information about controlling the fields on the Runtime Parameter Form, see Adding report definition file access.

To restrict required parameters:

1 In the **Destination Types** list, choose the destination types to which the report output can be sent.

2 In the **Destination Formats** list, choose the formats in which the report results can be output.

3 If you chose **Printer** in the **Destination Types** list, in the **Reports Printers** list, choose the printers to which report output can be sent.

Note The printers in this list are the ones associated with the Reports Servers chosen on the Report Name and Schema page of the Report Definition File Access Wizard. If a printer that you want to choose does not appear in this list, the printer may not be available to the Reports Servers you chose, or you may have not yet provided access to that printer. For information about how to make printers available to Reports Servers, see Adding Reports Server access. For information about how to add printer access, see Adding printer access.

4 In the **Parameter Form Template** list, choose the template that will be used to determine the appearance of the report's Runtime Parameter Form. To see what the template you have chosen looks like, click **Preview Template**.

For more information about templates, see About UI templates.

Restricting optional parameters

In addition to restricting the values of required parameters, you can also restrict the values of other optional parameters in an Oracle Reports report definition file. There are two types of optional parameters:

- **System parameters** are created by Report Builder, for example PAGESTREAM orCOPIES
- **User parameters** are created by the developer of the report

For example, suppose you have a salary report with a user parameter EMPID that enables a user to display only data associated with a particular employee. You could provide SMITH, a manager who wants to review the salary history of his team, with access to the report. In the Optional Parameters page of the Report Definition File Access Wizard, you could restrict the EMPID parameter to those employees on SMITH's team (e.g., those with employee IDs ranging from 1 to 10). When SMITH runs the report, he will only have access to information about his direct reports.

Optional parameters can be listed on the report's Runtime Parameter Form where the user can choose which values to use for processing the report. For information about controlling the fields on the Runtime Parameter Form, see Adding report definition file access.

To restrict optional parameters:

1. In the **Parameter Name** field, type the name of the system or user parameter. The name here must be the same as the name of the parameter in the Oracle Reports report definition file.
2. (Optional) Specify the values that the parameter can be set to by completing one of the following steps:
   - In the **LOV** field type the name of the List of Values that specifies the values that the parameter is restricted to. Click  to search for Lists of Values. The List of Values must already exist. For information about building a List of Values, see Building a static List of Values or Building a dynamic List of Values.
   - In the **Low Value** and **High Value** fields, specify the range of values to which the parameter can be set.

Notes

- If you do not specify values in the **LOV Name** or **Low Value** and **High Value** fields, the values of the optional parameter will not be restricted.
- The Optional Parameters page of the Report Definition File Access Wizard includes five rows for optional parameters. If you have more than five optional parameters, after entering details for the first five, click **More Parameters** to add more rows to the page.

Specifying a validation trigger

A validation trigger is a PL/SQL function that is executed when the user accepts the Runtime Parameter Form. Validation triggers are used to validate the Initial Value property of the parameters. The function must return a Boolean value (TRUE or FALSE). If the function returns TRUE, the report is processed using the values specified in the Runtime Parameter Form. If the function returns FALSE, an error message is displayed and the user can return to the Runtime Parameter Form to specify different parameter values.

To specify a validation trigger:

In the **Validation Trigger** field, type the PL/SQL code to define the validation function.

You can access parameters in your PL/SQL code. For example, the following validation function enables a user to run a report request only when the Destination Type system parameter is Printer and the EMPNAME user parameter is SMITH.

```
is
begin
   if upper(p_destype)='PRINTER' then
      return(true);
   else
      return(false);
   endif
end;
```

Oracle Reports destination types

The following table lists the devices to which you can send Oracle Reports report output:

Device	Description
Cache	Sends the output directly to the Reports Server's cache.
File	Saves the output to a file.
	Note A field (DESNAME) is automatically added to the Runtime Parameter Form where the user can specify the name of the file to save the output to.
Printer	Routes the output to a printer.
Mail	Sends the output to a e-mail address.
	Note A field (DESNAME) is automatically added to the Runtime Parameter Form where the user can specify the e-mail address to send the output to.

Oracle Reports destination formats

The following table lists the formats in which you can produce Oracle Reports report output:

Format	Description
HTML	Report output is sent to a file that can be read by an HTML 3.0 compliant browser (for example, Netscape Navigator 2.2 or later).
HTMLCSS	Report output is sent to a file that includes style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.
PDF	Report output is sent to a file that can be read by a PDF viewer.
RTF	Report output is sent to a file that can be read by standard word processors, such as Microsoft Word. **Note** When you open the file in Word, you must choose View-->Page Layout to view all the graphics and objects in your report.
Delimited	Report output is sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. **Note** If you choose to enable output to be produced in delimited format, the default delimiter character is TAB. If you want to enable the user to specify a different delimiter, you must include a DELIMITER parameter on the Optional Parameters page of the Report Definition File Access Wizard.
PostScript	Report output is sent to a file that can be read by a PostScript printer or viewer.
Character	The report is run in character mode. **Note** If you choose to enable output to be produced in character format, you must include a DESFORMAT parameter on the Optional Parameters page of the Report Definition File Access Wizard that the user can use to set the printer definition. Ask your system administrator for a list of valid printer definitions.

Monitoring WebDB and the Oracle Database

About monitoring features

WebDB provides charts and reports that you can use to monitor end user activity, component performance, and database activity. In addition, you can create your own custom monitoring reports and charts using one of the component build wizards.

End user activity

End user requests for components are logged in the WebDB activity log. The log includes information about the time of the request, the end user who made the request, the machine and browser that was used, and when a WebDB Developer created or last edited the component.

User requests are entered into the activity log if a WebDB Developer specified an option for logging the requests when creating or editing the component.

Component performance

You can view the volume of requests for a particular component to see which are most frequently used, as well as the time your system takes to respond to those requests.

Database activity

You can monitor information about the database where WebDB is installed, including:

- Database storage allocated to users, objects, and tablespaces
- Memory allocation
- Object creation dates
- Object created during a given time span
- Rollback segment attributes
- Session locks, redo logs, and DBMS jobs

Creating your own monitoring reports

WebDB provides a Query by Example form that allows you to track the component and performance information that WebDB records in the Activity Log. In addition, you can write your own custom SQL queries against the log, or create your own customized charts and reports. To create a customized chart or report:

- You must obtain Build In privileges in the schema that owns the table containing the Activity Log. This is the same schema where WebDB was installed.
- Specify WWV_ACTIVITY_LOG as the name of the table on which you are building a component on the Table or Views page of the report or chart build wizard.

In the build wizard, you can choose which columns of the WWV_ACTIVITY_LOG table to display in the report or chart, specify conditions to limit the data to display, and set a look and feel for the monitoring report of chart.

Terminating sessions

To terminate a current user session:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Sessions and Memory Structures**, then **Find and Kill Session(s)**.

   A report displays all currently connected users and other information such as how long a user has been connected and when they logged on.

3. Locate the session you want to terminate in the report, then click **KILL**.

Monitoring WebDB Users

Viewing storage allocation by user

To view a chart of the storage used by WebDB users:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Objects**, then **Top Storage Consumption by Database Schema**.

   A chart displays database storage consumed by all WebDB users.

3. (Optional) Click a user name to view all database objects owned by the user and the size of each object in bytes.

Note

- User names are synonymous with database schema names in WebDB.

Viewing when users created objects

To view a calendar that displays the number of objects created by users each day, including WebDB components stored in the database as packaged procedures:

1. Click  at the bottom of any WebDB page.
2. Click **Database Objects**, **Objects**, **Recently Created Objects**.

   A calendar displays the users who created objects and the number they created on each day.

3. (Optional) Click a user name to view more information about the objects created by the user on that day, including object names, creation times, and object status.

Viewing user I/O activity

To view a chart of I/O activity by users currently connected to WebDB:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Sessions and Memory Structures**, then **Top Activity by Username.**

   A chart displays the I/O session activity for all users currently connected to WebDB.

Managing user locks on components

When a user edits a component, WebDB locks the component to prevent other users from accessing the component version that the first user is editing. The Component Options page for the component displays a status of EDIT. The component remains locked until the first user finishes editing it.

A user with the DBA role can monitor currently locked components, and if necessary, override a user lock.

To force open a component a user currently has locked:

1. Click  on the bottom of any WebDB page.

2. Click **Manage Locks on Components**.

   A report of current user locks on WebDB components displays.

3. Click **Unlock** next to the component you want to unlock.

Viewing a report of connected users

To view all users currently connected to WebDB:

1. Click  at the bottom of any WebDB page.
2. Click **Database Objects**, **Sessions and Memory Structures**, then **Sessions**.

   A report displays all currently connected users and other information such as how long users have been connected and the machine they connected to.

Monitoring WebDB Components

Viewing component usage information

To view components that were requested since 12:01 AM today:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **Requests by Component Name**.

   A chart displays each component requested since 12:01 AM today and the number of page requests for each component.

3. (Optional) Click a component name to view users who requested the component and the number of times they made the request.

To view users who requested components since 12:01 AM today:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **Requests by Database User**.

   A chart displays each user who made at least one component page request since 12:01 AM today and the user's total number of requests for all components in that time period.

3. (Optional) Click a user name to view a report of all components that the user requested and the time of the request.

To view component page requests per day and hour:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **Requests by Day and Hour**.

   Two charts display. The first shows the number of page requests per day for the last 7 days. The second shows the number of page requests per hour since 12:01 AM today.

3. (Optional) Click a day on the **Distribution of Activity by DAY** chart to view all users who requested components on that day and the number of requests each user made. Click an hour on the **Distribution of Activity by HOUR** chart to view all users who requested components during that hour and the number of requests each user made.

Note

- User names are synonymous with database schema names in WebDB.

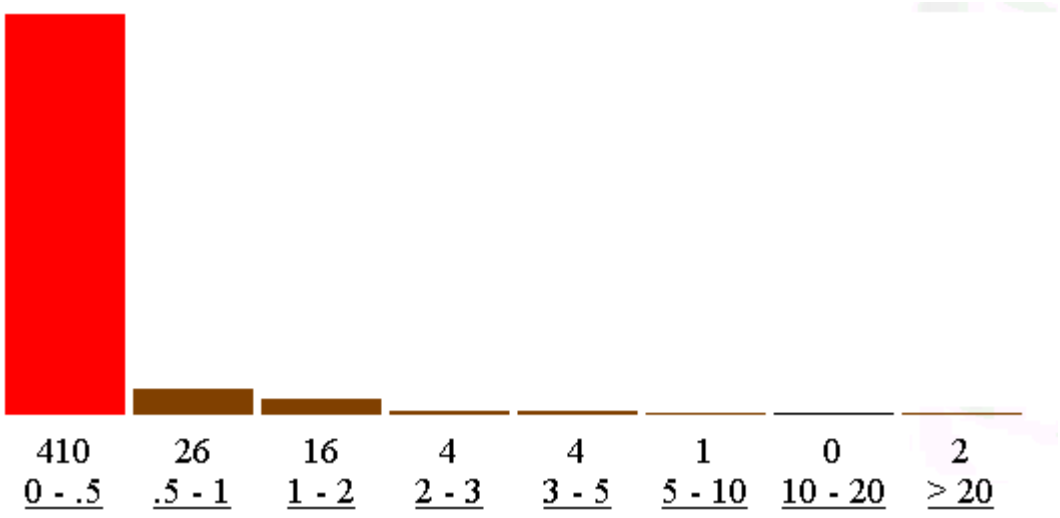
Viewing component performance information

To view a chart that displays the number of successful WebDB responses to component page requests within half second intervals after the requests were initiated:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **Response Times**.

   A chart displays WebDB responses to component page requests in half second intervals. In the example below, WebDB responded to 410 component page requests within a half second of receiving them.



3. (Optional) Click a response interval to view details about each request completed within the interval, including the name of the component that was requested, the user who requested it, and when the request occurred.

Note

- Response time equals the interval from when WebDB receives the request to display the component to when the HTML for the component is generated. It doesn't include the time required to display the component in the user's Web browser.

Viewing component requests by browser type

To view component page requests by browser type:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **Requests by Browser Type**.

   A chart displays showing the total number of component page requests originating from each supported browser type since 12:01 AM today.

3. (Optional) Click a browser type to view information about the requests that originated from the browser. This information includes the names of the components that were requested, the users who requested the components, the time when the requests were made, and how long WebDB took to respond to each request.

Viewing component requests by IP address

To view component page requests by IP address:

1. Click  at the bottom of any WebDB page.

2. Click **User Interface Components**, then **IP address**.

   A chart displays showing the total number of component page requests originating from each IP address connected to WebDB since 12:01 AM today.

3. (Optional) Click an IP address to view information about the requests that originated from the IP address. This information includes the name of each component that was requested from a user or users at that IP address, user names, the time when each requests were made, and how long WebDB took to respond to each request.

Using the activity log

About the Activity Log

The Activity Log contains a record of logged user requests for WebDB components. A request is logged if the developer who created the component selected the Log Activity option in the **Display Options** step of the component build wizard. The log includes information about the time of the request, the user who made the request, and the machine and browser type that originated the request.

WebDB provides options that allow you to browse the activity log using a Query by Example form, and to set how many days the current log will record information before switching to a new log.

Browsing the activity log

To query and display the contents of the WebDB activity log:

1. Click  at the bottom of any WebDB page.
2. Click **Browse Activity Log**.

   A QBE form displays. The QBE form contains entry fields that correspond to each column in the activity log; for example, **Time Stamp** or **Component Type**.

3. Type or select your query criteria in one or more of the column entry fields. For example, to query the log for all activity that occurred before 5PM today, type `<17:00` in the **Time Stamp** entry field.

4. Check the check box next to each activity log column name you want to display in your query results.

5. Click **Query**. A report displays containing your query results.

Note

- For more information about how to enter values in QBE column entry fields, click the small  button on the QBE form page.

Setting the Activity Log interval

To specify the number of days that the current Activity Log will log requests for components until switching to a new log:

1. Click  at the bottom of any WebDB page.

2. Click **Configure Activity Log**.

3. In the **Log Switch Interval** field, type the number of days that you want the current Activity Log to log component requests.

4. Click **Apply**.

   (Optional) Click **View Log Information** to view additional information about the Activity Log such as the first and most recent entries in the current log and the name of the database objects that support the log.

Monitoring the Oracle database

Viewing database version information

To view your database version and database option versions:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **About Your Database**, then **Version Information**.

Viewing HTP package installations

To view Oracle Web Application Server HTP packages currently installed:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Objects**, **HTP Package Installations**.

   A report displays showing all installations of the HTP package.

3. If more than one copy of the HTP package is currently installed, remove additional copies.

Viewing storage allocated to objects

To storage allocated to each object in the database.

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Objects**, then **Top Storage Consumption by Database Object**.

   A chart displays showing the amount of space in bytes allocated to each object.

3. (Optional) Click an object to view more information about it such as its size, creation date and the schema that owns it. If the object is a table, you can browse it using a Query by Example form.

Viewing storage allocation by tablespace

To view a chart of tablespace sizes:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Objects**, **Storage**, then **Chart of Tablespace Utilization.**

   A chart displays the number of used and free bytes in each tablespace, and total tablespace size.

To view the same information formatted as a report instead of a chart:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Objects**, **Storage**, then **Report of Tablespace Utilization.**

Viewing datafile information

To view how tablespaces map to operating system datafiles:

1. Click  at the bottom of any WebDB page.
2. Click **Database Objects**, **Storage,** then **Datafiles**.

To view a report of datafile read and write activity:

1. Click  at the bottom of any WebDB page.
2. Click **Database Objects**, **Storage**, then **Report of Datafile Activity**.

To view the same information formatted as a chart instead of a report:

1. Click  at the bottom of any WebDB page.
2. Click **Database Objects**, **Storage**, then **Chart of Datafile Activity**.

Viewing locked sessions

To view a report of sessions locking other sessions:

1. Click  at the bottom of any WebDB page.

2. Click **Database Objects**, **Locks,** then **Blocking Sessions**.

Creating Web Sites

About creating WebDB sites

The site you create using the WebDB Site Creation Wizard is contained entirely within the Oracle database where you installed WebDB. When you create the site, all the tools you need to control it are also installed in the database. You access these tools from the site itself.

After the Web site is created, you can navigate to it by clicking  at the bottom of any WebDB page, then clicking the Web site's name in the **Select a Recently Created Site: Name** list on the WebDB Site Creation Wizard page.

Once in the site, you typically assign other users as site administrators.

Creating WebDB sites

 To create a site, you must be a WebDB schema owner or have the following privileges:

- the DBA role
- Execute and Grant privileges on dbms_sys_sql.

After you create the WebDB site, you must log in to the administration account and change the passwords for the Owning Schema and administration account.

To create a new Web site:

1. Click  at the bottom of any WebDB page.

2. Click **Create**.

   The first page of the Site Creation Wizard appears. Follow the instructions shown on each page of the wizard until the last page. You can click  to return to any page to make changes.

   The last page of the wizard contains the following information:

   - A summary of the options you selected
   - The names of the Owning Schema and user accounts that will be created and their passwords
   - The URLs for the site

3. Make a note of the user name and password of the Administration user account and the URL for public access to the site. Click the small  button at the top of this page or any other page in the wizard for more information.

4. If you are satisfied with your choices, click **Finish** on the last page to start the site creation process.

   WebDB displays a status page to indicate the progress of the installation. Once installation is complete, click **Done**.

5. You can navigate to the Web site by clicking  at the bottom of any WebDB page, then clicking the site's name in the **Select a Recently Created Site: Name** list on the WebDB Site Creation page.

   The WebDB Site Install Menu displays.

6. Click **Site Administration** to display the Site Administration page. You will be prompted to log on. Type in the user name and password that you noted in step 2.

7. Change the default passwords for the Owning Schema and site administration user account you just created.

Changing default passwords after creating a Web site

Immediately after you create a WebDB site, you must log in to the administration account and change the passwords for the Owning Schema and administration account.

To change these passwords:

1. Click  at the bottom of any WebDB page.

   The Site Building page displays.

2. Click the name of the site you just created in the **Select a Recently Created Site: Name** list.

3. Click **Site Administration**. Type the user name and default password for the site administration user account; for example, `MYSITE_ADMIN/MYSITE_ADMIN`.

4. Click  or the **Administration** link from the site's navigation bar.

5. From the Administration page, click **User** under Access Managers to display the User Manager.

6. In the **Find User: User Name** field, type the user name you specified when you logged into the site; for example, MYSITE_ADMIN.

7. Click the Details tab.

8. Type the new password in the **Password** field.

9. In the **Confirm Password** field, type the password again to verify that you have entered it correctly.

10. Click **Apply.**

11. Repeat steps 6-10 to change the password for the Owning Schema.

Reference

Datatype icons

Description WebDB uses the following icons to represent datatypes in Query by Example forms.

	CHAR, VARCHAR2 **Note:** WebDB accepts up to 32K in VARCHAR2 data entry fields.
	DATE
	NUMBER
	RAW

Supported database object types

All WebDB components are based on objects in the Oracle database. WebDB supports these database object types:

Database object type	Button	Description
cluster		Created with the Oracle CREATE CLUSTER command. Clusters store tables that are related to one another and are often joined together in the same area on disk.
function		Created with the Oracle CREATE FUNCTION command. Functions are PL/SQL subprograms that perform a specified sequence of actions, and then return a value.
index		Created with the Oracle CREATE INDEX command. An index is an optional structure associated with a table used to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.
library		A library is a collection of one or more PL/SQL program units that are stored together in a database, that can be referenced by several applications at once.
package		Consists of a specification created with the Oracle CREATE PACKAGE command and an option body, created with the CREATE PACKAGE BODY command.
procedure		Created with the Oracle CREATE PROCEDURE command. A procedure is a PL/SQL subprogram that performs a specified sequence of actions.

sequence		Created with the Oracle CREATE SEQUENCE command. A sequence is a database object used to automatically generate numbers for table rows.
snapshot		Created with the Oracle CREATE SNAPSHOT command. A snapshot is a table that contains the results of a query on one or more tables, called master tables, in a remote database.

snapshot log		Created with the Oracle CREATE SNAPSHOT LOG command. A snapshot log is a table associated with the master table of a snapshot. It tracks changes to the master table.
synonym		Created with the Oracle CREATE SYNONYM command. A synonym is a name assigned to a table or view that can thereafter be used to refer to it.
table		Created with the Oracle CREATE TABLE command. A table is the basic storage structure in a relational database.
trigger		Created with the Oracle CREATE TRIGGER command. A trigger is a stored procedure associated with a table. It executes on one or more specified events.
type		User-defined, created with the Oracle CREATE TYPE command (available in Oracle 8 or higher).
view		Created with the Oracle CREATE VIEW command. A view is a virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

Notes

- You can browse for, but not use in WebDB components, these Oracle 8.1.5 object types:

  Consumer Groups

  Folder Images

  Java Classes

  Java Resources

  Resource Plans

Queues

Index Types

Operators

- WebDB uses the terms component and object to distinguish between the dynamically generated HTML Web pages and content that WebDB creates, such as charts and forms, and the Oracle database objects on which these components are based.

Supported database object privileges

You can grant the following Oracle database object privileges to users or roles using features within the WebDB Grant Manager. Use features within the Oracle database to grant object privileges not listed below.

Privilege	Grants the user or role permission to
SELECT	Query (i.e., fetch data from) a table, view, or sequence object using an Oracle SELECT statement.
INSERT	Insert a row into a table or view using an Oracle INSERT statement.
UPDATE	Update a table or view using an Oracle UPDATE statement.
DELETE	Delete a row from a table or view using an Oracle DELETE statement.
EXECUTE	Execute a function or procedure object using an Oracle EXECUTE statement.

Note

- If an object is being used to build a WebDB component, any required object privileges such as SELECT, INSERT, UPDATE, DELETE cannot be granted through a role. These object privileges must be explicitly granted to the Component Schema using the **Grant Manager**.

List of Values display formats

You can choose these formats for displaying Lists of Values on a form or component parameter entry form.

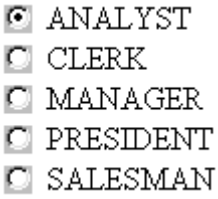
Field

Enables the end user to view an initial default value or click a button to see all values in the list. There are two types of fields:

- Single selection field - end user selects one value only
- Multiple selection field - end user selects one or more values

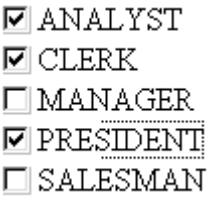
Radio group

Enables the end user to select one value at a time from a group of radio buttons.



Check box

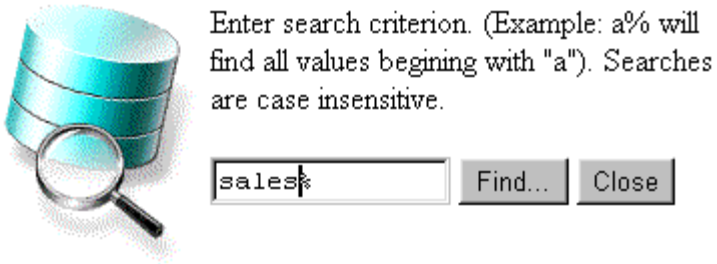
Enables the end user to select one or more values from a group of check boxes



Pop-up list

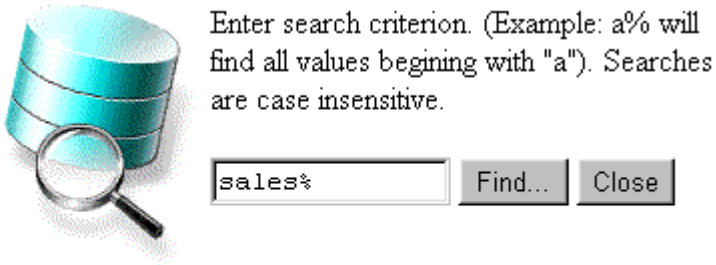
Enables the end user to enter values in a field. If the user is unsure what to enter, clicking displays a search dialog. The search dialog displays in a separate window that appears on top of the Web browser window.

The end user can enter search criteria in the dialog, including the % wildcard.



Clicking **Find** displays a list of all values that match the criteria.

The end user can then click a result in the list to make a selection.



SALESMAN

Pop-up lists are useful for displaying lists with large numbers of values. For example, to enable an end user to choose `Employee Names` from a table containing a thousand employees, you'd display a pop-up listrather than list all of these names in a field, radio group, or check box group.

Examples

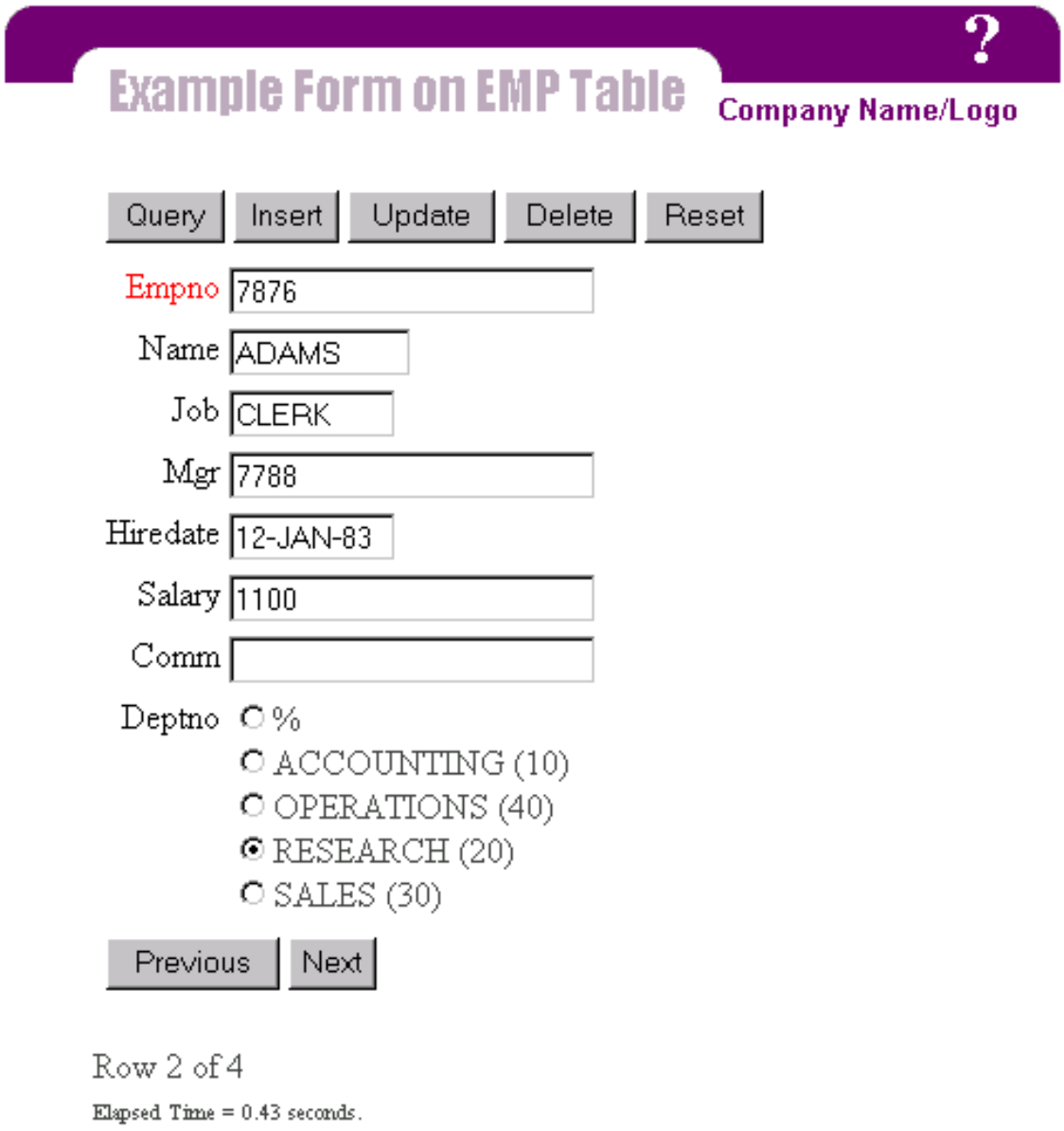
About examples in this document


All of the examples and example components described in this document are based on the EMP table in the SCOTT schema. To view these examples or use SCOTT.EMP to practice building your own components, you must ask your DBA for Build In privileges in the SCOTT schema.

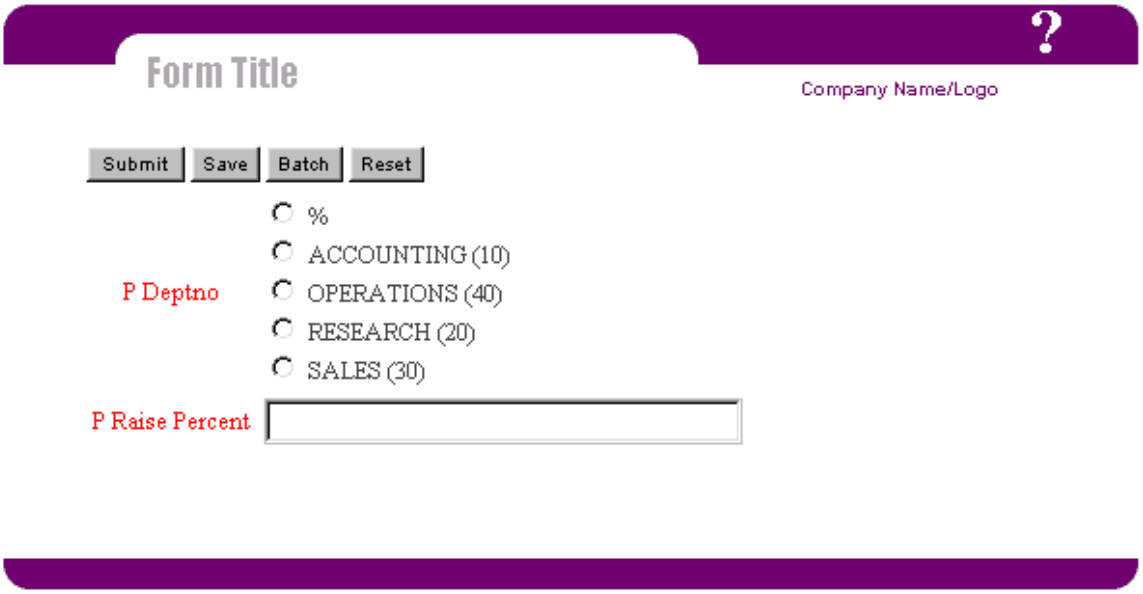
Example forms

An example form based on the SCOTT.EMP table is shown below. The form allows an end user to query, insert, update, or delete rows in SCOTT.EMP.

Next and **Previous** pagination buttons appear at the bottom of the form after an end user enters query criteria and clicks **Query**. The buttons enable end users to scroll through table or view rows returned by the query one row at a time. The total number of rows returned by query and the row currently displayed are also shown at the bottom of the page (`Row 2 of 4`).


?
Example Form on EMP Table
Company Name/Logo
Query
Insert
Update
Delete
Reset
Empno 7876
Name ADAMS
Job CLERK
Mgr 7788
Hiredate 12-JAN-83
Salary 1100
Comm
Deptno
%
ACCOUNTING (10)
OPERATIONS (40)
RESEARCH (20)
SALES (30)
Previous
Next
Row 2 of 4
Elapsed Time = 0.43 seconds.

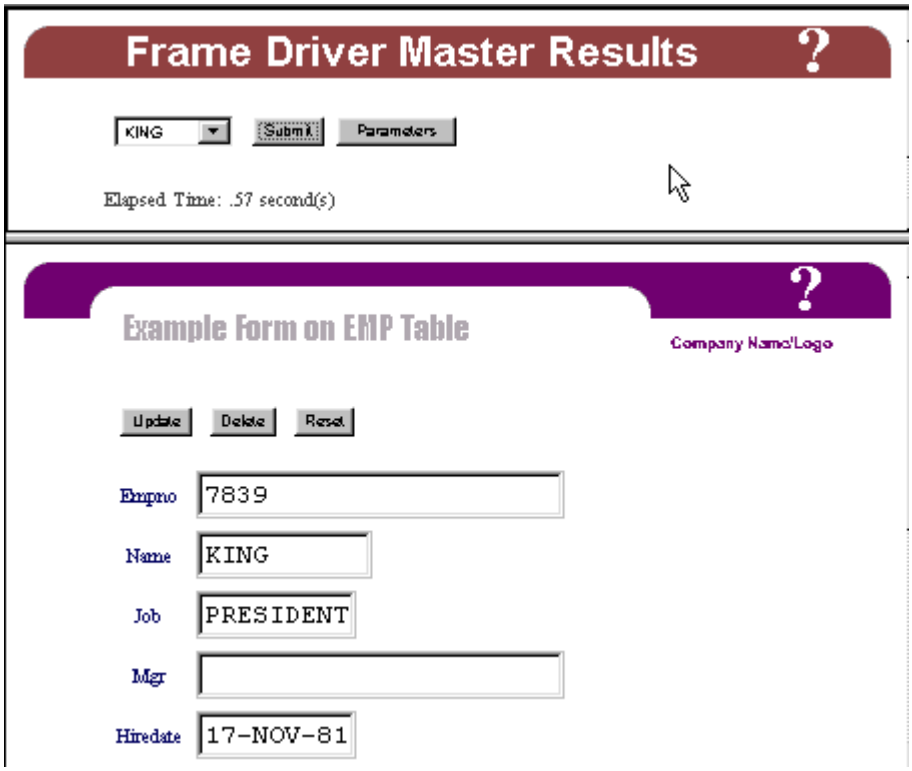
An example form based on a procedure is shown below. The form allows an end user to grant a raise to all members of a selected department.



Example frame driver

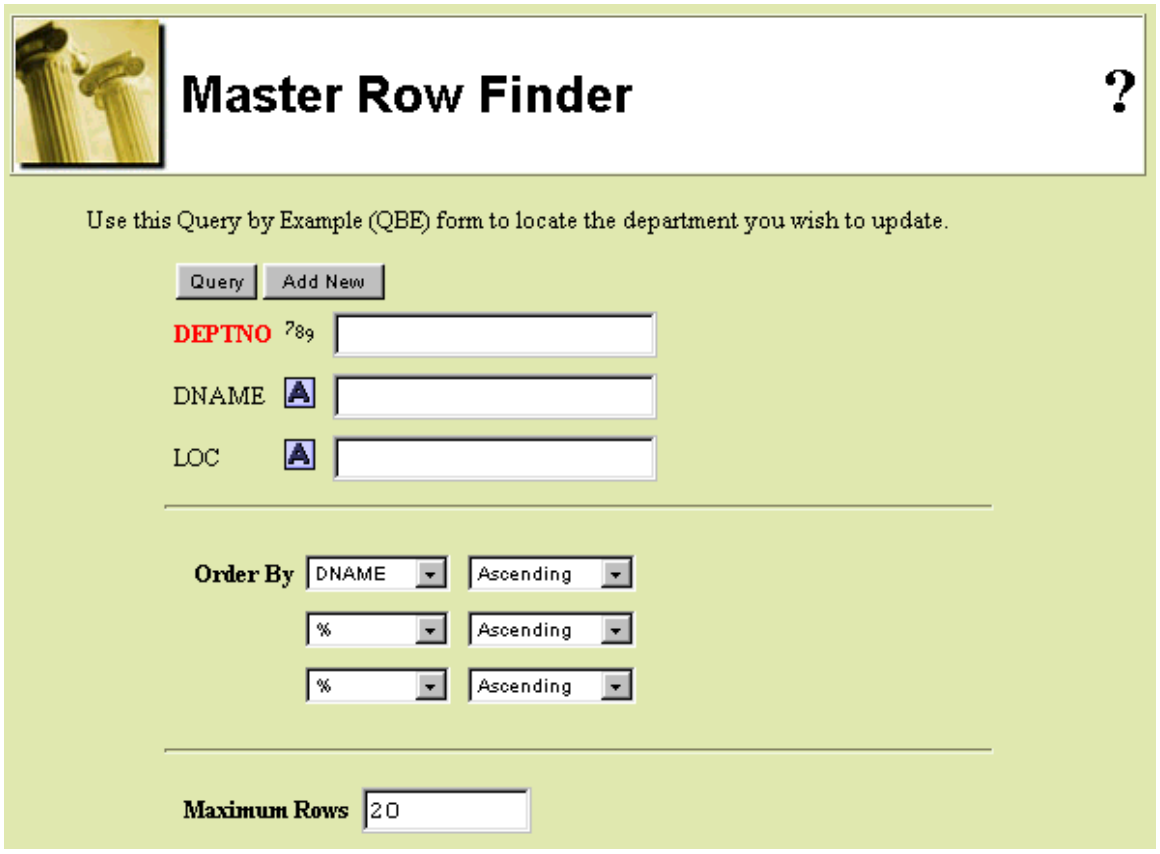
An example frame driver based on SCOTT.EMP is shown below. An end user can choose an employee name from the drop-down list in the upper frame to display a form containing information about the employee in the lower frame.

The end user can update the employee's information before choosing another employee name in the upper frame.



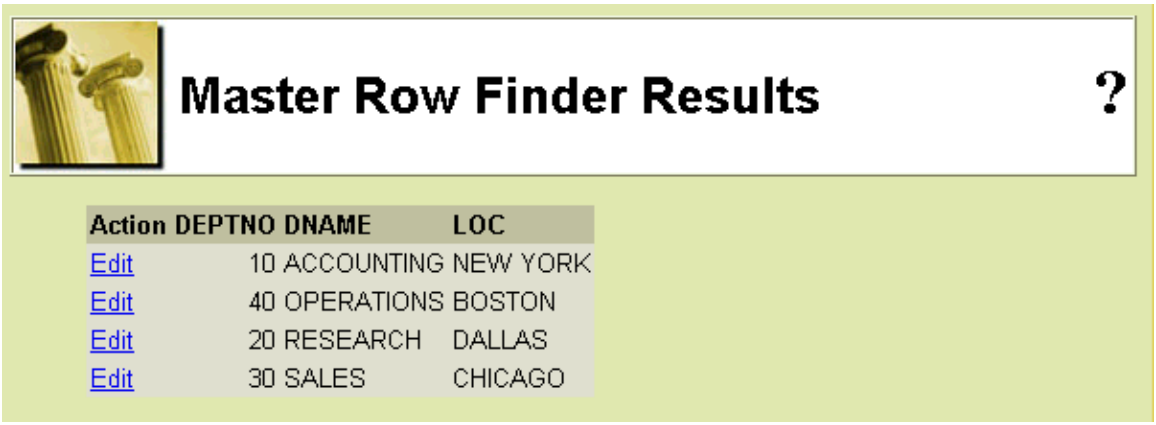
Example Master Row Finder page

The following is an example of a Master Row Finder page, based on the SCOTT.DEPT master table. End users can find master rows in the SCOTT.DEPT, or update master rows by specifying values in the column entry fields on the form.



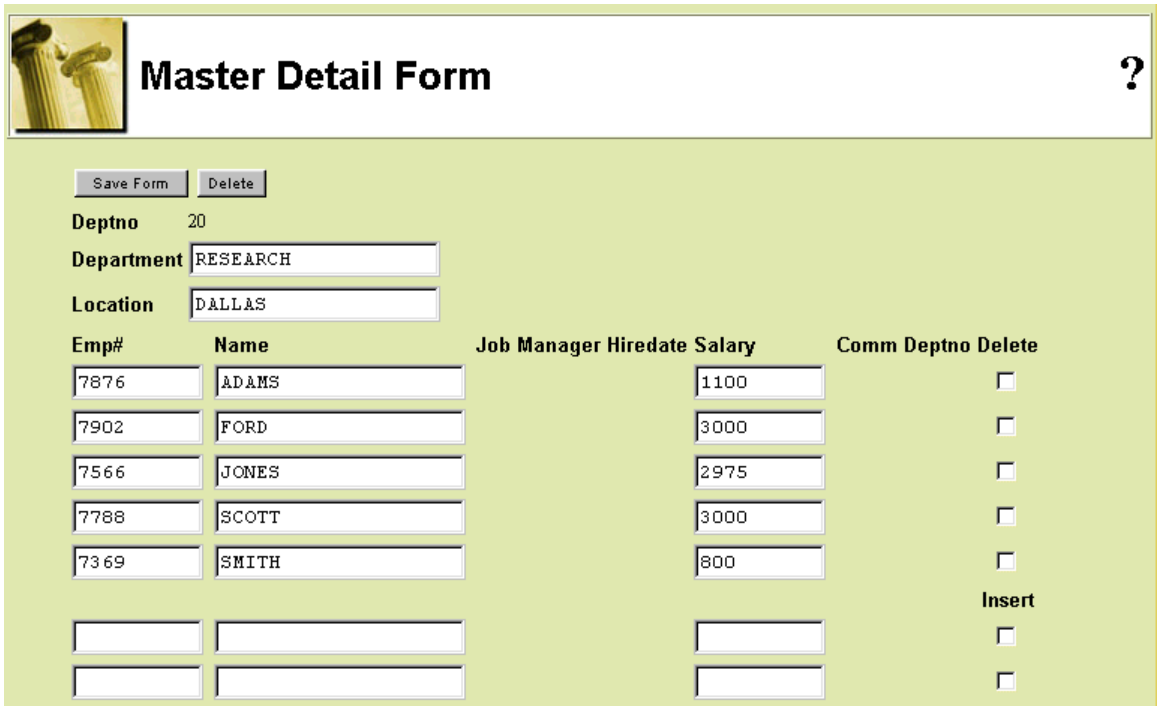
Example Master Row Finder Results page

The following shows the results of querying the SCOTT.DEPT table using the Master Row Finder page. The figure displays master rows in the SCOTT.DEPT table. An end user clicks **Edit** next to a master row to open a Master-Detail form. The Master-Detail form will contain the master row that the user selected and its associated detail rows.



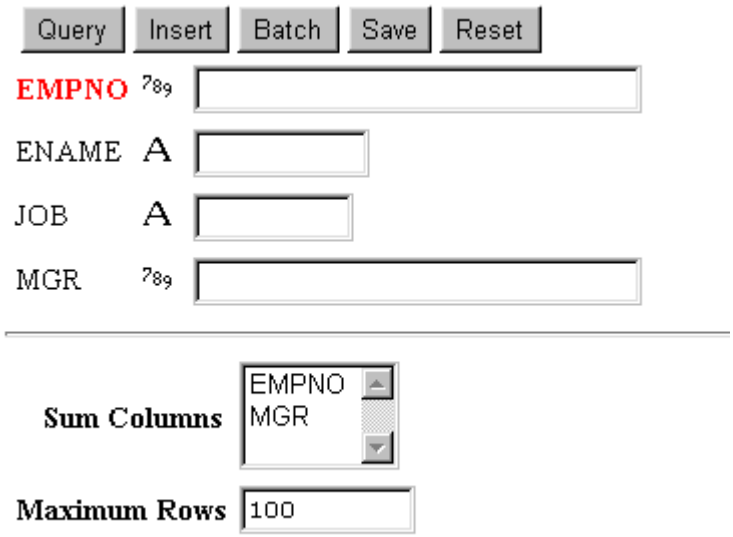
Example Master-Detail form

The following shows a Master-Detail form for the Department 20 row of the SCOTT.DEPT master table. End users of the form can update or delete the master row. They can also insert, update, or delete detail rows in the SCOTT.EMP table.



Example QBE form

An example QBE form based on four columns of the SCOTT.EMP table is shown below.



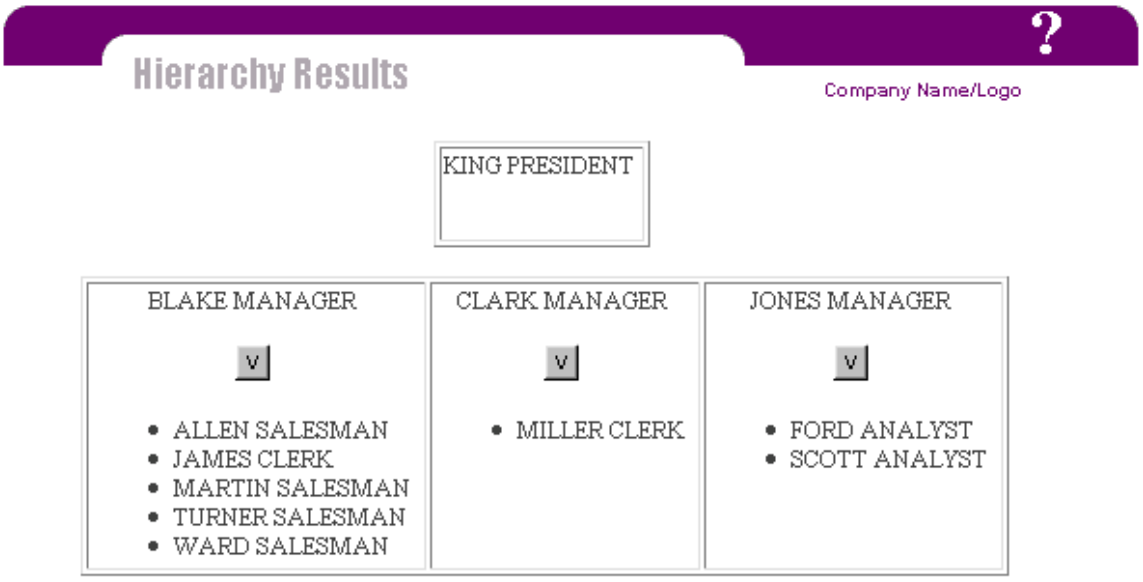
The page below displays the results from clicking the Query button in the sample QBE form shown above.

EMPNO	ENAME	JOB	MGR
7369	SMITH	CLERK	7902
7499	ALLEN	SALESMAN	7698
7521	WARD	SALESMAN	7698
7566	JONES	MANAGER	7839
7654	MARTIN	SALESMAN	7698
7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839
7788	SCOTT	ANALYST	7566
7839	KING	PRESIDENT	(null)
7844	TURNER	SALESMAN	7698
7876	ADAMS	CLERK	7788
7900	JAMES	CLERK	7698
7902	FORD	ANALYST	7566
7934	MILLER	CLERK	7782

Example hierarchy

An example hierarchy based on the SCOTT.EMP table is shown below. The hierarchy was created using the EMPNO column of SCOTT.EMP table as the Primary Key column, and MGR as the Parent Key column. The topmost level displays a manager, and on the next level, all employees who report to the manager.

End users can click the  arrow to drill down to another level of the hierarchy.



Tips

How do I make the WebDB site's Home page point to an HTML file?

To make the WebDB site's Home page point to an HTML file:

1 In a different schema from WEBDB, create or replace procedure index_html with text similar to the following:

```
begin
 htp.htmlOpen;
 htp.headOpen;
 htp.meta('Refresh', '', '0;
URL=http://tool.oracle.com/images/hello.html');
 htp.headClose;
 htp.htmlClose;
 end;
```

2 Save your index_html file to the `C:\orant\webdb\images\` directory.

3 Create the above page, index_html, in the WEBDB schema.

4 Access the page; for example:

```
http://tools-80-pc.us.oracle.com/WebDB/WEBDB.index.html
```

5 Make sure you set the default WebDB Home page to `WEBDB.index.html`.

How can I gain access to WebDB without logging on?

Typically, you do not want to eliminate manual authentication as this would break security. Doing so would allow any user with a Web browser to have access to WebDB and the database on your machine.

By default, Oracle WebDB requires manual authentication to gain access to WebDB. Thus, when you type the URL to access WebDB, you are prompted for a user name and password. To bypass manual authentication, you need to store the user name and password in the DAD.

To gain access to WebDB without logging on:

1. Click  at the bottom of any WebDB page.
2. Click **Listener Settings**. The Oracle WebDB PL/SQL Gateway Settings page displays.
3. In the appropriate Database Access Descriptor Settings panel, type the Oracle User Name and Oracle Password.
4. Click **Apply**. The next time you type the WebDB URL, you will not be prompted for a user name and password. The WebDB Home page displays.

How do I limit access to the WebDB Gateway page?

For security reasons, you may want to limit access to the Oracle WebDB PL/SQL Gateway Settings page to administrators only.

1. With a text editor, open the wdbsrv.app configuration file which is located in Oracle WebDB's listener configuration directory in your environment.
2. In the [WVGATEWAY] section, change the "administrators" setting from all (users) and replace it with a user name(s). Only those specified users will be able to access this page. Separate users with a comma (,). For example,

```
administrators = user1, user2, user3@webdb
```

Only the users who log on as user1, user2, and user3 with the correct password can access the gateway settings page on the webdb host (connect string).

Troubleshooting

Why can’t I see my newly-created WebDB Components when I am adding items?

If you are trying to a add a new WebDB component item in Oracle WebDB Site Builder and you don’t see your newly-created component from the**Item Type** list, you must make sure the component is granted EXECUTE privileges to PUBLIC.

To grant EXECUTE privileges to PUBLIC for your component:

1. In Oracle WebDB, click  to display the User Interface Components page.
2. Click  to locate the appropriate component.
3. On the Find an Existing Components page, complete the fields to narrow down your search.
4. Click Find to search for the component based on your selection criteria.
5. A list of components are displayed. Click the component name link.
6. On the component’s Manage Component page, click the Privileges link.
7. In the **User/Role** field, type Public.
8. Click the Grant Execute Privilege button to grant EXECUTE privileges to Public for the selected component.

You should now be able to see your WebDB component in the **Item Type** list when adding a WebDB component.

Glossary

activity log

A database table that provides a record of end user requests for WebDB components. The log includes information about the time of the request, the user who made the request, and the machine and browser type that originated the request.

application

In WebDB, a group of components connected to one another by links , designed to fulfill a particular business need. For example, a form can be linked a chart that, in turn, allows an end user to drill down to detailed reports about items displayed in the chart.

batch log

Running a component in the background using the WebDB batch job facility. An end user can run a component in batch mode by selecting options on the parameter entry form for the component. Batch processing is useful if the component is based on a large amount of data, or if the component displays many rows of data

bind variable

A variable in a SQL statement that must be replaced with a valid value or address of a value in order for the statement to execute successfully. WebDB component developers typically use bind variables to display a parameter entry field in a component’s parameter entry form. The entry field allows end users to choose the data that the component will display.

bookmark

A method provided by Web browsers for navigating to pages whose locations (URLs) you’ve saved.

Browse In privilege

A WebDB-specific privilege that allows a developer to search schemas for objects such as tables, views, and procedures on which WebDB components will be based. To view an object in WebDB, a developer must have Browse In privileges (or Build In privileges , which automatically includes the Browse In privilege) in the schema that owns it.

browsing

Searching the database for objects. In WebDB, you can browse the database using object names, object types, the schemas owning the objects, or any combination of these search criteria.

Build In privilege

A WebDB-specific privilege that allows a developer to build a component in a schema. Developers must have Build In privileges in at least one schema, including their own, to build a component. The schema will own the finished component.

calendar

A WebDB component that displays the results of a SQL query in calendar format. At least one of the table columns in the query must have the DATE datatype.

chart

A WebDB component that displays the results of a SQL query as a bar chart. Charts are based on at least two table or view columns: one that identifies the bars on the chart and another that calculates the size of the bars on the chart.

check box

A control that can appear alone or in groups on WebDB forms and parameter entry forms. Each check box can be switched either "On" or "Off." WebDB provides options for displaying Lists of Values as check boxes in forms and component parameter entry forms .

cluster

A database object used to store tables that are related to one another and that are often joined together in the same area on disk.

Common Gateway Interface (CGI)

The industry-standard technique for running applications on a Web server. Standard HTML documents retrieved from a Web server are static (the exact same text is retrieved every time). CGI enables a program running on the Web server to communicate with another computer to dynamically generate HTML documents in response to user-entered information.

component

A PL/SQL stored procedure created by a WebDB component build wizard ; for example, a chart, report, or form. Executing the stored procedure creates the HTML code used to display the component.

component schema

Any database schema in which a WebDB developer can build a component.

database access descriptor (DAD)

A set of values that specify how WebDB connects to the Listener or some other type of database server to fulfill an HTTP request. The information in the DAD includes the username (which also specifies the schema and the privileges), the user password, connect string, error log file, standard error message, and NLS parameters.

database administrator (DBA)

A WebDB user having the DBA role. The DBA role provides the user access to all WebDB menus and all privileges, including creating and dropping users, assigning build in and browse in privileges to users, and assigning roles to users.

end user

A WebDB user who executes a component. The user has been granted execute privileges by the owner of the component.

execute privilege

A permission that allows an end user to execute a procedure. In WebDB, the execute privilege is typically granted to allow an end user to execute a component stored in the database as a procedure.

export

To store a copy of an object, module, selected text or image to a file or a remote database.

field-level validation

A method for verifying that correct values have been entered into entry fields in components and parameter entry forms. Field-level validation is performed when the end user causes the OnBlur condition to occur after entering a value in an entry field, for example, when tabbing to another entry field. See also, form-level validation .

foreign key

A value or column(s) in a table that refers to a primary key in another table.

form

A WebDB component that provides an interface to one or more database tables, views, or procedures.

form-level validation

A method for verifying that correct values have been entered into entry fields in components and parameter entry forms. Form-level validation occurs after the user enters a value in an entry field and submits all values on the page, for example, when clicking an **OK** button. See also, field-level validation .

frame driver

A WebDB component consisting of a Web page divided into with two frames. One frame (the driving frame) contains a SQL query that drives the contents of a second target frame.

frames view

A tree displaying a hierarchical view of WebDB menus and pages. End users can click nodes in the tree to navigate to WebDB menus and pages. The frames view is accessible from the WebDB home page.

function

A PL/SQL subprogram that performs a specified sequence of actions, and then returns a value.

generate

To save a procedure containing a WebDB component to a file or database in a binary format so that it can be executed in run-time and batch mode.

grant

A privilege or role given to a WebDB user.

header

An optional region in an HTML-based Web page that contains introductory material for the page. In WebDB, the header can include text, graphics, data, and computations. The header appears first before the body.

hierarchy

A WebDB component that displays data from a self-referencing table or view (at least two columns in the table must share a recursive relationship). A hierarchy can contain up to three levels and displays data such as employees in an organization chart, or the hierarchical relationship between menus in a Web site.

home page

A PL/SQL procedure that, when executed, creates a Web page that is **t**he entry point to the WebDB product. A Listener setting specifies the default home page for WebDB.

HTML

Acronym for Hypertext Markup Language. A tag-based ASCII language used to specify the content, format, and links to other pages on Web servers on the Internet. WebDB consists of a collection of PL/SQL procedures that, when executed, generate HTML.

hypertext link

In WebDB, a reference from some point in a component or Web site to some point in another component or Web site. See also, link .

image

A bitmapped object that can be stored and loaded into a component or Web site, and displayed using a Web browser.

index

An optional structure associated with a table used to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

IP address

A four-part number with no more than three digits in each part that uniquely identifies a computer on the Internet, or on a local LAN.

join condition

Combining data from two (or more) tables or views in a single SQL SELECT statement.

JavaScript

A scripting language developed by Netscape that allows generation of dynamic components in otherwise static HTML. WebDB allows you to use JavaScript to create applications that validate entry fields in components and parameter entry forms .

library

A collection of one or more PL/SQL program units that are stored together in a database, and can be referenced by several applications at once.

link

A shared component that allows developers to add hypertext jumps between components. See also, hypertext link .

List of Values (LOV)

A shared component that allows developers to add selectable values to entry fields in components and parameter entry forms. An end user selects from the list one or more values for the entry field. A single List of Values can be displayed in different formats such as combo boxes, radio buttons, or check boxes.

Why can't I see my newly-created WebDB Components when I am adding items?

Listener

A lightweight Web server that is shipped as part of WebDB that helps build and deploy PL/ SQL- based applications such as WebDB.

lock

A setting automatically applied to a component when it is being edited. The setting prevents other users from editing the component.

master-detail form

A WebDB component that displays a master table row and multiple detail rows within a single HTML page. Values in the master row determine which detail rows are displayed for querying, updating, inserting, and deleting.

menu

A WebDB component that displays a Web page containing options that end users can click to navigate to other menus, WebDB components, or URLs.

mime type

A file format defined by the Multipurpose Internet Mail Extensions standard. A mime type describes the type of file being transferred to the Web browser.

navigation toolbar

1) Buttons located along the bottom of WebDB that allow users to navigate to other WebDB pages.

2) Graphics elements and buttons in a Web site created using WebDB that allow users to navigate to other pages in the site. The buttons and graphic elements can be displayed at the top, sides, or bottom of a page.

null value

The absence of a value in a table column.

object

A structure used to store data in the database. Developers can create objects using object build wizards provided by WebDB, or using Oracle database commands.

Although WebDB components are stored in the database as package objects, the term component is used to refer to charts, reports, forms, etc. *Object* is used when referring to tables, packages, triggers, etc.

object information

Information that WebDB provides about a database object, including its owner, type, and dependencies on other objects.

object schema

A schema that owns the objects on which WebDB components are based. To build a component based on an object, the schema where the component is being built must be granted explicit privileges on the object, such as SELECT, INSERT, UPDATE, DELETE, and EXECUTE.

Oracle Connect String

A Listener setting that can be used to set up a TNS names alias for a remote database installed on Windows NT.

Oracle Home

An environment variable that indicates the root directory of Oracle products.

package

A database object consisting of a specification and a body. The specification includes the datatypes and subprograms that can be referenced by other program units. The body includes the actual implementation of the package.

page request

An end user request for a WebDB component.

parameter

A value that an end user can specify before executing a component. Parameters can determine which data display in the component or how the component displays the data. The end user specifies parameters on a parameter entry form by choosing the Parameters option in on the Manage Component page. These values are passed to the component when it executes.

parameter entry field

A field on a parameter entry form that allows end users to enter values that will be passed to a WebDB component.

parameter entry form

A page that prompts end users for values to pass to a WebDB component. End users can view the parameter entry form for a component, if one has been created, by choosing the **Run with Parameters** option in WebDB.

Why can’t I see my newly-created WebDB Components when I am adding items?

port

A number that TCP uses to route transmitted data to and from a particular program.

primary key

A column in a database table consisting of unique values that can be used to identify rows in a table.

privilege

The right to perform an action on the database. These can either be general (system privileges) or specific to particular database objects (object privileges). They can also be grouped into roles. A user with the DBA role grants privileges in WebDB.

procedure

A PL/SQL subprogram that performs a specified sequence of actions.

profile

The system and Oracle database resources that are available to the user.

Query by Example (QBE) form

A WebDB component that provides an interface allowing end users to query or insert values into a database table or view. The Query by Example form contains entry fields that correspond to the columns in the database table or view on which the form is built.

query

A SQL SELECT statement that specifies which data to retrieve from one or more tables or views in a database.

radio button

A control (similar to a check box) appearing in sets of two or more, only one of which may be either "on" or "off" at any given time. WebDB provides options for displaying Lists of Values as radio buttons in forms or component parameter entry forms.

recursive relationship

A relationship that occurs when the values in a table column can be related to those in another column in the same table or another table; for example, between a primary key and foreign key **.**

remote database

A database running on a separate machine that can be accessed over the network.

report

A WebDB component that displays the results of a SQL query in a tabular format.

role

A group of database object privileges that can be granted and revoked as a unit. The DBA assigns a role to a group of users in order to grant them the database object privileges associated with the role.

row

A set of values in a table; for example, the values representing one employee in the SCOTT.EMP table.

schema

A collection of components and database objects under the control of a given database user. The schema has the same name as the user who owns it.

sequence

A database object used to automatically generate numbers for table rows.

session

The period between logging on and logging off WebDB.

shared components

Building blocks used by WebDB developers to create components. Shared components include links, Lists of Values, JavaScripts, and look and feel elements. Each shared component can be used by multiple developers to create WebDB components.

snapshot

A table that contains the results of a query on one or more tables, called master tables, in a remote database.

snapshot log

A table associated with the master table of a snapshot. It tracks changes to the master table.

Why can’t I see my newly-created WebDB Components when I am adding items?

stored attribute

A component build option setting stored in the database as an argument. The argument is passed to a procedure when it executes to create the component. For example, the **Schema** and **Name** option setting for the component named SCOTT.EXAMPLE_CHART appear as the following stored attributes:

P_OWNER SCOTT

P_COMPONENT_NAME EXAMPLE_CHART

stored results

The results of executing a component in batch mode. The owner of the stored results can make them available to other WebDB users.

structured UI template

A shared component that controls the look and feel of WebDB components. Structured UI templates display the same image and text in the same location on every component that uses the template.

substitution tag

An HTML tag used to create unstructured UI templates . When the HTML code that creates the template executes, substitution tags dynamically embed components, titles, headings, and other elements into the template.

SYLK

The file format used by the Microsoft Excel program to define formulas and data in spreadsheet, as well as transfer spreadsheet content from one file to another.

synonym

A name assigned to a table or view that can thereafter be used to refer to it.

table

The basic storage structure in a relational database.

tablespace

An allocation of space in the database that can contain objects.

temporary tablespace

An allocation of space in the database used for the creation of temporary table segments for operations such as sorting table rows.

trigger

A stored procedure associated with a table. It executes before or after one or more specified events.

unstructured UI template

A shared component that controls the look and feel of WebDB components. Unstructured UI templates are based on HTML code that, when executed, dynamically embeds components, titles, headings, and other elements.

user name

In WebDB, identical to a schema name. A user who logs into WebDB with the user name Scott can by default create components in the SCOTT schema.

user interface (UI) template

A shared component that controls the look and feel of WebDB components. Selecting a UI template when building a component automatically selects a title on the page where the component is displayed, a title background, links to other Web pages, and background colors and images.

version

Indicates the status of a stored procedure that contains a WebDB component. For example, ARCHIVE indicates an old version of the component that is being saved in the database. PRODUCTION with VALID PACKAGE indicates the most recent version of the component, which will run without errors. PRODUCTION with INVALID PACKAGE indicates that the most recent version of the component contains errors. There can be multiple versions of the same component.

view

A virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

Web server

A program that delivers Web pages.

WebDB role

Two special types of roles that control the user’s view of the WebDB product. The DBA role provides the user access to all WebDB menus and all privileges. The

Why can’t I see my newly-created WebDB Components when I am adding items?

WEBDB_DEVELOPER role provides the user access to all WebDB menus except System Administration and Monitoring. Users with this role can build components in the their own schema, plus any schemas in which the DBA has granted build and browse privileges.

wildcard

In WebDB, the percent (%) character, which is used to mean any single character or a contiguous set of characters within a word or phase.

wizard

A graphical interface that guides a user step by step through a process. In WebDB, there are wizards for creating of creating components, objects, Web sites, and items within Web sites.

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W