FrontBase®

Building a Database with Qilan®

Because of last-minute changes to FrontBase, some
of the information in this manual may be inaccurate.
Please read the Release Notes on the FrontBase
distribution for the latest up-to-date information.

Revised: 02/17/05 gab

How to Contact FrontBase:

U.S.A. and international	FrontBase Inc. 27042 Towne Center Drive Suite 265 Foothill Ranch, CA 92610 FrontBase Inc. Datavej 52 DK-3460 Birkerød Denmark
Ordering	Voice: +1 949 455 2913 Fax: +1 949 457 0966 Voice: +45 4582 6262 Fax: +45 4582 0816
World Wide Web	`http://www.frontbase.com`
Technical support	`support@frontbase.com`
Information	`info@frontbase.com`
Sales,marketing	`sales@frontbase.com`
Licensing	`license@frontbase.com`

Table of Contents

A

Building a Database with Qilan®

Qilan is a tool for manipulating data on a web server. It handles database connections, business logic and user interface screens.

Overview

Qilan sits between the web server, which handles client requests, and the database. It functions as a common gateway interface (cgi). When the web server receives a request, the web server launches a special Qilan document which in turn is executed by qilan.cgi. These specialized documents, also known as 'webtemplates', are created by the Qilan application (developer).

As we proceed through this tutorial, we will be working with the Qilan developer.

The topics in this section are:

- "The Database" on page 5.
- "The Schema" on page 11.
- "The User Interface" on page 19.

The Database

1. The first step is to create an empty database. Qilan works with an external database after it has been created and named. From the Viewer, locate FBManager.app and launch it.

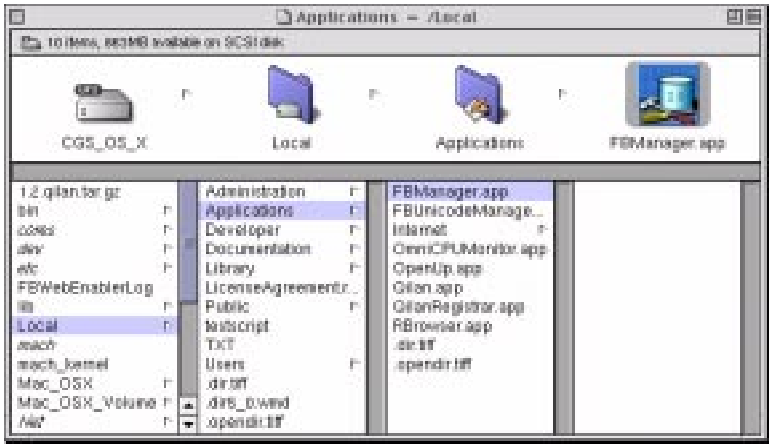
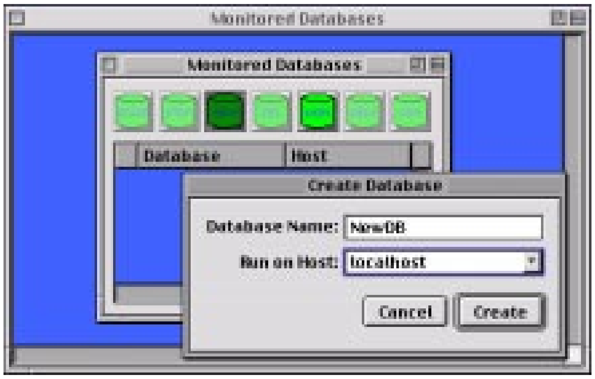


Figure A.1 The FBManager.app

Database Creation

2. Select ‘New’ from the Monitored Databases window. You can name the database almost anything you want; just remember the name is case sensitive. If you are running the database on the local machine, name the host, “localhost”. You can also use the machine’s IP number. Click, CREATE.

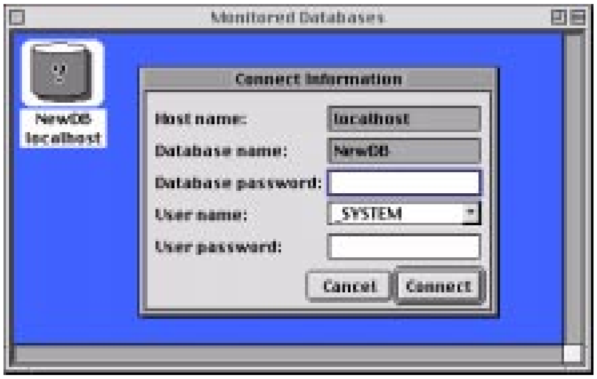


Database Name: NewDB
Run on Host: localhost

Figure A.2 Creating the Database

Starting the Database

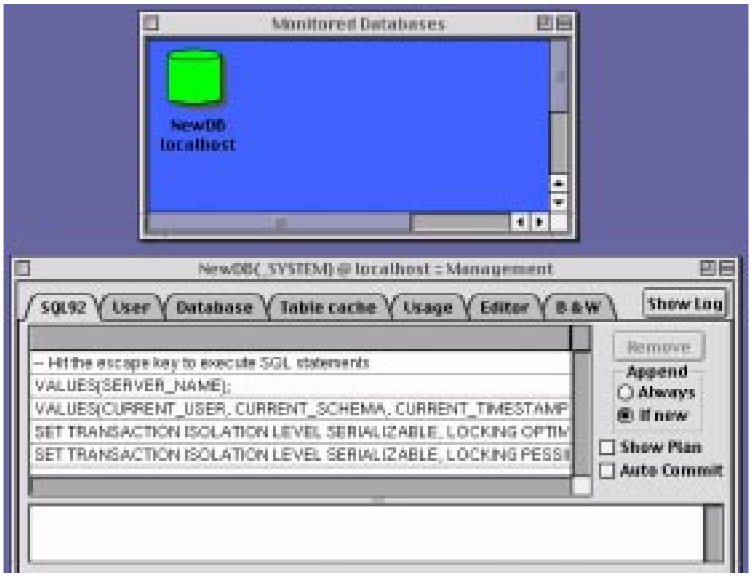
3. The database is now created, but not yet started. Double click on the NewDB cylinder () to start the database and connect to it. The Connect Information window will open. Note that a default user name has been entered. The name, “_SYTEM” is equivalent to the UNIX logon as ‘root’. In other words, it should be changed at some point to protect the database from unintended alterations. For the purposes of this tutorial, just leave it alone. Click CONNECT to connect and start the database.



Host name: localhost
Database name: NewDB
Database password:
User name: _SYSTEM
User password:

Figure A.3 Connecting to the Database

4. The database cylinder has turned green. This is your indication that it is running. The management window will open, but for now, just ignore it. Let's proceed to the Qilan developer.



```
-- Hit the escape key to execute SQL statements
VALUES(SERVER_NAME);
VALUES(CURRENT_USER, CURRENT_SCHEMA, CURRENT_TIMESTAMP
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, LOCKING OPTIM
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, LOCKING PESSI
```

Append
○ Always
● If new

□ Show Plan
□ Auto Commit

Figure A.4 Database is Running

Starting Qilan

5. Returning to the Viewer, double click on Qilan.app. This will start the Qilan developer.

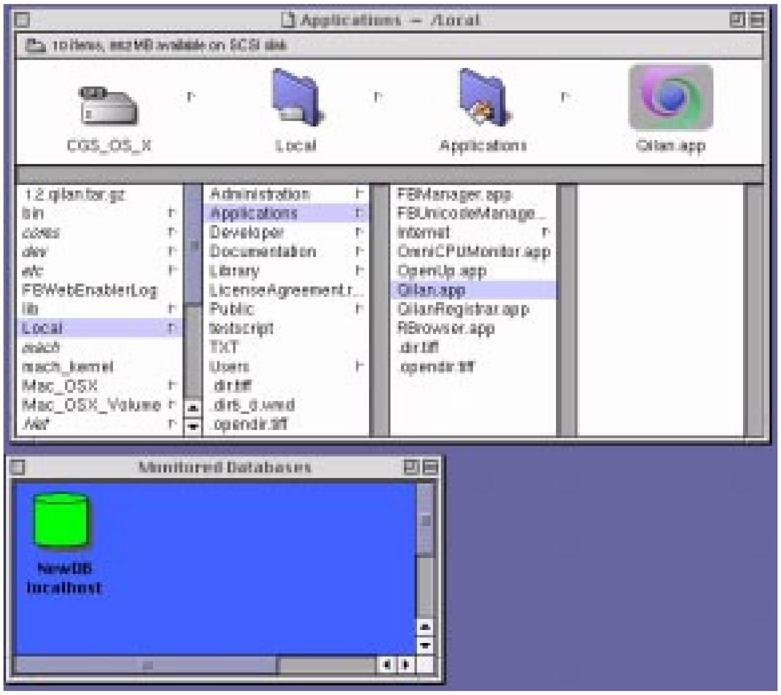


Figure A.5 Starting the Qilan Developer Application

6. Upon opening, Qilan will present an untitled window and a palette. From the palette, drag an Access icon into the untitled window. The Access icon reflects the database you are connecting to - in this case, FrontBase. Double click on the small access icon to reveal its contents.

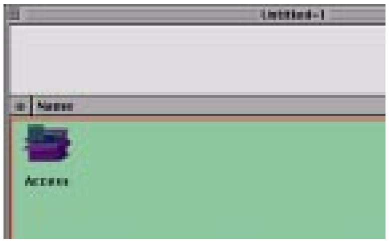
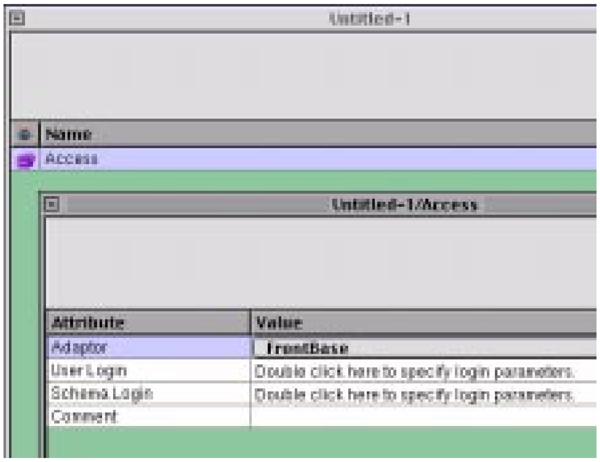


Figure A.6 Connect to FrontBase

7. The first aspect of the ‘accessing’ of a database is to login. Qilan provides two mechanisms to access a database: as a

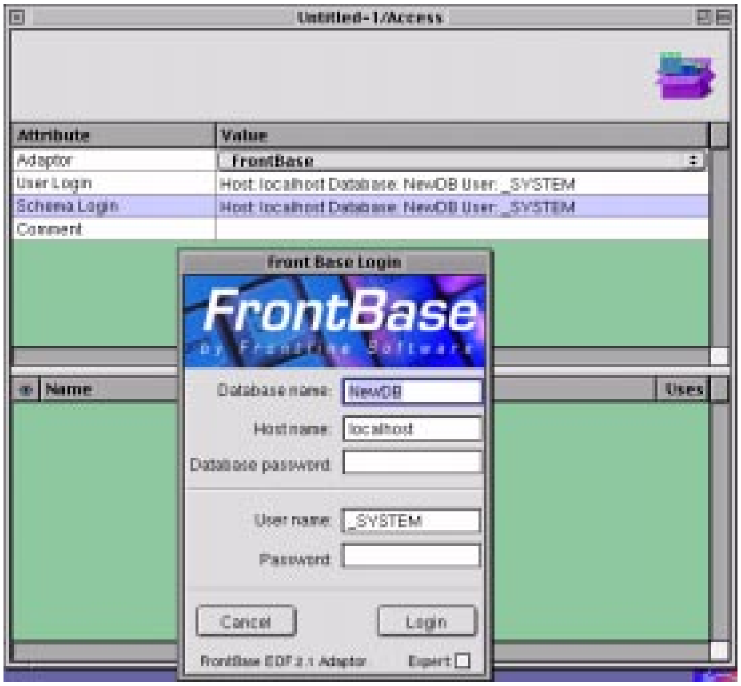
user and as a designer. A user has only data access. A designer has schema access. As we accepted FrontBase's default user name, "_SYSTEM", the same name / password will be used for user and schema login. Double click on the User Login parameter line to open the login panel.



Attribute	Value
Adaptor	FrontBase
User Login	Double click here to specify login parameters.
Schema Login	Double click here to specify login parameters.
Comment	

Figure A.7 Open the Login Panel

8. Enter the database name, host name and user name as required by FrontBase. Again note the database name is case sensitive, but user names are not. Leave password blank



Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database: NewDB User: _SYSTEM
Schema Login	Host: localhost Database: NewDB User: _SYSTEM
Comment	

Database name: NewDB
Hostname: localhost
Database password:
User name: _SYSTEM
Password:

Figure A.8 Enter Login Information

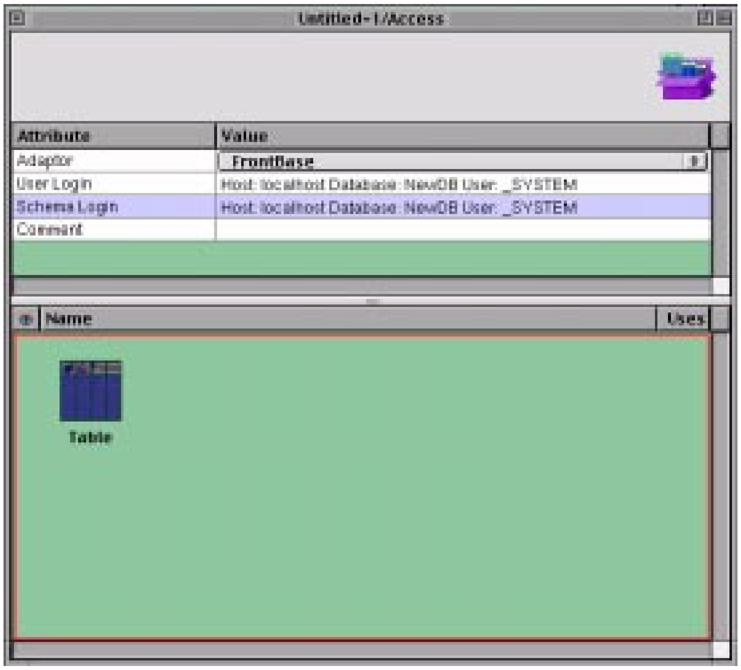
9. Repeat this for the Schema login parameter line.

The Schema

1. The next step is create our database schema (tables and fields). These objects are represented in Qilan, but created in our FrontBase database.

Schema Creation

2. This tutorial will create fields, but Qilan can also create relationships within or between tables. From the palette, drag the table icon into the untitled / access window.



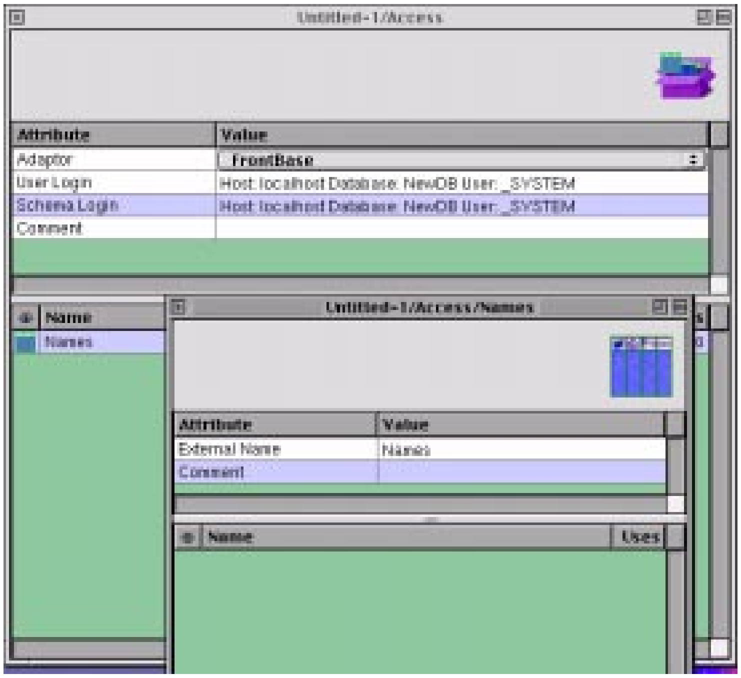
Untitled-1/Access

Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database: NewDB User: _SYSTEM
Schema Login	Host: localhost Database: NewDB User: _SYSTEM
Comment	

Figure A.9 Begin Schema Creation

3. Rename the table icon to ‘Names’, then double click on the icon to reveal it’s contents. The window, Untitled-1 / Access / Names will open.

External name is how you want the database to store this table. It may be the same as your Qilan name, but some organizations have existing naming conventions which you may be required to use. In any case, let’s name it, “Names”.



Untitled-1/Access

Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database: NewDB User: _SYSTEM
Schema Login	Host: localhost Database: NewDB User: _SYSTEM
Comment	

Name
Names

Untitled-1/Access/Names

Attribute	Value
External Name	Names
Comment	

Name	Uses

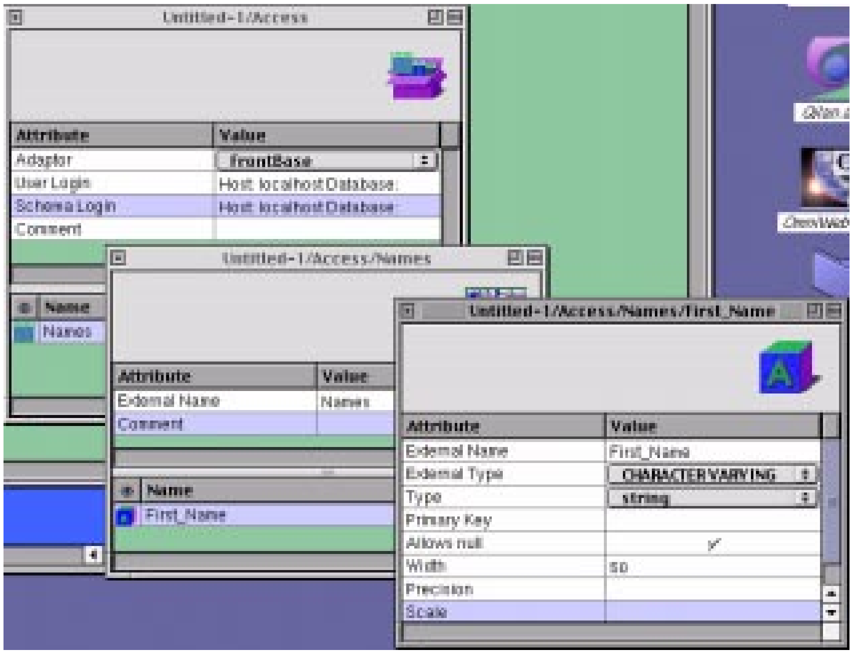
Figure A.10 Rename Table to Names

Tables usually have fields, which are the essential data storage mechanisms. From the palette, drag a field icon into the Untitled-1 / Access / Names window. Rename the default name to: “First_Name”.

4. As with the table, double click on the small icon to open the field window. Type an external name, which is how the database will name this object.

The default field type for the database is ‘character’. Qilan is very flexible about data types, but offers basic interpretations, such as string, date, float, etc. As data logic is handled in Qilan, you can default data types instead of data typing for each calculation or manipulation. As you will see later on, we can override these default settings as necessary.

Additional settings, such as, ‘Allows Null’, can also be set in Qilan. Make settings to match the Untitled-1 / Access / Names / First_Name window below.



Untitled-1/Access

Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database:
Schema Login	Host: localhost Database:
Comment	

Name
Names

Untitled-1/Access/Names

Attribute	Value
External Name	Names
Comment	

Name
First_Name

Untitled-1/Access/Names/First_Name

Attribute	Value
External Name	First_Name
External Type	CHARACTER VARYING
Type	string
Primary Key	
Allows null	✓
Width	50
Precision	
Scale	

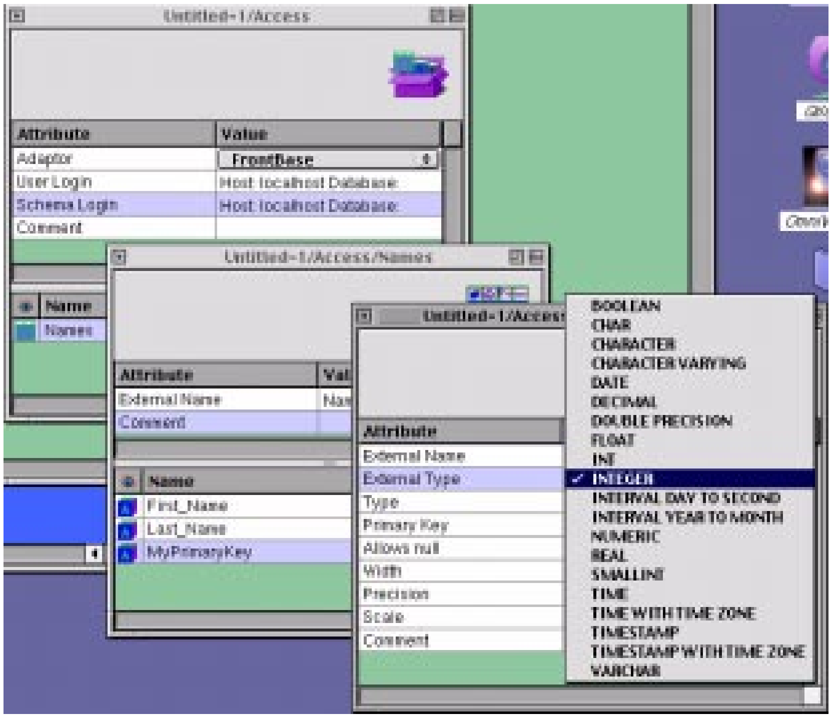
Figure A.11 Maker Settings for First_Name

5. We need to create two additional fields, “Last_Name” and “MyPrimaryKey”. Last_Name is the same as First_Name, a character field. You can duplicate First_Name, but insure you change the external name. No two external names can be the same within any table.

   Although you can create a table without a primary key, this practice is not advised. So let’s go ahead and create a primary key field.

6. With the window Untitled-1 / Access / Names / MyPrimaryKey open, type an external name, then choose the external type, ‘Integer’ from the external type popup. Also choose int from the type popup. Double click on the Primary Key so that a check mark appears.

   You might wonder why we are using a integer, as primary keys can be almost field type. The reason is that we will be using FrontBase’s ability to automatically create primary key values. Alternatively, the designer would be responsible for this function.



Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database:
Schema Login	Host: localhost Database:
Comment	

Attribute	Val
External Name	Nar
Comment	

Name
First_Name
Last_Name
MyPrimaryKey

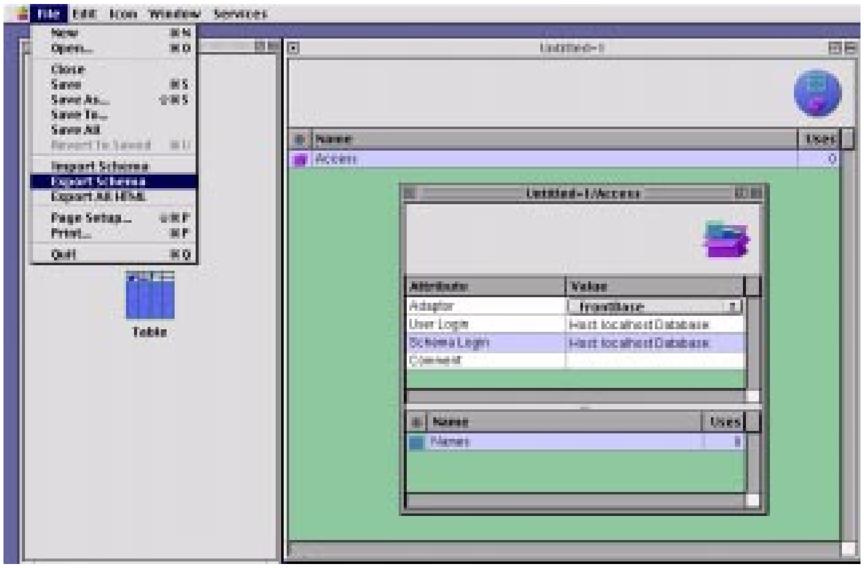
Attribute
External Name
External Type
Type
Primary Key
Allows null
Width
Precision
Scale
Comment

Figure A.12 Setting External Type

7. Close field and table windows so just the Untitled-1 / Access window is open.

Exporting to FrontBase

1. Now we export our schema to FrontBase. This action writes our access objects (tables and fields) to the database, “NewDB”. From the file menu, choose Export Schema.



Name	Uses
Access	0

Attribute	Value
Adaptor	FrontBase
User Login	Host localhost Database
Schema Login	Host localhost Database
Comment	

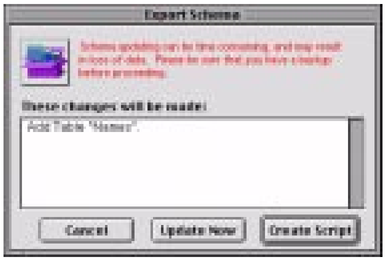
Name	Uses
Names	8

Figure A.13 Exporting the Schema

2. The Export Schema window will open informing you of the impending action. You can update immediately or defer your export by creating a file. Export files may be viewed, edited or stored or applied at a later date.

   Click the UPDATE NOW button.

3. Qilan automatically imports database schema to Qilan immediately after the export. This is performed in the event the database creates objects as the result of the Qilan export and helps to insure synchronization.



Export Schema

Schema updating can be time consuming, and may result in loss of data. Please be sure that you have a backup before proceeding.

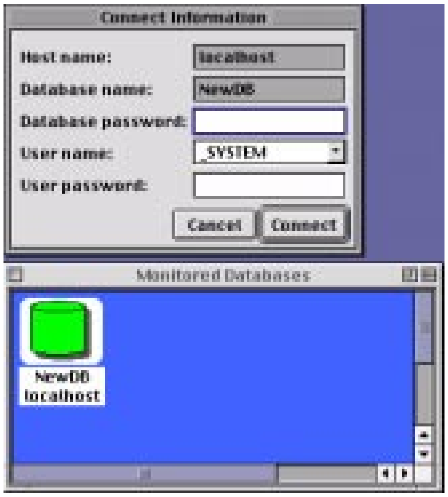
These changes will be made:

Add Table "Names".

Figure A.14 Synchronizing the Schema

Creation of the Primary Key

1. To invoke FrontBase's ability to automatically create primary key values, we will need to return to FrontBase. If FBManager is not running, launch it.
2. Double click on the NewDB icon in the Monitored Databases window. The Connect Information window will open; click CONNECT.



Connect Information

Host name: localhost

Database name: NewDB

Database password:

User name: _SYSTEM

User password:

Figure A.15 Connecting to the Database

3. After connecting, the Management window will open. Highlight '_SYSTEM' in the first column; then 'Tables'; followed by, 'Names'.

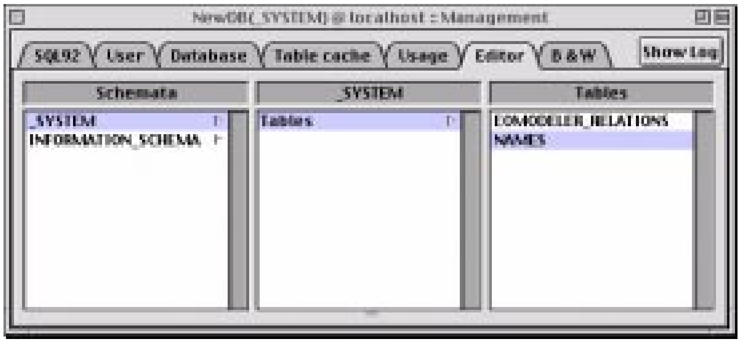


Figure A.16 Choosing the Names Table

4. With ‘Names’ highlighted, select Definition Editor from the Management menu.

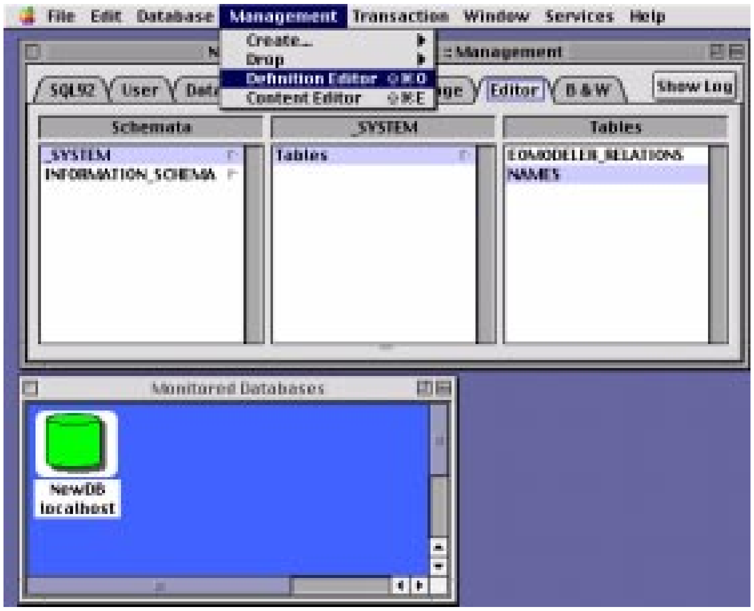


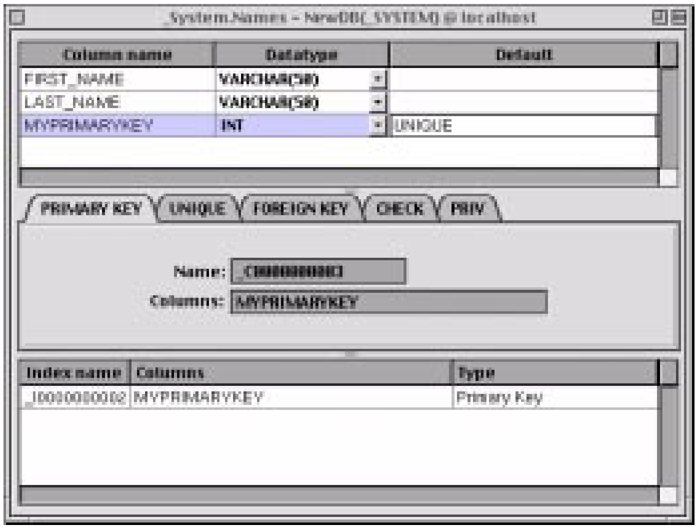
Figure A.17 Selecting the Definition Editor

5. Now we can modify our table fields. Locate the field named, ‘MyPrimaryKey’. Note how it is an integer, just as we created it in Qilan. Type in the word, “UNIQUE” in the default

column. This keyword invokes FrontBase’s automatic key value generation.

Commit Changes

6. Now close your windows. Click COMMIT when FrontBase asks if you want to commit your changes.



_System.Names – NewDB(_SYSTEM) @ localhost

Column name	Datatype	Default
FIRST_NAME	VARCHAR(50)	
LAST_NAME	VARCHAR(50)	
MYPRIMARYKEY	INT	UNIQUE

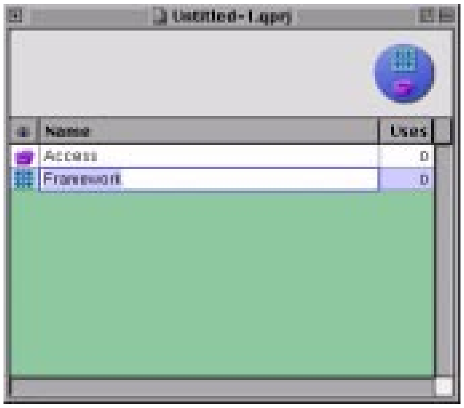
Name: _C0000000003
Columns: MYPRIMARYKEY

Index name	Columns	Type
_I0000000002	MYPRIMARYKEY	Primary Key

Figure A.18 Commit the Table Changes

The User Interface

Returning to Qilan’s Untitled-1 window, drag out a Framework icon from the palette. The Framework is used to build your user interface forms and create most of your database logic.



Untitled-1.qprj

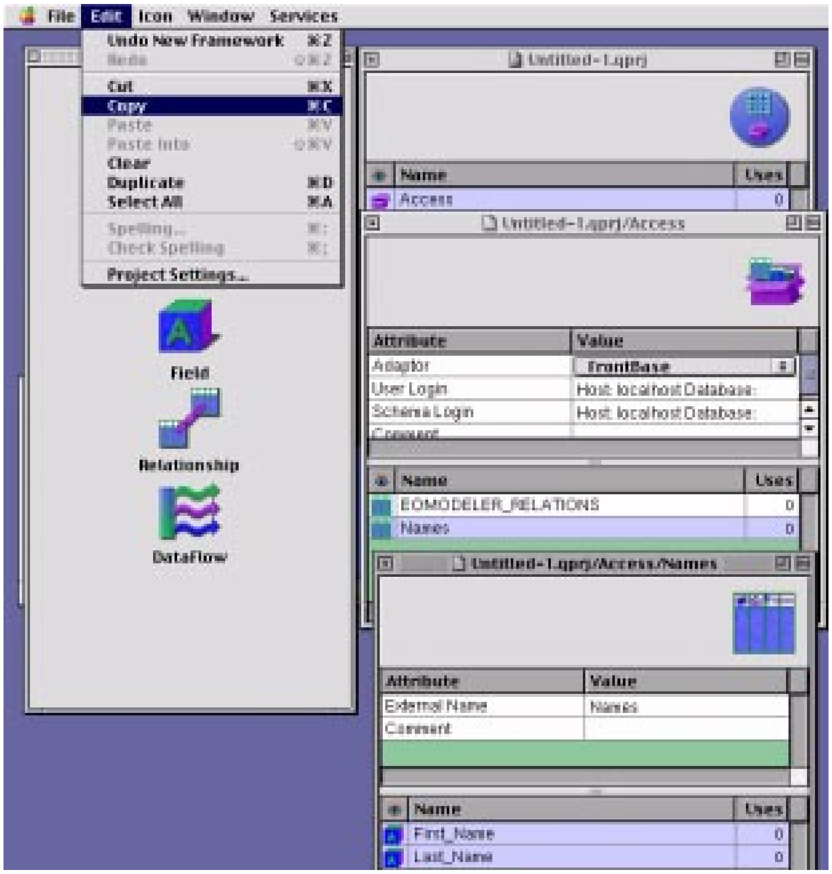
Name	Uses
Access	0
Framework	0

Figure A.19 Build UI through the Framework

Building the Framework

Although we can create fields in our Framework, just like we did in Untitled-1 / Access / Table, it might be easier if we could copy, then paste them directly into the Framework.

1. To accomplish this, open the Untitled-1 / Access / Names window and highlight the fields. Choose Copy from the Edit menu.



Untitled-1.qprj

Name	Uses
Access	0

Untitled-1.qprj/Access

Attribute	Value
Adaptor	FrontBase
User Login	Host: localhost Database:
Schema Login	Host: localhost Database:
Comment	

Name	Uses
EOMODELER_RELATIONS	0
Names	0

Untitled-1.qprj/Access/Names

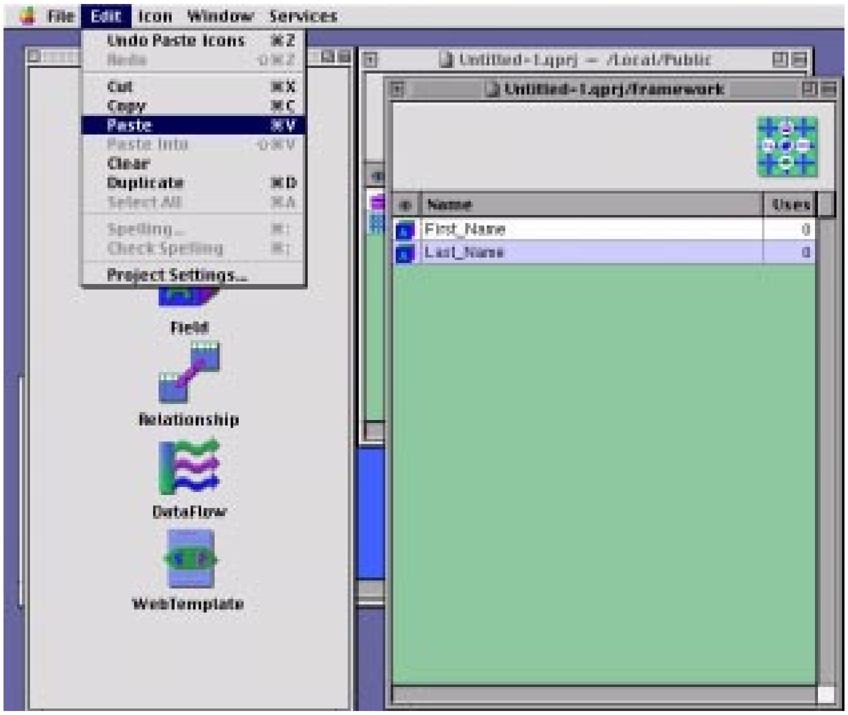
Attribute	Value
External Name	Names
Comment	

Name	Uses
First_Name	0
Last_Name	0

Figure A.20 Copy and Paste the Fields

2. Now open the Framework by double clicking on its icon in the untitled window. Select Paste from the Edit menu.

NOTE: If you double click on a field within a Framework, only the comment line appears. This is because Framework fields are used to hold data during qilan.cgi execution and do not actually represent database storage fields.



Name	Uses
First_Name	0
Last_Name	0

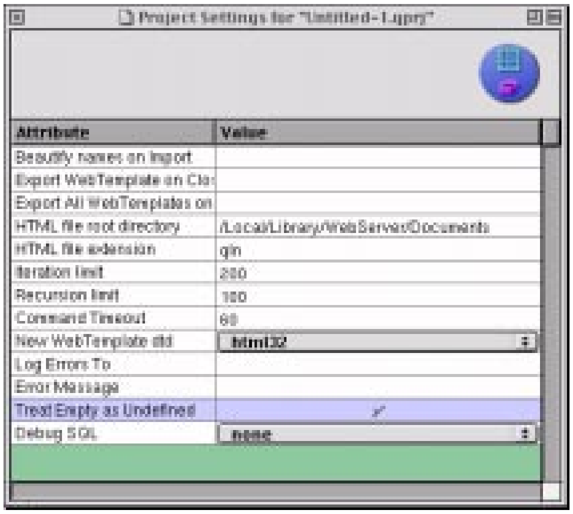
Figure A.21 **Pasting the Fields**

Modify Project Settings

Before we proceed, we need to set a few project settings. These settings control how Qilan will handle data, where webtemplates are exported and what type of dtd (document data definitions) will be used.

1. Select Project Settings... from the Edit menu. The default settings appear acceptable, but we want to treat empty values as undefined. Double click this line until a check appears.

   Why do we want to do this For most database designers, a field value is either defined or undefined. This is in contrast to a submitted html input value, which may be defined, empty or undefined. For our purposes, let's use the database designer's perspective.
2. Closing this window saves the settings.



Project Settings for "Untitled-1.qpry"

Attribute	Value
Beautify names on Import	
Export WebTemplate on Clo	
Export All WebTemplates on	
HTML file root directory	/Local/Library/WebServer/Documents
HTML file extension	qln
Iteration limit	200
Recursion limit	100
Command Timeout	60
New WebTemplate dtd	html32
Log Errors To	
Error Message	
Treat Empty as Undefined	✓
Debug SQL	none

Figure A.22 Save the Settings

Returning to the Untitled-1/Framework window, our first step is to create a user interface form.

Template Construction

1. From the palette, drag a webtemplate into the framework window. Rename the default name to, "Names", then double click to open the window. As specified in Project Settings, the webtemplate uses a html 3.2 dtd. Qilan will conform the webtemplate construction to the chosen dtd,

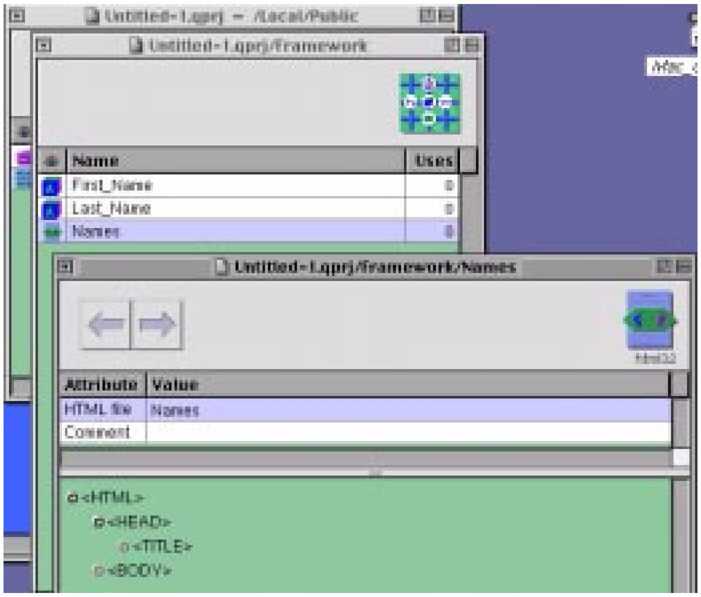
including available html tags and their placement. In other words, Qilan prevents syntax errors.

To use a different dtd, just change the dtd in Project Settings before dragging out a new webtemplate.

As you construct your webtemplate, observe that Qilan does not require closing tags and uses an outline structure.

The "HTML file" is how this webtemplate will be referred to by client users. Type, "Names". The URL will be something like this:

```
[hostname]/cgi-bin/qilan.cgi/[HTML file]
localhost/cgi-bin/qilan.cgi/Names
```


Untitled-1.qprj – /Local/Public
Untitled-1.qprj/Framework
Untitled-1.qprj/framework/Names
html32

Name	Uses
First_Name	0
Last_Name	0
Names	0

Attribute	Value
HTML file	Names
Comment	

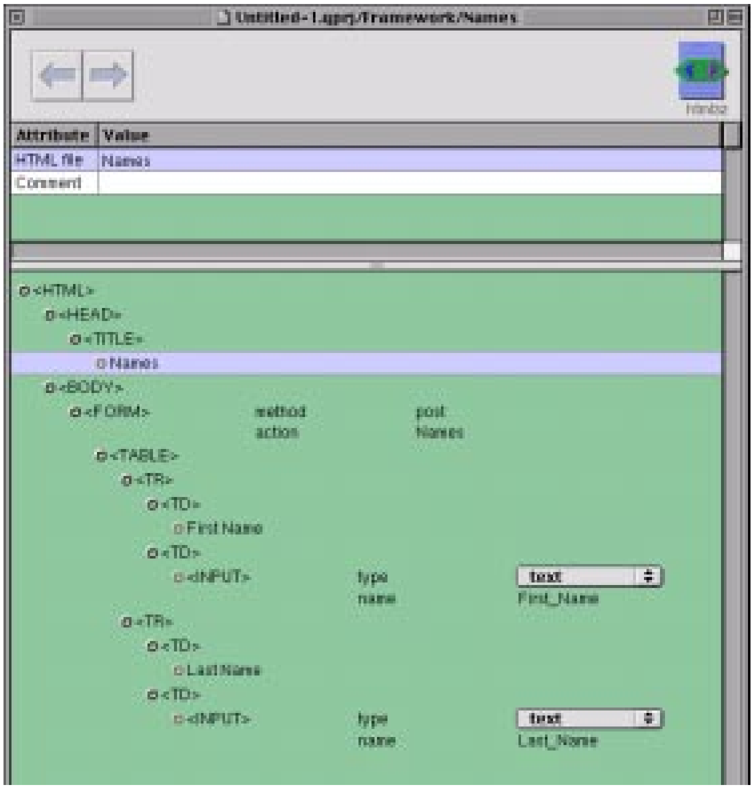
```
<HTML>
  <HEAD>
    <TITLE>
  <BODY>
```

Figure A.23 Construct the Template

2. Using the html tags from the palette, select tags that permit the submittal of data, such as 'form' and 'input'. In this tutorial, we have chosen to use a table structure.

Modify Tag Attributes

1. Most tags allow for attributes. Highlight a specific tag, then select the attributes submenu. Double click directly on the webtemplate (adjacent to the attribute) to enter a specific value. As you will see later on, you can also insert calculated or database values to create dynamic attributes.



Attribute	Value
HTML file	Names
Comment	

```
<HTML>
  <HEAD>
    <TITLE>
      Names
  <BODY>
    <FORM>          method      post
                    action      Names
      <TABLE>
        <TR>
          <TD>
            First Name
          <TD>
            <INPUT>       type        text
                          name        First_Name
        <TR>
          <TD>
            Last Name
          <TD>
            <INPUT>       type        text
                          name        Last_Name
```

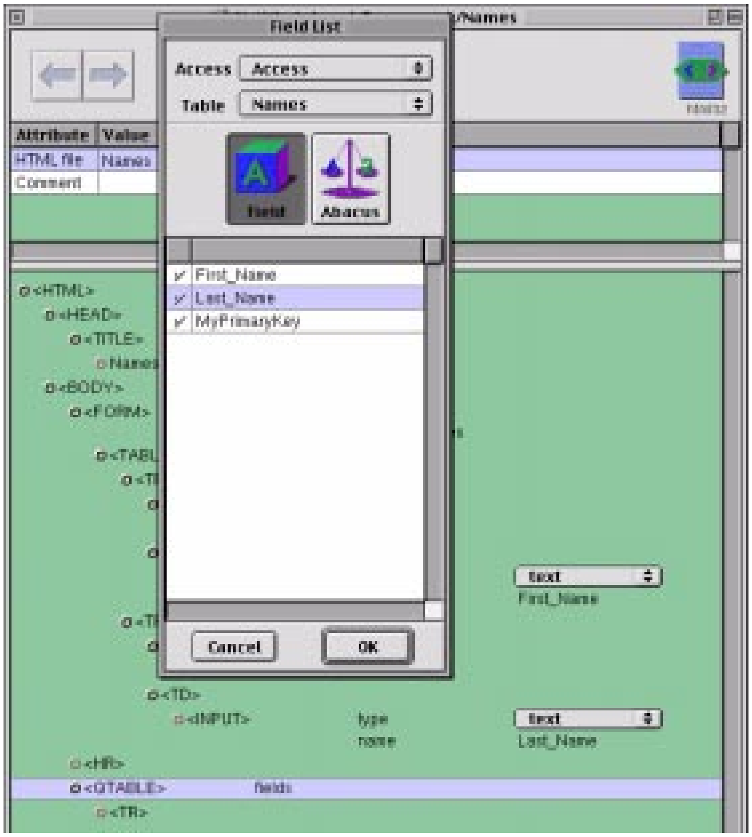
Figure A.24 Modify Tag Attributes

2. Now for some creative elements. Not wanting to have more than one window, let's show the records submitted (and created in the database) as we create them. This combines submittal and display elements on the same form.

3. After the html FORM tag in the webtemplate, drag to insert a QTABLE tag from the palette. 'Q' tags are special Qilan processing elements that extend functionality or reach directly into the database. The QTABLE tag combines the formatting

of a traditional html table tag with an interactive retrieval function.

4. Once the QTABLE is dragged into the webtemplate, the attribute, ‘fields’ will appear. Double click on this attribute; the field list window will open.
5. The field list window asks, ‘What Access’, ‘What Table’, What Fields’. Choose, ‘Access’, ‘Names’ from the pulldowns at the top, and then select all the fields by clicking in the column to the left of each field name. Click the OK button.



Field List

Access: Access

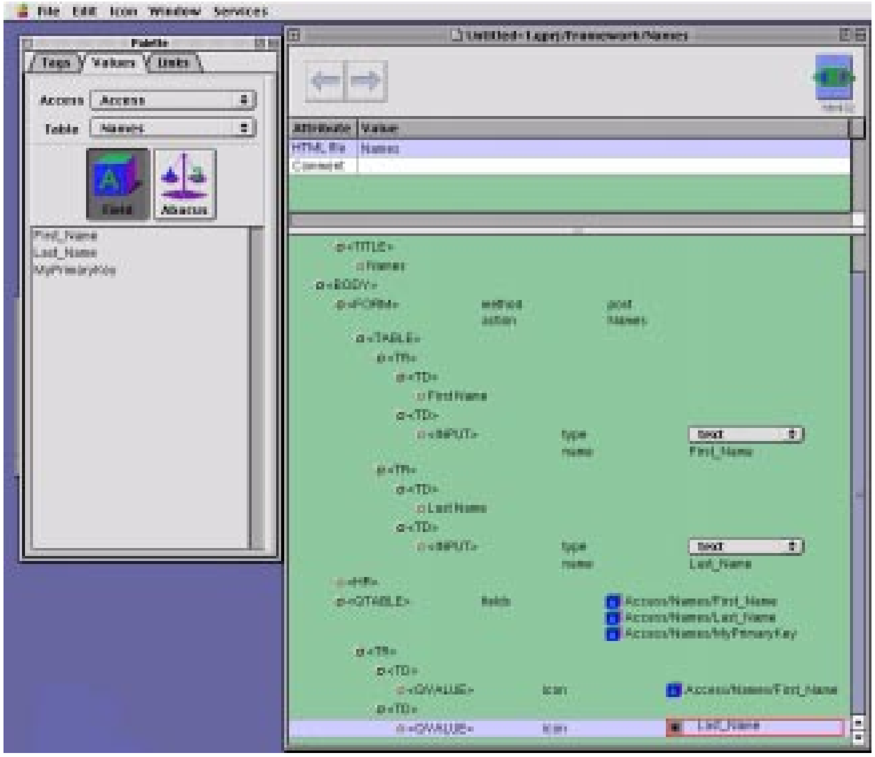
Table: Names

✓ First_Name
✓ Last_Name
✓ MyPrimaryKey

Figure A.25 Choosing the Field List

6. The selected fields will then appear within the QTABLE. Although we are not using a query for this tutorial, queries of considerable sophistication can be constructed without the need for SQL syntax.

7. As we build our QTABLE, we insert traditional html table tags, such as table rows (TR) and table data (TD). From a formatting standpoint, we want to display certain database field elements. This is performed by using a QVALUE tag; meaning, “Display this value here”.
8. From the palette, click on the Values tab, then choose the Access and Table from which the field values will be selected. Drag the fields to the webtemplate as QVALUE attributes.



Access: Access
Table: Names

First_Name
Last_Name
MyPrimaryKey

Attribute	Value
HTML file	Names
Comment	

```
  <TITLE>
    Names
<BODY>
  <FORM>          method        post
                  action        Names
    <TABLE>
      <TR>
        <TD>
          First Name
        <TD>
          <INPUT>        type          text
                         name          First_Name
      <TR>
        <TD>
          Last Name
        <TD>
          <INPUT>        type          text
                         name          Last_Name
  <HR>
  <QTABLE>        fields        Access/Names/First_Name
                                Access/Names/Last_Name
                                Access/Names/MyPrimaryKey
    <TR>
      <TD>
        <QVALUE>       icon          Access/Names/First_Name
      <TD>
        <QVALUE>       icon          Last_Name
```

Figure A.26 Selecting Field Values

The basic construction and formatting are finished. We have created an entry form and a mechanism to retrieve and display the data. What remains is the logic for moving the data to the database (upon submittal) and some method to retrieve selected data for editing.

Processing Q Tags

A Qilan webtemplate processes ‘Q’ tags in a hierarchal order, meaning top to bottom. This allows the designer to create logical constructs. In the following screen, we have added several ‘Q’ tags to create data when certain conditions are met. Let’s cover them one by one.

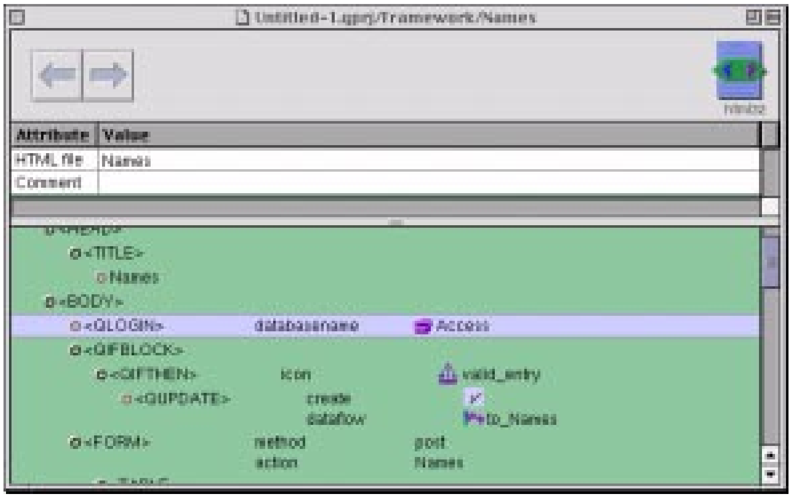
The QLOGIN tag

The QLOGIN tag instructs Qilan to log into FrontBase. The login is handled by the settings in our Access. Qilan can log into as many databases as necessary, there is no limit.

The next ‘Q’ tag is an ‘if’ statement. When the ‘valid_entry’ abacus returns “yes”, we proceed to the QUPDATE, otherwise this is skipped. The QUPDATE performs an ‘insert’ if a link does not exist, otherwise an ‘update’ is performed.

When is this logic performed Each time the client accesses the html file, “Names”.

NOTE: The FORM action is, “Names”. This means that each time the submit button is pressed the html file will be reloaded and the Qilan tags will be processed.



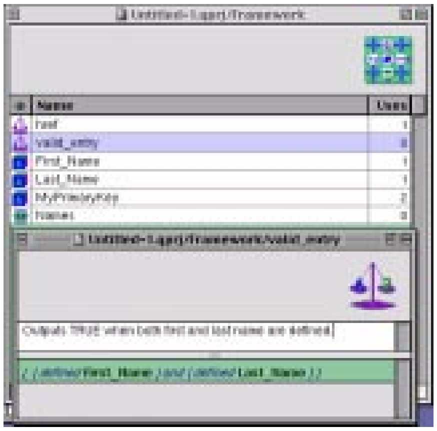
Untitled-1.qprj/Framework/Names

Attribute	Value
HTML file	Names
Comment	

```
    <TITLE>
      Names
  <BODY>
    <QLOGIN>          databasename        Access
    <QIFBLOCK>
      <QIFTHEN>         icon                valid_entry
        <QUPDATE>         create              ✓
                          dataflow            to_Names
    <FORM>            method              post
                      action              Names
```

Figure A.27 Modify Access Settings

The validation is simple, but prevents an empty database record. Valid_entry tests for the presence of a First_Name and Last_Name. An abacus is employed, with ‘defined’ operators selected from the palette.



Untitled-1.qprj/Framework

Name	Uses
test	1
valid_entry	0
First_Name	1
Last_Name	1
MyPrimaryKey	2
Names	0

Untitled-1.qprj/Framework/valid_entry

Outputs TRUE when both first and last name are defined.

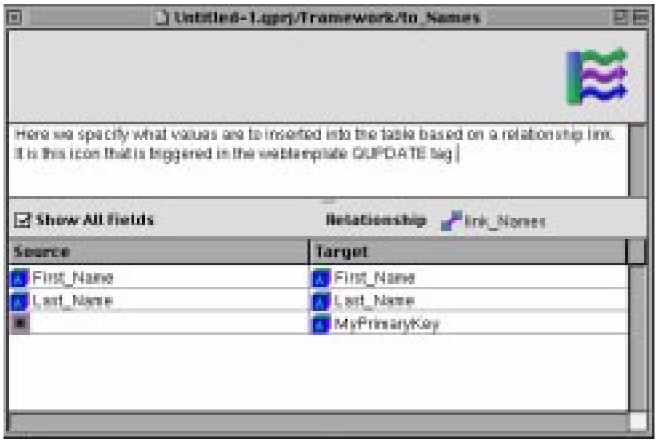
```
( ( defined First_Name ) and ( defined Last_Name ) )
```

Figure A.28 Validation Tests

The QUPDATE

The QUPDATE uses a dataflow object. This object instructs Qilan to insert or update target values (those existing in the database) from source values (those submitted by the user). Dataflows are always based on a link, or relationship.

NOTE: We are leaving 'MyPrimaryKey' undefined. This is because we have requested FrontBase to automatically value this field.



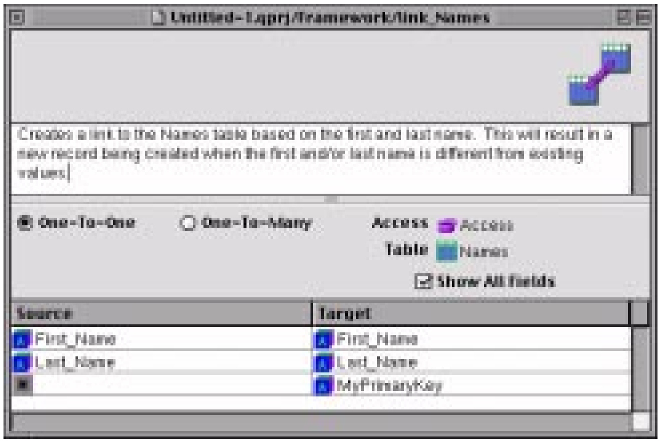
Here we specify what values are to inserted into the table based on a relationship link.
It is this icon that is triggered in the webtemplate QUPDATE tag

☑ Show All Fields   Relationship link_Names

Source	Target
First_Name	First_Name
Last_Name	Last_Name
	MyPrimaryKey

Figure A.29 The QUPDATE Validation

The relationship “link_names” used in the middle of the above dataflow defines matching or unique values to be used as linkage. In this relationship window below, we are saying that when ((First_Name = ‘First_Name) and (Last_Name = Last_Name)) we have a state of uniqueness. Dataflows (triggered by QUPDATE) will create a new database record when the relationship does not exist.



Untitled-1.qprj/framework/link_Names

Creates a link to the Names table based on the first and last name. This will result in a new record being created when the first and/or last name is different from existing values.

● One-To-One ○ One-To-Many

Access: Access
Table: Names

☑ Show All Fields

Source	Target
First_Name	First_Name
Last_Name	Last_Name
	MyPrimaryKey

Figure A.30 **Defining Relationship Validation scenarios**

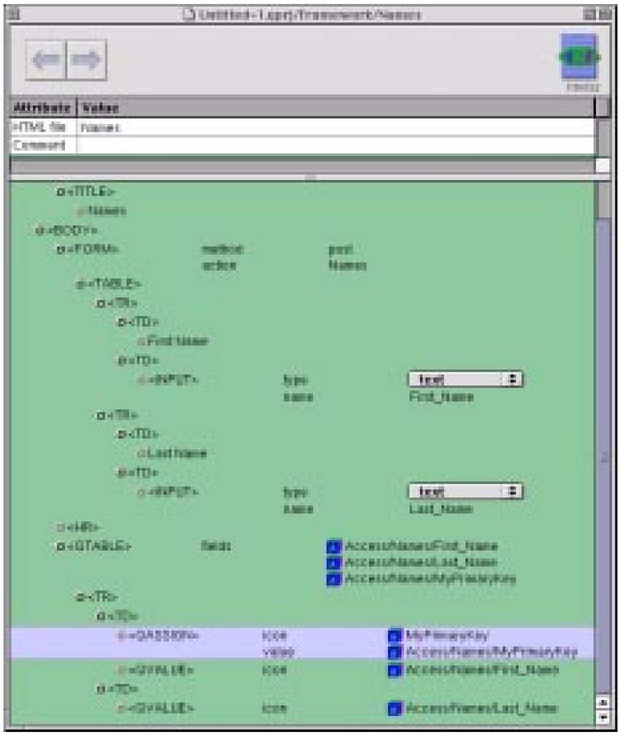
Accessing Existing Data

The next part of the tutorial covers accessing existing data for editing.

Before we start, consider we have already created a QTABLE to retrieve our data. Rather than creating new structures, let’s just use what we have. A web-centric approach typically involves the use of links. The user clicks the link and the record should automatically be retrieved for editing.

To perform this type of action, we will need to obtain a unique link for each record created. Let’s use MyPrimaryKey.

1. First, we double click on the QTABLE fields attribute in the webtemplate and select MyPrimaryKey. This retrieves the value.
2. Second, we create a new framework field, MyPrimaryKey. Other framework objects will use the value of this icon.
3. Third, we value this new framework icon using the value from the database table. Note how QASSIGN is used to accomplish this within the QTABLE. For each record retrieved by QTABLE, the framework icon will be valued by MyPrimaryKey. This will give each record a unique value.



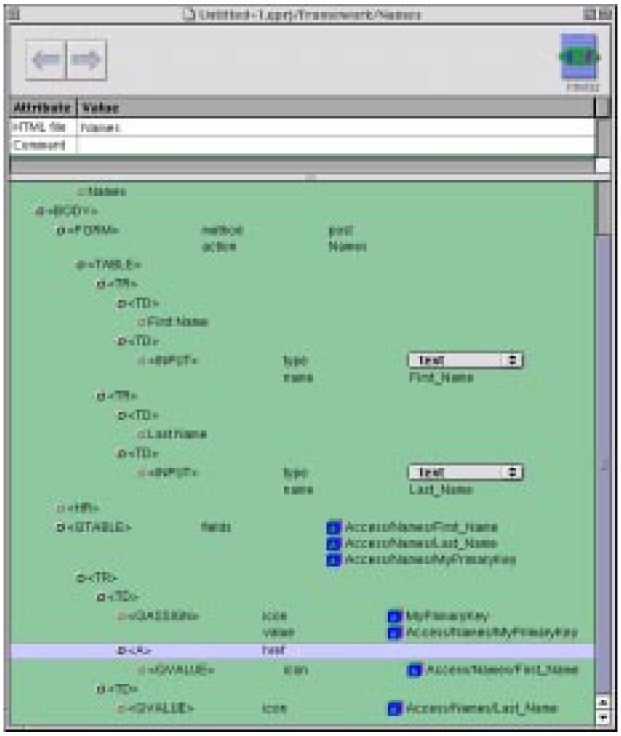
Untitled-1.pprj/Framework/Names

Attribute	Value
HTML file	Names
Comment	

```
    <TITLE>
      Names
  <BODY>
    <FORM>              method        post
                        action        Names
      <TABLE>
        <TR>
          <TD>
            First Name
          <TD>
            <INPUT>          type            text
                             name            First_Name
        <TR>
          <TD>
            Last Name
          <TD>
            <INPUT>          type            text
                             name            Last_Name
    <HR>
    <QTABLE>            fields        Access/Names/First_Name
                                      Access/Names/Last_Name
                                      Access/Names/MyPrimaryKey
      <TR>
        <TD>
          <QASSIGN>        icon            MyPrimaryKey
                           value           Access/Names/MyPrimaryKey
          <QVALUE>         icon            Access/Names/First_Name
        <TD>
          <QVALUE>         icon            Access/Names/Last_Name
```

Figure A.31 Accessing Existing Data

4. To create a html link, we need to use the ‘A’ tag with a href value attribute. We have chosen to make First_Name the clickable element.



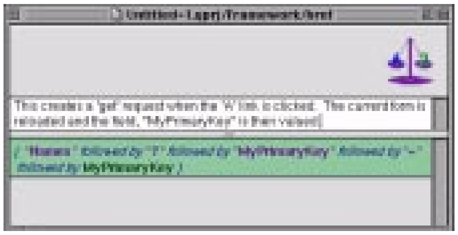
Untitled-1.qprj/Framework/Names

Attribute	Value
HTML file	Names
Comment	

```
    Names
  <BODY>
    <FORM>              method          post
                        action          Names
      <TABLE>
        <TR>
          <TD>
            First Name
          <TD>
            <INPUT>          type           text
                             name           First_Name
        <TR>
          <TD>
            Last Name
          <TD>
            <INPUT>          type           text
                             name           Last_Name
    <tR>
    <QTABLE>            fields          AccessNames/First_Name
                                        AccessNames/Last_Name
                                        AccessNames/MyPrimaryKey
      <TR>
        <TD>
          <QASSIGN>        icon            MyPrimaryKey
                           value           AccessNames/MyPrimaryKey
          <A>              href
            <QVALUE>         icon           AccessNames/First_Name
        <TD>
          <QVALUE>         icon            AccessNames/Last_Name
```

Figure A.32 Creating the HTML Link

5. The href value attribute is constructed as a dynamic element. This abacus is a formatted ‘GET’ request that reloads ‘Names’ and submits the value of MyPrimaryKey.



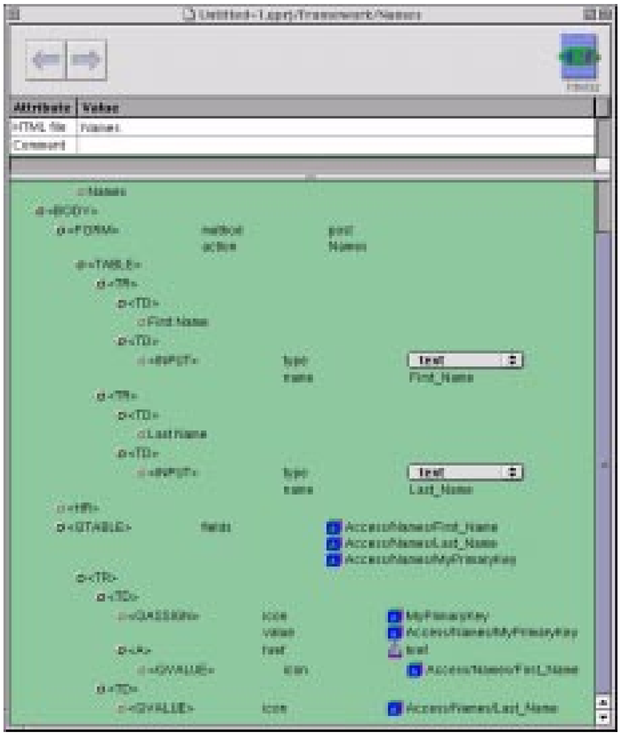
Untitled-1.qprj/Framework/href

This creates a 'get' request when the 'A' link is clicked. The current form is reloaded and the field, "MyPrimaryKey" is then valued.

("Names" followed by "?" followed by "MyPrimaryKey" followed by "=" followed by MyPrimaryKey)

Figure A.33 Setting the href value attribute

6. With the webtemplate open, select the abaus, “href” from the palette and drag it to the href value attribute.



Untitled-1.qprj/Framework/Names

Attribute	Value
HTML file	Names
Comment	

```
    Names
  <BODY>
    <FORM>            method        post
                      action        Names
      <TABLE>
        <TR>
          <TD>
            First Name
          <TD>
            <INPUT>          type          text
                             name          First_Name
        <TR>
          <TD>
            Last Name
          <TD>
            <INPUT>          type          text
                             name          Last_Name
    </TR>
    <QTABLE>          fields        Access/Names/First_Name
                                    Access/Names/Last_Name
                                    Access/Names/MyPrimaryKey
      <TR>
        <TD>
          <QASSIGN>        icon        MyPrimaryKey
                           value       Access/Names/MyPrimaryKey
          <A>              href        text
            <QVALUE>         icon        Access/Names/First_Name
        <TD>
          <QVALUE>         icon        Access/Names/Last_Name
```

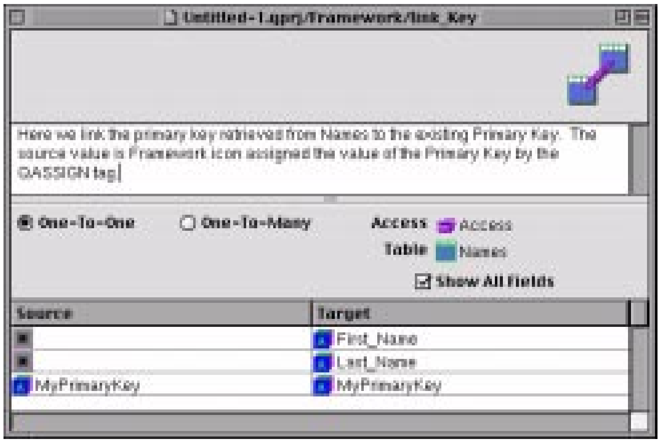
Figure A.34 Drag the HREF Attribute

7. How will the value of ‘MyPrimaryKey’ be used HTML inputs permit default values, also known as the ‘value’ attribute.

   Let’s default these input values to the record being selected from the list.

8. To do this, will be need to obtain First_Name and Last_Name. With MyPrimaryKey in hand, we can create a relationship to our table.

   The relationship below has been named “link_key”.



Untitled-1.qprj/Framework/link_Key

Here we link the primary key retrieved from Names to the existing Primary Key. The source value is Framework icon assigned the value of the Primary Key by the QASSIGN tag

● One-To-One ○ One-To-Many

Access: Access

Table: Names

☑ Show All Fields

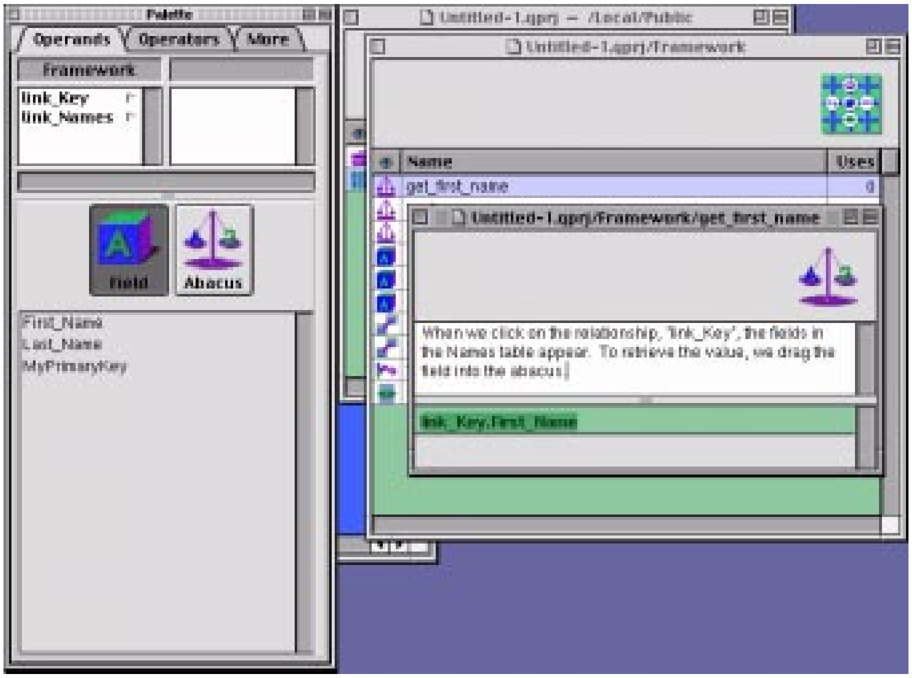
Source	Target
	First_Name
	Last_Name
MyPrimaryKey	MyPrimaryKey

Figure A.35 Creating the Relationship

Choose the Value to Retrieve

Based on this relationship, we merely select the value we wish to retrieve.

1. From the palette, we first choose the link_key relationship at the top of the window, then drag the First_Name field into the abacus window.

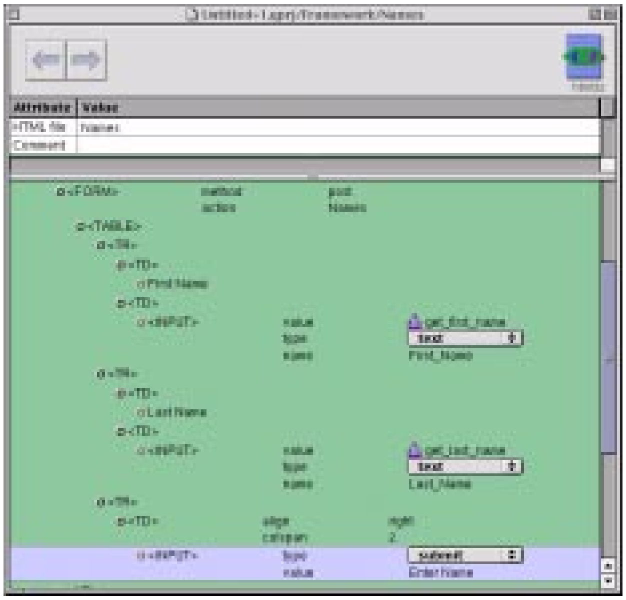


When we click on the relationship, 'link_Key', the fields in the Names table appear. To retrieve the value, we drag the field into the abacus.

link_Key.First_Name

Figure A.36 Identify the Relationship

2. With the webtemplate window open, we select the abacus we created above, 'get_first_name', and drag it to the appropriate html input value. Now, when MyPrimaryKey is valued, Qilan will establish the link and return the desired value.



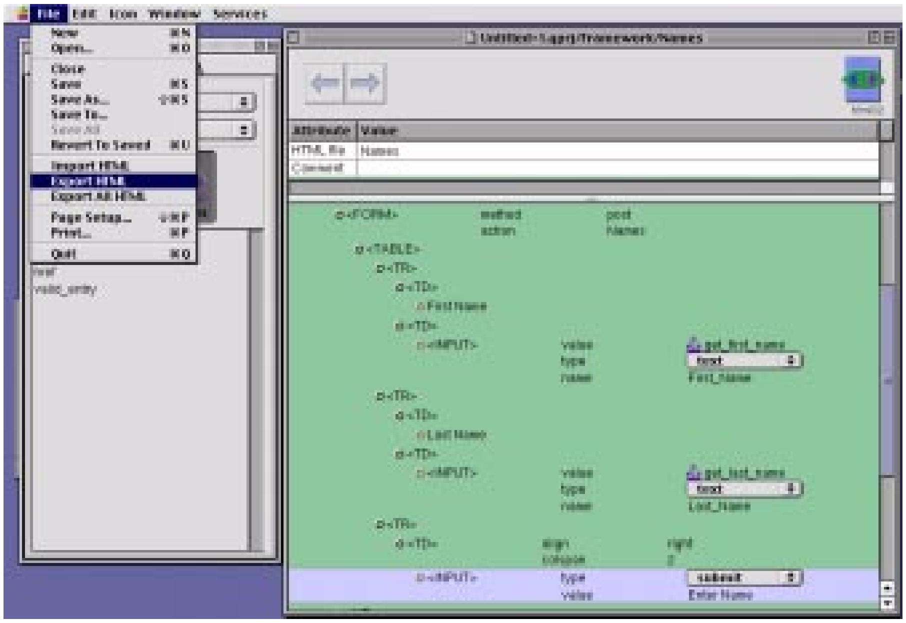
Attribute	Value
HTML file	Names
Comment	

```
<FORM>              method        post
                    action        Names
  <TABLE>
    <TR>
      <TD>
        First Name
      <TD>
        <INPUT>         value     get_first_name
                        type      text
                        name      First_Name
    <TR>
      <TD>
        Last Name
      <TD>
        <INPUT>         value     get_last_name
                        type      text
                        name      Last_Name
    <TR>
      <TD>            align       right
                      colspan     2
        <INPUT>         type      submit
                        value     Enter Name
```

Figure A.37 Establish the Link and Return Value

Exporting the Webtemplate

1. Our final step is to export the webtemplate for access by internet clients. Select Export HTML from the File menu.

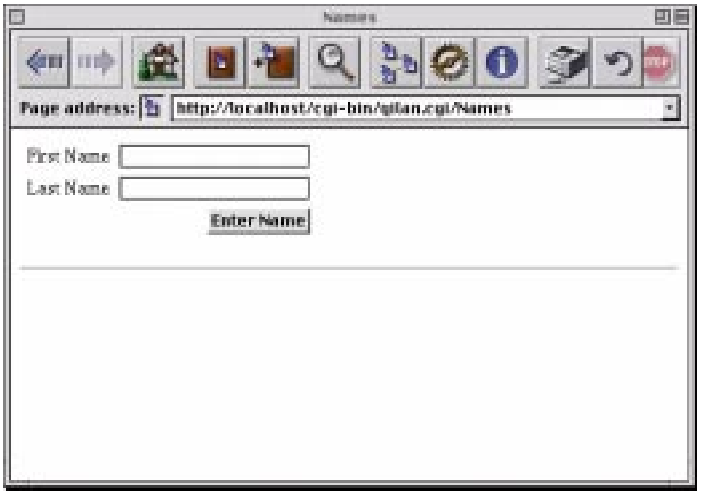


Attribute	Value
HTML file	Names
Comment	

```
<FORM>            method     post
                  action     Names
  <TABLE>
    <TR>
      <TD>
        First Name
      <TD>
        <INPUT>         value      get_first_name
                        type       text
                        name       First_Name
    <TR>
      <TD>
        Last Name
      <TD>
        <INPUT>         value      get_last_name
                        type       text
                        name       Last_Name
    <TR>
      <TD>            align      right
                      colspan    2
        <INPUT>         type       submit
                        value      Enter Name
```

Figure A.38 Export the webtemplate

Here is how our form looks from the browser. Note the path.

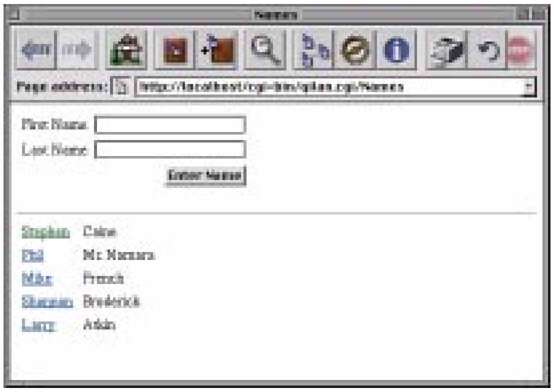


Page address: http://localhost/cgi-bin/qilan.cgi/Names

First Name
Last Name
Enter Name

Figure A.39 The Viewable Form

2. Enter 4 or 5 names. Try entering some names without a first or last name. If the validation was performed correctly, nothing should happen. Also observe how the first name is highlighted as the link.



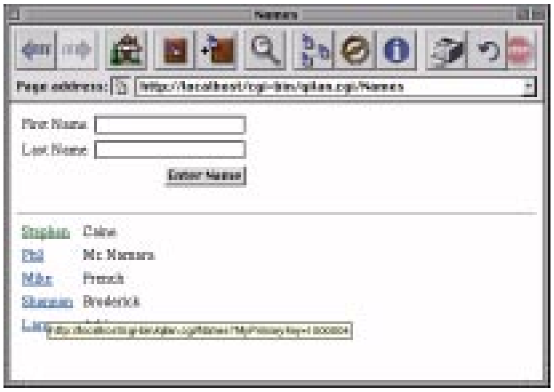
Page address: http://localhost/cgi-bin/qilan.cgi/Names

First Name:
Last Name:

Stephen	Caine
Phil	Mc Namara
Mike	French
Shannon	Broderick
Larry	Atkin

Figure A.40 Testing the Form Validation

3. Move the cursor over the first name in the Names window so that your web browser displays the link. This is the result of the href abacus.



Page address: http://localhost/cgi-bin/qilan.cgi/Names

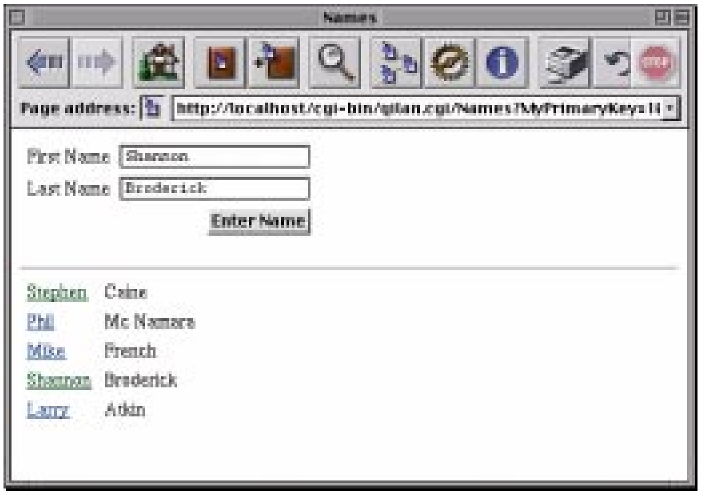
First Name:
Last Name:

Stephen	Caine
Phil	Mc Namara
Mike	French
Shannon	Broderick
Larry	Atkin

http://localhost/cgi-bin/qilan.cgi/Names?MyPrimaryKey=1000004

Figure A.41 Display the HREF Value

4. Clicking the link will submit MyPrimaryKey and reload Names. Qilan will perform the link (according to the relationship) and default the input values.



http://localhost/cgi-bin/qilan.cgi/Names?MyPrimaryKey=1

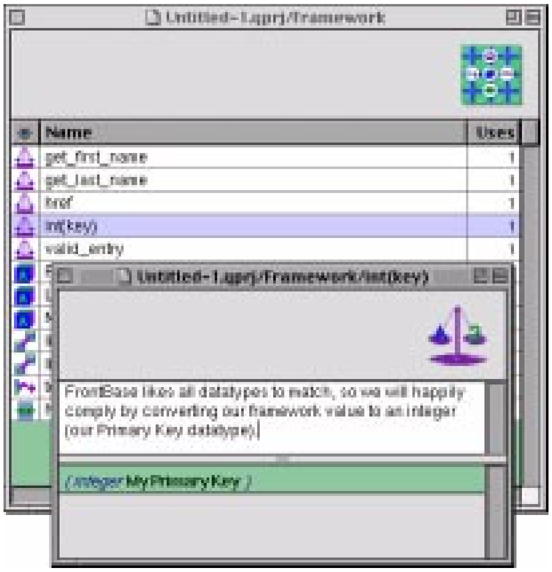
First Name: Shannon

Last Name: Broderick

Stephen	Caine
Phil	Mc Namara
Mike	French
Shannon	Broderick
Larry	Atkin

Figure A.42 Display the Link Values

NOTE: FrontBase may require specific field typing. If you receive errors, use an abacus to convert MyPrimaryKey to an integer, then use this abacus as the relationship link.



Untitled-1.qprj/Framework

Name	Uses
get_first_name	1
get_last_name	1
href	1
int(key)	1
valid_entry	1

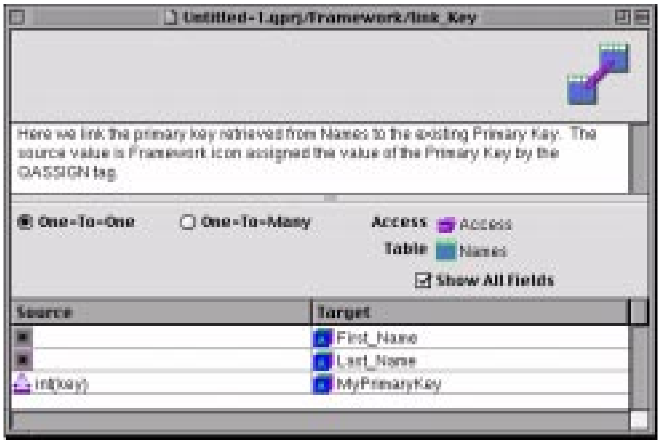
Untitled-1.qprj/Framework/int(key)

FrontBase likes all datatypes to match, so we will happily comply by converting our framework value to an integer (our Primary Key datatype).

(integer MyPrimaryKey)

Figure A.43 Convert MyPrimaryKey to an integer

5. And the new relationship...



Untitled-1.qprj/Framework/link_Key

Here we link the primary key retrieved from Names to the existing Primary Key. The source value is Framework icon assigned the value of the Primary Key by the QASSIGN tag.

● One-To-One ○ One-To-Many

Access: Access
Table: Names

☑ Show All Fields

Source	Target
■	First_Name
■	Last_Name
int(key)	MyPrimaryKey

Figure A.44 Display the New Relationship

Index