

Chapter 11

Database 101

In this Section:

- Gives an overview of databases.
- Explains the needs to make up a database.
- Describes the data table structure and how it relates to reporting.
- Explains database relationships.

Overview

Information to the left, information to the right, information above and below, we are swimming in a sea of information. The advent of the computer age has not helped. In fact, we now have the ability to store absolutely huge amounts of data. This can easily result in information overload for both individuals and managers. Something is needed to manage this overload potential. The first thing is that the information must be organized and stored in an efficient manner that aids in the quest to extract answer(s) from that wealth of data. One of the methods to retrieve answers is to use a reporting tool (like RAVE) to organize and display the answers in a manner that is easily understood by the reader of the report. But before a reporting tool can extract the information, it must be aware of the structure of the source of information. That container of the information in simple terms is a data table.

This section is intended to assist those that are not familiar with database design goals or might need a quick refresher. This section is not designed to cover this subject in depth, as this is best left to the many thick books already published on this subject and to more formal educational courses.

What is a database?



Let's begin with some real world examples to describe the concepts of a data table. For example, a library contains lots of information stored in rows and rows of shelves containing books, pamphlets, etc., which is usually arranged according to the Dewey decimal system. To find a book, a card catalog is used to look up the subject or author, noting the Dewey decimal code. Then, the desired book is found and retrieved by going through the stack that contains the book code (if it isn't checked out).

Look up options are far more flexible if the library has a computerized card catalog. Now picture a library with no Dewey decimal system, no order but still the same amount of books. Try and find something in that library. It would be very tough and take a long period of time.

Now, change the words. Change book to information and library to database and we are on the way to explaining what and why of database management system. A library is a collection of knowledge (books) arranged by the Dewey decimal system.



A data table is a collection of information arranged according to some predefined table structure. We will now examine what is needed to design or use a data table.

Terms

To better understand what it takes to create a database, it is necessary to understand the database lingo.

Database Management System

A Database Management System (DBMS) is the complete system for managing data (information). There are many commercial examples of companies that are known for their DBMS programs. Some of the more well known include Informix, Interbase, Microsoft Access, Microsoft SQL, Oracle, Sybase, and many others. A DBMS can be confused with the physical data file(s). It must be noted that the data table is only one part of a DBMS. However, the real world does require that when extracting information from a DBMS that the structural concepts of a data table be understood. Fortunately, most of the terms are common between the various DBMS. This does not mean a person needs to be an expert in the database field, just knowledgeable.

Table

A data "Table" is a collection of things like phone numbers, names and addresses. The "report" of this example table would be a telephone book. Of course, a telephone book has some restrictions, like do not include unlisted phone numbers in any print outputs. In some cases, the phone company will have a billing address that is different from the address location for the phone itself.



Now look at a phone book and notice that it is arranged in rows and columns. A row (sometimes called a record) shows the items related to a single member (person) of that table. Each column (sometimes called a field) is giving a specific piece of information about that member, like their name or phone number. While we are here, the order of the phone book (table) is done with an index (sometimes called a key). So, a phone book could be arranged by last name or by phone number.

Structure

Each specific piece of information about a member of a table has attributes that define it for its DBMS.

Typical attributes for each table (file) include:

- date the table was created
- date the table was last modified
- number of columns defined for table

Typical attributes for each row (record) include:

- Length
- Record number
- Status (locked - deleted)

Typical attributes for each column (field) include:

- Name - a column name that usually "describes" its contents

- Type - character, numeric, date, time, blob etc.
- Length - maximum size of this item (might include number of decimals for numeric types)

A data table is comprised of information arranged according to a pre-defined structure. The structure of each table is important, because the row and column definitions are used by a reporting tool to extract, arrange and show information.

In the phone table example, there are the following fields: name, address, and phone number information. How are those columns (fields) defined? They are defined by the name of the column; meaning name might be called "name", address called "address", and phone number called "phone". But defining the main fields may not end at these simple definitions. For example the "name" field might be broken into three or more sub-fields, like "first", "middle" and "last". These three pieces as well as the first three main fields could then be manipulated as needed by any reporting tools. The three name sub-fields could be combined for a "full name" output or the first letter of the "first" name could be combined with "last" to make a short version of the name. Also, "phone" could be called "number" or "telephone" and the phone number could probably contain only the numbers with no formatting characters. Rave makes the task of understanding the table structure relatively easy as it does most of the work. Once the data connection is made, Rave will extract the table structure and from there the columns (fields) can be chosen as desired.

Query

Rarely is there a need to print all the information in a database. Frequently, there is a need to reduce data by selecting a range of rows (records) in a data table. Data reduction is an important part of information management. One method of accomplishing a reduction is to do a query of the table to only show those rows that meet the range limits that need to be retrieved.

One of the more common query methods is called SQL, which stands for Structured Query Language. Some data tables are SQL compliant. A SQL query allows inquires of a DBMS for information, and the DBMS searches its tables according to the given SQL statement and return a "set" of rows (records) that meet that query.

Client-Server

A method of setting up computer programs used by many people, where the database resides on a central computer accessible to all (the back end), and the user interface resides on the user's computer (the front end). All selection of specific data rows and processing is done back on the host computer minimizing the transmission of data through the network cabling. Significant improvements are achieved to the overall system performance by only transmitting relevant data through the cables. Thus, a well-configured Client-Server system can yield excellent performance gains in applications used by several people.

Relational Table

Now the phone book example is good for simple data tables (flat files). However, it is not an efficient design for many real-world needs. We can stay with the phone company, just not the phone book for a more typical need. Every month the phone company needs to send a bill out to each customer. This bill is a good example of a "master-detail" type report. One part of the bill comprises the master (static) information about the customer like: their name, billing address and phone number. Another part of the bill contains the detail (variable) information, like an output line for each phone call made including its length and cost. The variable information could be only one page or for some customers many pages.

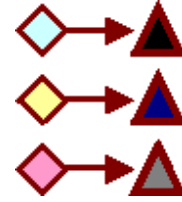
There is one important detail in any master-detail work, understanding the concept of a primary key (link key). When a customer gets a phone bill, that customer expects only to receive a listing of their calls; they don't expect calls made by other people to be listed. In order for this customer bill to be created, this means that there needs to be only one master row, called a primary key, for

all the detail rows. In the customer bill, the primary (link) key would be the phone number. The primary key must be unique, which in general usually makes it more difficult to change a primary key once it has been assigned. But, there are some DBMS that allow the primary key to be a combination of columns (fields).

That is the quick review of Database operations. To ensure efficient design, with table structure control, there are a few more subjects to be aware of like: "Client-Server", "Normalization", "Relational DBMS", "One to One" "One to Many" and "Many to Many" relations.

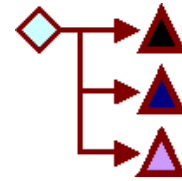
One-to-One

One-to-one is two separate tables with the same primary key. Therefore, knowing the primary key allows look up of data for a first row in one table, and for a second row in another table. One to one relationships are rare because it's usually easier to put both tables' columns into a single table. One-to-one relationships probably don't meet the first level of "Normalization" rules.



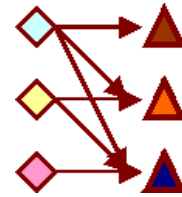
One-to-Many

This is accomplished by placing the primary key value of the ONE side of the relationship into an ordinary column of the MANY side of the relationship. For instance, if each employee can belong to only one department, then you could have a "department" column in the employee table, which, for each employee, would be filled with the primary key value for the row of his department in the department table.



Many-to-Many

This is difficult to explain, as the real-world examples are complex. If there were several employees that were part of several departments, then this would be a many to many situation



View

Some DBMS have an ability for a user to define a form to show contents of information taken from one or more other tables. A view is a definition of a form and has no data of its own. It often includes a query that is executed whenever the view is the subject of a command. Use views to control access to individual columns in a table, or to make two or more tables appear as one.

Reporting

How does all this relate to designing reports? With the definitions defined, understanding the report design structure and options will be easier. Now you are ready to read the sections on DataView Connection, DataField which will tell how to use Rave to connect to your data table(s) and how to identify the specific column (field) contents to the Rave designer.

Connecting to Data

In this Section:

- Explains what a database connection is.
- Show the different methods of retrieving data from a database.
- Explains the SQL Editor and how it is used.

Database Connections

A reporting tool would be quite pointless if it did not allow methods of displaying information contained in databases. One could only go so far with designing non-data reports. Connecting to data in Rave is one of the more powerful features, not only because of the wide variety of databases and methods supported, but also due to the simplicity of the steps required to make a report data-enabled.

A database connection in Rave is the primary pillar for connecting to a database with SQL DataViews.

Creating a Database Connection



To create a new Database Connection, click on the DataView icon on the Report toolbar, bringing up the Data Connection Wizard. The first step is to select Database Connection from the list of options available. When done, click Next.

After clicking on the Next button, the second step of the wizard displays the different type of connection options available. The selection depends on the different Rave DIBL Links that are installed on the system. DIBL stands for Database Independent Layer. It is a proprietary system that allows interactions with databases independently of what these database system are (whether it be SQL Server, Interbase, Oracle, etc).

In a folder under the Rave root directory, there should be a folder called DIBLLinks. Inside the folder there are a certain amount of files with extension rvd. The files are loaded when the application first starts. Depending on the number of files, the number of available connection options varies.

It is possible to have many types of connections displayed in the Database Connection Type Dialog screen. It is left up to the user to decide which connection type is the most appropriate. Fewer connection types may be displayed depending on which drivers were installed.

Once one has been selected, the corresponding dialog box will appear in the next step asking for the connection details. This varies from one to another, but generally consists of a path to the database, a server name if the database is not local and optionally a username and password to access the server.



After creating a database connection, it will appear in the Project Tree from where it can be accessed. The properties for a Database Connection can be shown just like any other component by clicking on the connection component in the Project Tree panel.

There are several properties that are specific to the Database component. The most important ones are the AuthDesign and AuthRun properties. Many times, the platform (and server) that one develops the report on does not coincide with that of deployment. Specifically, one characteristic that is most likely to change is the access codes to the server (username and password). Rave has been designed keeping this in mind, and once again making the deployment task easier. For

this, the AuthDesign and AuthRun properties can be used to provide design-time and runtime (deployment) information regarding the database and server.

AuthDesign contains the parameters that were specified when the Database component was made using the wizard and it can be changed by clicking on the ellipse button in the property field. AuthRun is the information for deployment.

LinkType represents the type that was selected when creating the Database connection. Again, similar to other components, there are common properties such as Name, FullName and Description, etc, which are not be explained again to prevent redundancy.

Direct DataViews (BE only)

DirectDataViews provide a link to data connection components located within an application. When a DirectDataView is created, a list of all available data connection components is displayed. The *Name* property value of the data connection components provides the link between it and the DirectDataView (through the *ConnectionName* property in Rave). If connecting to data connections that have event code attached to them, you must have the application actually running and the form containing the data connection components created.

Driver DataViews

DriverDataViews perform the same function as a DirectDataViews except they are self contained and typically obtain their data set from a SQL statement. DriverDataViews also require the use of a valid database connection.



Once the database connection has been established, click on New Data Object to create a new DriverDataView.

The Wizard will take the user through the necessary steps to complete the configuration. After selecting DriverDataView from the dialog box, choose the database connection that it is going to be connected to. If the connection to the database is successful, the wizard will then display the Query Editor dialog box. Before continuing with an explanation of the DriverDataView, let's examine the Query Editor.

Query Property Editor

The Query Property Editor is used for generating a SQL statement that returns a result set from the database.

There are two ways to use the editor. The first is to take advantage of the graphical interface for constructing the SQL sentence. On the left side of the dialog box, a list of tables available in the current database is displayed. By simply dragging and dropping from the list to the gray area, the corresponding SQL statement will be generated.

Once the table object appears, by clicking on the tick boxes, the list of fields that should appear in the result set can be defined. By default "*" is marked, which means that all fields will be returned. To select individual fields use the tick fields to make the appropriate selections.

Tables can also be linked (joined). To do this, simply drag another table over and trace a line from one field in the first table to another field in the second time (the figure above shows a join between PRODUCTID of table BUGS and PRODUCTID of table PRODUCTCODES. To view the results from here, click on the Results tab and the data will be displayed in a grid.

Alternatively, the editor can be used to write an SQL sentence directly. To do this, click on the Editor button that appears at the bottom. To enter a user-defined SQL statement, click on the "User Defined SQL" checkbox and type in the desired SQL.

Once the query has been constructed, clicking on the Ok button will return control to the Rave designer.

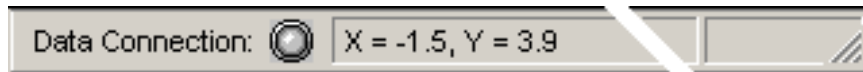
The SQL Data View component should now appear in the Project Tree. By expanding the + sign

next to it, a list of all the fields that have been selected in the SQL statement will drop down. Properties for each field can then be accessed from the Property Panel prior to selecting the field in the Project Tree.

Although fields can be placed in a report by using the corresponding component from the Report tab, it is much easier to work directly with the Project Tree to accomplish the same task. Pressing down the CTRL button, fields can be dragged and dropped directly from the Project Tree on to the form designer.





Once the DirectDataView has been setup using the Query Editor, none of the other properties are required for accessing the data.

Status Bar



At the bottom of the RAVE screen is the status bar. The status bar will provide some information about the item currently selected. So, watch the bar when designing a report.

The data connection LED will provide status of the Rave data system by the color of the LED.

LED	Color	LED Status
Gray		Connection(s) are inactive
Yellow		Connection active but waiting for response
Green		Connection active and busy
Red		Connection active but has exceeded time-out delay

The X and Y figures are the coordinates of the mouse pointer. Move the mouse around and watch the X, Y amounts change. When a shape is dropped on the page, the size of a shape will appear while it is being created as seen by the dX and dY position. The "d" stands for delta.

Chapter 13

Wizards

In this Section:

- List guides to help in simple report structure.
- Guides will provide detailed steps to complete a reporting task.

Wizards

Wizards are a new feature to Rave that allow certain types of reports to be created by answering a series of questions. They can be found under the "Tool" menu option. The Wizards can be added and tailored to each user's needs. It is a perfect way to minimize the end-users interface with reporting tasks.

There are two Wizards, a "Simple Table" and a "Master-Detail". Wizards can be designed to ask for the data connection and allow the user to choose the fields that are needed on a report. It is important to note that an active DataView should be available prior to running the wizards, whether it is a Direct Data View or an SQL Data View. Simple Reports are generally used for listings. Common uses include Client reports, Telephone lists, etc. The Master-Detail Wizard is used when more complex reporting is required, such as invoices, product order lists, etc.

Independent of the one that is executed, there are several steps that are common to both. These include the DataView selection, field selection, etc and are covered in greater detail in the exercises included at the end of this chapter.

It is very important that the user understands that these are general purpose reporting wizards and as such, some aspects are not treated with great detail (amount of text that would fit on a page, layout, etc). They can be used as the building blocks for a more complex report or can be adjusted to suite a particular layout required. If the intention is to build very complex reports, the wizards are not recommended for this purpose.

Exercises

Below are example exercises of both Simple and Master-Detail wizards. Since most of the steps are similar for both wizards, they will only be explained in detail in the first exercise. It is highly recommended to start with the Simple Wizard first to get a good grasp on the concepts presented.

Exercise: Simple Wizard

1. The first step is to select the desired DataView that will provide the data to the report.
2. Once the DataView has been selected, a list of fields that should appear on the report can be chosen. Fields are selected by clicking on the box on the left. All fields can be selected by clicking on the All button at the top. Note however that because this is a simple wizard, choosing more fields that would fit on the page will make them overlap. Manual intervention is required after the Wizard is complete to correct this problem.
3. If more than one field was selected in the previous step, the Wizard will ask for the ordering of the fields. Moving a field up will place it on the left side of the page. Move it to the lowest position will move the field to the right margin.
4. After the fields are placed in your desired order, the Report layout can be set. Values include the title and the page margins.
5. The last step of the Simple Report Wizard is to choose the fonts that are going to be used in the report. It gives the possibility of changing three fonts: title (of the report),

caption (headers of the field names) and the body (actual field values).

CustomerDVRegion: CustomerDVTitleBand (BGRDr gb 1PC)					
Simple Table Report					
CustomerDVRegion: CustomerDVBand (BGRDr gb 1PC)					
Contact	Company	Addr1	City	State	Zip
CustomerDVRegion: CustomerDVDataBand (Master 1PC)					
[Contact]	[Company]	[Addr1]	[City]	[State]	[Zip]

Exercise: Master Detail Wizard

The Master-Detail Report Wizard has a few additional steps more than the Simple Wizard Report. This is due to the fact that more than one table is going to be taking part in the report. In particular, there will be a master (for example customer information) and a detail (items ordered).

1. Similar to the Simple Report Wizard, the first step is to choose the DataView. However, in this case the DataView chosen corresponds to the Master table.
2. In step two, the Detail table can be selected. Note how the DataView selected in step 1 is no longer available to avoid errors.

Similar to the Simple Wizard, the next steps are for selecting the fields and ordering of both the master and the detail table.

A new step is determining the key fields. These are field that relate one table to the other.

After this, the remaining steps are identical to that of the Simple Report Wizard.

Once everything is complete, clicking on Generate will produce a simple master-detail report that, again, can later be adjusted and "touched up" to ones particular needs. The figure below shows a sample output of the Wizard.

SQLDataView1Region: SQLDataView1TitleBand (BGRDr gb 1PC)	
Master/Detail Report	
SQLDataView1Region: SQLDataView1Band (BGRDr gb 1PC)	
BUG_ID	DUPLICATE_ID
SQLDataView1Region: SQLDataView1DataBand (Master 1PC)	
[BUG_ID]	[DUPLICATE_ID]
SQLDataView1Region: SQLDataView2Band (BGRDr gb 1PC)	
STATUS	
SQLDataView1Region: SQLDataView2DataBand (BGRDr gb 1PC)	
[STATUS]	

Report Components

In this Section:

- Gives more detail on using report components.
- Examples are given to aid in comprehension.

Overview

The Report Toolbar is one of the most used when working with data-aware reports. It is very vital to understand the function of each component, especially since the basics of all database reports are based on the Report Toolbar.



The following components are from the Report Toolbar that are frequently used in designing your reports:

Band, CalcText, DataBand, DataMemo, DataText, Region

The following components from the Report Toolbar will be covered in the Advanced Components section of the manual.

CalcController, CalcOp, CalcTotal, DataCycle, DataMirrorSection

The following Report components are non-visual components

CalcController, CalcOp, CalcTotal, DataCycle

There are some general characteristics of the components that need to be discussed before describing each component individually. First, let's talk about the visual color of the components. For the most part the components look similar in color to all other components, they are gray. But, at the end of the toolbar there are four green colored components. The green indicates that these are non-visual components. When placed on the Page, they cannot be seen on the screen, in preview, or in print. Instead, to 'see' the components the user must use the Project Tree to select, move, and delete the component from the Page.

Most of the Report Components can simply be placed on the Page. The only two exceptions are the Band and the DataBand. These must be placed in a Region component. The behavior of the component is also controlled by the settings made with the "Band Style Editor", which is found through the BandStyle property for both components.

A Report can be designed into a typical report, like one that uses strictly Bands. But, the unique feature of Rave is that the report can also be designed to other specifications. The Bands of the report can be moved and resized to any desired shape.

When designing a Report, the designer window can be changed to the preferences of the Report Designer.

There are several ways to control the "look" of the Bands while in the design mode. The first is whether ALL Band Headers need to be seen or not. This setting can be changed with the Designer Toolbar. Another method is to toggle the visibility of the Band contents. If some of the Band designs are stable and they need to be temporarily hidden, then try setting the

DesignerHide property to True. Only the Band(s) with DesignerHide set to False will show. This might be needed if there are a large number of Bands or a Band that occupies a large space. The setting of the DesignerHide property has NO effect on what will be printed, only what is shown on the designer Page.

Region



Before one can use a Region component properly, it is important to understand what it is. A Region is a container for Bands. In its simplest form the Region could be the whole Page. This would probably be True for reports that are a list type.

Many master-detail reports could be made to fit a single Region design. However, do not be limited by thinking of Regions as the whole Page. The properties for a Region basically deal with its size and location on the Page. Creative use of Regions will give more flexibility when trying to design complex reports. Multiple Regions can be placed on a single Page. They could be side-by-side, one above the other or stagger about the Page. Do NOT confuse a Region with a section. Region components contain Bands and only Bands. A Section component can contain any group of components, including other Region components.

When working with Bands, there is a simple rule that must be followed: Bands must be in a Region. Notice that the number of Regions on a Page is not limited nor is the number of Bands within a Region. As long as the report can be "visualized" mentally, a combination of Regions and Bands can be used to solve any difficulties faced when actually putting the visual thoughts into design. There are two band types: Band and DataBand. The latter is used when iteration is required, for example, in a master-detail report. The first is used for non-iterating needs.

DataBand



The DataBand component is a data-aware Band is used to display iterating information from a data view. In general, a DataBand will contain several DataText components.

A typical use for the DataBand would be on an invoice. An invoice normally consists of a header including information such as the date, invoice number, clients name and address, and one or more lines containing the items that are being invoiced. In this scenario, the customer table would be the master table and the items would be the detail table. Information about the items would be placed in the DataBand and the controller would be the master table.

Band



The Band component is for items that are "fixed" and do not change on the Page. In general, the Band component will contain Text and CalcText components. The primary examples of this would be headers and footers. The Band component can contain data-aware components, so a table field can be in the Band. A group footer might have a '{CustomerDV.CustomerName} Totals' on this Band.

An important property for the Band component is the "ControllerBand". This property determines which DataBand this Band belongs to (or is controlled by). When the controlling Band has been set notice that the graphic symbol on the Band will point in the direction of that controlling Band and that the color of the symbols will match.

There is a preference setting called "Always Show Band Headers". This setting will change the appearance of the Bands while in the design mode. Having this setting "off" will give an appearance closer to the actual output, but will not show the Band Headers with their symbols and codes. When first starting to use Rave it might be beneficial to try it with this setting "on" and take advantage of the visual clues provided by the Band Headers. Once comfortable with the use of Bands, change this setting to fit any necessary needs or preferences. The letter codes shown on the right of the Band are explained later in this chapter under the section called "Editor - Band Style". Basically, they give information about Band behavior. The bold letters are ON or active while the subdued letters are off or inactive.

Editor - Band Style

Go to a Band Style property on a Band or DataBand component and click on the ellipse button to open the editor for Band Styles.

This provides a simple method to select the features wanted for that Band by using the check boxes to activate or deactivate them. Note that a Band can have several different features active at a time. This means that it is even possible for a Band to be both a Body Header and Body Footer at the same time.

The display area in the Band Style Editor has been designed to represent the flow of a Report in pseudo layout style. DataBand(s) are duplicated 3 times on purpose to show that this is a repeating data area. The current Band that is being edited will be displayed with both **Bold** and Underline formatting.

Both symbols and letters are used on the Band Style Editor display area and one the Bands in the Page Layout Area and are designed be informative about each Band's behavior. The major difference between these two representations is that the Band style editor display will arrange the Bands in a pseudo flow according to the definitions of each Band. The Region display of the Bands will be arranged in the order that they are placed during design. The order of operation is controlled in some cases by this order. Headers (capital letters, BGR) will print first, then the DataBand, then the Footers (lower case letters, bgr) for each level. However, if there is more than one header defined for a particular level, then each header Band will be processed in the order that they are arranged in the Region. So, it is technically possible to put all the Headers at the top, all the DataBands in middle and all the Footers at the bottom of a Region for all levels of a master-detail. Or each level could be "grouped", with the appropriate Headers, Footers and DataBands all together for each level. Rave allows the Region layout maybe be used in a way that makes the most sense to the design flow. Just remember the order of precedence of like Bands at the same level is controlled by their order within the Region.

There are several symbols that are designed to show the Parent - Child/Master - Detail relations of the various Bands. The triangle symbol (up/down arrows) indicates that the Band is controlled by a Master Band with the same color (level) and can be found in the direction of the arrow. The Diamond symbol represents a Master or Controlling Band. These symbols are both color coded and indented to represent the level of Master - Detail flow. Remember that we could have Master - Detail - Detail where both details are both controlled by the same Master or one of the Details could be controlled by the other Detail.

The title bar of each Band contains information about that Band. On the left side of the Band is a name that indicates the Region it is in - "RegionName:BandName". The right side of Band uses several letters to remind you of the Band style settings for that Band. The order of these letters on a master Band is "MASTER 1PC". The order of these letters on a controlled Band is "BGRDrgb1PC". If the letter is subdued (gray) then that setting is inactive (off). If the letter is bold then the setting is active (ON). The following table shows the various letters and what they mean.

DataText



The DataText component is data-aware. This means it can be used to display a field from a dataset anywhere on the Page layout. For example, this could be used to print the customer information inside of a DataBand. The DataText component however is not limited to printing only database data. Through the Data Text Editor (accessible through the DataField property), Project Parameters can be printed, as can Report Variables as well as the DataFields. See the topic Editor-DataText, for more information. The LookupDataView, LookupDisplay and LookupField properties define a lookup definition to be displayed instead of the DataView:DataField properties.

Editor - DataText

There are two options available for entering data in a DataField property. The first is to select a single field using the drop option. This is fine for normal database reporting needs where only a

single data field for each DataText item may be needed. However, there are many reporting requirements where various fields will need to be combined together. Two common examples are City State and Zip Code or Firstname Lastname combinations. In code this would be accomplished using a statement like the following:

```
City + '_' + State + '___' + Zip or FirstName + "_" + LastName
```

NOTE:The underscore character above represents a space for example purposes only.

The DataField property has a DataText Editor which assists in building complex composite fields. To do this click on the ellipsis and open the DataText Editor. This editor will give the power to concatenate fields, parameters, or variables together to build a very complex data aware text field simply by dropping the different list boxes and selecting the item desired.

There are a lot of combinations in this editor, they will be covered quickly here, but try the different combinations in practice and it should help the learning curve.

Note that the dialog box is divided into 4 groups: Data Fields, Report Variables, Project Parameters, Post Initialize Variables and Data Text. Data Text is the result window. Watch this window when inserting different items. The two buttons on the right side of this window are +, "plus", or &, "ampersand". The "plus" sign will add the two items together with no spaces while the "ampersand" will concatenate them with a single space. So, the first step is to decide on doing a + or &, then selecting the text from one of the three groups above the Data Text window.

So as an example, to add the field "OrderNo" to the "CustNo", click once on the "+" sign, go up to the DataField group, drop the DataField list box, and select "OrderNo". Then, click once on the "Insert Field" button and that will be added for the DataText window. The result in the DataText window would be "CustNo + OrderNumber". Even more DataFields can still be added. Notice the "Selected" item in the DataView group. If there is more than one data view active, then select another Data View, and then pick a field from that Data View.

However, do not be restricted to thinking of combining only DataFields. "Report Variables" and "Project Parameters" can be combined as well. Go to the "Report Variables" group and drop the list box for variables and take note of the ones that are available.

Another item available is Project Parameters. This could be a "UserName", "ReportTitle" or "UserOption" parameter initialized by the application. To create the list of "Project Parameters", select the Project node in the Project Tree (very top item). Then in the "Properties" panel there will be a "Parameters" property. Click on the ellipsis button to get a typical strings editor where you can enter the different parameters that will pass to Rave from the application, like "UserName" etc.

Caution:

Remember to use a "+" or "&" between each item that you are combining in the Data Text window. You can type in the Data Text window, so you can correct errors by highlighting, deleting or replacing erroneous entries made in the Data Text stream.

DataMemo



The DataMemo component is data-aware. This means it can display a memo field from a DataView just about anywhere on the Page layout. The main difference between the DataMemo component and the DataText component is that the DataMemo component is for printing text that may take more than one line and will thus need to be wrapped. For example, this could be used to print the remarks about a customer invoice at the bottom of each Page of an invoice.

One use of the DataMemo component is to do mail merge functions. The easiest way to accomplish this is to set the DataView and DataField property control to the source of the Memo field. Then launch the mail merge editor by clicking on the ellipsis button of the MailMergeItems property. This will allow the items to be set within that Memo that will be changed.

Note: The "Case sensitive" check box is empty (off). If case is important, then be sure to check

this box.

To use the Mail Merge Editor, click on the "Add" button. Now type in the "Search Token" window the item that is in the Memo and will be replaced. After the token is entered, either type the replacement string in the "Replacement" window or click on the "Edit" button and it will start the DataText Editor that will help in selecting different DataViews and Fields.

CalcText



The CalcText component is data-aware. The main difference between the DataText component and the CalcText component is that the CalcText component is specially designed to do some form of calculation and display the results of that calculation. The CalcType property determines the type of calculation being performed and includes Average, Count, Maximum, Minimum, and Sum. For example, this can be used to print the Totals of an invoice at the top of each page of an invoice.

The CountBlanks property determines whether blank field values are included in the Average and Count calculation methods. If RunningTotal is True then the calculation will not be reset to 0 each time it is printed.

Project Components

In this Section:

- Main function of the Project Toolbar discussed.
- Project Manager and special properties covered.
- Report, Page and Global Page components and their special properties covered.
- Data Connections and Objects used to connect to Reporting Server discussed.
- Differences of Security components explained.
- Explanation of SQL and Direct DataView are given.

Overview

The Project Toolbar provides the basic functionalities for a project. These functions are essential building blocks for all reports. What makes these functions so important is the fact that these functions also provide basic building block components.



The main functions we will be concerned with the first button (Project Manager) and the second section in the toolbar (New Report, New Global Page, New DataView and New Page). These functions are used to add the supporting structure to your report design.

Each will be discussed in the following sections.

Project Manager



The Project Manager is the component from which all other components are created. Everything is placed under the Project Manager (a.k.a. RaveProject) in the Project Tree. This component is created when any new Rave file is created. There can only be one Project Manager per Rave file, this is unlike most of the other components.

For clarification, Project Manager, RaveProject, and New Project all refer to the same thing. They refer to the main Project file that contains everything.

Like every component in Rave, the Project Manager has properties. To see them, select RaveProject in the Project Tree, and then look in the Property Panel.

To quickly create a new project, simply click on the New Project button in the Project Toolbar. This will create a new file, as well as the new Project Manager.

Noted Project Manager Properties

The AdminPassword property allows the Rave Server administrator to limit who has access to the data from which the reports are created. This is important to help limit capabilities in Rave.

Parameters property allows items calculated by applications (such as Delphi) to be passed into the report to be used by other components with in the report. When clicking on the Parameters property ellipse button, the DataText Editor will allow a parameter name for each line.

PIVars Parameter is for content that is not typical, and needs to be dynamically modified within the report. This allows for content that needs to be modified in a way that is not typical of table calculation or manipulation.

SecurityControl

Report



The Report component contains the Pages of a Report. There can be multiple Reports in one Project, and each Report can have multiple Pages in that Report.

Each Report has properties. To see the properties, select the Report and view the Property Panel.

Page



The Page component is the base visual component upon which Drawing, Reporting, Standard, and Barcode components are placed. This is where all the designing and layout of a Report are completed.

The Page also has properties which can be viewed by clicking in a blank area somewhere on the page.

Global Page



The Global Pages are located under the Global Page Catalog node in the Project Tree. These Global Page Definitions are used as Master Page definitions. These Pages can also be Mirrored. Global Pages can contain Page layouts for things like letterheads, forms, watermark designs, and other Page layouts that can serve as a foundation for several Reports in the Project. An example would be mirroring a Global Page where you wanted to print the same contents (link an "Invoice"), but you wanted to put a different caption at the bottom of the Pages like "Original", "File Copy", and "Shipping".

Global Pages also have properties. Select the Global Pages from the Project Tree and look at the Property Panel to see all the properties.

Data Objects



Data Objects are the Data Connection components used to connect to data, or components used to control who is allowed to see which Reports from the Reporting Server.

Clicking on the New Data Object button on the Project Toolbar creates each Data Connection. After clicking on the button, the Data Connection dialog will appear. From this point, one of five selections can be made. The Data Object component choices available are DataLookupSecurity Controller, Database Connection, Direct Data View, Simple Security Controller, and SQL Data View.

The details of each Data Connection component will be covered in the next few sections. Once a Data Object is selected, the Data Object component will be placed in the Data View Dictionary in the Project Tree. Selecting the component and looking in the Property Panel will reveal it's associated properties.

Database Connection



The Database Connection component is the Data Object used to connect to data. This component can be added to a Project by clicking on the New Data Object button on the Project Toolbar, then selecting Database Connection from the Data Connections dialog.

Once the Database Connection is chosen from the Data Connection dialog, the Database Connection Type dialog will appear. This is where the type of connection that will be used to

connect to that data will be chosen. In this image there is only one selection, but depending on what types of connections you have, there could be many more selections. After choosing the right data connection type, click Finish.

The Database Connection component, like all other components, does have properties. Select the component from the Project Tree and see the properties in the Property Panel.

Security Components

Using a Security component can control security to individual Reports. TRaveBaseSecurity cannot be used itself, as it is an abstract. However, you can create a descendant or TRaveBaseSecurity to implement your own security scheme or use one of the pre-built TRaveBaseSecurity descendants included with Rave. Rave currently includes two TRaveBaseSecurity descendants: SimpleSecurity and LookupSecurity.

To use a Security component, create one and then set the Report's SecurityControl property to the Security component instance that you want to control access to the Report.

Security components currently only affect Reports when served via the Reporting Server. When an unauthenticated user attempts to access a Secured Report, HTTP authentication will be used to authenticate the user.

Simple Security Controller



TRaveSimpleSecurity implements the most basic form of security by using a simple list of username and password pairs in the UserList property. UserList contains one username and password pair per line in the format:

Username=password

CaseMatters is a Boolean property that controls whether or not the password is case sensitive. Username is always case insensitive.

Data Lookup Security Controller



TRaveLookupSecurity allows username and password pairs to be checked against entries in a database table.

DataView specifies the DataView to use to lookup the username and password.

UserField is the field that is used to look up the username. PasswordField is the field that contains the password to verify against.

SQL Data View



SQL Data Views are used for creating self-contained DataViews to SQL databases. A database is specified using the Database property.

Parameters can be specified using the Params property.

The SQL to use is entered into the SQL property. At design time, a property editor for the SQL property invokes a visual query builder.

Bar Code Components

In this Section:

- Describes how to place Bar Codes
- Describes how to encode Bar Codes.
- Gives brief descriptions of the Bar Code types.

Bar Code Component Basics

Bar Code components are used to create many different kinds of Bar Codes in a Report. Bar Codes are for users who know exactly what they need, as it requires background knowledge about Bar Codes and how they are used. It is not expected that the beginning user would have the background to use these components.



To place a Bar Code, click on the needed Bar Code component button and click on the Page.

To define the Bar Code value, go to the Property Panel and type the value into the Text property box. For Bar Codes containing check characters, please do not enter the check character, as it will be calculated automatically.

Brief Bar Code Descriptions

PostNetBarCode



PostNetBarCode uses the POSTNET (POSTal Numeric Encoding Technique) bar code and is specifically used by the Postal Service. It encodes ZIP Code information on letter mail for rapid and reliable sorting by bar code sorters. The PostNetBarCode can represent a five-digit ZIP code (32 bars), a nine-digit ZIP+4 code (52 bars), or an eleven-digit delivery print code (62 bars). Be aware that for the Post Office to recognize the bar code as being valid strict adherence to the guidelines must be met. Current standards require that the five digit zip plus four be used plus the two digit carrier route, which is most often obtained from the first two digits of the street address. It is recommended that the user check with the Post Office to obtain a copy of their current guidelines.

Example PostNetBarCode: 85210304119

2of5Bar Code



2of5Bar Code (interleaved 2 of 5) is a numbers-only bar code; in the Property Panel it is labeled 2of5BarCode. The symbol can be as long as necessary to store the encoded data. The code is a high-density code that can hold up to 18 digits per inch when printed using a 7.5 mil X dimension. "Interleaved" comes from the fact that the digit is encoded in the bars and the next digit is encoded in the spaces. There are five bars, two of which are wide and five spaces, two of which are wide.

Example 2of5BarCode: 2632534

Code39BarCode



Code39BarCode is an alphanumeric bar code that can encode decimal numbers, the uppercase alphabet, and the following special symbols "_", ".", "*", "\$", "/", " ", and "+". The characters are constructed using nine elements, five bars and four spaces. Of these nine elements, two of the bars and one of the spaces are wider than the rest. Wide

elements represent binary ones (1), and narrow elements represent binary zeros (0).

Example Code39: CODE 39

Code128Bar Code



Code128Bar Code is a very high-density alphanumeric Bar Code. The symbol will be as long as necessary to store the encoded data. It is designed to encode all 128 ASCII characters and will use the least amount of space for data of 6 characters or more of any 1-D symbology. Each data character is made up of 11 black or white modules. The stop character is made up of 13 modules. Three bars and three spaces are formed out of these 11 modules. Bar and spaces vary between 1 and 4 modules.

Example Code 128: Code 128

UPCBarCode



UPCBarCode (Universal Product Code) has a fixed length of 12-digits and can only encode numbers. UPC was designed for coding products. The format allows the symbol to be scanned with any package orientation. The check digit is calculated so there is no need to enter it when typing the value into the Text property.

Example UPCBarCode: 712345678935

EANBarCode



EANBarCode (European Article Numbering System) is identical to the UPC, except for the number of digits. EAN has a length of 13 digits - 10 numeric characters, and 2 "flag" characters that are usually country codes, and a check digit. This Bar Code is typically used for Non-U.S coding. The check digit is calculated for you so you do not need to enter it when typing the value into the Text property.

Example EANBarCode: 3847348484584

Advanced Components

In this Section:

- Use of the FontMaster and PageNumInit are explained.
- Details of some advanced components are given.
- Calc Component details are given.

FontMaster



Each Text component in a Report has a Font property. By setting this property, a specific font can be assigned to the component. In many cases it may be somewhat useful and necessary to set the same font properties of more than one object. Although this could be accomplished by selecting more than one component at a time, this method has a drawback. One has to keep track of which fonts have to be of the same typeface, size and style, which is not an easy task when there is more than one report. This is where the FontMaster comes into play. Apart from allowing the user to define a global font for various components, the FontMaster also allows the user to define standard fonts for different parts of the Report, like for instance the headers, body, and footers.

The FontMaster is a non-visual component (designated by the green color of the button). To use one, simply click on the button and then click anywhere on the Page Designer. Note that being a non-visual component, there will be no visual reference of it on the actual Page, and like other non-visual components, it can be accessed using the Project Tree.

As mentioned previously, the main purpose of the FontMaster is to set fonts. It has very few properties. As most other components, it has the DevLocked, Locked, Name and Tag properties. They also contain the most important one of all, which is the Font property.

To set the FontMaster property, use the Font property on the Property panel, click on the ellipse button to get to the Font Dialog box and select the font and size settings. Click OK when done.

Once the FontMaster component is set, linking it to a body of text is simple. On the Report select a Text/Memo component, then use the down arrow button on the FontMirror property in the Property Panel to choose a FontMaster link. Any component that has the FontMirror property set to the FontMaster will be affected by and change to the FontMaster's font property. This allows the user to change fonts on various Text components at once and at the same time keep things consistent across the Report (when required).

It is important to note that by setting the FontMirror property on a component, the Font property will be overridden by the Font property of the FontMaster. This means that if a text object has a Font setting of Font A, by assigning a FontMaster to it with Font B, the Text component will automatically be assigned Font B and it's Font property will be ignored. Another side effect of using the FontMaster is that the Font Toolbar is disabled when the FontMirror property is set for that component.

There can be more than one FontMaster per Page; however, it is good practice to usefully rename a FontMaster. The Project Tree image shown previously has three FontMaster components on the Page. Two of them begin with FM, for FontMaster, and they follow with the name and size of the font that they represent. This is one naming convention that can be used to make the name descriptive and useful to the user.

PageNumInit



PageNumInit is a non-visual component that allows the restart of page numbering within a Report. Using a PageNumInit is similar to other non-visual components. It is used when more advanced formatting is required.

An example of using PageNumInit is in a CustomerStatement Report of a checking account. Account statements that customers receive every month may vary in the number of pages. In the monthly statements, the first page can define the account summary page layout, the second can define the customer's credits/deposits, and the third the withdrawals and debits. The first two pages may only be one page, but if the account activity is high for the customer then a section like withdrawals could be several pages. If the user producing the report wants each section numbered individually, the summary and deposits would have the pages marked "1 of 1" for both. For Withdrawals, an active customer account could have three pages of withdrawals and debits. Since this is a different section in the statement, it needs to be marked "1 of 3", "2 of 3", and "3 of 3". Using this kind of multiple page numbering is where PageNumInit comes in handy.

There are a couple of steps that must be completed to use the component efficiently. First define the Report Pages as usual. Add a DataText component from the Report Toolbar to the Page, placing it where the Page requires a Page number to appear.

Once it is in place correctly, in the Property Panel of the DataText component, click on the ellipse button in the DataField property.

This will open up the DataText Editor. Click on the Report Variables down arrow button and select Relative Page and click on Insert Report Var button to use the variable on the Report.

In the DataText area, "Report.RelativePage" should appear. After the text type in the following: + ' of ' +

From the ReportVariables drop-down list select TotalPages (choose it and press the Insert Report Var button). This will add the remaining text "Report.TotalPages" to the Data Text edit box area. When finished click OK.

The Property Panel now displays the sentence that was entered using the Data Text Editor.

Expanding the DataText component, the text from the editor will also appear.

To initialize the Page to a desired Page number, select PageNumInit from the Project Tree. On the Property Panel, type the desired Page number into the InitValue property box.

For every Report definition created in a Project, the PageNumInit will allow each report definition to be numbered independently of another.

DataCycle



The DataCycle component is an invisible data-aware band that would be used to control iterating information from a DataView. In general, a DataCycle component would be used to "Loop" or "Cycle" through the detail records until a change occurs at the master record level.

For example, imagine a bill being generated for multiple customers and each customer has many different purchases associated with them. For this bill you need to have each person's purchases on their own billing page, which has all and only their own listing of purchases. The table design for this might look like the following tables, where there is a master table listing of all the unique customers, and each customer has their own purchasing table, with all their own purchases listed.

The bill being created will need to have each unique customer ID number printed on one page as well as the items purchased on that page. Thus, the report will have to cycle through the table containing all the purchases made before proceeding to printing the next customer and the associated purchased items.

The main table containing the list of customer ID's would be found in what is called the MasterDataView. In the Property Panel, this master table name would be chosen from the property called MasterDataView. Once that is chosen, we will need to tell Rave what key will link the two tables together. This key is called the DetailKey. The DetailKey, in our example, would be the Customer ID's. Next, the table that will be cycled through, which is the table containing purchases to be printed, will be the DataView table. So, in essence the MasterDataView (driving table) and the DataView (details table) are connected by the Detail Key (the common unique key in each table).

The DataCycle can also be limited, or sorted while the Report is being generated. For example, supposed you just wanted to create reports for the Customers in Arizona. To limit Report creation to just the Arizona customers, the MasterKey will have to be set.

Setting the MasterKey is done by clicking on the ellipse (...) button next to the Property in the Property Panel. The Data Text Editor will appear. In the Data View area, choose the "Selected" radio button and then select the appropriate MasterDataView table, which is the Master Table in our example, by using the drop-down menu. Once you have done that, next select the field that will be used as the filter of the Master Table, or MasterDataView table. Once selected, click on Insert Field that is just underneath the Data Field drop down menu. This will place the appropriate text in the Data Text area at the bottom of the Editor. Now in the Data Text box, type in "=AZ" to limit that field to just process the AZ customers. Click OK when done, and that will filter out non-Arizona customers.

DataMirror Section



The DataMirrorSection component is a section that will "Mirror" other "Sections" based on the contents of a "DataField". Mirroring Sections allow this component to be very flexible. Remember that Sections can contain any other component including graphics, regions, text, etc.

An example using this "DataMirrorSection" component is included in the RaveDemo project file. This example shows how to have a single report produce different envelope formats when sending to International or US addresses. The template for the International customers includes the country line and is centered on the envelope, while the US format does not have the country line and is offset to the right of center and lower on the envelope.

This example shows several techniques on how to use Sections and Mirrors. On page 2 there are three Sections. Section 3 has the common address lines used by both templates. Section 1 and 2 contain Mirrors to Section 3. The International Section has the additional country line added in its template. Note, that "remarks" have been added to the page 2 design and they are not part of any of the sections. The "US Template" and "International Template" are just plain Text remarks that assist in the purpose of each Section shown.

Once understanding how the Master Templates were done, go to Page 1, and select the "DataMirrorSection". Notice the "DataView" and "DataField" properties have been set to the field that will be used to select which Format Template to Mirror. To examine that logic, go to the "DataMirrors" property and click on the ellipsis to open the Data Mirror Editor. Select each "DataMirror" item and note the settings in the bottom portion of the dialog box. This example has two "templates", but feel free to add more to fit more complex reporting needs.

NOTE:

Normally one of the settings will be defined as the default. If a default is NOT defined and the field value does not match any of the other settings, then the format used will be the normal contents of the DataMirror Section component.

CalcOp



The CalcOp component is a non-visual component that allows an operation (defined by the Operator property) to be performed on two values from different sources

(Src#CalcVar, Src#DataField or Src#Value). The result can then be saved in a project parameter like CalcTotal (through the DestParam and DisplayFormat properties).

For example, there could be two DataText components that need to be calculated together. Like $A+B=C$, where A and B represent the two DataText component values and C represents the result. This is where the CalcOp component would be used.

To choose the DataText components in the above example, use the Src1X and Src2X properties, where X is either CalcVar (a calculation variable), DataField, or DataValue. These are the three types of value sources that can be used for calculation in a CalcOp component. Note that the Src (source) can only use ONE of the three available source types, and there can only be two sources, Src1 or Src2.

The three source types have many different values associated with them. For a CalcVar source, the drop down menu will list all the calculation variables available in that Page available for use. This calculation variable is just that, a variable for holding a value. This value can be from another CalcOp component or from some other calculation component. For the DataField source to get available values first designate what DataView the DataField will come from. In other words the DataField is a field in a table, which is the DataView. So, in order to choose a field, the DataView must first be chosen. For a Value source, typing in a numeric value is all that is needed to fill this property. Again, remember only one of these three sources can be assigned to a source.

Once the sources are chosen the operation to be used between them has to be chosen. To choose the desired operation, use the Operator property by using the drop down menu and making the appropriate choice. In the example, $A+B=C$, the operator would be "coAdd".

There may be times when a function needs to be performed on a value before it is processed with the second value. This is where the property SrcYFunction, where Y is 1 or 2 for Src1 or Src2, will become handy. With SrcYFunction, the value can be converted (like from hours to minutes), or have a trigonometric function performed on it (like take the sin of the value), or have other functions performed on the value (like take the square root of the value, or take the absolute value).

Once the two values are chosen, it is now time to deal with the result. In the example, $A+B=C$, C is the value of the result. The result can either be written out to a project parameter (most likely a DataText component) or just held as an intermediate value. To write out the value to a parameter, use the DestParam property to choose the desired parameter to write the result write to. If the result is going to be in intermediate result, there is nothing else to do to the component. Although, it is highly suggested to rename the component, by using the Name property, to easily recognize the component for future use. This is because, as in intermediate values, you will most likely use the components in another CalcOp component, and a good reference will help to easily create the next step in the calculation process.

After setting the values and setting the result destination, it may be necessary to format the result or to even perform one last function on the result. Using the DisplayType and DisplayFormat properties to format the result into any necessary format. The DisplayType has two options, DateTimeFormat and NumericFormat. DisplayFormat has many options that must be typed into the edit box. To find these options look at the Formatting Appendix in the back of this manual. Like the two values, A and B, in the example, $A+B=C$, the result C can also have a function performed on it before being written out to a parameter, or as a holding value. To select a function, use the ResultFunction property drop down menu.

The CalcOp components can also be chained together for more complex expressions using the Src#CalcVar properties, which can be set to other CalcOp or CalcTotal components. For example, to create the complex expression shown to the right, it will be necessary to break up the expression into simple 2 value steps.

To break up the complex expression, four new expressions will need to be created and saved as CalcOp components. So the resulting components, Z, Y, X, and W will be four separate intermediate CalcOp's. But, as you can see from the flow of the expression, Z and Y will have to be completed before X can be created, as it is dependent upon the two previous operations. And finally, W will be completed last, after X, to get the final correct value.

$$\frac{(A+B)*(C*D)}{0.80} = E$$

$$A+B=Z$$

$$C*D=Y$$

$$Z*Y=X$$

$$\frac{X}{0.80}=W = E$$

Just as it is important to do the calculations in order, it is important to make sure the components are in order in the Project Tree. When a report is executed, the report executes components down the Project Tree. For CalcOp components or any calculation component, this means that they need to be in the correct order. In the example above, if Z, Y, X, and W were in the Project Tree, Z would have to be first in the tree and W would have to be last in the tree. This would mean that at execution, Z would be processed first, then Y, and so on. It is also important to note that if Z is dependent upon any other components (like other CalcOp components or DataText components), those components must come first in the Tree before the Z component is processed. Making sure the components are in correct order is known as setting the print order. To move the components up and down within the Project Tree, use the Alignment Palette to change the print order of the components.

CalcController



CalcController is a non-visual component that acts as a controller, along with DataBands, for CalcText and CalcTotal components through their Controller properties. When the controller component is printed, it signals all calculation components that it controls to perform their summation operation. This allows totals to be performed on group bands, detail bands or whole pages depending upon the location of the CalcController component. Another feature of the CalcController component is its ability to initialize a CalcText or CalcTotal component to a specific value (through the InitCalcVar, InitDataField and InitValue properties). A CalcController component will only initialize values if it is used in the Initializer property of CalcText or CalcTotal property.

CalcTotal



The "CalcTotal" is a non-visual version of the CalcText component. When this component is "printed", its value is typically stored in a project parameter (defined by the DestParam property) and formatted according to the DisplayFormat property. This can be useful when performing totaling calculations that will be used with other calculations before being printed. Leave DestParam blank if the value of CalcTotal will only be used by other calculation components such as CalcOp.

Exercise: Using Font Master

1. Create a new page in the report. Select New Report Page from the Project window.
2. Go to the Project Tree and select the Page.
3. In the Property Panel, type in "FontMaster" in the Name property, while the Page is still selected.
4. Go to the Standard Toolbar. Find the FontMaster component and click on the button. Then click on the Page.
5. Look at the Project Tree Panel and notice that there is a new component underneath the Page called FontMaster1. But, when looking at the Page, there is no visual component. FontMaster is a non-visual component.

6. Make sure the FontMaster component is selected by clicking on it in the Project Tree. The component will be highlighted in the Project Tree to indicate that it is selected.
7. While the FontMaster component is selected, look at the Property Panel. In the Font Property, click on the ellipse button (the button with three dots).
8. The Font Dialog will appear after clicking the ellipse button. Use the scroll buttons and check mark boxes to make your desired selections. For this first FontMaster component, select Arial Font, Bold Font style, 14 size, and Underline in Effects.
9. Look at the Property Panel, with the component still selected. The Font Property will reflect the selection made in the Font Dialog.
10. Now with the Font Property still selected, scroll to the Name Property and put "FMArial14BldUndrln" in the Name property edit box.
11. Next look at the Project Tree and notice the name change. It becomes very useful and helpful to rename the components in the Project Tree in order to distinguish each component from each other. This is why we demonstrate renaming of the FontMaster component and the Page component.
12. Now, complete steps 4 to 11 three times using the following names and settings. FMArial16ItlcUndrln: Arial Font, size 16, Italic, an Underlined. FMTimesNwRmn12: Times New Roman Font, and size 12. FMCourier12: Courier Font, and size 12. To make understanding the renaming of many components, sometimes it is helpful to use a specific naming convention. In these examples we use the following naming convention for the FontMaster component names:
13. Next, drop down five Text components on the page, as well as one Memo component.
14. We will pretend to write a "letter" using components dropped on the Page.
15. Use one Text component to use for a date holder, and use the other four Text components to make an address label. The Memo component can be used for the body of the message. So, fill your components appropriately. For more information on how to fill the components with your own text, see the Property Descriptions Listing in the appendix, or see the chapter on Standard Components. Feel free to use the alignment tools to get the "letter" components to align correctly. Also, after some of the following steps it may be necessary to resize the text components to display all the text correctly.
16. Next select the Text component that has the date. While selected, change the FontMirror Property to FMCourier12 by using the drop down menu. Notice that the font of the Text component changed to the pre-set font settings of the FMCourier12.
17. Next select the four address Text components. The Property Panel will show the designation that this is a multiple selection of Text components. In the FontMirror property, choose FMTimesNwRmn12.
18. Select the Memo component, which contains the body of the "letter". While it is selected, choose FMArial16ItlcUndrln in the FontMirror drop down menu.
19. This ends mirroring of fonts to components on the Page. One last thing to cover is what to do when you change your mind about the fonts?
20. Select all the address Text components. Change the FontMirror property to FMArial14BldUndrln. Now, this would be a way to change many Text/Memo components from one font setting to another, without changing each Font property in each component.
21. Sometimes there may be many components linked to one FontMirror component, because they all relate to one specific area of the report. In this example, we can assume that all address headers would be linked to the FMArial14Bldunderln Font Mirror property. So, to change this we would have to redo the one FontMirror property.
22. Select the FMArial14BldUnderln FontMirror component in the Project Tree.
23. In the Property Panel, while the component is still selected, click on the Font property ellipse button.
24. Change the Font Dialog properties to the following: Times New Roman, size 10 font, with no other Font Effects.

25. In the Name property of the FontMirror property, put the following name:
FMTimesNwRmn10.

Exercise: Setting up PageNumInit for Page Numbering

1. First create a new Page, or use what is left from the previous example, as we will do here. If this is a new Page, drop some Text components on the Page. These are just filler components in this example.
2. Go to the Report Toolbar, and find the Data Text component. Click on the Data Text component and then click on the Page.
3. Place the DataText component in an area where a Page number might appear. For example, place the DataText component at the bottom of the Page.
4. Make sure the DataText component is placed correctly, and make sure it is still selected. Check for selection by the visual appearance of the green pip surrounding the component, or by the component being highlighted in the Project Tree.
5. While the DataText component is selected, go to the Property Panel and look for the DataField property. Click on the ellipse button.
6. The Data Text Editor will next appear. Look for the *"Report Variables"* section in the Editor.
7. There will be an arrow at the end of the Report Variables selection window. Click on it to see all the selections available. Use the scroll bar to move up and down through the menu.
8. In the Report Variables section, look for *"Relative Page"*. Click on it to select.
9. Next click on the Insert Report Var button to the right of the Report Variables section. This will place the appropriate text into the Data Text area at the bottom of the Data Text Editor.
10. Go to the Data Text area and type in the following: + 'of' + . This will include spaces before and after each symbol.
11. Now return to the Report Variables drop-down list. Scroll to look for TotalPages. Select it when you find it.
12. Click the Insert Report Var button, to get the remaining text into the Data Text area. "Report.TotalPages" will be added to the Data Text area.
13. Click OK when done. Your DataText area will look like the following.
14. After clicking OK, you will be returned to the Designer and to the Property Panel. If you expand out the Property Panel and look at the DataField property, you will see the results of the DataText are from the Data Text Editor.
15. On the Page, the DataText component will probably look like the DataText component labeled A below. If you expand the borders of the DataText component (as seen in B), you can see that the results of the Data Text Editor appear. There is really no need to expand the component; it is just for you to see in this exercise.
16. There is one more thing left to do to complete the page numbering, we need to initialize the Page to the desired page number. So, go to the Project Tree and find the PageNumInit component. Select the component when you find it.
17. While PageNumInit component is still selected, go to the Property Panel and find the InitValue property.
18. In the InitValue property, type in 1. This will mean that our pages will begin with page 1.
19. That is it. That is all that is needed to setup the PageNumInit. Note that in one project, with this feature, each Report can be number independently of each other.

