



Enterprise Reporting Solution

# DataBlock Designer Guide

*Product Version 6.4*

*Last Updated 12/12/2019*

# Trademark, Publishing Statement, and Copyright Notice

---

© 1998-2019 Evisions, Inc. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. No part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Evisions, Inc.

The information contained herein is subject to change without notice and is not warranted to be error-free. Product features referenced herein for a period of time may not match product contents. Evisions, Inc. does not warrant that the functions contained in the software will meet your requirements or that the operation of the software will be uninterrupted or error free. Evisions, Inc. reserves the right to make changes and/or improvements in the software without notice at any time.

This software and documentation may provide access to or information on content, products, and services from third parties. Evisions, Inc. and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Evisions, Inc. and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services. Evisions, Inc. does not endorse the content or developer of any products or web sites mentioned.

All information in this guide is designed for instructional purposes only. Evisions, Inc. makes no guarantees regarding the accuracy or performance of any techniques used in this guide. Software configurations and environments may vary, and some techniques used in this guide may not operate efficiently under all configurations. This guide may contain examples of various technologies or products, which are the sole property and responsibility of their creators.

Trademarks are the property of the respective owners for any products mentioned herein.

# Table of Contents

---

<b>Argos Introduction</b> .....	<b>7</b>
Evisions Support Site .....	7
In Product Help .....	7
Sample Database .....	8
<b>Logging In</b> .....	<b>9</b>
Logging In to the eLauncher .....	9
Launching Application Clients .....	10
Logging In to the Application Clients .....	15
Server Selection .....	16
<b>Navigation</b> .....	<b>17</b>
Action Area .....	17
Navigation Area .....	17
Explorer view .....	17
Shortcuts View .....	19
<b>Definitions</b> .....	<b>20</b>
DataBlock .....	20
OLAP Data Cube .....	23
Charting .....	23
<b>The DataBlock Designer</b> .....	<b>24</b>
Object Toolbar .....	24
The Alignment Toolbar .....	27
Alignment of a single object .....	27
Alignment of multiple objects .....	27
<b>Building a Form</b> .....	<b>28</b>
Introduction .....	28
Add DataBlock icon to Explorer .....	28
Select a database connection .....	28
Add a Description, Author, and Notes .....	28
Launch the Argos DataBlock Designer .....	29
Add the Graphics Object .....	31
Adjust Object Properties .....	32
Add the Date Objects and Labels .....	33
Align the objects .....	34
Add Employee Name Label and Shape Object .....	34
Create the object for selecting Employee Name .....	35

<b>Building a Query</b> .....	<b>39</b>
Form Query .....	39
Create the multi-column list box to contain the query results .....	40
Identify the fields to appear in the report .....	43
Create a calculated field to determine sale amount .....	44
Use input selections to limit the query .....	45
Use of button to control the execution of the query .....	48
Summary .....	50
Report Query .....	51
Introduction .....	51
Create a New Report .....	51
Copy/Paste the Existing Query .....	51
<b>Advanced Form Objects and Properties</b> .....	<b>53</b>
Data Aware Property Introduction .....	53
Add the Data Aware Object .....	53
Set Data Aware property for edit box .....	54
Execute the Report .....	55
Summary .....	55
Using Multiple Forms in a DataBlock .....	56
Introduction .....	56
Create the DataBlock .....	57
Add the Form Names .....	57
Create the Main Form .....	58
Edit Properties of the Buttons .....	58
Create the form/query associated with each button .....	59
Summary .....	60
Charting with multiple series .....	61
Introduction .....	61
Create the Chart Object .....	62
Name the first series .....	63
Creating the new dataset for the first series .....	64
Select the Chart Types, labels, and other options .....	65
Create the next series for fy2007 .....	66
Summary .....	68
Creating a Filtered OLAP Data Cube .....	68
Introduction .....	68
Create the OLAP Object in the DataBlock Designer .....	69
Build the Fact Table (the query) .....	70
Identify the Measures .....	72
Identify the Dimensions .....	73
Commit and Test .....	75
Drilling through Reports to Obtain Greater Detail .....	76
Summary .....	79
<b>Advanced Query Techniques</b> .....	<b>80</b>
Summing and Grouping .....	80

Creating the Form .....	80
GROUP BY and SUM fields .....	82
HAVING Tab .....	82
Results .....	84
Free Type Report Queries .....	85
Entering the Free Type Query .....	85
Validating the Query .....	86
Apply Field Security .....	87
Filter and Sort the Report Query .....	88
Adding Special Characters for Filtering Reports .....	89
Adding Special Characters for Sorting Reports .....	90
Edit Query Properties .....	90
Summary .....	93
Scalar Subqueries .....	93
Creating the Main Query .....	94
Creating the Subquery .....	94
Results .....	94
Correlated Subqueries .....	95
Creating the Main Query .....	95
Creating the Subquery .....	96
Placing the Subquery in the Main Query .....	98
Results .....	100
Non-Correlated Subqueries .....	100
Creating the Main Query .....	100
Creating the Subquery .....	101
Placing the Subquery in the Main Query .....	101
Results .....	102
Inline View Subquery .....	102
Creating the Main Query .....	102
Creating the Subquery .....	103
Placing the Subquery in the Main Query .....	103
Results .....	105
Unions .....	105
Creating the Main Query .....	105
Creating the Unions .....	106
Results .....	109
<b>Scheduling Reports .....</b>	<b>110</b>
Editing Schedules .....	110
General .....	111
Schedule .....	112
Tasks .....	114
Events .....	115
API .....	116
Copying Schedules .....	117
Execute the Report Task .....	117
Process and Save Task .....	118
The Bursting Task .....	120

Copy File Task .....	122
Delete File Task .....	125
FTP File Task .....	126
Send an Email Task .....	127
The Save Execution State Task .....	130
Managing Schedules .....	132
<b>Data Dictionary .....</b>	<b>133</b>
Introduction .....	133
Managing Data Dictionaries .....	133
Importing and Exporting Data Dictionaries .....	134
Using the Data Dictionary in Argos .....	134
Adding Aliases to the Dictionary from within Argos .....	135
Adding Joins from within Argos .....	136
Alias and Join Info Applied to Query .....	136
<b>Library of Objects .....</b>	<b>137</b>
Library Manager .....	137
Types of Objects in the Library .....	138
Using the Library .....	138
Adding Styles to Banded Reports .....	139
Report Templates .....	139
Styles .....	139
Report Preview .....	140
Adding Styles to OLAP Data Cubes .....	140
<b>Security .....</b>	<b>141</b>
Object Level Security .....	141
Field Level Security .....	143
<b>The CO-OP User Community .....</b>	<b>145</b>
Searching for DataBlocks and Templates .....	146
Submitting Files .....	147
<b>Argos Support Resources .....</b>	<b>148</b>
In-Product Help .....	148
User Guides .....	148
Training .....	148
Additional Resources .....	148
<b>Getting Help .....</b>	<b>149</b>

# Argos Introduction

---

Argos is a powerful reporting solution designed for everyone from novice users to the most seasoned technical experts. For ease of use, Argos users are divided into three distinct types:

**DataBlock Designers:** Argos “power users” who create DataBlocks.

**Report Writers:** Intermediate users who use DataBlocks to build a variety of reports.

**Report Viewers:** Casual users who are able to run reports, then save and distribute the output in a variety of useful formats.

Each user type has a corresponding guide associated with it. This guide is intended for DataBlock Designers.

A prerequisite to reading this guide is reading the Argos Report Viewer Guide and the Argos Report Writer Guide. The Report Viewer Guide describes how to log into Argos, how to navigate through the Argos User Interface, and how to execute reports. Therefore, this basic information need not be repeated in this guide.

This guide assumes that the reader is familiar with database concepts and has some familiarity with SQL.

Once you have completed this guide, you should be able to:

- Create DataBlock forms
- Build Database queries
- Design a chart
- Design an OLAP cube
- Schedule a report
- Publish a DataBlock to the Evisions CO-OP
- Create Security at the Report and Item Level
- Utilize the Library of Objects and Data Dictionary

This guide is not intended to be a comprehensive reference guide that covers each and every option within Argos. The intent is to provide a sufficient number of examples to aid a new Argos user to get started quickly. Additional information on each feature is available through In-Product Help.

## Evisions Support Site

---

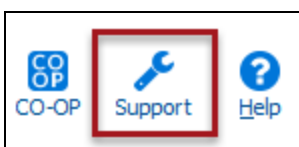
The easiest way to get to the Evisions Support site is to access it through Argos. Under the Help menu is a link to the Support page. All the technical information available for download (documentation, tutorials, training videos, DataBlock library in the CO-OP Share) is found under the Support page.

Evisions can also provide more in-depth and even customized training via our Professional Services department. Visit the Consulting Services page on the Evisions web site at <https://www.evisions.com/services/consulting/>.

## In Product Help

---

In addition to the Support site is In-Product Help. You can access In-Product Help a few different ways. There is a link under the Help menu to Argos Help. There is a button on the toolbar, and you can also use your F1 key. Most screens within Argos also have a link to Argos Help as well.



## Sample Database

---

The examples within this guide are based upon an MS Access database that was created to assist you with becoming familiar with Argos. You can [download the sample database and DataBlocks](#) from the Documentation and Software section of the Evisions website.

The name of the file is "Sample.zip" and after unzipping the file name will be "Sample.accdb". Six DataBlocks are included in the zip file that are used by the Argos Report Writer Guide.

You can use the database to follow along with the examples in this guide. We have also provided a [Sample Database Description](#) that explains the structure of the database and provides instructions for creating the data connection to the sample database.

This document also describes how to create the data connection to the sample database.

Contact your MAPS Administrator to install the database and create the data connection.



# Logging In

---

## Logging In to the eLauncher

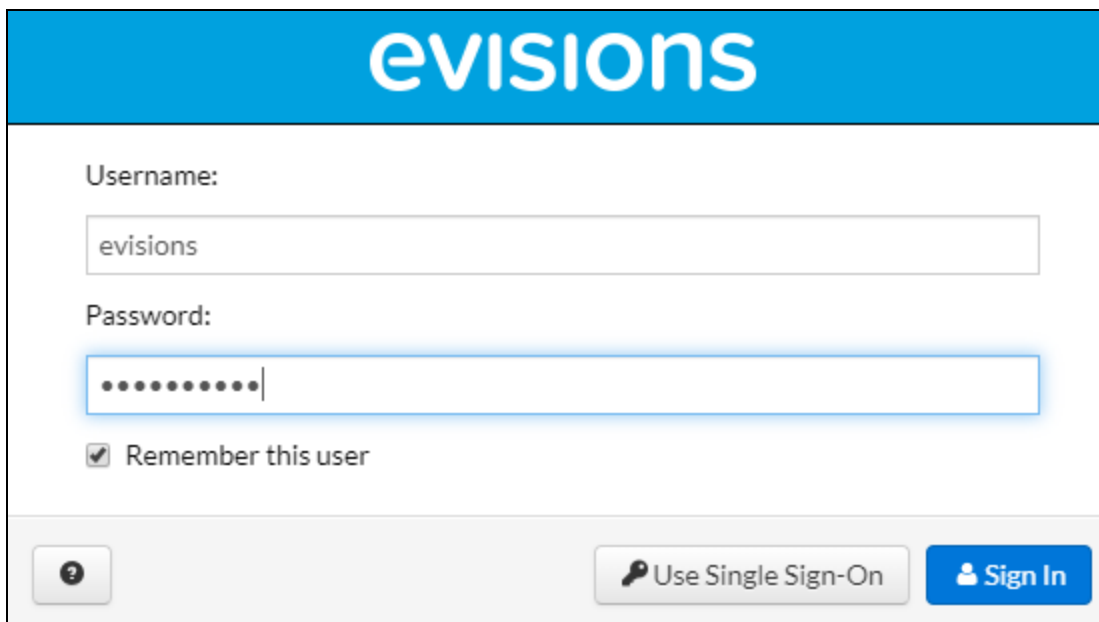
---

The MAPS eLauncher provides a single location from which you can launch any of the MAPS applications, access online training and support resources, or see the new features in the latest releases.

In your web browser, navigate to the URL provided by your MAPS administrator. If you are prompted to log in at this time, do one of the following:

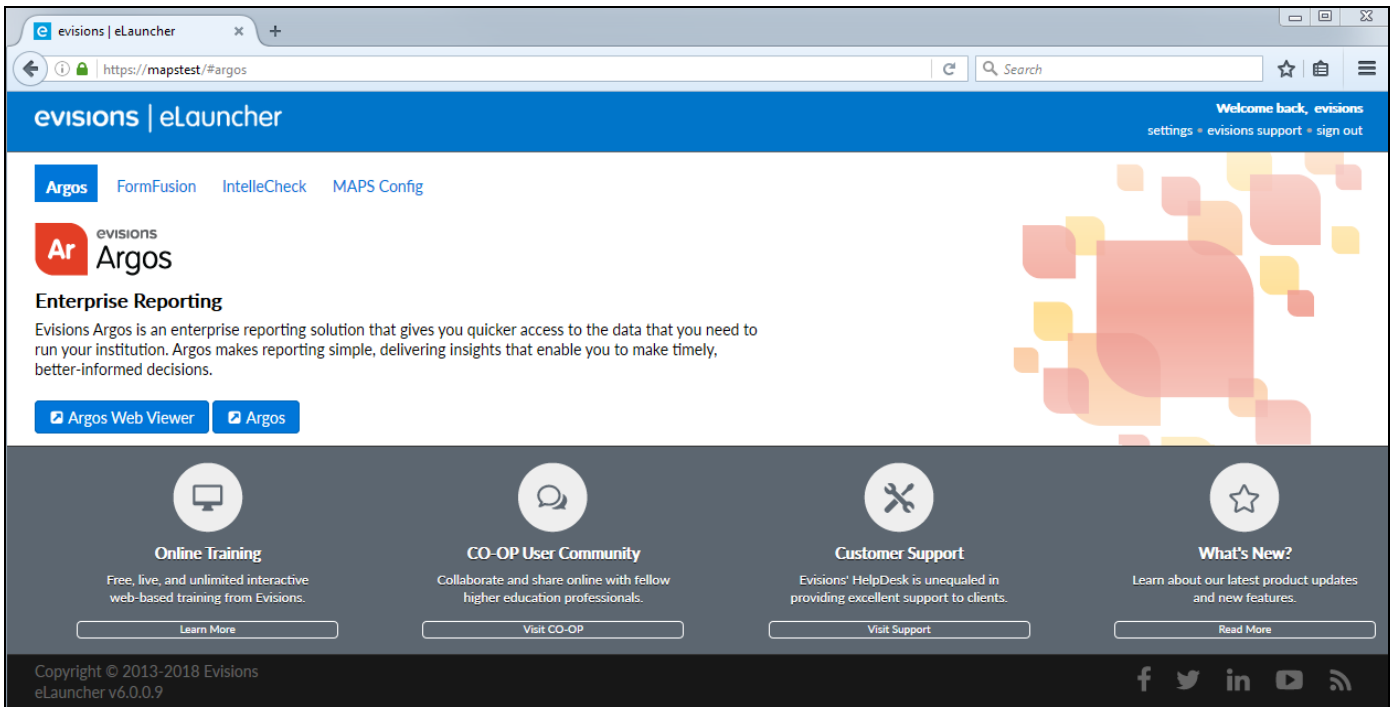
- For HTTPS, enter the username and password provided by your MAPS administrator.
- Enter your LDAP credentials if your institution uses LDAP for single sign-on.

If your institution has single sign-on configured for use with a SAML server, you do not need to enter your credentials here. Select the **Single Sign-On** button and enter your username and password on your institution's SSO page. (The Single Sign-On button is only displayed when SAML SSO is available.)



The screenshot shows the login interface for 'evisions'. It features a blue header with the 'evisions' logo. Below the header, there is a 'Username:' label followed by a text input field containing 'evisions'. Below that is a 'Password:' label followed by a password input field with masked characters. A 'Remember this user' checkbox is checked. At the bottom, there is a help icon, a 'Use Single Sign-On' button, and a blue 'Sign In' button.

The MAPS eLauncher displays after your credentials are validated.

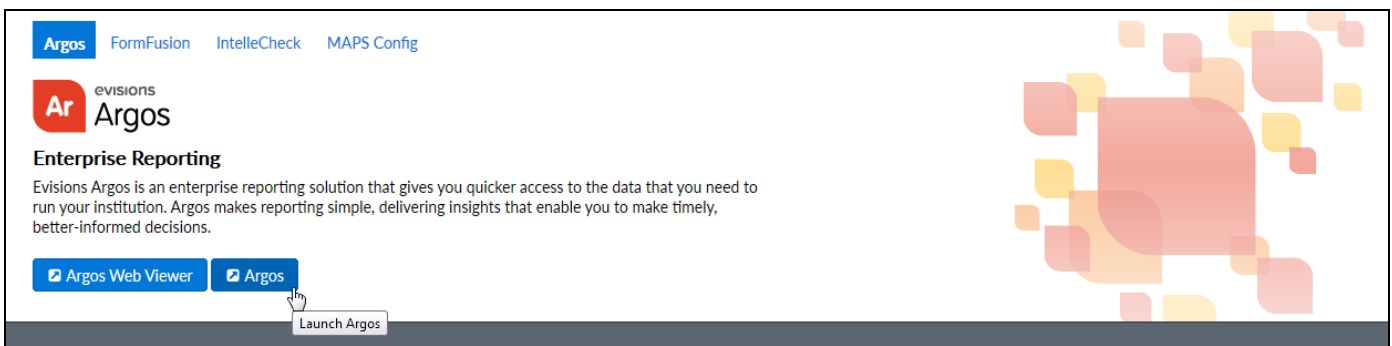


### Login Issues

If you cannot log in, it may be due to local or network firewall settings. You may need to configure your firewall to allow access on the selected port.

## Launching Application Clients

To launch an application client, navigate to the product you wish to use and then select the launch button for that product.

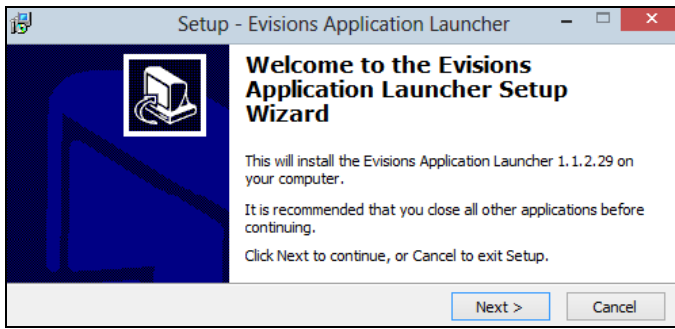


If you are prompted to install the **Evisions Application Launcher (EAL)**, follow the prompts on your screen to continue. Expand the link below, for more information.

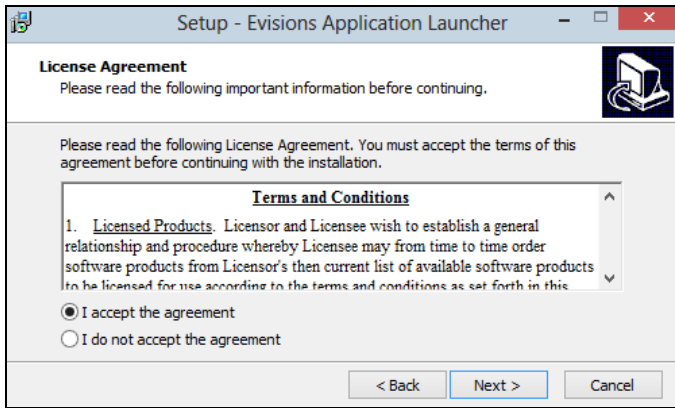
**Note:** If you do not have permission to install applications on your computer, you may need to consult with your IT department for assistance.

### Installing the Evisions Application Launcher

Select the link to download the **setup.exe** file. When it has finished downloading, open the file to run the installer.



Follow the prompts on the screen. Select **Next** to continue.

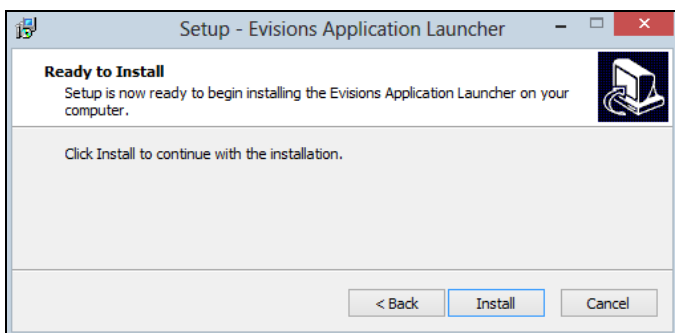


Review and accept the license agreement, then select **Next**.

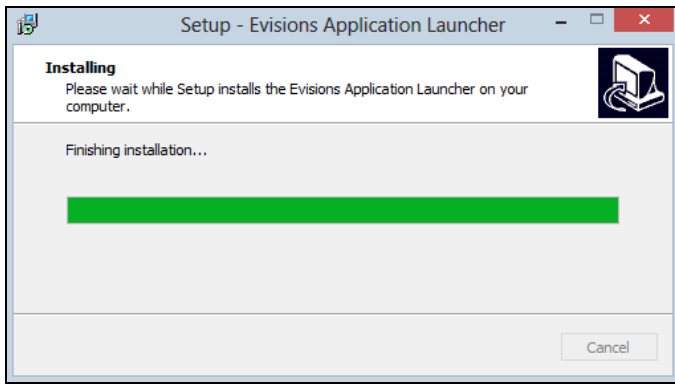
### ***Microsoft .NET Framework is Required***

If your computer does not have the Microsoft .NET framework already installed, you will be prompted to install it as part of the EAL setup.

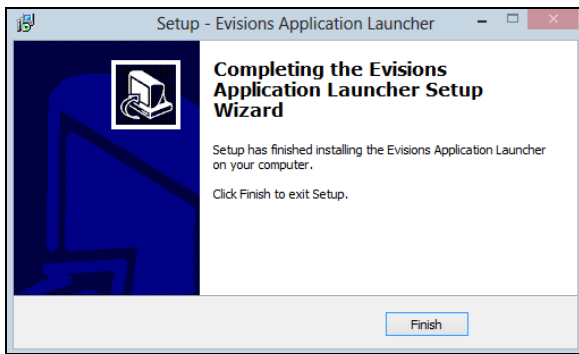
Review and accept its license agreement, then select **Install** to proceed.



Select **Install** to continue. If you are installing the Microsoft .NET Framework at the same time, it will install alongside the Evisions Application Launcher.

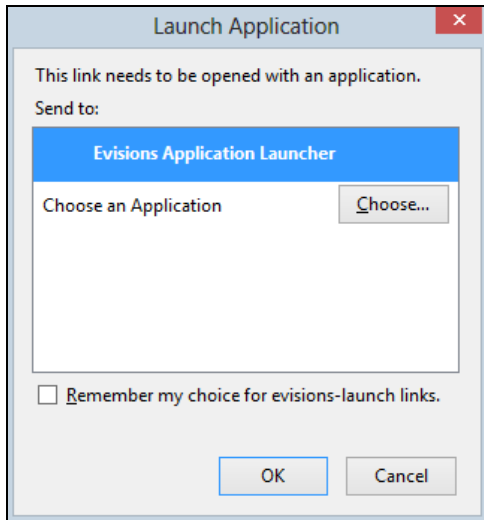


Once the installation is complete, select **Finish** to close the installer.

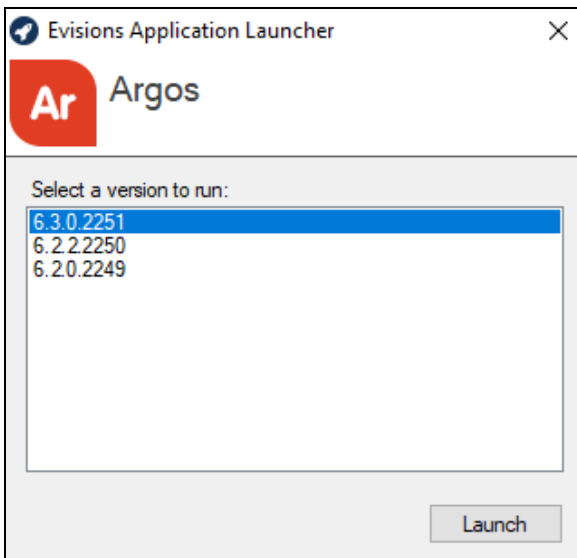


When you launch an application (Argos, for example) you may see a prompt informing you that the browser is trying to run an external program, and asking whether you want to proceed. Select **OK** or **Allow** to allow the launcher to run.

Note that the prompt is different in different browsers.



If there are multiple versions of the application available, choose the version to use, then select **Launch**.



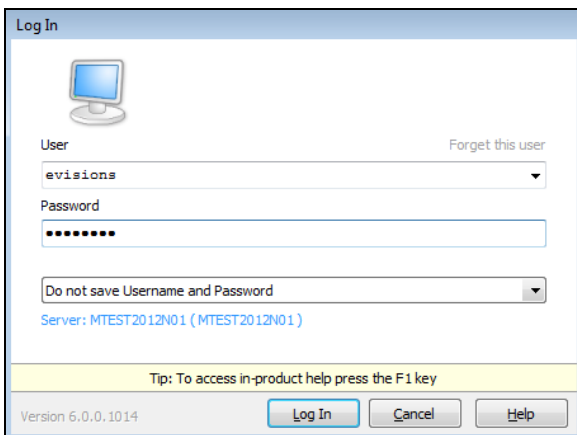
If the application or application version you selected is not already installed on your computer, it will download. This may take some time, especially over slow connections.

## Logging In to the Application Clients

If you have not yet logged in to the application, the main login screen displays. The screen for each MAPS application is similar to the one below. You must enter your credentials to access the application under the following circumstances:

- You did not use the HTTPS version of the eLauncher
- You did not log in through a single sign-on server
- You are logging in to MAPS as an administrator
- You are logging in to IntelCheck (or accessing a secured area)

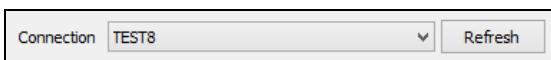
Otherwise, you do not need to enter your credentials again unless you are accessing a different secured program or feature, or need to reconnect.



There are several options you can configure on this screen:

- **Forget this user** - removes this user from the list of saved usernames in the dropdown list.
- **Do not save username and password** - do not save either the username or the password that you enter.
- **Save username** - adds this user to the list of saved usernames in the dropdown list.
- **Save username and password (if allowed by server)** - saves the password for this username, if password saving is enabled in MAPS. **Note:** for security reasons, you cannot save passwords when logging in to the MAPS Config application.
- **Server** - Click on the server name for additional options to change the server you are logging in to or the port number for that server.

IntelCheck users have an additional option to select the database connection prior to login:



Your data connection permissions (established by the MAPS administrator) determine the connections displayed in the list. If you do not see the connection you want to use, enter your username and password and select **Refresh**. Your MAPS administrator can provide further assistance with data connection permissions.

Once you have entered all information, select **Log In** to launch the application.

## Failed Login Attempts

Your MAPS administrator determines how failed login attempts are handled, based on your institution's password policy. If allowed, you may be able to log in after waiting a period of time specified by the administrator (often an hour).

## Server Selection

Select the **Server** link on the Login dialog to select a different server, if necessary. Note that the Servers section looks slightly different depending on the product and version you are running:

Version 4.3.0.588

Version 6.0.0.1014

The following actions are available from this area:

- **Green "+" sign / Add button** allows you to add a server to the list of available servers.
- **Red "X" / Delete button** deletes the selected server.
- **Server Address** enter the name/address of the server.
- **Use the default port** check this box to use the default port for the application (27467) or uncheck to enter a specific port number.
- **Port** allows you to enter a different port number, if your MAPS administrator has instructed you to do so.



# Navigation

---

Argos has been designed with an intelligent interface that knows your user type and configures menus and buttons to show only those actions permitted.

Menus across the top allow you to take simple actions such as logging in to the product, finding items in the Explorer, customizing your Argos toolbars and changing your password. You can also access the integrated Help system or visit the Evisions web site that has many helpful resources available. The most common actions are replicated as buttons just beneath the menus.

At the very bottom of the screen, the status bar tells you what server you are logged in to, your username and user type.

Between the top and bottom toolbars is the Argos work area. The work area is broken into two halves. The left half contains the Navigation area while the right half contains the Action area.

## Action Area

---

The right hand side of the screen will have buttons for any actions you can take on a selected object. The buttons that show will depend on the type of object you select in the Navigation area.












## Navigation Area




---

This area contains the objects that you can perform actions on. There are three different views you can use for the Navigation area by clicking the desired tab (Explorer or Shortcuts). Each of these views is described below.

### Explorer view

The Explorer is the default view for the Navigation area, in which a simple menu of available folders and objects is displayed. Argos objects that can be found in the Explorer include:

<b>Icon</b>	<b>Explorer Object</b>	<b>Description</b>
	<b>Folders</b>	<i>Contains objects, including other folders.</i>
	<b>DataBlocks</b>	<i>The "parent" object for one or more reports.</i>
	<b>Dashboards</b>	<i>Display-only reports for quick reference.</i>
	<b>System Created Dashboard</b>	<i>The dashboard that is automatically created by Argos for every new DataBlock. The system dashboard is always called "Dashboard" and cannot be renamed or deleted.</i>
	<b>User Created Dashboard</b>	<i>This dashboard is created by users. Dashboards can be used to view information quickly without having to run a report.</i>
	<b>CSV Report</b>	<i>A comma-separated values report.</i>
	<b>Banded Report</b>	<i>A fully-formatted report.</i>
	<b>Extract Report</b>	<i>A text report that meets pre-defined specifications.</i>
	<b>Delimited Extract Report</b>	<i>An extract report that uses a user defined delimiter (tab, commas, and spaces are common delimiters).</i>
	<b>Fixed Width Extract Report</b>	<i>An extract report the width of each field is defined by the user.</i>
	<b>XML Extract Report</b>	<i>An extract report whose output is an XML file.</i>

<b>Icon</b>	<b>Explorer Object</b>	<b>Description</b>
	<b>Private Report (Banded)</b>	<b>A report that only appears for the report creator and the administrator. The eye icon on top of the regular report icon indicates that it is a private report.</b>
	<b>Schedule</b>	<b>Reports may be scheduled to run automatically.</b>
	<b>Trash Bin</b>	<b>Contains items deleted from the Explorer Tree.</b>

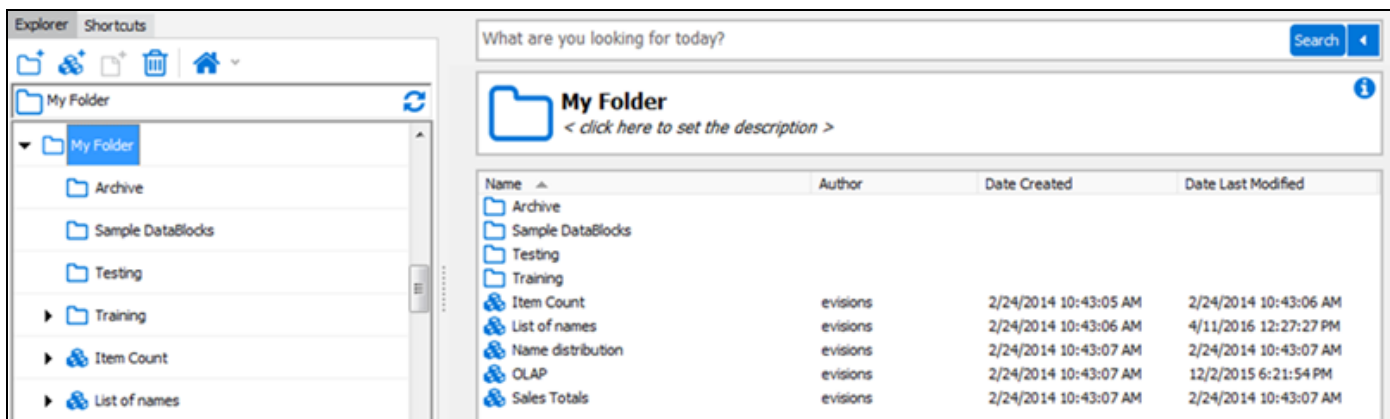
**Note:** Report Viewers do not have privileges to add, modify, or delete objects within the Explorer tree.

**Are my reports secure?** You may be wondering about the security of reports in Argos. Each object can be assigned to groups or individuals as needed. Unauthorized users would not be able to see objects they lack permissions for. It is even possible to have security all the way down to individual fields in a CSV or Banded report.

**Is my data secure?** All data transmitted from the server to Argos is “point-to-point” encrypted, meaning that anyone other than the intended user would see only gibberish. Once a report is created, care should be taken with resulting file(s) to ensure data security.

Some objects can be flagged as private. Private objects will not show up in the Explorer for users other than the creator and the administrator. These objects will have the “private eye” icon like the sample private Banded report icon on the list above.

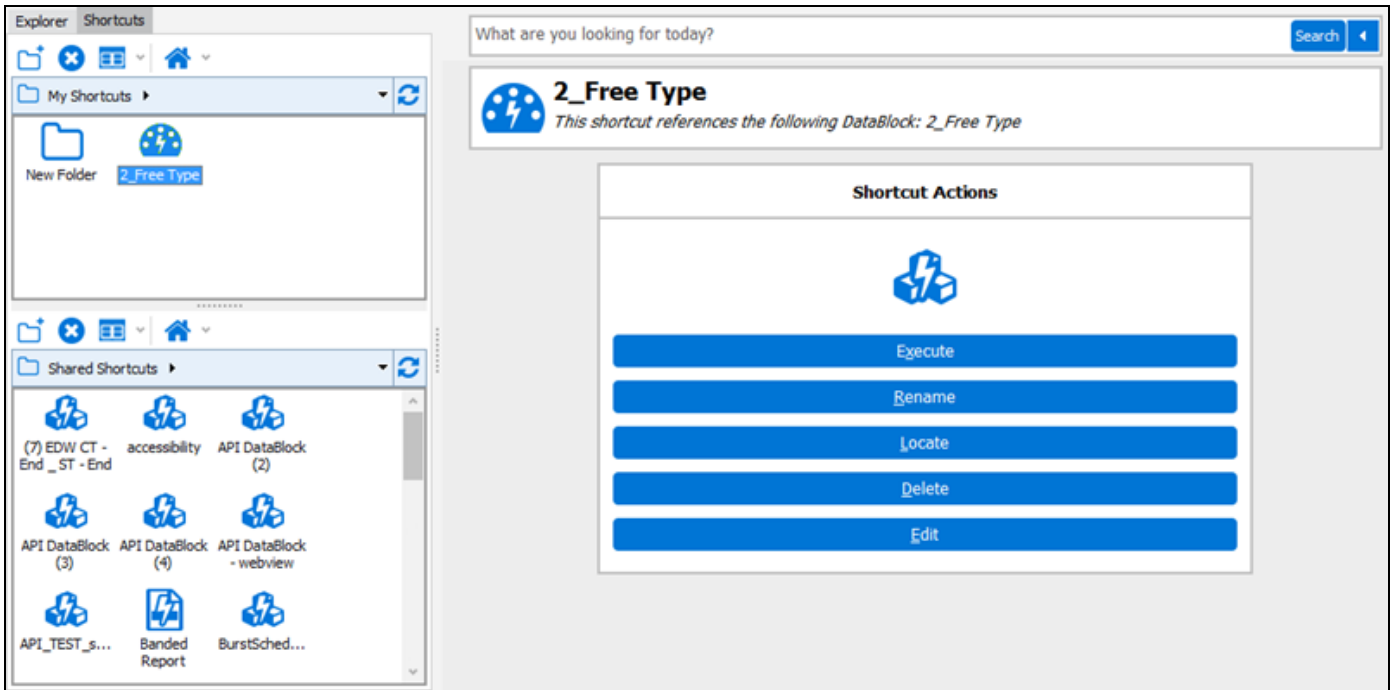
Within the Explorer tree, reports always reside beneath a DataBlock parent. A DataBlock can have many “child” reports. Any object that has child objects will have a “+” next to it. Simply click the “+” to expand the object to view its children.



## Shortcuts View

If you use a DataBlock or report frequently, you can right-click on it and choose “Add to Shortcuts”. Adding objects to the Shortcuts makes it easier to find what you need. Click on the **Shortcuts** tab to view shortcuts.

This view can be very convenient as it shows only your available shortcuts. You can even rename a shortcut to something other than the original name. To find the original object in the Explorer view, right-click a shortcut and choose “Locate” (see figure on the right). Deleting or renaming a shortcut has no effect on the original object.



# Definitions

---

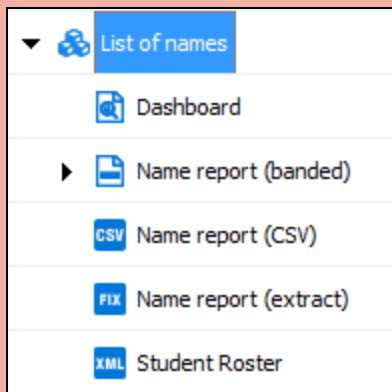
## DataBlock

---

The DataBlock is the foundation from which all reports are created and contains **Forms** and **Queries**. Only users with DataBlock Designer privileges can create DataBlocks.

The **Queries** obtain data from a database. When results of queries are displayed on a form as shown in the figure below, this is called a Dashboard. In the case of CSV, Banded, and Extract Reports, the results of the queries become input to the report design with the results displayed or stored elsewhere. Within the Argos Explorer tree, reports exist as "child" objects of a DataBlock.

### Reports as children of a DataBlock:



All report types residing under the same DataBlock use the same form. Reports are "children" of the DataBlock.

The **Form** created by the DataBlock Designer using the Argos DataBlock Designer, is used for two purposes:

- To obtain input selections from the user executing the report. The input selections can be passed to the queries to limit the results. For example, in the figure below a report is created which lists sales information for selected employees. The input selections are employee names and date range which limit the extent of the query.
- To display the results of the report. Dashboards can display results on the form. The figure below shows the query results displayed on the same form as the input selections.

Whenever a user executes a report, the form designed as part of a DataBlock will be displayed as shown below. **The input selections made by the user become query parameters.**

The screenshot shows the Argos DataBlock Designer interface. At the top left is the Argos logo. Below it is a title box labeled 'Employee Name'. To the right of the title are two date pickers: 'Start Date' set to 01/03/2005 and 'End Date' set to 3/31/2006. Below the date pickers is a 'Go' button. On the left side, there is a list box containing employee names with columns for 'last\_name' and 'first\_name'. The list shows 12 items, with 1 selected. Below the list box is a data table with 5 columns: 'last\_name', 'first\_name', 'sale\_date', 'product\_name', and 'sale\_amount'. The table contains 27 items. The data in the table is as follows:

last_name	first_name	sale_date	product_name	sale_amount
Washington	Mark	1/8/2005	HP dv7-3080us Notebook	12990
Washington	Mark	1/8/2005	Norton Internet Security 2010	80
Washington	Mark	1/8/2005	HP Scanjet 8300 Profession Image Scanner	998
Washington	Mark	1/8/2005	HP Pavillion Elite Desktop PC	1058
Washington	Mark	1/8/2005	HP 2509m 25" Diagonal Full HD Widescree...	978
Washington	Mark	7/15/2005	Lenovo SL510-28472MU	4543
Washington	Mark	7/15/2005	Canon MP980 All-in-one	1813
Washington	Mark	8/31/2005	HP dv7-3080us Notebook	1299
Washington	Mark	1/8/2005	HP dv7-3080us Notebook	1299

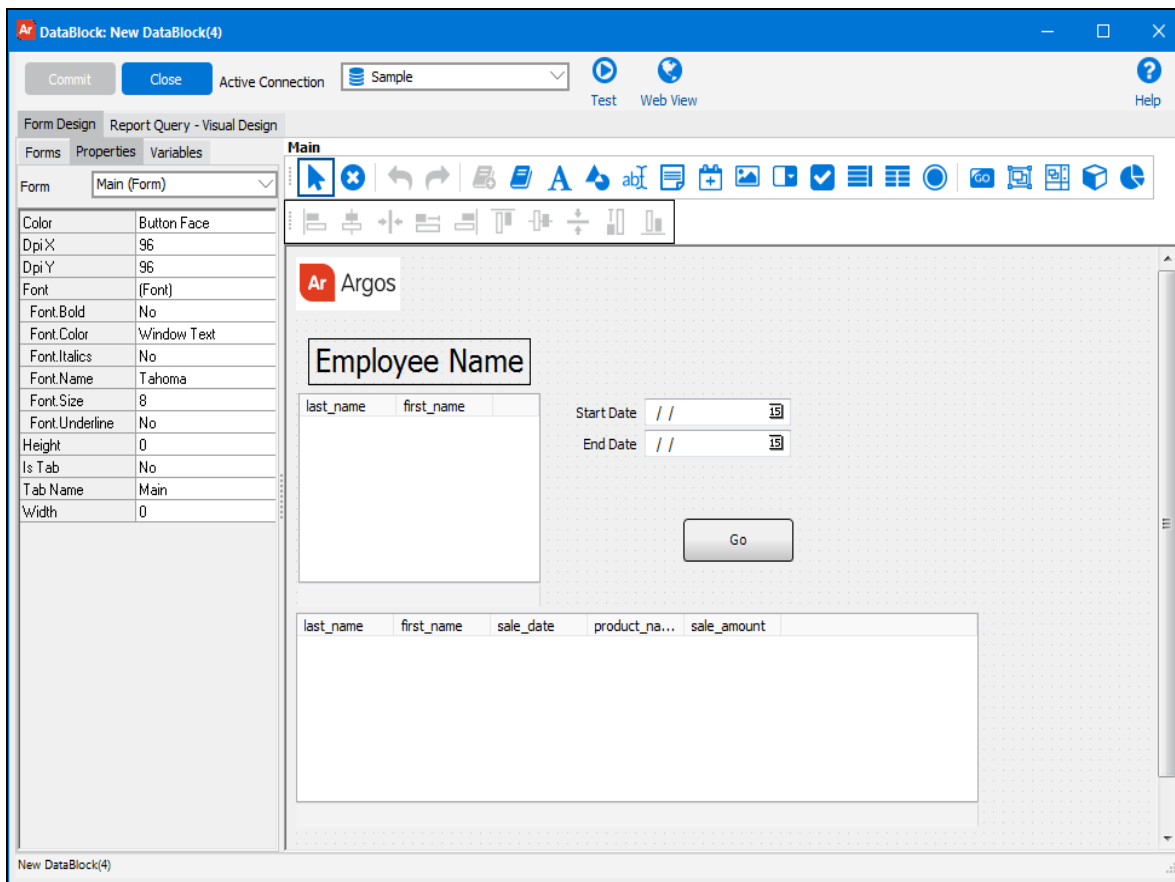
A Form can contain various types of objects such as:

- Edit boxes, list boxes, drop down boxes, scroll boxes - all of which can be used for data entry and display.
- Static and dynamic labels.
- Graphics images from a file or a database.
- Buttons used to control navigation and query execution.
- Shape objects
- Date objects to enter/display dates with an intuitive calendar to ease selection.
- A wide variety of charts/graphs (requires Interactive Charts module).
- OLAP Data Cubes for intuitively viewing and analyzing data (requires OLAP module).

#### Form Queries and Report Queries:

- Within a DataBlock, the queries used by Dashboards are called the Form Queries, and the query used by CSV and Banded Reports is called the Report Query.
- There can only be one Report Query per DataBlock.
- Within a DataBlock, multiple Form Queries can be created to populate list boxes and drop down boxes. Queries can also be used to assign values to Argos Variables.
- All report types within a DataBlock share the same form.

The Argos DataBlock Designer is used to create the forms and contains tools for adding objects to the form. The figure below shows the form design for the report shown in the previous figure.



The objects created on this form are as follows:

- A graphics image (“evisions”) at the upper left.
- Date objects to allow the user to select beginning and end dates.
- A button (Go) to initiate execution of the report.
- Static text labels (Start Date, End Date, and Employee Name) for each object.
- A shape object, which is the rectangle surrounding “Employee Name”.
- A multi-column list box containing a list of employees to be included in the report. The person executing the report selects the employees of interest.
- A multi-column list box to display the query results at the bottom of the form.

Other object types could have been included on the form. For example, a chart or OLAP cube could have been designed with its results displayed on the form. A description of available objects is included within the next section.

It is important to note that certain objects (such as edit boxes, list boxes, etc.) can be used to obtain input from the person executing the report and can also be used to display the results of the query. In this example, multi-column list boxes were used to select employee names and also to display the results of the query. Other types of objects can also be used which are able to obtain input and display results.

The advantage of having reports tied to DataBlocks is that you don't need to create separate input selections and separate queries for every report you create. Similar reports can share the same form, and the same query.

As mentioned earlier, CSV, Banded, and Reports use the same form as Dashboards within the DataBlock to gather input selections. However these reports do not display data on the form as is done with Dashboards. Although CSV, Banded, and Extract reports share the same form within a DataBlock, they do not share the same query with a Dashboard (this will be explained further in the examples section of this document).

The components of a DataBlock described above mention that the DataBlock contains a form and query; however DataBlocks could contain multiple forms and queries. Creation of forms and queries (Form Queries) used for Dashboards is done within the “Form Design Tab” within the DataBlock Designer. Creation of a query used by CSV, Banded, and Extract reports (Report Queries) is done within the “Report Query – Visual Design” tab. The creation and use of both types of queries will be explained in the examples that follow.

## OLAP Data Cube

---

OLAP is a specific way to represent statistical data for executives, specialists and analysts. It is designed to aid in decision-making and better information understanding. The main idea is to answer the user’s questions, arising at the work time, on-the-fly, quickly. A popular definition is “A million spreadsheets in a box.” The key to OLAP is its ability to allow the end user to configure different views of the same data.

An OLAP system allows user to get into details or generalize, filter, sort and regroup data at the time of analysis. Intermediate and final totals are recalculated instantly.

The user is presented data in an electronic spreadsheet format. By moving rows and columns or clicking them, the user makes the system perform calculations and show data in different aspects. Thus, the user can produce many reports out of a single dataset on his own, without any assistance from IT-specialists.

**Filtered OLAP cubes:** Argos supports filtered OLAP cubes in which cubes can be built off of ANY data source – you are not dependent on a data warehouse.

**Note:** You must be licensed for the OLAP module or the Interactive Charts module in order to create OLAP cubes or charts.

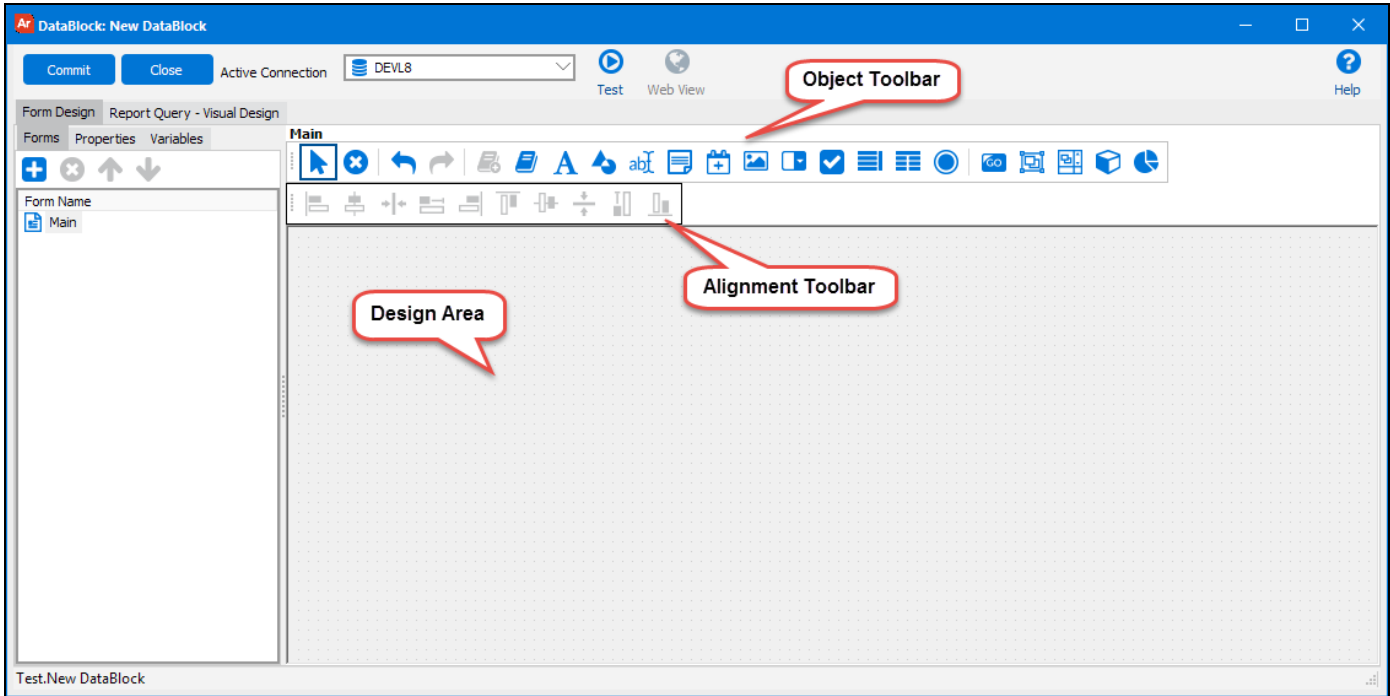
## Charting

---

This option provides display of data using charts, gauges, maps, and other graphical elements providing an at-a-glance understanding of information. Over 40 chart types (bar, line, gauge, 3D, and more) are provided. Charts can be included in dashboards and banded reports.

# The DataBlock Designer

The first step in creating a new DataBlock is to create the objects used by the forms and queries. This is done within the DataBlock Designer, shown in the figure below.







To add an object, select that object in the toolbar and then click in the design area where you want to place the object. You can then use the alignment tools to position the objects relative to each other.









## Object Toolbar











The following table describes the function of each icon within the Object Toolbar. The Examples section of this document will explain the use of the most commonly used objects.

Icon	Description
	<b>Selection Arrow</b> - This tool does not create form objects, instead it is used in the typical fashion to select, move or resize objects. If you hold down the SHIFT key while using this tool, you can select multiple objects. If you click and drag, the selection tool will draw a box which can be used to select multiple objects as well.
	<b>Delete</b> - Use this tool to delete selected object(s).
	<b>Undo / Redo</b> - Use these tools to undo and redo actions. These tools do not create new objects, although Redo can be used to restore a deleted object.
	<b>Add to Library of Objects</b> - Adds the object(s) you currently have selected to the <a href="#">Library of Objects</a> so that you can use them again later.



Icon	Description
	<b>Insert from Library of Objects</b> - Opens the <a href="#">Library of Objects</a> so you can select from previously saved objects to add to your dashboard.
	<b>Static Label</b> - This tool allows you to create labels on your form. Labels can contain static text or be dynamically filled using the Data Aware property. You can configure a custom cursor (such as a handpoint for a link) as well as assign events using the On-Click property.
	<b>Shape</b> - This tool allows you to create rectangles, squares, circles and ellipses. You can choose a rounded variation of rectangles and squares.
	<b>Edit Box</b> - This tool allows you to display a single line of static text or to collect text input from a user. Edit boxes also support: <ul style="list-style-type: none"> <li>■ Populating with data (Data Aware property)</li> <li>■ Collecting lists of items (Multi Entry property)</li> <li>■ Setting to Read Only</li> </ul>
	<b>Memo</b> - This tool allows you to display or collect multi-line input from a user (scroll bars will be added as needed). Memo Fields also support: <ul style="list-style-type: none"> <li>■ Populating with data (Data Aware property)</li> <li>■ Collecting lists of items (Multi Entry property)</li> <li>■ Setting to Read Only</li> </ul>
	<b>Date Edit</b> - This tool allows you to collect date input from a user. The user can either type in a date, or click the icon to bring up a calendar. Date display format is controlled by the local PC regional settings, although the date itself is handled internally in a standard format regardless of local PC settings. Date Edit objects also support: <ul style="list-style-type: none"> <li>■ Populating with data (Data Aware property)</li> <li>■ Ability to default "today" or a fixed date</li> <li>■ Ability to define first day of the week</li> </ul>
	<b>Image</b> - This tool allows you to add images to your DataBlock. When you add an image, you have three options: <ul style="list-style-type: none"> <li>■ Image stored in the DataBlock - This option allows you to browse for an image on your hard disk and store the image statically in the DataBlock.</li> <li>■ Image stored on an accessible server - This option allows you to link to an image that is stored in a location accessible to the MAP Server. This is useful as the image itself can be changed without having to update your DataBlocks (assuming the location and filename remain the same). Note that this location can be dynamic by incorporating expression logic in the pathname. Remember that the location of the image must be accessible to the MAP Server and the location is relative to MAPS, not the local PC.</li> <li>■ Image retrieved from a variable - This option allows the image to be inserted from a database query (or other Argos variable). This can be very useful for dynamically displaying data-driven images. You will need to specify which variable holds the image and the MIME type of the stored image.</li> </ul> <p>Image objects can be set to autosize (to match the exact size of the image) or you can turn that property off and enable stretch if you need to make the image larger or smaller.</p>
	<b>Drop Down</b> - Create a list of options for the user in a Drop Down object. The source of the choices can be manually entered, or dynamically generated from a SQL statement. Note that Drop Down objects only display a single field. If your query returns more than one field, Argos will prompt you for which field to display (use a Multi-column list box to display more than one column). The other SQL fields are still available for use, they just cannot be displayed. Drop Down objects also support: <ul style="list-style-type: none"> <li>■ Ability to automatically select first list item (Auto Select property)</li> <li>■ Customized column headers (Columns property)</li> <li>■ Ability to type in choice (Free Type property)</li> </ul>

Icon	Description
	<b>Check Box</b> - Create a check box to gather Boolean (Yes/No) input from users. You can define the value of the object when it is checked and when it is unchecked, and the default state.
	<p><b>List Box</b> - Display a single column of data and allow users to select one (or more) items from the list (scroll bars will be added as needed). The source of the choices can be manually entered, or dynamically generated from a SQL statement. Note that List Box objects only display a single field. If your query returns more than one field, Argos will prompt you for which field to display (use a multi-column list box to display more than one column). The other SQL fields are still available for use, they just cannot be displayed. List Box objects also support:</p> <ul style="list-style-type: none"> <li>■ Ability to automatically select first list item (Auto Select property)</li> <li>■ Customized column headers (Columns property)</li> <li>■ Ability to select multiple list items (Multi Select property)</li> <li>■ Ability to show the list item count (Show Item Count property)</li> </ul>
	<p><b>Multi-column List Box</b> - Works the same as the List Box except that it can display multiple columns of data. It shares the same properties as the List Box and also supports:</p> <ul style="list-style-type: none"> <li>■ Ability to autosize a column (Auto Size Column property)</li> <li>■ Ability to add column and/or row lines</li> <li>■ Ability to define a custom multi-field sort order (Sort property)</li> </ul>
	<p><b>Radio Buttons</b>- allows for the creation of a panel containing of any number of radio buttons. Radio button choices can be manually entered or dynamically generated from an SQL query. The value of the selected radio button will be stored in the variable associated with the radio button panel. For example, in the column to the right, if "Button 2" is selected at runtime, "Button 2" will be stored in the variable.</p> <ul style="list-style-type: none"> <li>■ To manually create radio buttons, select the radio button icon on the toolbar, then click within the work area to add the radio panel . Click the "Choices" property then select "Manual Entries". You will then be prompted to enter the choices to be displayed for each button. For example, in the column to the right, "Button 1, Button 2, Button 3" were entered as the choices. At runtime, the choice associated with the selected radio button will populate the variable. For example, if "Button 2" was selected at runtime, the variable for the radio button will contain "Button 2".</li> <li>■ To dynamically create radio buttons, add the radio button as described above, but in this case after clicking "Choices", select "SQL statement" . The Visual Query Builder will then appear where you can create a query to return values that will be displayed as choices for radio buttons. If your SQL query returns more than one field, Argos will prompt you for which field to display. The other SQL fields are still available for use, they just are not displayed. As above, at runtime the value of the selected radio button will populate the associated variable. A maximum of 100 radio buttons can be created. If your query creates larger number, an error message will be displayed.</li> </ul>
	<p><b>Button</b> - Buttons can be used to control when other objects are run, which form is displayed, etc. The On-Click property is of special importance to button objects as it is used to define what occurs when a user clicks the button. The value of the button variable is undefined until it is clicked, at which point it takes on a value of 1. This can be a useful way to control when SQL queries run as Argos will not run a query if one of its dependent variables is undefined. By placing a line into the WHERE clause of a query like the following, you can ensure that the query will not execute until the button is clicked (replace ":button" with the name of your button object): WHERE :button IS NOT NULL.</p> <p>There are a number of events that can be activated using the On-Click Property as follows:</p> <p>Activate Form – activate another form in the forms list</p> <p>Clear Variables - Clear the value of selected DataBlock variables</p> <p>Fetch File – Retrieve a file from the Server</p> <p>Hyperlink – Launch a URL in the default web browser</p>

Icon	Description
	Refresh Variables - Refresh the value of selected DataBlock variables Reset Variables – Reset the value of selected DataBlock variables back to their default value
	<b>Panel and Scroll Box</b> - Creates an area on which other objects can be placed. Objects placed onto a panel or scroll box become children of their parent object. Panels and scroll boxes operate in essentially the same fashion except that scroll boxes automatically create scrollbars as needed. While neither object stores text, both allow you to define a default font. Objects placed onto the panel or scroll box will "inherit" their parents' font settings. You can, of course, override the parent font settings by setting an objects' font directly.
	<b>OLAP Data Cube</b> - Create On-Line Analytical Processing (OLAP) Data Cubes.
	<b>Chart</b> - Create a variety of charts such as bar graphs, pie charts, line charts, etc. When adding a chart, the Chart Wizard is used to define the data source for the chart and other chart options. An advanced mode is available that gives complete control of all chart properties. However, once a chart has been modified in Advanced mode, you cannot revert back to the Chart Wizard.

## The Alignment Toolbar



A variety of alignment tools are available to make it easy to create attractive forms. The alignment tools are shown graphically on each button, but if you place your mouse over a button, it is described in text as well.

### [Alignment of a single object](#)

When a single object is selected, the only available options are to align horizontally or vertically. Note that this alignment is with respect to the selected object's parent. So if you align an object on a panel, it will be in relation to the panel, not the main form.

### [Alignment of multiple objects](#)

The alignment tools become more useful when you select multiple objects (hold down the SHIFT key while clicking to select multiple objects). All alignment options then become available which allow you to align objects in relation to the first object selected.

# Building a Form

---

## Introduction


---

This example will demonstrate how to create a DataBlock (consisting of a form and a Dashboard) for the Employee Sales report shown earlier in this document. The report contains sales information for selected employees within a specified date range. The user executing the report selects employees of interest from a list-box and enters a date range into date objects.

The DataBlock uses the Employees, Orders, Order\_Details, and Products tables within the Sample MS Access Database. The steps for creating the DataBlock are as follows:

## Add DataBlock icon to Explorer

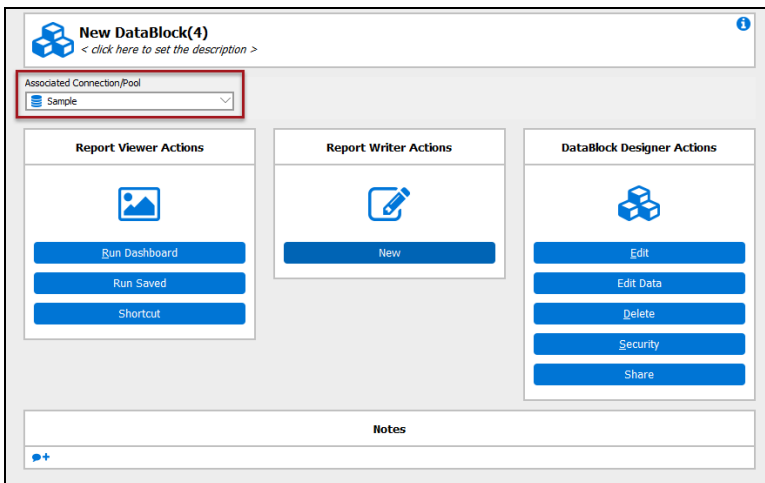
---

Log into Argos and navigate to the location where you would like to add the DataBlock. Right-click and select **New >> DataBlock**. Rename the “New DataBlock” to something of your choosing. Notice that Argos also created a system dashboard underneath the DataBlock. The system dashboard is represented in the Explorer with this icon .

## Select a database connection

---


Choose the appropriate database connection from the **Associated Connection/Pool** drop down. See your system administrator to find out which database connection to use.



## Add a Description, Author, and Notes

---

In the figure above, note the phrase **<click here to set the description>** at the top of the figure above. Click on the phrase to enter a description for the DataBlock.

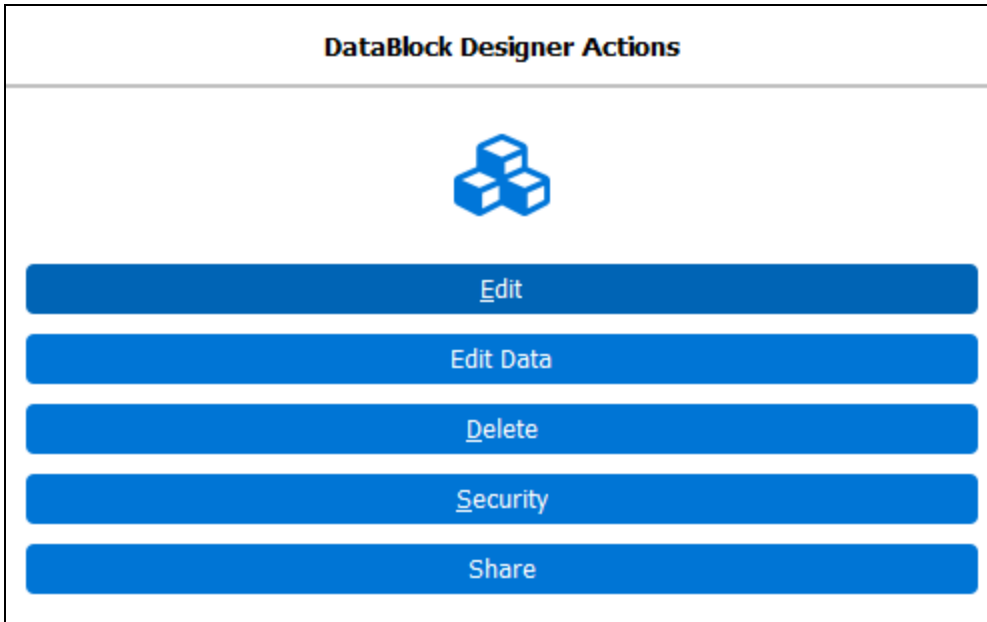
You can also click the blue info icon  in the upper right corner, and click **Author** to edit the author of the DataBlock. The default author is set to the Argos user name of the person currently logged in.

You can also click the green plus sign next to “Notes” at the bottom of the figure to add any number of notes.

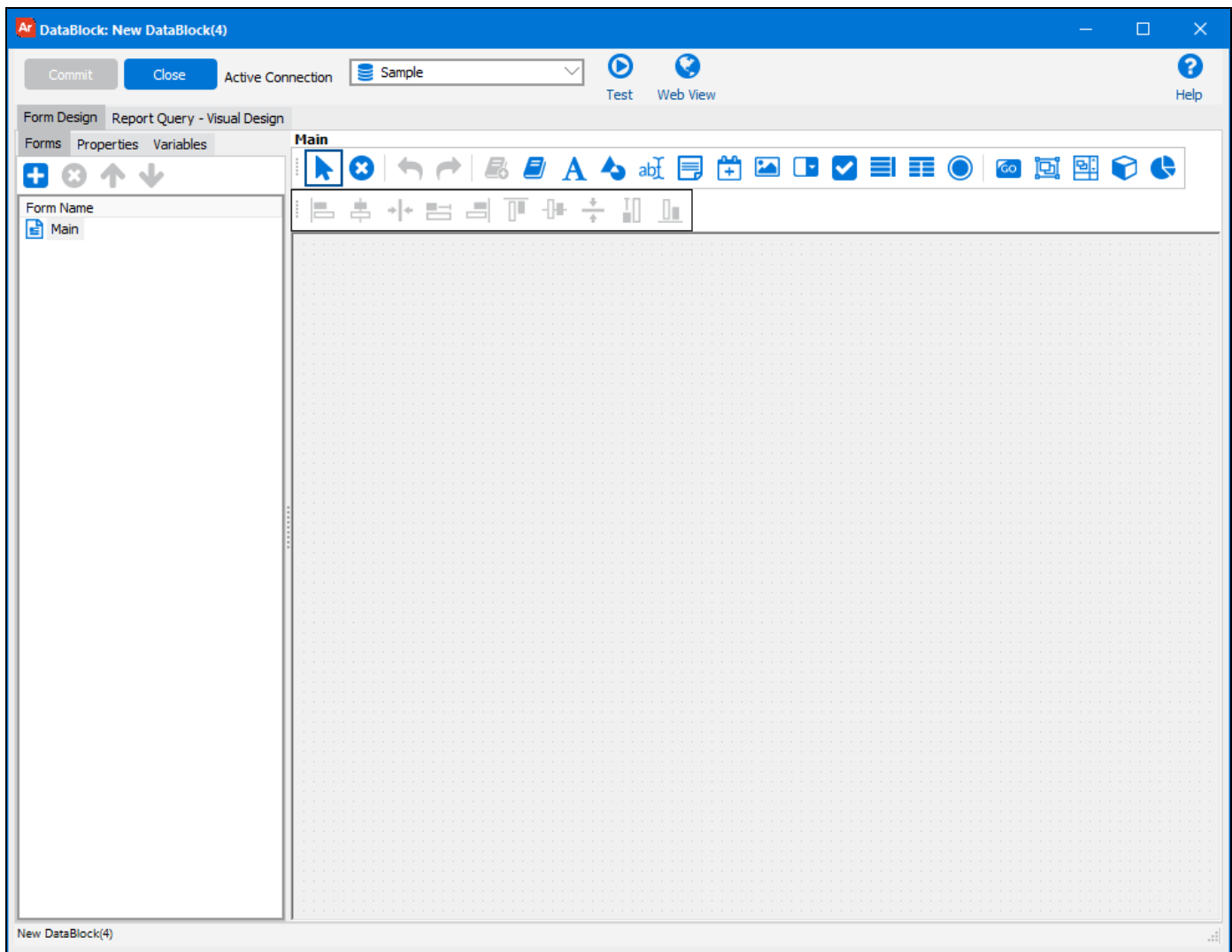
## Launch the Argos DataBlock Designer

---

Under **DataBlock Designer Actions**, click **Edit** to edit the DataBlock. This launches the Argos DataBlock Designer where you will create the DataBlock.



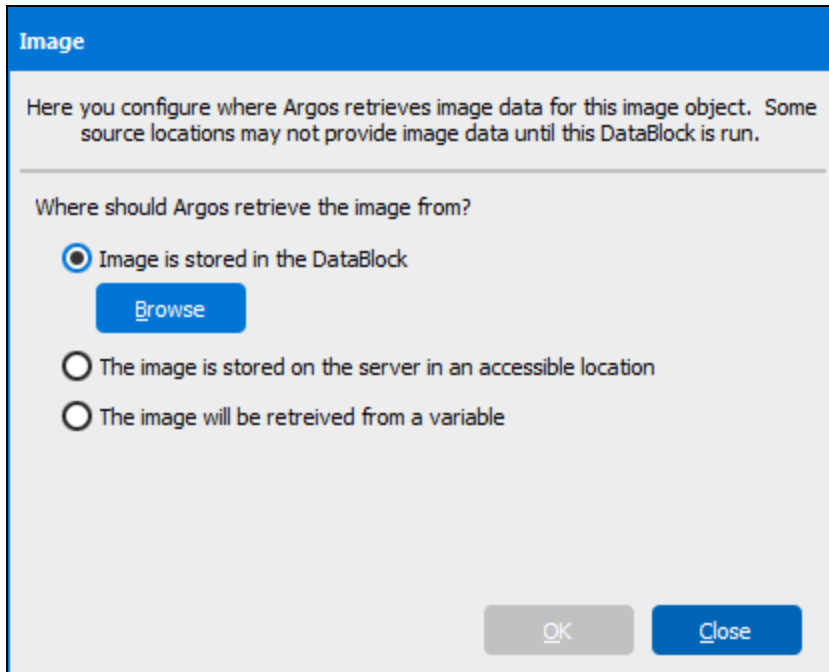
Before adding any objects, click the "Form Design" tab on the upper left of the dialog box which displays the following screen.



## Add the Graphics Object

---

Click the **Add an image** icon , then click anywhere within the Design Area. The following dialog box appears where you enter the location of the graphics image.



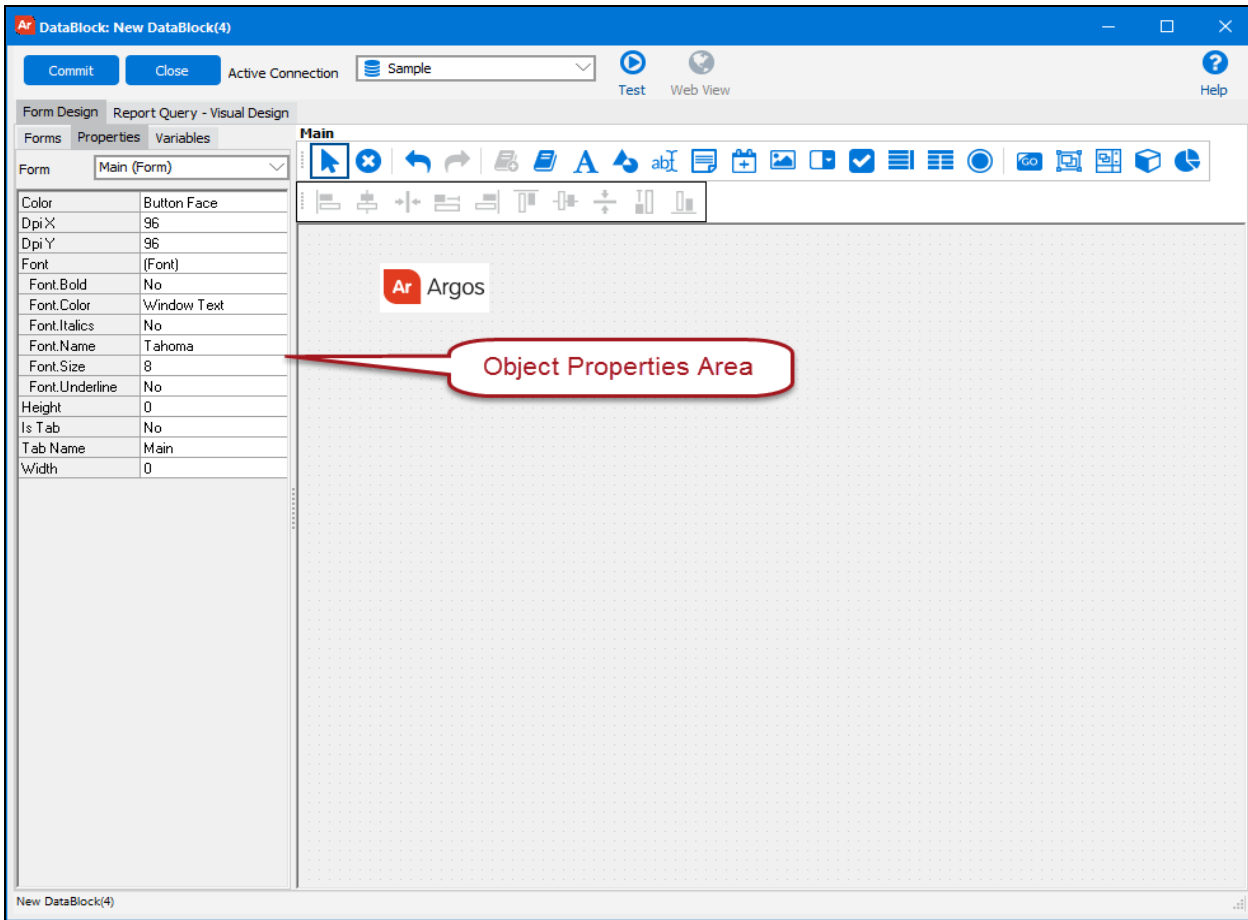
In this case, the image is stored on a local PC, so select **Image is stored in the DataBlock** and click the Browse button. A dialog box will appear in which you enter the location of the image. After supplying the location, the image will be placed on the Design Area. Drag the image to the desired location.

There are three options for accessing an image:

- **Image stored in the DataBlock:** this option allows you to browse for an image on your hard disk and store the image statically in the DataBlock.
- **Image stored on an accessible server:** this option allows you to link to an image that is stored in a location accessible to the MAP Server. This is useful as the image itself can be changed without having to update your DataBlocks (assuming the location and filename remain the same).
- **Image retrieved from a variable:** this option allows the image to be inserted from a database query (or other Argos variable). This can be very useful for dynamically displaying data-driven images. You will need to specify which variable holds the image and the MIME type of the stored image.

# Adjust Object Properties

Single-click the object to display the list of applicable properties (left side of screen).



Note the three tabs on the left of the DataBlock Designer:


- **Forms** tab: displays a list of all forms used within a DataBlock. This example uses only one form, but examples later in this guide show how to create and link several forms together.
- **Properties** tab: displays the properties for the selected object. If no object is selected, nothing is displayed.
- **Variables** tab: displays a list of all variables existing in the DataBlock. Use of variables will be demonstrated in this example.

For each object that is created, a list of properties applicable to the particular object appears on the left side of the screen as shown above. You can modify properties by entering values directly into each field, or you can select from a list of options that appear when you click on the property. The list of properties appears when the object is first created or when you click on the object.

For this example, change the **Auto Size** property from "no" to "yes". Setting Auto Size to "yes" will increase the size of the field to accommodate the image.



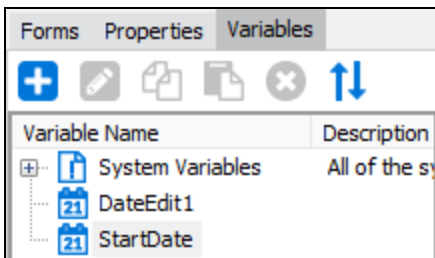
## Add the Date Objects and Labels


Click the “date edit” icon  then click anywhere in the design window to create the object. Repeat this to create a second date object. Drag the objects to the desired locations.

Note the property called **Variable Name** in the figure below. This is the default name assigned to the object by Argos. Since the date objects will be referred to later when building SQL queries, it is helpful to create meaningful names. Therefore, rename the first date object “StartDate” and the second date object “EndDate”. This is done by typing the new names into the “Variable Name” field in the Properties area.

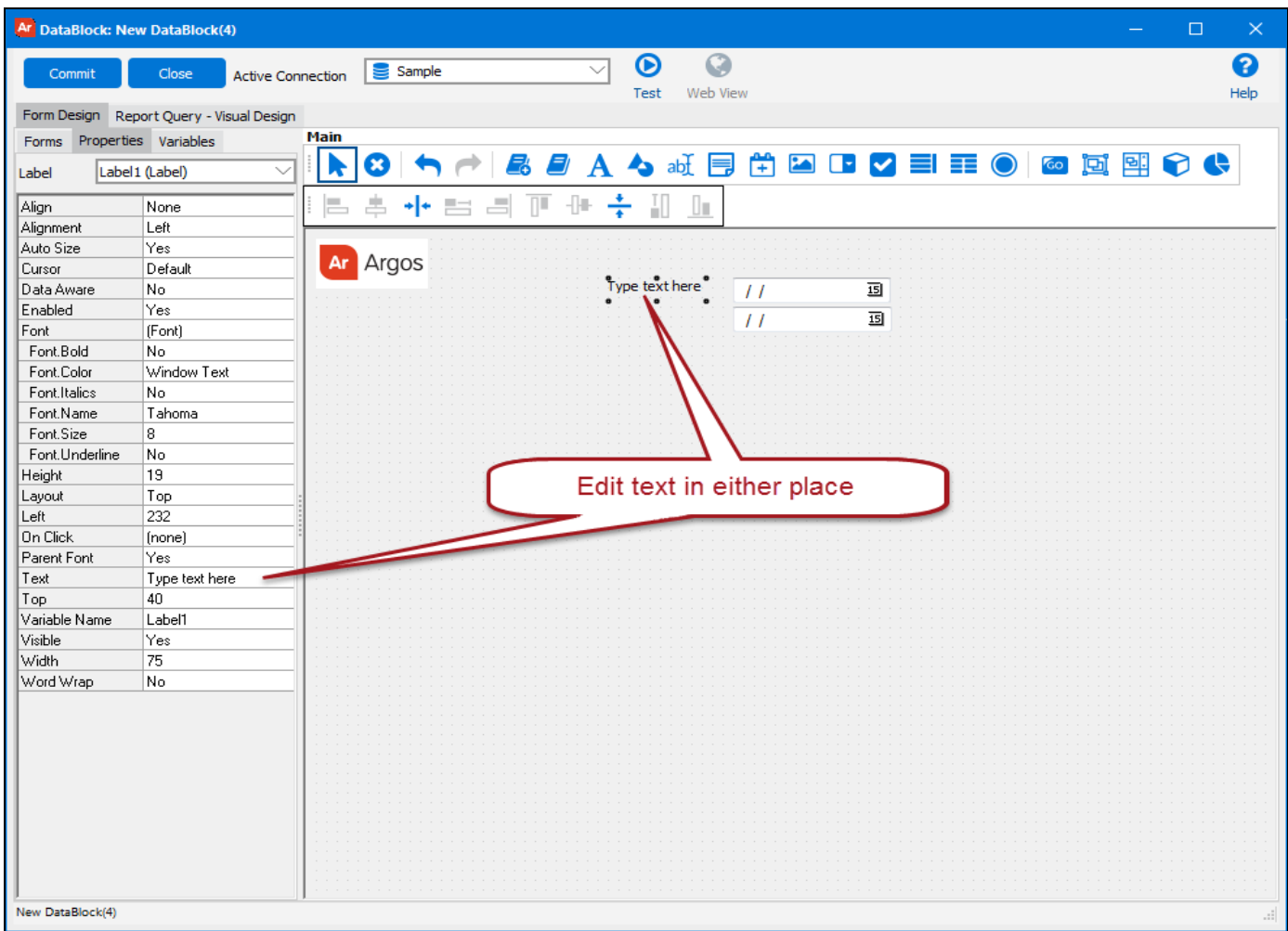
Start Of Week	Sun
Tab Order	1
Tab Stop	Yes
Top	40
Variable Name	StartDate
Visible	Yes
Width	121

In the figure below, note the existence of the two new variables (StartDate and EndDate) within the Variables tab.




The next step is to add a label to each Date object. Click the text  icon on the toolbar to create a static label, then click anywhere within the Design Area. To enter the label “Start Date”, either click on the “Text” property on the left of the window or double-click within the “Type text here” field that was created for you by Argos. Enter “Start Date” into the appropriate area. Repeat the process for the “End Date”. Drag the labels to the desired position near the Date Objects that have already been created.

Note the various font properties that are available (bold, color, italics, font name, font size, font underline) that you may want to change.



## Align the objects



To align the date fields by their left edges:

1. Select both date objects by highlighting them, or by holding the **Shift** key and clicking on both.
2. Click the "Align selected controls by their left sides" icon  on the toolbar.

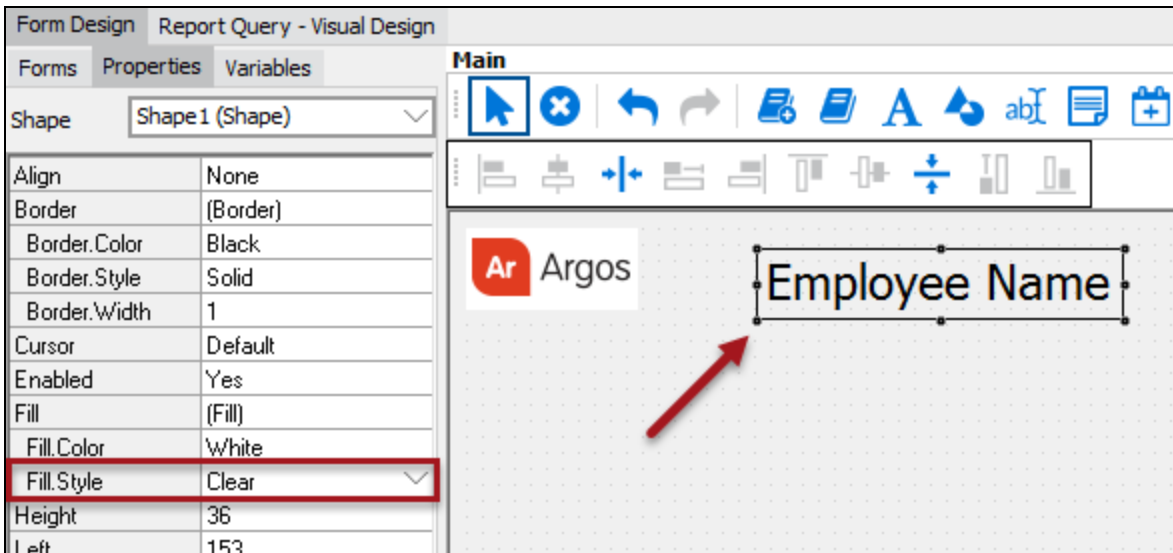
Argos aligns the left side of the second object to the left side of the object selected first. You can use the same technique to align the top of each date label to its corresponding date edit box. Alternatively, click and drag the objects to the desired locations.

## Add Employee Name Label and Shape Object

The next step is to add the "Employee Name" label and surround it with a rectangle.


1. Add a static label  with the text "Employee Name" to the Design Area. Adjust the look of the label on the properties tab.
2. Click the "Create a shape on the form" icon . Click and drag it over the "Employee Name" label.

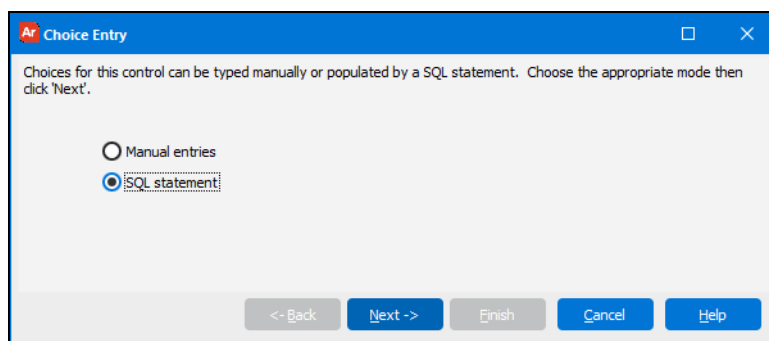
- By default, the box will be have a white fill, which obscures the label. Change the **Fill.Style** to "Clear" to leave a simple black border around the label.
- You can then modify the size of the border by dragging the handles at the corners.



## Create the object for selecting Employee Name

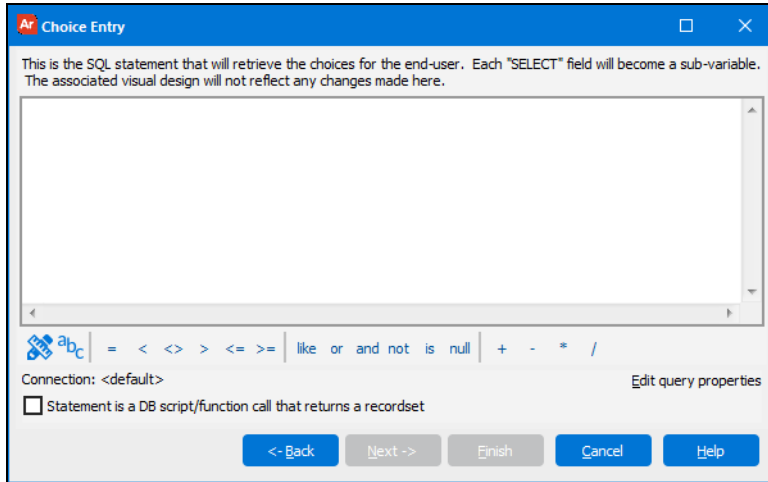
A multi-column list box will be used to display a list of employee names. The person executing the report will select desired names to include in the query. Both employee last name and first name will be contained in the list box.


- Begin by clicking the multi-column list box icon  then click anywhere in the design area to place the object.
- In the properties area for this object, change the Multi Select property from no to yes. This will allow the user to select more than one entry.
- Since this object is used later when building an SQL query, give the object a meaningful name such as "EmployeeList" by editing the Object Properties area.
- Double-click on the new object to bring up the following dialog box that allows you to specify which employees are to be included in the query.
- Choose **SQL statement** to obtain names from the database.

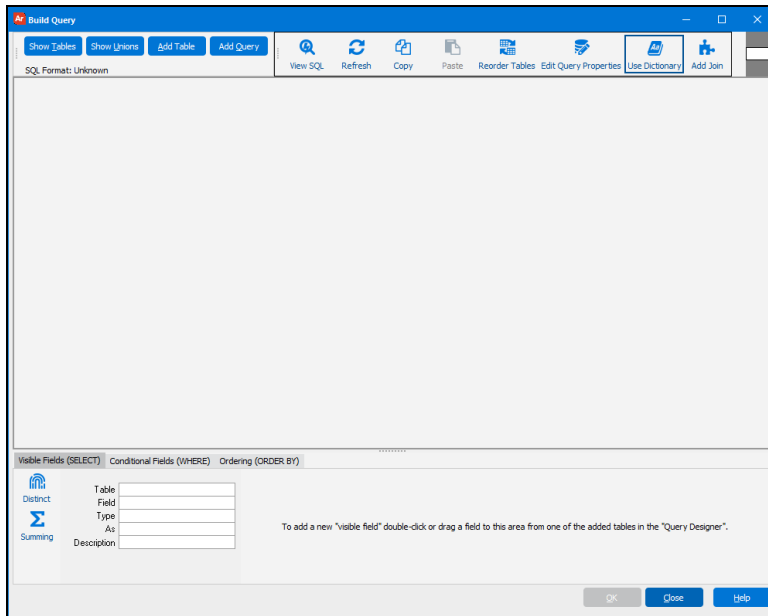


**Object Choices Property:** Drop-down boxes, list boxes, and multi-column list boxes contain the “choices” property which allows these objects to accept input from the user and display query results as well.

6. After selecting “SQL statement”, the following dialog box appears:



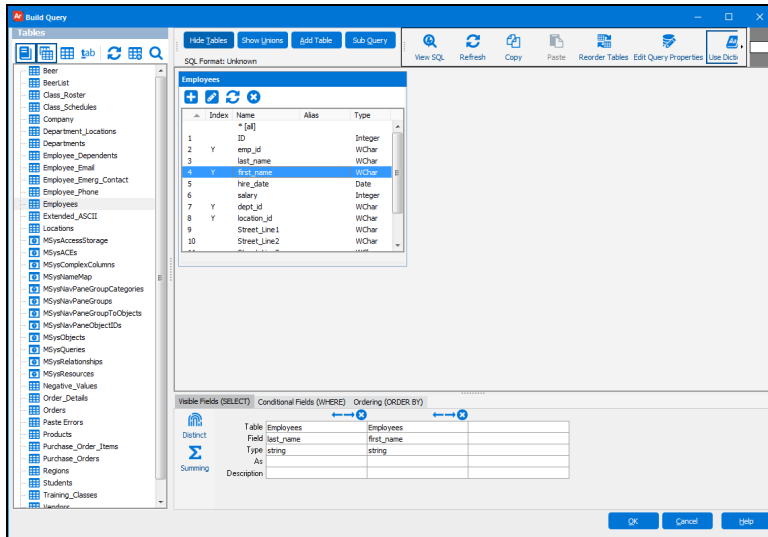
7. Click the Build Query icon  which brings up the Build Query dialog box. Visit [this page](#) for more information on the Build Query dialog box.



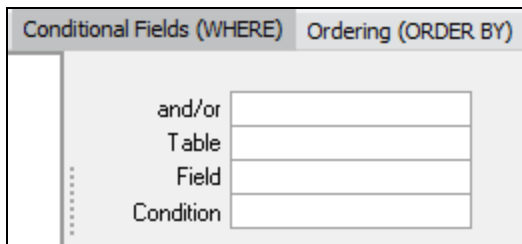
8. Click **Show Tables** at the upper left which displays the list of tables within the database.

9. Double-click the “Employees” table to add it to the query. The Employees table contains the employee names.

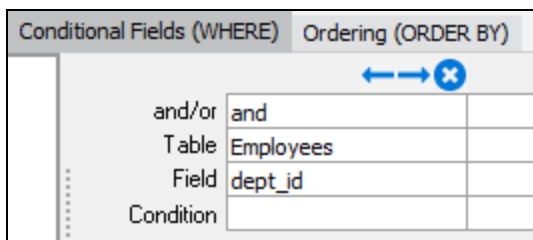
- Double-click "last\_name" and "first\_name" to move them to the query table at the bottom of the screen. Fields that appear in the query on the **Visible Fields** tab will be included in the SELECT statement.



- We only want to include the employees in the Sales department, so we need to add a condition (WHERE clause) to the SQL query. To add a WHERE clause to obtain only the employees in the Sales Department, click the "Conditional Fields (WHERE)" tab.

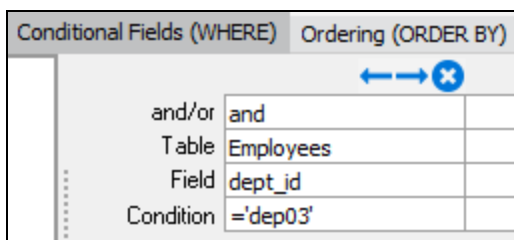


- Double-click on dept\_id to autofill the **and/or**, **Table**, and **Field** fields.

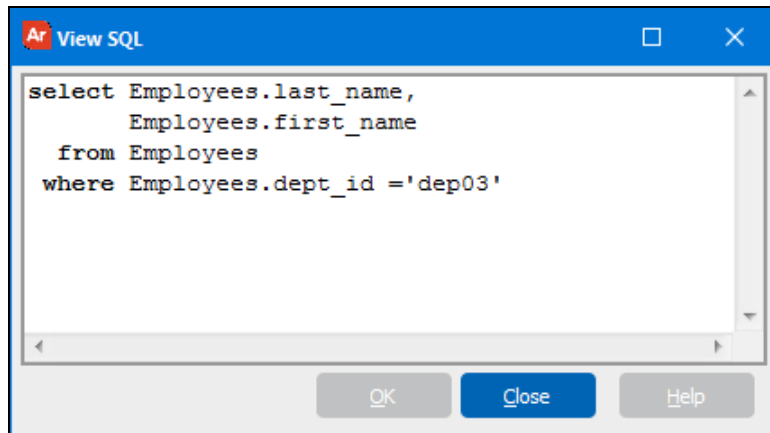



- Click the "Condition" field, then click the ellipsis button  which brings up the SQL Editor. Enter = 'dep03' in the text edit box.

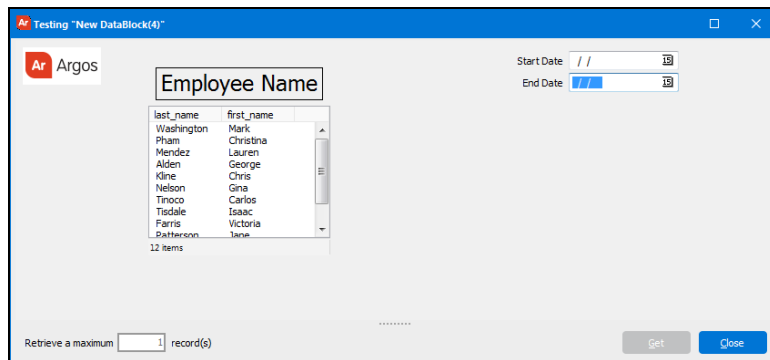
- Click OK to continue. The completed condition is shown in the figure below.



- Click the **View SQL** button at the top of the Build Query dialog box which shows the resulting SQL Query translated by the Visual Design. It shows that only employees within 'dep03' will be listed in the multi-column list box.



- Click **Next**. The next screen shows a preview of the listbox.
- Click **Finish**.
- To check your work, click the **Commit** button at the top left of the Build Query dialog box to save your work, then click the **Test** icon  to test the form and query design. Note that you can specify how many results you want to display for the test at the bottom of the form. Enter 0 to display all results.



- Note that the multi-column list box is now populated with the list of employees in the Sales Department. Click **Close** to continue the DataBlock design.

This completes the activities required to create the form that is executed when running the report. The next section in the guide describes how to create the query that will obtain and display information from the database onto the form.

# Building a Query

---

There are two types of queries discussed here:

- [Form Query](#): used to display results on the form.
- [Report Query](#): used to generate Argos reports.

## Form Query

---

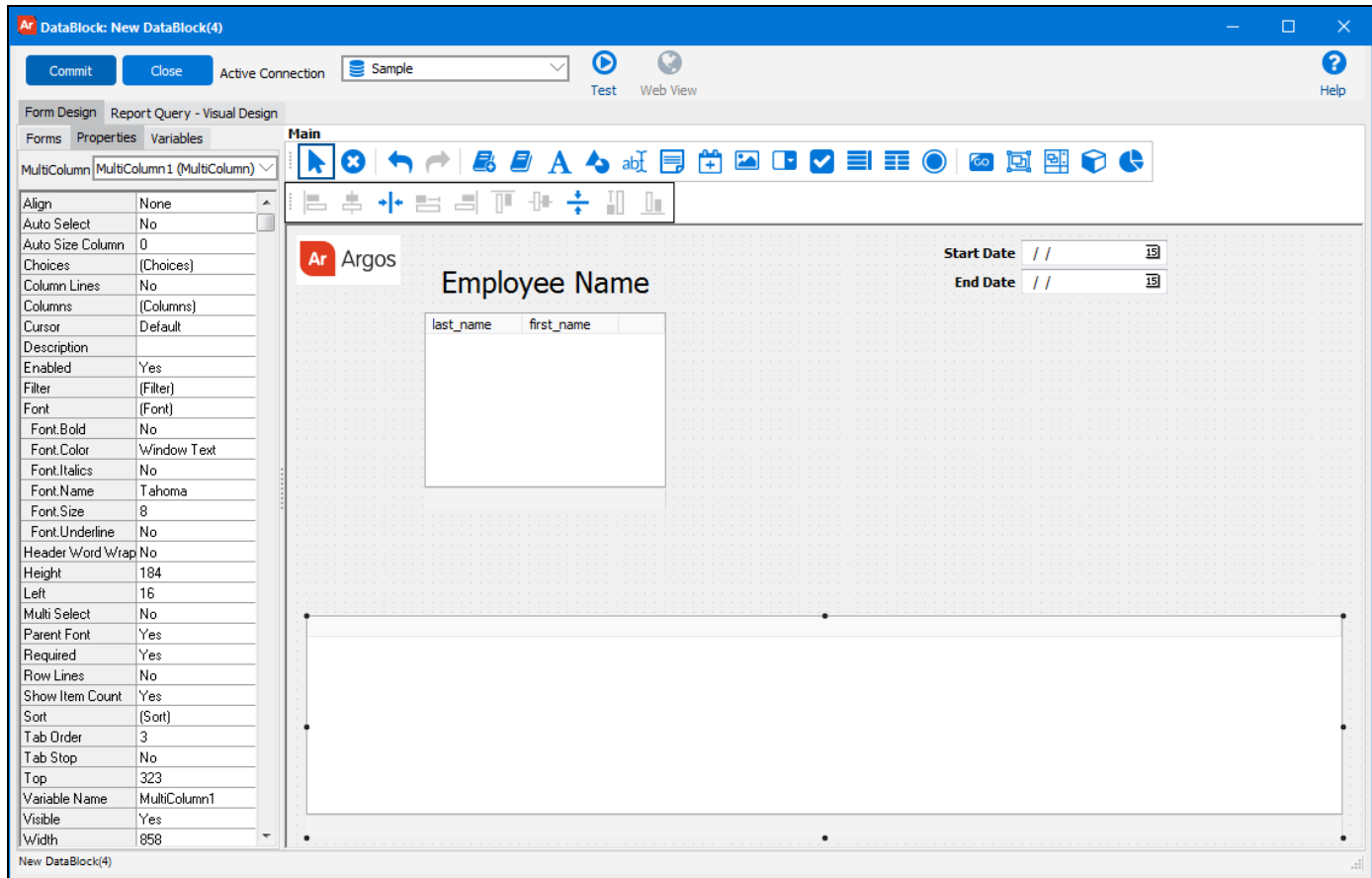
This is a continuation of the DataBlock design which began in example 1. The form has been designed; the next step is to create the query to obtain data from the database and display the query results on the form.


The query results are to be displayed on the form into a multi-column list box. The query to be created is called the Form Query since it is created within the Form Design tab in the Argos DataBlock Designer.

**Tip:** Although this exercise uses only one object (and its associated query) to display results on the form, multiple Form Queries could be created if several objects require display of query results.

## Create the multi-column list box to contain the query results

The next step is to create the multi-column list box at the bottom of the form which will contain the results of the query. Click the multi-column list box icon on the toolbar, then click anywhere within the Design Area. Increase the size of the object (horizontally and vertically) to accommodate the data that will populate the object. This is done by dragging the corners of the object to the desired locations.



As was done with the previously created multi-column list box, double-click on the new object, and select **SQL statement**, click the Build Query icon  to bring up the Build Query dialog box, then click Show Tables.

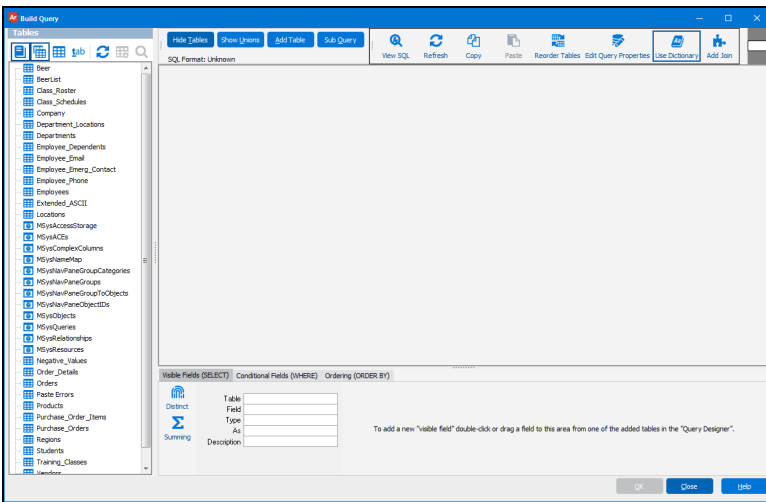
The columns in the report will contain the:

- employee last name
- employee first name
- order date
- product name
- sales amount (quantity x unit cost)

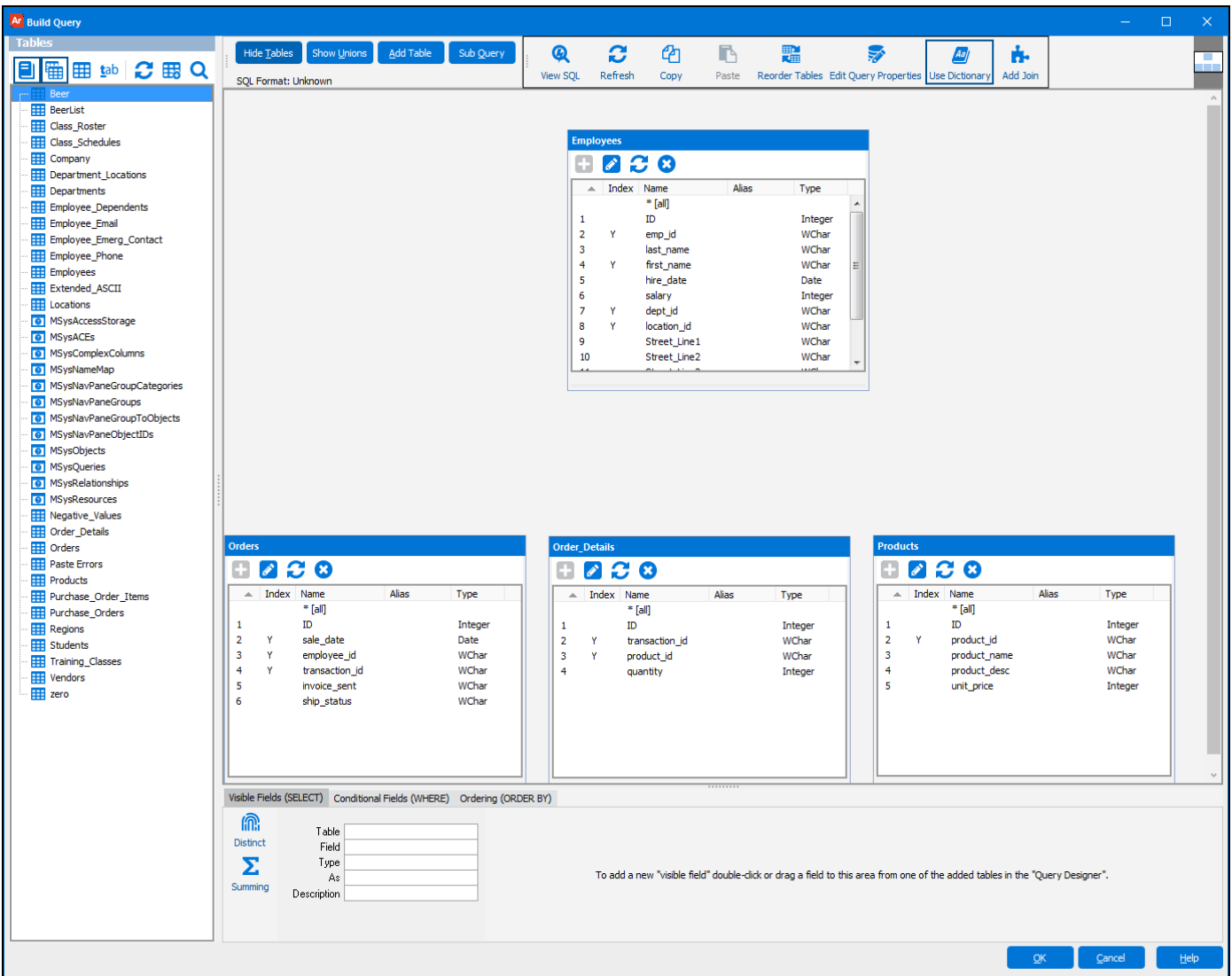
These items are located in several tables within the sample database (Employees, Orders, Order\_Details, Products), thus the SQL query must contain a join to link the tables together. Note that the sales amount does not exist within the database and must be calculated using the quantity and unit cost fields within the database. The use of the Build Query dialog box to build the appropriate SQL query follows.

At this point the Build Query dialog box appears as shown below. The next step is to select the tables to be included within the query. Click **Show Tables** to see the list of tables.





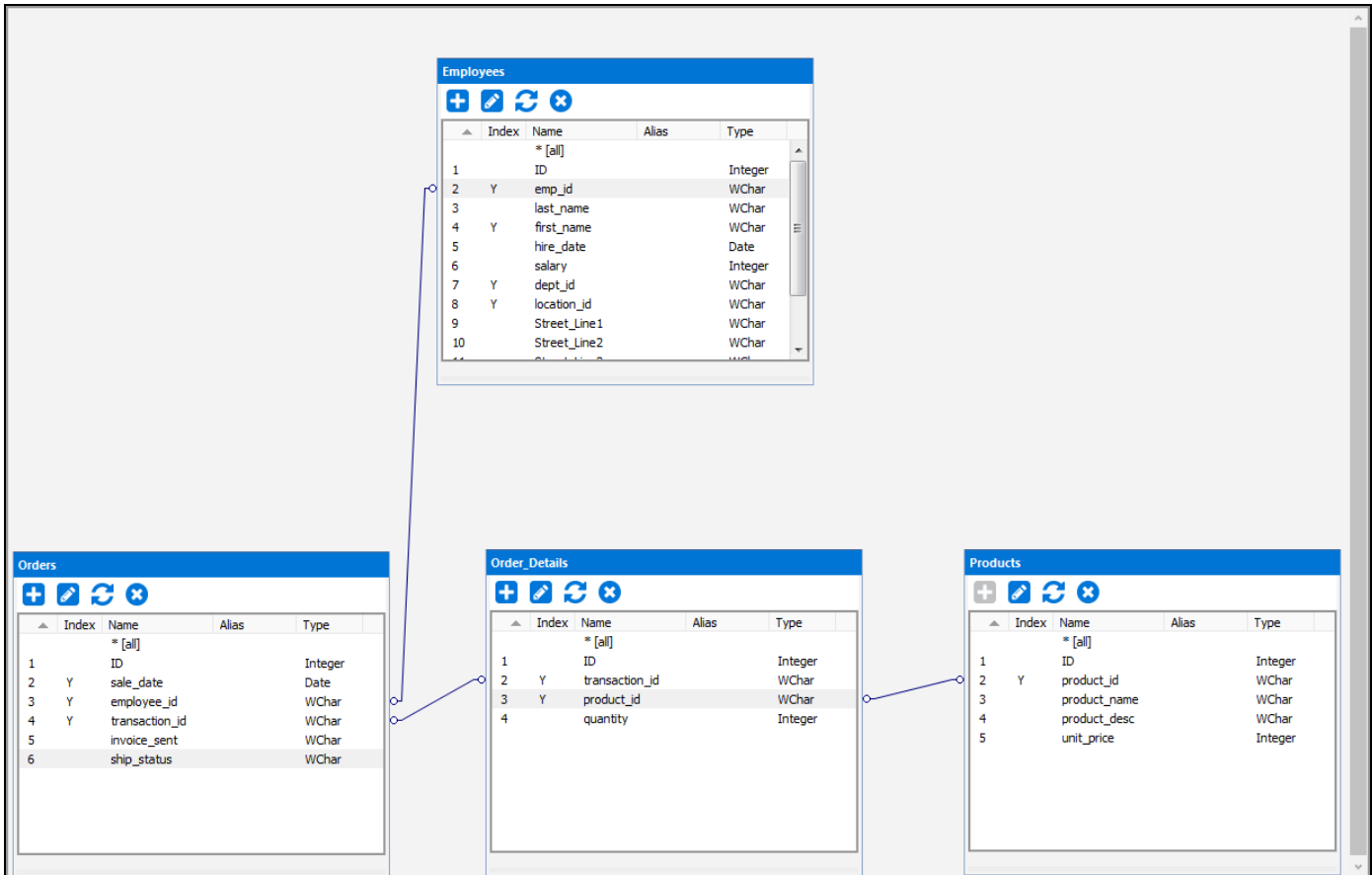
Double-click the **Employees**, **Orders**, **Order Details**, and **Products** tables. This moves the tables to the right where you can join them to each other. Maximize the screen to provide space. Drag the tables to position them as shown in the figure below.



The following joins need to be created:

- `Employees.emp_id` with `Orders.employee-id`
- `Orders.transaction_id` with `Order_Details.transaction_id`
- `Order_Details.product_id` with `Product.product_id`

This is done visually by selecting the field in one table, holding the mouse down, then dragging the mouse to the corresponding field in the other table. Lines indicating the joins are then displayed as shown below:



## Modifying the joins

To modify a join, right-click on the joining line. The following options will appear:



Selecting "Edit Join" will allow you to change the type of join.

**Edit Join**

Edit the properties of this table join

Left Table:  Right Table:

Left Field:  Right Field:

Inner Join  
 Include all records from 'Order\_Details' and 'Products' where 'product\_id' equals

Outer Left Join  
 Include all records from 'Order\_Details' and only those records from 'Products' where 'product\_id' equals 'product\_id'

Outer Right Join  
 Include all records from 'Products' and only those records from 'Order\_Details' where 'product\_id' equals 'product\_id'

Contribution  
 Suggestion

## Identify the fields to appear in the report

The next step is to specify which database fields will appear in the report. Double-click on each field to be included to move it to the Visible Fields tab as shown below. These fields will also appear in SELECT statements in the SQL that is being created.

The 'last\_name' and 'first\_name' fields were selected from the Employees table, 'sale\_date' was selected from the Orders table, and the 'product\_name' field was selected from the Products table.

Visible Fields (SELECT)		Conditional Fields (WHERE)		Ordering (ORDER BY)	
Distinct	Table	Employees	Employees	Orders	Products
	Field	last_name	first_name	sale_date	product_name
	Type	string	string	date/time	string
Summing	As				
	Description				

### SELECT/WHERE/ORDER BY tabs:

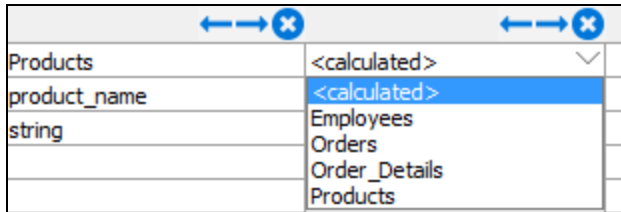
- The Visible Fields (SELECT) tab contains the fields that will appear in the report. These fields exist in SELECT statements.
- The Conditional Fields (WHERE) tab contains the fields that appear in the WHERE clause of the SQL query.
- The Ordering (ORDER BY) tab determines the sort order of fields that the query returns.


## Create a calculated field to determine sale amount

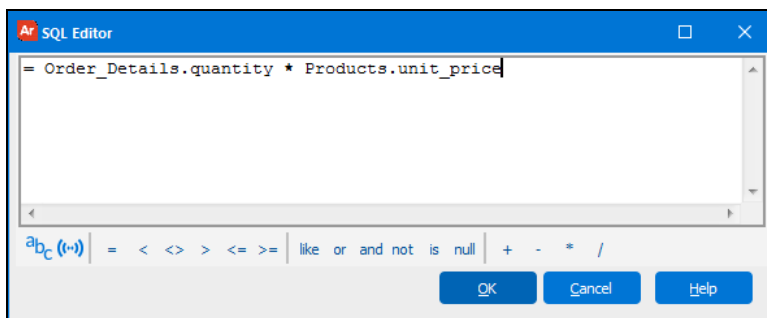
The report must show the total sale amount, which is not a field in the database, so it must be calculated. The total sale amount is equal to the product quantity multiplied by the unit cost, which are fields in the database. A calculated field must be created and added as an entry underneath the Visible Fields tab.

Within the **Visible Fields** tab:

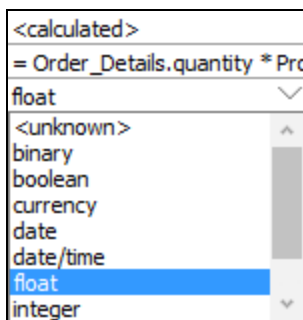
1. Click the empty **Table** field to the right of the "Products" table entry.
2. Click the down arrow that appears in the field to get a list of available tables.
3. Select '<calculated>' from the drop-down.



4. Click inside the **Field** field (below the Table field), then click the ellipses icon  that appears inside the field to launch the SQL Editor. Enter the following into the text field:  
`= Order_Details.quantity * Products.unit_price`



5. Click **OK** to return to the Build Query.
6. Click in the **Type** field, then click the down arrow within the field to get a drop-down with a list of data types.
7. Select **Float** from the drop-down.

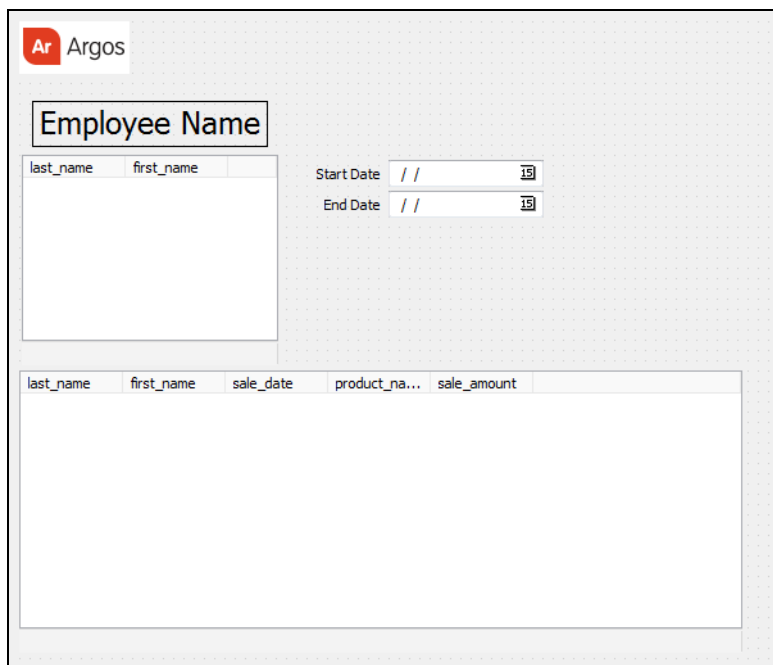


8. Click in the **As** field and enter 'sale\_amount'. This new calculated field can now be used in the same fashion as other fields in the database.

9. Click **OK**, then **Finish** to continue.

<calculated>
= Order_Details.quantity * Pro
float
sale_amount

You can now see the new multi-column list box object shown within the DataBlock Designer with all of the fields added.



## [Use input selections to limit the query](#)

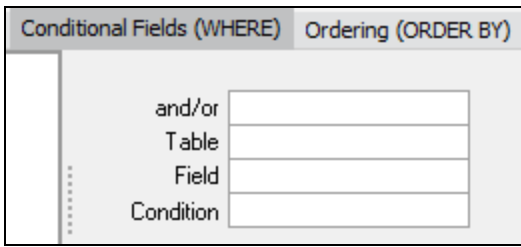
This query in its current state will return all sales records for all employees. However the query is to be limited by the input selections entered on the form. Therefore, a WHERE clause is required to limit the query to selected employees within a date range.

Within the DataBlock Designer, double-click on the object that was just created which contains the results of the query. Then click the hardhat/hammer (Edit Visual Design) icon to bring up the Build Query dialog box. Click the "Conditional Fields (WHERE) tab where you can create the required WHERE clause.


The WHERE clause should only obtain records where:

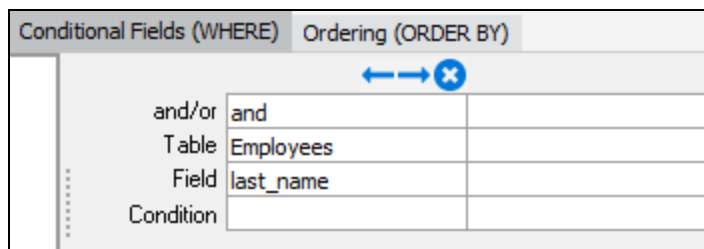
- The employee last name selected = the employee last name in the database and,
- The employee first name selected = the employee first name in the database and,
- The sale date is greater than or equal to the start date selected and,
- The sale date is less than or equal to the end date selected



The above limits the query to the selections entered by the person executing the report. Each of the above conditions will be entered under the Conditional Fields tab.

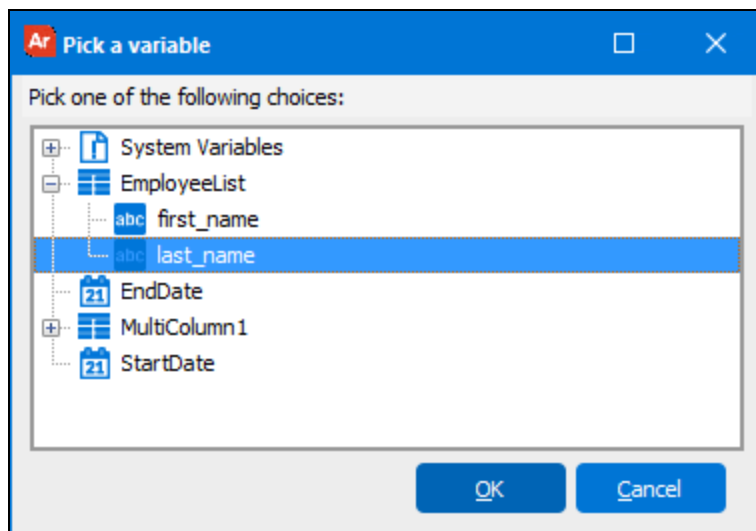


The condition for 'Employees.last\_name' will be created first. If you already have the Build Query open, then skip to Step 3.

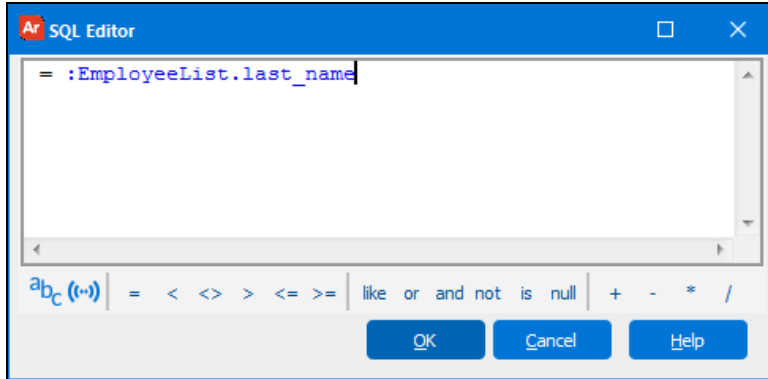
1. Double-click on the multi-column listbox, that we edited in the [previous exercise](#), to launch the Choice Entry window.
2. Click **Edit Visual Design** (  Edit Visual Design ) to launch the Build Query window.
3. Select the **Conditional Fields (WHERE)** tab.
4. Double-click the 'last\_name' field in the 'Employees' table to auto-populate the first three fields.



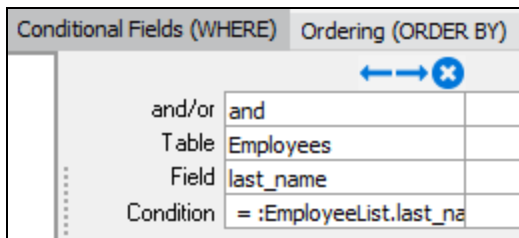
5. Click inside the **Condition** field, then click the ellipsis button  that appears inside the field.
6. Insert an Equals sign (=).
7. Click the **insert variable** icon  to display the variables that exist within the DataBlock.
8. Select 'last\_name'. Recall that EmployeeList is the name of the multi-column list box containing the list of employee names. The two fields within the Employee table are 'last\_name' and 'first\_name'.



9. Click **OK** to continue.

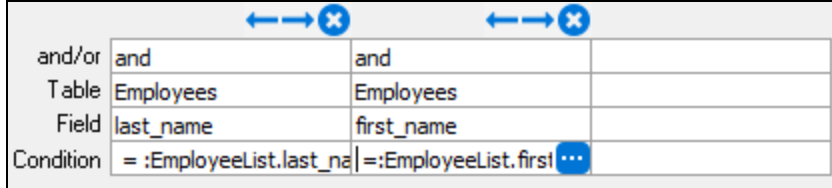


10. Click **OK** to continue.





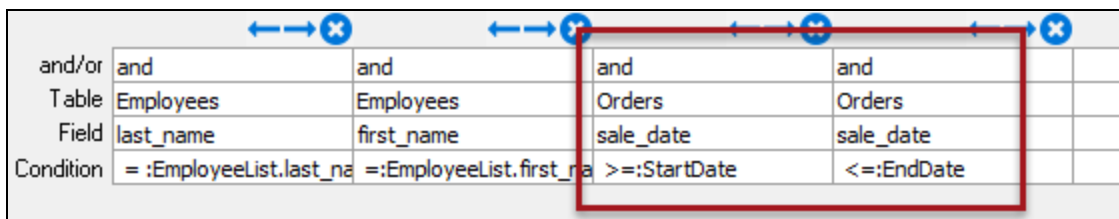
The condition for 'last\_name' has been created, as shown in the above figure.


11. Click on the blank field to the right of the field just created and repeat Steps 4 through 10 for the employee first name.

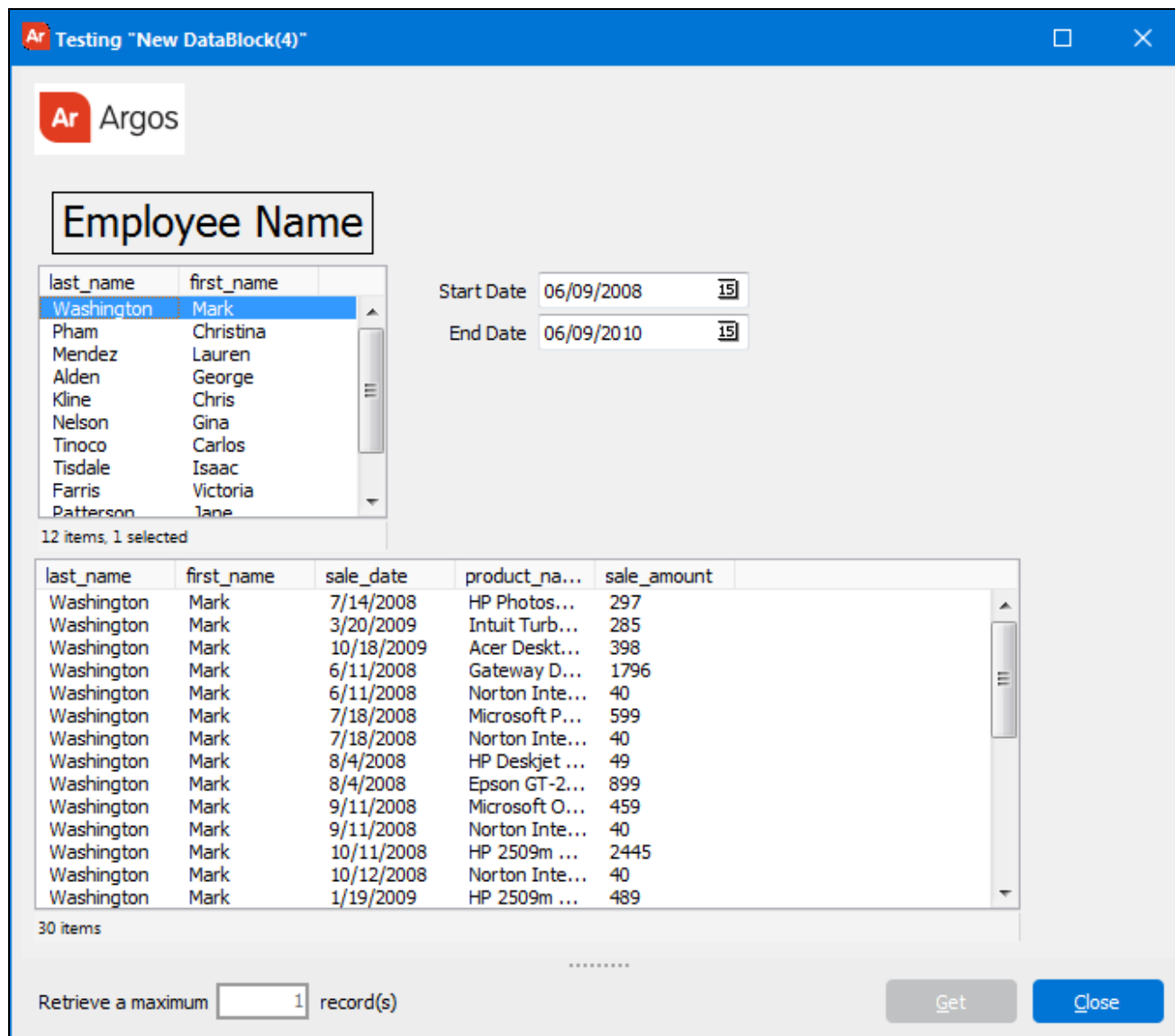


After adding the employee first and last names, we will need to add date conditions using 'sale\_date' from the 'Orders' table, and the 'StartDate' and 'EndDate' variables (which were added to the dashboard in a [previous exercise](#)).

1. Double-click 'sale\_date' in the 'Orders' table to auto-populate the **and/or**, **Table**, and **Field** fields.
2. Click inside the **Condition** field, then click the ellipses button  that appears inside the field to launch the SQL Editor.
3. Enter Greater than or equal to (>=).
4. Click the **insert variable** icon  to see the list of available variables.
5. Select **StartDate** from the variable list.
6. Repeat from Step 12 to add the condition Less than or equal to the End Date (<= :EndDate) to 'sale\_date'.




- Click **OK** and validate the query.
- Click **Finish** to complete the design of the WHERE clause and return to the DataBlock Designer.
- Click **Commit** to save your work, then click the **Test** button  to test the dashboard.



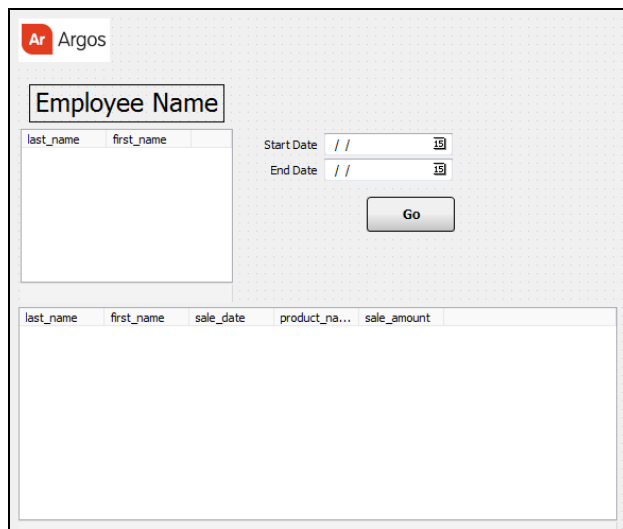
## Use of button to control the execution of the query

When you execute the Dashboard, you will notice that the query results are shown as soon as you finish selecting the employee name and dates, not giving you a chance to change selections before executing the report.

To remedy this situation, we will create a 'Go' button so that you have control over the execution of the query.





- Add a button to the dashboard by clicking the button icon  on the toolbar. A button will then be placed on the form. Move the button to the desired location and adjust the size by using the drag handles.

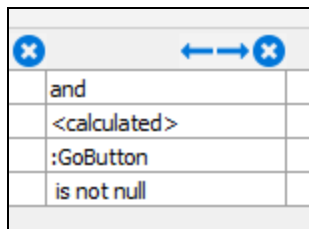




2. Change the variable name of the Button to 'GoButton' on the Properties tab of the DataBlock Designer.

Tab Order	4
Tab Stop	No
Text	Go
Top	176
Variable Name	GoButton
Visible	Yes
Width	83
Word Wrap	Yes

3. Change the text displayed on the button to 'Go' by editing the Text property.
4. Double-click on the object that displays the query results (the multi-column list box at the bottom), click the Build Query icon , then select the "Conditional Fields (WHERE) tab.
5. In the WHERE clause (that was edited in the [previous exercise](#)), click inside the **Table** field in the next empty column to the right.
6. Click the down arrow that appears inside the field and select '<calculated>' from the list.
7. Click inside the **Field** field, then click the ellipses button  that appears inside the field. The SQL Editor will launch.
8. In the SQL Editor, click the **variable** icon  to launch the Pick a variable dialog.
9. Choose **GoButton** from the variable list and click **OK**.
10. Click **OK** to exit the SQL Editor.
11. Click the ellipses button  in the **As** field, and enter `is not null` into the SQL Editor.
12. Click **OK** to return to the Build Query. The condition for the Go button has now been added.



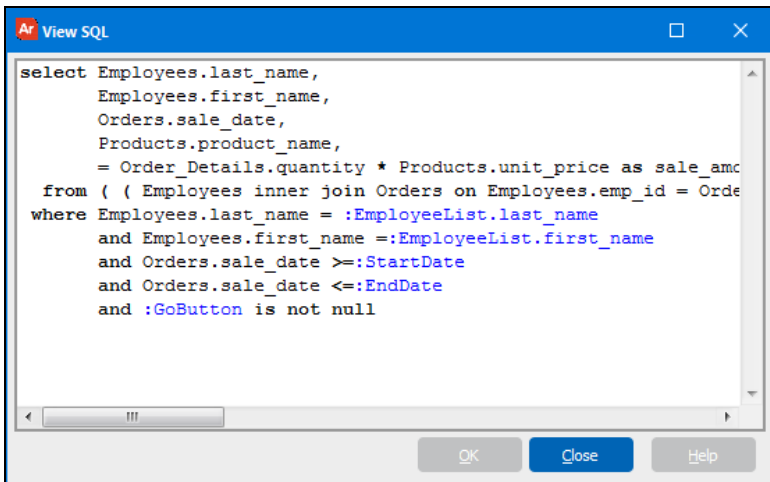
13. Click **OK** and **Finish**, then verify the query to return to the DataBlock Designer.

14. **Commit** your changes and click **Test**  to view the dashboard and test your new button.

Until the button is clicked, the variable representing the state of the button object is null, and the query will not execute. When clicked, the value is no longer null and the query will execute.

## Summary

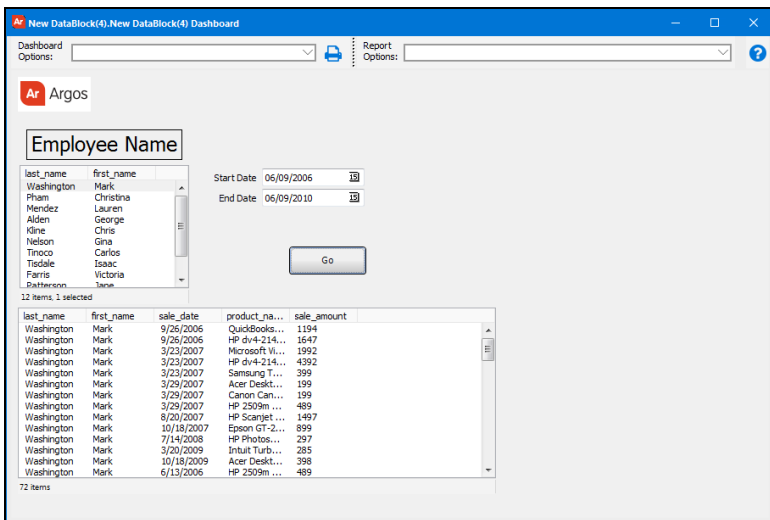
The final SQL for this example appears below. Note the existence of the variable names (in blue) and the fields contained in the select and where statements.



```
select Employees.last_name,
       Employees.first_name,
       Orders.sale_date,
       Products.product_name,
       = Order_Details.quantity * Products.unit_price as sale_amount
from ( ( Employees inner join Orders on Employees.emp_id = Order
where Employees.last_name = :EmployeeList.last_name
      and Employees.first_name =:EmployeeList.first_name
      and Orders.sale_date >=:StartDate
      and Orders.sale_date <=:EndDate
      and :GoButton is not null
```

The form and query design for this example is now complete.

If desired, return to the Argos Main Interface, right-click on the DataBlock and create a dashboard for this DataBlock. The figure below shows the dashboard after execution.



Dashboard Options:  Report Options:

Argos

Employee Name

last_name	first_name
Washington	Mark
Pham	Christina
Mendez	Lauren
Alden	George
Kline	Chris
Nelson	Gina
Tinoco	Carlos
Tisdale	Isaac
Farris	Victoria
Dalsherson	Jose

Start Date: 06/09/2006  
End Date: 06/09/2010

Go

last_name	first_name	sale_date	product_na...	sale_amount
Washington	Mark	9/26/2006	QuickBooks...	1194
Washington	Mark	9/26/2006	HP dv4-214...	1647
Washington	Mark	3/23/2007	Microsoft Vi...	1992
Washington	Mark	3/23/2007	HP dv4-214...	4392
Washington	Mark	3/23/2007	Samsung T...	399
Washington	Mark	3/29/2007	Acer Desk...	199
Washington	Mark	3/29/2007	Canon Cam...	159
Washington	Mark	3/29/2007	HP 2509m ...	489
Washington	Mark	8/20/2007	HP Scanjet ...	1497
Washington	Mark	10/18/2007	Epson GT-2...	899
Washington	Mark	7/14/2008	HP Photos...	297
Washington	Mark	3/20/2009	Intuit Turb...	285
Washington	Mark	10/18/2009	Acer Desk...	398
Washington	Mark	6/13/2006	HP 2509m ...	489

72 items

# Report Query

## Introduction

The previous example showed how to create a DataBlock used by a Dashboard. This example will describe how the same DataBlock can be used with a CSV, Banded, or Extract Report.

Suppose you want to create a CSV, Banded, or Extract report using the same input selection form and query that was used for the Dashboard. The DataBlock created for Example 1 will be used in this example.

## Create a New Report

To create a new report using the same DataBlock, right-click on the DataBlock, then select **New Report or Dashboard**. The dialog box shown in the figure below will appear. Note that the CSV, Banded, and Extract reports are greyed-out. This is because a query does not exist for these reports. **The query created for the Dashboard is not accessible to CSV, Banded, or Extract Reports.** Therefore a **Report Query** must be created since CSV, Banded, and Extract reports use only Report Queries.

**Form Queries and Report Queries:** Within a DataBlock, the queries used by Dashboards are called the Form Queries, and the query used by CSV, Banded, and Extract Reports is called the Report Query.

All reports share the same form.

Create a new report or dashboard

Create a new Argos report or dashboard for the selected DataBlock. After clicking "Create" you will be able to edit the design as well as other details.

Name  
New Report  Shared  Private

Description

Choose the type that Argos should generate

Dashboard Displays the form configured by the creator of the DataBlock. The report query is never executed even if one exists for the DataBlock.

CSV Comma delimited (CSV)

Banded (graphics, text, charts, etc)

Extract text report

Create Cancel Help

## Copy/Paste the Existing Query

To use the same query for CSV, banded, and extract reports, you must copy the form query into the report query:

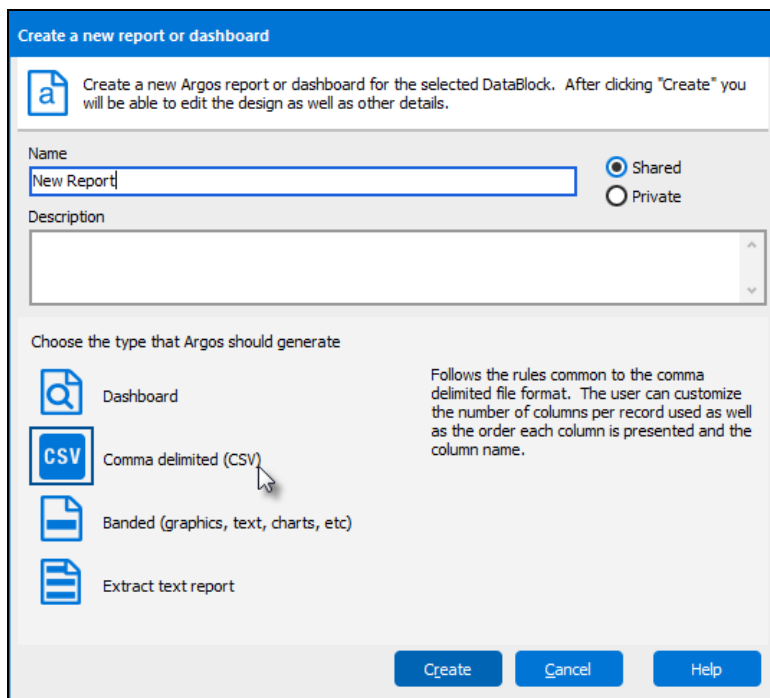
1. Launch the Argos DataBlock Designer by double-clicking on the DataBlock used in Example 1.
2. Double-click the object that displays the query results.

3. Click the hardhat/hammer to launch the Build Query dialog box.
4. Click **Copy** at the top:



5. Click Close to leave the Build Query dialog box.
6. Click Cancel to return to the DataBlock Designer.
7. Click the "Report Query- Visual Design" tab at the upper left of the DataBlock Designer (shown in the above figure).
8. Click **Paste**.
9. The report query now appears in the Report Query - Visual Design Tab, and can be modified if needed.
10. Click Commit then Close to save your work and leave the DataBlock designer.

Create a new report from the main Argos interface. Since the DataBlock now has a report query, the CSV, banded, and extract reports are no longer grayed-out and you may create reports based on this query.



You do not need to create a form query in order to create a report query. If you only need to create a report and not a dashboard, simply go to the Report Query - Visual Design tab and create the query directly.

# Advanced Form Objects and Properties

---

This section describes various advanced Form Objects and properties that are available, including the Data Aware property, creating multiple forms within a DataBlock, Charting, Drilling through reports to obtain greater detail, and OLAP Data Cubes.

[Data Aware Property](#)

[Using Multiple Forms in a DataBlock](#)

[Charting with Multiple Series](#)

[Creating a Filtered OLAP Data Cube](#)

[Drilling Through Reports to Obtain Greater Detail](#)

## Data Aware Property Introduction

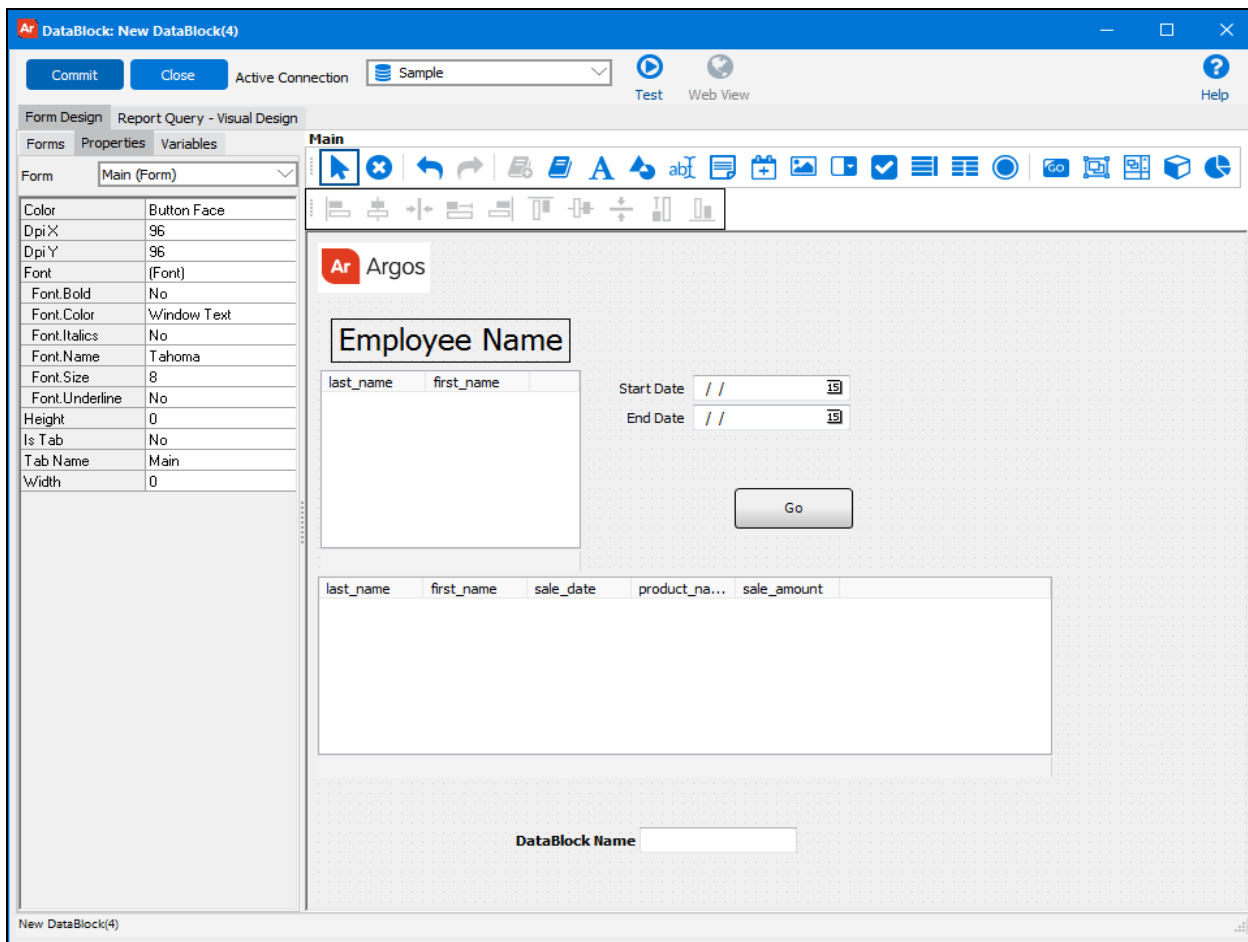
---

The [Data Aware property](#) allows variable data to be used in place of static text for Label, edit box, and Memo Field objects. When an object has the Data Aware property set to True, the Text property becomes a drop down list from which the user can select the data field they need. The dropdown list includes all of the DataBlock variables and SQL variables for the DataBlock.

To illustrate the use of the Data Aware property, the DataBlock used in Example 1 will be expanded to include an edit box to display the name of the DataBlock.

### [Add the Data Aware Object](#)

1. Select the DataBlock created in [Example 1](#), right-click and select **Edit DataBlock**.
2. Resize the multi-column list box to leave room for the new object at the bottom of the form.
3. Add the edit box with its associated label near the bottom of the form as shown in the figure below.



## Set Data Aware property for edit box

1. Click on the "DataBlock name" edit box which displays its properties shown below.
2. Change the **Data Aware** property to **Yes** and click on the Text property. The **Text** property, which is used to enter text by default, becomes a drop-down menu as a result of setting the Data Aware property to **Yes**.
3. Select \$DataBlock.Name from the **Text** property drop-down list. The DataBlock name for this example will appear in the edit box when the report is executed.

### Objects with Data Aware Property

- Label
- Edit Box
- Memo field

## Execute the Report

Executing the Dashboard produces the following report. Note the DataBlock name in the newly added edit box.

The screenshot shows the Argos dashboard interface. At the top left is the Argos logo. Below it is a section titled "Employee Name". This section contains a multi-column list box with columns for "last\_name" and "first\_name". The list includes names like Washington, Mark; Pham, Christina; Mendez, Lauren; Alden, George; Kline, Chris; Nelson, Gina; Tinoco, Carlos; Tisdale, Isaac; Farris, Victoria; and Patterson, Jane. To the right of the list box are two date input fields: "Start Date" set to "6 /6 /2006" and "End Date" set to "6 /9 /2010". Below these is a "Go" button. Underneath the list box is a table with the following data:

last_name	first_name	sale_date	product_na...	sale_amount
Washington	Mark	9/26/2006	QuickBooks...	1194
Washington	Mark	9/26/2006	HP dv4-214...	1647
Washington	Mark	3/23/2007	Microsoft Vi...	1992
Washington	Mark	3/23/2007	HP dv4-214...	4392
Washington	Mark	3/23/2007	Samsung T...	399
Washington	Mark	3/29/2007	Acer Deskt...	199
Washington	Mark	3/29/2007	Canon Can...	199
Washington	Mark	3/29/2007	HP 2509m ...	489
Washington	Mark	8/20/2007	HP Scanjet ...	1497
Washington	Mark	10/18/2007	Epson GT-2	899

At the bottom of the dashboard, there is a "DataBlock Name" field containing the text "New DataBlock(4)". A red arrow points to this field.

## Summary

The system variable containing the DataBlock name was chosen to populate the edit box; however other variable types that exist in the DataBlock could have been used as well. For instance, the fields within the multi-column list box (see figure below) exist as Argos variables, and are included in the drop-down for Data Aware objects. This list is displayed when clicking the Text Property for the Data Aware object.

Text	\$DataBlock.Name
Top	EndDate
Variable Name	GoButton
Visible	MultiColumn1.first_name
Width	MultiColumn1.last_name
	MultiColumn1.product_name
	MultiColumn1.sale_amount
	MultiColumn1.sale_date
	StartDate

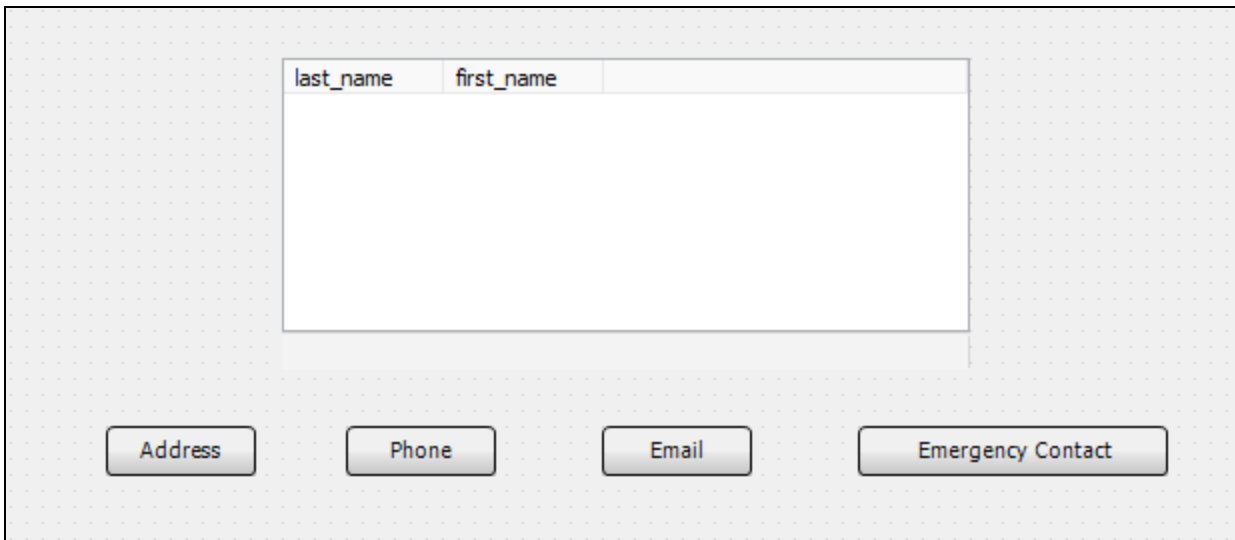
# Using Multiple Forms in a DataBlock

## Introduction

[Example 1](#) demonstrated how to create a Dashboard in which the input selections and query results are displayed on the same form. There are situations where you will want to display report results on a different form than the form where input selections were made. This can be accommodated in Argos and will be demonstrated in this example.

This simple example is used to quickly obtain contact information for an individual employee within a company. A list of employees is presented from which the user can select one from the list. The user can then choose which of the three types of contact information to display. The type of contact and the location of the information in the sample database are shown in the table below.

<b>Contact type</b>	<b>Database table</b>
<b>Home address</b>	<b>Employees</b>
<b>Phone (home and mobile)</b>	<b>Employee_Phone</b>
<b>email</b>	<b>Employee_Email</b>
<b>Emergency contact (name, relationship, phone)</b>	<b>Employee_Emerg_Contact</b>



The screenshot shows a web form interface. At the top, there is a multi-column list box with headers 'last\_name' and 'first\_name'. Below the list box, there are four buttons: 'Address', 'Phone', 'Email', and 'Emergency Contact'.

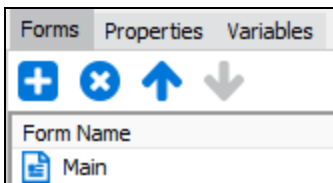
During execution of the form, the user selects an employee name from the multi-column list box, then clicks the appropriate button to display the type of contact information to be provided. After clicking the button, another form will be displayed containing the corresponding contact information.

Five forms need to be created for this example. The form above is named "Main". The remaining forms will be named "Address\_Form", "Phone\_Form", "Email\_Form", and "Emergency\_Form".




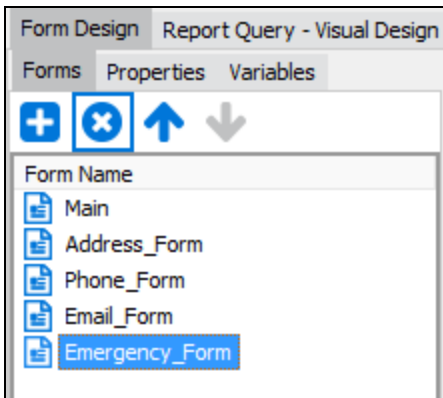
## [Create the DataBlock](#)

The first step is to create a DataBlock and launch the [DataBlock Designer](#), then click on the Forms tab.



## [Add the Form Names](#)

The default name of the form that appears when launching the DataBlock Designer is "Main" as shown in the above figure. Click the blue plus button  to create the additional four forms. The default names are Form1, Form2, Form3, and Form4. Right-click on each form name and rename them to Address\_Form, Phone\_Form, Email\_Form, and Emergency\_Form as shown below. You can use the up and down arrows to reorder the forms on the tab.



Click "Main" under the Forms tab to design the main form.

## Create the Main Form

Within the DataBlock Designer, create the main form containing a multi-column list box for the employee names using the same method as in [Example 1](#). Then create button objects named `Address_button`, `Phone_button`, `Email_button`, and `Emergency_button`.

## Edit Properties of the Buttons

The properties for the Address button are as follows:

Background	(Background)
Background.Color	Button Face
Background.Style	Default
Border Color	Black
Cursor	Default
Data Aware	No
Description	
Enabled	Yes
Font	(Font)
Font.Bold	No
Font.Color	Window Text
Font.Italics	No
Font.Name	Tahoma
Font.Size	8
Font.Underline	No
Height	25
Left	714
On Click	(none)
Parent Font	Yes
Tab Order	1
Tab Stop	No
Text	Button
Top	230
Variable Name	Button1
Visible	Yes
Width	75
Word Wrap	Yes

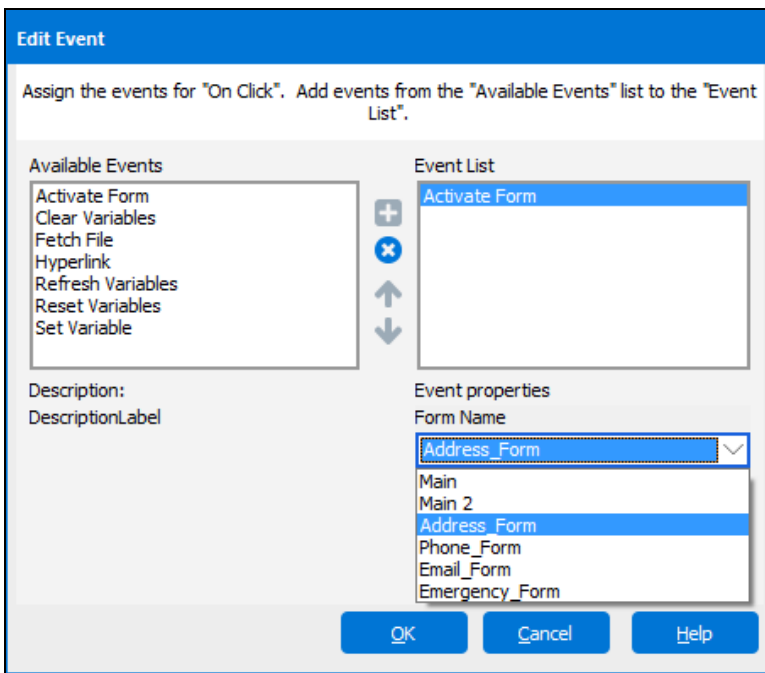
Note the Variable Name property is set to "Address\_button", which is the name of the button object. Similarly, the Text property is set to "Address", which is the text displayed on the button.

When the report is executed, we want to launch the Address form when the **Address** button is clicked. The Address form will show the address information for the selected employee on the main form. The **On Click** property is used to execute the appropriate form when this button is clicked.

Click in the **On Click** property, then click the ellipses button  to launch the Edit Event dialog.

The Edit Event dialog allows you to select which form to present when this button is clicked. Select "Address\_Form" to launch the Address Form when the **Address** button is clicked.

**The On Click Property:** This property is available for buttons, shape objects, static labels, graphics, and chart objects.




Note that various events can be activated by the "On Click" event. For this example only the "Activate Form" event is used. Click the **Help** button to see a description of all event types.

Repeat this process for each of the buttons.

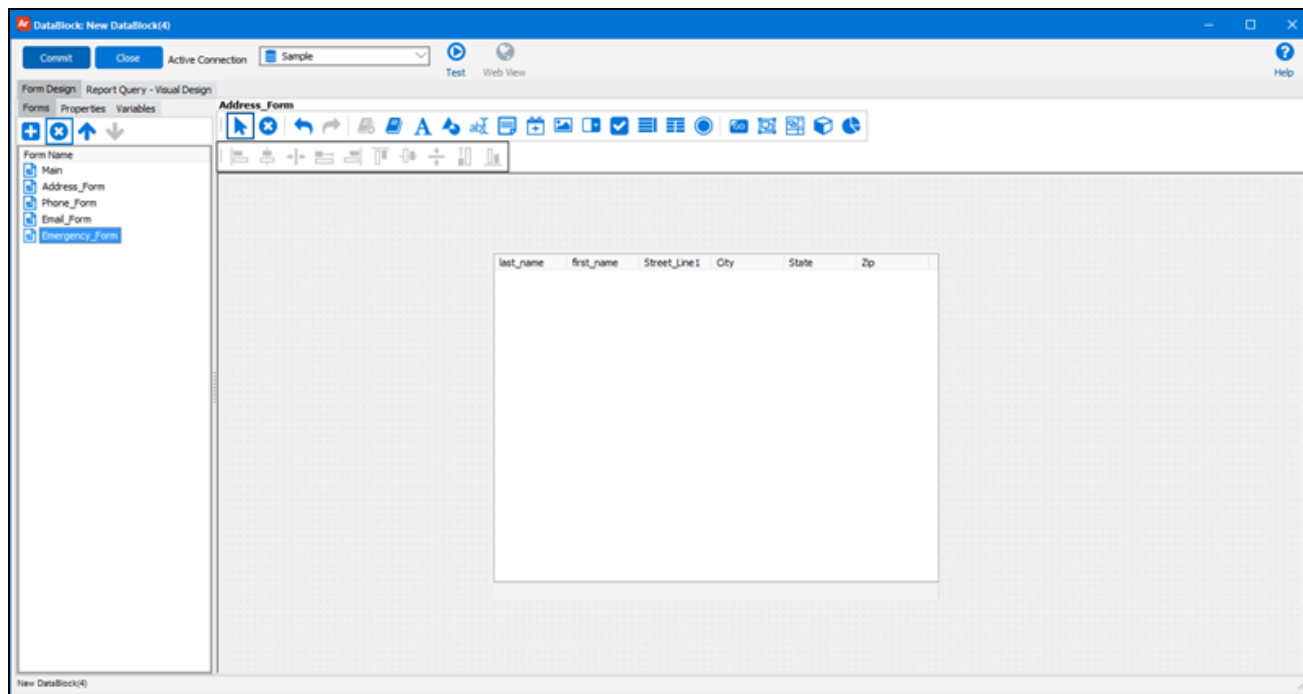
Once you have set the properties for the remaining buttons, the final step is to create the form for each button.

## [Create the form/query associated with each button](#)

To create the Address form:

1. Click on the Forms tab within the DataBlock Designer, then
2. Select the Address form.
3. The Design Area of the DataBlock Designer will be blank. You can now place a multi-column list box object  to contain the results of the query.

- Use the Employees Table for the query, and include the desired fields ('last\_name', 'first\_name', 'Street', 'City', 'State', and 'Zip').

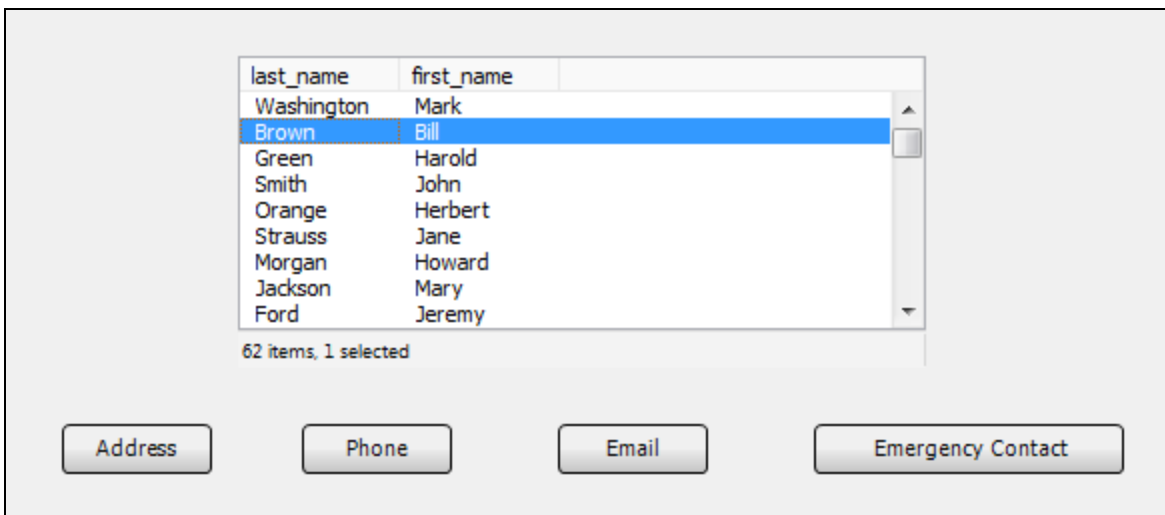


- After the query for the Address form is complete, click on the "Phone\_form" form in the Forms tab to design the form to display employee phone numbers.
- Again, you will see the Design Area blank. As was done above, create a multi-column list box to contain the phone numbers obtained from the Employee\_Phone table.
- You will need to [join](#) the Employees table with the Employee\_Phone table and [create a WHERE clause](#) such that the query finds the phone number only for the employee selected on the Main Form.
- Repeat this procedure to create the forms for the **Email** and **Emergency** buttons.

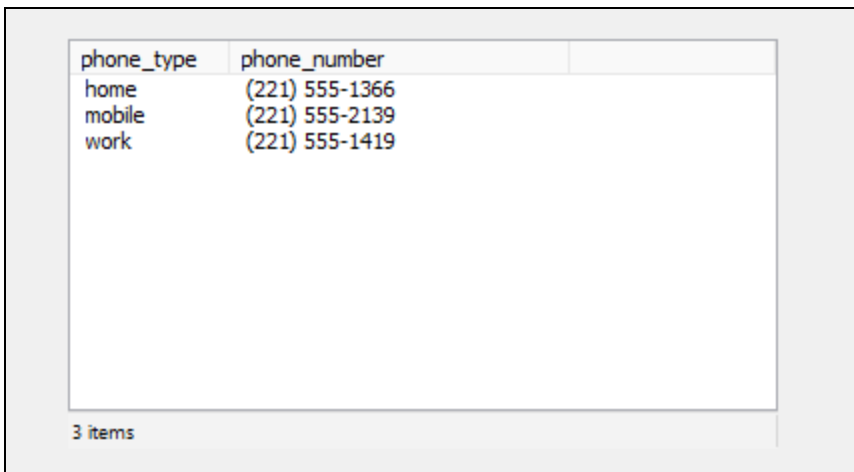
## Summary

The multi-form creation process is now complete. Buttons were created to activate associated forms, and each type of query result is displayed on its own form.

To show how the above report design appears during execution, the figure below shows Bill Brown as the employee selected.



After clicking the **Phone** button, the following report is displayed on the Phone form, and lists the various phone numbers for Bill Brown.



**Tip:** You can also use tabs to navigate between forms instead of buttons. Refer to the in-product Help for a description of how the **Is Tab** property is used to create a tabbed interface.

## Charting with multiple series

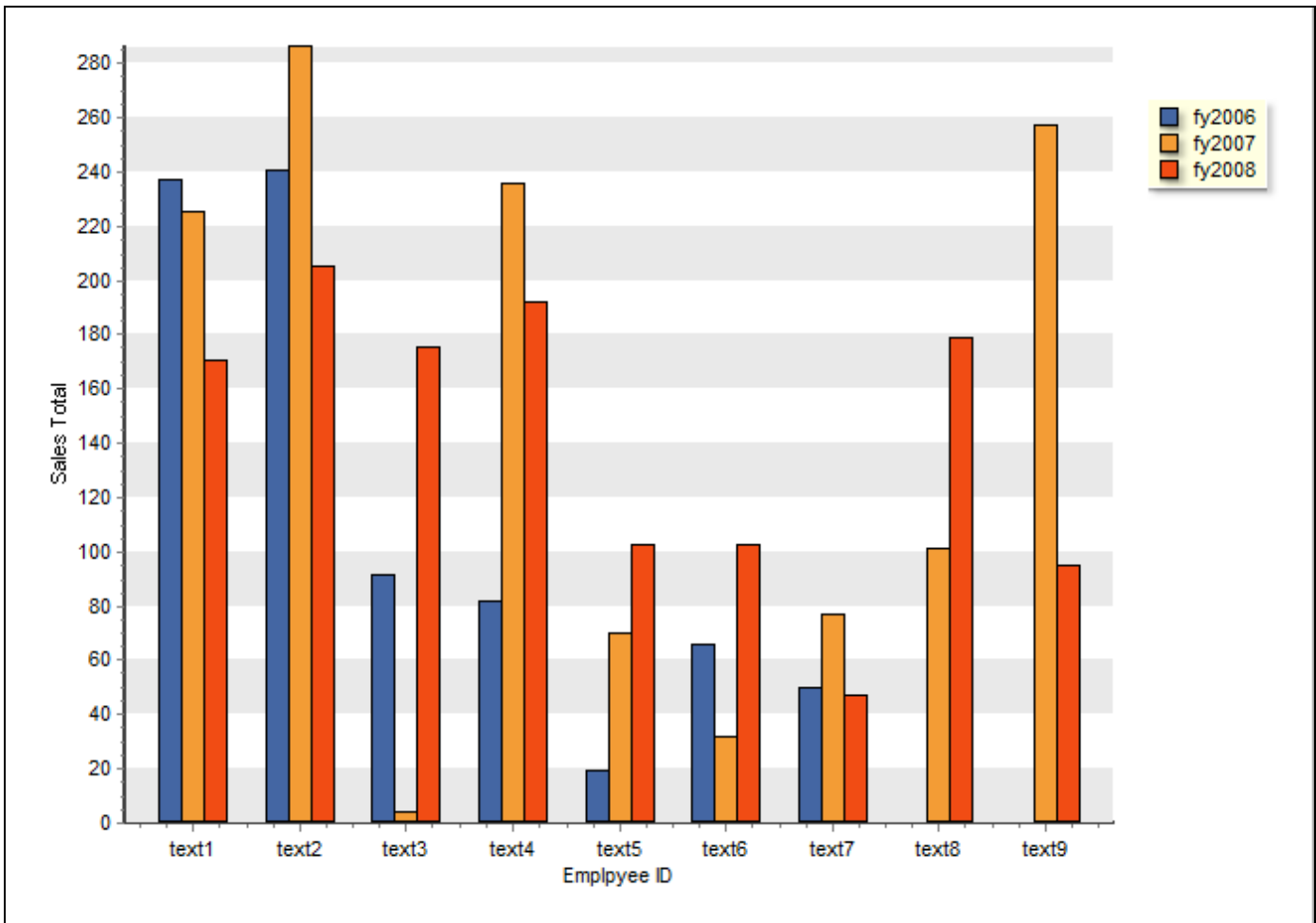
### Introduction

This example demonstrates how to create the chart shown below that requires multiple series of data and datasets.

The chart displays sales totals for individual employees as a bar chart with each color representing sales for a fiscal year. The chart includes sales from fiscal years 2006 – 2008. This chart does not require input selections from the user.

A series of data will be created for each fiscal year, with a dataset associated with each series.

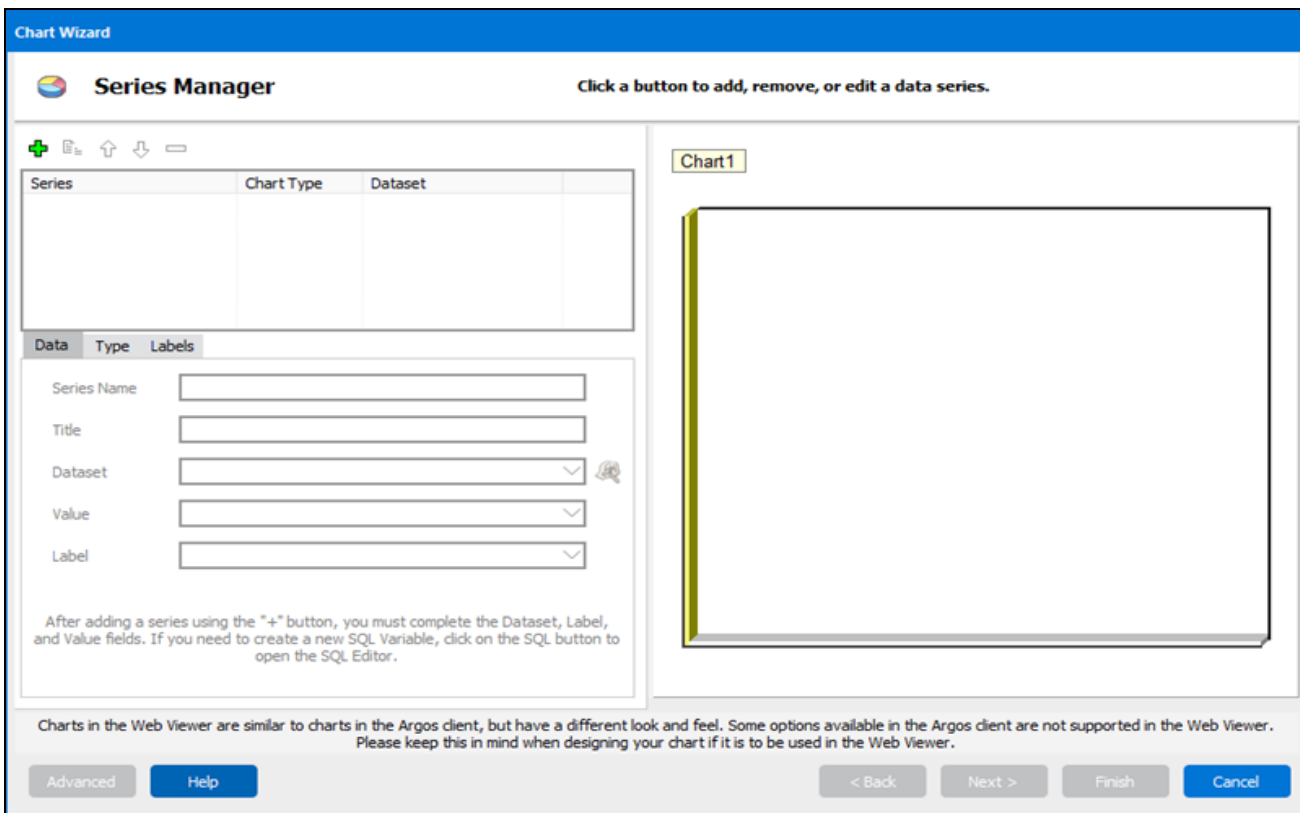
The chart uses the same database tables and similar DataBlocks to those used in the [Summing and Grouping](#) example (found later in this document under Advanced Query Techniques – Example 8) that reports sales totals grouped by employee ID. Please read the Summing and Grouping example before proceeding further with this example.





## [Create the Chart Object](#)

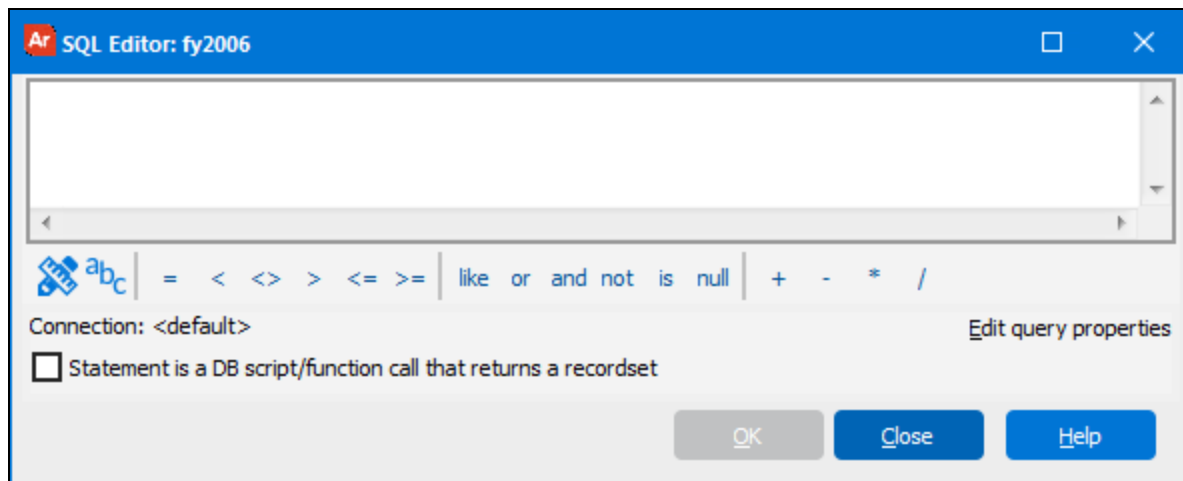
The Chart Wizard was [demonstrated](#) in the [Argos Report Writer Guide](#). It is recommended that you go through that guide first, if you have not already done so. For this example, we will be using a classic Argos chart. For information on how to create a standard chart, please refer to the [Chart Wizard](#) section of the Argos Help.

1. Click the chart icon, then click within the Design Area to create the chart object. Expand the object to the desired size.
2. Double-click the object to launch the Chart Wizard.



## Name the first series

1. Click the green plus icon  to create a series.
2. Change the Series name to fy2006
3. Click the hardhat/hammer  to create a new dataset.
4. Enter 'fy2006' as the name of the dataset and click **OK**.
5. Select **SQL Statement** and click **OK** to launch the SQL Editor.





6. Click the Build Query icon  to launch the Build Query dialog box.

## Creating the new dataset for the first series

Within the Build Query dialog box, create the query shown in the figure below using the Employees, Orders, Order\_Details, and Products tables that were used in the [Summing and Grouping](#) example.

The SELECT clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Aggregate Conditionals (HAVING)	Ordering (ORDER BY)																								
 Distinct  Summing	<table border="1"> <tr> <td>Table</td> <td>&lt;calculated&gt;</td> <td>Employees</td> <td></td> </tr> <tr> <td>Field</td> <td>Products.unit_price *Order_De</td> <td>emp_id</td> <td></td> </tr> <tr> <td>Type</td> <td>float</td> <td>string</td> <td></td> </tr> <tr> <td>As</td> <td>total_sales</td> <td></td> <td></td> </tr> <tr> <td>Description</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Summing</td> <td>Sum</td> <td>&lt;Group By&gt;</td> <td></td> </tr> </table>	Table	<calculated>	Employees		Field	Products.unit_price *Order_De	emp_id		Type	float	string		As	total_sales			Description				Summing	Sum	<Group By>			
Table	<calculated>	Employees																									
Field	Products.unit_price *Order_De	emp_id																									
Type	float	string																									
As	total_sales																										
Description																											
Summing	Sum	<Group By>																									

The WHERE clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Aggregate Conditionals (HAVING)	Ordering (ORDER BY)																
<root>	<table border="1"> <tr> <td>and/or</td> <td>and</td> <td>and</td> <td></td> </tr> <tr> <td>Table</td> <td>Orders</td> <td>Orders</td> <td></td> </tr> <tr> <td>Field</td> <td>sale_date</td> <td>sale_date</td> <td></td> </tr> <tr> <td>Condition</td> <td>&gt;=#01/01/2006#</td> <td>&lt;=#12/31/2006#</td> <td>...</td> </tr> </table>	and/or	and	and		Table	Orders	Orders		Field	sale_date	sale_date		Condition	>=#01/01/2006#	<=#12/31/2006#	...		
and/or	and	and																	
Table	Orders	Orders																	
Field	sale_date	sale_date																	
Condition	>=#01/01/2006#	<=#12/31/2006#	...																

**Tip:** The use of the '#' delimiter before and after the date is the required syntax for MS-Access (which uses the Jet database engine). Other databases may require a different delimiter.

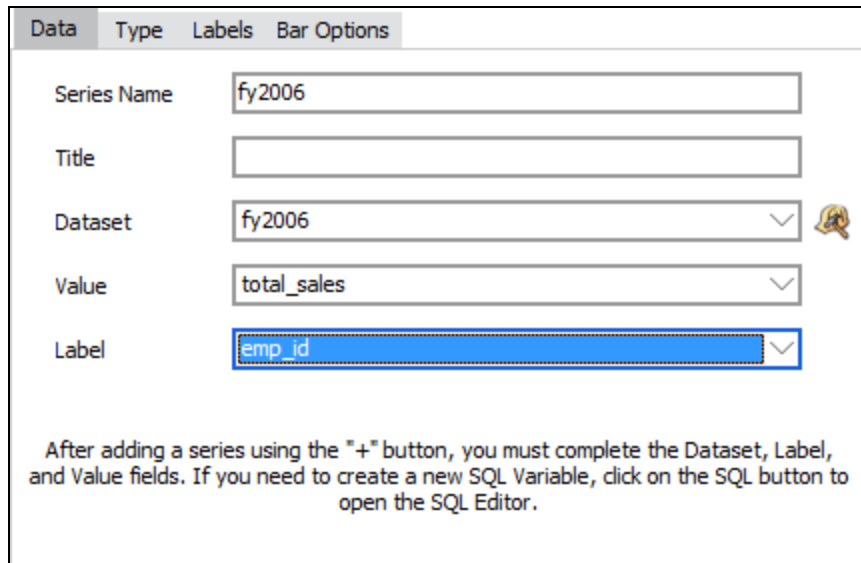
This completes the query for the fy2006 dataset. Later, queries will be developed for fy2007 and fy2008 with the only difference being in the WHERE clause where the corresponding date range will be changed. The queries for years 2007 and 2008 will be pasted from the first query that was developed for fy2006. After pasting, the date range shown above will be modified to reflect the corresponding fiscal year.

Click **OK** and validate the query and return to the Chart Wizard.




## Select the Chart Types, labels, and other options

1. On the Data tab of the Chart Wizard, Select 'total\_sales' for the Value field.
2. Select 'emp\_id' for the Label field.



Series Name:

Title:

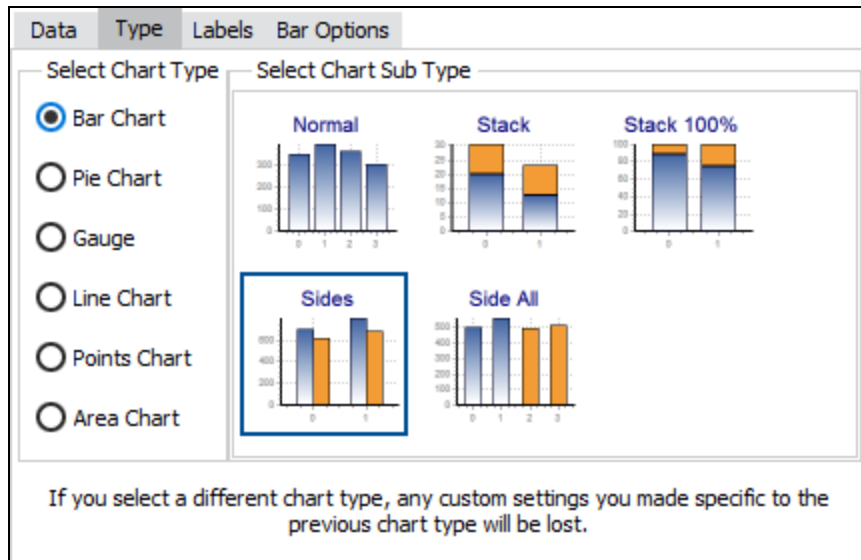
Dataset:  

Value:

Label:

After adding a series using the "+" button, you must complete the Dataset, Label, and Value fields. If you need to create a new SQL Variable, click on the SQL button to open the SQL Editor.

3. Under the **Type** tab, select the Bar Chart radio button and click on the "Sides" Chart Sub Type.



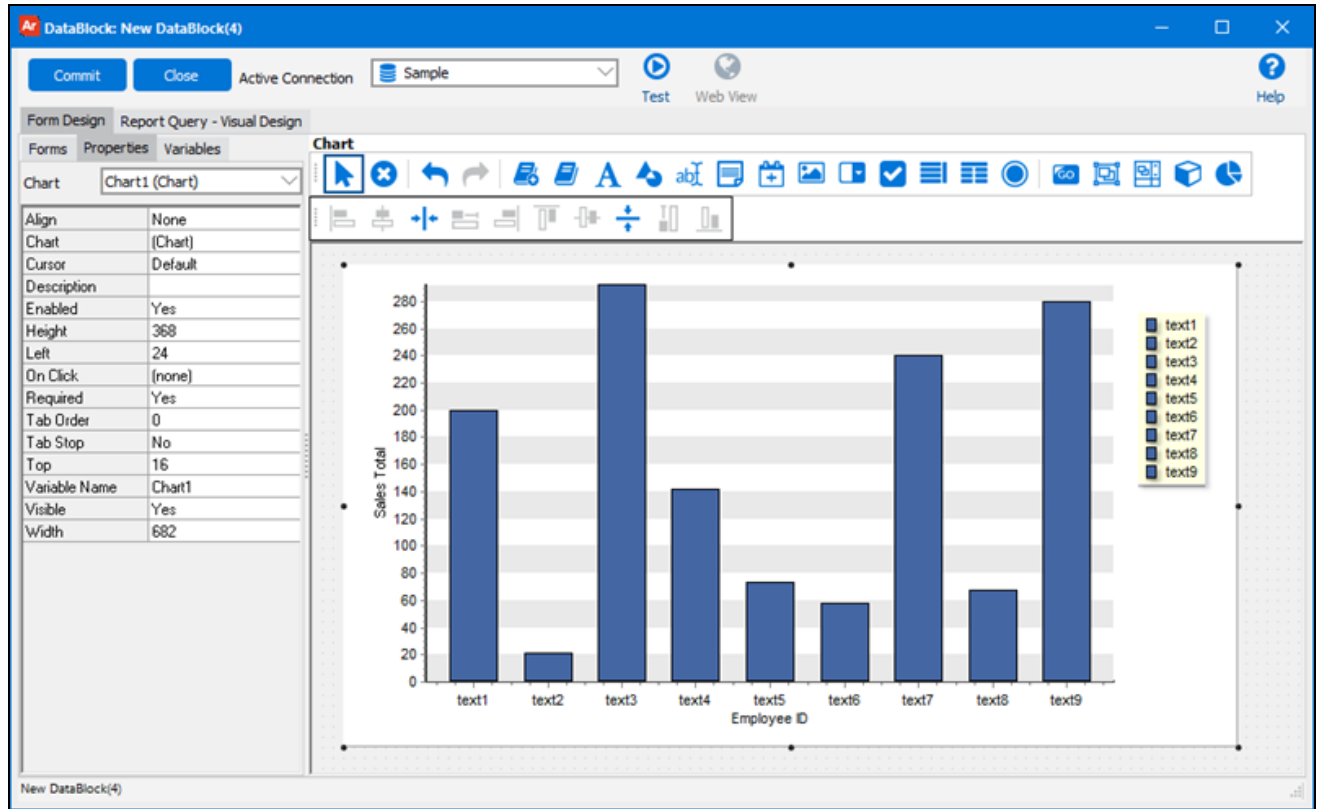
Select Chart Type:  Bar Chart,  Pie Chart,  Gauge,  Line Chart,  Points Chart,  Area Chart

Select Chart Sub Type: Normal, Stack, Stack 100%, Sides, Side All

If you select a different chart type, any custom settings you made specific to the previous chart type will be lost.

4. Under the **Labels** tab, uncheck the Visible Label box. This removes the labels that are on the bars.
5. Click the **Next** button to view the **Chart Theme and Panel** selections. Choose whatever theme and palette you desire.
6. Click the **Next** button and enter the titles. This example used "Sales Total" for the vertical axis title and "Employee ID" for the horizontal axis title.

7. Click the **Finish** button to complete the design of the first series. This brings you back to the Argos DataBlock Designer.






8. **Commit** and **Test** if you desire.




This completes the series and dataset for fiscal year 2006. When you are ready to create the chart for the next series, double-click on the chart to launch the Chart Wizard again.

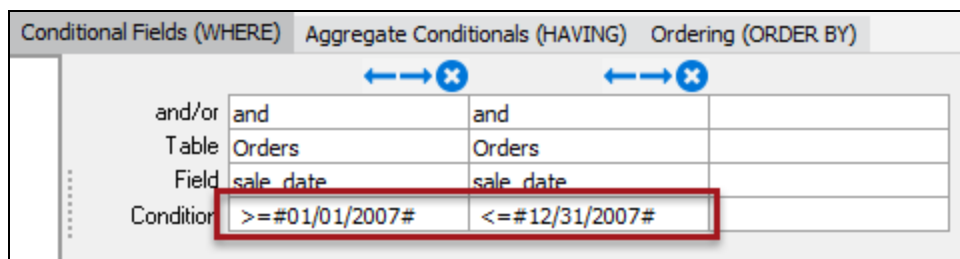
### Create the next series for fy2007

Before beginning the design for fy2007 you need to copy the fy2006 dataset.

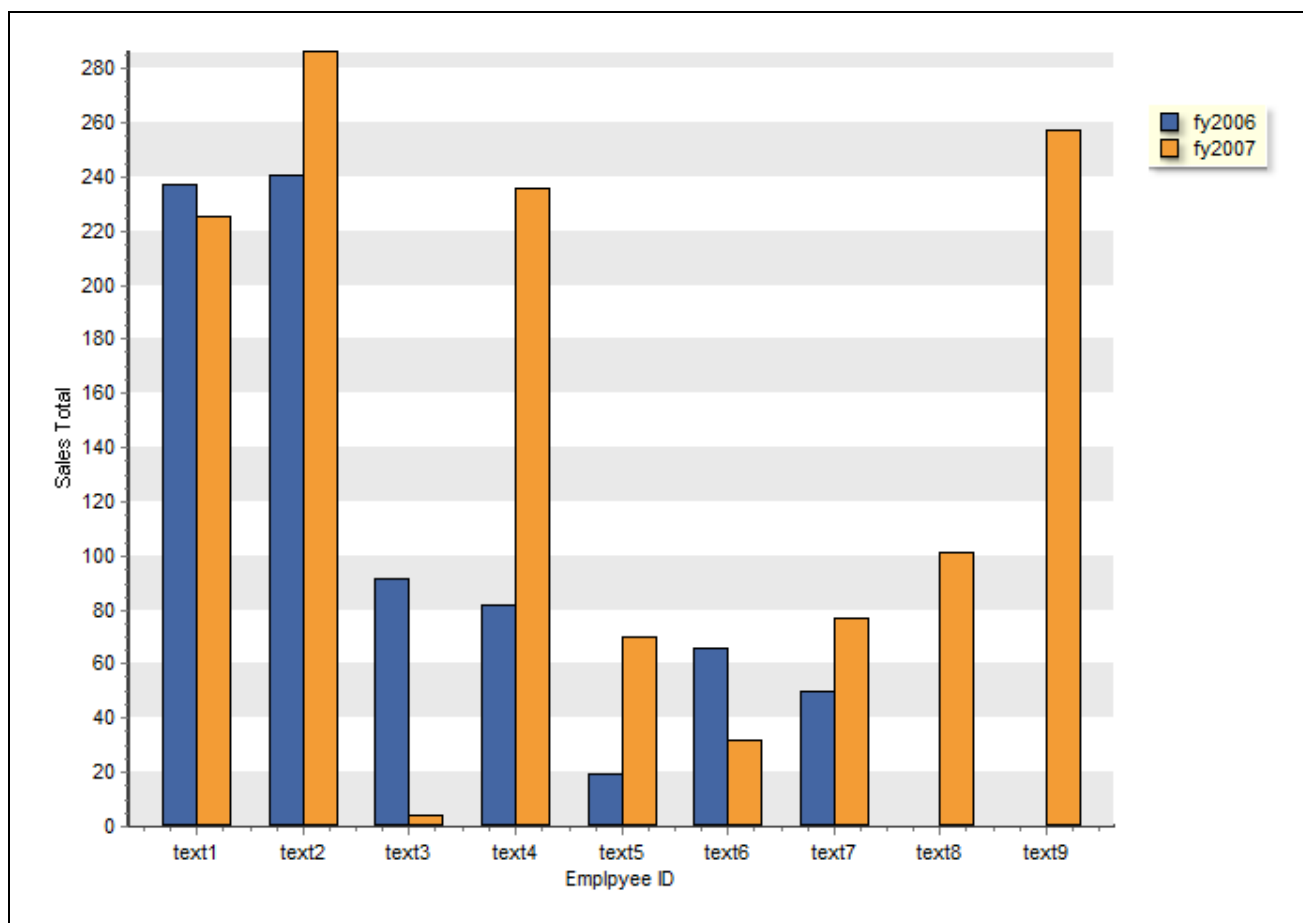
1. Click fy2006 under "Series" and click the hardhat/hammer icon  to launch the SQL Editor.
2. Click **Edit Visual Design** to launch the Build Query dialog.
3. Click the **Copy** icon  at the top of the dialog box.
4. Click **Close** to return to the SQL Editor. Click **Close** again to return to the Chart Wizard.
5. Click the green plus icon  to add a new series.
6. Name the new series 'fy2007'.
7. Select fy2007 from the list of series at the top of the screen.

Series	Chart Type	Dataset
fy2006	Bar (Sides)	fy2006
fy2007	Bar (Sides)	Undefined

8. In the Dataset field, select "Create a new dataset..." then click the hardhat/hammer icon .
9. Name the new dataset 'fy2007' and click **OK**.
10. Select **SQL Statement** and click **OK** to launch the SQL Editor.
11. Click the Build Query icon  to launch the Build Query editor.
12. Click **Paste**  to paste the dataset created for fy2006.
13. Within the Build Query dialog box, click the WHERE tab and change the year in each date from 2006 to 2007.



14. Click **OK** to exit the Build Query editor. Click **OK** again and validate the query to return to the Chart Wizard.
15. Select total\_sales for the **Value** and select emp\_id for the **Label**.
16. Under the **Type** tab, select Bar Chart and Sides for the type and sub type, respectively.
17. Click the **Labels** tab and uncheck 'Visible Label'.
18. Click **Next** and adjust the colors and theme as desired. The bars for the second series should be a different color to distinguish them from the first series.

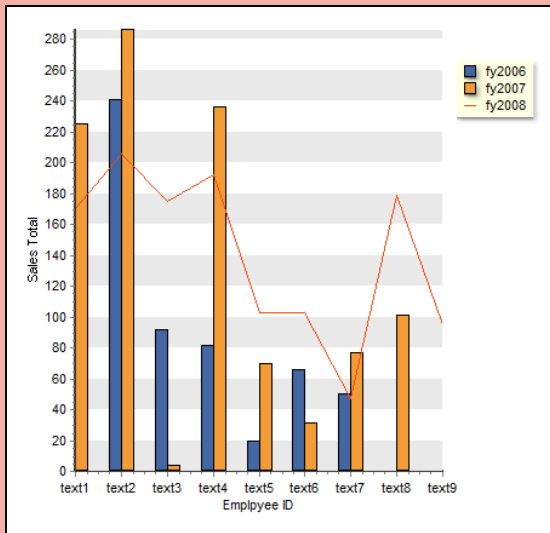


The dataset for fy2007 is now complete. Repeat this process for the third dataset (fy2008).

## Summary

This example illustrated how to create several data series to be displayed on one chart and how to associate datasets with each series. Save this as a Dashboard.

**Note:** Do not mix chart styles among series on the same chart. For instance, if series 1 is a bar chart and series 2 is a line chart, both styles will appear in the plot area, which is not desirable.



**Note:** Be sure that the correct Series is highlighted when editing otherwise you may be unintentionally changing a Series that you didn't want to change.

## Creating a Filtered OLAP Data Cube

### Introduction

One of the unique features of Argos is the ability to create filtered OLAP Data Cubes. That is, information from a data source can be filtered before providing the information to the OLAP cube for analysis. Also, OLAP cubes can be built off of ANY data source – you are not dependent on a data warehouse. The underlying theory for OLAP Cubes is discussed in the Argos Report Viewer Guide, so it will not be discussed here.

This example will create an OLAP Data Cube consisting of sales information contained in the sample database. Items of interest include sales totals by region, by employee, by date, and by product type for all sales transactions.


Data will be filtered by a date (specified by the Report Viewer) before it is sent to the cube for analysis. Even though date ranges could be filtered within the cube, for the sake of example, data will also be filtered by a date provided on the input selection form. Only sales after the date entered on the form will be made available to the OLAP cube.

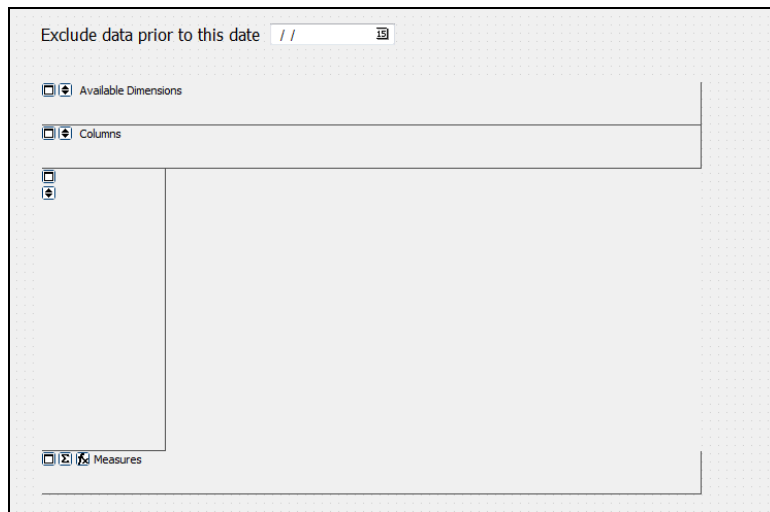
**Tip:** There are numerous sources that can be found on the web that discuss the underlying technology for OLAP.

## Create the OLAP Object in the DataBlock Designer

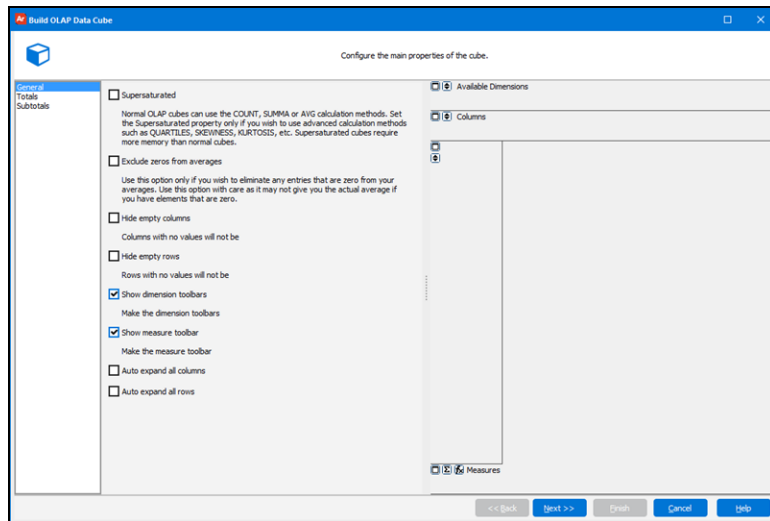
1. Create a DataBlock in the Argos Explorer tree and select the appropriate connection. Edit the DataBlock which launches the Argos DataBlock Designer.
2. In the DataBlock Designer, add a date field which will be used to filter the data. The date will be used to exclude sales records prior to the date entered on the form.



3. Click the OLAP cube object  in the toolbar, then click anywhere in the Design Area. The skeleton of the OLAP cube will appear as shown in the figure below. Click on the corners of the cube to expand the size as desired.




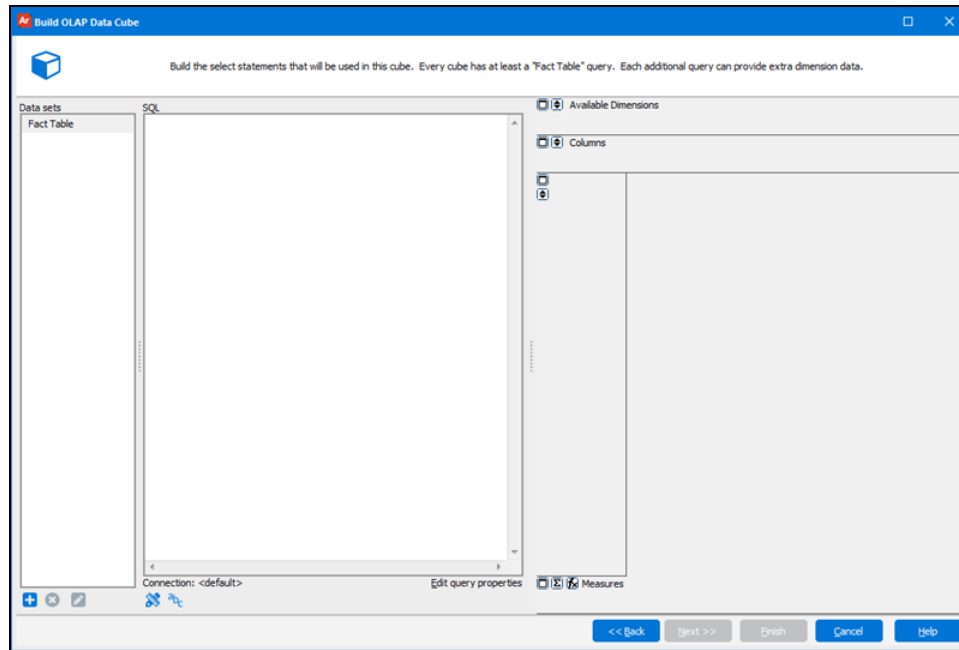
4. Double-click inside the cube to configure the [main properties of the cube](#).



5. In this simple example, the default selections will be used. Click the **Next** button to [build the Fact Table](#).

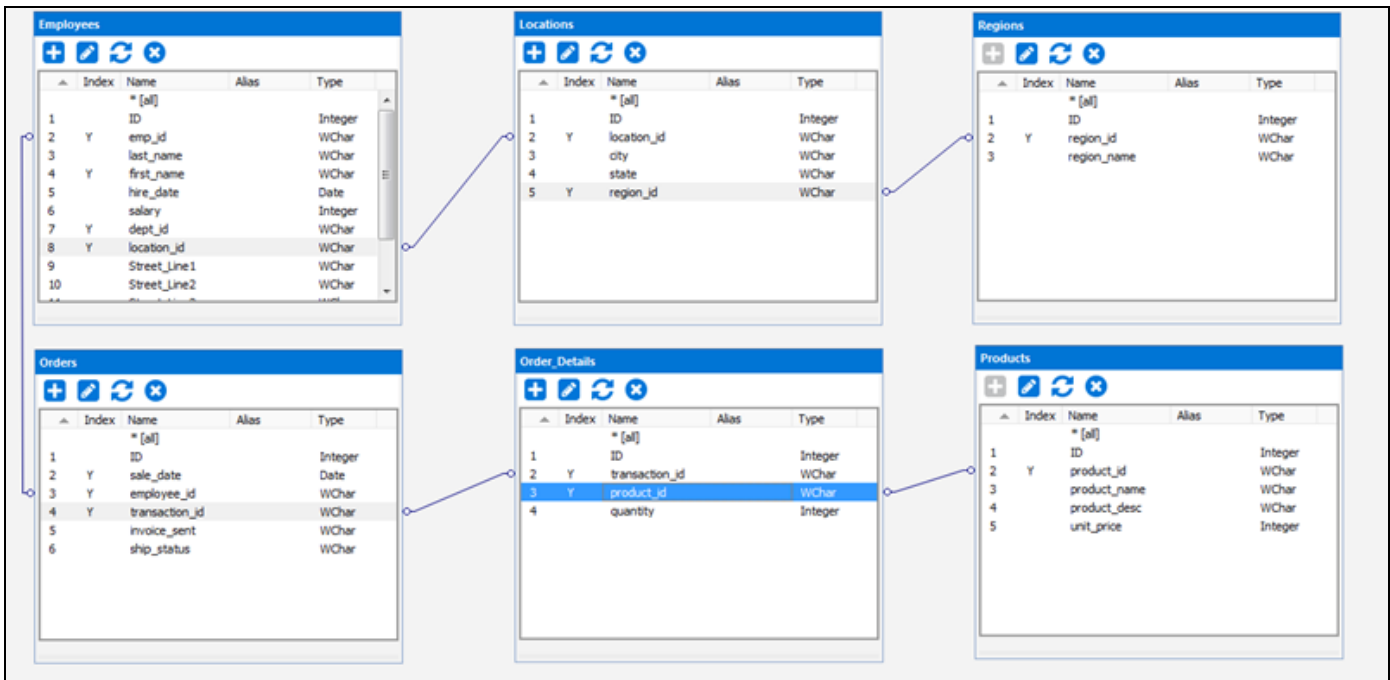
## Build the Fact Table (the query)


1. On the Fact Table screen of the Build OLAP Cube dialog, click on the Build Query  icon at the lower left to launch the Build Query dialog box that you have seen in previous examples.





1. Build the query using the Employees, Locations, Regions, Orders, Order\_Details, and Products tables.
2. Join the tables as follows:
  - `Employees.emp_id` with `Orders.employee-id`
  - `Employees.location_id` with `Locations.location_id`
  - `Locations.region_id` with `Regions.region_id`
  - `Orders.transaction_id` with `Order_Details.transaction_id`
  - `Order_Details.product_id` with `Product.product_id`



Your tables should look like the following:



3. In the **SELECT** tab, enter the following fields by double-clicking them within their tables: Employees.last\_name, Orders.sale\_date, Products.product\_name, and Regions.region\_name.
4. In the last column, select <calculated> in the **Table** field.
5. Click the ellipses button  to launch the SQL Editor.
6. Multiply the unit price by the quantity to get the total sales in the <calculated> column:  
`Products.unit_price *Order_Details.quantity`
7. Click **OK** to return to the Build Query editor.
8. Enter 'sales\_total' in the **As** field in the <calculated> column.
9. Your SELECT clause should now look like this:

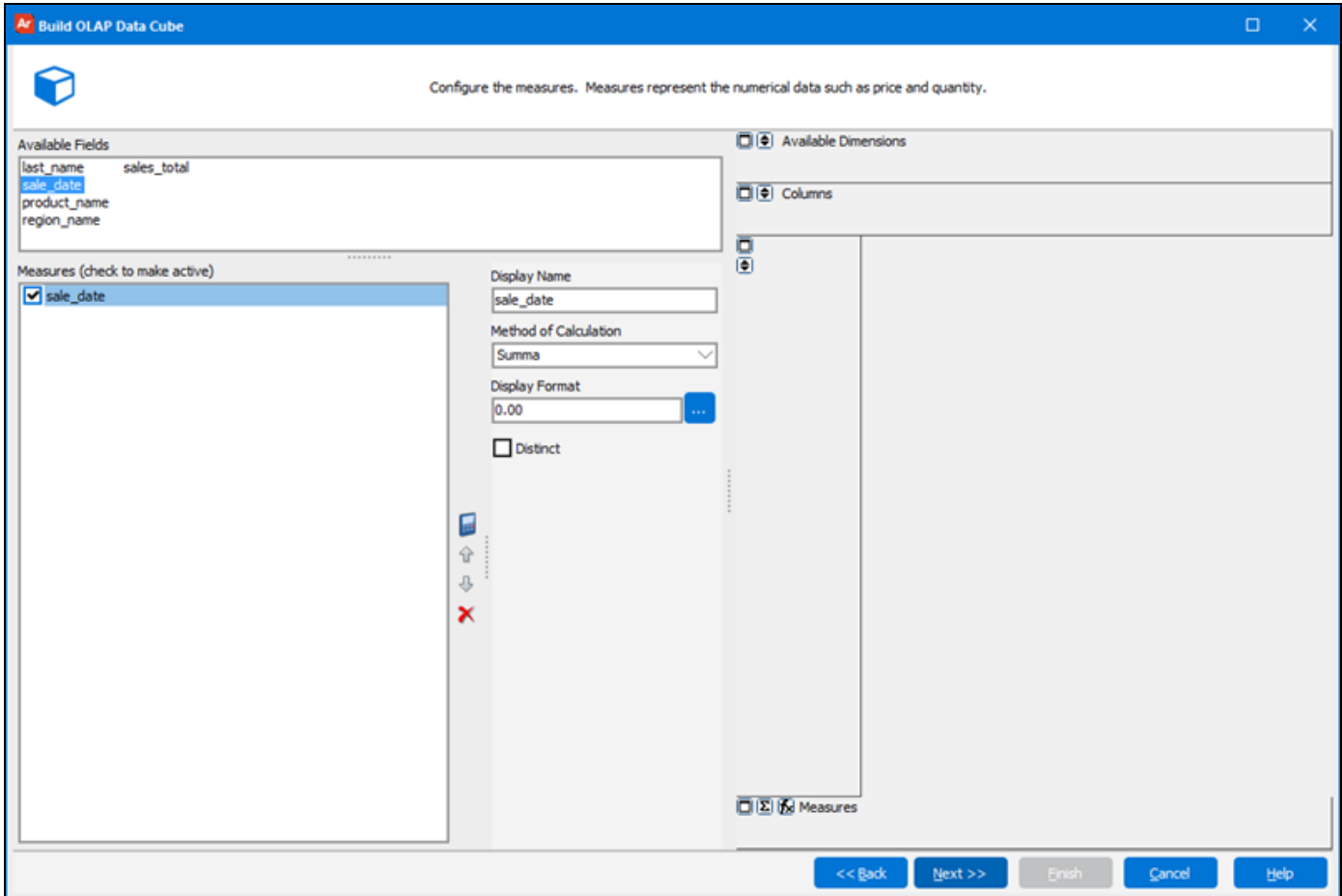
	Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)		
Distinct 	Table: Employees	Orders	Products	Regions	<calculated>
	Field: last_name	sale_date	product_name	region_name	Products.unit_price *Order_De
	Type: string	date/time	string	string	string
Summing 	As:				sales_total
Description					

To set a condition to find only records where the sale\_date is greater than the date input by the user:

10. Click the WHERE tab.
11. In the Orders table, double-click the sale\_date field to autofill the query fields.
12. Click the ellipses button  in the **Conditions** field to launch the SQL Editor, and enter > (greater than) and click the user-defined variable button  and select the date object you placed on your dashboard.
13. Click **OK** to return to the Build Query editor.
14. Click **OK** again to return to the OLAP cube builder.
15. Click **Next** to continue.

## Identify the Measures

You will now identify which of the fields above will be used as measures. For this example, sales\_total is the only field used as a measure. Double-click on the field to move it from the Available Fields window into the Measures window as shown below. For this example the **Method of Calculation** will use the default of Summa and the Display Format will be the default as well.

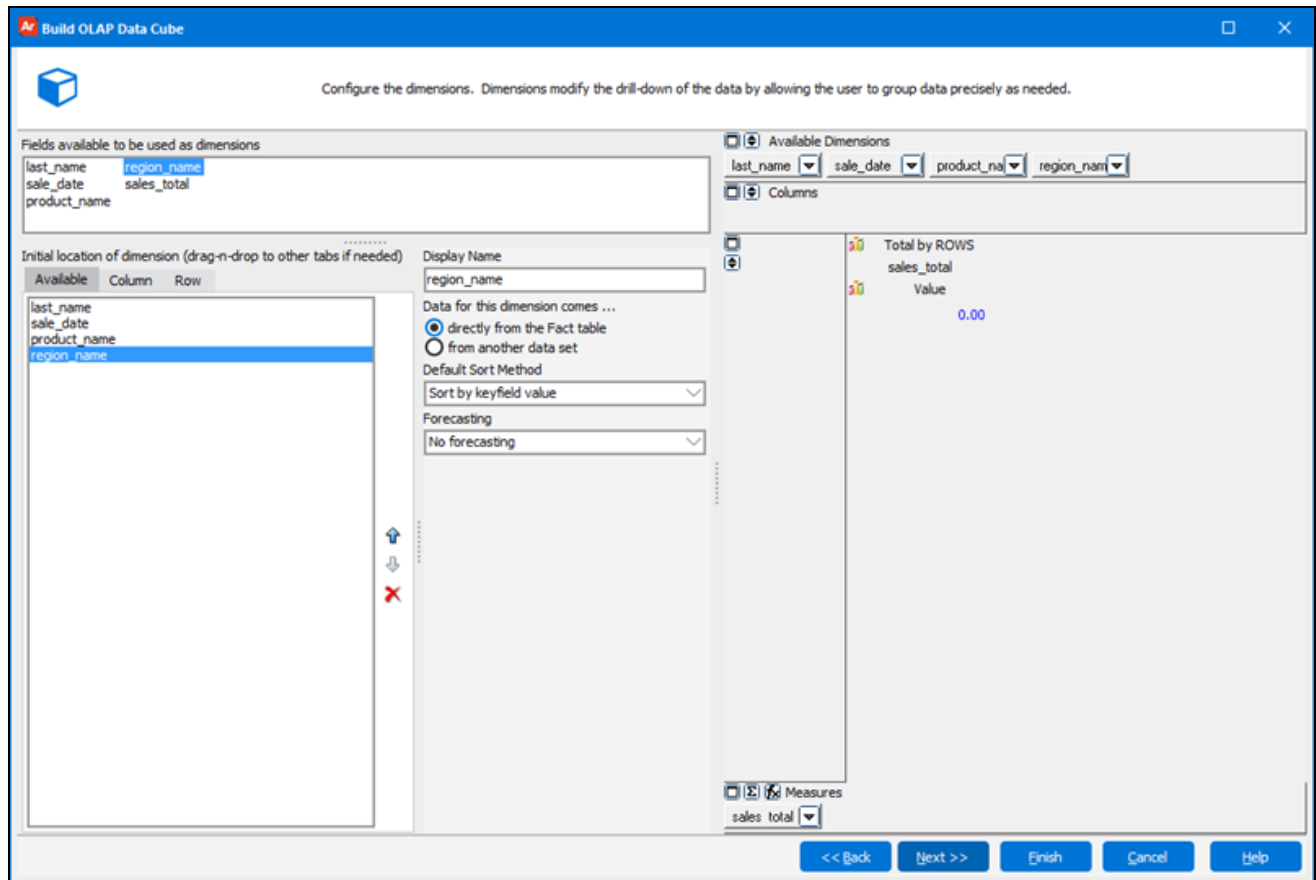


Click the **Next** button, which brings up the next screen where you will [Identify the Dimensions](#).



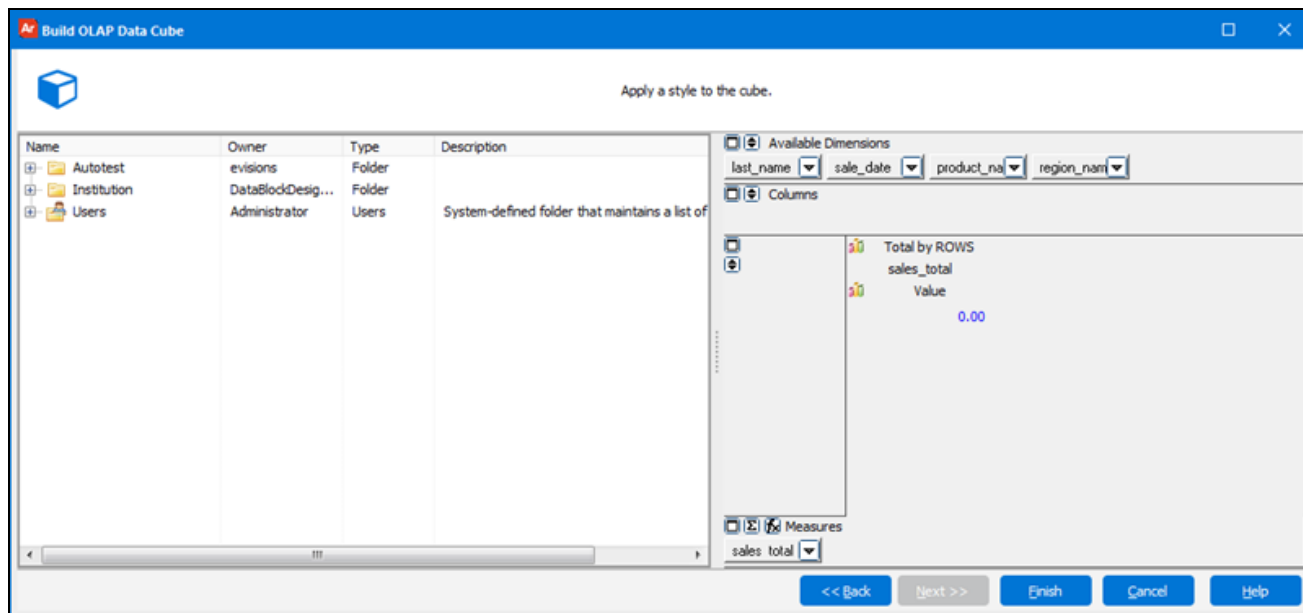
## Identify the Dimensions

1. Double-click last\_name, sale\_date, product\_name, and region\_name.
2. Leave the defaults in place for the remaining options.




3. Click the **Next** button to continue.

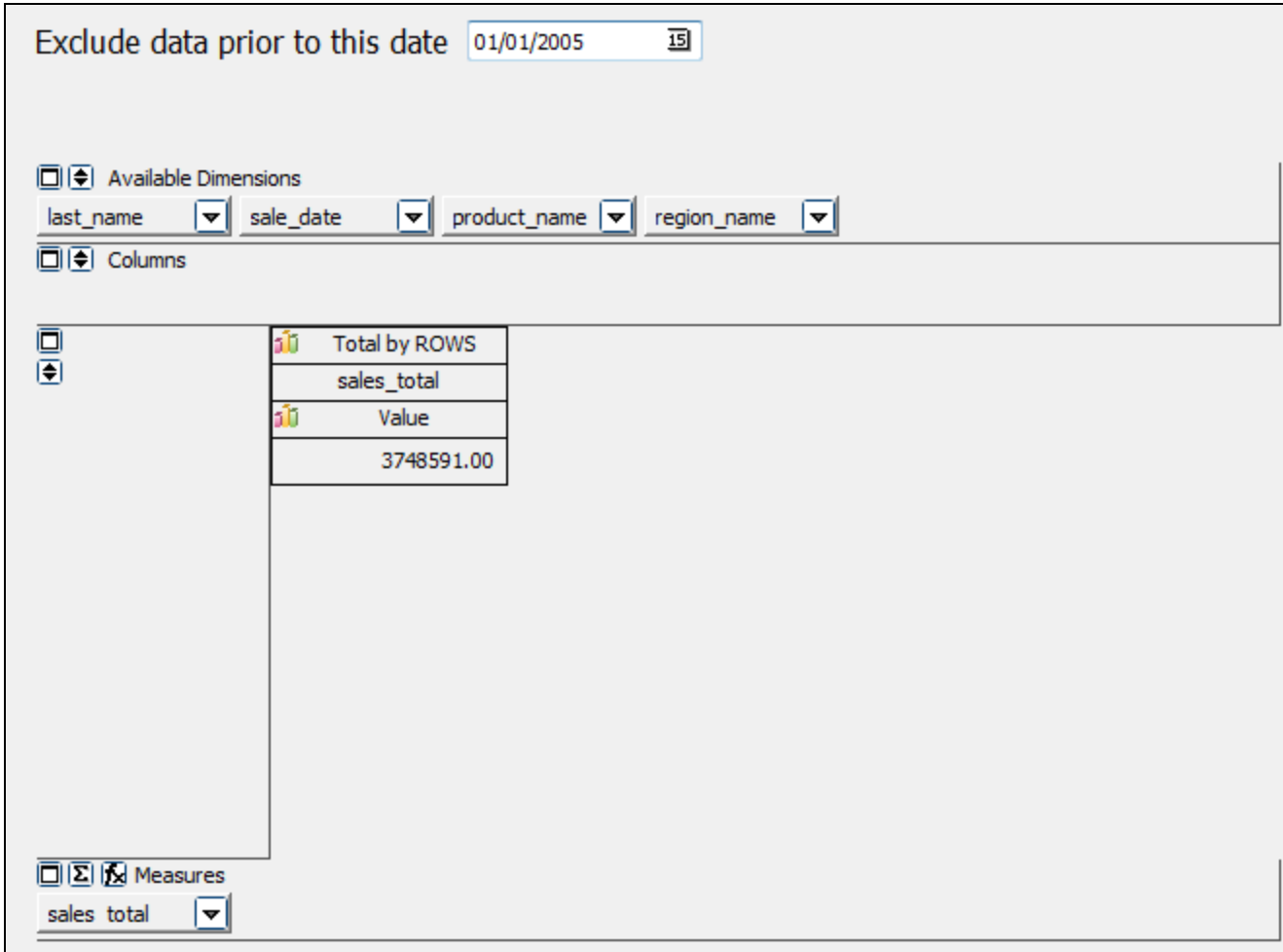
4. On the next screen, you can apply a style to the cube.



5. In this example, we will not use a pre-defined style. Click the **Finish** button to return to the DataBlock designer.

## Commit and Test

Click the **Commit** button to save your work, then click Test  test your work. Enter the desired date into the date object. The results are shown in the figure below.



The screenshot shows a software interface for data analysis. At the top, there is a filter: "Exclude data prior to this date" with a date input field set to "01/01/2005". Below this, there are sections for "Available Dimensions" and "Columns". The "Available Dimensions" section contains dropdown menus for "last\_name", "sale\_date", "product\_name", and "region\_name". The "Columns" section is currently empty. At the bottom left, there is a "Measures" section with a dropdown menu set to "sales total".

Total by ROWS
sales_total
Value
3748591.00

You can now manipulate the measures and dimensions to obtain the view of data that you desire. Experiment by moving the Available Dimensions into rows and columns to obtain different views of the data.

The figure below shows the effect of adding the region\_name as a row, and product\_name as a column.

Exclude data prior to this date 01/01/2005

Available Dimensions  
last\_name sale\_date

Columns  
product\_name

region\_name

product_name	Acer Desktop PC	Asus VW266H 25.5" LCD Monitor	Canon CanoScan 8800F Scanner
region_name	sales_total	sales_total	sales_total
	Value	Value	Value
Midwest	1393.00	1316.00	2587.00
Northeast	4179.00	5593.00	398.00
Northwest	2587.00	2961.00	597.00
South	38009.00	31584.00	0.00
Southeast	1393.00	2961.00	4378.00
Southwest	398.00	58562.00	34427.00
Total by COLUMNS	47959.00	102977.00	42387.00

Measures  
sales total

## Summary

Since the OLAP cube is merely another type of Argos object, the form can be saved as a Dashboard like any other form.

There are many options for manipulating the cube to obtain your desired view of the data. For more information on how to manipulate OLAP cubes, please visit the [Argos Report Viewer Guide](#) and the in-product Help.

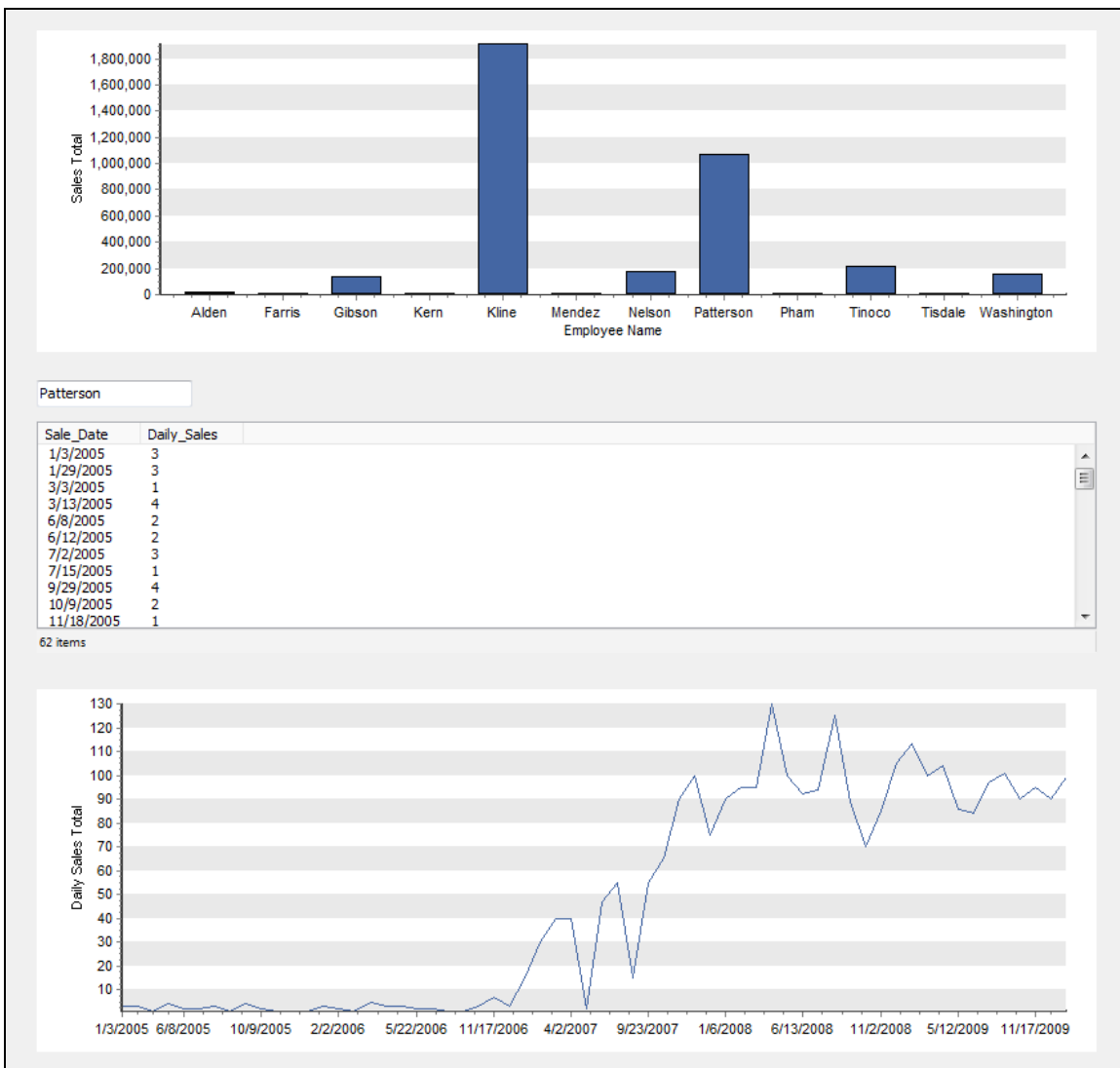
## Drilling through Reports to Obtain Greater Detail

One of the unique features that Argos provides is the ability for the Report Viewer to click on an object in a report and drill down to display a greater level of detail for the object. Consider the Dashboard in the figure below which uses sales information from the Sample Database.

- The bar chart at the top displays a summary of sales for each employee from 2005 – 2009.
- Clicking any of the bars (employees) will drill down and display daily sales details for the selected employee in the multi-column list box and in the line chart at the bottom.

The detail information is not displayed until one of the employees is selected in the bar chart.





Using what you've learned so far, let's walk through creating this dashboard.

1. Create a bar chart at the top of the report area. Name the chart 'Chart1' with a dataset series named Series\_1.
2. In the dataset's SQL query, put the employee's last name and the total sales into the SELECT clause.
3. Turn on Summing, group by the employee's last name, and sum the total sales.
4. In the WHERE clause, limit search results to sale dates between 2005 and 2009.
5. Finish designing the bar chart and return to the DataBlock Designer.
6. Place an edit box (abt) below the bar chart. Make the edit box data aware and set the text to 'Chart1.Series\_1.last\_name', so that it displays the name of the employee selected on the bar chart.
7. Place a multi-column listbox in the middle of the report area. Name the listbox MultiColumn2.
8. In the SELECT clause of the SQL query, enter the sale date (named Sale\_Date) and the quantity (named Daily\_Sales).
9. Turn on Summing, group by the sale date, and sum the quantity.
10. In the WHERE clause, limit search results to the last name of the employee selected in the bar chart.
11. Also in the WHERE clause, limit search results to sale dates between 2005 and 2009.
12. Copy the query, click **OK**, and validate the query to return to the DataBlock Designer.
13. Place a second chart below the multi-column listbox. Name the chart Chart2 with a dataset names Series\_2.

14. Paste the query you copied from the multi-column listbox.
15. Set the chart style to Line.
16. **Commit** your changes and test the dashboard.

**Tip:** Don't forget to join your tables!

## Summary

The procedure to allow drilling down within a report is a straightforward process. The key is to include a common item (last\_name in this example) in the WHERE clause of the query used to obtain the detail information, and link it to the variable representing the selected object.

Save this as a Dashboard.

# Advanced Query Techniques

---

This section describes various advanced query techniques that are available, including unions, summing and grouping of database fields, free-type report queries, scalar subqueries, and correlated and non-correlated subqueries.

- [Unions](#)
- [Summing and Grouping](#)
- [Free Type Report Queries](#)
- [Scalar Subqueries](#)
- [Correlated Subqueries](#)
- [Non-Correlated Subqueries](#)

## Summing and Grouping

---

This example demonstrates how to use the Summing and Grouping capabilities in the Argos DataBlock Designer. This capability allows you to visually create SQL HAVING and GROUP BY statements.

This simple example will create a Dashboard in which sales results of employees will be presented. Results will be grouped by employee; thus the SQL GROUP BY statement will be created. The results will show only sales totals below a specified amount, with an SQL HAVING statement created. Also, the SQL aggregate SUM function will be visually generated to create the sales totals.

The Employees, Orders, Order\_Details, and Products tables within the sample database will be utilized. These are the same tables that were used in Example 1.

**New SQL Statements in this example:** GROUP BY, HAVING, and SQL AGGREGATE functions.

## [Creating the Form](#)

The input required from the user for this report will be a date range, and a sales total threshold that will be used to filter the query.

Create a form as shown below. The variable names for the three input selection fields are StartDate, EndDate, and SalesTotal. The SalesTotal variable will be used to specify a sales total that is used to display names of sales persons whose total are less than this amount.





Start Date // 15 End Date // 15

Sales Total

emp_id	total_sales
--------	-------------

The query for the multi-column list box containing the query results are shown in the figure below. Only two fields will be displayed in the Dashboard, the employee ID and a calculated field (total\_sales) containing the sales total (quantity times unit price).


Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)															
 Distinct  Summing	<table border="1"> <tr> <td>Table</td> <td>Employees</td> <td>&lt;calculated&gt;</td> </tr> <tr> <td>Field</td> <td>emp_id</td> <td>Products.unit_price *Order_De</td> </tr> <tr> <td>Type</td> <td>string</td> <td>string</td> </tr> <tr> <td>As</td> <td></td> <td>total_sales</td> </tr> <tr> <td>Description</td> <td></td> <td></td> </tr> </table>	Table	Employees	<calculated>	Field	emp_id	Products.unit_price *Order_De	Type	string	string	As		total_sales	Description			
Table	Employees	<calculated>															
Field	emp_id	Products.unit_price *Order_De															
Type	string	string															
As		total_sales															
Description																	

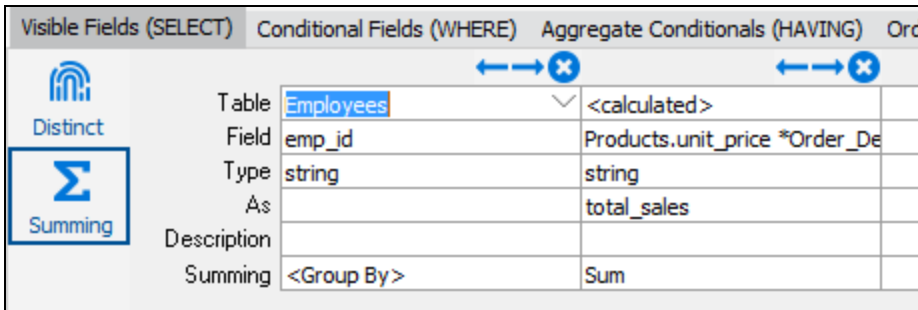
The WHERE clause below is used to find records within the date range specified by the person executing the report.

Conditional Fields (WHERE)	Ordering (ORDER BY)
and/or	and
Table	Orders
Field	sale_date
Condition	>=:StartDate1
	<=:EndDate1

## GROUP BY and SUM fields

To sum the sales records and group them by employee,

1. Click the “Summing” icon  which displays an additional row titled “Summing” as shown below.
2. Under the Employees column, select **<Group By>**.
3. Under the <calculated>, select **Sum** for the calculated field since this data is to be summed. These selections are used to create the GROUP BY statement.

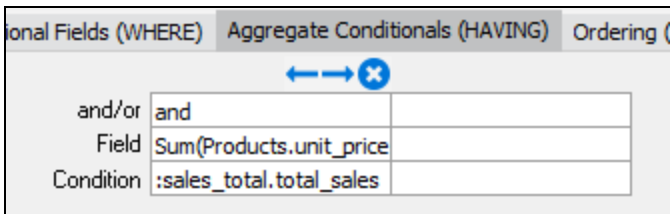


The screenshot shows the 'Summing' icon (a blue square with a white sigma symbol) selected. Below it, a table configuration is shown with the following columns: Table, Field, Type, As, Description, and Summing. The 'Table' column contains 'Employees', the 'Field' column contains 'emp\_id', the 'Type' column contains 'string', the 'As' column contains 'total\_sales', and the 'Description' column contains '<Group By>'. The 'Summing' column contains 'Sum'.

Table	Field	Type	As	Description	Summing
Employees	emp_id	string	total_sales	<Group By>	Sum

## HAVING Tab

The HAVING tab in the figure below specifies a condition where sales total is less than the amount entered. The SalesTotal variable contains the value entered by the user executing the report with that value compared to the calculated field above it. If the SalesTotal is less than or equal to the input selection, that employee will be included in the report. This data is used to create the SQL HAVING statement.



The screenshot shows the 'Aggregate Conditionals (HAVING)' tab. It contains a table with the following columns: and/or, Field, and Condition. The 'and/or' column contains 'and', the 'Field' column contains 'Sum(Products.unit\_price \* Order\_Details.quantity)', and the 'Condition' column contains ':sales\_total.total\_sales <=:SalesTotal'.

and/or	Field	Condition
and	Sum(Products.unit_price * Order_Details.quantity)	:sales_total.total_sales <=:SalesTotal

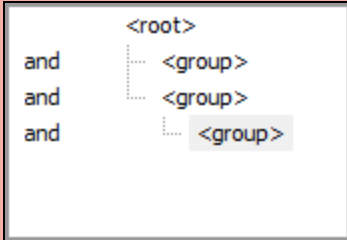
The final SQL:

```
select Employees.emp_id,
       Sum( Products.unit_price * Order_Details.quantity ) as total_sales
from Employees,
     Products,
     Order_Details,
     Orders
where ( Orders.employee_id = Employees.emp_id
       and Order_Details.product_id = Products.product_id
       and Orders.transaction_id = Order_Details.transaction_id )
       and ( Orders.sale_date >= :StartDate
            and Orders.sale_date <= :EndDate )
group by Employees.emp_id
having Sum(Products.unit_price * Order_Details.quantity) <=:SalesTotal
```

Note the existence of the GROUP BY and HAVING statements.

### Creating complex WHERE and HAVING statements:

Note the box at the left portion of the window containing the SELECT, WHERE, HAVING, ORDER BY tabs. By clicking <root> then clicking the plus sign, you can create nested WHERE or HAVING clauses, each enclosed in parentheses and containing its own set of conditionals.



The tree structure above would create SQL structured as follows:

```
WHERE [Join conditions]
  AND [conditions in #1 ]
    AND [conditions in #2]
    AND [conditions in #3]
    AND [conditions in #4]
```

If the Summing fields are no longer desired, make sure to click the Summing icon to remove the row with the Summing fields. Otherwise Argos will expect entries in the Summing fields.

## Results

Executing the Dashboard report produces the following which lists sales employees with sales less than \$1,000,000 between 1/3/2005 and 4/5/2010.

Testing "New DataBlock(4)"

Start Date: 06/13/2008

End Date: 06/13/2016

Sales Total: 1000000


emp_id	total_sales
026	652200
056	12386
101	14157
102	24792
107	10748
139	5412
174	64438
191	24168
238	51424
381	85690
470	12870
788	16176

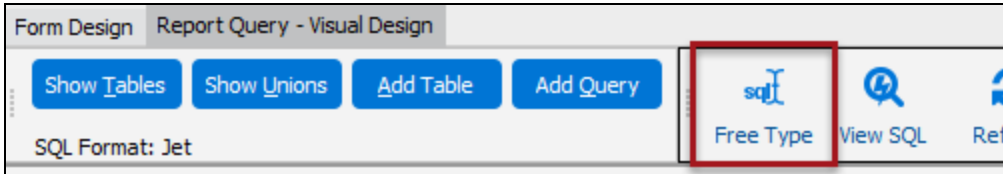
12 items

Retrieve a maximum 1 record(s)

Get Close

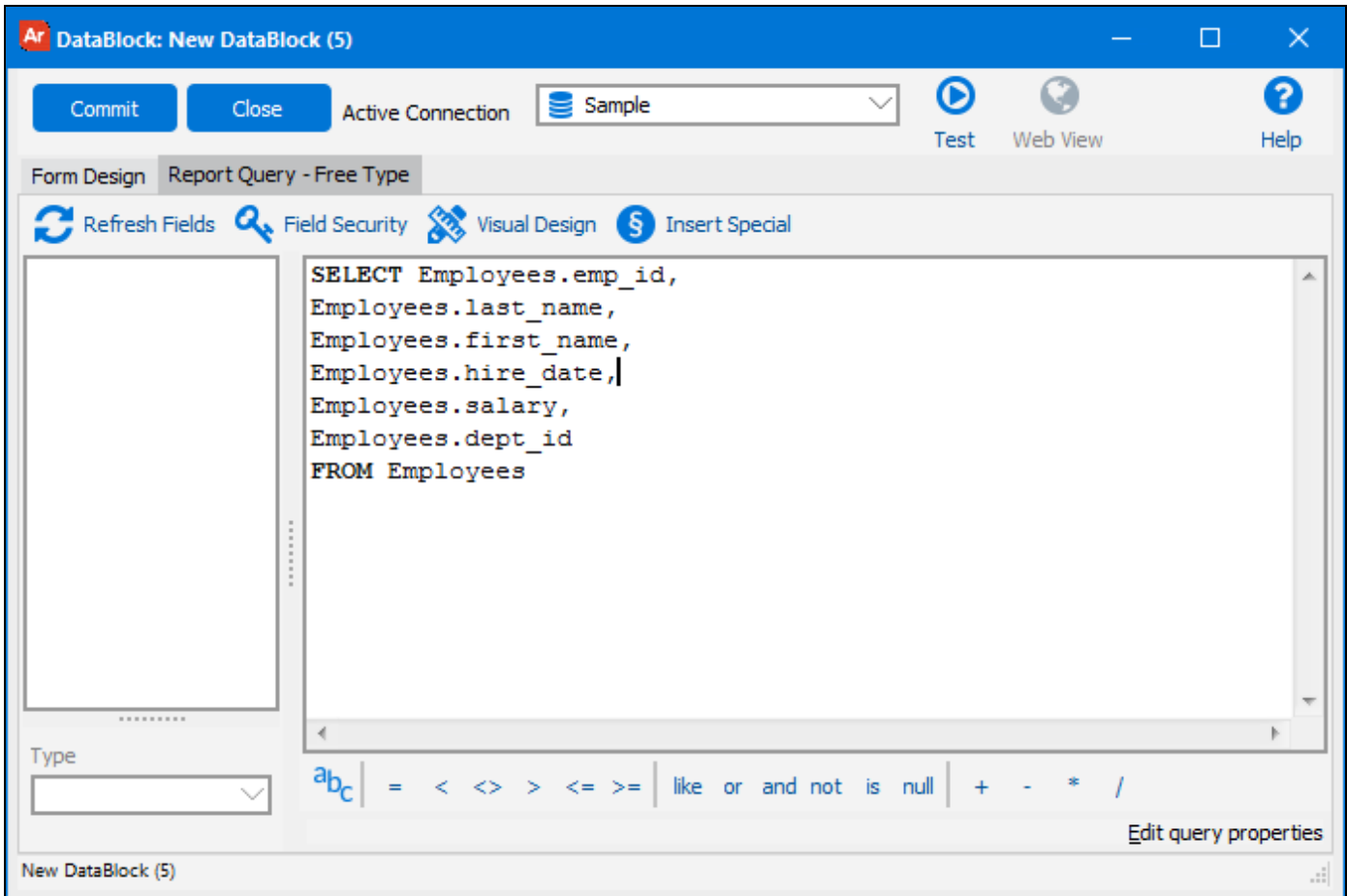
## Free Type Report Queries

Previous examples demonstrated how to use Argos tools to visually create SQL queries. There are times when you already have an SQL query that you would like to enter into Argos without having to type in the query manually. This is possible using the Free Type SQL Editor which exists under the Report Query Tab. In the figure below the free type icon  on the lower right is used to launch the Free Type SQL Editor.



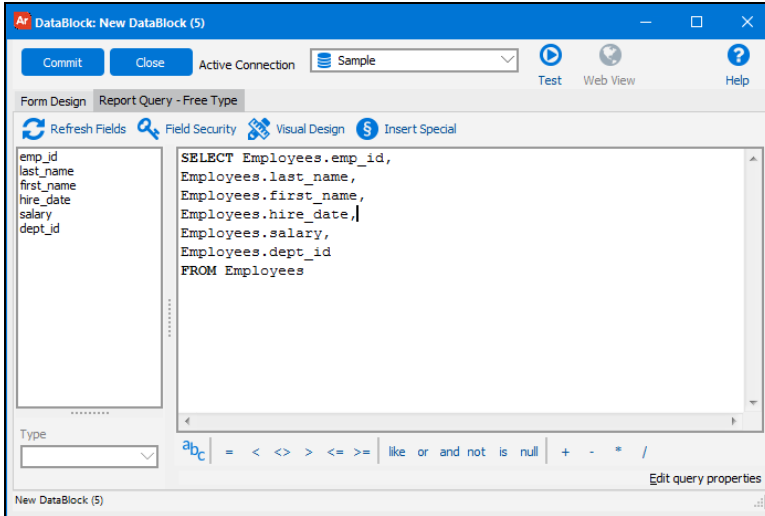
## Entering the Free Type Query

Clicking the Free Type icon launches the Free Type Editor shown below. Either type-in your query or paste it into the right hand window shown below.



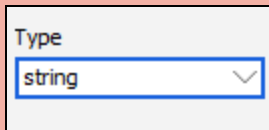
## Validating the Query


Click the Commit button to save the query. After committing, you will be prompted to validate the query by Refreshing the Fields. Click the Refresh Fields icon to create the fields. Argos will populate the window on the left with the fields that it found in the SQL (see figure below). This step is required for Argos to validate that the query matches the database that it is connected to. Click the test button to run the query and verify that it works properly in Argos.




### A few tips:

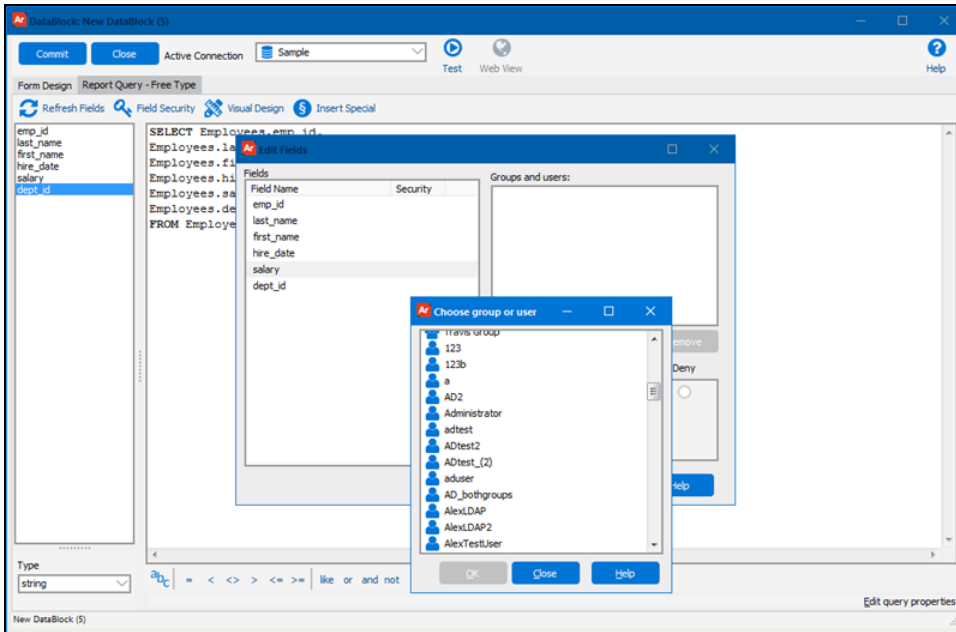
- Below the fields list, in the lower left corner of the free type tab, is the **Type** drop-down. This allows you to highlight a field and change the data type by selecting a new type from the drop-down.



- Clicking the user-defined variable icon  displays a list of System Variables that can be added to your query.
- The operators at the bottom of the window allow you to add logical operations to your query.
- You can use Ctrl-F to find text or Ctrl-H to do a find/replace within the query text.

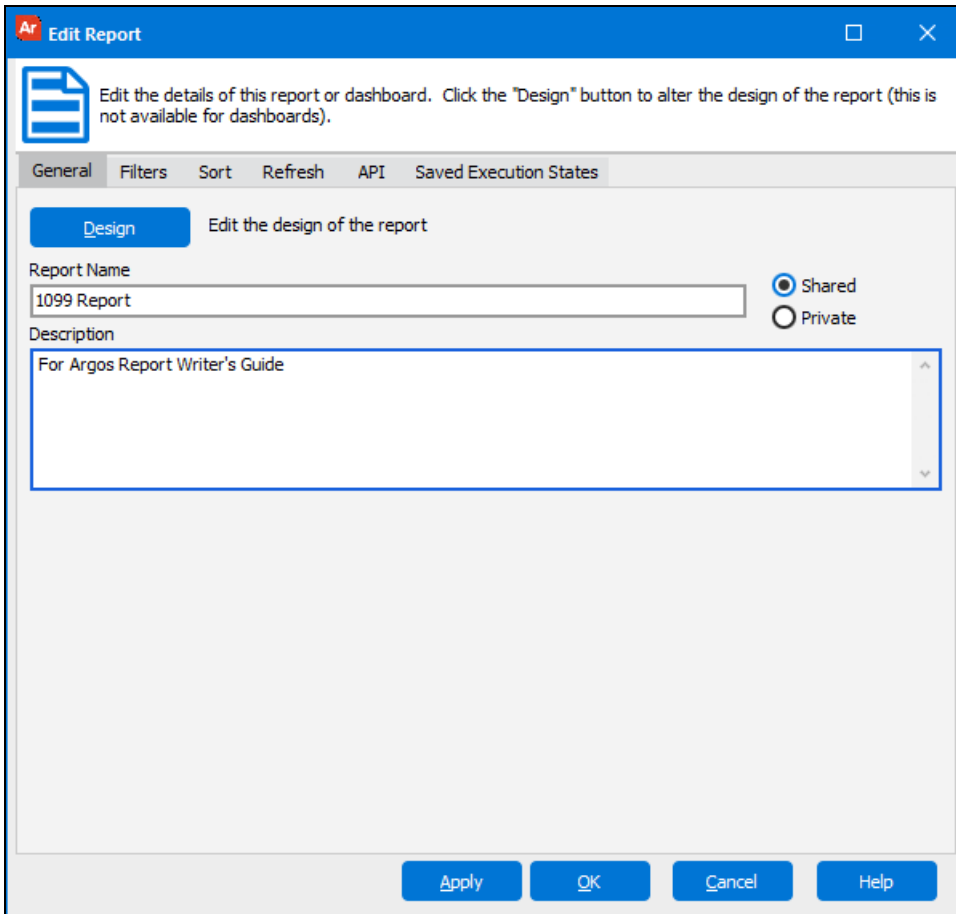
## Apply Field Security

If you wish to apply security to individual fields, this can be accomplished by clicking the Field Security icon . The following dialog box will appear in which you can apply Allow/Deny privileges to Argos users or groups for each selected field. The figure below shows that security is to be applied to the “salary” field, which is highlighted.



## Filter and Sort the Report Query

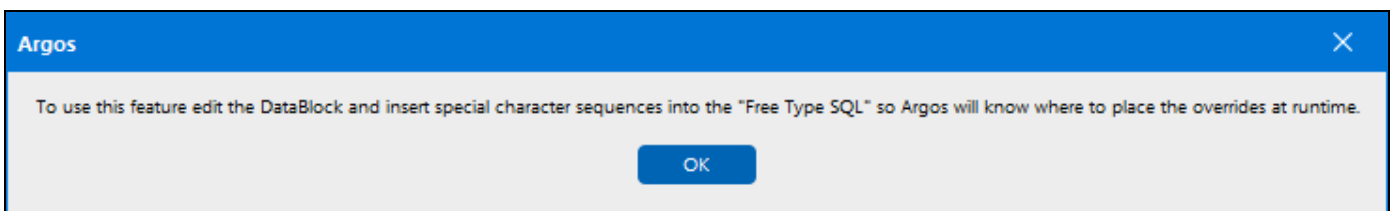
Report Writers can use the Filters and Sort tabs in the Edit Report dialog to modify the query results obtained from the DataBlock when the report is executed.



When creating filters and sorts, Argos adds SQL WHERE and ORDER BY statements to the SQL existing in the DataBlock.

Since Free Type queries are not created by Argos, Argos needs to be guided as to where the WHERE and ORDER BY statements should be added to the SQL query. This is handled by special characters added to the query. The special characters are added by clicking the "Insert Special" icon within the Free Type Editor. The use of special characters is discussed in the next section.


If the Filter or Sort tab is selected during the design of the **report**, and a Free Type query exists in the **DataBlock**, then the message in the figure below will direct you to add the special characters to the Free Type query.



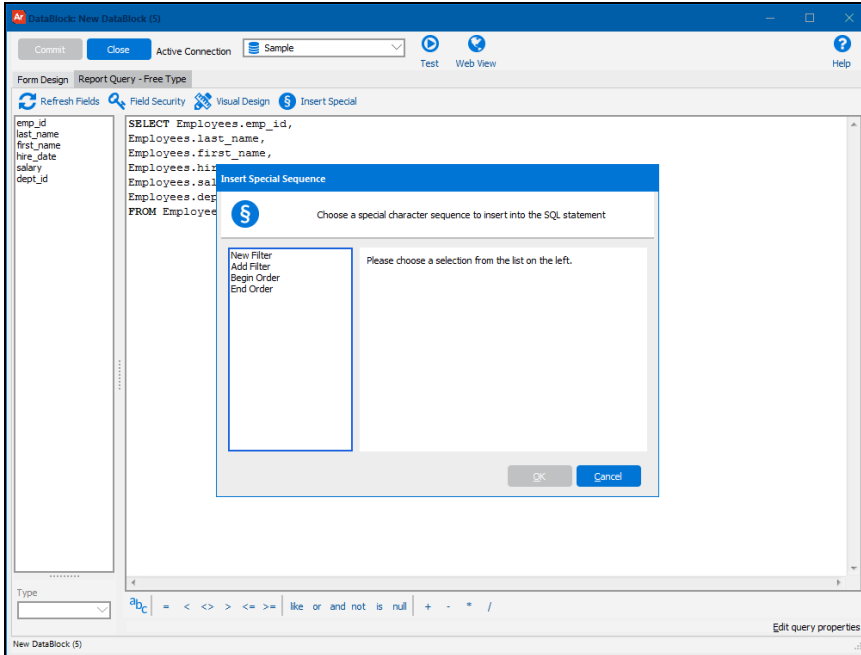


## Adding Special Characters for Filtering Reports

If you feel that your report will require the use of a filter, then return to the Free Type Query Editor and add a special character into the query where the filter (WHERE statement) should be placed.

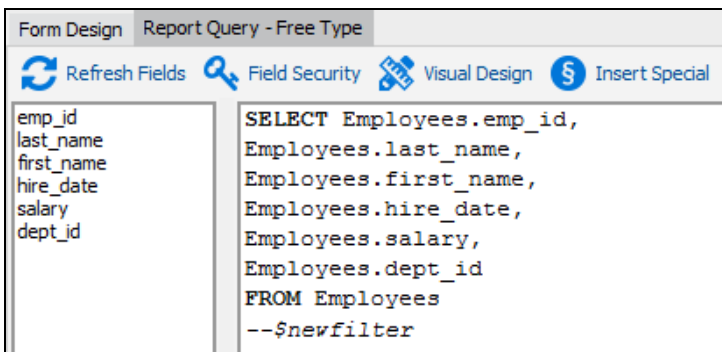
In this example, the Employees table in the sample database is used. A filter will be applied to the salary in the report, therefore the WHERE clause needs to be added to the end of the query. Position the cursor on the first blank line at the end of the query, and click the Insert Special icon . This brings up the “Insert Special Sequence” dialog box shown below.

Select “New Filter”, then click the **OK** button.



The string `--$newfilter` is inserted at the end of the query. This string serves as a placeholder such that when the Filter tab is selected when creating the report, the WHERE clause that the filter creates will replace the `-$newfilter` string. Note the location of the special character string in the figure below. This special character must be placed at the appropriate location within the query since it will be replaced by the WHERE statement created by the report.

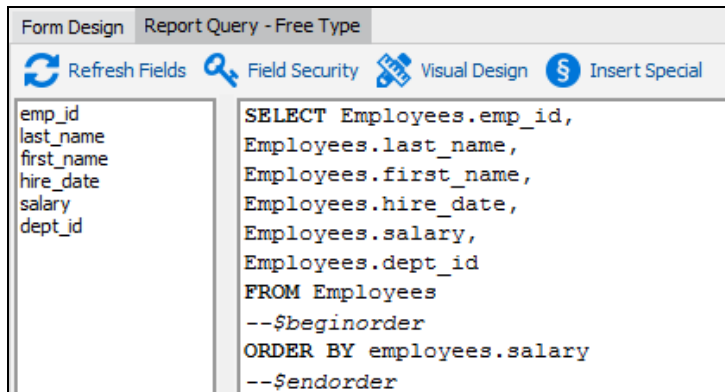
If your Free Type query already has a WHERE statement, (and you are adding a second filter through the report design), then select “Add Filter” instead of “New Filter”. This will prevent Argos from replacing the existing WHERE statement and will add the second WHERE statement.



You can now use the Filter tab when designing your report.

## [Adding Special Characters for Sorting Reports](#)

Instead of using a ORDER BY statement in your free type SQL, you can use sort criteria in the report design, similar to setting up a filter. The query in this example uses an ORDER BY statement to sort employees by salary. To override this sort in the report, you must add the special Begin Order and End Order statements immediately before and immediately after the ORDER BY clause. You can use the **Insert Special** button to add the statements, or type them in manually.



Place the cursor at the desired location, then insert the special character strings and the ORDER BY statement. SQL statements between `--$beginorder` and `--$endorder` will be replaced by the ORDER BY statement created as a result of the sort criteria in the report design.

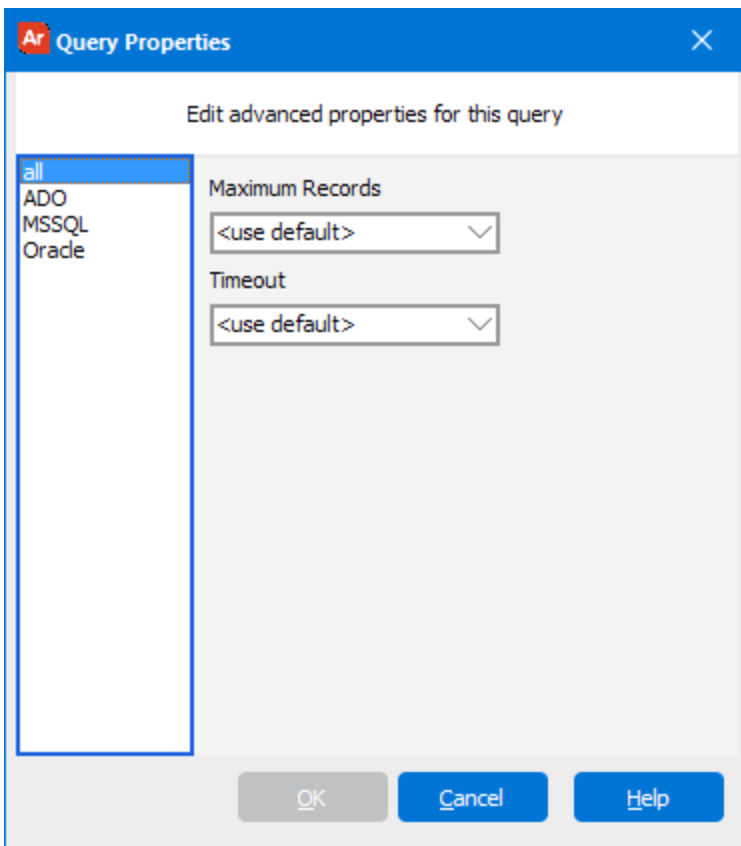
These special characters must be placed at the appropriate location within the query and surround the ORDER BY statement. The statements surrounded will be replaced by the ORDER BY statement created in the report.

## [Edit Query Properties](#)

When your MAPS administrator sets up a data connection in MAPS, they can change various settings that determine how Argos (or any other MAPS application) communicates with the database. Typically, these settings can be left at their default values. However, you can choose to customize them if needed in order to communicate properly with your database. You should consult with your MAPS administrator or DBA for questions on how to connect to a specific database.

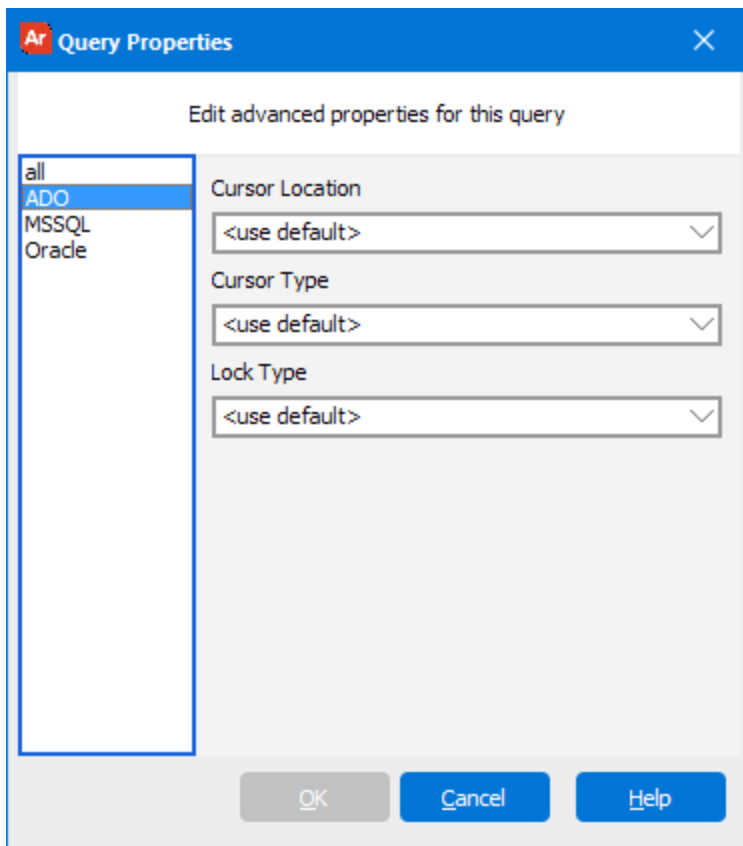
The **Edit Query Properties** dialog in Argos allows you to override the settings configured in MAPS so that you can use different values for a specific query. Changes made here will not affect any other queries or users of this data connection.

Some query properties pertain to all database types:



- **Maximum Records** - This setting is not part of the data connection settings in MAPS, however Argos includes it to provide you with a way to limit the number of records returned by the query. It is similar to the record limit you can specify when testing your query in Argos. It is intended to provide you with a way to limit the amount of time the query takes to run in cases where there may be a significant number of records returned. When set to its default value, there is no limit imposed.
- **Timeout** - The number of seconds Argos will wait for the query to return its first result. The default is 30 seconds.

ADO connections have some additional options:



- **Cursor Location** - Determines where to store the data while the database cursor is open for a query. The default is *Use Server*.
- **Cursor Type** - Specifies how you move through the data and whether changes made to the database are visible in the recordset after you retrieve it. The default is *Open Forward Only*.
- **Lock Type** - Tells the provider what type of locks should be placed on records during editing. Locking can prevent one user from reading data that is being changed by another user, and it can prevent a user from changing data that is about to be changed by another user. The default is *Lock Read Only*.

There are currently no configurable options that are specific to the MSSQL or Oracle database drivers.

### ***Database Drivers***

A connection to a SQL Server or Oracle database may use either an ADO connection or the native MSSQL/Oracle drivers to connect to the database. If you are not sure which driver your connection is using, speak to your MAPS administrator.

It is important to note that your MAPS administrator may have changed the default values when creating the data connection in MAPS. This screen does *not* show the values specified by the MAPS administrator, if they are something other than their default.

### ***MAPS Administrators***

Refer to the [MAPS Help](#) for more detailed information on configuring data connections and their options.

## Summary

Use of special characters within **Free Type** queries is required if the CSV, Banded, or Extract Report Filters or Sorts the query results provided by the DataBlock. The special characters must be placed at the appropriate location in the query such that Argos can replace the special characters with WHERE or ORDER BY statements from the report design.

## Scalar Subqueries

---

Some databases allow entire SQL statements (called a subquery) to be used as a field in the SELECT clause of another query. As each row is processed in the “main” query, the subquery is called to fill in the value for that particular field. These subqueries must return a single value (or no value) for each row in the main query. In the SQL below (using the sample database), a scalar subquery (highlighted) is being used to return the latest PO date for an individual employee (if it exists). In other words, the subquery creates a select statement that returns the latest PO.

```
SELECT Employees.emp_id, last_name, first_name,  
       (SELECT MAX (po_date)from Purchase_Orders WHERE Employees.emp_id = Purchase_Orders.employee_id)  
AS latest  
FROM Employees  
WHERE Employees.emp_id = '501'
```

Note the use of the MAX function which will return one value.

This technique can be used for retrieving a data element from a table if you don't wish to include that table in the main query. This can be a useful way to avoid the use of outer joins, which can negatively impact query performance.

The technique for creating the query/subquery to find the latest Purchase Order created by Employee\_ID = 501 follows.

## Creating the Main Query

Begin by entering emp\_id, last\_name, and first\_name into the SELECT clause. Enter emp\_id into the WHERE clause and set the condition to ='501'.

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)												
<root>	<table border="1"> <tr> <td>and/or</td> <td>and</td> <td></td> </tr> <tr> <td>Table</td> <td>Employees</td> <td></td> </tr> <tr> <td>Field</td> <td>emp_id</td> <td></td> </tr> <tr> <td>Condition</td> <td>= '501'</td> <td></td> </tr> </table>	and/or	and		Table	Employees		Field	emp_id		Condition	= '501'		
and/or	and													
Table	Employees													
Field	emp_id													
Condition	= '501'													

## Creating the Subquery

On the SELECT tab, create a calculated field by creating a new column, click within the "Field" row, then click the ellipsis to bring up the SQL Editor. Enter the subquery as shown below. Remember to enclose the subquery with parentheses.

SQL Editor

```
(select MAX(po_date)
from Purchase_Orders
where Employees.emp_id = Purchase_Orders.employee_id)
```

abc (---) = < > <= >= like or and not is null + - \* /

OK Close Help

Give the calculated field an alias as shown below.

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)	Ordering (ORDER BY)	Ordering (ORDER BY)	Ordering (ORDER BY)
Distinct	Table: Employees	Table: Employees	Table: Employees	Table: <calculated>	
Summing	Field: emp_id	Field: last_name	Field: first_name	Field: (select MAX (po_date)from Pur	
	Type: string	Type: string	Type: string	Type: string	
	As:			As: latest	
	Description:				

## Results

**Commit** your changes and execute the query. Executing the query displays the latest PO date for employee 501. There are 19 purchase orders created by this employee, with 11/19/2008 being the latest.

emp_id	last_name	first_name	latest
501	Orange	Herbert	11/19/2008

## Correlated Subqueries

Some queries require that the output be limited by the results of a subquery. For example, a table may contain a field that includes an effective date and you may want your query to limit the results to the record that contains the latest effective date.

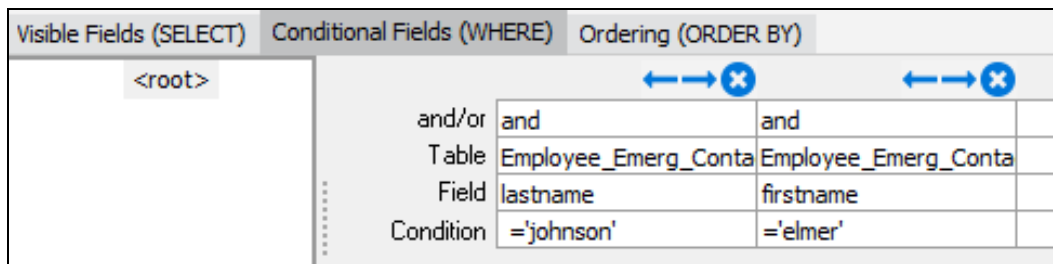
In this example, the "Employee\_Emerg\_Contact" table from the sample database will be used. This table contains emergency contact information for each employee and is updated whenever contact information changes. The table contains items such as the employee ID number, first and last name, phone number, and effective date. The "effective\_date" field contains the date that the record was updated.

The example will find the latest contact information for "Elmer Johnson", for which multiple records exist containing various effective dates in the database. Elmer Johnson is an emergency contact of employee 018 (Priscilla Johnson) in the Employees Table.

### Creating the Main Query

Create the main query using the Employee\_Emerg\_Contact table. Add lastname, firstname, phone, and effective\_date to the SELECT clause. For the WHERE clause, filter by lastname = 'johnson' and firstname = 'elmer'.

Note that this query can return multiple records for that individual.



Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
<root>	←→ × ←→ ×	
	and/or	and
	Table	Employee_Emerg_Conta Employee_Emerg_Conta
	Field	lastname firstname
	Condition	= 'johnson' = 'elmer'

This query shown below will retrieve three records for Elmer Johnson with effective dates of 1/1/2000, 1/1/2001, and 1/1/2002, each with a different phone number.

```
SELECT Employee_Emerg_Contact.lastname,  
Employee_Emerg_Contact.firstname,  
Employee_Emerg_Contact.phone,  
Employee_Emerg_Contact.effective_date  
FROM Employee_Emerg_Contact  
WHERE Employee_Emerg_Contact.lastname = 'johnson'  
AND Employee_Emerg_Contact.firstname = 'elmer'
```

A subquery is needed (shown in the figure below) to return the record with the latest effective date.

```
SELECT Max( Employee_Emerg_Contact1.effective_date ) AS Max_effective_date  
FROM Employee_Emerg_Contact Employee_Emerg_Contact1  
WHERE Employee_Emerg_Contact1.emp_id = Employee_Emerg_Contact.emp_id
```

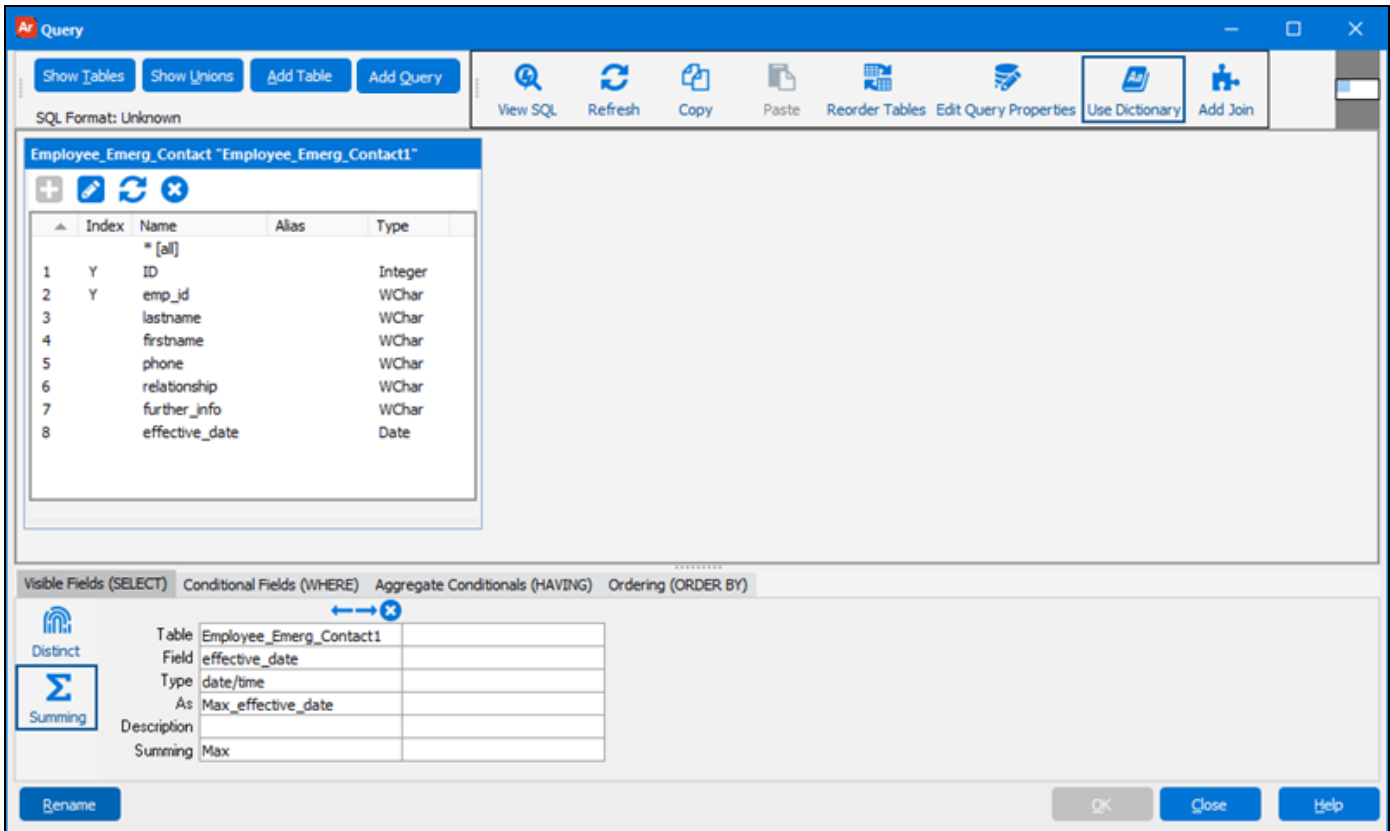
Notice that the subquery is "linked" to the main query in the very last line. As each row from the main query is examined, it will be compared against the results of the subquery. These types of subqueries are called "correlated", as the results of the main query are dependent on the results of the subquery. Also notice that in this example, the main query and the subquery are using the same table. Whenever a table is used multiple times in the same query, it is important to add a table alias for each instance of the table (Argos adds the table aliases automatically). Although both queries use the Employee\_Emerg\_Contact table, in one case the alias is Employee\_Emerg\_Contact and in the other it is Employee\_Emerg\_Contact1. The database considers these as separate tables and does not get confused as long each field is prefaced with the table alias.

Recall that the subquery is to provide a single value for Employee\_Emerg\_contact.effective\_date in the WHERE clause. That is, a multiple lines of SQL are used to provide a value to be used in the WHERE clause.

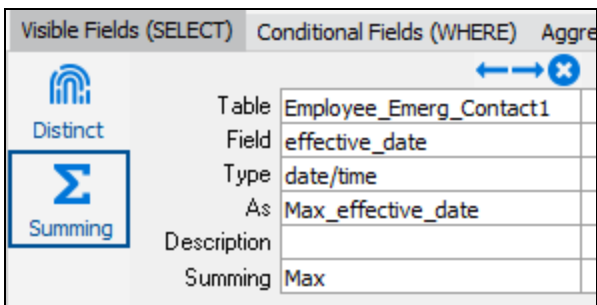
## Creating the Subquery

The steps for creating the subquery shown above will now be discussed. Within the Build Query dialog box, click the **Add Query** button. The Query dialog box will be displayed again such that you can use it to create the subquery. Click the Show Tables button and select the Employee\_Emerg\_Contact table. Double-click the effective\_date field which moves the field under the SELECT tab as shown below.

Notice that in the subquery, the table name is called Employee\_Emerg\_Contact1 because the main query and subquery are using the same table (as discussed above).



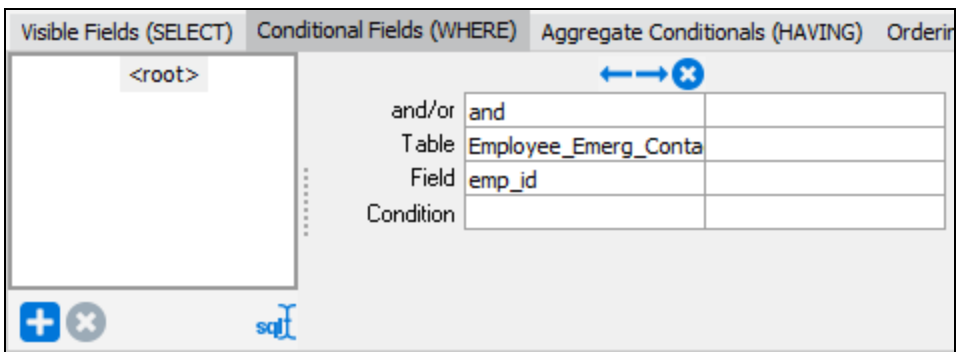
Click the "Summing" icon and select "Max" since the subquery is to find the latest effective date.



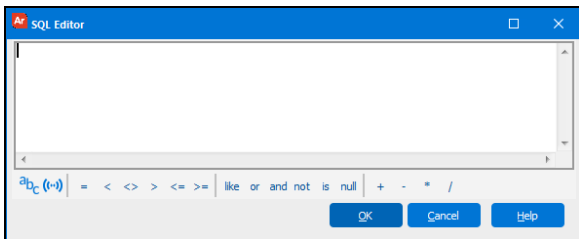
Next create a WHERE statement to link the main query to the subquery using the employee ID as the common field. When using the editor to create the WHERE statement, since the emp\_id field exists in both Employee\_Emerg\_Contact and Employee\_Emerg\_Contact1 tables, thus both must be referenced.


Select the Employee\_Emerg\_Contact1 table as shown below.

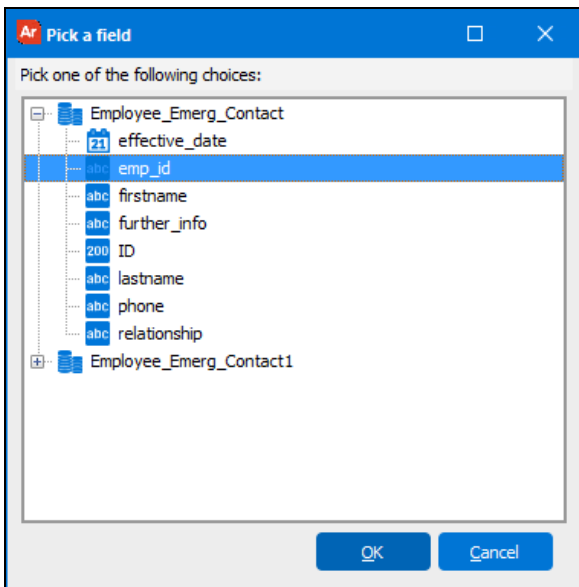




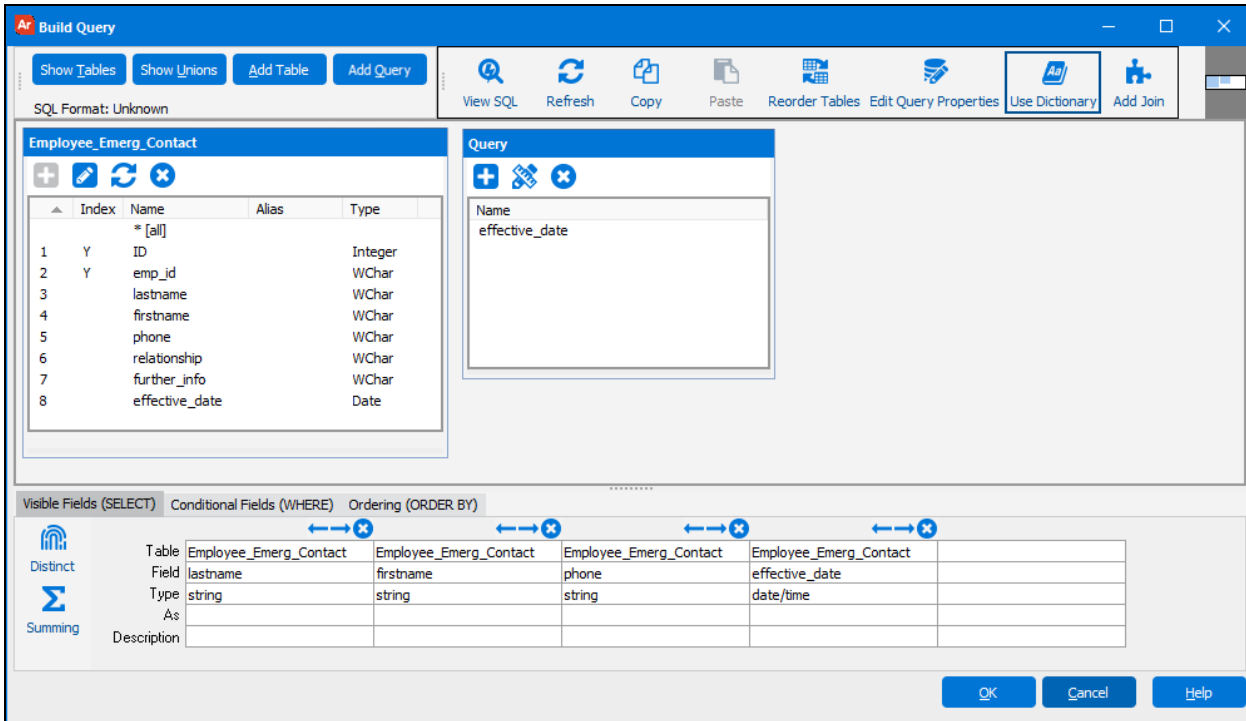
Click the ellipses button  inside the Condition field to launch the SQL Editor.



Click the insert field icon  to display a list of database fields. Select Employee\_Emerg\_contact.emp\_id. This now links the emp\_id between the main query and the subquery.



The subquery now appears within the Build Query dialog box as shown below.

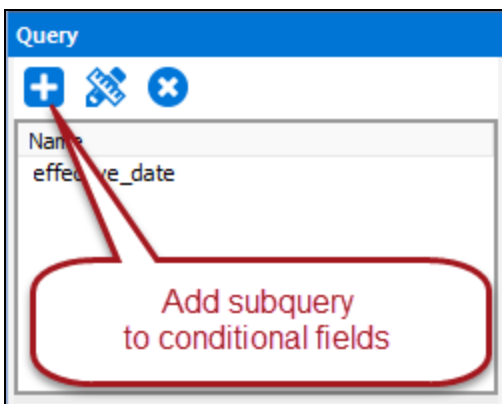


The subquery is shown below.

```
select Max( Employee_Emerg_Contact1.effective_date ) as Max_effective_date
from Employee_Emerg_Contact Employee_Emerg_Contact1
where Employee_Emerg_Contact1.emp_id = Employee_Emerg_Contact.emp_id
```

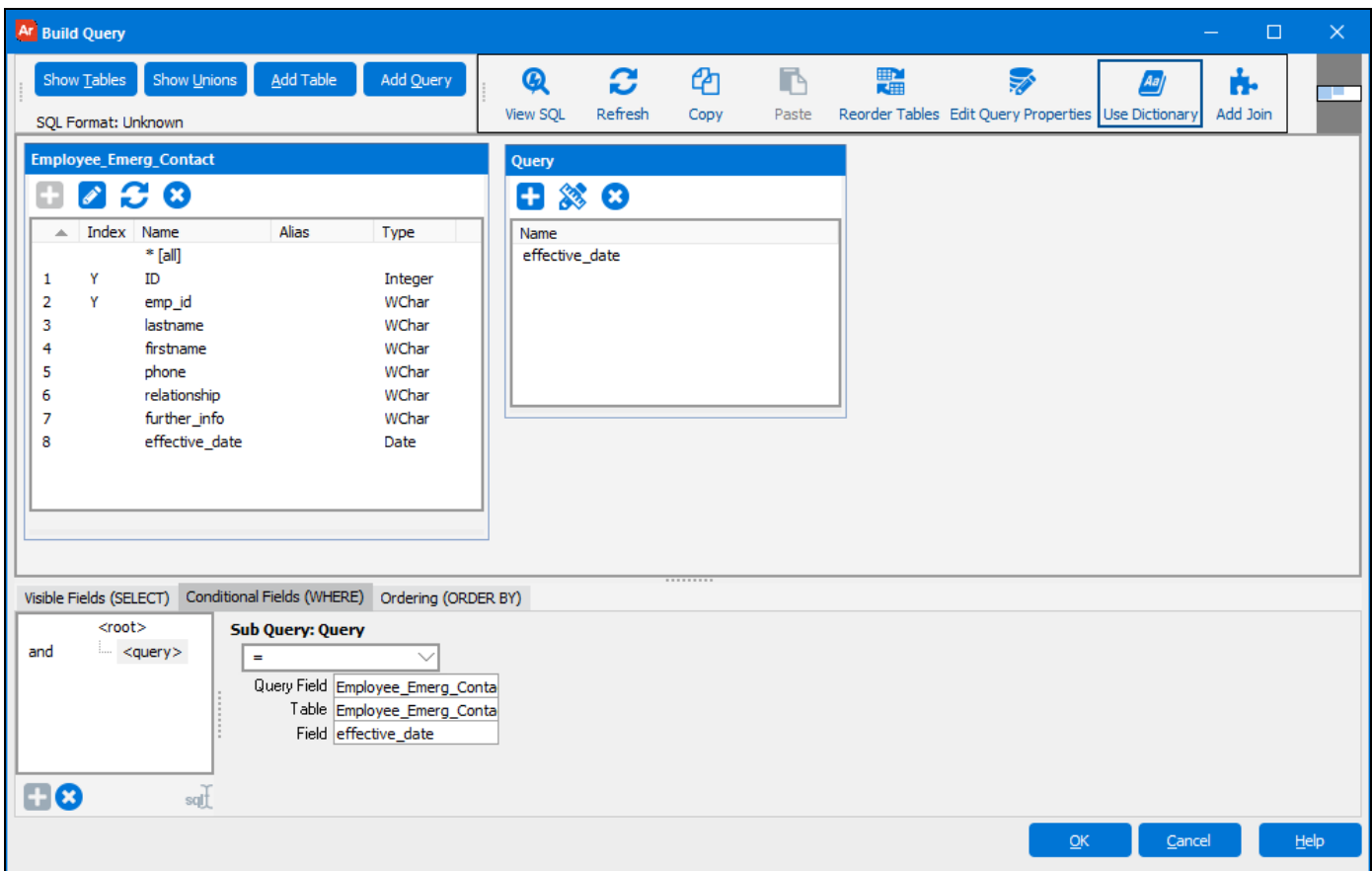
### Placing the Subquery in the Main Query

Now that the main query and subquery have been built, the subquery needs to be placed into the appropriate section of the main query. To perform this linkage, click the "Add this Subquery to the Conditional Fields Tree" button within the Subquery window.



Recall that for this individual, three records with different effective dates exist in the database. The query/subquery obtained the latest.

The Conditional Fields (WHERE) tab is displayed at the bottom of the window where you can use the appropriate aggregate function to link the subquery to the main query. In this case choose the = operator since we want the record that has an effective date equal to the effective date returned by the subquery. Make sure to choose the effective\_date field from the Employee\_Emerg\_Contact table in the Main Query.



The process is now complete which creates the complete SQL shown below.

```

SELECT Employee_Emerg_Contact.lastname,

       Employee_Emerg_Contact.firstname,

       Employee_Emerg_Contact.phone,

       Employee_Emerg_Contact.effective_date

FROM Employee_Emerg_Contact
  where Employee_Emerg_Contact.lastname = 'johnson'

AND Employee_Emerg_Contact.firstname = 'elmer'
      AND Employee_Emerg_Contact.effective_date =
      ( select Max( Employee_Emerg_Contact1.effective_date )
        AS Max_effective_date

FROM Employee_Emerg_Contact Employee_Emerg_Contact1
  WHERE Employee_Emerg_Contact1.emp_id = Employee_Emerg_Contact.emp_id )

```

## Results

Executing the query yields the results shown below in which the latest effective date for Elmer Johnson is provided.

lastname	firstname	phone	effective_date
Johnson	Elmer	(221) 555-6980	1/1/2002

## Non-Correlated Subqueries

At times it is useful to create a subquery that is not directly related to the main query. For example, you may want to create a report that includes open vendor invoices but only for those vendors in a certain state within the U.S.

It is possible to accomplish this with some tricky join logic, but a “non-correlated” subquery is often easier. These types of subqueries are generally not tied to a particular field in the main query using the = operator, rather they typically use other operators such as IN, NOT IN, EXISTS, and NOT EXISTS. These are called non-correlated subqueries as the subquery is not really dependent on the main query data. The results are the same regardless of which row is being examined in the main query. This can also result in faster performance for many queries.



In this example using the sample database, the non-correlated subquery will return a list of vendors (vendor ID's) in the state of Arizona, and the main query will return invoices that have been received (invoice\_recd = 'y' in the Purchase Order Table). The Vendor\_ID field is the common field in both tables. Vendors with vendor\_ID of “ven002” and “ven007” are located in Arizona.

## Creating the Main Query

The process is essentially the same as used for the Correlated subquery discussed in the [previous example](#).

Create the main query to return all Purchase Orders with invoice\_reqd field = “Y”.

The SELECT clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)	
 Distinct			
 Summing			
Table	Purchase_Orders	Purchase_Orders	Purchase_Orders
Field	vendor_ID	PO_ID	invoice_recd
Type	string	string	string
As			
Description			

And the WHERE clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
<root>		
and/or	and	
Table	Purchase_Orders	
Field	invoice_recd	
Condition	= 'y'	

## Creating the Subquery

Click the **Add Query** button to add the subquery. The subquery finds all vendors located in Arizona. Note that this query is not tied to a particular set of records in the main query like the previous example was. Add 'vendor\_ID' from the Vendors table

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
<root>	and/or and	← → ×
	Table Vendors	
	Field vendor_state	
	Condition ='az'	

## Placing the Subquery in the Main Query

This query is non-correlated, and is likely to return multiple records, so for this example the "IN" operator is used to link the subquery to the main query. Select Purchase\_Orders for the Table, and select vendor\_ID for the Field.

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
and <query>	<b>Sub Query: Query</b> IN	
	Query Field Vendors.vendor_ID	
	Table Purchase_Orders	
	Field vendor_ID	

+ × sql

This results in the following query:

```
SELECT Purchase_Orders.vendor_ID,  
Purchase_Orders.PO_ID,  
Purchase_Orders.invoice_recd  
FROM Purchase_Orders  
WHERE Purchase_Orders.invoice_recd ='y' AND Purchase_Orders.vendor_ID  
IN ( select Vendors.vendor_ID FROM Vendors  
WHERE Vendors.vendor_state ='az' )
```

## Results

Executing the query yields the results shown below where all vendors in Arizona whose invoices have been received are listed.

vendor_ID	PO_ID	invoice_recd
ven002	1002	y
ven002	1007	y
ven007	1010	y
ven002	1015	y
ven002	1016	y
ven002	1017	y
ven007	1021	y
ven002	1026	y
ven002	1047	y
ven002	1060	y
ven007	1063	y
ven002	1068	y

## Inline View Subquery



At times it is useful to create a subquery that is used within the FROM clause as if it were a table name. This is known as an inline view or inline view subquery. The SQL statement in the inline view defines the source of the data for the FROM clause.

## Creating the Main Query

The process is essentially the same as used for the Correlated subquery discussed in a [previous example](#).

Create the main query to return all Purchase Orders with invoice\_recd field = "Y".

The SELECT clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)	
			
Table	Purchase_Orders	Purchase_Orders	Purchase_Orders
Field	vendor_ID	PO_ID	invoice_recd
Type	string	string	string
As			
Description			
			

And the WHERE clause:

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
<root>		
	and/or	
	Table	Purchase_Orders
	Field	invoice_recd
	Condition	= 'y'

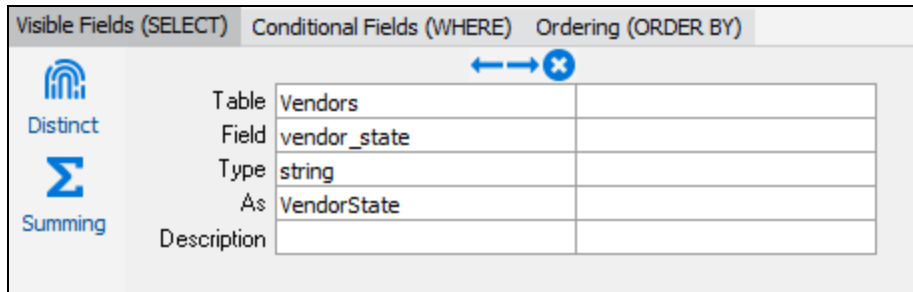
## Creating the Subquery



Click the **Add Query** button to add the inline view subquery.

The build subquery editor displays, and allows you to select a table to work with. Use the fields in the table and the tabs at the bottom of the page to build your subquery. For example, you might select a field from a table, and add a condition (the field equals a specific value, for example).

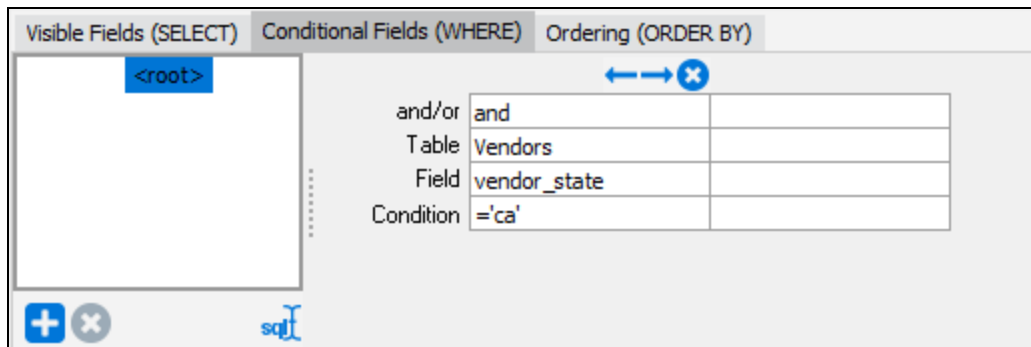
If you want to limit the query to vendors located in California, you would create the inline view based on the vendor\_state field of the Vendors table:

The SELECT clause:



Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
 Distinct		
 Summing		
Table	Vendors	
Field	vendor_state	
Type	string	
As	VendorState	
Description		

And the WHERE clause:



Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)
<root>		
	and/or	and
	Table	Vendors
	Field	vendor_state
	Condition	='ca'

Use the **Rename** button at the bottom of the editor page to give your subquery a user-friendly name, such as "CA\_Vendors," so that you can recognize it later.

Select OK to return to editing the main query.

## Placing the Subquery in the Main Query

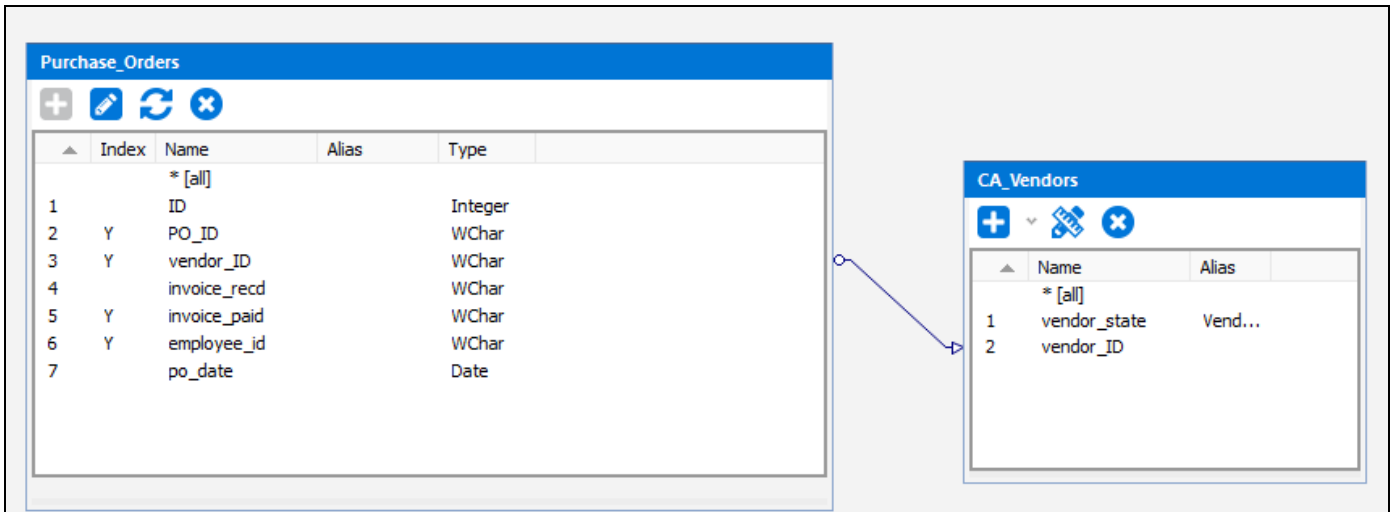
The new subquery displays in the main query editor in a window similar to a table.

Make sure that the Visible Fields (SELECT) tab is active at the bottom of the editor, then double-click a field from your new subquery to add it to the main query as the target for the FROM clause. In this example, add the vendor\_state field.

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)		
Table	Purchase_Orders	Purchase_Orders	Purchase_Orders	CA_Vendors
Field	vendor_ID	PO_ID	invoice_recd	"VendorState"
Type	string	string	string	string
As				
Description				

Note that if you are using an Oracle database, you must enclose the field name in quotes. The dropdown for the Field cell provides both options (with and without quotes).

In most cases, you would then join the vendor\_ID field from the Purchase\_Orders table to the new CA\_Vendors table:



This results in the following query:

```
SELECT Purchase_Orders.vendor_ID,
       Purchase_Orders.PO_ID,
       Purchase_Orders.invoice_recd,
       CA_Vendors.VendorState
FROM Purchase_Orders left join ( select Vendors.vendor_state as VendorState,
                                       Vendors.vendor_ID
                               FROM Vendors
                               WHERE Vendors.vendor_state ='ca' ) CA_Vendors on Purchase_Orders.vendor_ID = CA_
Vendors.vendor_ID
WHERE Purchase_Orders.invoice_recd ='y'
```



## Results

Executing the query yields the results shown below, listing the vendors in California whose invoices have been received.

vendor_ID	PO_ID	invoice_recd	VendorState
ven011	1100	y	CA
ven011	1101	y	CA
ven012	1102	y	CA
ven012	1103	y	CA
ven013	1104	y	CA

Retrieve a maximum  record(s) Get Close





5 rows

## Unions

The UNION statement allows you to combine multiple queries together to create a single result set. All queries in the union SELECT statements must have an equal number of expressions. In addition, these expressions (column names, literals, dates, results from functions, etc.) must be of compatible data types.

This example (using the sample database) will create a union using 3 queries to obtain information from the Employees table. The first query will obtain employees in the Sales Department. The second query will obtain employees in the Marketing Department, and the third query will obtain employees in the Training Department. The Union will combine the results of the three queries.



These are the union types available in Argos:

Icon	Description
	<b>A standard union combines the results of the two queries into one result set excluding any duplicate records.</b>
	<b>A UNION ALL union combines the results of the two queries into one result set and does not remove any duplicate records.</b>
	<b>An INTERSECT union returns only those records that exist in both queries.</b>
	<b>A MINUS union returns all records from the first query except for those records which also exist in the second query.</b>

## Creating the Main Query

Launch the Argos DataBlock Designer, add a multi-column list box to the Design Area, double-click on the object to launch the Build Query dialog box. Create the query shown below to select last name, first name, and department name for employees in the Sales Department. This query will be used as the basis for the creation of the other two queries.

The SELECT clause:

Visible Fields (SELECT)		Conditional Fields (WHERE)		Ordering (ORDER BY)	
 Distinct   Summing	Table	Employees	Employees	Departments	
	Field	last_name	first_name	dept_name	
	Type	string	string	string	
	As				
	Description				

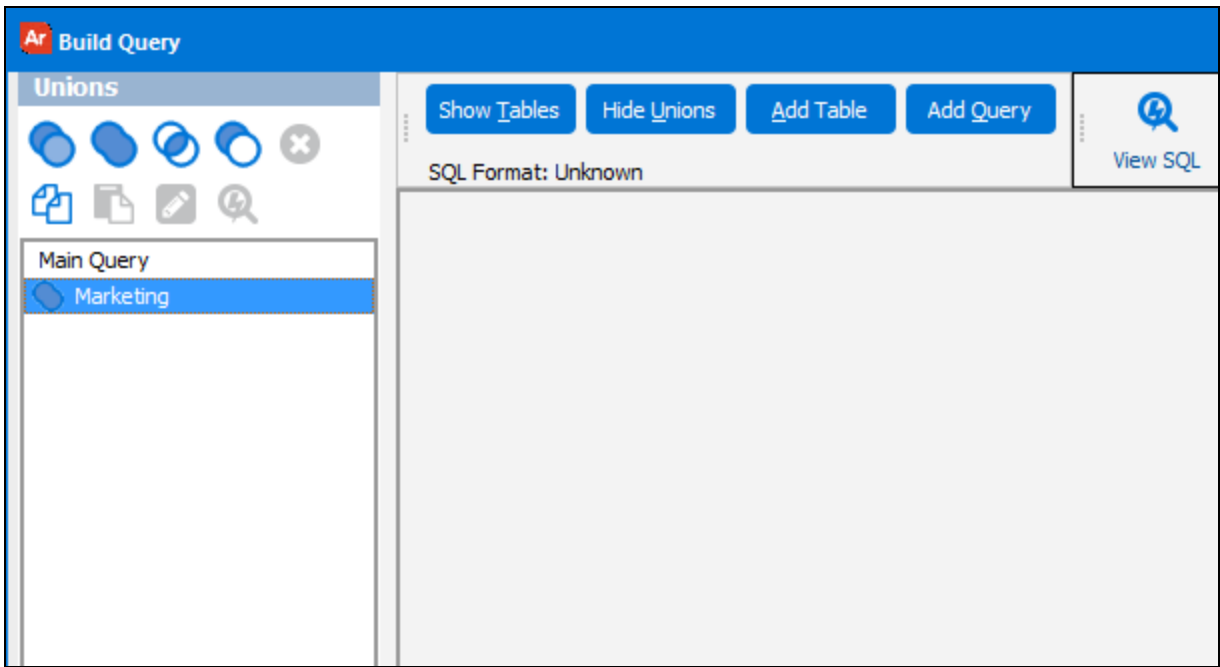
The WHERE clause:

Visible Fields (SELECT)		Conditional Fields (WHERE)		Ordering (ORDER BY)	
<root>		and/or	and		
		Table	Departments		
		Field	dept_name		
		Condition	= 'Sales'		

After creating the query, you can click OK to test the query, or merely move on to the next step to create the Unions.

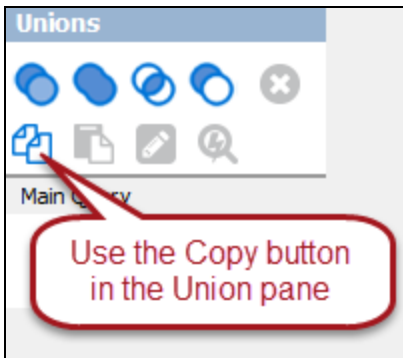
## Creating the Unions


Click the “Show Unions” button at the top of the Build Query dialog box, and the Unions pane will appear as shown in the figure below.

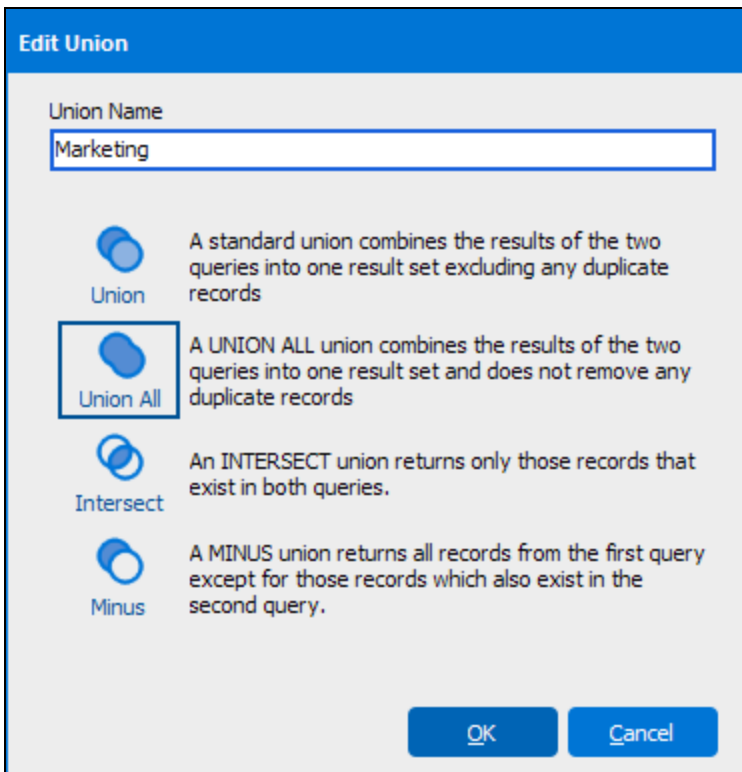


The screenshot shows the 'Build Query' dialog box. On the left, there is a 'Unions' pane with a list of queries. The 'Main Query' section is expanded, showing a query named 'Marketing'. At the top of the dialog, there are buttons for 'Show Tables', 'Hide Unions', 'Add Table', and 'Add Query'. The 'SQL Format' is currently set to 'Unknown'. A 'View SQL' button is also present on the right side of the dialog.


The query created in the previous step is shown as "Main Query". Since the Main Query will be used as the basis for the other queries, click the Copy icon (**Under the Unions pane**) to place the query into the clipboard. **Do not click the Copy on the top of the Build Query dialog box; you must use the Copy icon within the Unions pane.**



Click the **Union All** icon  which creates "Union All" under the Main Query. Right-click on "Union all" to edit the Union. Change the Union Name to "Marketing". Note that this example uses Union All, however you should use the type of Union that applies to your needs.

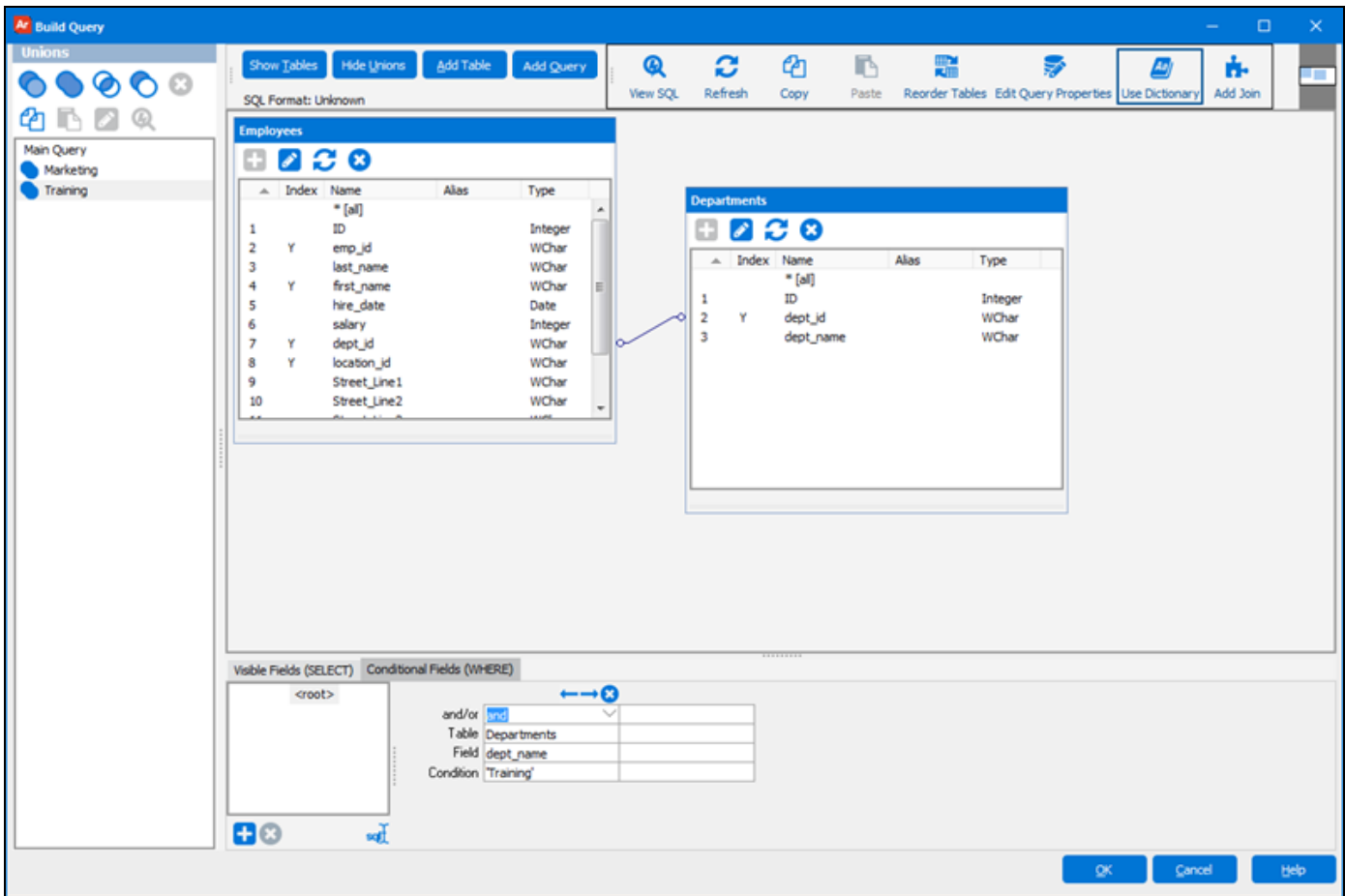


Click Paste (under Unions Pane) to bring up the previous query, which now appears within the Build Query dialog box. In the WHERE tab, change the Condition to "Marketing". The Union SQL statement for the Marketing Department has now been created.

Click the **Union All** icon  again to create the third query and name it "Training".

Click **Paste** again and change the Condition to 'training' in the WHERE tab of the Build Query dialog box. This creates the Union SQL statement for the Training Department.

The screen should now look as shown in the figure below. Note the existence of the main query and the two additional queries used for the union.



Click "View SQL" to review the resulting SQL shown below.

```
SELECT Employees.last_name,
Employees.first_name,
Departments.dept_name
FROM Employees INNER JOIN Departments ON Employees.dept_id = Departments.dept_id
WHERE Departments.dept_name ='Sales'
UNION ALL
SELECT Employees.last_name,
Employees.first_name,
Departments.dept_name
FROM Employees INNER JOIN Departments ON Employees.dept_id = Departments.dept_id
WHERE Departments.dept_name = 'Marketing'
UNION ALL
SELECT Employees.last_name,
Employees.first_name,
Departments.dept_name
FROM Employees inner join Departments ON Employees.dept_id = Departments.dept_id
WHERE Departments.dept_name = 'Training'
```

## Results

Executing the query as a Dashboard produces the results shown below in which the three queries are merged together to obtain employees from the Sales, Marketing, and Training Departments.

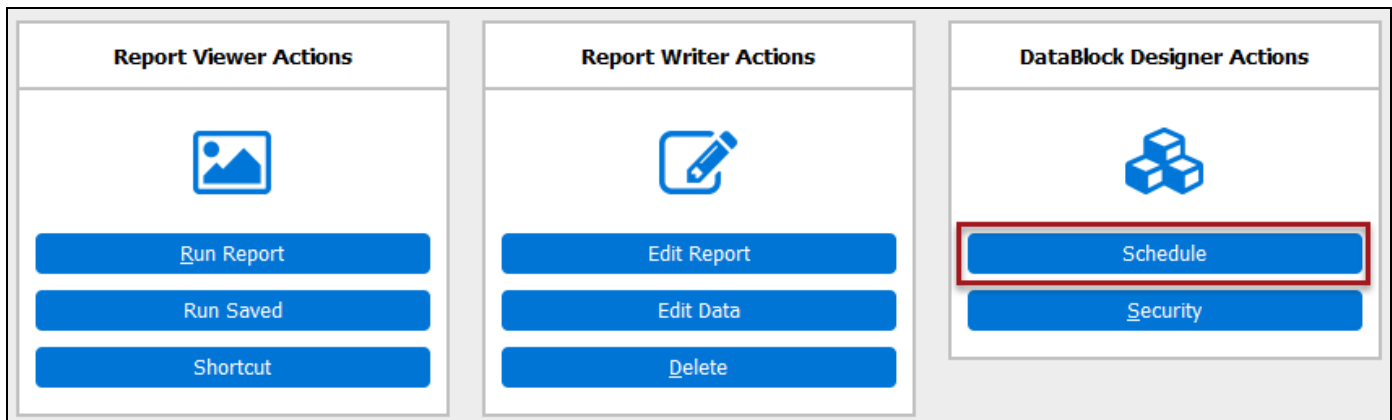
last_name	first_name	dept_name
Washington	Mark	Sales
Pham	Christina	Sales
Mendez	Lauren	Sales
Alden	George	Sales
Kline	Chris	Sales
Nelson	Gina	Sales
Tinoco	Carlos	Sales
Tisdale	Isaac	Sales
Farris	Victoria	Sales
Patterson	Jane	Sales
Gibson	Susan	Sales
Kern	Doug	Sales
Jackson	Mary	Marketing
Bailey	Millie	Marketing
Holmes	Lincoln	Marketing
Hai	Paul	Marketing
Wallace	Gary	Marketing
Ford	Jeremy	Training
Hale	Gus	Training
Manalo	Aaron	Training
Roberts	Donald	Training
Becker	Brian	Training

# Scheduling Reports

This section discusses report scheduling and delivery, which is one of four [modules](#) that can be added to Argos. To check if your institution is licensed for the Scheduling & Delivery module, a MAPS administrator can go to the License screen in the MAPS Config and select *Argos*. If you are not licensed for this module, you will see a message to this effect when attempting to create a schedule in Argos.

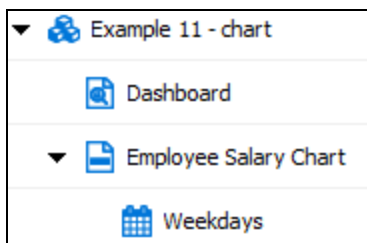
Report Scheduling allows Argos Administrators to automate the process of running a report. Reports can be scheduled to run on certain days, at specific times, or periodically with a specified frequency. Along with running the report, you can also tell the schedule to perform tasks or create various kinds of output.

To schedule a report, select an existing report and click the **Schedule** button on the right-hand pane of the Argos Interface as shown in the figure below. You can also right-click on a report and select **Schedule**.



The **Edit Schedule** dialog will then appear, where you configure the various options for the schedule.

Schedules are listed underneath the report that they pertain to. In the screenshot below, the schedule "Weekdays" has been created for the Employee Salary Chart report.



You can create any number of schedules for any type of report in Argos. Different schedules could be used to produce different output types at varying frequencies, as needed.

## Editing Schedules

The **Edit Schedule** dialog consists of five tabs that allow you to configure various aspects of the schedule.

- [General](#)
- [Schedule](#)
- [Tasks](#)
- [Events](#)
- [API](#)

The **Test** button at the bottom of the Edit Schedule dialog allows you to test execution of the schedule at any time. Clicking Test executes all configured tasks for the schedule immediately, using your own permission settings.

**Warning:** If you have configured the Email task for a schedule, please be certain that you intend to send the test output to the recipient(s) specified in the To field. If you are testing a production schedule, you may wish to replace the list of recipients with your own email address.

### Schedule Permissions

If a "Run As" user has been specified in MAPS for this schedule or if a system-level user has been specified for running all schedules, when the schedule runs it will execute as the designated user instead of as your own user. This user may have different permissions for the database than you do. You should contact your institution's MAPS administrator if you have a question regarding schedule permissions.

## General

The screenshot shows the 'Edit Schedule' dialog box with the 'General' tab selected. The dialog has a blue header with the title 'Edit Schedule' and a calendar icon. Below the header is a subtitle: 'Edit details of this schedule report such as name and active status.' The 'General' tab is active, showing a text field for 'Name' containing 'Dashboard Schedule'. There are two checkboxes: 'Active' (checked) and 'Delete after final run' (unchecked). Below these is a 'Maximum timeout (# minutes)' section with a text field containing '0' and a checked 'Use default' checkbox. At the bottom is a 'Note:' section with a large text area. The dialog has four buttons at the bottom: 'Test', 'OK', 'Cancel', and 'Help'.

The General tab allows you to give the schedule a name, make the schedule active, and specify if this schedule is to be deleted after it runs for the last time (if a last date is specified).

If the **Active** box is not checked, this schedule will never execute. If you are still testing a schedule, you may wish to uncheck this box while it is in development.

**Note:** It is very important to give your schedule a descriptive name. As multiple schedules are put into production, it becomes very difficult to administer them in MAPS if they all have the same name.

## Schedule

**Edit Schedule**

Configure the date and time settings to run the scheduled report.

General | **Schedule** | Tasks | Events | API

Next scheduled date/time: 09/16/2016 10:22:09 AM

Final date of execution: // 12:00:00 AM

Specify the days of the week that this schedule can run on

Monday  Wednesday  Friday  Saturday

Tuesday  Thursday  Sunday

Frequency: Only once Amount: 1

Only run between 6:00:00 AM and 6:00:00 PM

Actual next run date/time:  
Monday, September 16, 2013 at 10:22:09 AM

Test OK Cancel Help

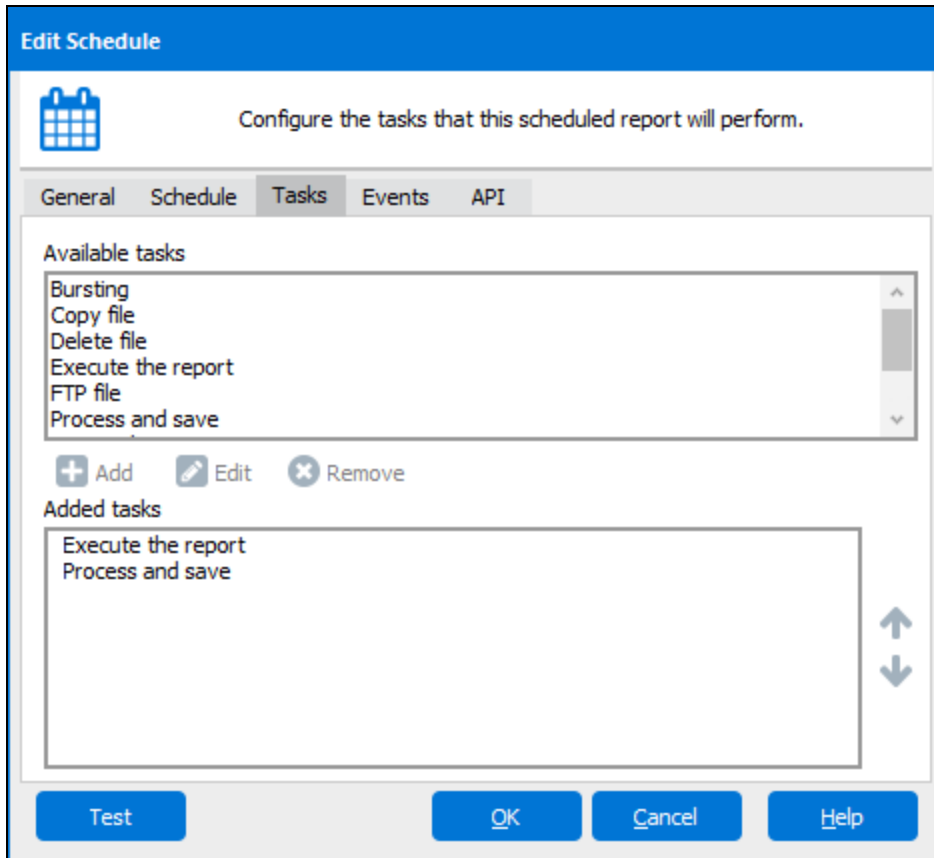
The Schedule tab allows you to control when a schedule will run:

- **Next scheduled date/time** - Select the next date and time that the schedule will run. **Note:** Schedules execute based on the timezone of the server where MAPS is installed; if your local machine uses a different timezone, you may wish to adjust the scheduled time accordingly. You should choose a next scheduled date/time in the future, as the schedule will execute immediately upon saving if you enter a date/time that has already passed.
- **Final date of execution** - Select the last date and time that the schedule should run.
- **Specify the days of the week that this schedule can run on** - This check box will allow you to specify on which day(s) the schedule is permitted to run. Note that this does not necessarily mean that the schedule will run on that day, only that it is permitted to do so.



- **Frequency** - This determines how often the schedule will run. You have the following options:
  - **Only once**
  - **Every day** - Increments the run date by one day.
  - **Every week** - Increments the run date by one week.
  - **Twice per month** - The first execution date is determined by the time you setup the schedule. From that point forward the run time is determined by the following logic: If the first execution date was in the first half of the month, the next execution date is determined by adding half of the days in the month to the first execution date. If the first execution date falls in the second half of the month, the next execution date is determined by incrementing the date to the end of the month, then adding half the days of the following month.
    - For example, if the first execution date is set at 6/10, the next execution date would occur on 6/25 (15 days after the first execution date). For the succeeding months the jobs would then run on the 10th and 25th of each month.
    - Similarly, if the first execution date is set at 6/20, the next execution date would occur on 7/15 (incrementing the first execution date to the end of the month then adding 15 days). Jobs would then run on the 15th and 20th of each succeeding month.
  - **Every month** - Increments the run date by one month. This selection has an additional option, "From the end of the month". Checking this box will calculate the next run date based on an offset from the end of the month. For example, if the next scheduled date is Feb 28th, checking this box will cause the run date after that to be March 31st instead of March 28th.
  - **Twice per year** - Increments in the same manner as **Twice per month** except the term is by year rather than by month.
  - **Every year** - Increments the run date by one year.
  - **# of minutes** - Increments the run date by the specified number of minutes.
  - **# of hours** - Increments the run date by the specified number of hours.
  - **# of days** - Increments the run date by the specified number of days.
  - **# of weeks** - Increments the run date by the specified number of weeks.
  - **# of months** - Increments the run date by the specified number of months. This selection has an additional option, "From the end of the month". Checking this box will calculate the next run date based on an offset from the end of the month. For example, if the next scheduled date is Feb 28th, and the month increment is 3, checking this box will cause the run date after that to be May 31st instead of May 28th.
  - **# of years** - Increments the run date by the specified number of years.
- **Only run between [time] and [time]** - Specify the times that the schedule is permitted to run. This option can be used to configure, for example, a schedule that runs every hour between 8 AM and 5 PM. When not checked, schedules may run at any time.
- **Amount** - Choose the increment of time to elapse between schedule runs. For example, if **# of minutes** is selected and **Amount** was set to 5, then the schedule will run every 5 minutes.
- **Actual next run date/time** - displays the next scheduled run date and time.

## Tasks



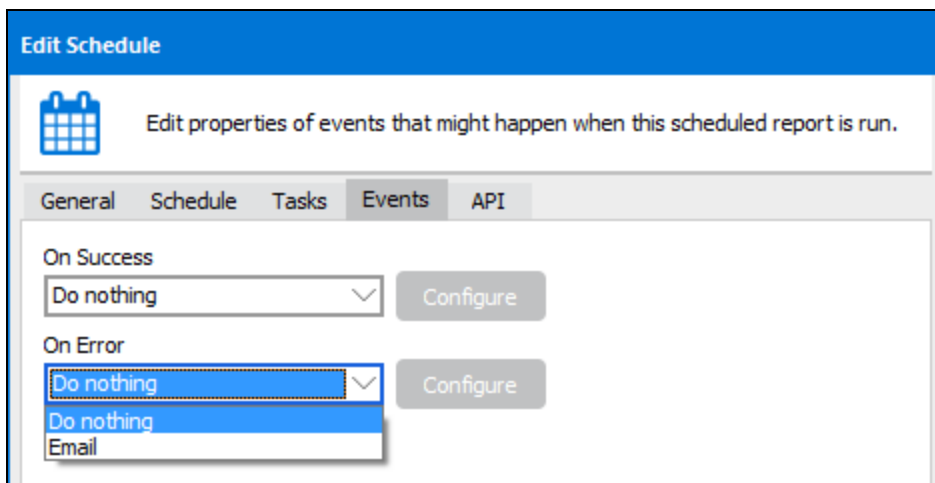
The Tasks tab allows you to control the events performed during execution of a schedule. The **Execute the report** and **Process and save** tasks are added by default since they are required in order for the schedule to do anything. The following tasks are available:

<b>Available Tasks</b>	<b>Description</b>
<a href="#"><u>Bursting</u></a>	<b><i>Bursting allows you to separate out the report into several individual reports.</i></b>
<a href="#"><u>Copy file</u></a>	<b><i>Copies the output file from the schedule to a location on the network.</i></b>
<a href="#"><u>Delete file</u></a>	<b><i>Delete a file from a location on the network.</i></b>
<a href="#"><u>Execute the report</u></a>	<b><i>Brings up the parameter entry form so you can select the parameters needed for the execution of the report (if necessary).</i></b>
<a href="#"><u>FTP file</u></a>	<b><i>FTP the report results to a specified destination.</i></b>
<a href="#"><u>Process and save</u></a>	<b><i>Choose the output format for the report. You can choose from the following options:</i></b> <ul style="list-style-type: none"> <li>■ Print the report</li> <li>■ Save it as a .PDF</li> <li>■ Save it as a .XLS</li> <li>■ Save it as a .RTF</li> <li>■ Save it as a .TXT</li> </ul>

Available Tasks	Description
<a href="#">Run application</a>	<i>Execute an application such as a batch file.</i>
<a href="#">Save execution state</a>	<i>Save the data as it was at the time the report was run so that it can be reproduced exactly as it was at that time. When you run the report, you will be prompted to select which saved execution state you would like to use. You can manage the <a href="#">Saved Execution States</a> when you edit the report.</i>
<a href="#">Send an Email</a>	<i>Send an email and distribute the report to other users. The report will be sent out in the output type selected in the Process and save task.</i>

Tasks execute in the order specified, from top to bottom. To add a task from the Available tasks, double-click the task, or select the task and click the **Add** button. After entering the configuration details for that task, it will appear in the **Added tasks** list and will execute when the schedule runs.

## Events



The Events tab allows you to specify an event to perform upon success or failure of a scheduled report. You can either send an email or do nothing for each condition. When opting to send an email, use the Configure button to configure the email.

## API

The screenshot shows the 'Edit Schedule' interface with the 'API' tab selected. The interface includes a header 'Edit Schedule' and a sub-header 'Edit the API settings for this schedule. API functionality is accessible via the built-in MAPS HTTP server.' Below this are tabs for 'General', 'Schedule', 'Tasks', 'Events', and 'API'. The 'API' tab contains a checked checkbox 'Make this schedule accessible to API', a 'Schedule Unique Identifier' field with a sample value '2UG7JTH67ZLMH2L4FEEV3RESPM4R3WRZA3TBI' and 'Sample POST'/'Sample GET' buttons, and 'User Name (optional)' and 'Password (if LDAP user)' fields.

The API (Application Programming Interface) tab allows Argos administrators to make a report accessible via the Argos API. The information on this tab is only visible to administrators. An API accessible report can be called without having to log into Argos. The call can be made from a web page, portal or other 3rd-party application. Note that a MAPS administrator must specifically identify the servers that are permitted to make API calls to Argos. This is performed in the MAPS Config on the HTTP Referrers screen.

API calls can use a variety of techniques to authenticate users. One option is to pass the username and password on the API call. Another option is to enter the username and password on this form (password only required if user is LDAP).

For more information, please refer to the [Argos API](#) page.

Option	Description
<b>Make this report accessible to API</b>	<b>Set this checkbox to make this report API accessible. Once the checkbox is marked, MAPS will generate a Report Unique Identifier for the report.</b>
<b>Password</b>	<b>If the default user is an LDAP user, you must assign a password here or pass in the password via the API call.</b>
<b>Report Unique Identifier</b>	<b>Identifies the report. This will be one of the parameters passed in an API call to MAPS. If the report or the DataBlock are moved or copied, a new Report Unique Identifier will need to be generated.</b>
<b>Sample GET</b>	<b>Clicking on the Sample GET button will copy an example of the HTML GET code to the clipboard that can then be placed in a web page.</b>
<b>Sample POST</b>	<b>Clicking on the Sample POST button will generate an example of the HTML POST Form in its simplest form. POST forms can also be created to have checkboxes, listboxes, drop down boxes and other controls. The API call in POST form will be copied into your Windows clipboard for later use.</b>
<b>User Name</b>	<b>A default username and password for this report may be defined here. If there is a username and password that is passed in via the GET, POST form, it will override the username and password entered here.</b>

## Copying Schedules

You can copy a schedule within a report, or from one report or DataBlock to another. Copying a schedule preserves all of that schedule's settings, with the exception of the "Active" checkbox and the API settings.

When copying a schedule to a different report or DataBlock, it is very important to verify that all of the settings are still correct under the new location. In particular, if you copy a schedule from one report type to another, you may need to edit the various tasks to make sure any generated files have the correct file extension. For example, when copying a schedule from a banded report to a CSV report, you will need to adjust the file extension in the "Send an Email" task from .pdf to .csv.

Similarly, if you copy a schedule from one DataBlock to another you may need to edit the "Execute the report" task to use parameters from the new DataBlock.

## Execute the Report Task

Argos automatically adds the "Execute the report" task to the list of tasks. If you would like to enter parameters to be used when running the schedule, double-click on the **Execute the report** task or select this task and click **Edit**.

If you have set the AutoSelect property of a list or drop-down object to "yes" in the datablock, then the very first entry in the list or drop down will be used as the value when running the schedule.

Auto Select	No
Choices	(Choices)
Columns	(Columns)
Cursor	Default
Description	
Enabled	Yes
Font	(Font)
Font.Bold	No
Font.Color	Window Text
Font.Italics	No

Otherwise, you need to enter the values to be used for each parameter on the form.

Ar Edit Execute Task

Configure the parameter values

Department Name: HR

Location:

DropDown1  
DropDown2

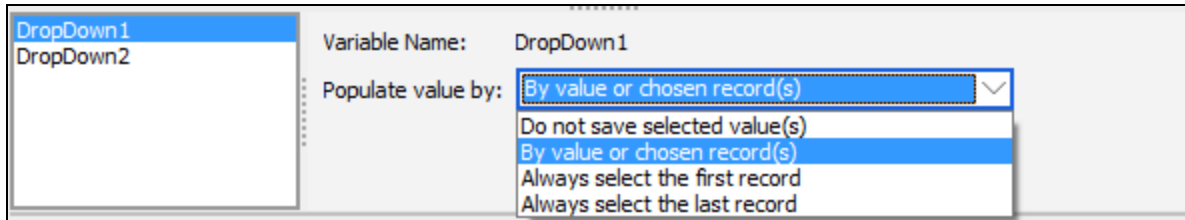
Variable Name: DropDown2

Populate value by: By value or chosen record(s)

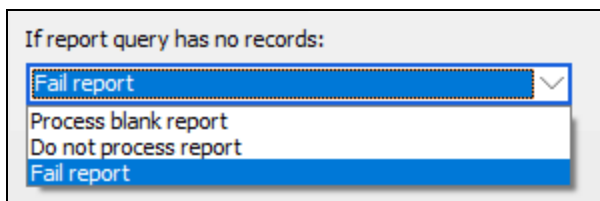
Select a parameter in the box in the lower left corner, then use the "Populate value by" dropdown to specify the value for this variable. The available choices depend on the type of field.

<b>Populate Value By</b>	<b>Description</b>
<b>Do not save selected value(s)</b>	<i>Use for data-aware fields where you want the value to be calculated at run time.</i>
<b>By value or chosen record(s)</b>	<i>Specify a value or choose a specific item from a list.</i>
<b>Always select the first record</b>	<i>Select the first item in a list.</i>
<b>Always select the last record</b>	<i>Select the last item in a list.</i>
<b>Offset report run date by</b>	<i>Enter a number of days, weeks, months, or years to offset the run date by.</i>

Clicking one of the input fields will activate the corresponding object name (in this case DropDown1 and DropDown2).



The option selected for **If report query has no records** determines how reports with zero results will be handled.



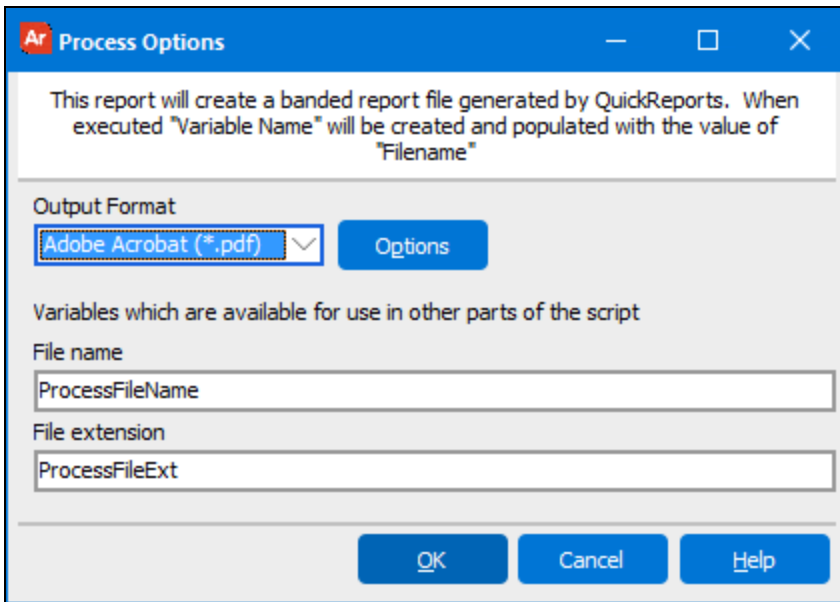
- **Process blank report** produces the report as usual, but with blank results.
- **Do not process report** ends the execution of the report, but does not generate a failure condition.
- **Fail report** generates an error if the query returns no results. This error condition can be used to trigger a schedule error email. See the description of the Events tab in "Editing Schedules" on page 110 for more information.

Click **OK** when finished. These parameters will be used when the scheduled report executes.

## Process and Save Task

The **Process and Save** task appears in the **Added task** list by default. This is the task that saves a copy of the report on the server where MAPS is installed; therefore, it must be included in the task list in order to generate output from a scheduled report.

Editing this task allows you to configure various options regarding the output format and the names of the variables used to store the file name and extension.



**Output Format:** Select the desired output format. The output format also determines the default file extension that will be used, although you can choose to customize this in the FTP, Email, and Copy tasks.

The available output formats differ depending on the report type:

- **Banded report:** Print (send output to the specified printer and tray), .pdf, .xls, .rtf, or .txt.
- **CSV report:** .csv
- **XML report:** The output type and file extension are defined in the report settings.

**File name:** This variable stores the name of the output file. You can use the variable as part of other tasks by typing `%%ProcessFileName%%`. To change the variable name, edit it here.

**File extension:** This variable stores the default file extension, determined by the report type and the selected output format. You can use the variable as part of other tasks by typing `%%ProcessFileExt%%`. To change the variable name, edit it here.

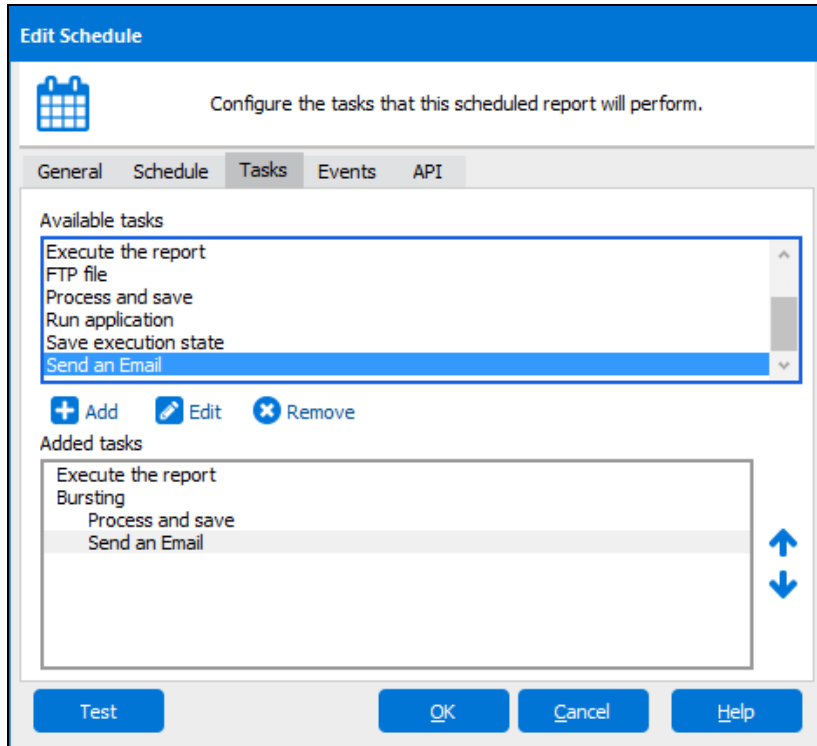
## PDF Options

The available output formats include print, PDF, XLS, RTF, and plain text. If you select PDF format, you can specify additional [PDF Options](#).

## The Bursting Task

Bursting is typically used to separate individual records of a report. For example, reports containing information about employees can be burst to print each employee on a separate page. Depending on the output format selected in the process and save task, each record can be sent as a separate email, printed on a separate page, saved as an individual file, etc.

Note that the process and save and email tasks are sub-tasks of the bursting task:



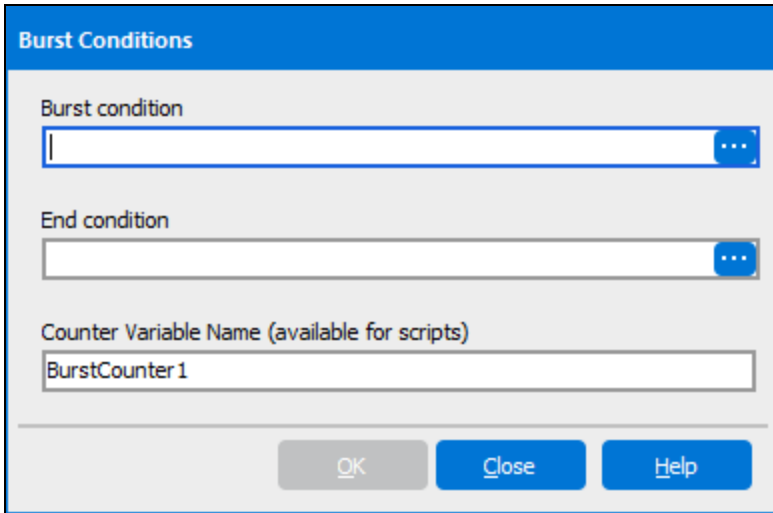
The task order in the image above means:

1. The report executes
2. For each record:
  1. Process that record and save it
  2. Send that record via email.


You can use the up and down arrow buttons to move tasks underneath other tasks. For example, selecting the execute task and clicking the down arrow would move it to a sub-task of the bursting task. Clicking the down arrow again would move it down the list of bursting tasks, and then to the position after the bursting task.

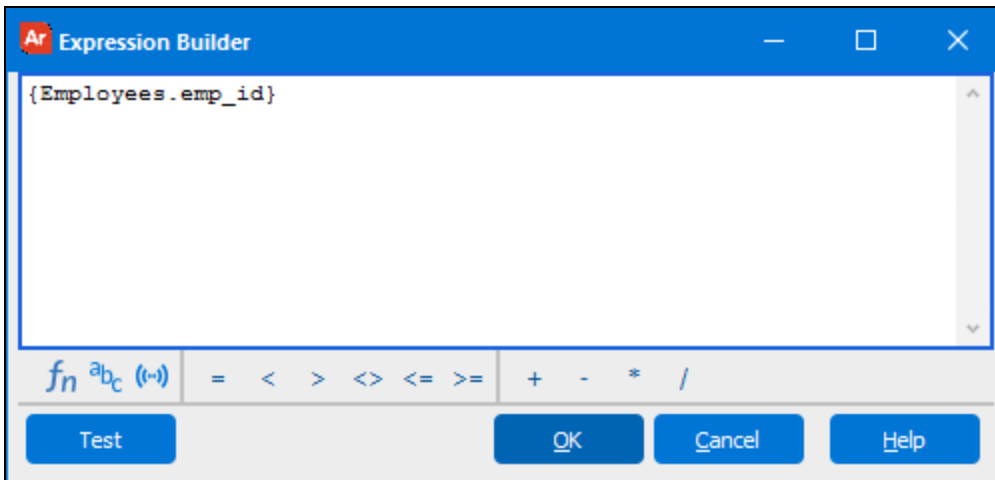


When you add the bursting task to the added tasks list, the following dialog appears:



The image shows a dialog box titled "Burst Conditions". It has three input fields: "Burst condition" (empty), "End condition" (empty), and "Counter Variable Name (available for scripts)" (containing "BurstCounter 1"). Each of the first two fields has a blue ellipsis button to its right. At the bottom, there are three buttons: "OK", "Close", and "Help".

The only required field on this form is Burst condition. To enter a Burst condition, click on the ellipsis  which brings up the Expression Builder. In this example, the employee ID was selected as the Burst condition. This creates a separate report for each employee.



The image shows the "Expression Builder" dialog box. The title bar says "Ar Expression Builder". The main area is a text box containing the expression "{Employees.emp\_id}". Below the text box is a toolbar with various symbols: a function symbol  $f_n$ , a subscript  $a_b$ , a superscript  $c$ , a negation symbol  $(-)$ , an equals sign  $=$ , less than  $<$ , greater than  $>$ , less than or equal to  $<=$ , greater than or equal to  $>=$ , plus  $+$ , minus  $-$ , multiplication  $*$ , and division  $/$ . At the bottom, there are four buttons: "Test", "OK", "Cancel", and "Help".

## Copy File Task

To save the output as a file on the network, the “Copy File” task must be added as an available task after the Process and Save task. The Copy File Task allows you to copy the source file (the variable mentioned above - ProcessFileName) to a destination on the network using standard file naming conventions. The destination directory must have been identified in the MAP Server Configuration Tool (by your MAPS Administrator) as a directory available to MAPS applications. See your MAPS Administrator to obtain the destination directory where you can save files.

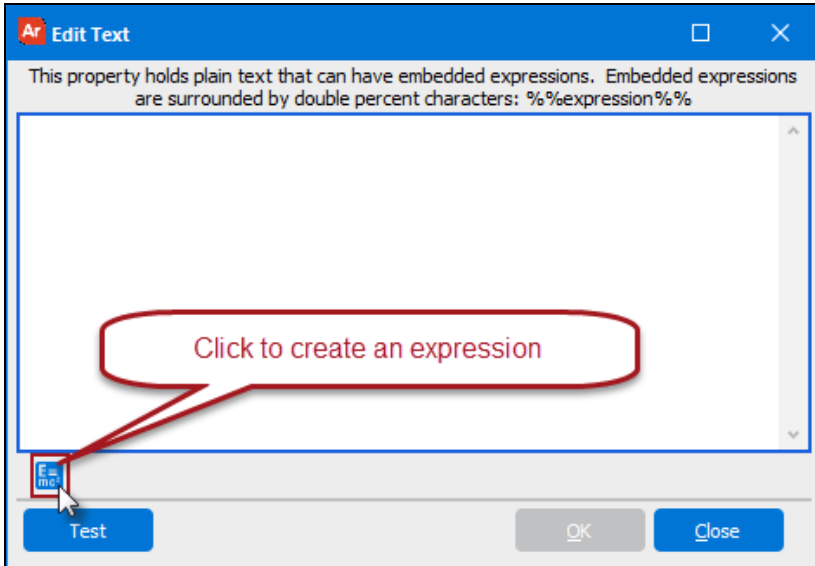
The directory on the MAP server where the MAPS applications files are to be stored is entered in the Server/File Operations option within the MAP Server Configuration Tool. The user account running the scheduled report must also have permission to store files on the network drive. Check with your MAPS Administrator to see if you have permission. The MAPS Administrator determines which users can run a schedule via the Scheduling option in the MAP Server Configuration Tool.


The Destination File name must be consistent with the file naming convention of the Operating System the file is to be stored upon.

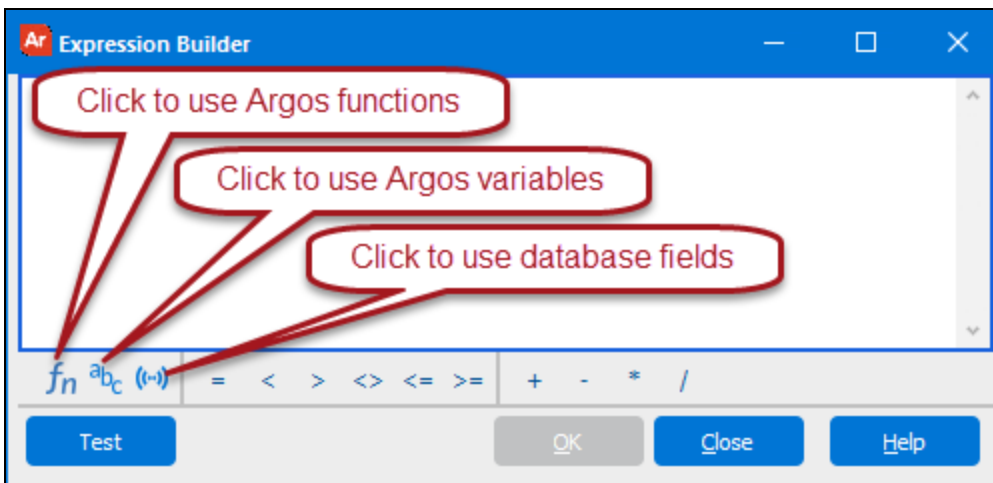
For example, to save the output file named BandedReport.pdf on the c:\reports\directory on a MAP Server named mapsprod, enter the following in the Destination File field: **\\mapsprod\c\$\reports\BandedReport.pdf**. As mentioned above, the c:\reports directory on the MAP server must have been identified in MAPS as the directory to store files.


## Using an Argos variable in the Destination File Name

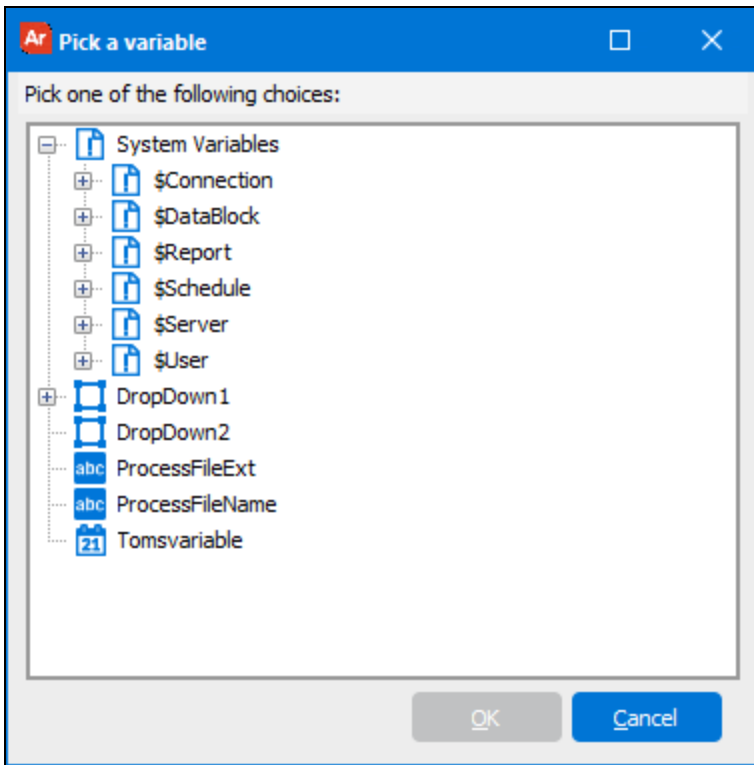
You can also use an expression to create the Destination File Name by clicking the ellipsis to the right of the Destination File field (see above figure). This brings up the Edit Text dialog box shown below.



Click the expression icon  to bring up the Expression Builder shown below. Within this dialog you can build expressions using Argos functions, Argos system variables, and database fields by clicking the icons shown below.



For example, to create a file name consisting of the Banded Report name followed by .pdf extension, use the Expression Builder to create the expression. The Argos System Variable \$Report.Name contains the Report name. Selecting the variables icon  displays the "Pick a variable" dialog where you can select System and Argos variables.



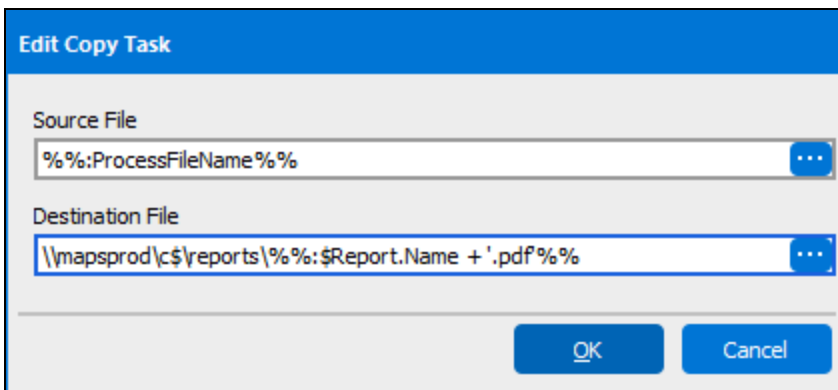
Expand the **\$Report** System variable and select Name. Click OK, and complete the following expression.

```
\\mapsprod\c$\reports\%%:$Report.Name + '.pdf'%%
```

The Report Name and extension is appended to the destination directory.

Click the **Test** button to test your expression.

After clicking **OK**, the completed Destination File name is displayed in the "Edit Copy Task" dialog below. Click OK to conclude.



**Note:** Any variables used in the file name must be surrounded by %% characters. If you use the expression builder to construct the filename, the %% will be added automatically.

## Using dates in the Destination File Name

You may want to include the date the scheduled report was run in the Destination File Name. The Argos Now() function returns today's date and can be used in the expression to build the file name. However, since the Now() function returns the date with slash characters (which aren't allowed in file names), you must strip the slashes to build a valid file name. The expression below will build the file name including today's date with slashes removed. The FormatDate function is used to remove slashes from the date obtained from the Now() function:

```
\\mapspod\c$\reports\%%FormatDate(Now(),"yyyymmdd")%%.pdf
```

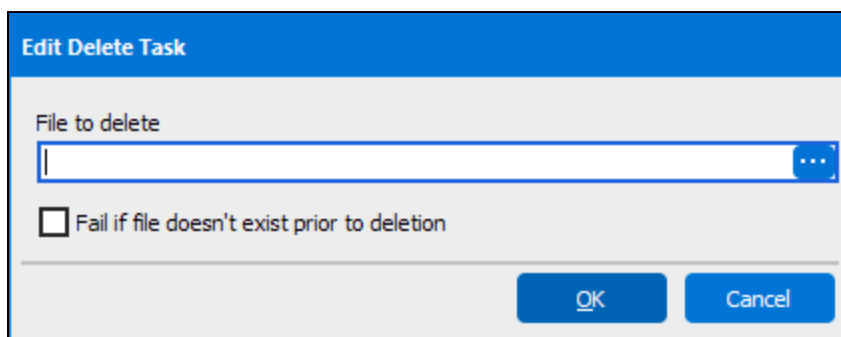
The saved file name will then be named yyyymmdd.pdf

## Use of Source File Name

In the above examples, the default Source File name (ProcessFileName) was used throughout the example. If you are only scheduling one report it is not necessary to change the Source File name. However, if your schedule includes several reports with output to be saved, you will need to assign a Source File name (in the Process Options dialog) to each of the reports. Then in the "Edit Copy Task" dialog (see above figure), the corresponding Source File name is entered.

## Delete File Task

The Delete File task allows you to enter a path to a file that should be deleted when the schedule runs. This could be an intermediary file or any other file you wish to remove.

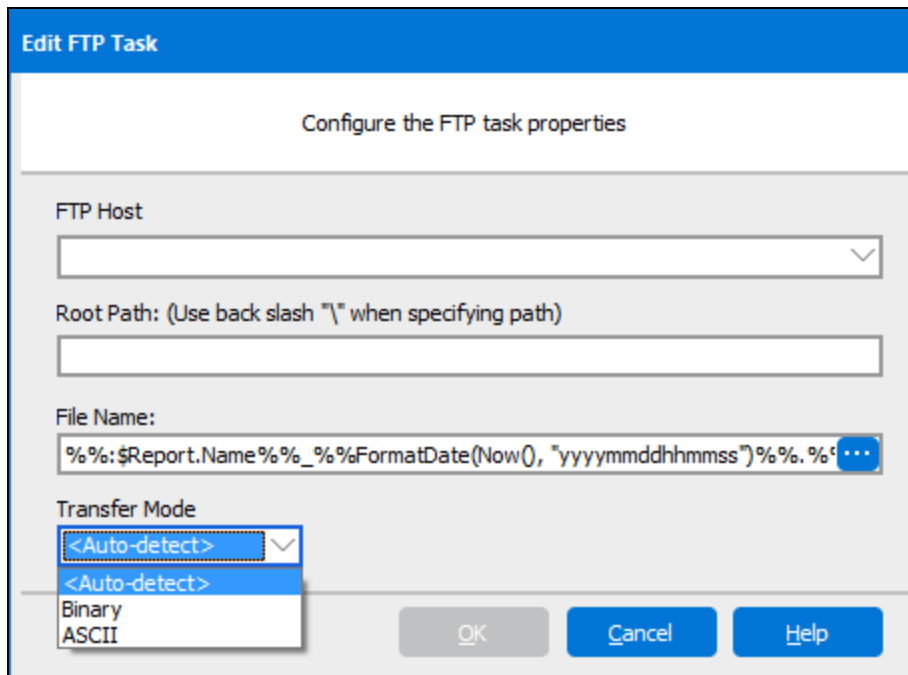


Use the ellipsis button if you want to use the expression editor to construct the path and filename of the file to be deleted.

The option to **Fail if file doesn't exist prior to deletion** will cause the schedule to fail if the file to be deleted does not exist. This failure condition can be used to trigger a failure email in the [Send an Email task](#).

## FTP File Task

Selecting the FTP File task displays the dialog box shown below which allows you to identify the destination and file name of the scheduled report to be sent via FTP.



**FTP Host** - choose an existing FTP host name from the drop down list. Connection names are created within MAPS by a MAPS Administrator.

**Root Path** - If the Root Path was not specified in MAPS, enter the path name. If the Root path was specified in MAPS, it will be shown in this field. You may not change the path, but you can add folders beneath the provided path.

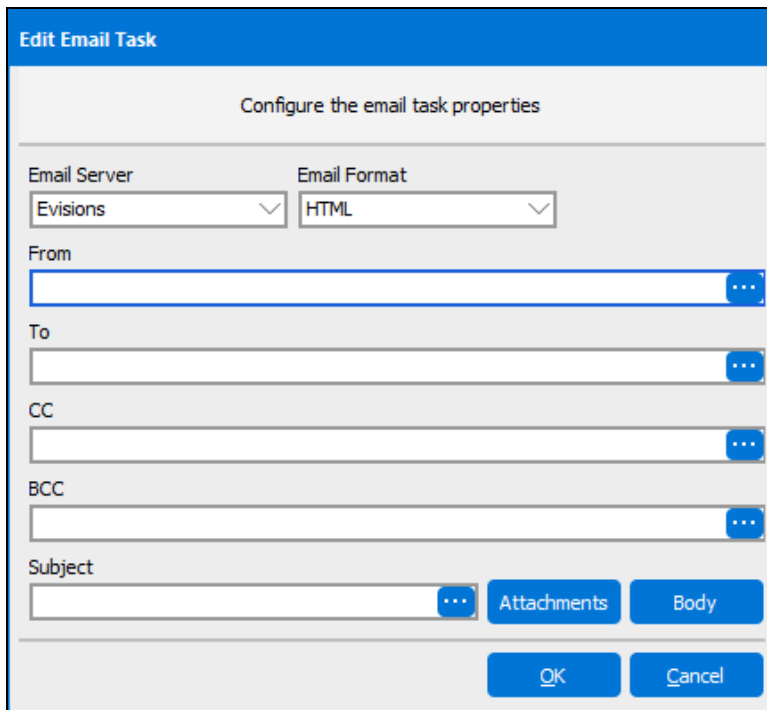
**File Name** - The name of the file as it should be saved on the FTP server. The default name of the file is shown in the figure above. The default name consists of a concatenation of the Report Name (obtained from the Argos Variable `$ReportName`) with the date and time the report was run, followed by the file extension (from the `%%ProcessFileExt%%` variable configured in the [Process and Save](#) task).

- To customize the filename of the output, edit the File Name field as desired. The ellipsis button brings up a text editor that gives you access to the [expression builder](#), which allows you to construct more complex filenames.
- You can replace the file extension variable with any file extension of your choosing, or remove it (and the preceding ".") to have no file extension.

**Transfer Mode**- Choose the transfer mode from the drop down box. If Auto-detect is selected, Argos will choose between Binary and ASCII based on the file type to be transferred.

## Send an Email Task

Selecting the Send an Email task displays a dialog box that allows you to configure an email to be sent. By default, the generated report is included as an attachment to the email.



**Email Server** - Select an email server from the drop-down. This list includes all of the email servers that have been added to MAPS that are authorized for use with Argos and which you have permission to use.

**Email Format** - You have the option to use either plain text or HTML for the email body. If you choose HTML, note that you will need to specify line breaks with the <br> tag.

**From, To, CC, and BCC** - Enter the desired email address(es). To enter an expression or variable, use the ellipses buttons to open the expression editor.

**Subject** - The subject line for the email. Select the ellipsis button if you want to use the expression editor to construct the subject line.

## Email Attachments

Use the **Attachments** button to configure any attachments that should be included with the email. The scheduled report is included by default, but you can add more attachments or modify the filename of the report if desired.

**Edit Email Attachments**

Attachments

- file\_%%FormatDate(Now(), "yyyyMMddhhmmss")%%.pdf  
MyAttachment.pdf

+ Add    x Delete

Filename (at time of execution on the server)  
%%:ProcessFileName%%

Attachment Filename (the name the recipient will see)  
file\_%%FormatDate(Now(), "yyyyMMddhhmmss")%%.pdf

MIME Content-Type  
<Let MAPS determine>

OK    Close

**Add/Delete** - Add or delete an attachment.

**Arrows** - Use the up and down arrows to change the order of the attachments as they will appear in the email.

**Filename (at time of execution on the server)** - The name of the file, as it exists on the MAPS server during runtime, which should be attached to the email.

**Attachment Filename** - The name of the attachment that the recipient will see in their email.

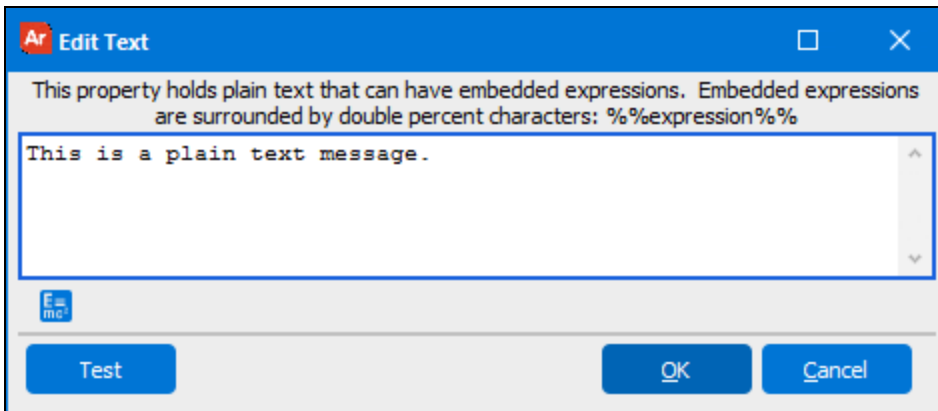
You can use variables, expressions, and other constructs in the expression editor to edit the filename and attachment filename. Enclose variables and expressions in double percent signs( %%:variable%%) to evaluate them.

**MIME Content-Type** - The content type of the attachment. You can normally leave this set to the default of <Let MAPS determine>.



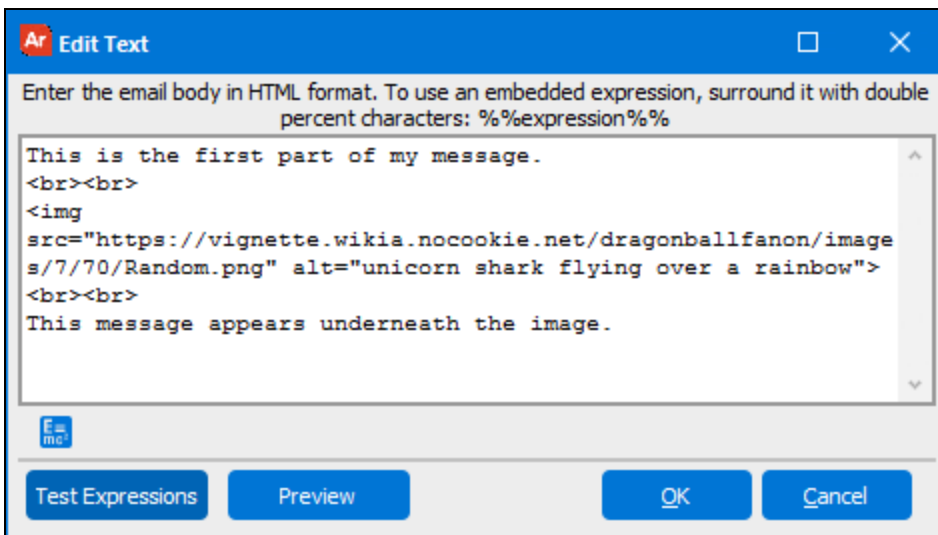
## Email Body

Use the **Body** button to construct the body of the email. The edit dialog will differ depending on whether you chose plain text or HTML as the email format. The plain text editor allows you to enter the desired text, as well as embed expressions using the double percent notation.



- The [Expression Builder](#) is available underneath the text edit box to assist you with creating expressions.
- Use the **Test** button to see a preview of the message with the expressions evaluated.

If you chose HTML as your email format, the editor will allow you to enter HTML tags to construct a fully-formatted email.

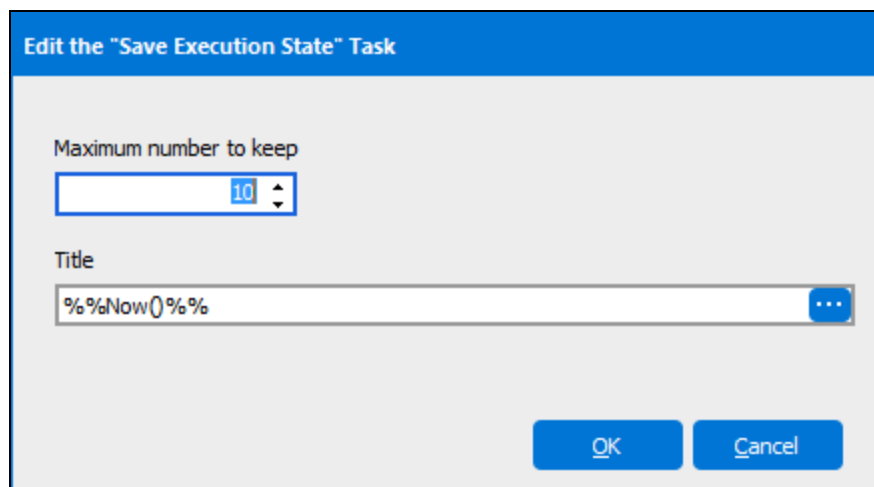


- Remember that you will need to use the `<br>` tag for line breaks when using HTML.
- The [Expression Builder](#) is available underneath the text edit box to assist you with creating expressions.
- Use the **Test Expressions** button to see what the HTML will look like with the expressions evaluated.
- Use the **Preview** button to see a preview of the email with the HTML rendered in your browser.

## The Save Execution State Task

One of the available schedule tasks is the ability to save an execution state. This allows you to save the data that was on the dashboard at the time the report was run. The saved data includes the values of form variables and any OLAP cube data. It does **not** include the results of the report query, which will run live each time you run the report. Each saved execution states is stored in a file on the server where MAPS is installed, so that it can be loaded at a future date.

When you select the "Save execution state" task, the dialog box in the figure below appears. You can enter the maximum number of execution states to save for the report involved. You can also create a title for the execution state. As you can see, the default title is today's data and time (using the Argos "Now" function) and is saved as an Argos variable.



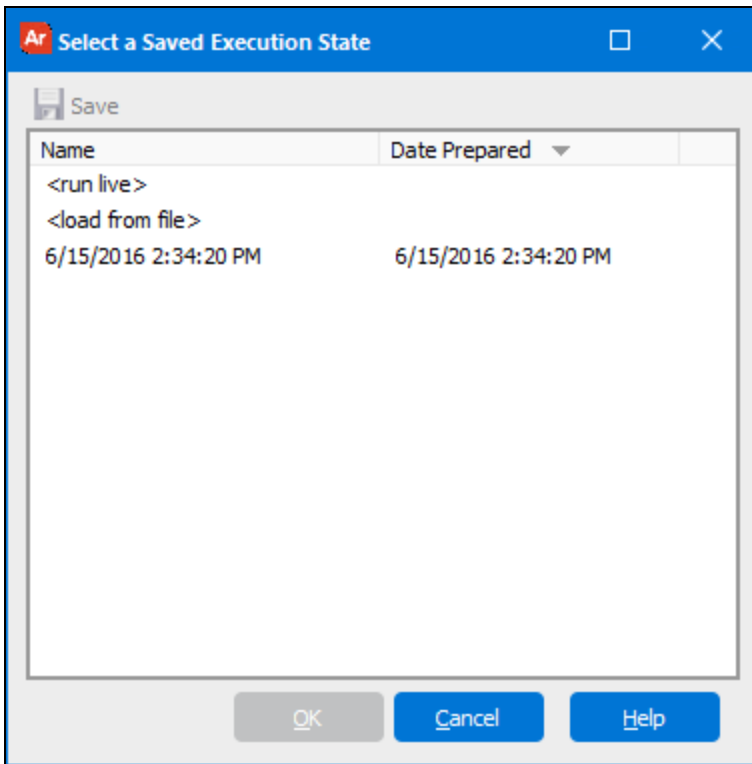
The screenshot shows a dialog box titled "Edit the 'Save Execution State' Task". It features a blue header bar. Below the header, there are two input fields. The first is labeled "Maximum number to keep" and contains the value "10". The second is labeled "Title" and contains the text "%%Now()%%". To the right of the title field is an ellipsis button. At the bottom of the dialog are two buttons: "OK" and "Cancel".

Clicking the ellipsis at the right of the Title field launches the Expression Builder where you can create an expression to be used as the title.

**Note:** Any variables used in the title must be surrounded by %% characters. If you use the expression builder to construct the title, the %% will be added automatically.

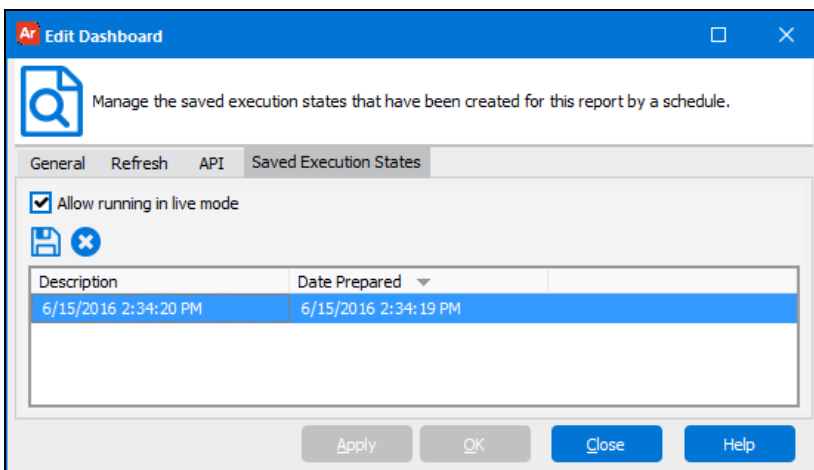
## Running Saved Execution States

After the execution state is saved, when running the report a list of saved execution states will be listed as shown in the figure below. Running saved execution states is described fully in the Argos Report Viewer Guide.



## Managing Saved Execution States

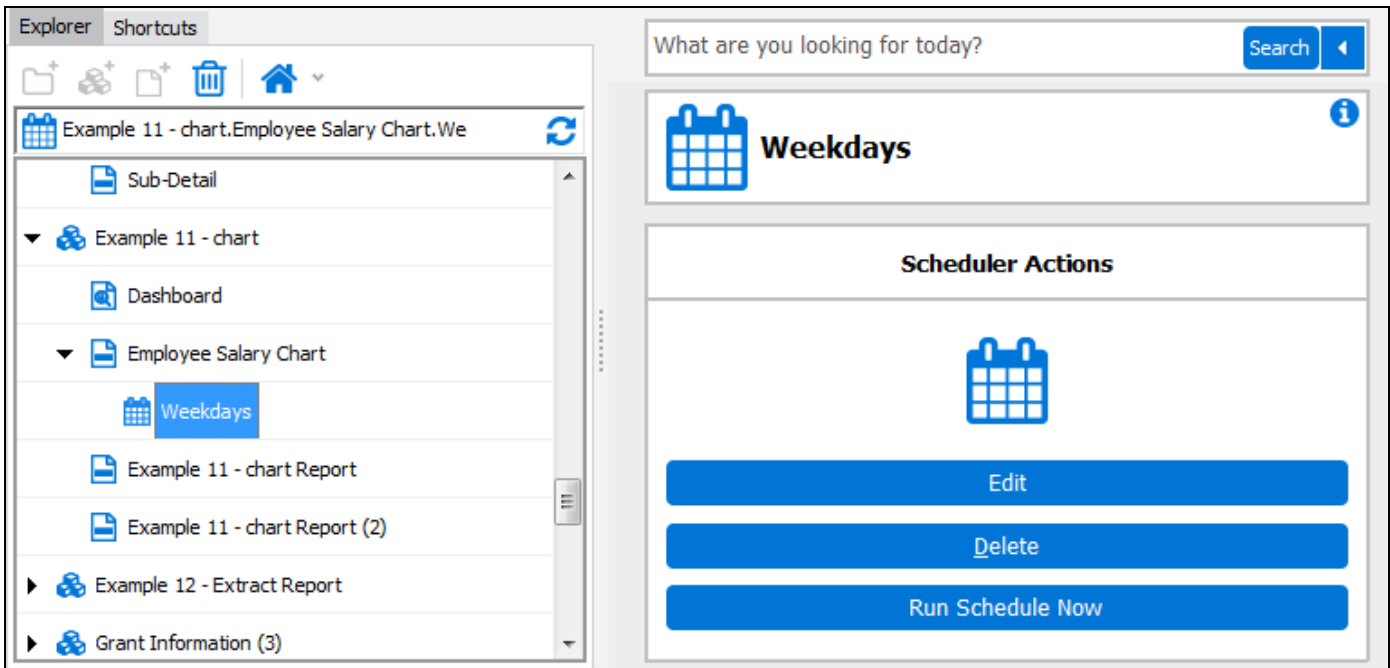
Saved execution states can be managed (saved to any location on the network, or deleted) by the Report Writer or DataBlock Designer. This is done by editing the report and selecting the Saved Execution States tab as shown below. To edit a report, highlight the report in the Explorer tree then select Edit Report within the design area. Note the options to save the execution state to any desired location or to delete it.



Checking the **Allow running in live mode** box allows the report to be run using the current state of the database.

## Managing Schedules

Once a schedule has been created, you can access it by selecting the schedule underneath the report.



For users who have the ability to modify schedules, you will see the list of available **Scheduler Actions** on the right. The available actions include the ability to **Edit** or **Delete** the schedule.

You can also choose to execute the schedule on demand instead of waiting for its scheduled time by selecting **Run Schedule Now**. Selecting this option will execute the schedule immediately, as though its next scheduled date were the current time.

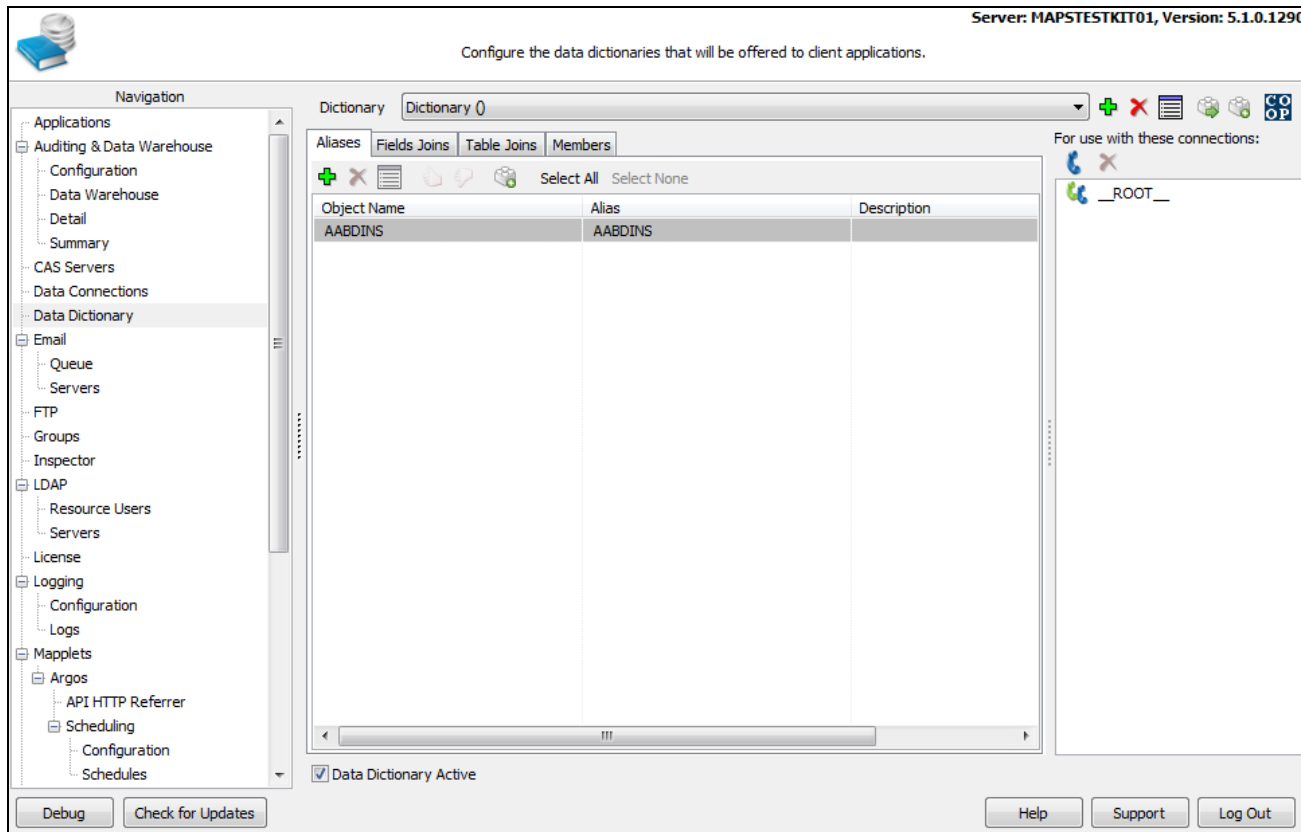
When selecting Run Schedule Now:

- The schedule will execute as your current user regardless of who created the schedule, unless there is a system-level override configured in MAPS.
- Running the schedule now has no effect on the next scheduled date of execution.
- Unlike the Test button in the Edit Schedule dialog, choosing Run Schedule Now does not populate the schedule name variable with "Test of <schedule name>".
- As with any active schedule, if the Email task is configured it's a good idea to make sure that it lists only the desired recipients before running the schedule.

# Data Dictionary

## Introduction

The Data Dictionary is a feature that stores information about your data. Information stored in the Data Dictionary includes table and field aliases and joins. The Data Dictionary is stored in MAPS, but it can be used and edited within Argos using the Argos DataBlock Designer. Evisions has created several Data Dictionaries that are available for administrators to download from the [CO-OP User Community Share](#) at Evisions.com and then import into MAPS.



## Managing Data Dictionaries

Data Dictionaries are created and managed by Administrators using the MAPS Configuration Tool. By selecting Data Dictionary in the left-side Navigation tree, you will gain access to the Data Dictionary configuration screens. You can create as many separate Data Dictionaries as needed. For each Data Dictionary, you can define Aliases, Field Joins, Table Joins, Users, and Associated Connections.

<b>Item</b>	<b>Description</b>
<b>Aliases</b>	<b>Field level descriptive names that enable users to more quickly understand the tables and data they are working with and add consistency to your reports and DataBlocks.</b>
<b>Field Joins</b>	<b>Standard single field joins between two tables.</b>
<b>Table Joins</b>	<b>Custom joins between two tables that are not joined on just two fields. May contain a where clause in the join and is defined using text.</b>

<b>Item</b>	<b>Description</b>
<b>Users</b>	<b>Add users and the rights that they have to this dictionary.</b>
<b>Associated Connections</b>	<b>Data Dictionaries need to be assigned to a connection so that they can be used by the products (Argos) using that connection.</b>

## Importing and Exporting Data Dictionaries

Data Dictionary files can be shared using the import and export process. As stated earlier, Evisions has created several Data Dictionaries that are stored in the CO-OP. Once downloaded, they can be imported into MAPS. You can also share your Data Dictionaries on the CO-OP using the Publish button.

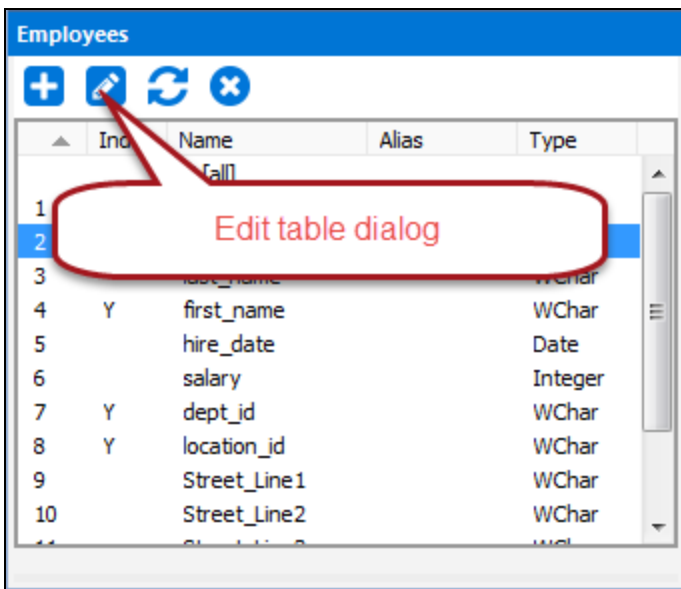
## Using the Data Dictionary in Argos

You will need to make sure "Use Data Dictionary" is enabled. To enable, go to the menu bar on the main Argos screen and click **Tools >> Options** to launch the options dialog. On the **Query Designer** tab, make sure that "Use Data Dictionary" is checked.

In the DataBlock Design window shown below, if Use Dictionary is toggled on you will see Alias and Join information displayed, if it exists in the Data Dictionary. **Note: you need to refresh the tables to view alias information on existing queries.**

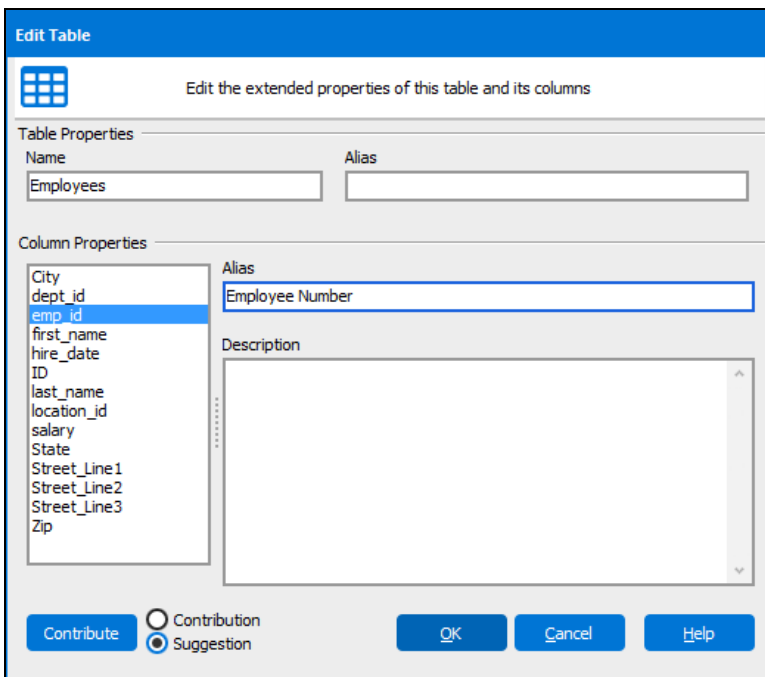
The screenshot shows the DataBlock Designer interface. The toolbar includes a button labeled "Use Dictionary". A red callout bubble with the text "Click here to use the Data Dictionary" points to this button. The main workspace displays two tables: "Employees" and "Orders". The "Employees" table has columns: ID (Integer), emp\_id (WChar), last\_name (WChar), first\_name (WChar), hire\_date (Date), salary (Integer), dept\_id (WChar), location\_id (WChar), Street\_Line1 (WChar), and Street\_Line2 (WChar). The "Orders" table has columns: ID (Integer), sale\_date (Date), employee\_id (WChar), transaction\_id (WChar), invoice\_sent (WChar), and ship\_status (WChar). Below the tables is a query design grid with columns for Table, Field, Type, As, and Description. The grid shows a join between the Employees table (emp\_id) and the Orders table (employee\_id). A green line connects the emp\_id field in the Employees table to the employee\_id field in the Orders table, indicating a join imported from the Data Dictionary. Other blue lines represent explicit joins created by the user.

Note in the figure above that joins imported from the Data Dictionary are shown with green lines connecting the database tables (as opposed to blue lines when joins are explicitly created by the user).



## Adding Aliases to the Dictionary from within Argos

Depending upon the user permissions, suggestions and contributions can be made to the Data Dictionary using the Edit Table dialog box. Simply add aliases and descriptions to the selected field and contribute your entry.



Users can contribute to the Data Dictionary by submitting alias and join information from within the Visual Query Builder by clicking the **Contribute** button in the Edit Table dialog or the Edit Join dialog. Depending upon permissions, information may be submitted as a suggestion which can then be reviewed by an administrator. The administrator can then change the suggestion to a contribution.

Only items in the Data Dictionary with Contributions status will be visible within Argos.

## Adding Joins from within Argos

Adding joins to the Data Dictionary is also straightforward. Click the Add Join button on the toolbar, choose the type of join and then enter the details. The join is a standard single field join between two tables. Don't forget to specify the type of join (inner, left, or right).

**Edit Join**

Edit the properties of this table join

Left Table: Employees Right Table: Orders

Left Field: emp\_id Right Field: employee\_id

Inner Join  
Include all records from 'Employees' and 'Orders' where 'emp\_id' equals 'employee\_id'

Outer Left Join  
Include all records from 'Employees' and only those records from 'Orders' where 'emp\_id' equals 'employee\_id'

Outer Right Join  
Include all records from 'Orders' and only those records from 'Employees' where 'emp\_id' equals 'employee\_id'

Contribute  Contribution  Suggestion OK Cancel Help

If the join is more complex or you prefer to type it yourself, you can use a text join. Select the tables and then enter the necessary text for your join. Both join types can be contributed to the Data Dictionary.

## Alias and Join Info Applied to Query

If a Data Dictionary is in use and aliases are present, when a field is added to the query, the alias and description will be added as well.

Visible Fields (SELECT)	Conditional Fields (WHERE)	Ordering (ORDER BY)	
Distinct			
Summing			
Security			
Table	Employees	Employees	Employees
Field	emp_id	last_name	first_name
Type	string	string	string
As	Employee Number	Employee last name	Employee first name
Description	Employee Number	Last Name	First Name

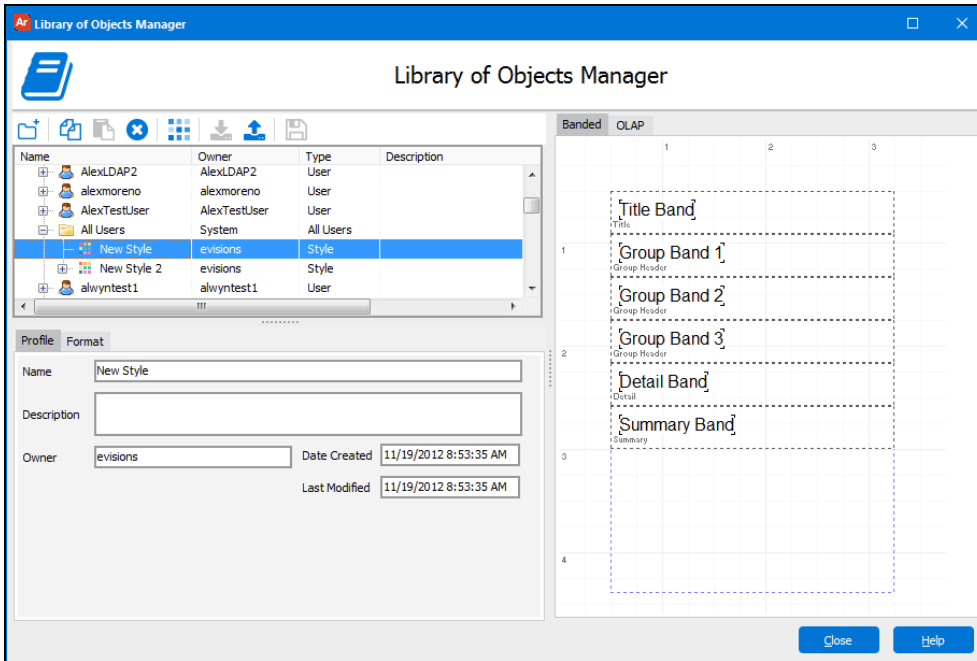
```
SELECT Employees.emp_id AS Employee Number,  
Employees.last_name AS Employee last name,  
Employees.first_name AS Employee first name,  
Orders.sale_date  
FROM Employees inner join Orders on Employees.emp_id = Orders.employee_id
```



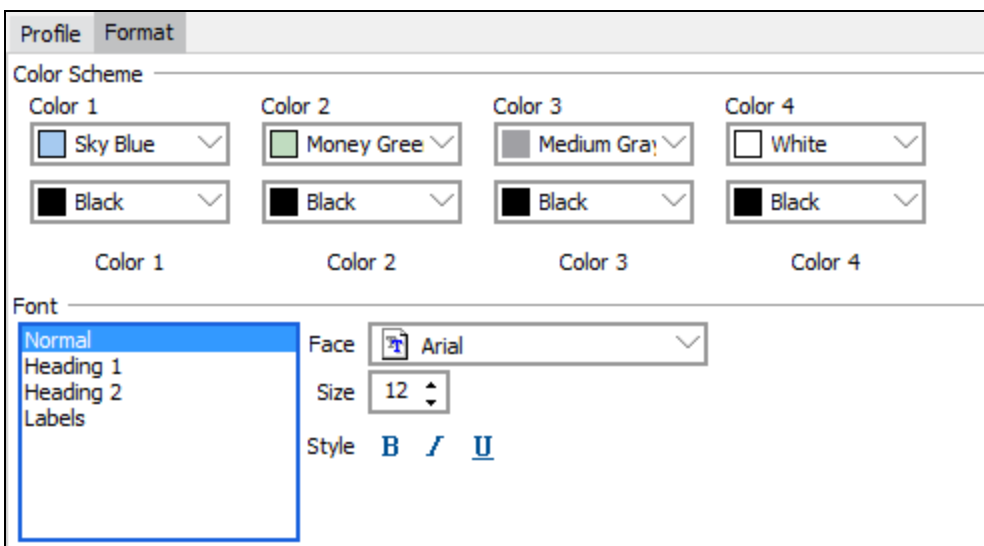
# Library of Objects

The Library of Objects (the Library) is made up of a set of features based on the concept of saving and reusing objects throughout Argos. The Library interacts with various parts of Argos in different ways, but sets the foundation for sharing objects throughout the tool. The main interface for the Library is the Library of Objects Manager (the Library Manager). The Library Manager can be accessed from the Tools menu in Argos.

## Library Manager



The Library Manager has two main functions. First, it is used to manage and organize all of the available objects. A folder structure can be created to organize objects. Second, the Library Manager is where Styles are created and edited. Once a Style is created, use the Format tab to set all of the color and font properties.



A single Style can be applied to both Banded Reports and OLAP Cubes.

## Types of Objects in the Library

The Library can contain many types of objects from the various parts of Argos, including:

<b>Object Type</b>	<b>Used In</b>	<b>Description</b>
<b>Style</b>	<b>Banded Reports, OLAP Cubes</b>	<b>Sets color and font options</b>
<b>Report Template</b>	<b>Banded Reports</b>	<b>An entire report definition</b>
<b>Report Band</b>	<b>Banded Reports</b>	<b>An entire band from a report and all of its contents</b>
<b>Image</b>	<b>Banded Reports, Form Design</b>	<b>A single image</b>
<b>Text Object</b>	<b>Banded Reports, Form Design</b>	<b>A single text object</b>
<b>Expression</b>	<b>Banded Reports, Form Design</b>	<b>A calculated field</b>
<b>Form Template</b>	<b>Form Design</b>	<b>A group of objects on a form that are saved together as a unit</b>
<b>Form Controls (list box, drop down box, etc.)</b>	<b>Form Design</b>	<b>One of various controls used on a form. Each can be saved as an individual object in the Library.</b>
<b>Chart</b>	<b>Banded Reports, Form Design</b>	<b>A chart on a form or report</b>
<b>OLAP Cube</b>	<b>Banded Reports, Form Design</b>	<b>An OLAP Cube on a form or report</b>


## Using the Library


Objects can be added to the Library from both the Banded Report Editor and the Form Design window. Select the items you want to add, click the icon on the toolbar and complete the Add Object form. You can select multiple objects at once. To create a Report Template, do not select any objects from the Banded Report before adding to the Library. If no objects are selected, the Library knows you are saving a Report Template.

If you are building a form or a report and you would like to add objects from the Library, click the icon to get an object and select the object you would like to add. You will need to know the name and folder of the object you want to add.

If you choose to use a Report Template from the Library on an existing report, it will overwrite the entire report with the template.

Styles can also be added to a report. The Style will overwrite the existing colors and fonts. However, you must check the band properties to ensure that the background color option is set to Use Color if you want the new background style color to show on the final version of the report.

<b>Icon</b>	<b>Description</b>
	<b>Add to Library</b>

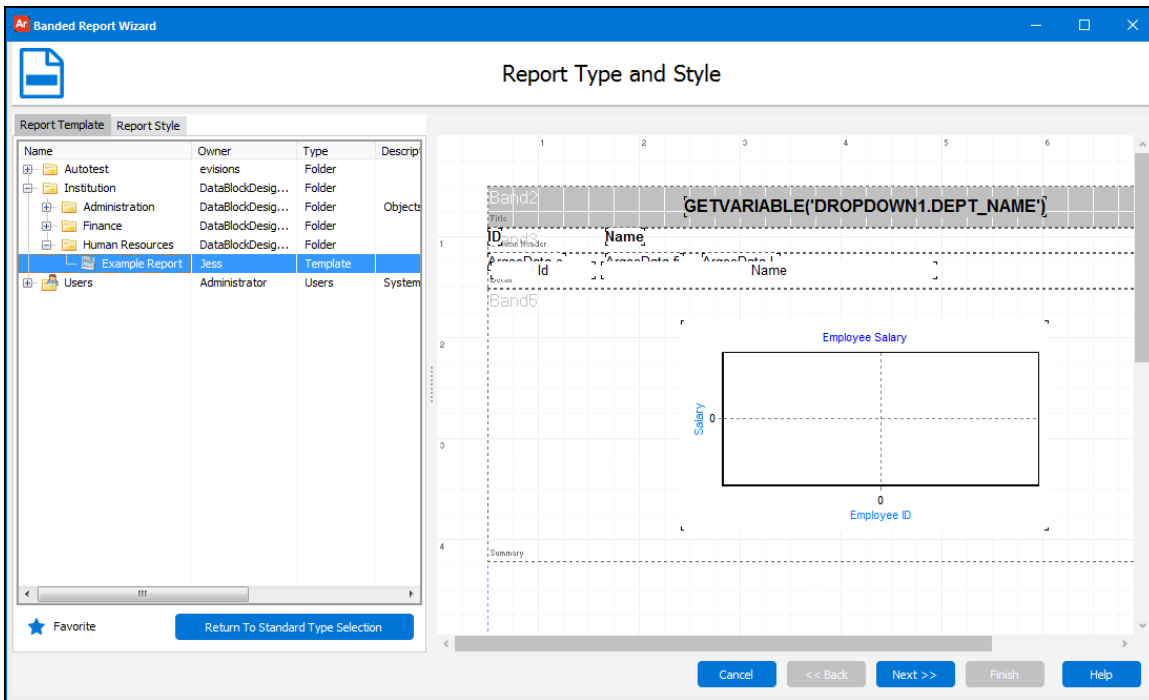
Icon	Description
	Add from Library
	Add a Style

## Adding Styles to Banded Reports

The Banded Report Wizard includes the ability to base a report on a saved report template which is stored in the Library of Objects. You can also add a custom saved style to the report. Styles set the colors and fonts within the bands of the report.

### Report Templates

The Library of Objects can contain report templates and other objects, such as images, you would like to add to the report. In the first step of the report wizard, you can select Add from Library which opens the Report Template tab. Here you can select an existing report template to use for this new report.



### Styles

A style in Argos refers to a saved set of colors and fonts that can be applied to a report or an OLAP cube. Styles are also stored in the Library of Objects however they are handled separately from other objects because they are different. To create a Style you need to open the Library of Objects Manager from the Tools menu on the main Argos screen.

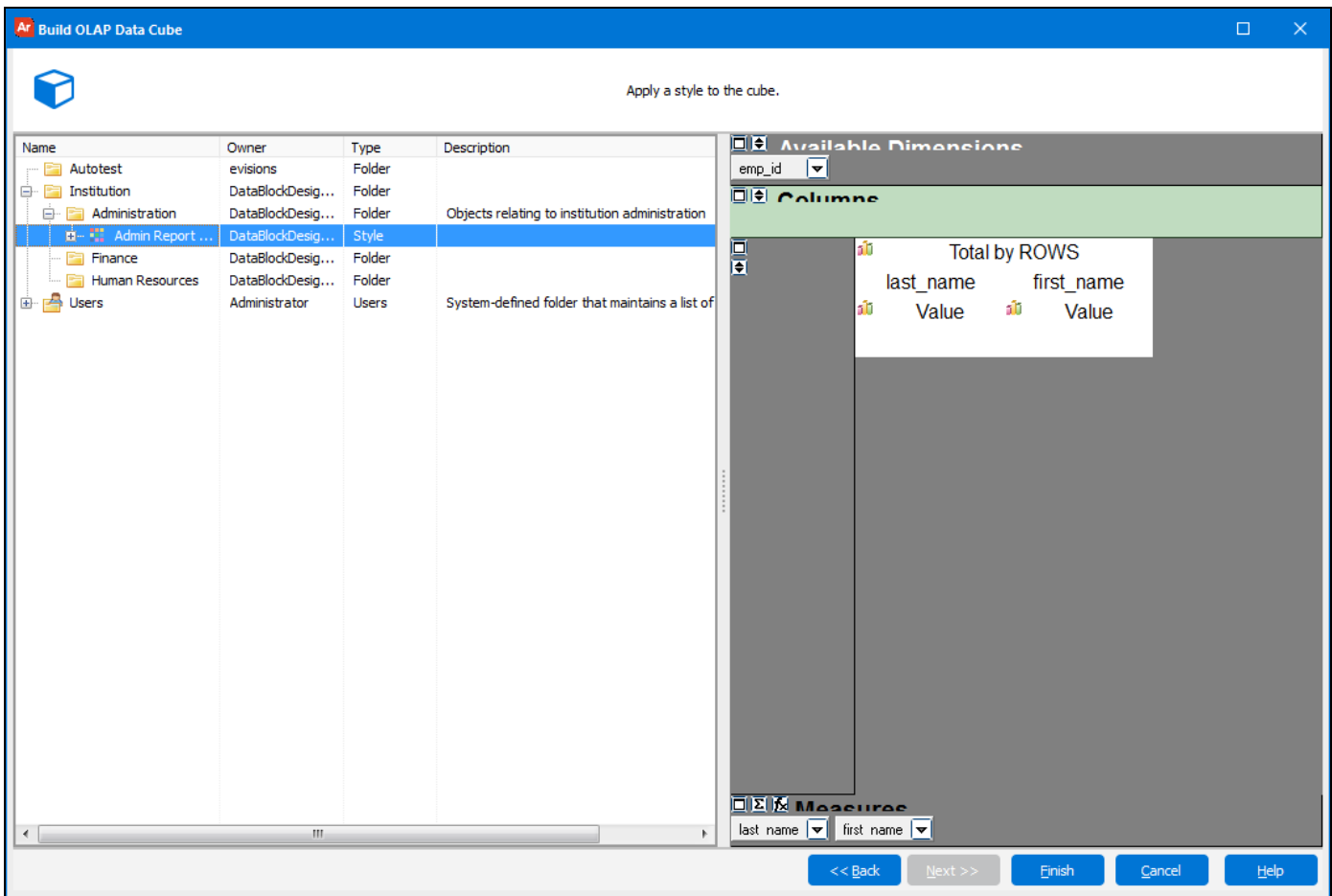
From within the Banded Report Wizard you can select a Style to use with your report. Styles are optional and can overwrite some or all style settings in the Banded Editor.

## Report Preview

The Banded Report Wizard contains a sample preview so you can see the impact of your changes before accepting them.

## Adding Styles to OLAP Data Cubes

Styles can also be used with an OLAP Cube. A step has been added to the wizard process of creating an OLAP Cube for you to select a Style. Also, a preview of the cube has been added to all the steps within the OLAP Data Cube Wizard.




# Security

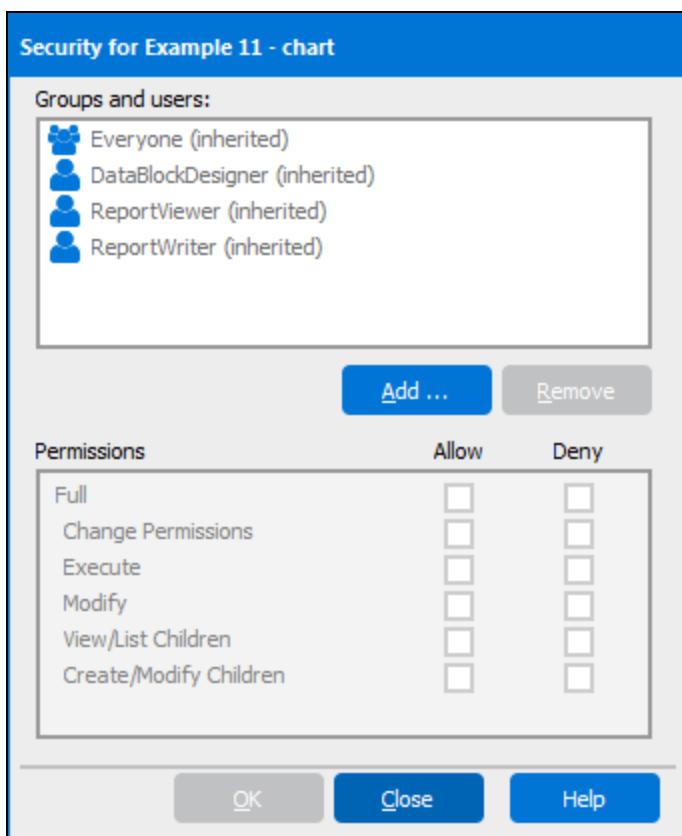
## Object Level Security

Argos DataBlock Designers and Administrators can configure security for all objects (Folders, DataBlocks, and Reports) in the Explorer Tree.

A dialog box is used to define security for each object and is accessed in two ways:

- Highlight the object in the Explorer tree and click the Security icon  on the Argos toolbar.
- Right click the object then select **Security**

Both methods will bring up the Security dialog box shown below. Allow or Deny permission can be granted to any number of groups and users. The figure below shows the security dialog for the “Finance Report” report.



### Adding Security

Select a user or group from the list to configure permissions, or click the **Add. . .** button to add a new user or group to the list. You can Allow or Deny the permissions listed below:

<b>Permission</b>	<b>Description</b>
<b>Full</b>	<b>Perform any action, including modifying permissions for any user, including themselves. Checking this option has the same effect as checking all the other options individually.</b>

<b>Permission</b>	<b>Description</b>
<b>Change Permissions</b>	<i>Modify permissions for any user, including themselves. Note that this permission is the same in some respects as Full, since the user could easily enable Full if granted Change Permissions.</i>
<b>Execute</b>	<i>Run the selected object.</i>
<b>Modify</b>	<i>Modify the selected object.</i>
<b>View/List Children</b>	<i>View the object and the children of the object. Typically this permission would be placed on a folder or DataBlock to enable children of that object to be viewed. If you cannot determine why a given user or group is able to see a given object, examine the parent of the object in question for this permission.</i>
<b>Create/Modify Children</b>	<i>Create or Modify child objects of the selected object. Typically this permission would be placed on a folder of DataBlock to enable children to be created. If you cannot determine why a given user or group is able to Create or Modify certain objects, examine the parent of the object in question for this permission.</i>

### Removing Security

To remove security for a user or group, select the user or group that needs to be removed and click the **Remove** button. Note that they may still have permissions due to membership in other groups (see notes below).

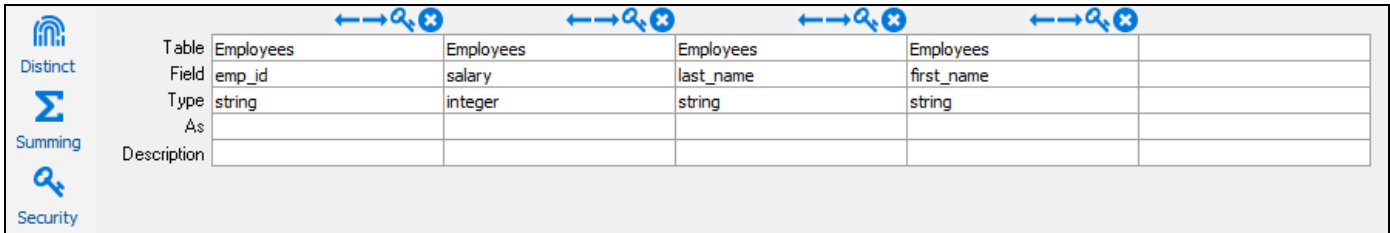
### Group Security

<b>Type</b>	<b>Description</b>
<b>Direct Security</b>	<i>Security explicitly defined on an individual user supersedes any permissions they may have due to membership in one or more groups. For example, if a user is Denied permission to Modify a given object, they will not be able to do so even if one or more of their groups has this permission Allowed.</i>
<b>Inherited Security</b>	<i>Note that objects in Argos automatically 'inherit' the permissions of their parent. For example, if a folder has a given set of permissions, all objects in that folder will have the same permissions unless explicitly overridden. Furthermore, this inheritance extends to the objects' 'grandchildren', 'great grandchildren', etc.</i>
<b>Accumulated Group Permissions</b>	<i>When a user attempts to access an Argos object, Argos will examine the permissions of all groups that they are a member of. Argos grants the user all accumulated positive group permissions to the object (unless that user has been directly Denied permissions - see above.) For example, if a user has been granted direct permission to Execute an object, and their membership in a group granted them permission to Modify the object, their accumulated positive permissions would be Execute and Modify. Even if the user was in a second group that was denied the ability to Modify the object, they would still be able to Modify the object due to their membership in the first group.</i>
<b>Everyone Group</b>	<i>Pay special attention to permissions granted to the Everyone group as all users are automatically members of this group. It is often a good policy to Deny permissions to the Everyone group once other groups have been established.</i>

## Field Level Security


The above described how Security can be applied to entire entities such as Folders, DataBlocks, and Reports. However there are times when it is desirable to allow someone to execute a report, but limit the fields they can see.

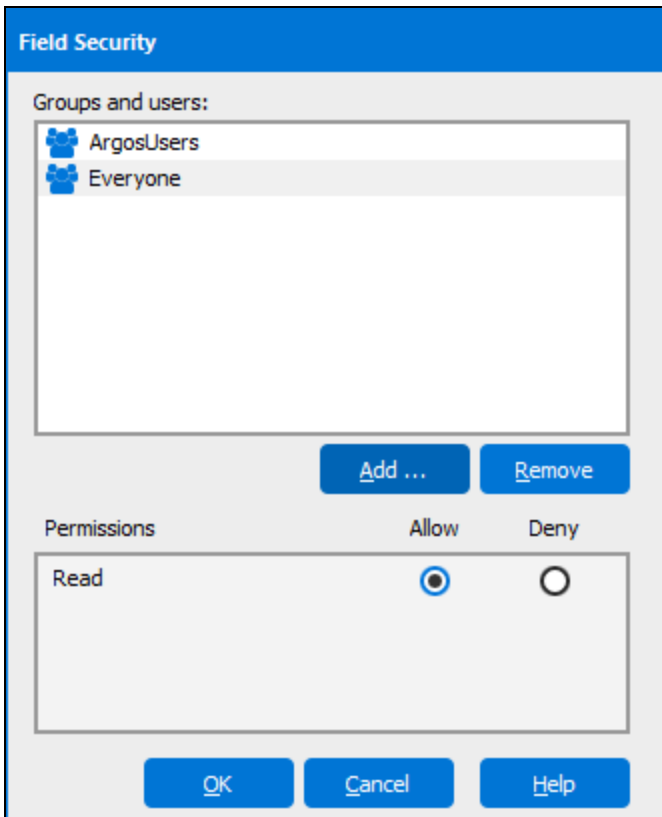
This is accomplished within the **Report Query - Visual Design** tab shown in the figure below. The key shaped icons are used to apply security to individual fields.



The screenshot shows a table with four columns, each representing a field from the 'Employees' table. Above each column is a key icon with a double-headed arrow, indicating that field-level security is applied. On the left side of the table, there are icons for 'Distinct', 'Summing', and 'Security'. The table structure is as follows:

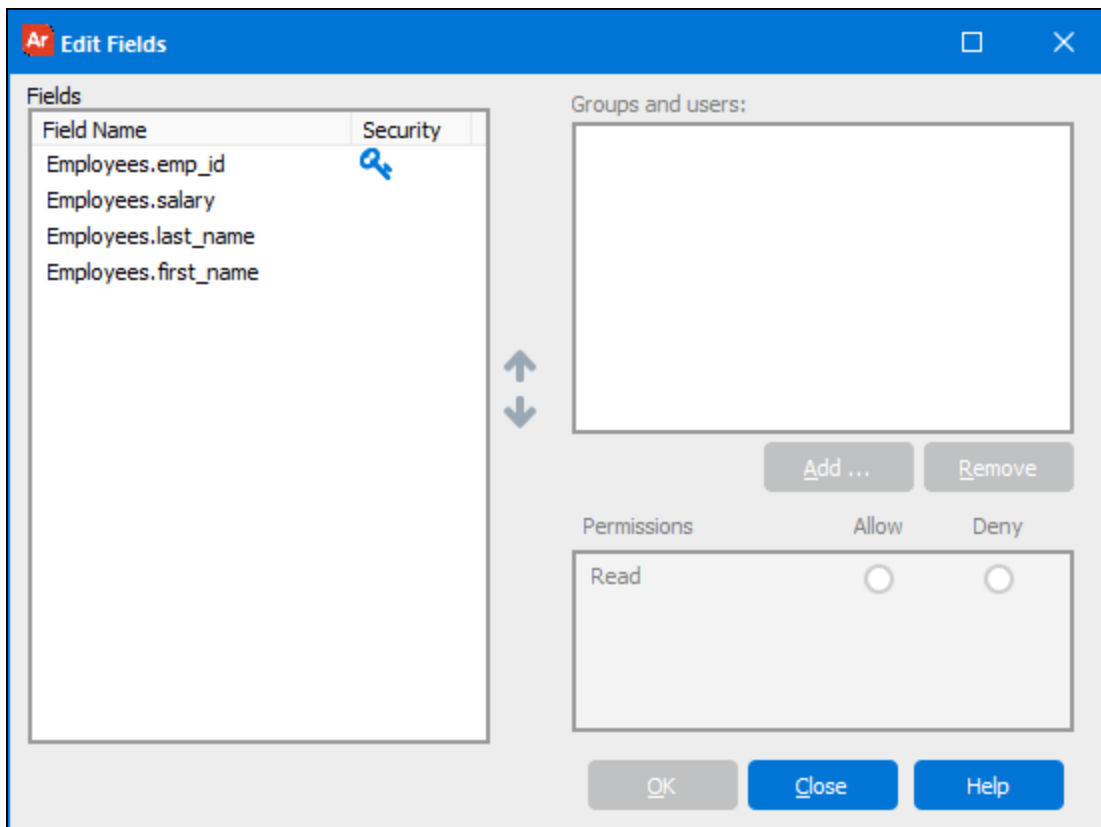
Table	Employees	Employees	Employees	Employees
Field	emp_id	salary	last_name	first_name
Type	string	integer	string	string
As				
Description				

To limit access to a particular field, click on the **key** icon  above the field. The Field Security dialog box shown below will be displayed. Click the Add button to display the list of Argos Groups and Users. Select the user or group then pick the Allow or Deny radio button. Click the Add button multiple times to add security for additional Groups and Users. The figure below shows two groups that are given Read privilege to the selected field.



The Field Security dialog box is titled 'Field Security'. It contains a list of 'Groups and users' with 'ArgosUsers' and 'Everyone' selected. Below the list are 'Add ...' and 'Remove' buttons. Under the 'Permissions' section, the 'Read' permission is set to 'Allow' (indicated by a selected radio button). At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Security can also be added by clicking the set of keys to the left of the visible field list. Using this method allows the user to review what field level security has been set for all fields in the DataBlock. If security has already been set for the field, the key icon will be displayed next to the field name as shown in the figure below. You can also add, remove, or modify security on any field by clicking the field name, then clicking the Add or Remove button.

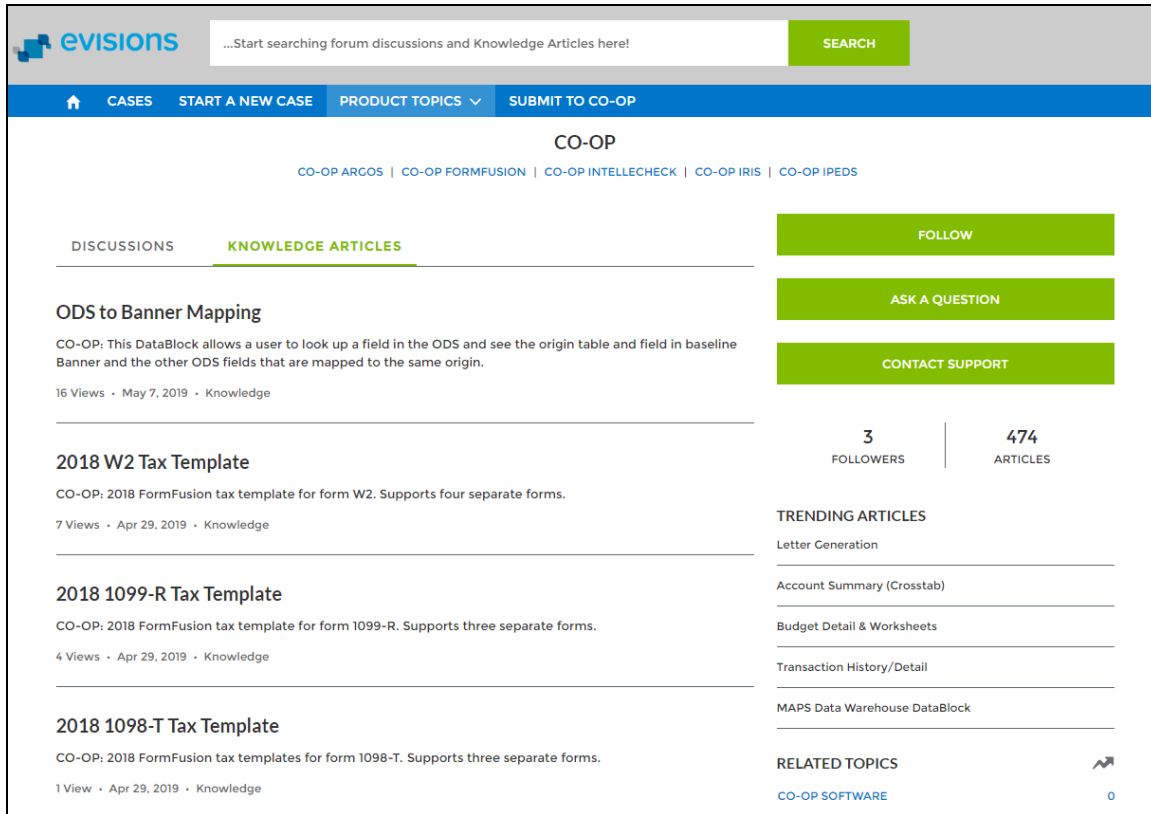




# The CO-OP User Community

The Evisions [CO-OP](#) provides a place for users to collaborate by sharing Argos DataBlocks, FormFusion templates, Data Dictionaries, and other objects. It contains items uploaded by Evisions as well as by clients like yourself. The shared DataBlocks and templates make a great starting point for your own development, since much of the work is already done for you.

To access the CO-OP, navigate to it from the [Evisions Support website](#), or click the  button in the toolbar.



The screenshot shows the Evisions CO-OP user community interface. At the top, there is a search bar with the text "...Start searching forum discussions and Knowledge Articles here!" and a green "SEARCH" button. Below the search bar is a navigation menu with options: "CASES", "START A NEW CASE", "PRODUCT TOPICS", and "SUBMIT TO CO-OP". The main content area is titled "CO-OP" and includes links for "CO-OP ARGOS", "CO-OP FORMFUSION", "CO-OP INTELLECHECK", "CO-OP IRIS", and "CO-OP IPEDS". There are three tabs: "DISCUSSIONS", "KNOWLEDGE ARTICLES" (which is selected), and "FOLLOW". Below the tabs, there are three knowledge articles listed:

- ODS to Banner Mapping**  
CO-OP: This DataBlock allows a user to look up a field in the ODS and see the origin table and field in baseline Banner and the other ODS fields that are mapped to the same origin.  
16 Views · May 7, 2019 · Knowledge
- 2018 W2 Tax Template**  
CO-OP: 2018 FormFusion tax template for form W2. Supports four separate forms.  
7 Views · Apr 29, 2019 · Knowledge
- 2018 1099-R Tax Template**  
CO-OP: 2018 FormFusion tax template for form 1099-R. Supports three separate forms.  
4 Views · Apr 29, 2019 · Knowledge
- 2018 1098-T Tax Template**  
CO-OP: 2018 FormFusion tax templates for form 1098-T. Supports three separate forms.  
1 View · Apr 29, 2019 · Knowledge

On the right side of the interface, there are three green buttons: "FOLLOW", "ASK A QUESTION", and "CONTACT SUPPORT". Below these buttons, there are statistics: "3 FOLLOWERS" and "474 ARTICLES". There is also a "TRENDING ARTICLES" section with a list of articles: "Letter Generation", "Account Summary (Crosstab)", "Budget Detail & Worksheets", "Transaction History/Detail", and "MAPS Data Warehouse DataBlock". At the bottom right, there is a "RELATED TOPICS" section with a link to "CO-OP SOFTWARE" and a small icon of a person.

The file uploads are under the **Knowledge Articles** tab at the top.

# Searching for DataBlocks and Templates

To search the CO-OP for Argos DataBlocks, select **CO-OP ARGOS** in the list of categories at the top. Similarly, if you want to view FormFusion templates, select **CO-OP FORMFUSION**. If you are looking for DataBlocks/templates specific to Banner, Colleague, etc, you can further filter your results by selecting only that subcategory.

### CO-OP ARGOS

[CO-OP ARGOS BANNER](#) | [CO-OP ARGOS COLLEAGUE](#) | [CO-OP ARGOS POWERCAMPUS](#) | [CO-OP ARGOS ADVANCE](#) | [CO-OP ARGOS JENZABAR CX](#) | [1 More...](#)

---

[DISCUSSIONS](#)   **[KNOWLEDGE ARTICLES](#)**

---

#### ODS to Banner Mapping

CO-OP: This DataBlock allows a user to look up a field in the ODS and see the origin table and field in baseline Banner and the other ODS fields that are mapped to the same origin.

16 Views · May 7, 2019 · Knowledge

---

#### Examine Oracle Rollback Segments

CO-OP: This datablock examines Oracle Rollback Segments

35 Views · Apr 25, 2019 · Knowledge

---

#### Examine Oracle Memory

**FOLLOW**

**ASK A QUESTION**

**CONTACT SUPPORT**

---

**2** FOLLOWERS | **456** ARTICLES

**TRENDING ARTICLES**

Letter Generation

---

Account Summary (Crosstab)

Click on the name of any file to go to its detail page.

[HOME](#)   [CASES](#)   [START A NEW CASE](#)   [PRODUCT TOPICS](#)   [SUBMIT TO CO-OP](#)

---

**ARGOS**

### Graduation Completion

CO-OP : This Advancement DataBlock contains Graduation Completion information on graduation trends and to monitor and improve graduation rates.

Apr 9, 2019 · Knowledge

**PRODUCT**  
Argos

---

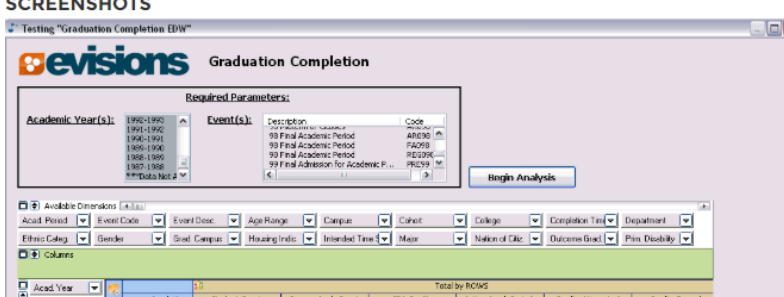
**PRODUCT VERSION**

---

**WHAT'S NEW**

---

**SCREENSHOTS**



**FOLLOW**

**Files (1)**

- Graduation Completion EDW 20160...  
Apr 9, 2019 · 100KB · argosexport

[View All](#)

**RELATED ARTICLES**

- Argos Samples 9
- Test 34
- Student Directory 14
- Addresses by Major Year 2
- Graduates 1

**TRENDING ARTICLES**

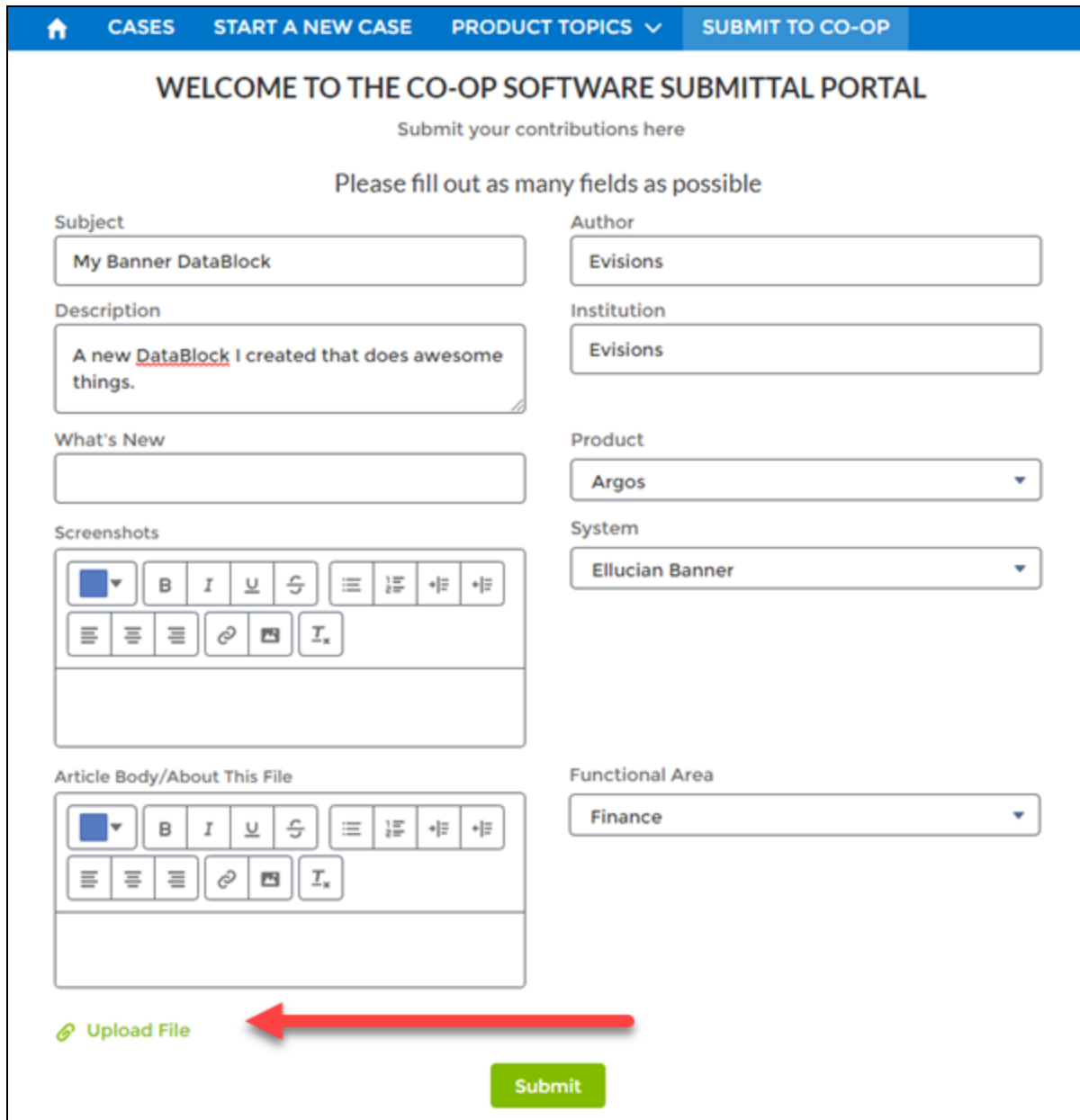
- MAPS Setup
- MAPS Installation Guide

From here, you can choose to download the document by clicking on the filename. You can then import the selected objects into the product and edit them as needed.

## Submitting Files

To upload a DataBlock, template, or other object to the CO-OP, first export it to a file. You may wish to make a copy first so that you can clean up any notes or sensitive information that may be associated with the object.

In the CO-OP, select **SUBMIT TO CO-OP** in the menu. Fill out the required information such as the subject, product, system, your name and institution, and other information. For example, an Argos DataBlock developed for Ellucian Banner that pertains to Finance.



The screenshot shows the 'SUBMIT TO CO-OP' page in a web application. The navigation bar at the top includes 'CASES', 'START A NEW CASE', 'PRODUCT TOPICS', and 'SUBMIT TO CO-OP'. The main heading is 'WELCOME TO THE CO-OP SOFTWARE SUBMITTAL PORTAL' with the subtext 'Submit your contributions here'. Below this, it says 'Please fill out as many fields as possible'. The form is divided into two columns. The left column contains: 'Subject' (text input: 'My Banner DataBlock'), 'Description' (text area: 'A new DataBlock I created that does awesome things.'), 'What's New' (empty text input), 'Screenshots' (rich text editor with icons for bold, italic, underline, link, unlink, and image), and 'Article Body/About This File' (another rich text editor). The right column contains: 'Author' (text input: 'Evisions'), 'Institution' (text input: 'Evisions'), 'Product' (dropdown menu: 'Argos'), 'System' (dropdown menu: 'Ellucian Banner'), and 'Functional Area' (dropdown menu: 'Finance'). At the bottom left, there is a green 'Upload File' link with a red arrow pointing to it. At the bottom center, there is a green 'Submit' button.

Use the Upload File link to attach the file(s) you want to upload. When finished, click the **Submit** button. The files will appear publicly on the CO-OP once they have been approved by a moderator.

# Argos Support Resources

---

Evisions provides several sources of support for Argos users, including in-product Help and user guides; live and recorded training; and a support website with many useful resources.

## In-Product Help

---

The [Argos Help](#) serves as a reference guide for every feature in Argos. You can access Help by pressing the **F1** key, clicking the **Help** button in a dialog box, or by selecting **Help** from the Argos toolbar. The Help contains a table of contents, an index, and a search feature.

The [Argos Web Viewer Help](#) is separate from the main Argos Help, and is accessible from the Help menu in the Web Viewer. There is also a section in the main Argos help pertaining to Web Viewer settings in the Argos client.

## User Guides

---

The Argos Help includes three user guides. Each guide contains step-by-step instructions for common and advanced tasks performed by that user type, and provides an excellent mechanism for getting started with Argos.

- [Argos Report Viewer Guide](#)
- [Argos Report Writer Guide](#)
- [Argos DataBlock Designer Guide](#)

## Training

---

Evisions provides free, [live online training](#) for all products. Visit the Training page to register for a session, or watch [recorded training sessions](#) at your convenience.

## Additional Resources

---

The [Support](#) page contains links to additional support resources:

- [Argos Installation](#) - Installation and upgrade information.
- [Release Documentation](#) - Release guides and release notes for all current and previous versions of Argos.
- [Knowledge Base](#) - Answers to common questions.
- [HelpDesk](#) - Technical support.
- [CO-OP Share](#) - Pre-built DataBlocks, data dictionaries, and other resources.
- [Forums](#) - Collaborate with other Argos users.

# Getting Help

---

For information on using the software, please refer to the in-product Help, which contains detailed information on all aspects of the product.

If you are having problems with the installation or configuration, you can search our [support site](#), which includes a knowledge base of common issues. If you are unable to find the solution, submit a HelpDesk request with a detailed explanation of the problem you are experiencing.

Please do not hesitate to contact the Evisions HelpDesk if any questions or problems arise. We are here to help you and want to ensure your success.

If you find that areas of this documentation could benefit from additional detail or clarification, please let us know. We are constantly trying to improve the installation process to make it as easy as possible.