evisions

argos

Enterprise Reporting Solution

DataBlock Designer Guide

Document version 1.0

Last Updated 8/12/2013

Trademark, Publishing Statement, and Copyright Notice

© 2013 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

Argos Introduction **8**

Evisions Support Site 8

In Product Help 8

Sample Database 9

Getting Started **10**

Starting Argos 10

Connect to the Server 10

Start Argos 10

Change Password 10

Navigation **11**

Action Area 11

Navigation Area 11

Explorer view 11

Shortcuts View 12

Definitions **13**

DataBlock 13

OLAP Data Cube 15

Charting 15

The DataBlock Designer **16**

Object Toolbar 16

The Alignment Toolbar 19

Alignment of a single object 19

Alignment of multiple objects 19

Building a Form **20**

Introduction 20

Add DataBlock icon to Explorer 20

Select a database connection 20

Add a Description, Author, and Notes 21

Launch the Argos DataBlock Designer 21

Add the Graphics Object 22

Image stored in the DataBlock 22

Image stored on an accessible server 22

Image retrieved from a variable 22

Adjust Object Properties 23

Add the Date Objects and Labels 23

Align the objects 25

Add Employee Name Label and Shape Object 25

Create the object for selecting Employee Name 25

Building a Query **30**
Form Query 30
Create the multi-column list box to contain the query results 30
Identify the fields to appear in the report 32
Create a calculated field to determine sale amount 33
Use input selections to limit the query 34
Use of button to control the execution of the query 37
Summary 38
Report Query 39
Introduction 39
Create a New Report 39
Copy/Paste the Existing Query 40

Advanced Form Objects and Properties **41**
Data Aware Property 41
Introduction 41
Add the Data Aware Object 41
Set Data Aware property for edit box 42
Execute the Report 42
Summary 42
Using Multiple Forms in a DataBlock 43
Introduction 43
Create the DataBlock 44
Add the Form Names 44
Create the Main Form 44
Edit Properties of the Buttons 45
Create the form/query associated with each button 46
Summary 47
Using tabs instead of buttons to navigate between forms 47
Charting with multiple series 48
Introduction 48
Create the Chart Object 48
Name the first series 49
Create the new dataset for the first series 49
Select the Chart Types, labels, and other options 50
Create the next series for fy2007 51
Summary 52
Creating a Filtered OLAP Data Cube 52
Introduction 52
Create the OLAP Object in the DataBlock Designer 52
Build the Fact Table (the query) 54
Identify the Measures 55
Identify the Dimensions 55
Commit and Test 56
Summary 57
Drilling Through Reports to Obtain Greater Detail 57
Summary 59

Advanced Query Techniques **60**
Summing and Grouping 60
Introduction 60
Create the Form 60
Select GROUP BY and SUM fields 61
Select HAVING Tab 62
Summary 62
Free Type Report Query 63
Introduction 63
Entering the Free Type Query 63
Validate the Query 64
Apply Field Security 64
Filter and Sort the Query in CSV, Banded, and Extract Reports 65
Adding special characters for filtering in CSV, Banded, and Extract Reports 66
Adding special characters for Sorting in CSV,Banded, and Extract Reports 67
Edit ADO Properties 68
Summary 68
Scalar Sub Query 68
Introduction 68
Create the Main Query 69
Create the Sub Query 69
Execute the query 70
Summary 70
Correlated Sub Query 71
Introduction 71
Create the Main Query 71
Create the Sub Query 72
Place the Sub Query into the Main Query 75
Execute the query 76
Non-Correlated Sub Query 77
Introduction 77
Create the Main Query 77
Create the Sub Query 78
Place the Sub Query into the Main Query 78
Execute the query 79
Unions 80
Introduction 80
Create the Main Query 80
Create the Unions 80
Summary 82

Report Scheduling **83**
Introduction 83
The General Tab 83
The Schedule Tab 84
The Tasks Tab 84
The Events Tab 86

The API Tab 86
Execute the Report Task 87
The Process and Save Task 88
Copy File Task 89
Using an Argos variable in the Destination File Name 89
Using dates in the Destination File Name 91
Use of Source File Name 91
The Saved States Task 92
Executing the Saved State 92
Managing Saved States 93
The Bursting Task 93
Managing the Schedule 95

Data Dictionary 96

Introduction 96
Managing Data Dictionaries 96
Importing and Exporting Data Dictionaries 97
Using the Data Dictionary in Argos 97
Adding Aliases to the Dictionary from within Argos 97
Adding Joins from within Argos 98
Alias and Join Info Applied to Query 99

Library of Objects 100

Library Manager 100
Types of Objects in the Library 101
Using the Library 101
Adding Styles to Banded Reports 102
Report Templates 102
Styles 102
Report Preview 102
Adding Styles to OLAP Data Cubes 103

Security 104

Object Level Security 104
Adding Security 104
Removing Security 105
Group Security 105
Field Level Security 106

The CO-OP User Community Share 108

Publishing to the CO-OP 108
Complete the Upload Form 108
Keywords 108
CO-OP Upload Form 109

Argos Resources **110**
Support Site 110
CO-OP User Community 110
Important Links 111

Getting Help **112**

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.

Argos

Evisions' solution for building and deploying reports and dashboards across the enterprise.

DataBlock

DataBlocks are the foundation of Argos. They contain user input forms and queries to retrieve information from one or more data sources. Reports in Argos have a DataBlock as their "parent" and each DataBlock can contain multiple reports.

Explorer Tree

The Argos Explorer Tree is a way to view and navigate the folders, DataBlocks, and reports in Argos.

Support icon on Argos toolbar

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

http://www.evisions.com/Services/Overview.aspx.

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 database from the Evisions Web site at

http://www.evisions.com/Default.aspx?tabid=67&id=745.

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. A description of the database contents (.pdf file) can be downloaded from the Evisions web site and is named "Argos User Guides Sample Database Description" and is located at http://www.evisions.com/Default.aspx?tabid=67&id=746.

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

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

Getting Started

Starting Argos

Argos is Windows PC software which is web-enabled, meaning it is accessible from your web browser via an Internet connection. Before launching Argos, you may need to disable any pop-up blockers running on your computer. To disable the pop-up blocker in Microsoft (MS) Internet Explorer, select Tools, Pop-up Blocker, Turn off Pop-up Blocker. You should be able to re-enable the pop-up blocker once you have downloaded the software.

Connect to the Server

Type the web address provided by your system administrator into the address bar of your browser to access the Multiple Application Platform Server (MAPS) launch page. This webpage provides a central access point for all MAPS applications, including Argos, FormFusion, IntelleCheck, Datamasque and the MAP Server Configuration Tool.

Start Argos

Click **Argos** from the list of Evisions applications.

Click the **Argos** button to launch Argos.

Enter your user name and password (obtained from your system administrator) in the Login box. The **Remember this user** and **Remember the password for this user** are check boxes (available as determined by the MAPS Administrator) that are optional and should not be used on shared computers. Check them as desired.

Click the Login button.

Change Password

To change your password, select Tools from the menu at the top of the screen and then select Change Password. You must know your current password in order to change it. The password strength indicator will help you determine if your password is secure enough. It is advisable to contact your system administrator before changing your password to verify that the change will not produce any undesirable results.

Browser support

Argos is designed to operate using MS Internet Explorer, Mozilla Firefox, Google Chrome, and Apple Safari browsers.

Java

Launching applications is done via Java. If Java is not installed on your PC, you will be prompted to install it.

Multiple Application Platform Server

MAPS is the server that delivers the Argos software to users. Once Argos is installed, users connect to MAPS which fetches data and performs other tasks for them.

FormFusion

Evisions' solution for enhancing documents and managing distribution via email, imaging software, print, etc. Users can rearrange and add data and images, change the layout, format fonts, etc.

IntelleCheck

Evisions solution for payment processing – Accounts Payable, payroll and refund checks, Direct Deposit Advices and electronic refunds.

DataMasque

Evisions solution for removing Personally Identifiable Information from any SQL-compliant database.

Password Strategies

The following strategies will help you make a more secure password.

- Increase password length.
- Include letters and numbers.
- Use both upper and lower case.
- Use special characters [,{!?, etc.

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:

Folders – Contains objects, including other folders

DataBlocks – The "parent" object for one or more reports

Dashboards - Display-only reports for quick reference.

CSV Report – A comma-separated values report

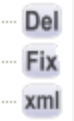
Banded Report – A fully-formatted report

Extract Report - A text report that meets pre-defined specifications

Schedule – Reports may be scheduled to run automatically

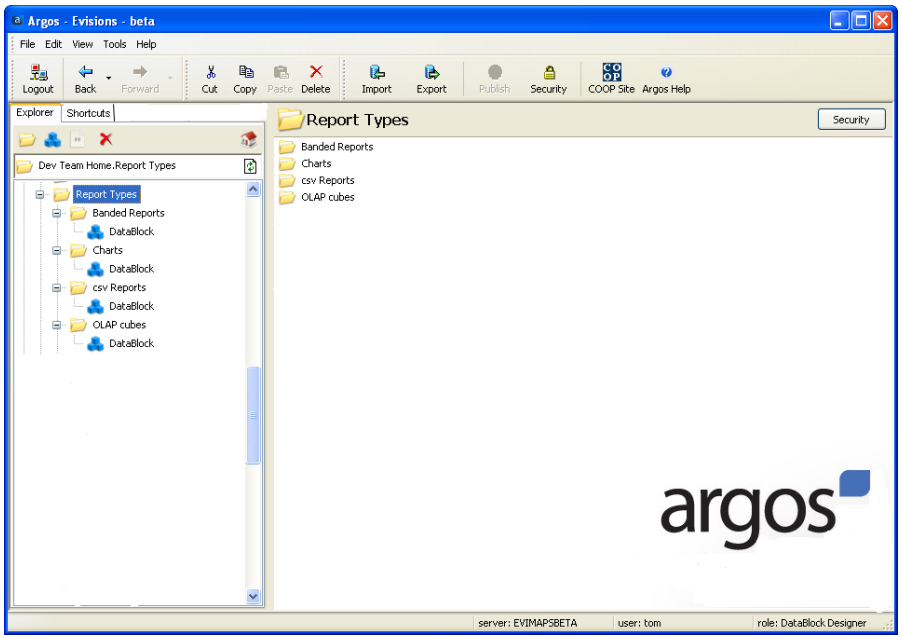
Some objects can be flagged as private. Private objects will not show up in the Explorer for users other than the creator and the administrator. These objects will have the "private eye" icon like the sample private Banded report icon on the list to the right.

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.

Icon	Explorer Objects
	Folder
	DataBlock
	CSV Report
	Banded Report
	Extract Reports (Delimited, Fixed Width, XML)
	Private Report (Banded)
	System created Dashboard
	User created Dashboard
	Schedule

Explorer objects

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

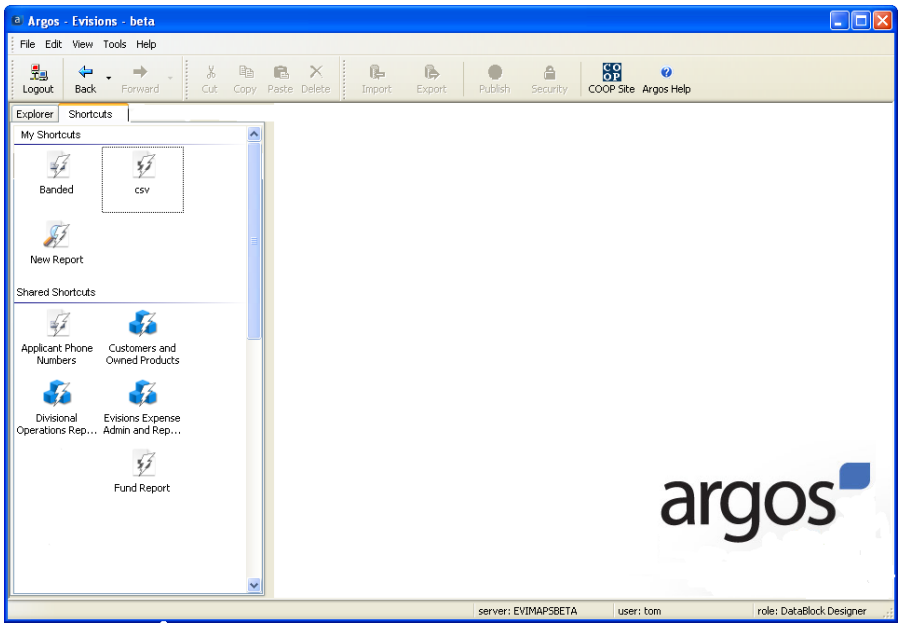


Explorer Navigation

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.

Shortcuts View

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.



Are my reports secure?

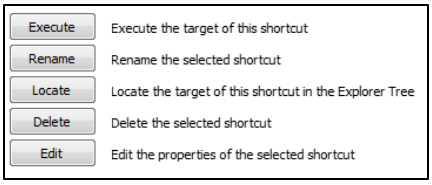
You may be wondering about the security of reports in Argos. Each object can be assigned to groups or individuals as needed. Unauthorized users would not be able to see objects they lack permissions for. It is even possible to have security all the way down to individual fields in a CSV or Banded report.

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.

Shortcut

A Shortcut is a reference to a DataBlock or Report. It can be shared (so all users can see it) or private (My Shortcuts) so only the creator can see it. Click the Shortcuts tab to switch to the Shortcuts view.



Options for Shortcuts

Definitions

DataBlock

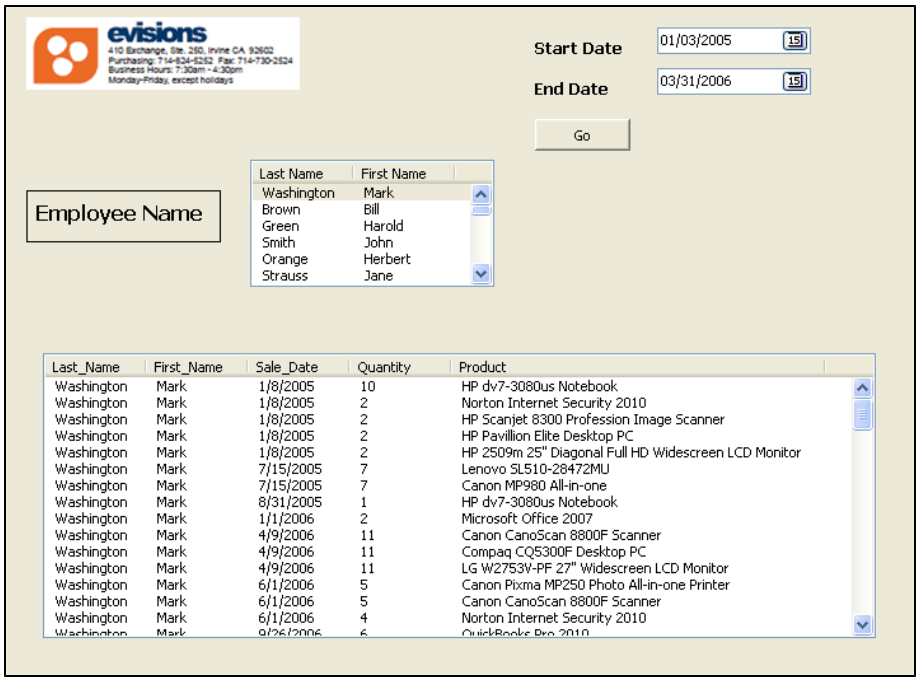
The DataBlock is the foundation from which all reports are created and contains **Forms** and **Queries**. Only users with DataBlock Designer privileges can create DataBlocks.

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.

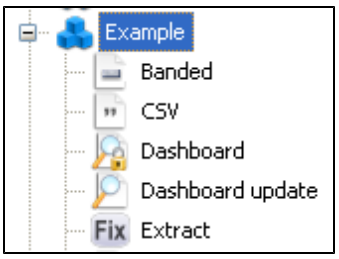
The **Form** created by the DataBlock Designer using the Argos DataBlock Designer, is used for two purposes:

- 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 form containing input selections and query results



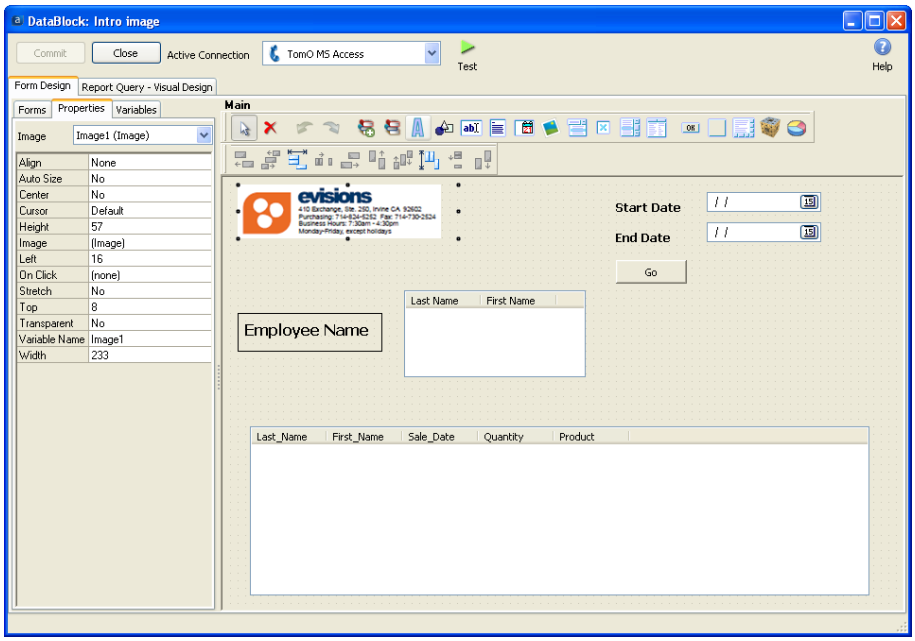
Reports as children of a DataBlock

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

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 and graphs (requires Charting module).
- OLAP Data Cubes for intuitively viewing and analyzing data (requires OLAP module).

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.



Form Design within DataBlock Designer

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.

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.

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.

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.

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

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.

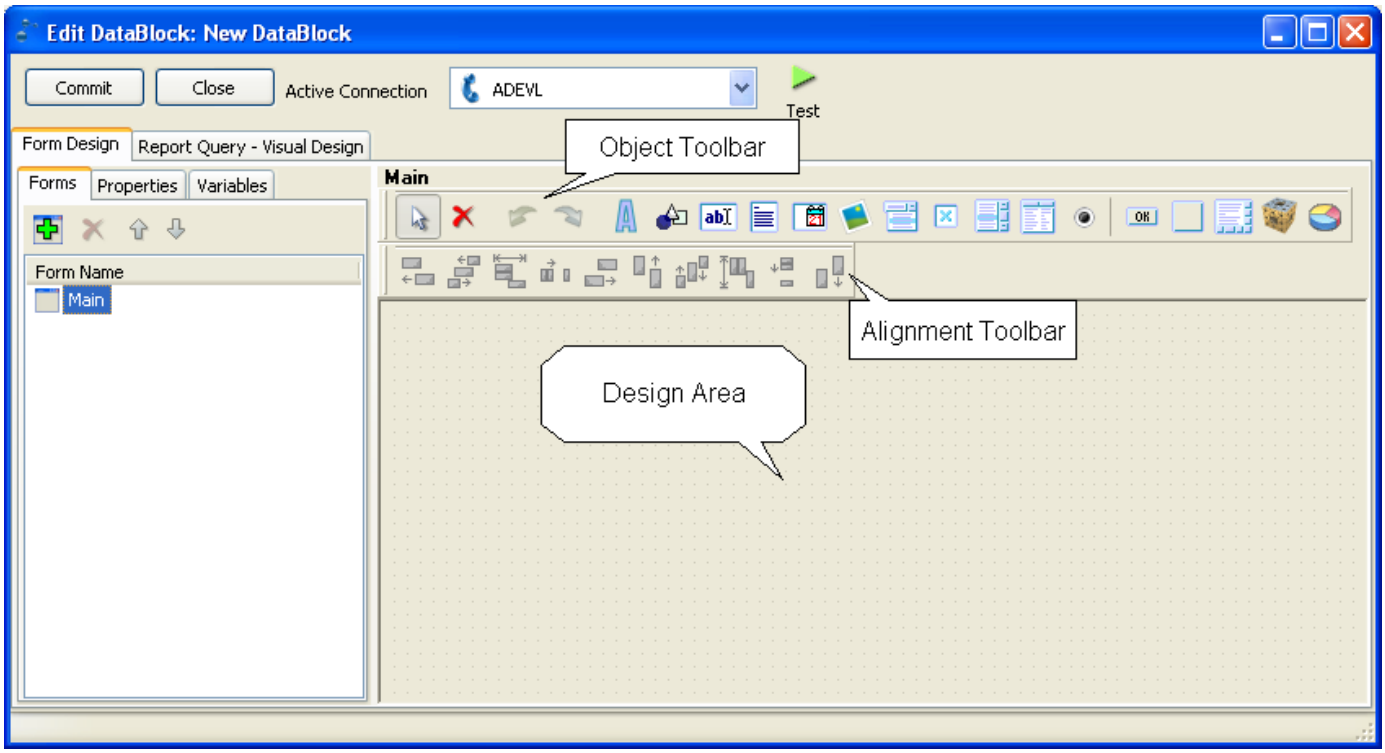
The user is presented data in an electronic spreadsheet format. By moving rows and columns or clicking them, the user makes the system perform calculations and show data in different aspects. Thus, the user can produce many reports out of a single dataset on his own, without any assistance from IT-specialists.

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.



The Argos DataBlock Designer

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
	Selection Arrow - 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.
	Delete - Use this tool to delete selected object(s).
	Undo / Redo - 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.

Icon	Description
	Static Label - 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.
	Shape - This tool allows you to create rectangles, squares, circles and ellipses. You can choose a rounded variation of rectangles and squares.
	Edit Box - This tool allows you to display a single line ofstatic text or to collect text input from a user. Edit boxes also support: ▪ Populating with data (Data Aware property) ▪ Collecting lists of items (Multi Entry property) ▪ Setting to Read Only
	Memo - 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: ▪ Populating with data (Data Aware property) ▪ Collecting lists of items (Multi Entry property) ▪ Setting to Read Only
	Date Edit -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: ▪ Populating with data (Data Aware property) ▪ Ability to default "today" or a fixed date ▪ Ability to define first day of the week
	Image - This tool allows you to add images to your DataBlock. When you add an image, you have three options: 1. 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. 2. 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. 3. 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. 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.
	Drop Down - 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: ▪ Ability to automatically select first list item (Auto Select property) ▪ Customized column headers (Columns property) ▪ Ability to type in choice (Free Type property)

Icon	Description
	Check Box - 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.
	List Box - 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: ■ Ability to automatically select first list item (Auto Select property) ■ Customized column headers (Columns property) ■ Ability to select multiple list items (Multi Select property) ■ Ability to show the list item count (Show Item Count property)
	Multi-column List Box - 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: ■ Ability to autosize a column (Auto Size Column property) ■ Ability to add column and/or row lines ■ Ability to define a custom multi-field sort order (Sort property)
	Radio Buttons- 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. ■ 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". ■ 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.

Icon	Description
	Button - 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. There are a number of events that can be activated using the On-Click Property as follows: Activate Form – activate another form in the forms list Clear Variables - Clear the value of selected DataBlock variables Fetch File – Retrieve a file from the Server Hyperlink – Launch a URL in the default web browser Refresh Variables - Refresh the value of selected DataBlock variables Reset Variables – Reset the value of selected DataBlock variables back to their default value
	Panel and Scroll box - 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 Panels can be assigned a color (Scroll boxes are always white) and Scroll boxes automatically create scrollbars as needed. While neither objects 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.
	OLAP Data Cube - Create On-Line Analytical Processing (OLAP) Data Cubes.
	Chart - Create a variety of charts such as bar graphs, pie charts, line charts, etc. When adding a chart, the Chart Wizard is be 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.

Description of each item within the Object Toolbar

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

Example 1

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:

DataBlock icon

This icon represents a DataBlock in the Explorer Tree and other locations throughout Argos.

Add DataBlock icon to Explorer

system Dashboard icon

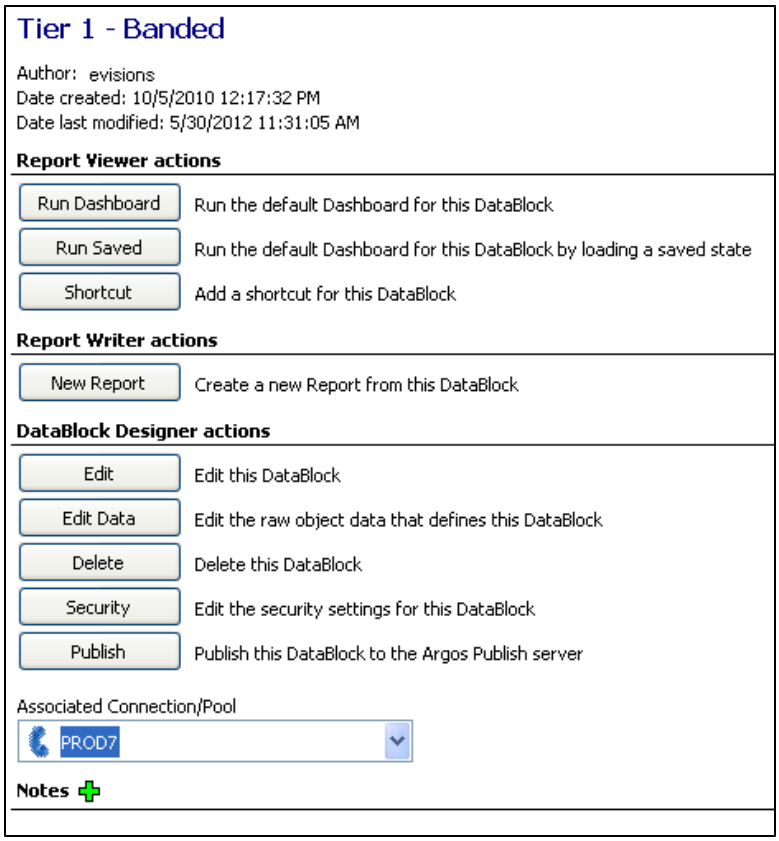
Log into Argos and navigate to the location where you would like to add the DataBlock. Right-click and select New -> DataBlock. Rename the "New DataBlock" to something of your choosing. Notice that Argos also created a system dashboard underneath the DataBlock.

Select a database connection

Choose the appropriate database connection from the **Associated Connection/Pool** drop down.

Associated Connection/Pool

See your system administrator to find out which database connection to use.



Selecting the database connection

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.

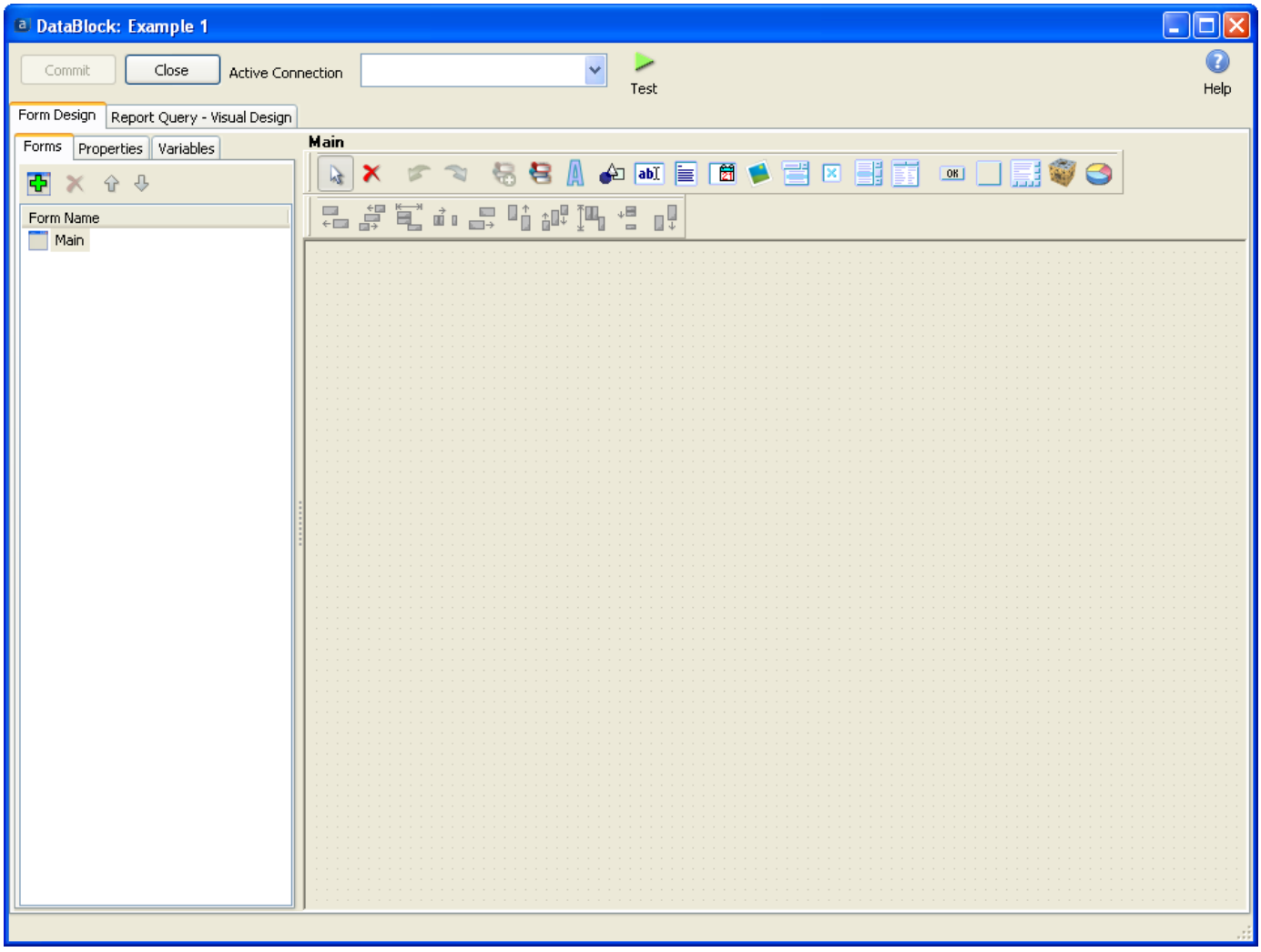
You can also click "Author" to edit the author of the DataBlock. The default author is set to the Argos user name of the person currently logged in.

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

On the right side of the screen, 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.

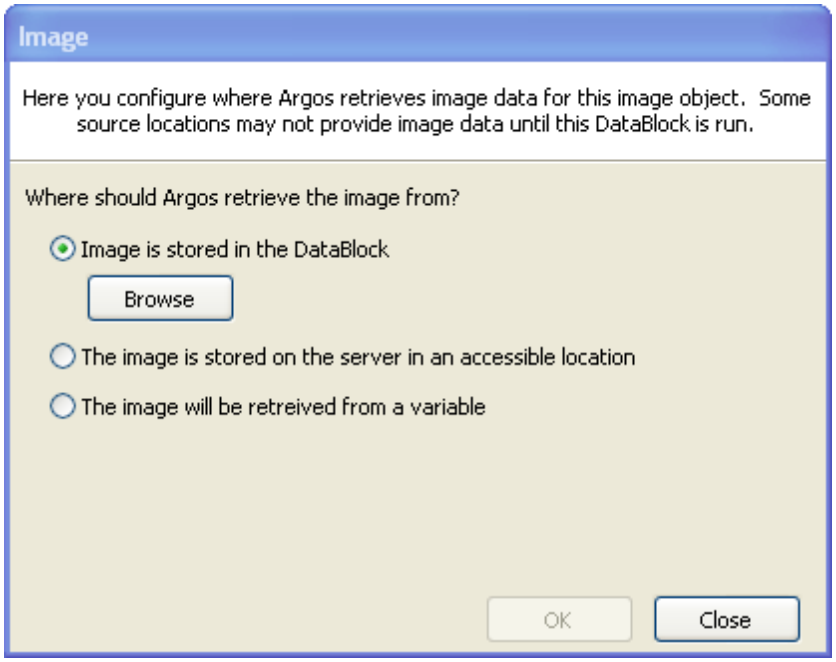


The Argos DataBlock Designer

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.

Add image icon on toolbar



Adding the Graphics Object

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.

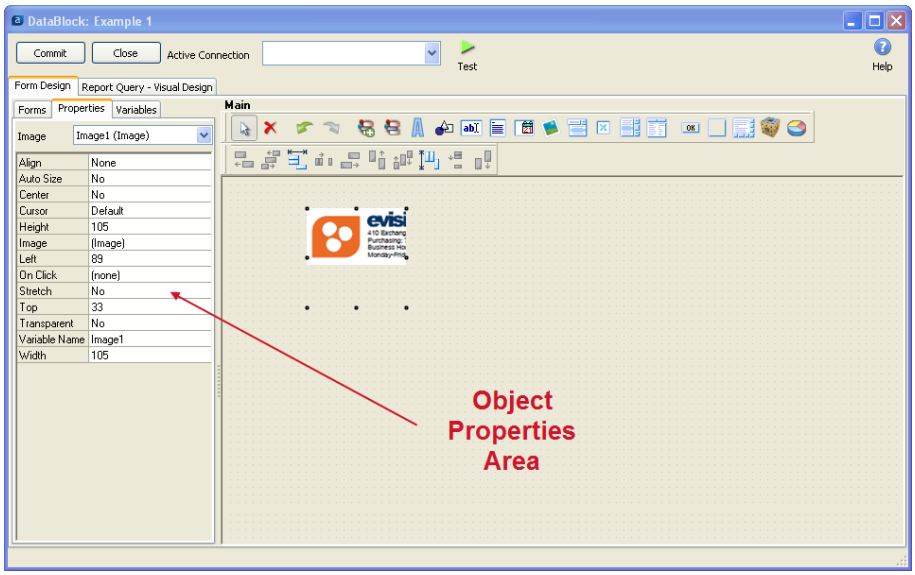
Image stored on an accessible server

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



The Object Properties Area within the DataBlock Designer

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.

Object Properties

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

Forms/Properties/Variables Tabs

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

The **Forms** tab displays a list of all forms used within a DataBlock. This example uses only one form, but other examples show how to create and link several forms together.

The **Properties** tab displays the properties for the selected object. If no object is selected, nothing is displayed.

The **Variables** tab displays a list of all variables existing in the DataBlock. Use of variables will be demonstrated in this example

Add the Date Objects and Labels

Click the "date" 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.

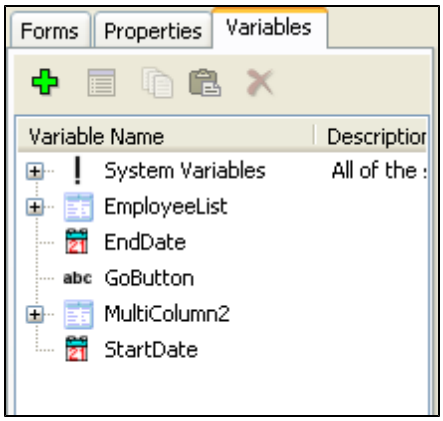
Note the property called "Variable Name" in the figure below. This is the default name assigned to the object by Argos. Since the date objects will be referred to later when building SQL queries, it is helpful to create meaningful names. Therefore, rename the first date object "StartDate" and the second 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	64
Variable Name	EndDate
Width	121

Renaming the EndDate object

Date icon on toolbar

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

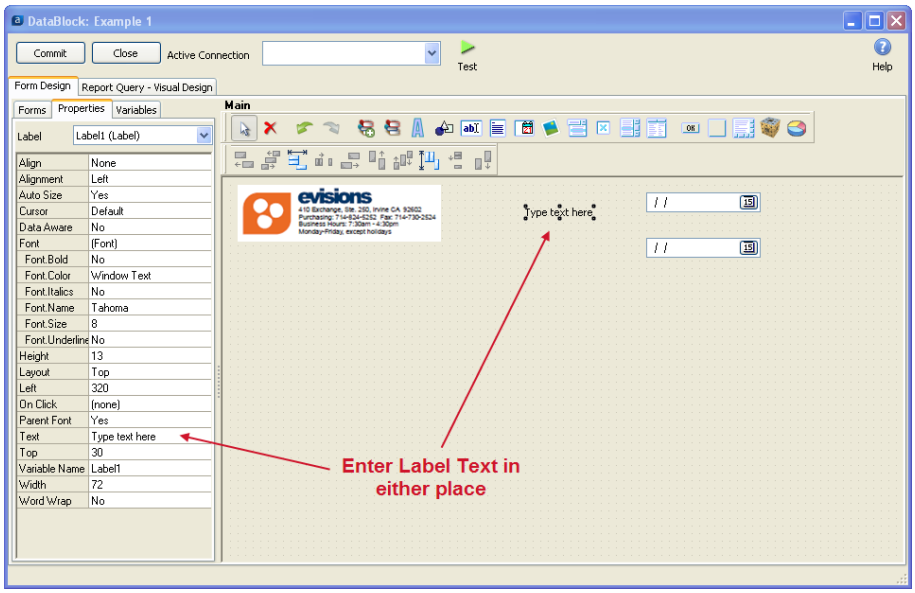


The Variables Tab within the DataBlock Designer

Static label icon on toolbar

The next step is to add a label to each Date object. Click the large "**A**" 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.



Adding Text for a Label Object

Align the objects

You will want to align the date fields along their left sides, so single-click on the Start Date object, hold the shift key, then single-click on the End Date object. Then, 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.

The alignment toolbar

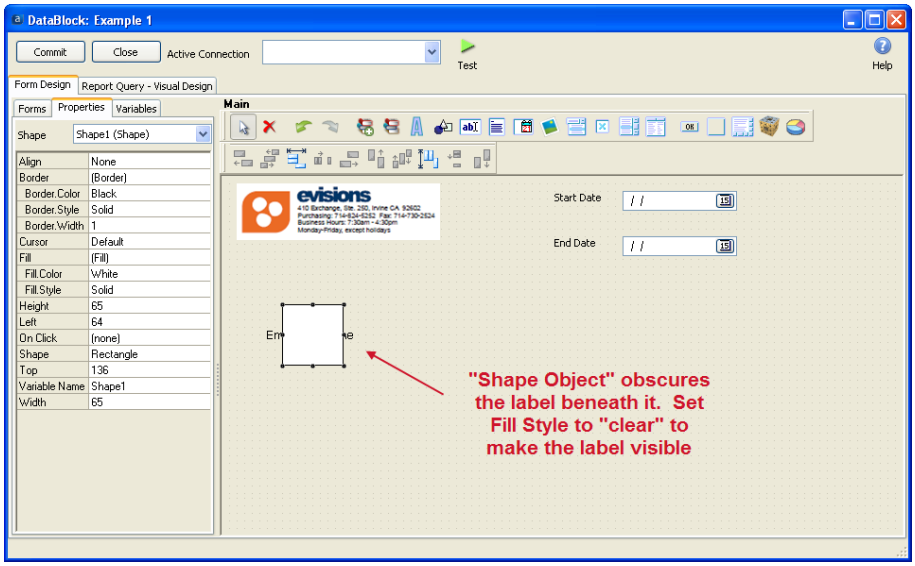
Hovering the cursor above each icon explains its purpose.

Add Employee Name Label and Shape Object

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

The Shape icon on the toolbar

Using the technique described above, add a static label "Employee Name" to the Design Area. Then click the "Create a shape on the form" icon. A box with a white fill color will be placed on the Design Area, so drag it over the "Employee Name" label. The white fill obscures the label, so change the Fill Style to "Clear". You can then modify the size of the rectangle by dragging the handles at the corners to surround the "Employee Name" label.



Adding a Shape Object

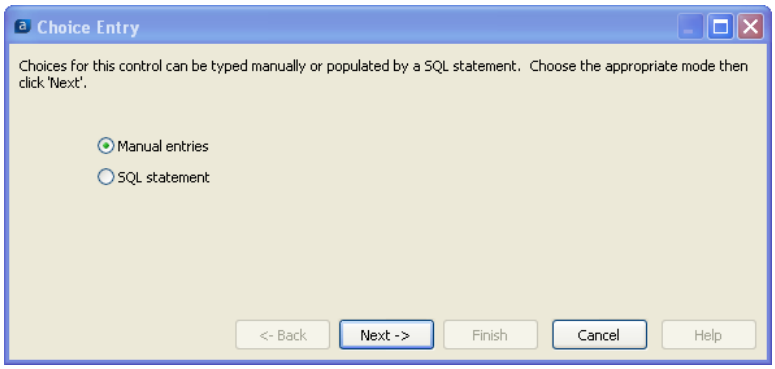
Create the object for selecting Employee Name

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.

Multi-column list box icon on toolbar

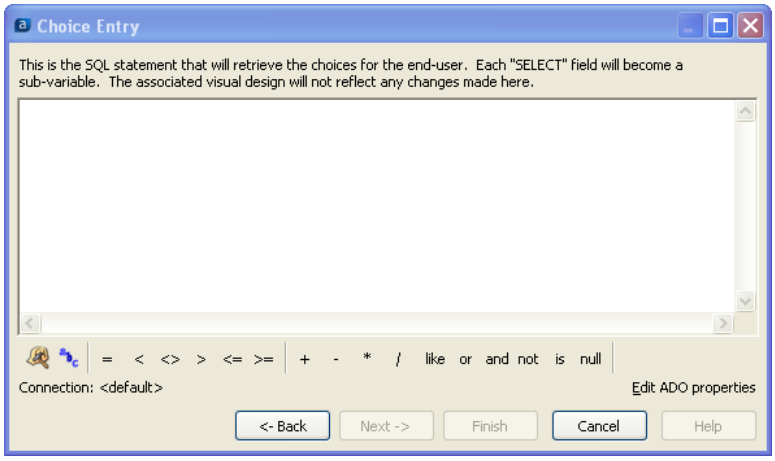
Begin by clicking the multi-column list box icon then click anywhere in the design area to place the object. Since this object is used later when building an SQL query, give the object a meaningful name such as "EmployeeList" by editing the Object Properties area.

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. You can choose to enter the employee names manually or obtain them from the database. In this case, entering the names would be laborious and error prone, so choose "SQL statement".



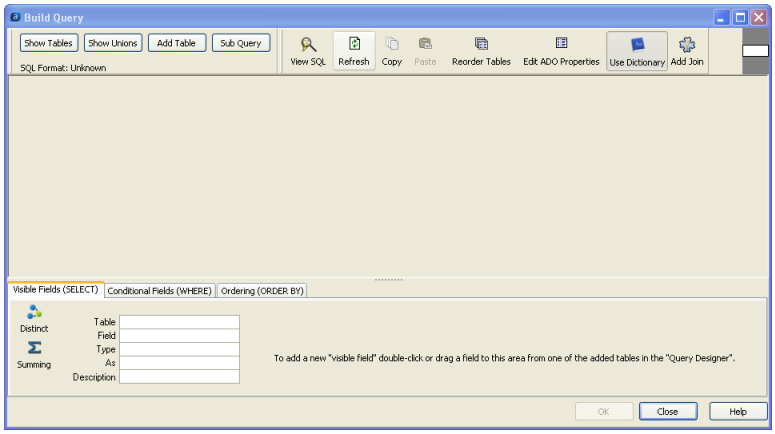
Choosing the method to populate the employee list box

After selecting "SQL statement", the following dialog box appears:



Creating choices for the multi-column list box

Click the hardhat/hammer icon (on the lower left of the dialog box) which brings up the Build Query dialog box. Click "Show Tables" at the upper left which displays the list of tables within the database.



The Build Query dialog box

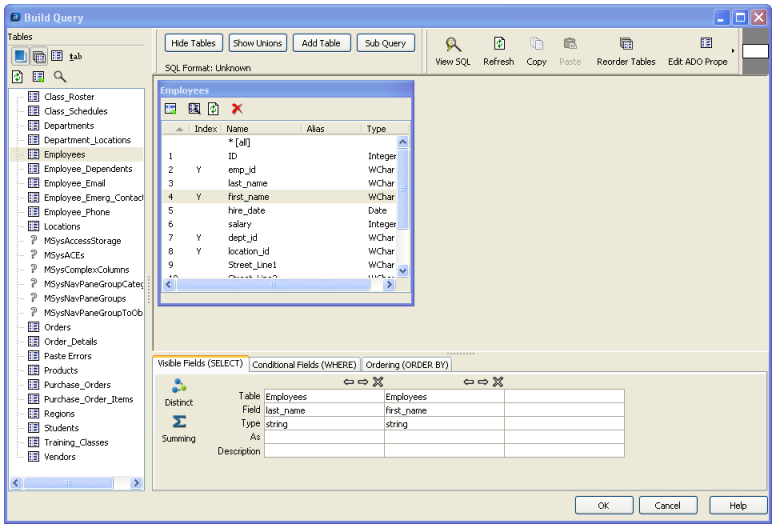
Object Choices Property

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.

Multi-select property

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

The hardhat/hammer icon

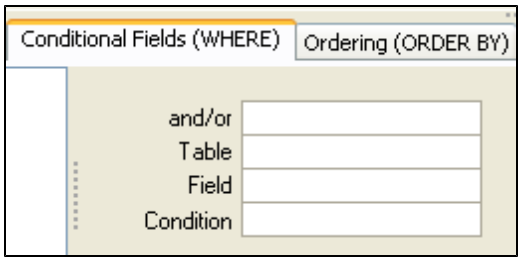


Identifying the database fields to be visible on the form

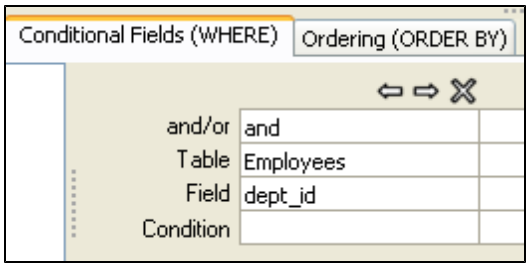
The Employees table contains the employee name. Double-click Employees from the list to add the table to the query. The Employees table appears on the right, listing its fields. Double-click last_name and first_name to move them to the query table at the bottom of the screen. Fields that appear in the query within the Visible Fields tab will become visible on the form and are included in the SELECT statement.

The Employees table contains a list of all employees in all departments. Since only employees in the Sales Department are to be listed, a WHERE clause must be added to obtain only the appropriate employees. The department that the employee belongs to exists in the field dept_id within the Employees table. The dept_id for the Sales Department is 'dep03'.

Click the "Conditional Fields (WHERE)" tab which displays the following:



Click on each field (and/or, Table, Field) and select the following values to add a condition to the query:



The Build Query dialog box

The Build Query dialog box shown at the left provides a means for the Argos user to create an SQL query visually, that is, without entering SQL statements directly.

The Builder allows the user to visually declare which database tables, fields, and conditions are to be used in the query.

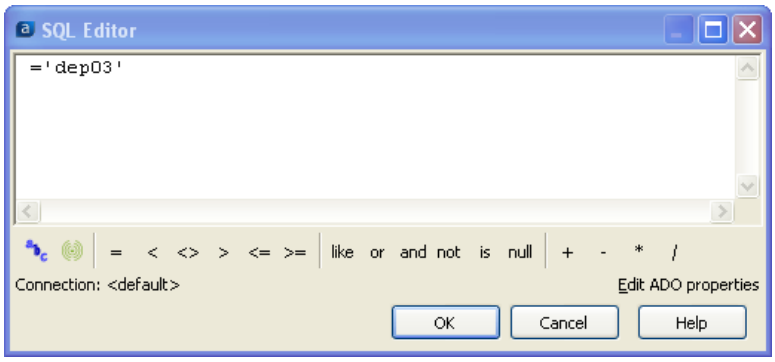
The left hand portion of the window displays the available tables. The user selects the tables of interest from this list.

The middle portion displays the contents of each table and allows you specify which database fields are to be included in the query. Table joins are also created in this area.

The bottom portion of the window shows fields and conditions that become SQL statements. SELECT, WHERE, and ORDER BY statements are created here.

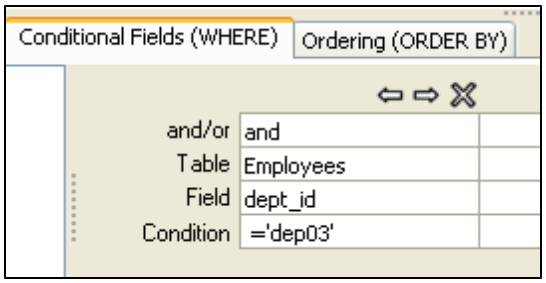
Clicking the "View SQL" icon at the top displays the SQL statements that were visually created.

Click the "Condition" field, then click the ellipsis button which brings up the SQL Editor. Enter the department ID as shown in the figure below.



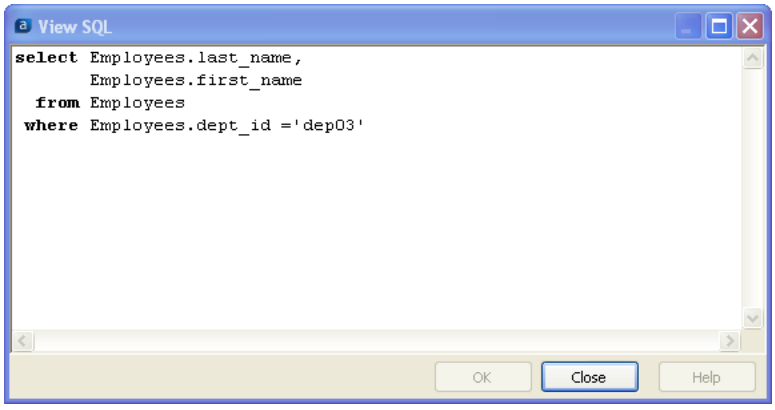
Entering the Sales Department ID into the SQL Editor

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



The completed condition

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.



The resulting SQL

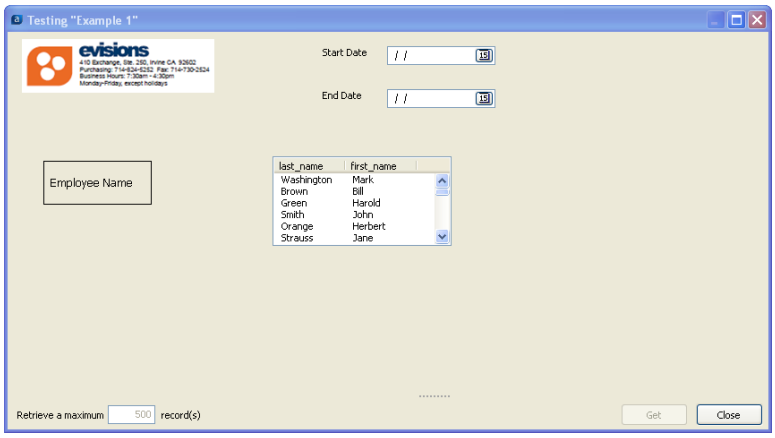
Click "Next" and "OK" buttons on each form as they appear to complete the creation of the SQL. You have now visually created the SQL to display the employees in the Sales Department into the multi-column list box.

The list of variables and database fields in the SQL Editor

This icon displays the list of variables

This icon displays the list of database fields

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 Green "Test" icon to test the form/query design.

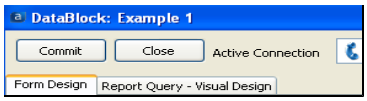


Testing the current design

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.

The Commit and Test Buttons



Save your work periodically by clicking the Commit button.

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.

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, select "SQL statement" (to populate the list box with results of an SQL query), click the hard hat hammer to bring up the Build Query dialog box, then click Show Tables.

The columns in the report will contain the:

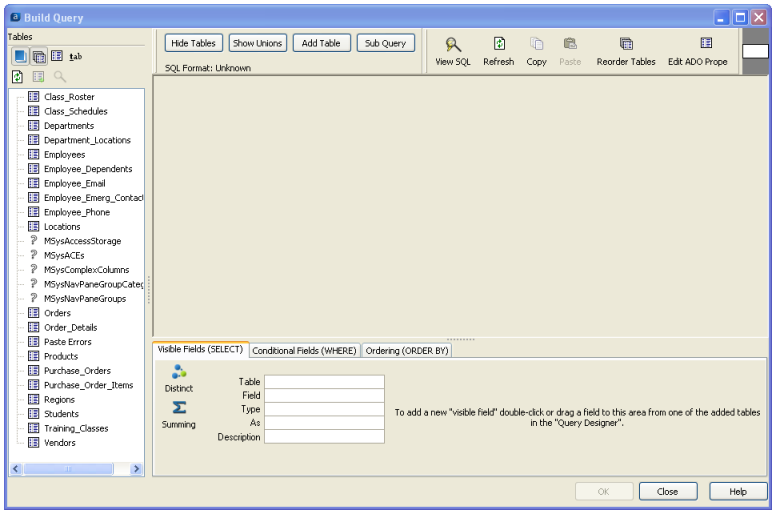
- employee last name
- employee first name
- order date
- product name
- sales amount (quantity x unit cost)

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.

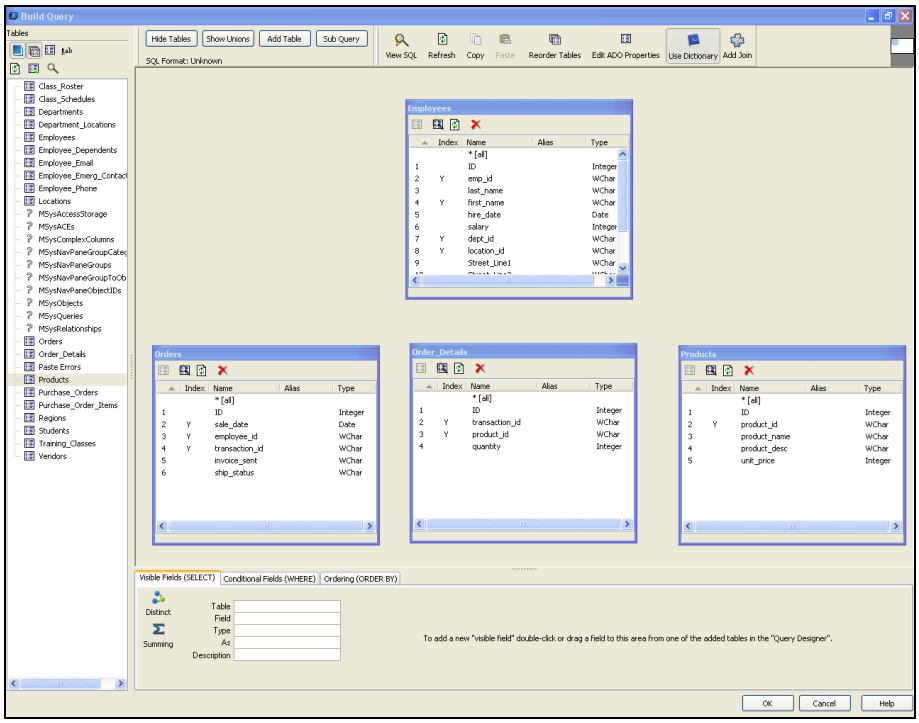
Form Queries

Although this examples 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.



Showing all database tables within the Build Query dialog box

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.



Selecting the tables applicable to this query

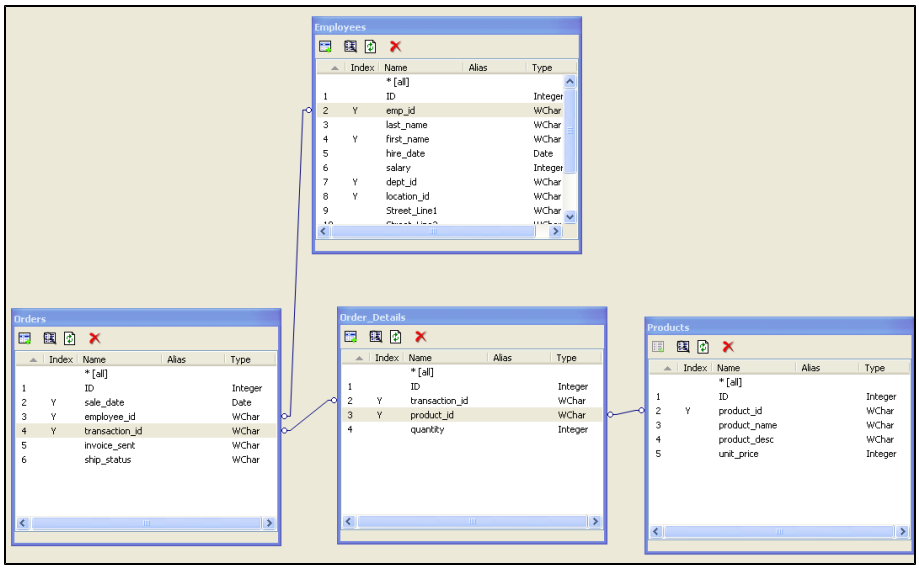
The following joins need to be created:

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

Modifying the joins

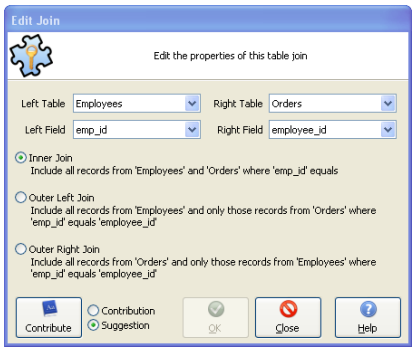
To modify a join, place the cursor on the line joining the two tables,then right click. The following options will appear:

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:



Joining the tables within the Build Query dialog box

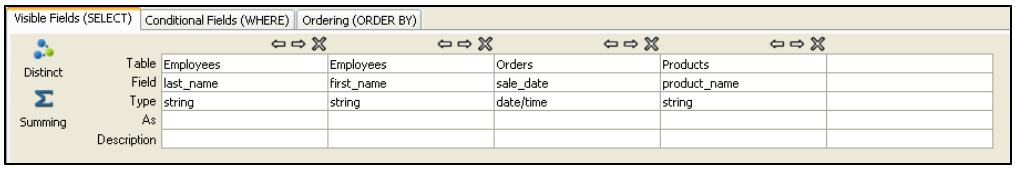
Selecting "Edit Join" will allow you to change the type of join as shown below



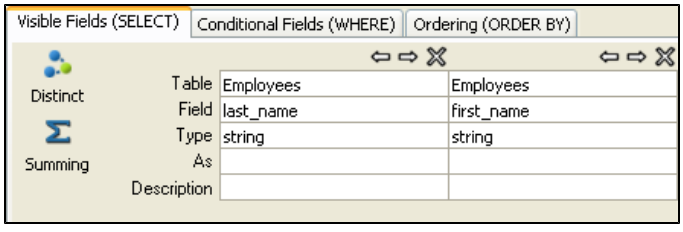
Identify the fields to appear in the report

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 employee last name, first name, sale date and product name were selected and appear within the Visible Fields tab.



SELECT/WHERE/ORDER BY tabs

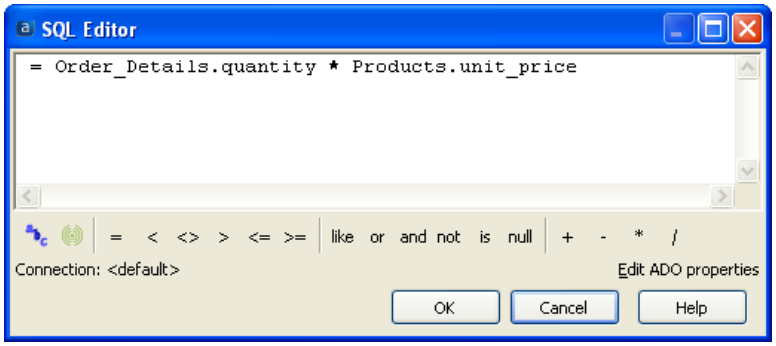


- The Visible Fields (SELECT) tab contains the fields that will appear in the report. These fields will also exist in SELECT statements.
- The Conditional Fields (WHERE) tab contains the fields that appear in the WHERE clause of the SQL.
- The Ordering (ORDER BY) tab determines the sort order of fields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

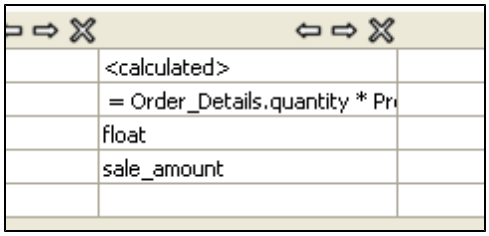
Create a calculated field to determine sale amount

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, click the empty field to the right of "Products" and use the SQL Editor as shown below to create the calculated field.



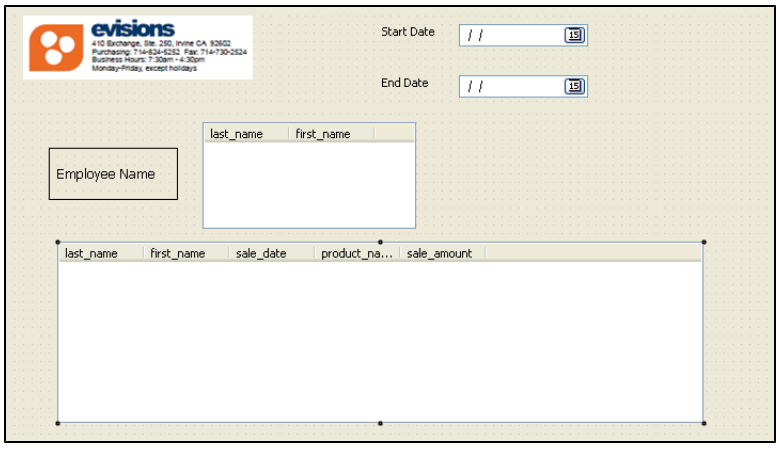
Entering the calculated field into the SQL Editor



The calculated field shown within the Visible Fields tab

The new calculated field named sale_amount can now be used in the same fashion as other fields in the database.

Click OK, then Next to continue. You will then see the new multi-column list box object shown within the DataBlock Designer with all of the fields added.



The multi-column list box to contain the query results

Centering an object

To center an object on the form, click the object. two of the icons on the alignment toolbar will then become active as shown below.

These two icons are highlighted when only one object is selected. After clicking the desired icon, the object will be centered within its parent region horizontally or vertically.

All alignment icons become active when two or more objects are selected. In this case, objects are aligned with respect to each other.

Use input selections to limit the query

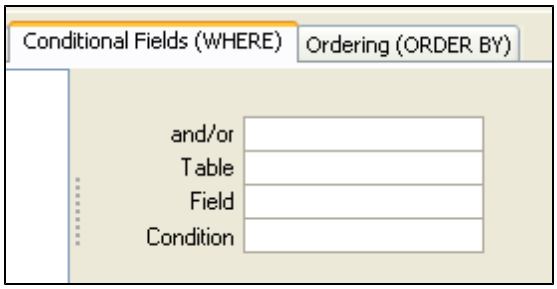
This query in its current state will return all sales records for all employees. However the query is to be limited by the input selections entered on the form. Therefore, a WHERE clause is required to limit the query to selected employees within a date range.

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:

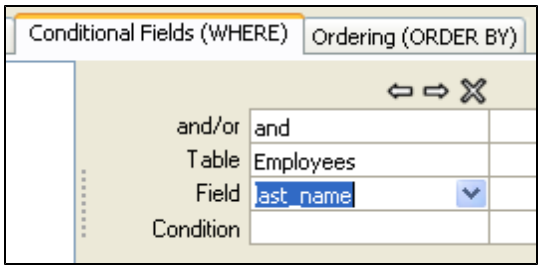
- The employee last name selected = the employee last name in the database and,
- The employee first name selected = the employee first name in the database and,
- The sale date is greater than or equal to the start date selected and,
- The sale date is less than or equal to the end date selected

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 as follows:



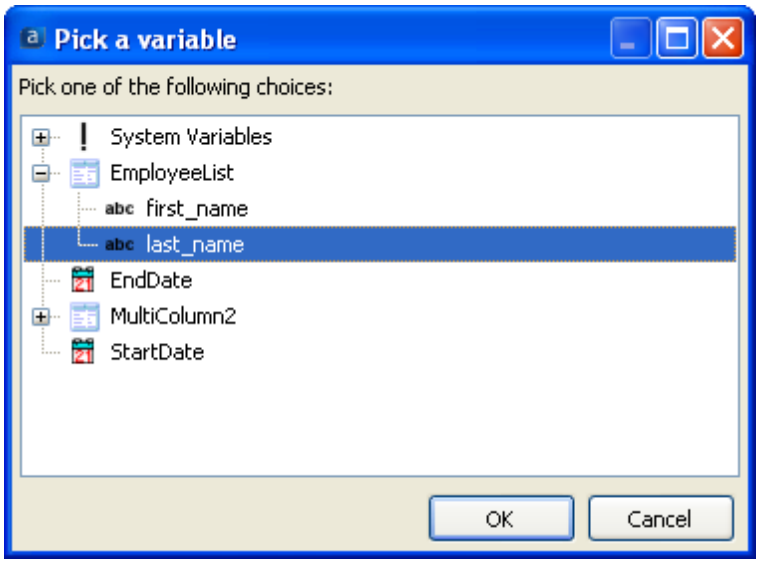
The Conditional Fields (WHERE) tab in the Build Query dialog box

The condition for last name will be created first. Enter the data shown in the figure below in the "and/or, Table, Field" locations, then click "Condition" beneath them.



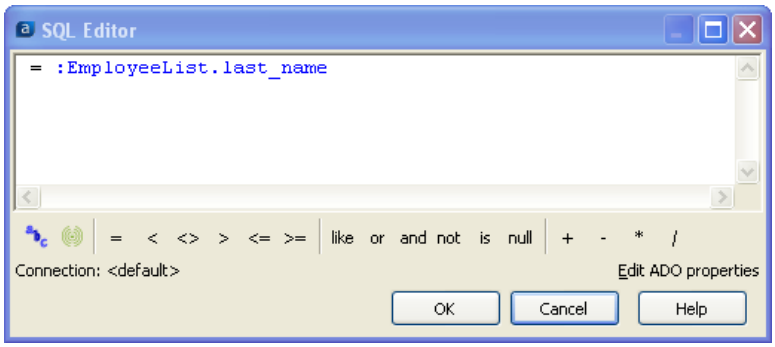
Adding last name to the condition

Click the ellipsis, then the abc icon which displays the variables that exist within the DataBlock. Note that the list of variables contains the first name and last name under the EmployeeList object (the multi-column list box where the user selects employees of interest). The variable list also contains the StartDate and EndDate objects. When the report is executed, each of these variables contains the selection made by the person executing the report.



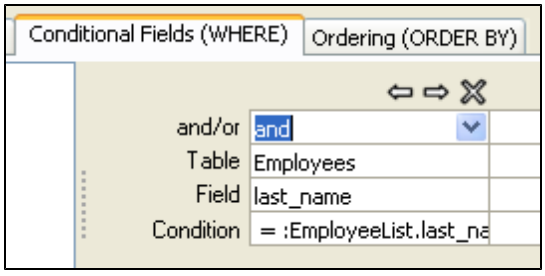
Selecting the last name variable to be used in the Condition

Select last_name as shown above. Recall that EmployeeList is the name of the multi-column list box containing the list of employee names. The two fields within the EmployeeList object are last_name and first_name. Click OK to continue.



The last name condition within the SQL Editor

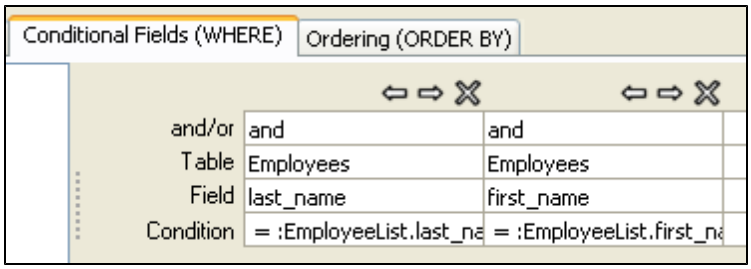
Click OK to continue



The completed condition for last name

The condition for last name has been created as shown in the above figure.

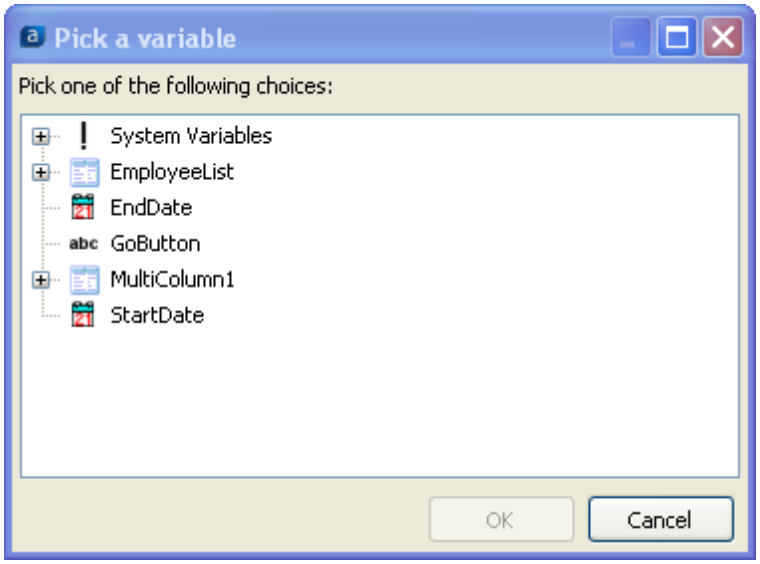
Click on the blank field to the right of the field just created and repeat the process for the employee first name.



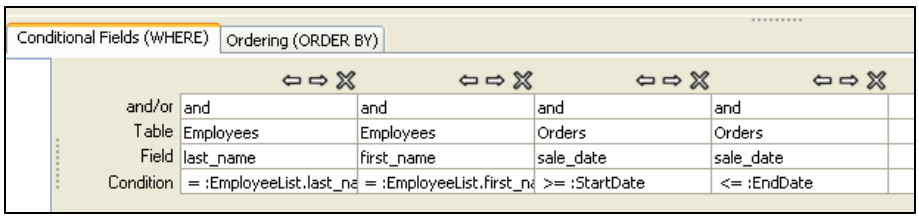
Adding the condition for first name

To add the date conditions, click on the empty fields on the right of "employees first name" to create conditions where sale date is greater than or equal to the input start date and sale date is less than or equal to the input end date. Use the same methods that were used to create conditions for last name and first name.

Note the StartDate and EndDate variables that are displayed in the list of variables. It is now very apparent why giving meaningful names to variables is useful.



The complete variable list



Creating the conditions for start date and end date

Click OK then Next to complete the design of the WHERE clause and the entire form/query design. You are now back in the main screen of the DataBlock Designer. Click Commit to save your work, then click the green right arrow to test.

Saving your work

Remember to click the Commit button on the top left of the Visual Designer to periodically save your work.

Use of button to control the execution of the query

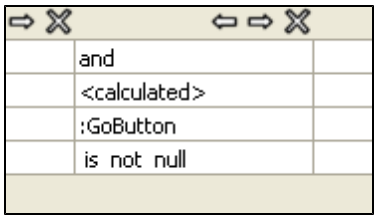
When you execute the Dashboard, you will notice that the query results are shown as soon as you finish selecting the employee name and dates, not giving you a chance to change selections before executing the report. This is why the GO button was created; to prevent the query from executing until the GO button is clicked.

To remedy this situation, edit the DataBlock (bringing up the DataBlock Designer) and add a button by clicking the button icon on the toolbar (the icon with OK). A button will then be placed on the form. Move the button to the desired location. Change the variable name of the Button to "GoButton" within the properties area of the DataBlock Designer. Change the text displayed on the button to "Go" by editing the Text property.

The Button icon on the toolbar

Double-click on the object that displays the query results (the multi-column list box at the bottom), click the hardhat/hammer, then select the "Conditional Fields (WHERE) tab.

Within the WHERE clause that has already been created, in the empty column in the right, a calculated field will be created to inhibit execution of the query until the Go Button is clicked. Create the condition shown below

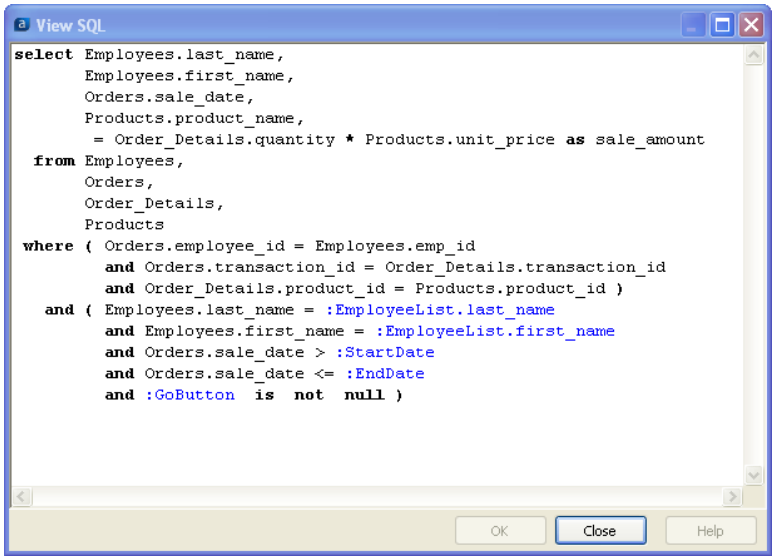


Creating the condition for the GO Button

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 then execute.

Summary

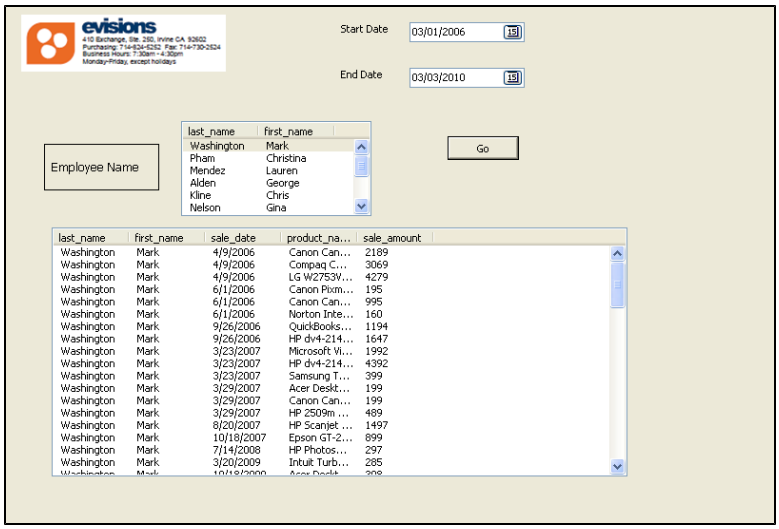
The final SQL for this example appears below. Note the existence of the variable names (in blue) and the fields contained in the select and where statements.



The final query for the query display

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.



The dashboard

Report Query

Example 2

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.

The DataBlock and associated Dashboard appear in the Argos Explorer as shown below.

DataBlock/Dashboard

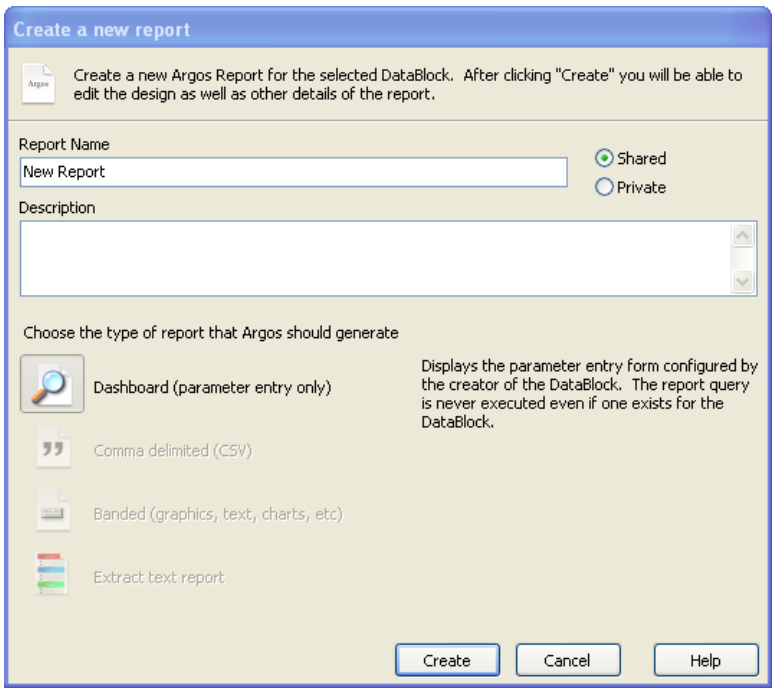
Form Queries and Report Queries

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

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.

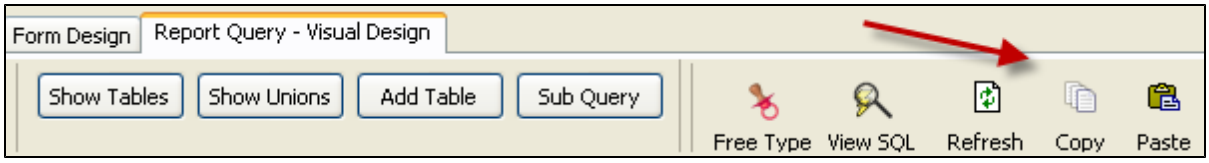


Creating a new report

Copy/Paste the Existing Query

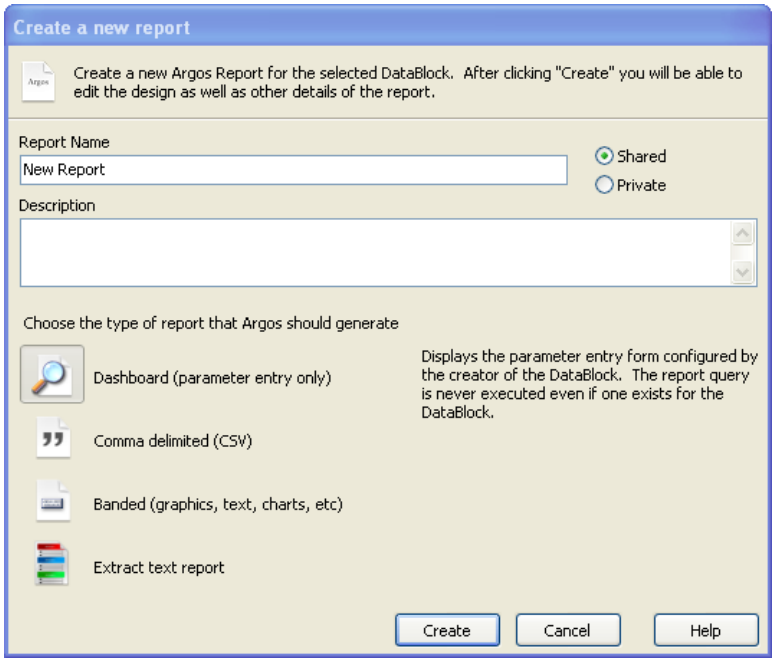
To use the same query for CSV, banded, and extract reports, you must copy the form query into the report query:

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.



CSV, Banded, and Extract Reports now selectable

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

Example 3

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.

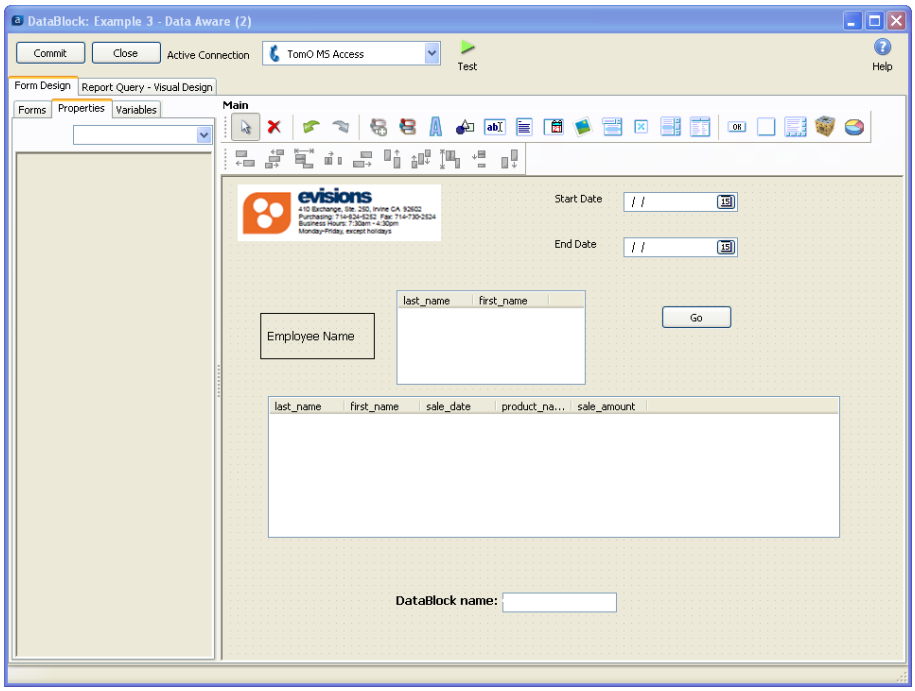
To illustrate the use of the Data Aware property, the DataBlock used in Example 1 will be expanded to include an edit box to display the name of the DataBlock.

Add the Data Aware Object

Select the DataBlock created in Example 1, right-click and select "Edit DataBlock".

Resize the multi-column list box to leave room for the new object at the bottom of the form.

Add the edit box with its associated label near the bottom of the form as shown in the figure below.



Edit Box added to the form

Objects with Data Aware Property

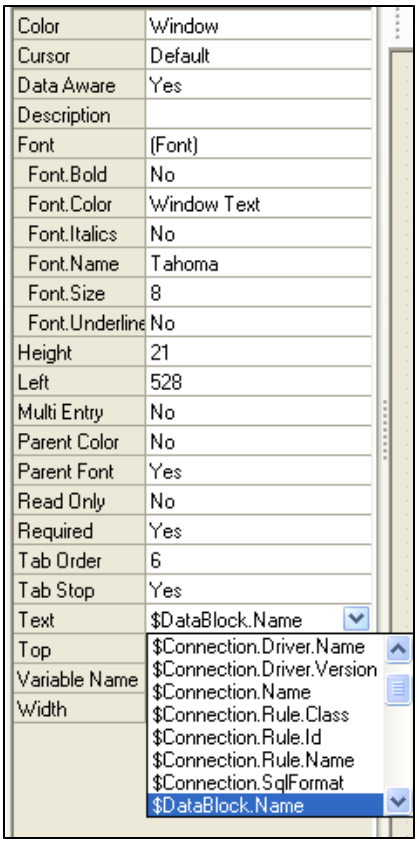
- Label
- Edit Box
- Memo field

Align	None
Alignment	Left
Auto Size	Yes
Cursor	Default
Data Aware	No **Data Aware**
Font	(Font)
Font.Bold	No
Font.Color	Window Text
Font.Italics	No
Font.Name	Tahoma
Font.Size	8
Font.Underline	No
Height	13
Layout	Top
Left	285
On Click	(none)
Parent Font	Yes
Text	Type text here
Top	74
Variable Name	Label6
Width	72
Word Wrap	No

Set Data Aware property for edit box

Click on the "DataBlock name" edit box which displays its properties shown below. Change the Data Aware property to Yes and click on the Text property. This object, which by default is used to enter text, becomes a drop-down box as a result of setting the Data Aware property to Yes.

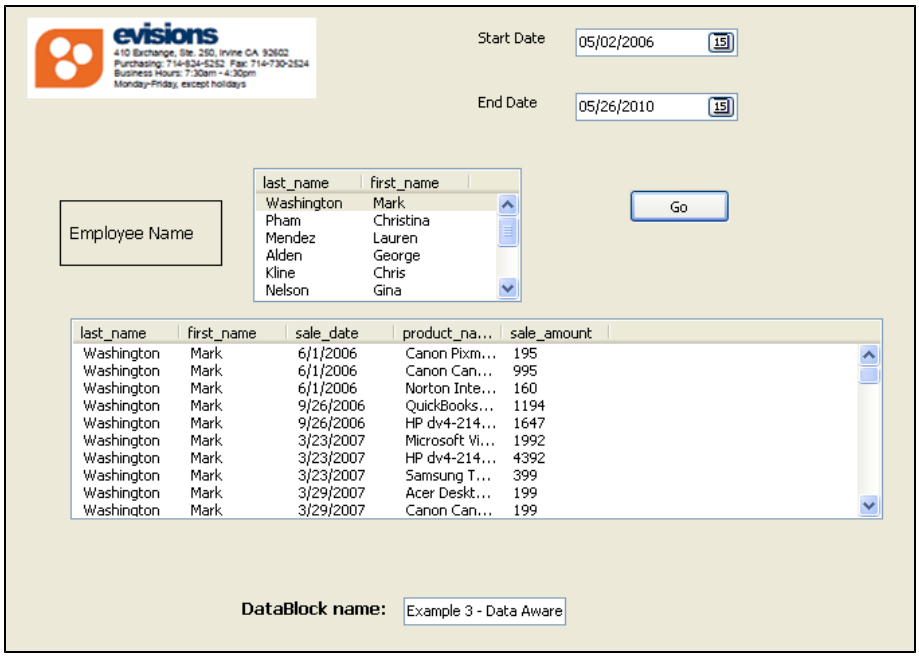
Select $DataBlock.Name from the drop-down list. The DataBlock name for this example will then appear in the edit box when the report is executed.



Properties for edit box

Execute the Report

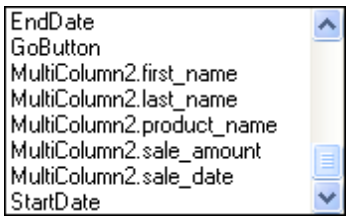
Executing the Dashboard produces the following report. Note the DataBlock name in the newly added edit box.



The report results

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.



Other variables for this DataBlock

In the drop-down box on the left, the variables beginning with "MultiColumn2" are the fields in the list box that display the sales results.

Using Multiple Forms in a DataBlock

Example 4

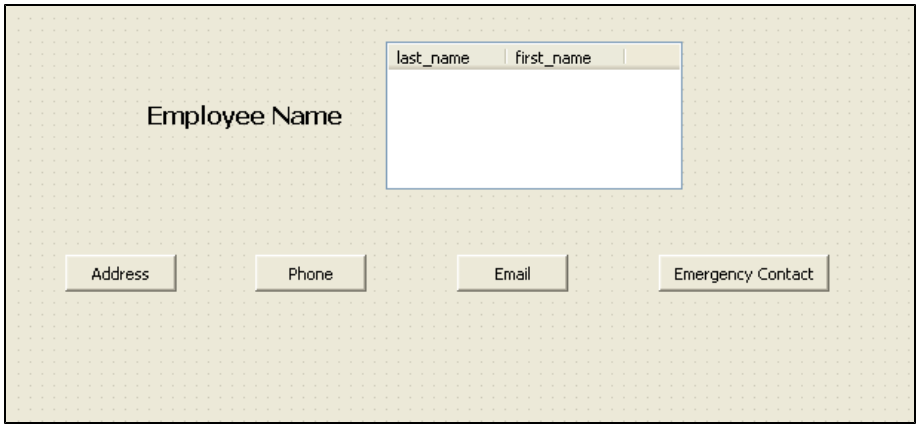
Introduction

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.

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

Contact types used in this example



The Main Form

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".

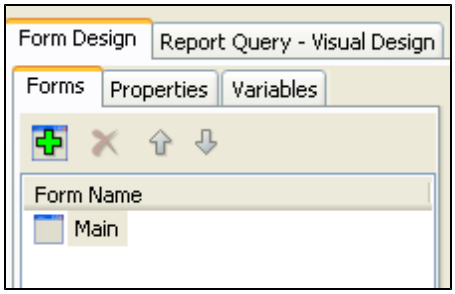
All of the concepts for building the forms in this example have already been presented in Example 1; therefore not every action that will be discussed needs detailed explanation.

The Sample Database

See the Introduction of this guide to find the location and description of the sample database.

Create the DataBlock

The first step is to create a DataBlock and launch the DataBlock Designer. Then, click on the Forms tab.



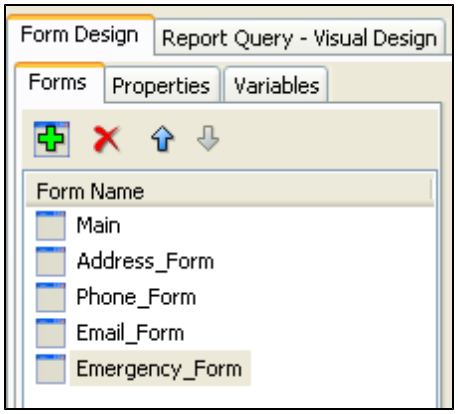
The Forms Tab in the DataBlock Designer

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 green plus button to create the additional four form names. The new 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.

Form Names

Use the up/down blue arrows to change the order of the forms. Click the red **X** to delete a form.



Adding the four additional forms

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

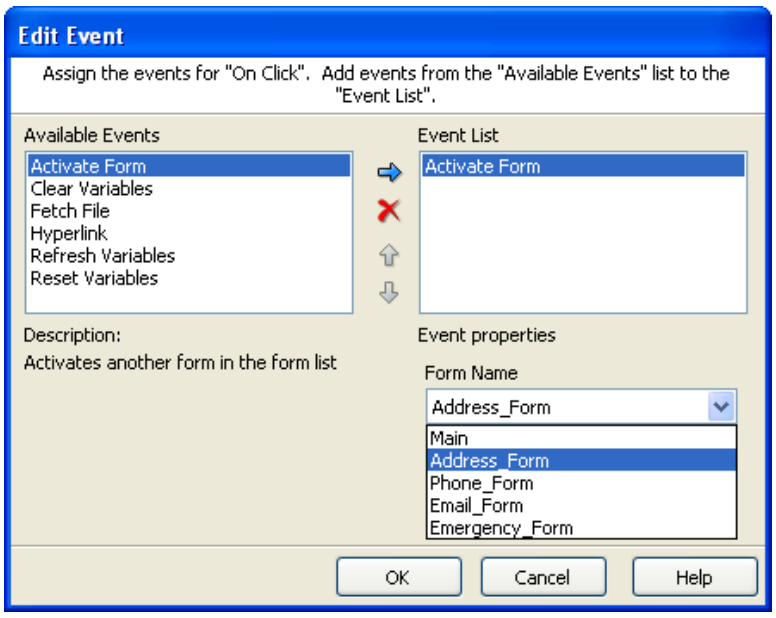
The properties for the Address button are as follows:

Cursor	Default
Description	
Font	(Font)
Font.Bold	No
Font.Color	Window Text
Font.Italics	No
Font.Name	Tahoma
Font.Size	8
Font.Underline	No
Height	25
Left	40
On Click	(click to change)
Parent Font	Yes
Tab Order	1
Tab Stop	Yes
Text	Address
Top	168
Variable Name	Address_button
Width	75

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, clicking the Address button launches another form which displays address information for the selected employee. The "On Click" property is used to activate (execute) the appropriate form when this button is clicked.

When the On Click property is changed, the following dialog box appears which allows you to select which form to present when this button is clicked. The form named "Address_ Form" was selected since this is the form to execute when the Address button is clicked.



The On Click Property

This property is available for buttons, shape objects, static labels, graphics, and chart objects.

The Forms Tab

In the figure to the left, the Address Form is highlighted, thus the Address Form is displayed in the Design Area.

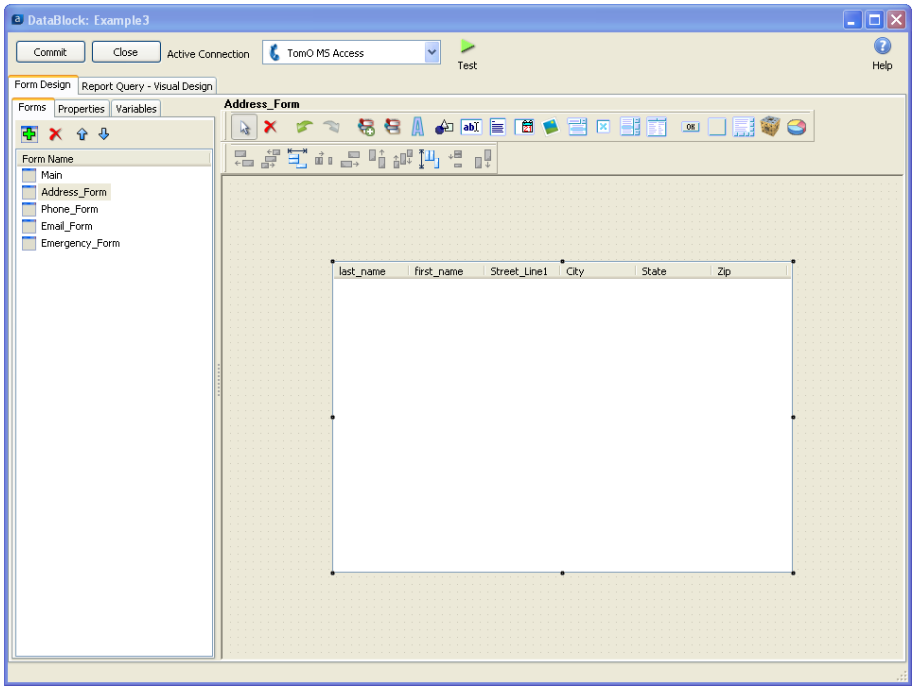
Note that various events can be activated by the On Click event. For this example only the "Activate Form" event is used. Click the Help button to see a description of all event types.

Repeat this process for each of the buttons. That is, enter the Variable name, Text name, and the form to activate via the On Click property.

Now that the properties have been defined for all of the buttons, the final step is to create the forms/queries executed when the Address, Phone, Email, and Emergency buttons are clicked.

Create the form/query associated with each button

To create the Address form/query, click on the Forms tab within the DataBlock Designer, then select the Address form. The Design Area of the DataBlock Designer will be blank. You can now create 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. The last_name, first_ name, Street, City, State, and Zip fields were chosen.



The design of the address form

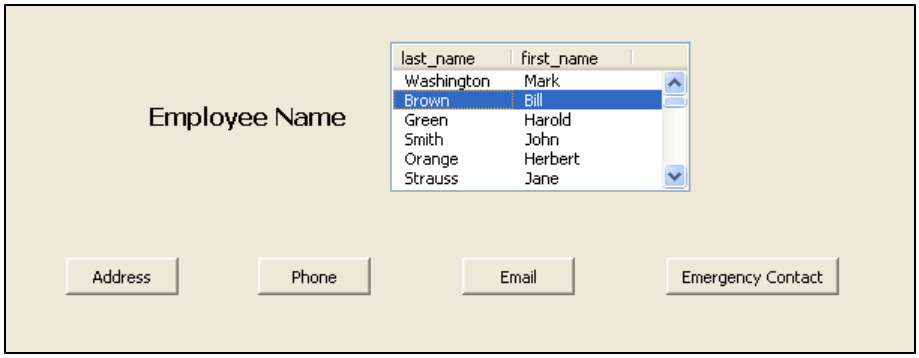
After the query for the Address form is complete, click on the "Phone_form" form within the Forms tab to design the form/query to display employee phone numbers. Again, you will see the Design Area blank. As was done above, create a multi-column list box to contain the phone numbers obtained from the 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. This is done using the same techniques described in Example 1.

Repeat this procedure to create the form/queries for the Email and Emergency buttons.

Summary

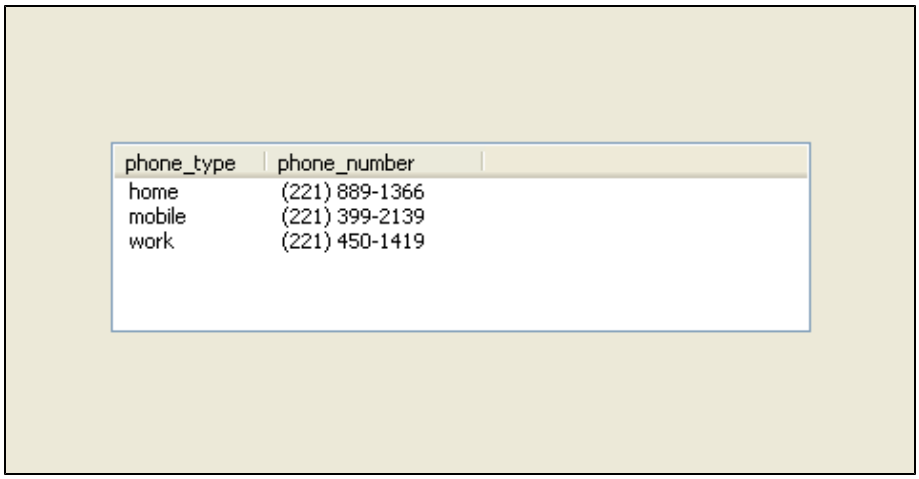
The multi-form creation process is now complete. Buttons were created to activate associated forms, and each type of query result is displayed on its own form.

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



Selecting the employee on the main form

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



Query results for the phone number form

Using tabs instead of buttons to navigate between forms

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 created a tabbed interface.

Charting with multiple series

Example 5

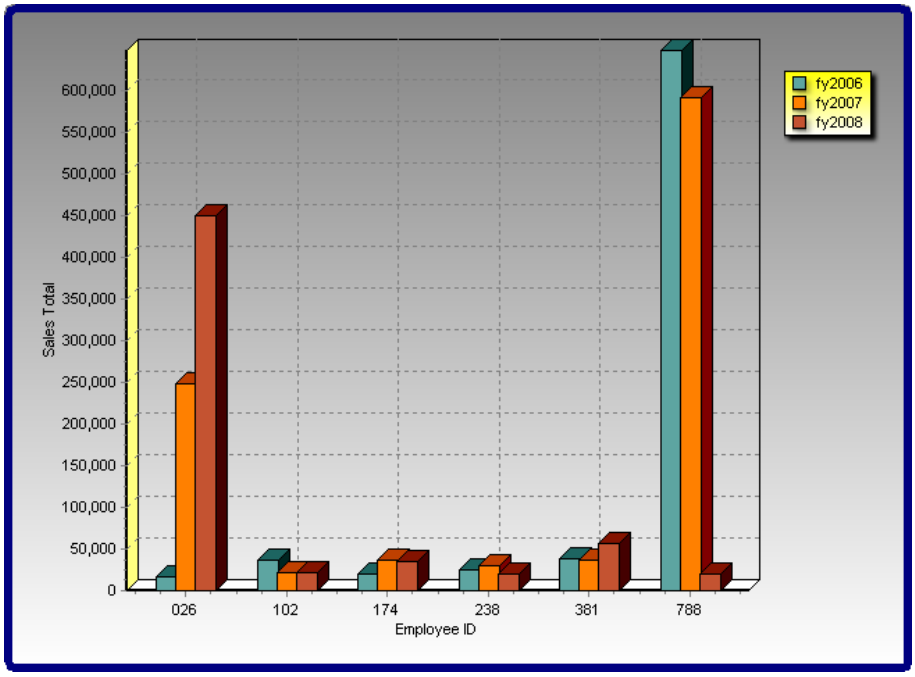
Introduction

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.



The chart for fiscal years 2006 - 2008

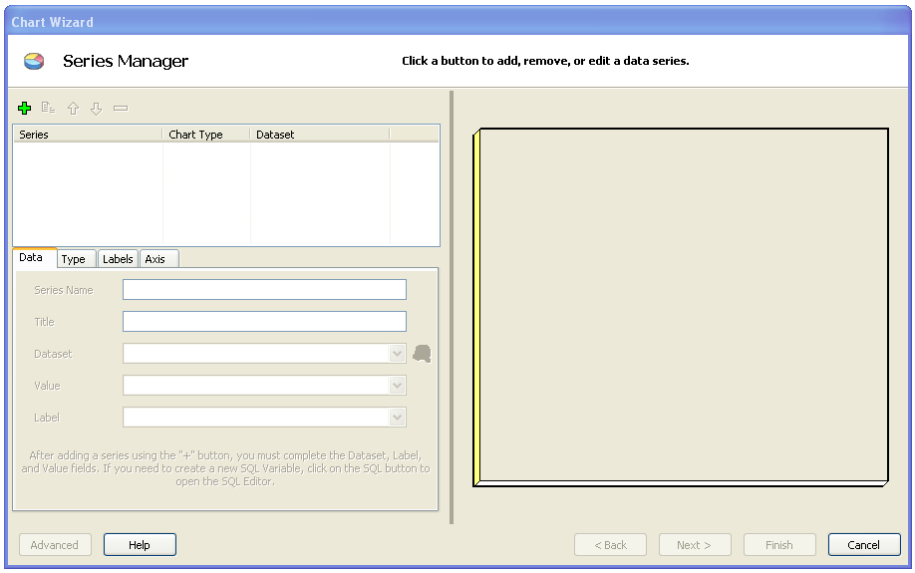
Create the Chart Object

The Chart Wizard was demonstrated in the Argos Report Writer Guide. It is assumed that this guide was previously read and understood, thus every step of the Wizard need not be explained again in this guide.

Click the chart icon, then click within the Design Area to create the chart object. Expand the object to the desired size.

Double-click the object to launch the Chart Wizard.

The chart icon on the DataBlock Designer toolbar



The Main Chart Wizard Interface

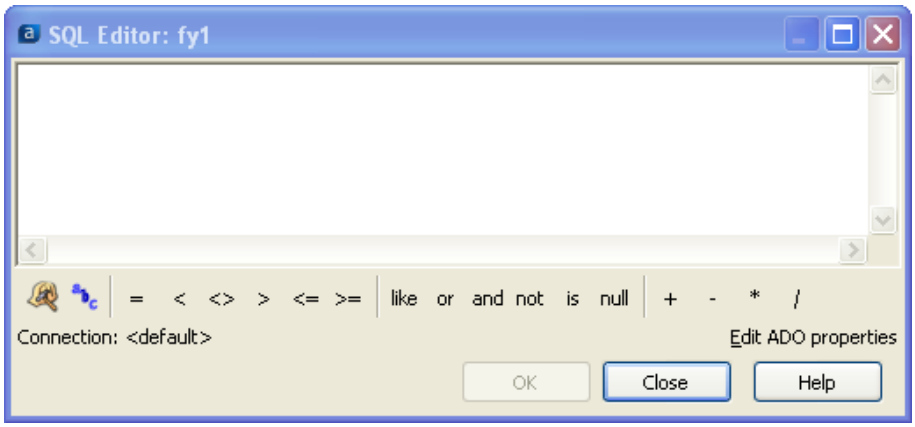
Name the first series

Click the green plus sign to create a series.

Change the Series name to fy2006

Click the hardhat/hammer to create a new dataset.

Enter 'fy2006' as the name of the dataset. Click OK twice to display the SQL Editor shown below.

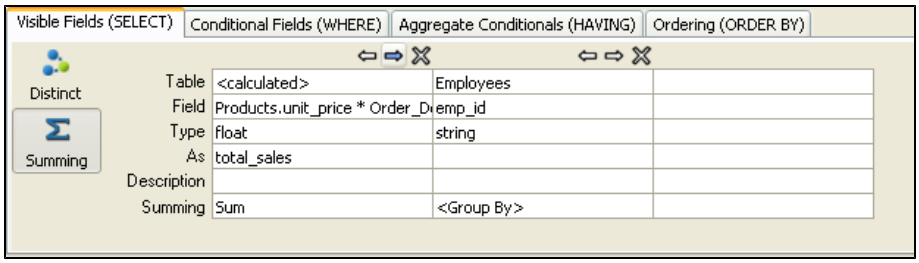


The SQL Editor

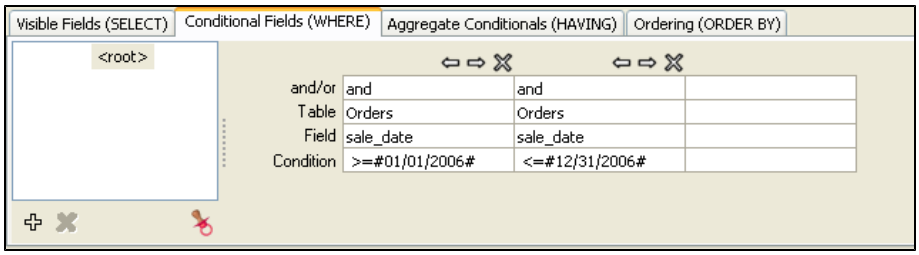
Click the hardhat/hammer to launch the Build Query dialog box.

Create 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 tab for the fy2006 dataset



The WHERE tab for the fy2006 dataset

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 twice to return to the Chart Wizard main interface.

Select the Chart Types, labels, and other options

Select total_sales for the Value field.

Select emp_id for the Label field.

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

Under the "Labels" tab, uncheck the Visible Label box.

Under the "Bar Options" tab, select the desired color for **this series** and uncheck the "Color Each" box.

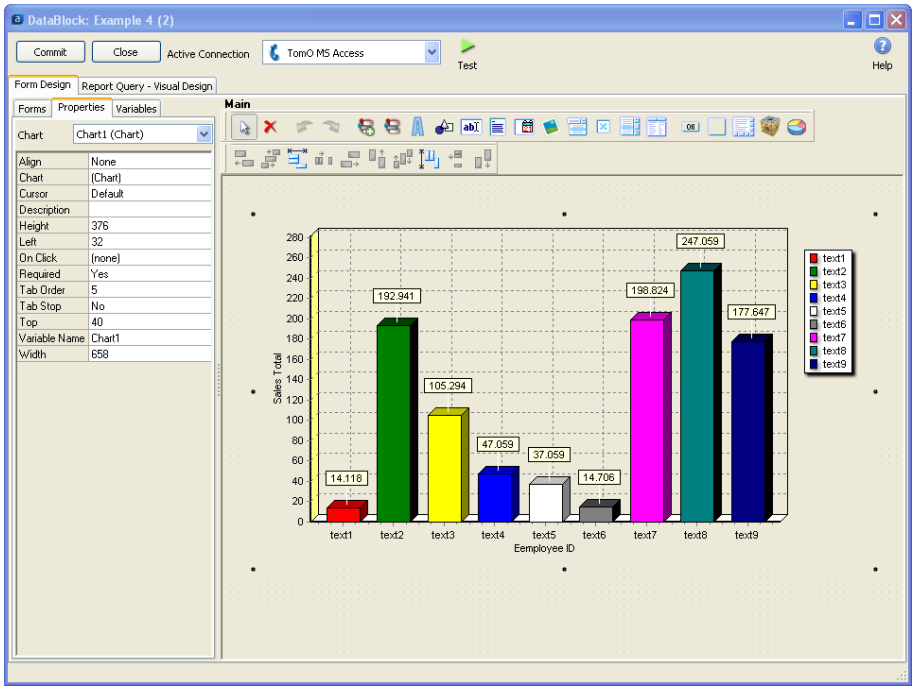
Click the Next button to view the Chart Theme and Panel selections. Choose whatever theme and palette you desire.

Click the Next button and enter the titles. This example used "Sales Total" for the Left Axis Title and "Employee ID" for the bottom Axis Title.

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

Entering the Date Condition

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



After completing the dataset for fy2006

Commit and test if you desire.

This completes the series and dataset for fiscal year 2006

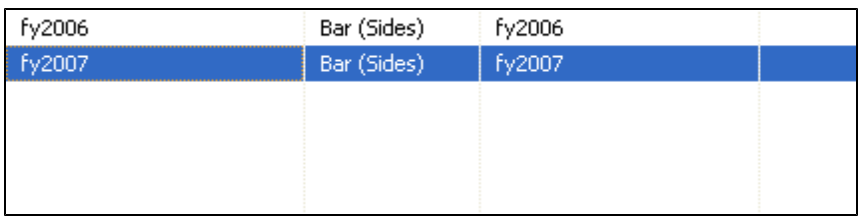
Double-click on the chart again to launch the Chart Wizard again.

Create the next series for fy2007

Before beginning the design for fy2007 you need to copy the fy2006 dataset. Therefore click fy2006 under "Series" and click the hardhat/hammer to bring up the Build Query dialog box. Click the Copy icon at the top of the dialog box then close the dialog box to return to the main Chart Wizard screen. The query that was copied will be pasted into the fy2007 dataset that is being created below.

Click the green button to add a new series and change the name.

Under "Series", click on fy2007 where you will now create all of the parameters for the fy2007 dataset. Remember to keep fy2007 highlighted as shown below. If you highlight the wrong series you will be applying updates to the wrong one. The figure below shows that you are creating/updating the fy2007 series.



Selecting the Series that is being created/updated

In the Dataset field, select "Create a new dataset…" then click the hardhat/hammer. Enter the name of the new dataset, which is fy2007. Click OK twice to bring up the Build Query dialog box. Click **Paste** to bring in the dataset created for fy2006. Within the Build Query dialog box, click the WHERE tab and change the dates from 2006 to 2007. Click OK twice to get back to the main Chart Wizard.

Select the same values for the Value and Label fields as was done for fy2006.

Choose the same selections for the Type, Labels, Axis, and Bar Options as was done for fy2006, except for this series select a different color for the bar under the Bar Options tab.

The dataset for fy2007 is now complete. Repeat the process for 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.

It is imperative that for each series the appropriate selections (Bar Options, labels, axis) are the same for each series. Otherwise you will mix options on the same chart, producing undesired results. For example, you wouldn't necessarily want one series producing a bar chart, and another series producing a line chart.

Also, 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.

Save this as a Dashboard.

Creating a Filtered OLAP Data Cube

Example 6

Introduction

One of the unique features of Argos is the ability to create filtered OLAP Data Cubes. That is, information from a datasource can be filtered before providing the information to the OLAP cube for analysis. Also, OLAP cubes can be built off 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 n the sample database. Items of interest include sales totals by region, by employee, by date, and by product type for all sales transactions.

References

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

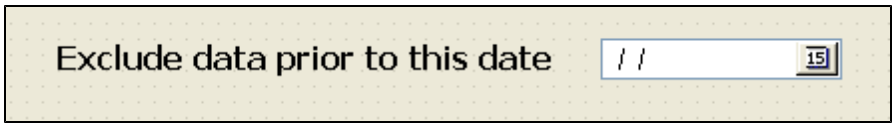
Data will be filtered by a date (specified by the Report Viewer) before it is sent to the cube for analysis. Even though date ranges could be filtered within the cube, for the sake of example, data will also be filtered by a date provided on the input selection form. Only sales after the date entered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

The steps for accomplishing this are as follows:

Create the OLAP Object in the DataBlock Designer

Create a DataBlock in the Argos Explorer tree and select the appropriate connection. Edit the DataBlock which launches the Argos DataBlock Designer.

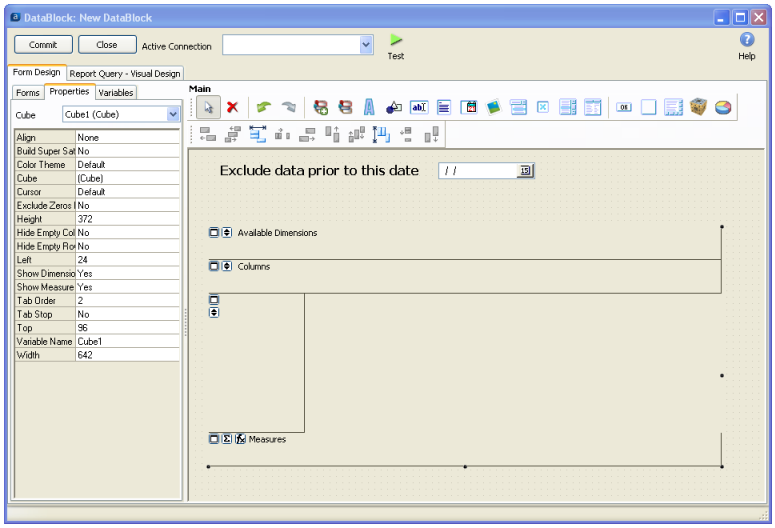
In the DataBlock Designer, add a date field which will be used to filter the data. The date will be used to exclude sales records prior to the date entered on the form.



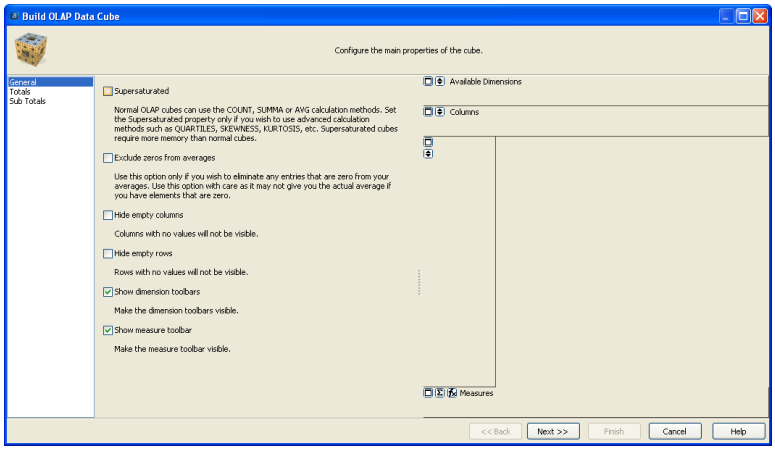
The input selection object

The OLAP cube icon within the DataBlock Designer Toolbar

Next, single-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.



Double-click inside the cube to configure the main properties of the cube.



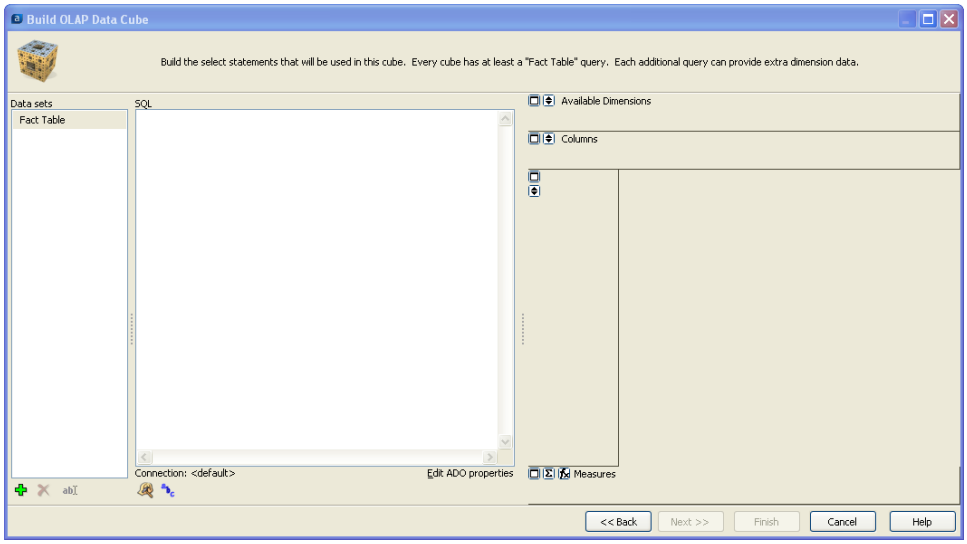
The Main Properties of the cube

Refer to Help for a description of each selection that appears on the Main Properties dialog box.

In this simple example, the default selections will be used, so click the Next button.

Build the Fact Table (the query)

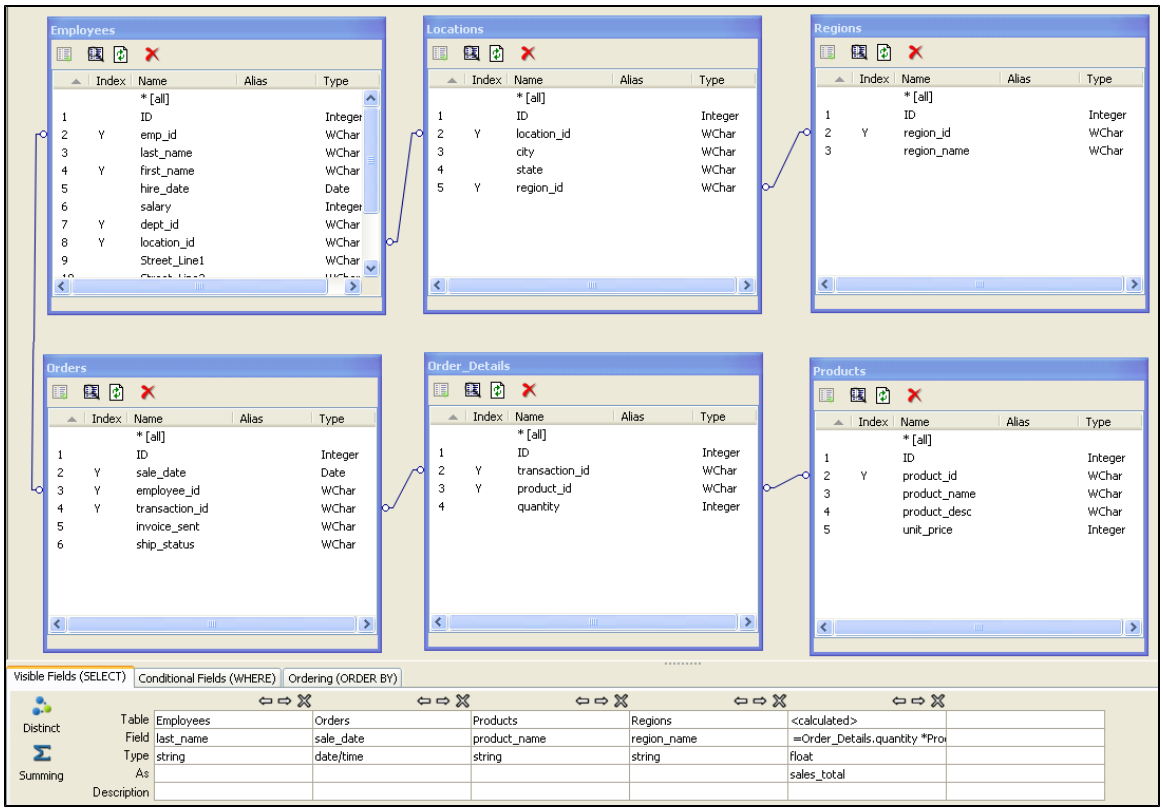
The following appears in which you create the Fact Table (the query used to obtain the appropriate records).



Click on the hardhat/hammer icon at the lower left to launch the Build Query dialog box that you have seen in previous examples. Build the query using the Employees, Locations, Regions, Orders, Order_Details, and Products tables. Join them as shown in the figure below.

Select the fields as shown below, and also create a calculated field which multiplies quantity times unit_price. **The fields shown will become the measures and dimensions within the OLAP cube.**

Also, use the WHERE tab to find only records where the sale_date is greater than the date input by the user.



The fact table (query) within the Build Query dialog box

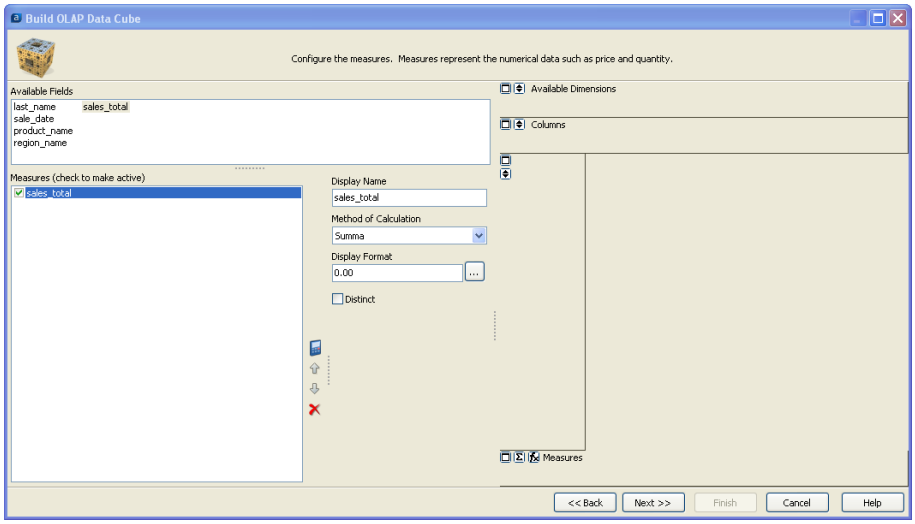
Click the Next button to continue.

Identify the Measures

You will now identify which of the fields above will be used as measures. For this example the 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.

Measures

The numbers (values) in the OLAP or cube. They are also referred to as facts.



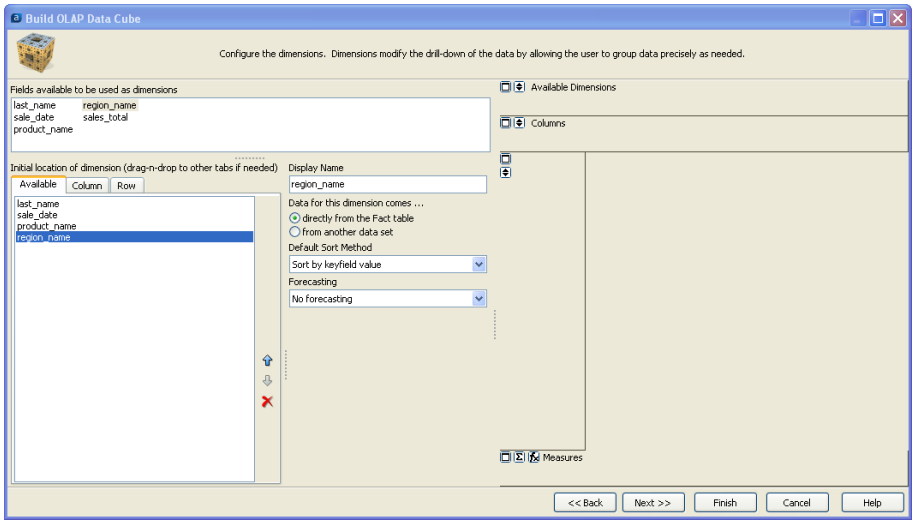
Click the Next button which brings up the figure below where you will identify the Dimensions.

Identify the Dimensions

In the same fashion, the fields to be used as dimensions are selected via double-click. Select the fields as shown in the figure below. Use defaults for the remaining options.

Dimensions

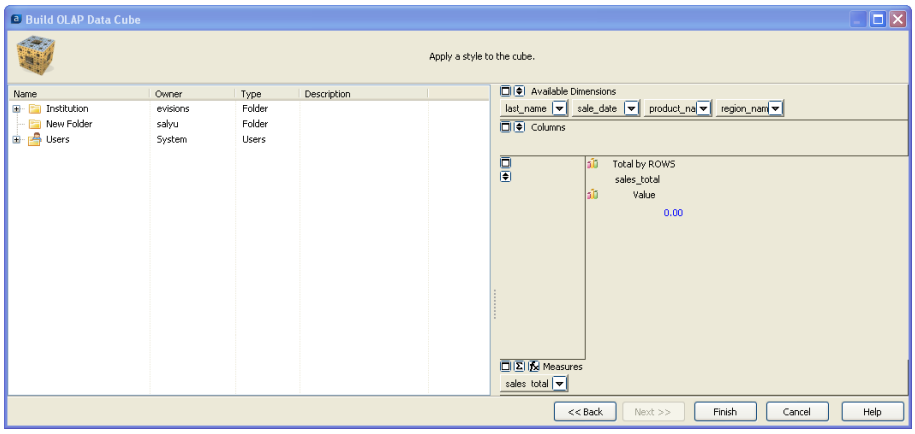
Fields by which data records are grouped.



The last_name, sale_date, product_name, and region_name fields were selected as dimensions.

Click the Next button to continue.

Next the window shown below appears in which you can apply a style to the cube.

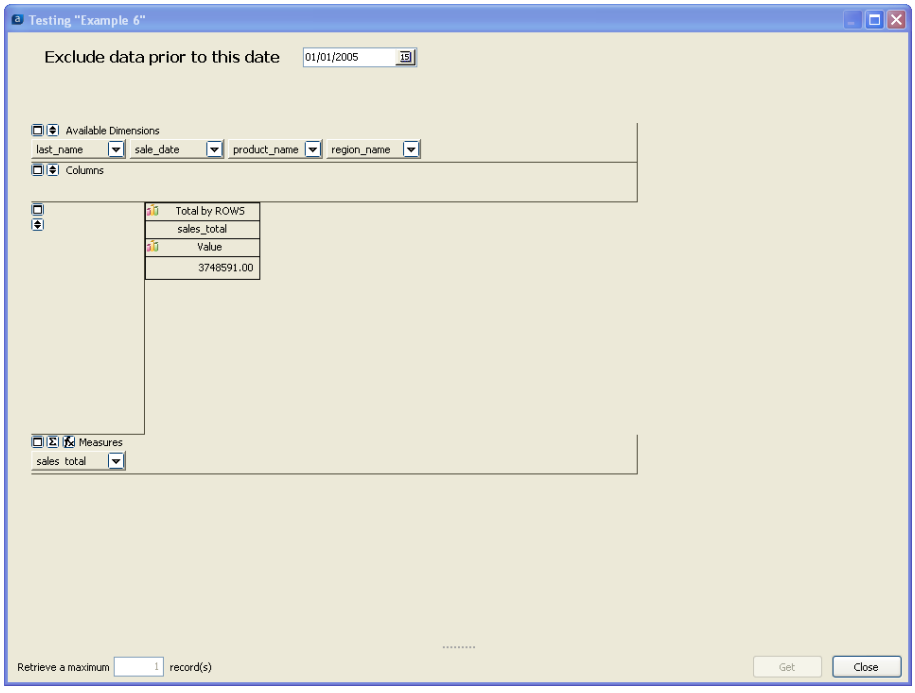


Applying a style

In this example, we will not use a pre-defined style, therefore click the Finish button.

Commit and Test

You are then brought back to the main screen of the DataBlock Designer. Click the Commit button to save your work, then click the green Test arrow to test your work. Enter the desired date into the date object. The results are shown in the figure below.



The OLAP cube

You can now manipulate the measures and dimensions to obtain the view of data that you desire.

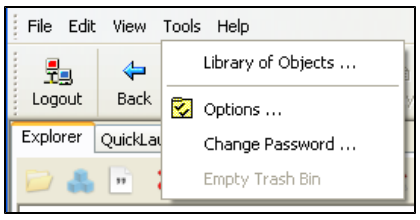
The figure below shows the effect of adding the region_name as a row and product_name as a column.

Viewing the data

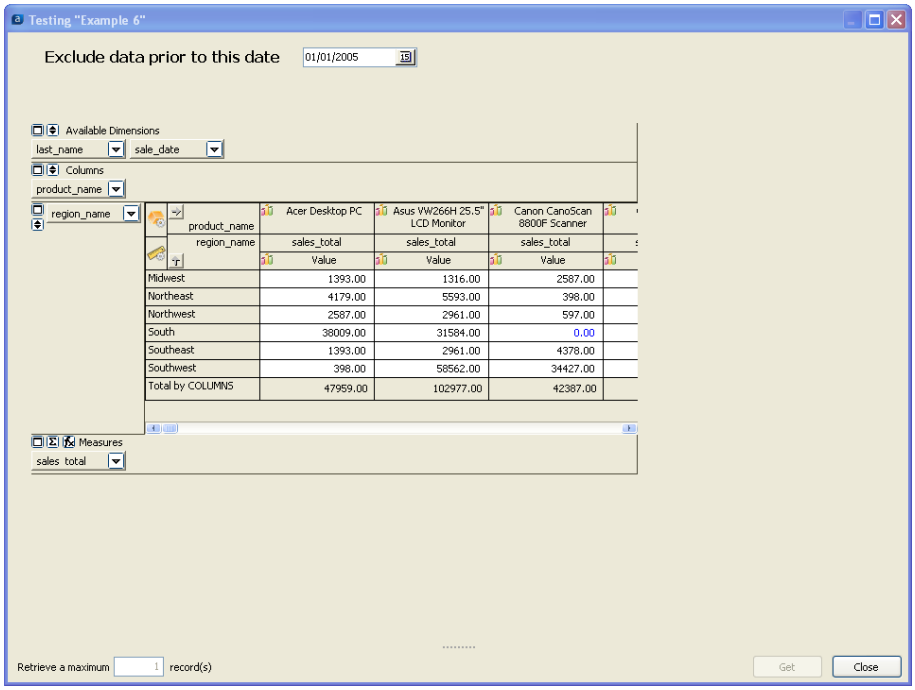
Experiment by moving the Available Dimensions into rows and columns to obtain different views of the data.

Styles

Styles are created within the Library of Objects Manager located under the Main Argos Interface under the Tools menu. Use of styles allows you to customize colors and fonts.



Library of Objects selection



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

The OLAP cube after moving dimensions into rows and columns

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; however explanation of the options is outside the scope of this document. Explanations of how to manipulate the cube further are explained in the Argos Report Viewer Guide and in the In-Product Help.

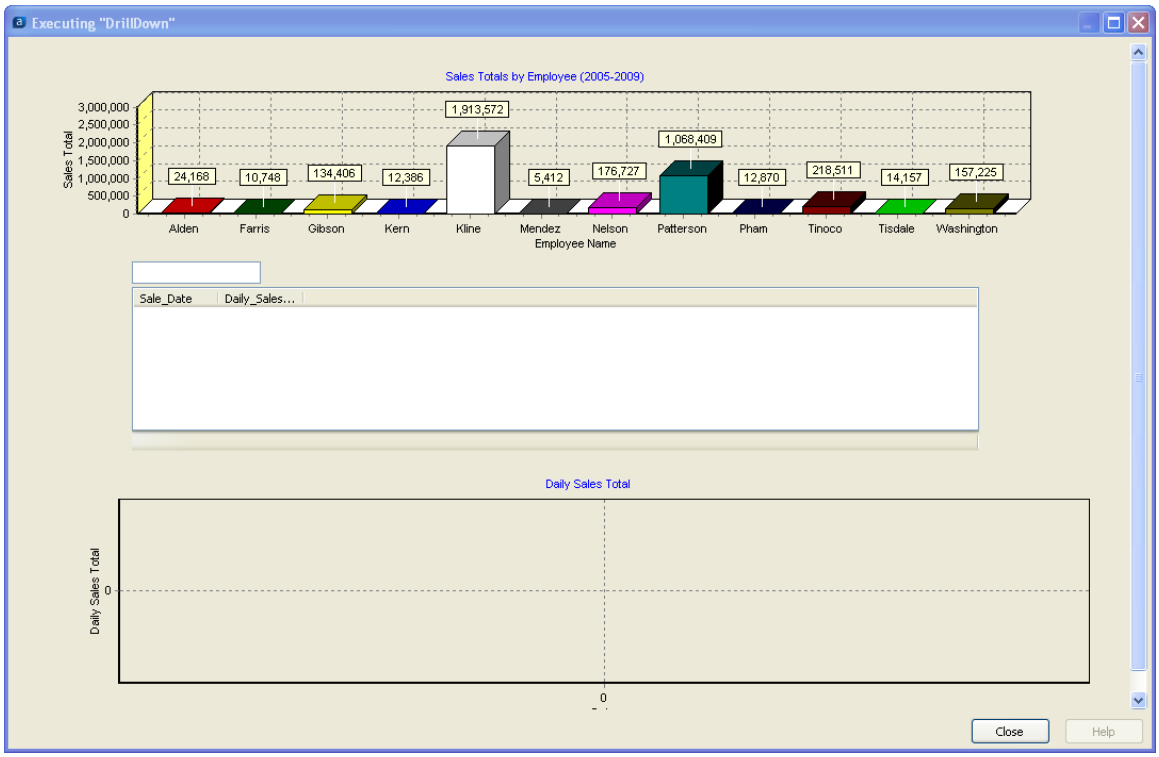
Drilling Through Reports to Obtain Greater Detail

Example 7

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.

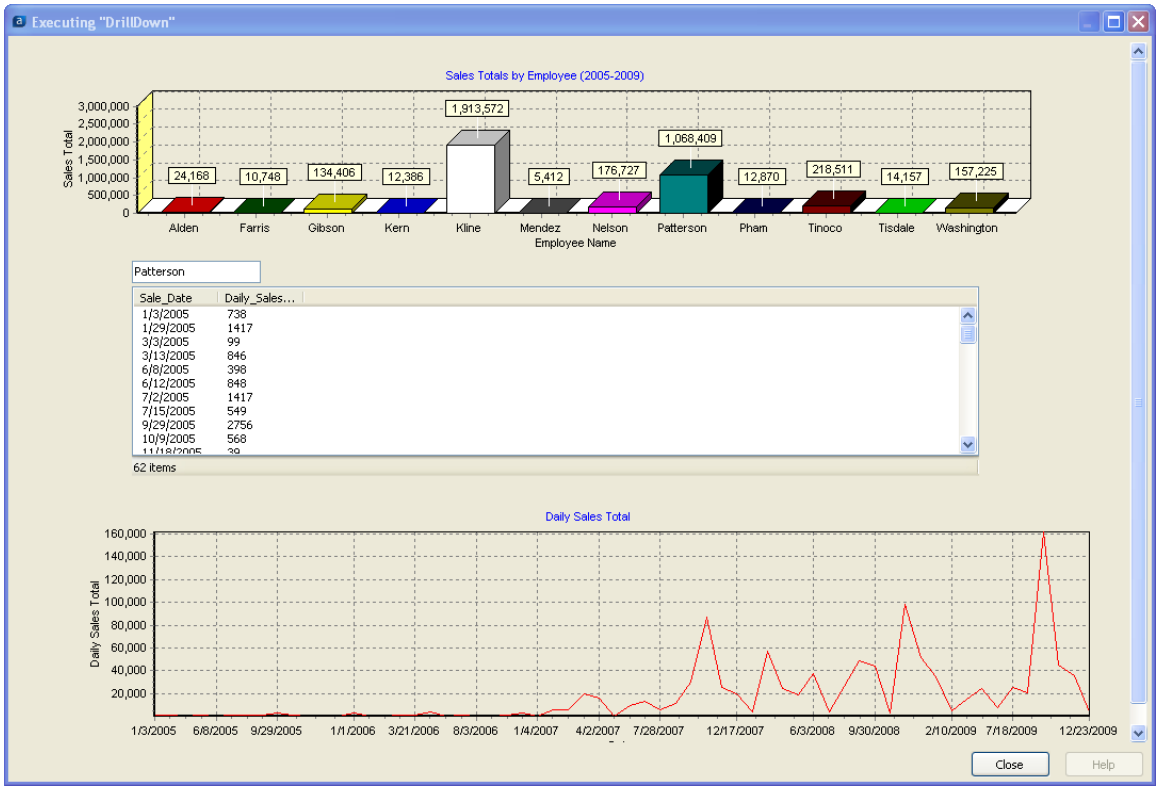
- The bar chart at the top displays a summary of sales for each employee from 2005 – 2009.
- Clicking any of the bars (employees) will drill down and display daily sales details for the selected employee in the multi-column list box and in the line chart at the bottom.

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



The Dashboard prior to selecting an Employee

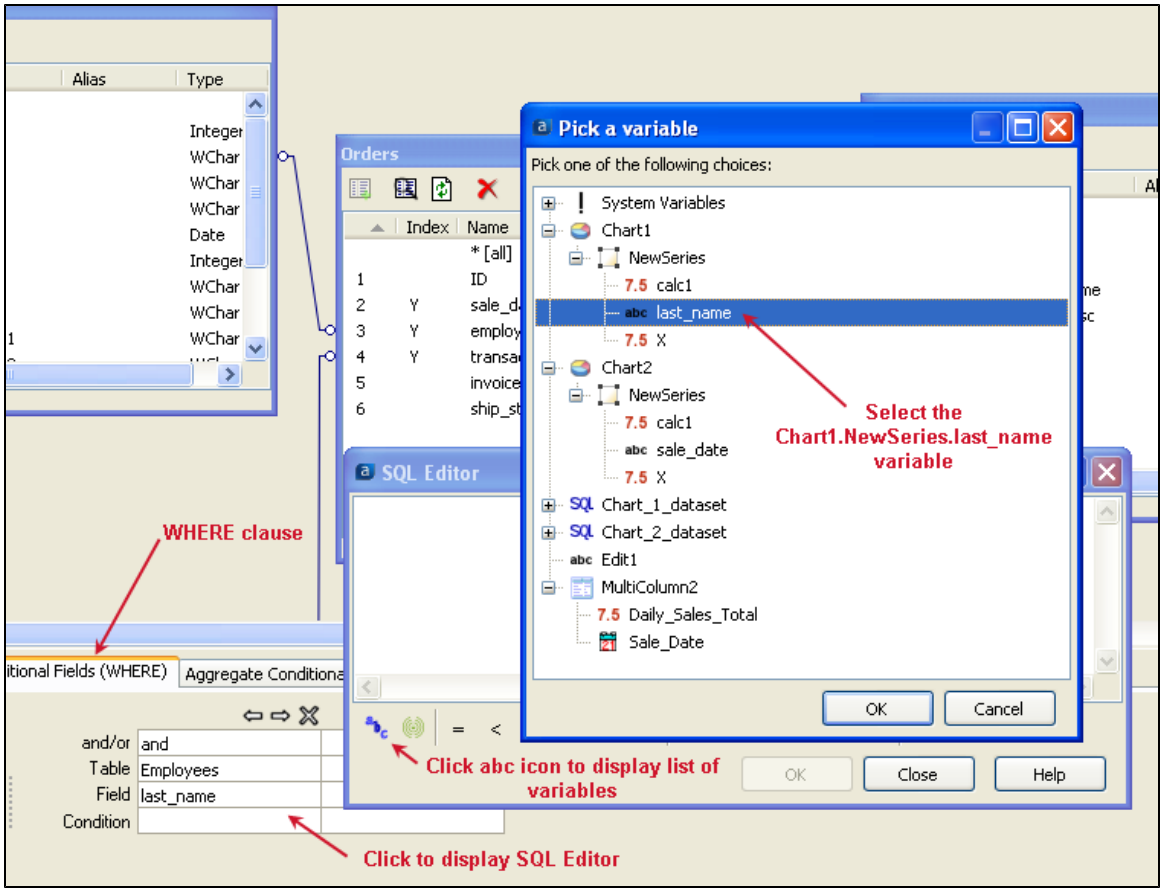
For example, selecting employee Patterson displays the detail information in the following figure:



The Dashboard report after selecting an employee

The procedure used to create the DataBlock for this example follows:

- A bar chart was created at the top of the report area. The chart object is named Chart1 with a dataset series named NewSeries. Variables calc1 (containing the totals from 2005-2009) and last_name are used in the SQL query that creates the dataset. The figure below shows the variables used by each of the objects in this DataBlock.
- A multi-column list box was created at the middle of the report area. The object is named MultiColumn2 with variables Sale_Date and Daily_Sales_Total. A query was created which calculates the sum of daily sales for the employee selected. The key step in the list box query is to create a WHERE clause which links the last_name variable in this query to the employee selected in the bar chart (the **Chart1.NewSeries.last_name** variable). The actions required to add the WHERE clause to the list box query are shown in the figure below. When the report is executed and an employee is selected on the bar chart, the Chart1.NewSeries.last_name variable is populated with the employee last name, then other objects referencing this variable in a WHERE clause will be updated.
- The same procedure is used to add the WHERE clause to the query for the line chart at the bottom of the report area.



Including the Chart1.NewSeries.last_name variable in the WHERE clause

Summary

As can be seen, 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.

You can save this as a Dashboard.

Advanced Query Techniques

This section describes various advanced query techniques that are available including Summing and Grouping of database fields, Free Type Report Query, Scalar Sub Query, Correlated Sub Query, Non-Correlated Sub Query, and Unions.

Summing and Grouping

Example 8

Introduction

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.

.

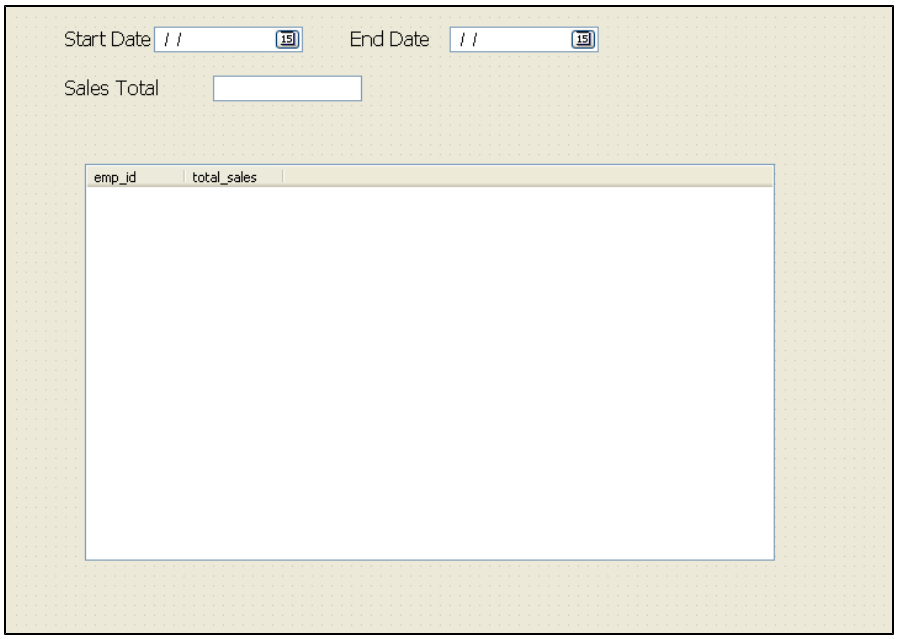
New SQL Statements in this example:

GROUP BY, HAVING, SQL AGGREGATE functions.

Create the Form

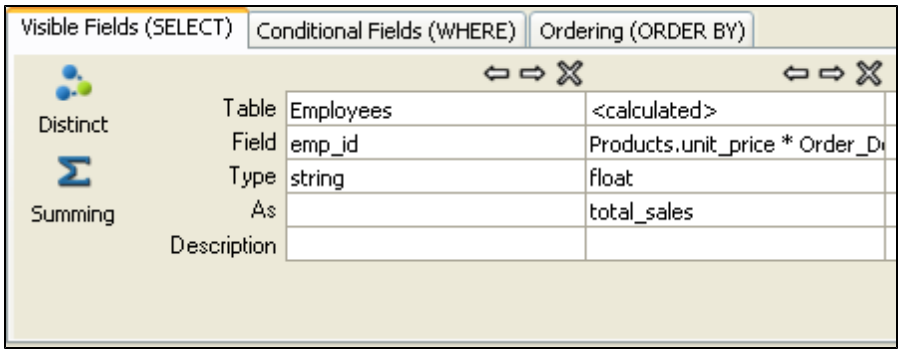
The input required from the user for this report will be a date range, and a sales total threshold that will be used to filter the query.

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.



The Form showing the 3 input selections

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).

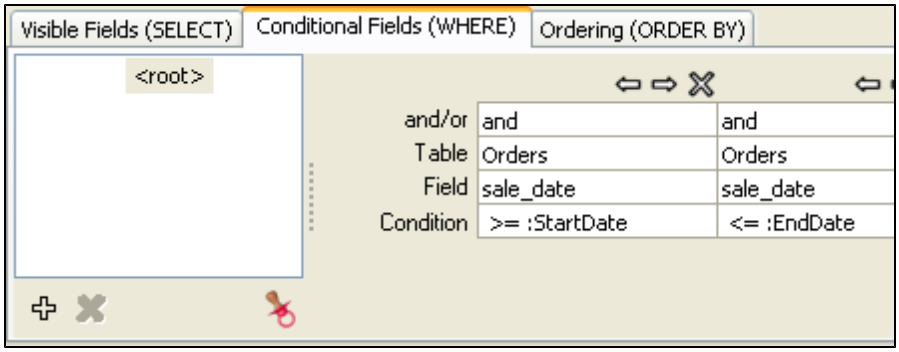


Entering data into the SELECT tab

SQL Aggregate Functions

Avg, Count, Max, Min, Sum aggregate SQL functions can be selected.

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

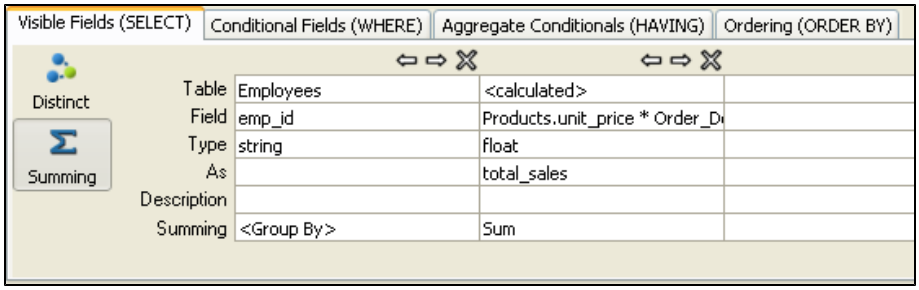


Entering data into the WHERE tab

Select GROUP BY and SUM fields

To sum the sales records and group them by employee, click the "Summing" icon which displays an additional row titled "Summing" as shown below.

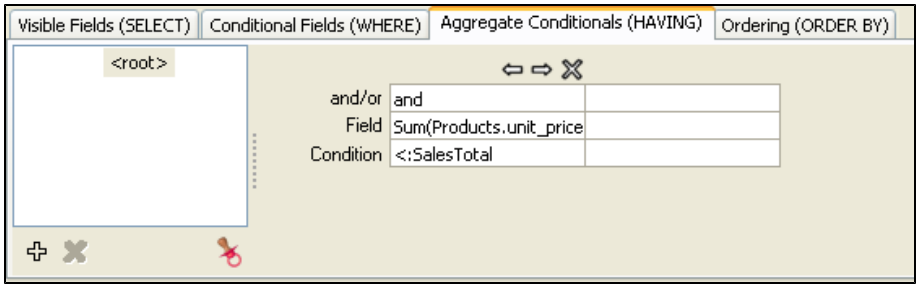
Under the Employees column, select <Group By> as shown below since the results are to be grouped by the Employee. Then select "Sum" for the calculated field since this data is to be summed. These selections are used to create the GROUP BY statement.



Entering data into the Summing field

Select 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.



Entering data into the HAVING tab

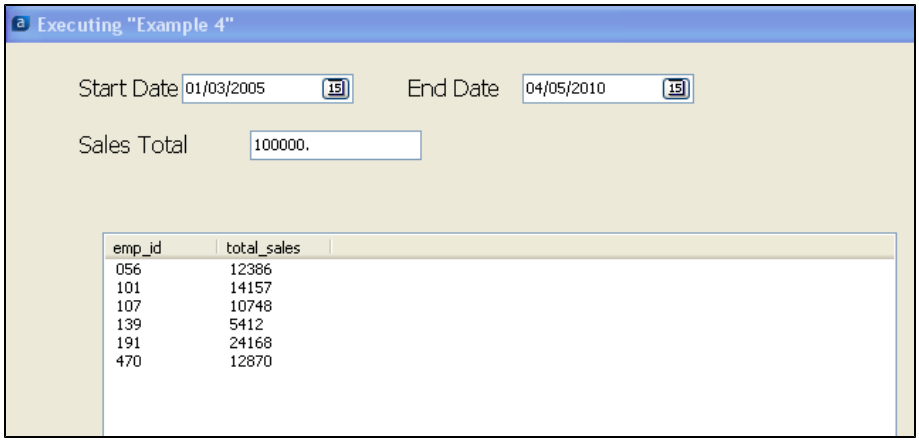
The final SQL:

```
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.

Summary

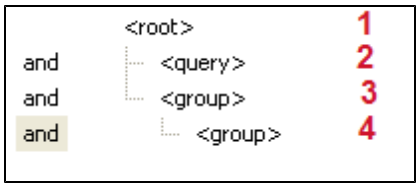
Executing the Dashboard report produces the following which lists sales employees with sales less than $1,000,000 between 1/3/2005 and 4/5/2010.



The Dashboard

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)))

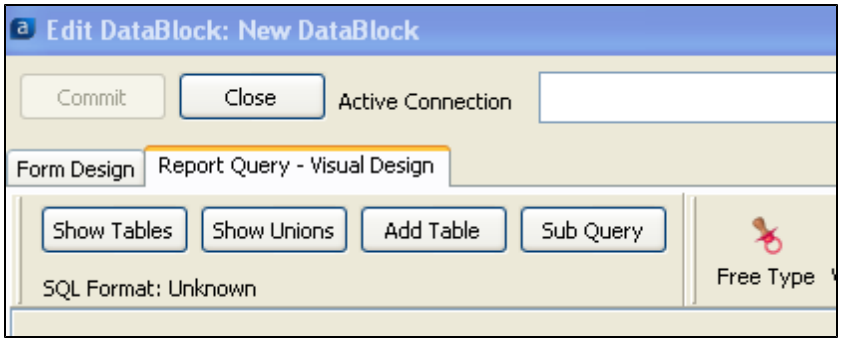
If the Summing fields are no longer desired, make sure to click the Summing icon to remove the row with the Summing fields. Otherwise Argos will expect entries in the Summing fields.

Free Type Report Query

Example 9

Introduction

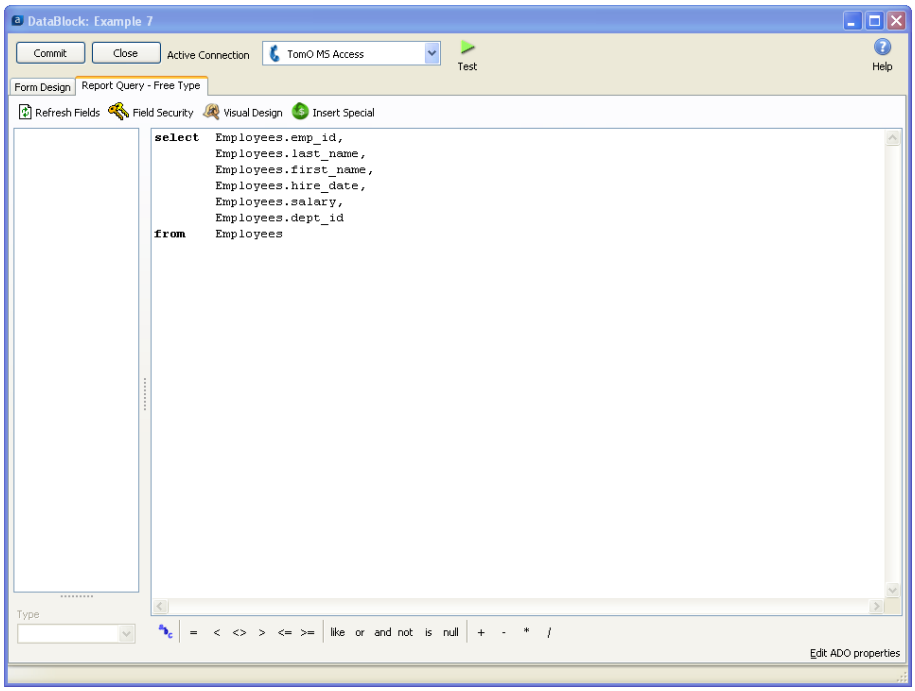
Previous examples demonstrated how to use Argos tools to visually create SQL queries. There are times when you already have an SQL query that you would like to enter into Argos without having to type in the query manually. This is possible using the Free Type SQL Editor which exists under the Report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



The Free Type icon within the Report Query –Design tab

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.



The Free Type Editor

Free Type/Visual Design toggle

After selecting the Free Type query by clicking this icon, the icon changes to the hardhat/hammer (Visual Design) icon. If you subsequently choose to create your query within the Visual Designer, click the hardhat/hammer icon to launch the Visual Designer (however the contents within the Free Type query will then be lost).

Type (lower left)

Allows you to highlight a field and change the data type from the drop down list.

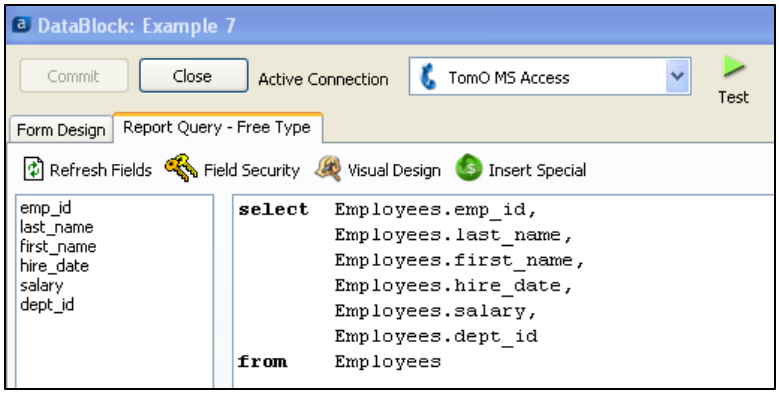
Clicking this icon displays a list of System Variables that can be added to your query.

Operators

The operators at the bottom of the window allow you to add logical operations to your query.

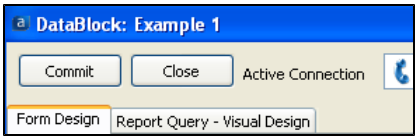
Validate the Query

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. You can click the Green test button to run the query and verify that it works properly in Argos.



Refresh Fields icon

The Commit and Test buttons

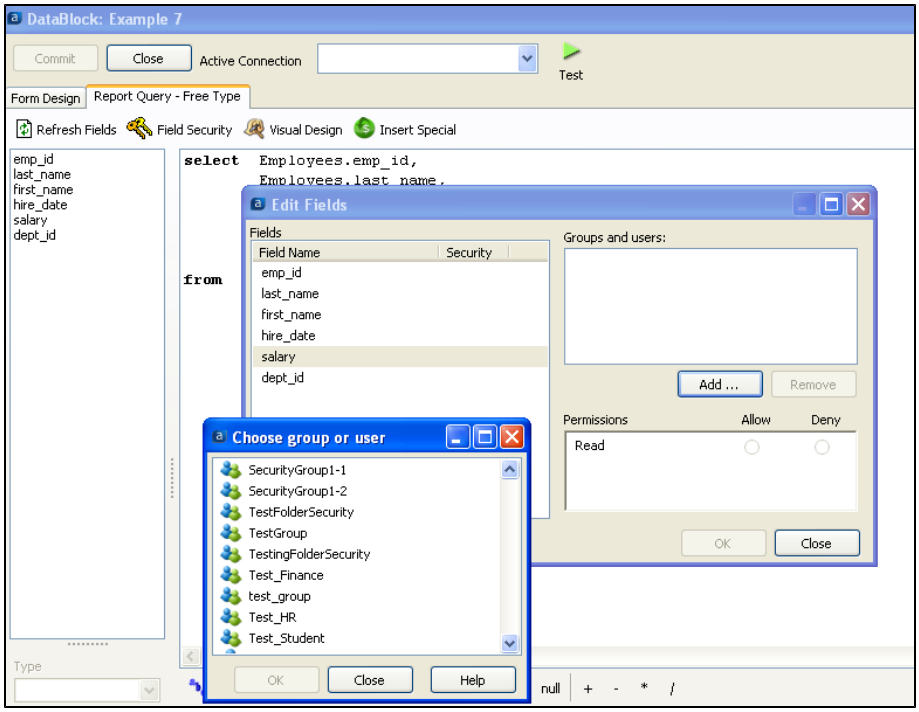


Remember to Commit (Save) your work and test the query as you make changes.

Apply Field Security

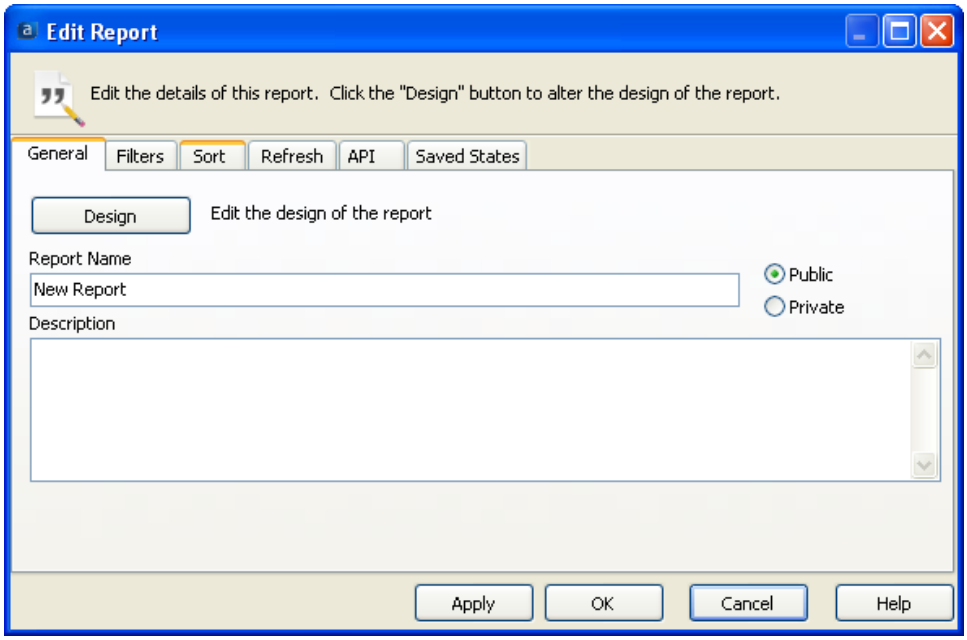
If you wish to apply security to individual fields, this can be accomplished by clicking the Field Security icon. The following dialog box will appear in which you can apply Allow/Deny privileges to Argos users or groups for each selected field. The figure below shows that security is to be applied to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

Field Security icon



Filter and Sort the Query in CSV, Banded, and Extract Reports

When creating CSV, Banded, and Extract Reports, the person creating the report can make use of the Filters and Sort Tabs (see figure below) to modify the query results obtained from the DataBlock when the report is being executed.

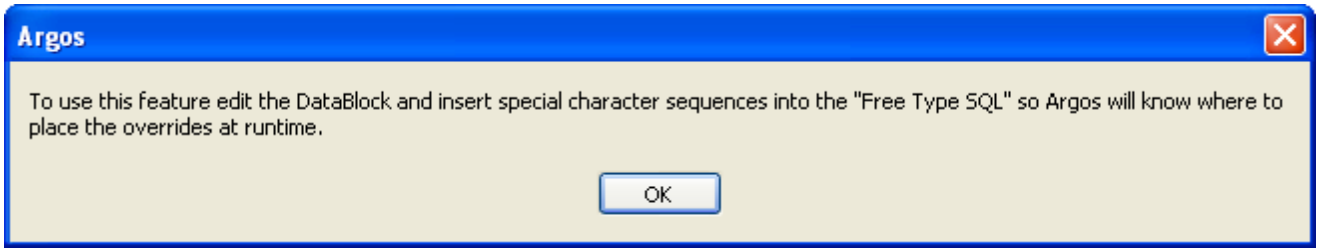


The Filters and Sort tabs used for Report Design

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.



Reminder to add special characters when using Free Type Query

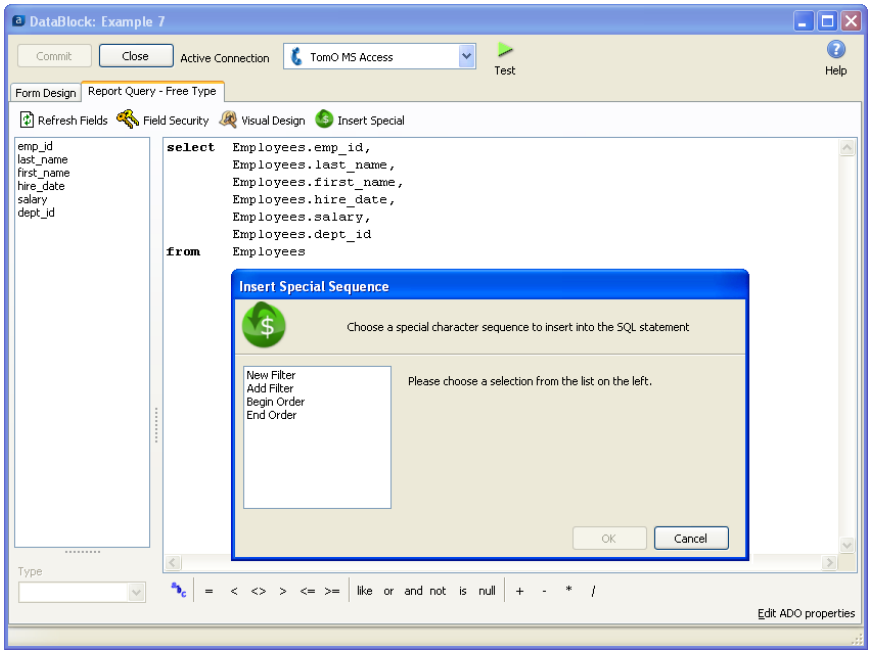
Adding special characters for filtering in CSV, Banded, and Extract Reports

If you feel that your CSV, Banded, or Extract 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 CSV/Banded/Extract report, therefore the WHERE clause needs to be added to the end of the query. Therefore, 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.

Insert Special icon

Select "New Filter", then click the OK button.



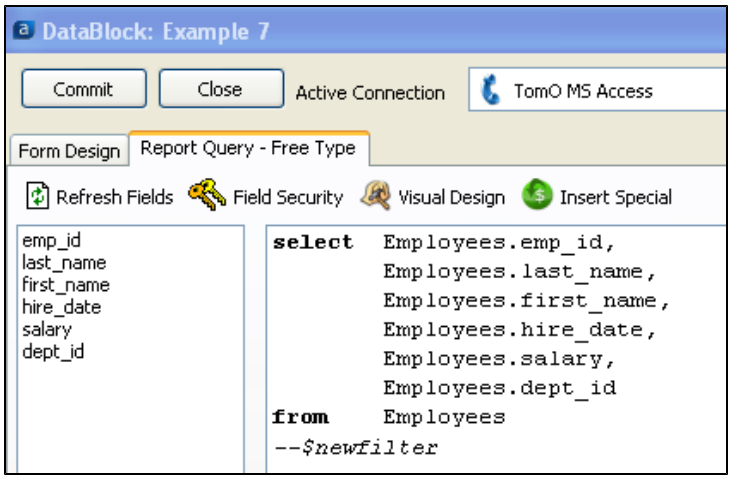
Adding special characters for filtering

The string "--$newfilter" is inserted at the end of the query. This string serves as a placeholder such that when the Filter tab is selected when creating the CSV/Banded /Extract report, the WHERE clause that the Filter creates will replace the -$newfilter string. Note the location of the special character string in the figure below.

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.

$newfilter

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 CSV/Banded/Extract Report design.

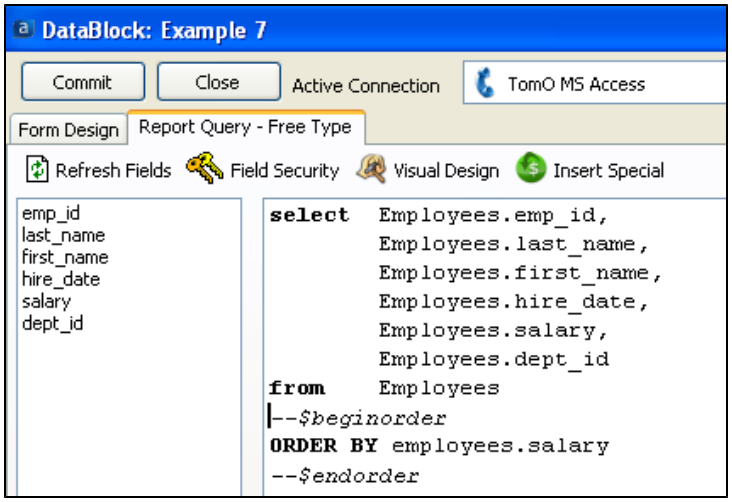


The Free Type query after adding special characters for filtering

You can now use the Filter tab when designing your CSV, Banded, or Extract Report.

Adding special characters for Sorting in CSV,Banded, and Extract 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 as shown below. You can use the **Insert Special** button to add the statements, or type them in manually.



The Free Type Query after adding special characters for sorting

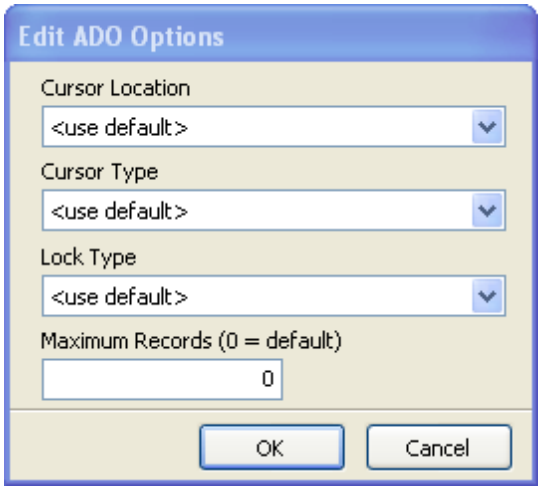
Place the cursor at the desired location, then insert the special character strings and the ORDER BY statement. SQL statements between *--$beginorder* and *--$endorder* will be replaced by the ORDER BY statement created as a result of the sort criteria in the report design.

$beginorder and $endorder

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 by the CSV/Banded/Extract Report design.

Edit ADO Properties

This option allows you to change the way Argos communicates with the database. Consult your MAPS Administrator before changing ADO properties.



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 Sub Query

Example 10

Introduction

Some databases allow entire SQL statements (called a Sub Query) to be used as a field in the SELECT clause of another query. As each row is processed in the "main" query, the Sub Query is called to fill in the value for that particular field. These Sub Queries 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 Sub Query (highlighted) is being used to return the latest PO date for an individual employee (if it exists). In other words, the Sub Query 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'
```

The Sub Query must return a single value

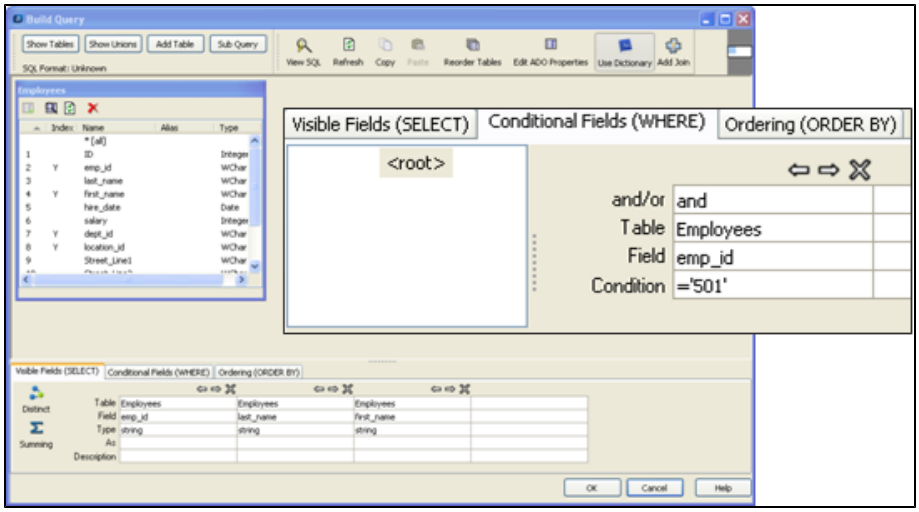
Note the use of the MAX function which will return one value.

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/Sub Query to find the latest Purchase Order created by Employee_ID = 501 follows.

Create the Main Query

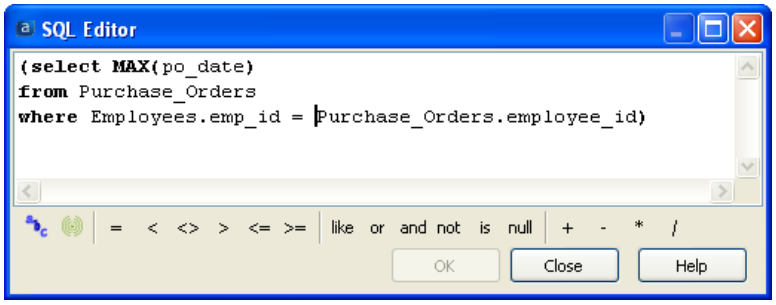
Begin by creating the non-highlighted sections of the SQL for the main query shown above.



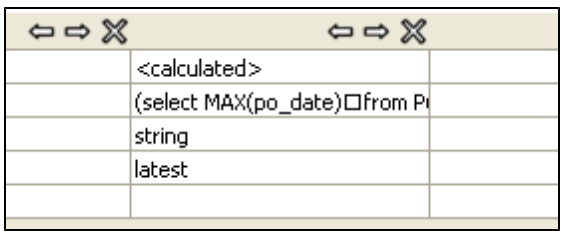
Create the Sub Query

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 Sub Query (preceded and followed by parentheses) as shown below.

Remember to enclose the subquery with parentheses.



Give the calculated field an alias as shown below.



Entering the alias

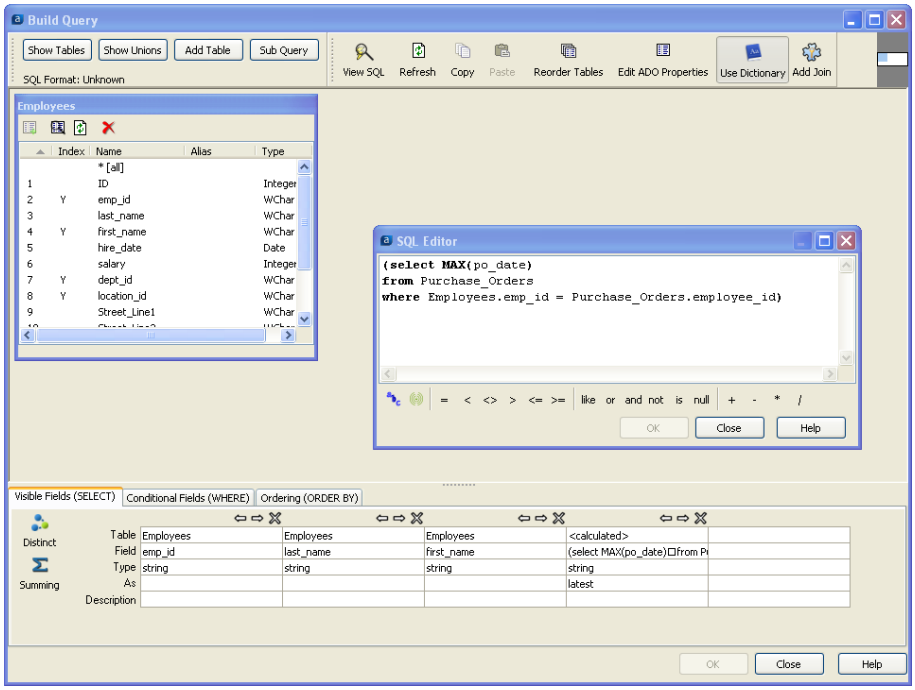
Execute the query

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

Summary

The figure below shows the query that selects several fields for employee ID 501 from the Employees table, a calculated field that returns the latest PO date for the employee from the Purchase_Orders table. Notice how the scalar Sub Query is linked to the main query. If the PO date doesn't exist in the Purchase_Orders table, the main query row will still be returned, but the "latest" filed will be blank.



The final Query/Sub Query

Correlated Sub Query

Example 11

Introduction

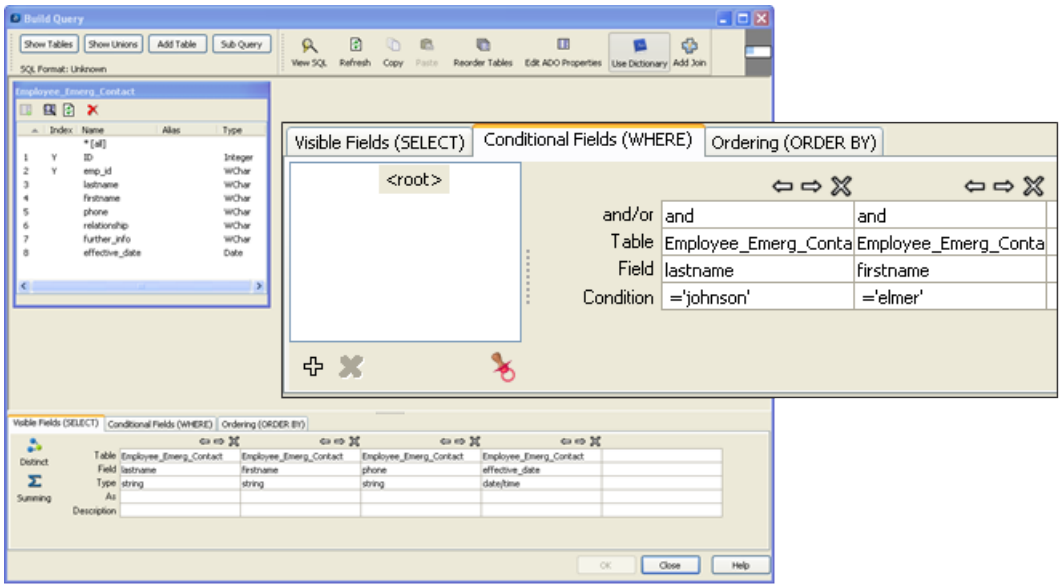
Some queries require that the output be limited by the results of a Sub Query. For example, a table may contain a field that includes an effective date and you may want your query to limit the results to the record that contains the latest effective date.

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) inthe Employees Table.

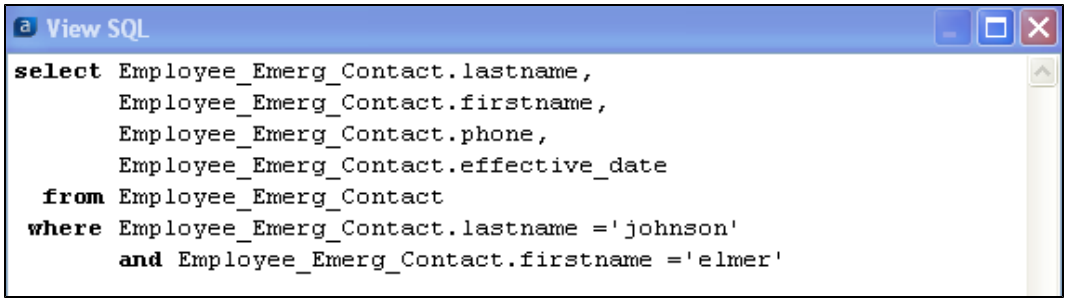
Create the Main Query

Create the Main Query shown below using the Employee_Emerg_Contact table containing the WHERE clause to find records for Elmer Johnson. Note that this query can return multiple records for that individual.



The Main Query shown within the Build Query dialog box

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.



The Main Query

A Sub Query is needed (shown in the figure below) to return the record with the latest effective date. The Sub Query is the portion of the query below the line containing ….firstname='elmer'.

View SQL

```
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 )
```

The Sub Query added to the Main Query

The Sub Query returns one value to be used within the WHERE clause.

Notice that the Sub Query 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 Sub Query. These types of Sub Queries are called "correlated" as the results of the Main Query are dependent on the results of the Sub Query. Also notice that in this example, the Main Query and the Sub Query 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.

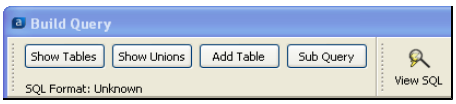
Recall that the Sub Query is to provide a single value for Employee_Emerg_ contact.effective_date in the WHERE clause. That is, a multiple lines of SQL are used to provide a value to be used in the WHERE clause.

Create the Sub Query

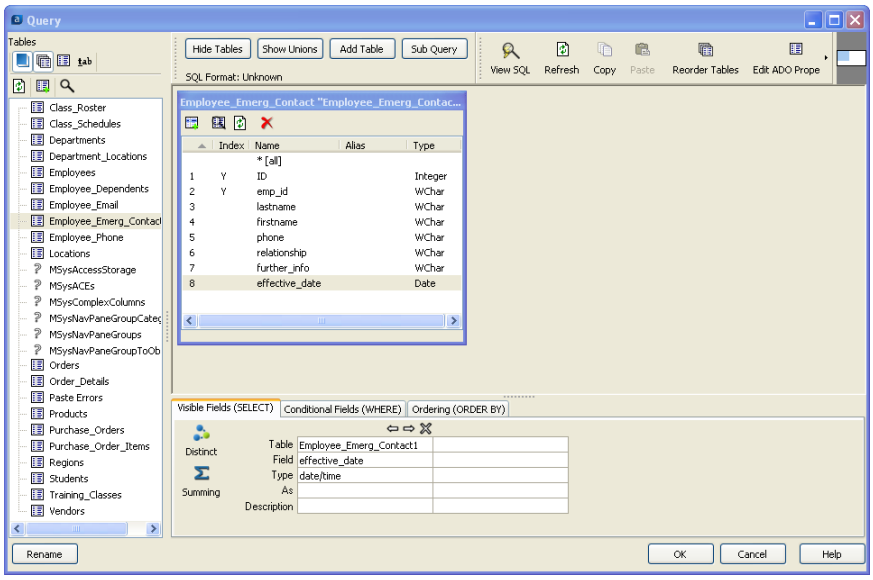
The steps for creating the Sub Query shown above will now be discussed. Within the Build Query dialog box, click the Sub Query button. The Query dialog box will be displayed again such that you can use it to create the Sub Query. Click the Show Tables button and select the Employee_Emerg_Contact table. Double-click the effective_date field which moves the field under the SELECT tab as shown below.

Notice that in the Sub Query, the table name is called Employee_Emerg_Contact1 because the main and sub queries are using the same table (as discussed above).

Location of the Sub Query button

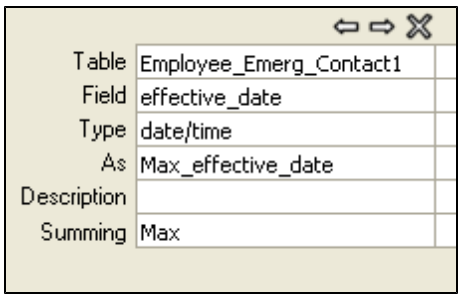


The location of the Sub Query Button



Selecting the table for the Sub Query

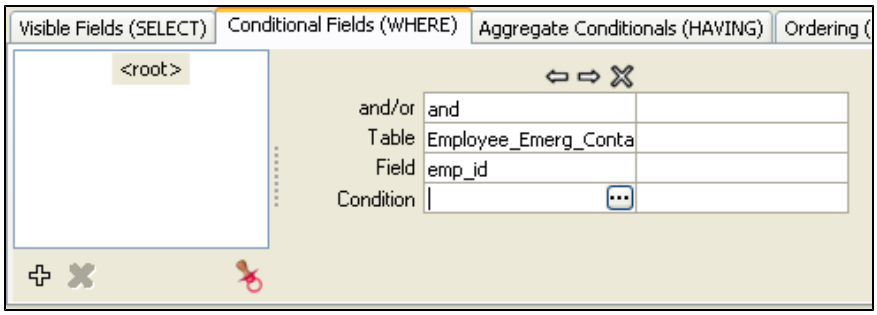
Click the "Summing" icon and select "Max" since the Sub Query is to find the latest effective date.



Specifying the latest effective date

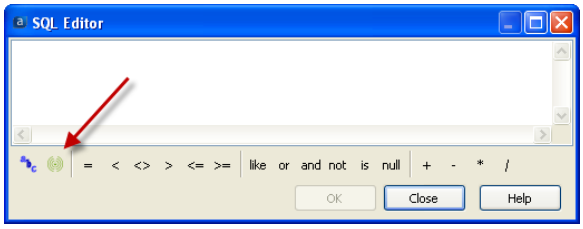
Next create a WHERE statement to link the Main Query to the Sub Query using the employee ID as the common field. When using the editor to create the WHERE statement, since the emp_id field exists in both Employee_Emerg_Contact and Employee_Emerg_Contact1 tables, thus both must be referenced.

Select the Employee_Emerg_Contact1 table as shown below.

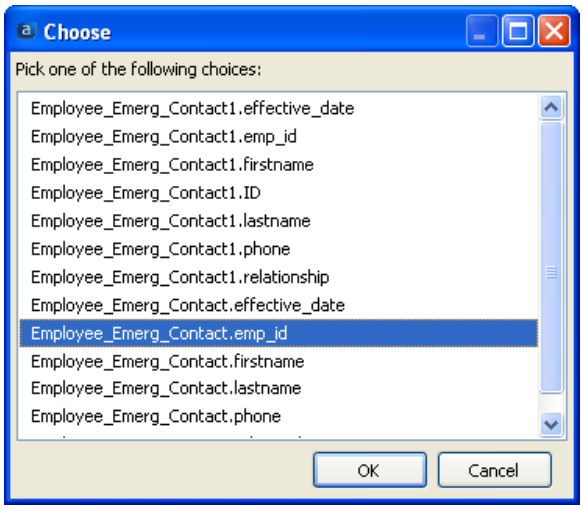


Creating the WHERE clause in the Sub Query

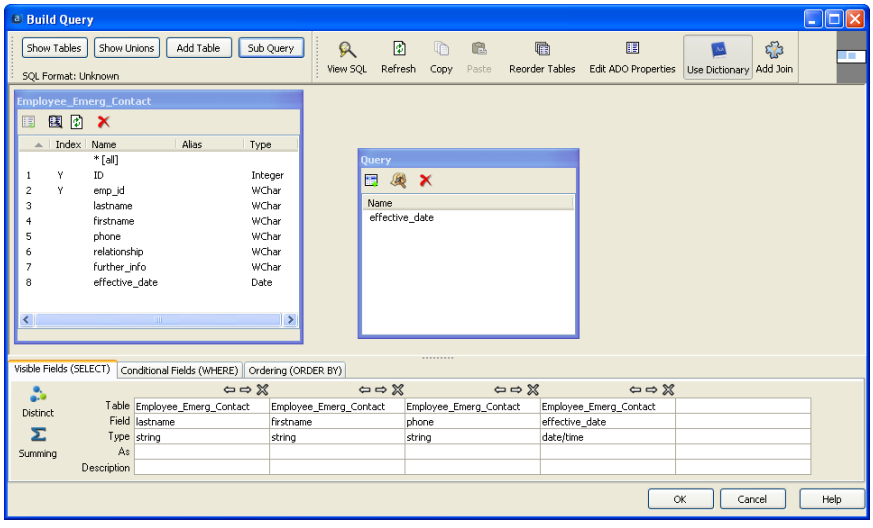
After clicking the ellipsis in the Condition field the SQL Editor is displayed such that you can locate the other table.



Click the green circular 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 Sub Query.



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



The Sub Query shown within the Build Query dialog box

The Sub Query is shown below.

View SQL

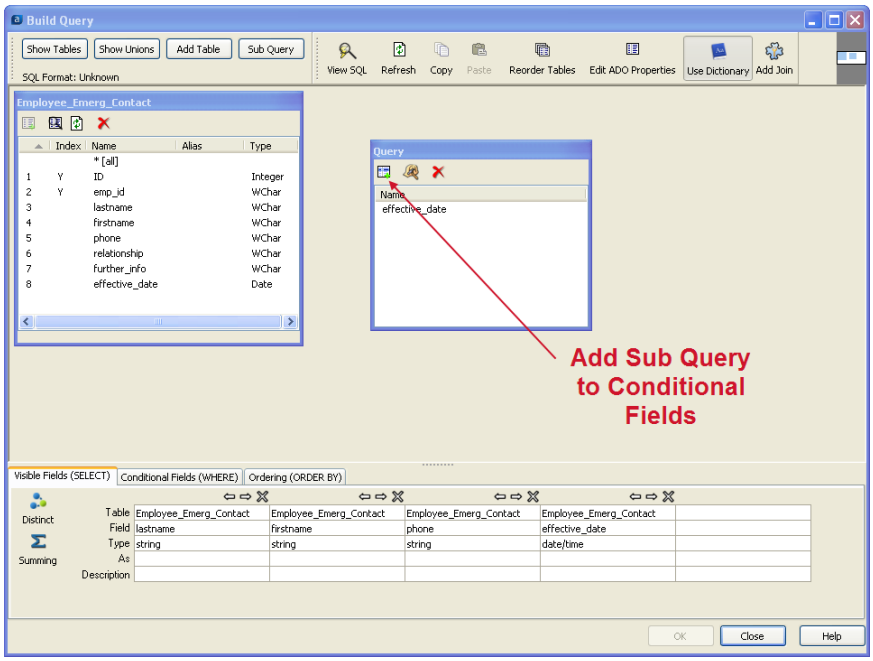
```
select Max( Employee_Emerg_Contact1.effective_date ) as Max_effective_date
  from Employee_Emerg_Contact Employee_Emerg_Contact1
 where Employee_Emerg_Contact1.emp_id =Employee_Emerg_Contact.emp_id
```

The Sub Query

Place the Sub Query into the Main Query

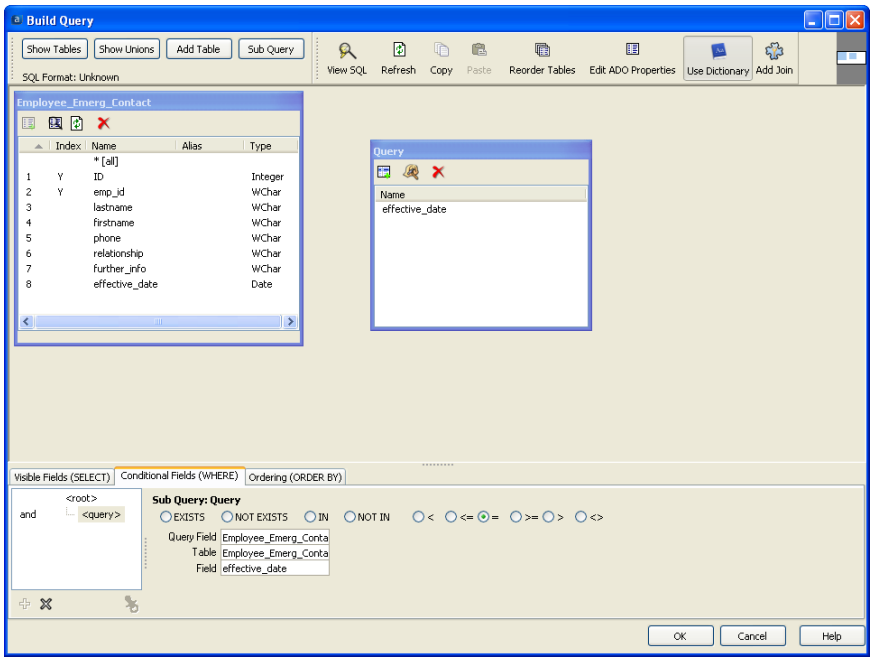
Now that the Main Query and Sub Query have been built, the Sub Query needs to be placed into the appropriate section of the Main Query. To perform this linkage, click the "Add this Sub Query to the Conditional Fields Tree" button within the Sub Query Window.

Add this Sub Query to the Conditional Fields Tree icon"



Positioning the Sub Query into the Main Query

The Conditional Fields (WHERE) tab is displayed at the bottom of the window where you can use the appropriate aggregate function to link the Sub Query 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 Sub Query. Make sure to choose the effective_date field from the Employee_Emerg_Contact table in the Main Query.



The aggregate function linking the Sub Query to the Main Query

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

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

View SQL

```
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 )
```

The complete SQL

Execute the query

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

The query results displaying the latest effective date

Non-Correlated Sub Query

Example 12

Introduction

At times it is useful to create a Sub Query 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.

Non-Correlated Sub Queries are not tied to a field within the Main Query

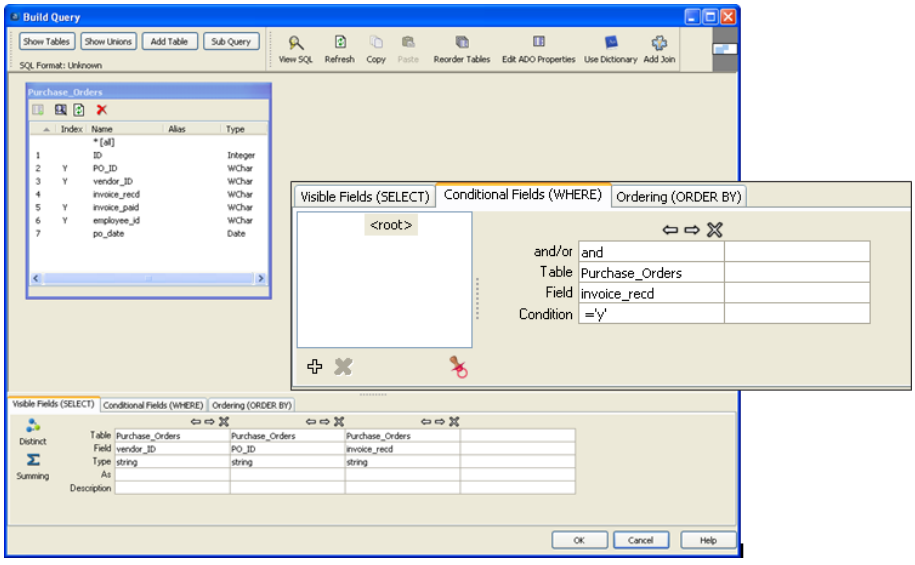
It is possible to accomplish this with some tricky join logic, but a "non-correlated" Sub Query is often easier. These types of Sub Queries are generally not tied to a particular field in the Main Query using the "=" operator, rather they typically use other operators such as IN, NOT IN, EXISTS, and NOT EXISTS. These are called non-correlated Sub Queries as the Sub Query 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 Sub Query 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.

Create the Main Query

The process is essentially the same as used for the Correlated Sub Query discussed in the previous example, thus less detail is required for this example.

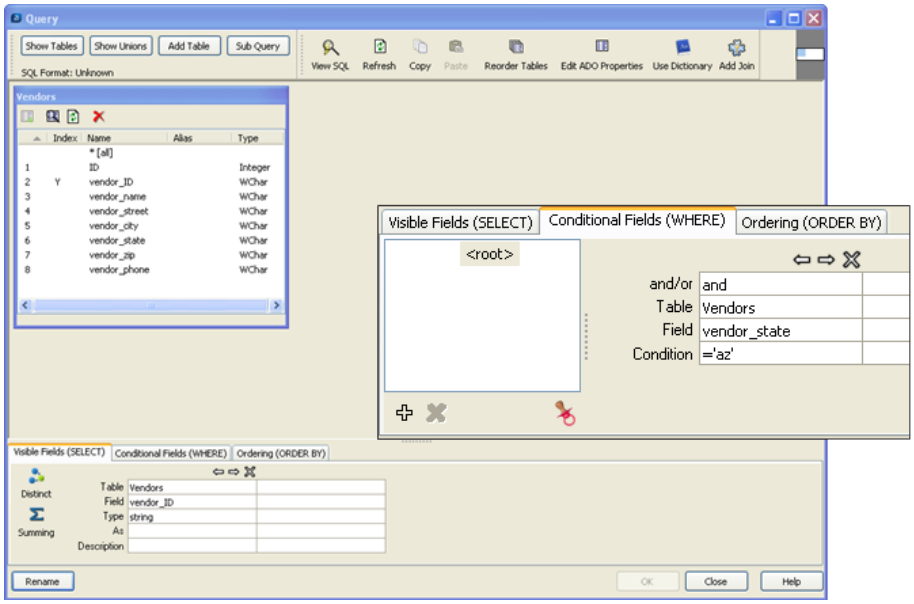
Create the Main Query to return all Purchase Orders with invoice_reqd field = "Y".



The Main Query

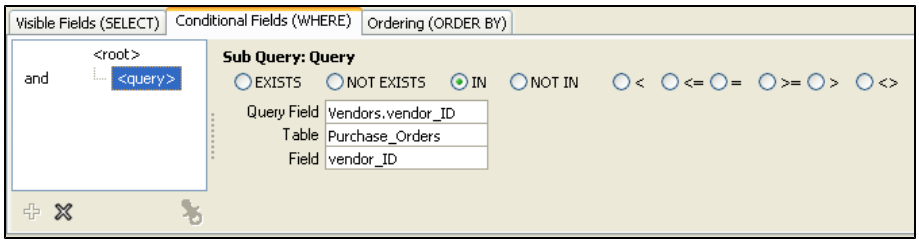
Create the Sub Query

The Sub Query 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.



Place the Sub Query into 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 Sub Query to the Main Query.



In this example, the "IN" operator is used since multiple records likely will be found.

Placing the Sub Query into the Main 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' )
```

The completed query

Execute the query

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
ven002	1072	y
ven002	1082	y
ven007	1088	y

The query results

Unions

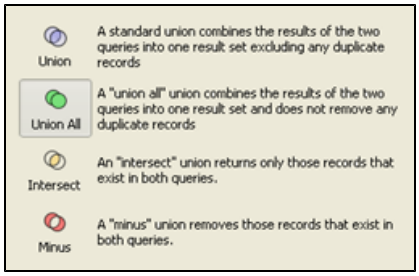
Example 13

Introduction

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.

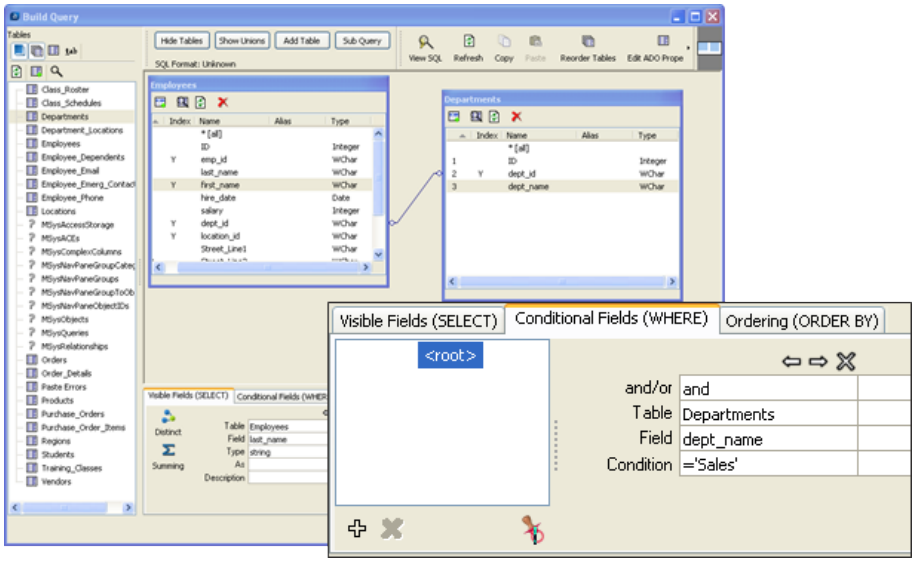
This example (using the sample database) will create a union using 3 queries to obtain information from the Employees table. The first query will obtain employees in the Sales Department. The second query will obtain employees in the Marketing Department, and the third query will obtain employees in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

Union Types available in Argos



Create 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 Main Query

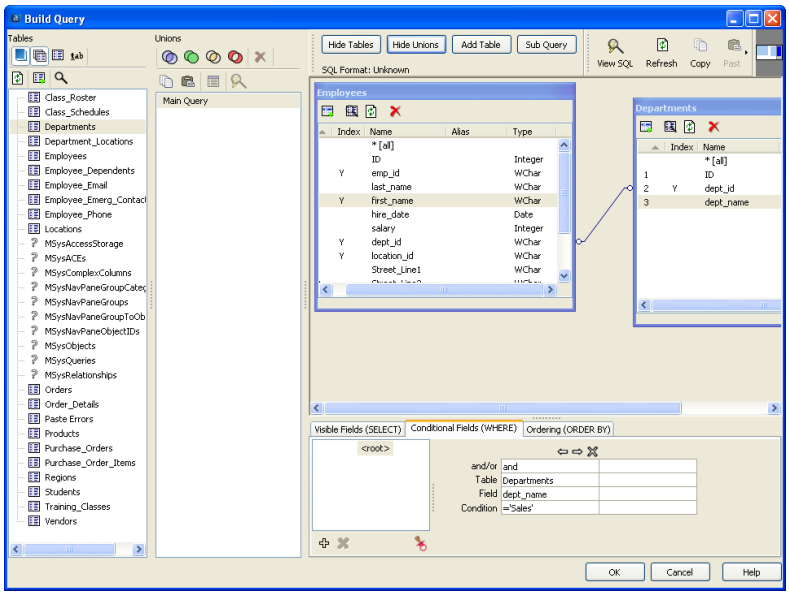
After creating the query, you can click OK to test the query, or merely move on to the next step to create the Unions.

Create 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 query created in the previous step is shown as "Main Query".

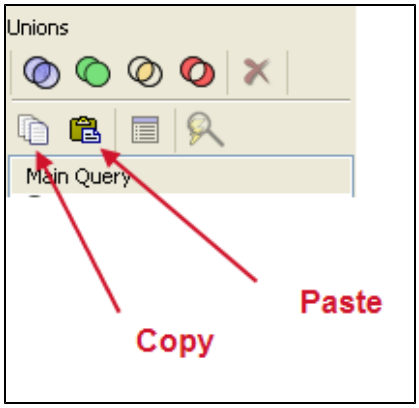
Since the Main Query will be used as the basis for the other queries, click the Copy icon (**Under the Unions pane**) to place the query into the clipboard.

Do not click the Copy on the top of the Build Query dialog box; you must use the Copy icon within the Unions pane.

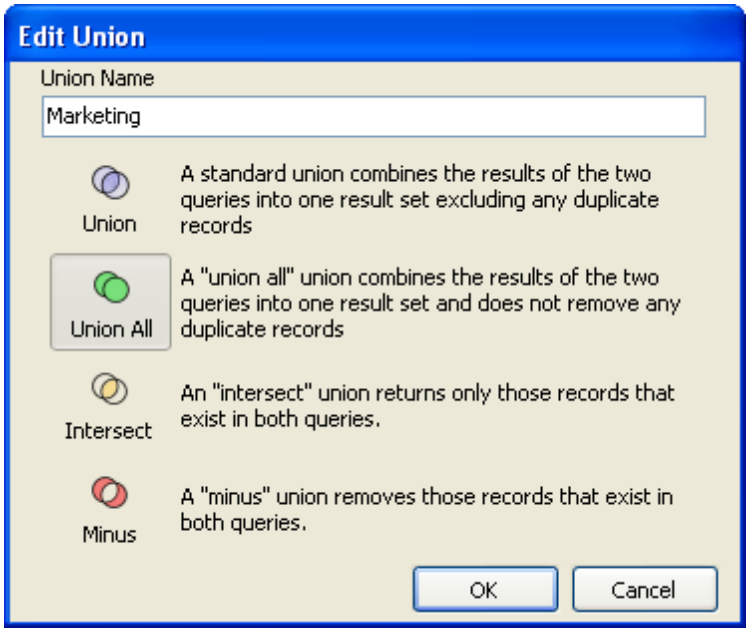


The Unions pane displayed

Use the Copy/Paste icon under the Unions Pane. Do not use the Copy/Paste icons at the top of the Visual SQL Builder



Click the "Union All" icon (the two green overlapping circles) 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.



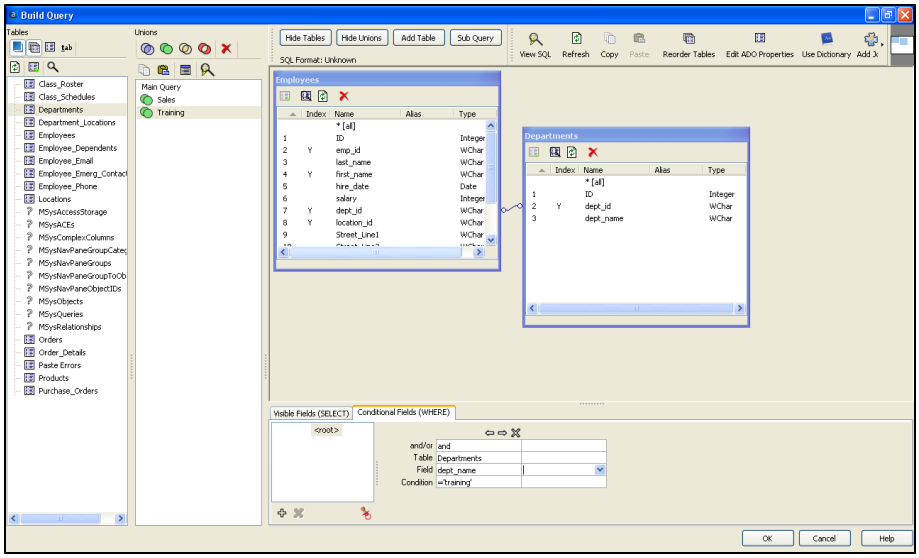
Editing the Union Name

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 overlapping green circles again to create the third query and name it "Training".

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.



The three queries shown in the Unions pane

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'
```

The final SQL

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

The query results

Summary

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

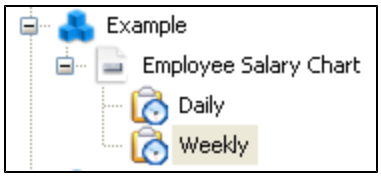
Report Scheduling

Example 14

Introduction

The DataBlock Designer has permission to schedule reports to be run at pre-determined times. Reports can be scheduled to run at specific dates, times, and periodically. All report types (Dashboard, CSV, Banded, and Extract) can be scheduled. Scheduled reports can be processed in a number of ways (printed, email, saved on MAPS server). This example will describe the process for scheduling a Banded Report, although the same process applies to all report types.

Any number of scheduled reports can be created for an existing report. The two items below the Banded Report in the figure below shows that two scheduled reports exist for the Report.



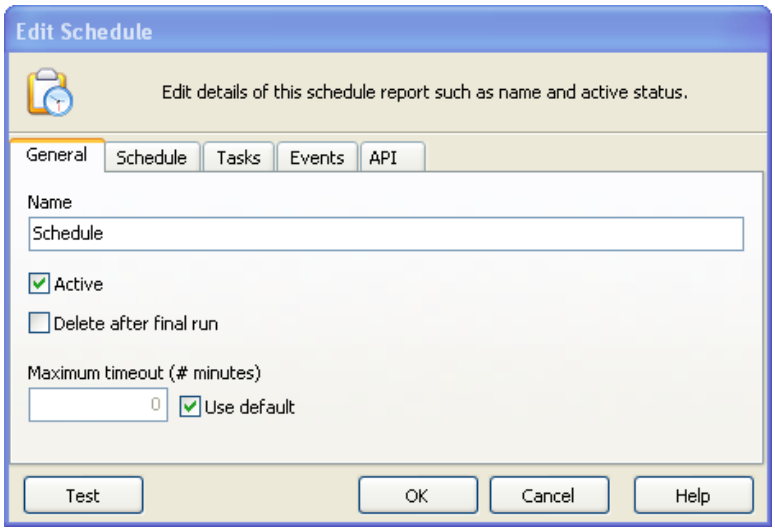
A schedule shown in the Explorer Tree

Report types that can be scheduled

All report types (Dashboard, CSV, and Banded) can be scheduled. Creation of OLAP cubes can be scheduled as well.

The General Tab

To create a schedule for a report, within the Explorer tree, highlight the report, right-click then select "Schedule". The following dialog box will appear:

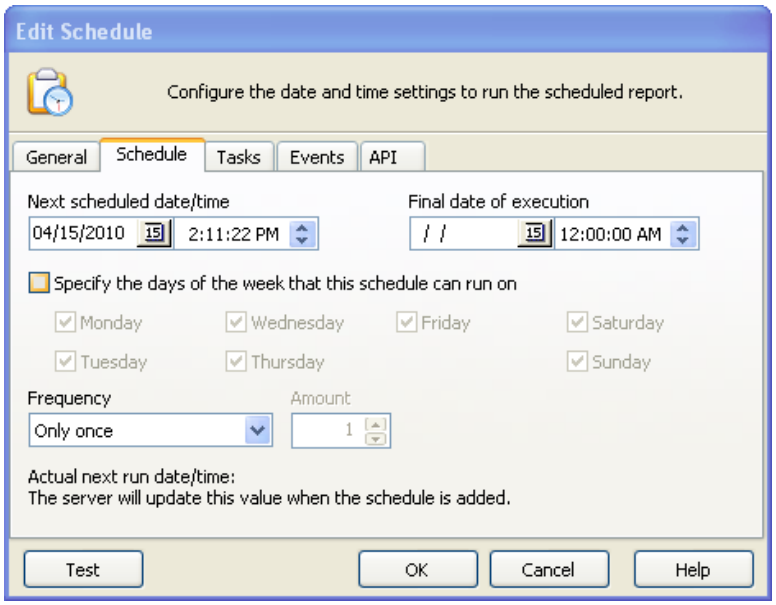


The General Tab

Change the name from "Schedule" to something more descriptive. If you create a large number of scheduled reports, use of descriptive names will be useful.

Make sure the "Active" check box is checked; otherwise the schedule will be inactive and will not take effect.

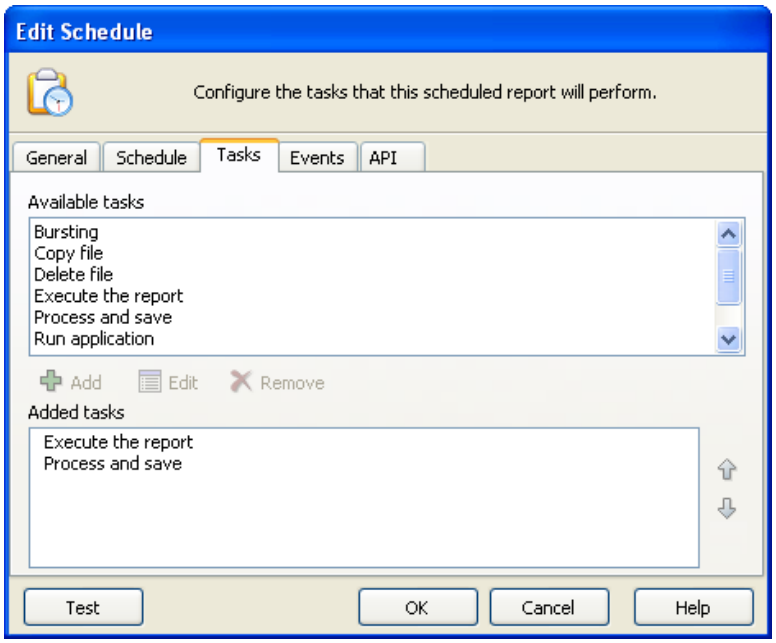
The Schedule Tab



Enter the schedule dates/times, and frequency.

The check box where the days of the week that the schedule CAN run on does not necessarily mean that the schedule will run on that day; only that it is permitted to do so.

The Tasks Tab



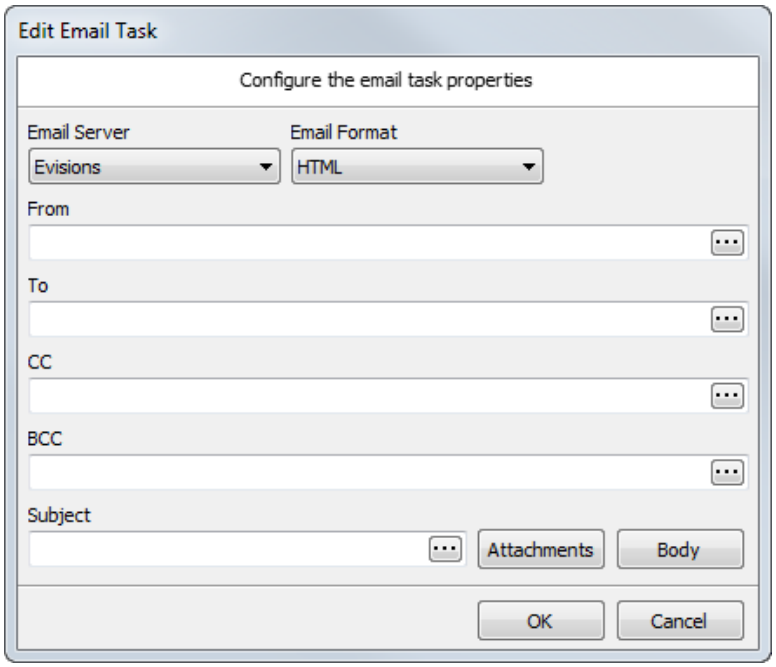
Scroll down within the Available tasks area to view the entire list of Tasks

This tab contains a list of available tasks that can be included in the scheduled report. Moving a task from the Available tasks section to the Added tasks section will cause the task to be included in the scheduled report.

The tasks will execute in the order listed. To add a task from the Available tasks, double-click the task or select the task and click the green "Add" button. After entering the details for that task, it will appear in the Added tasks list and will execute with the report.

Note: Argos automatically adds the "Execute the report" and "Process and save" tasks since they are both required for the scheduled report to execute.

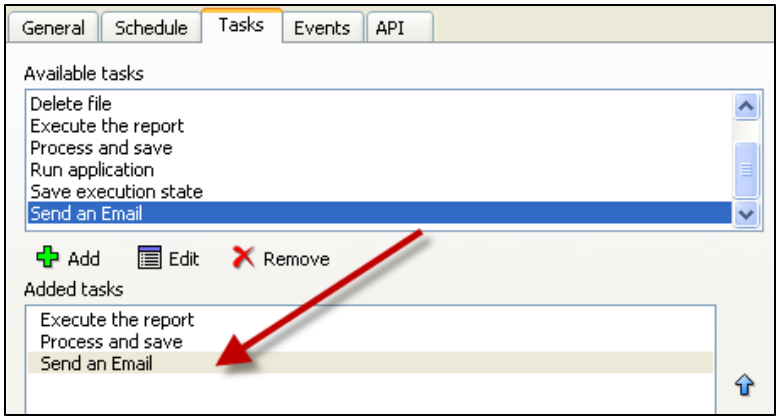
For example, to send an email as one of the tasks, double-click "Send an Email" and the following dialog box will appear in which you fill out the email properties.



Configuring email properties

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.

After clicking OK, the added tasks now include the email task as shown below.



The email task added to the list of added tasks

The same process is used when adding other available tasks to the added tasks list. Explanations for the tasks that are fairly involved are provided in sections that follow.

Available tasks:

Bursting

Allows you to burst(separate) a report into individual reports.

Copy file

Allows you to copy thereport output file to a location on the network.

Delete file

Allows you to delete a file from a location on the network.

Execute the report

Brings up the form containing input selections so you can select parameters needed for execution of the report (if necessary).

Process and save

Lets you choose the output format for the report.

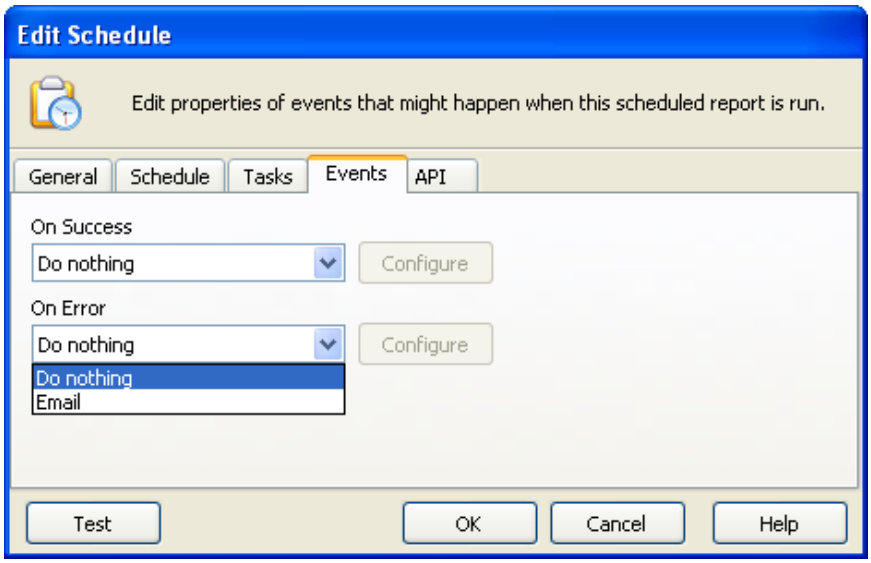
Run application

Allows you to execute an application such as a batch file.

Send an Email

Allows you to send an email containing the output of the scheduled report to users. The report will be sent in the format type selected in the Process and save task.

The Events Tab



The Events Tab allows the user to either specify which event (do nothing, or send an email) upon success or failure of a scheduled report. Selecting "Email" will make the "Configure" button active which, when selected, allows you to configure email properties.

The API Tab

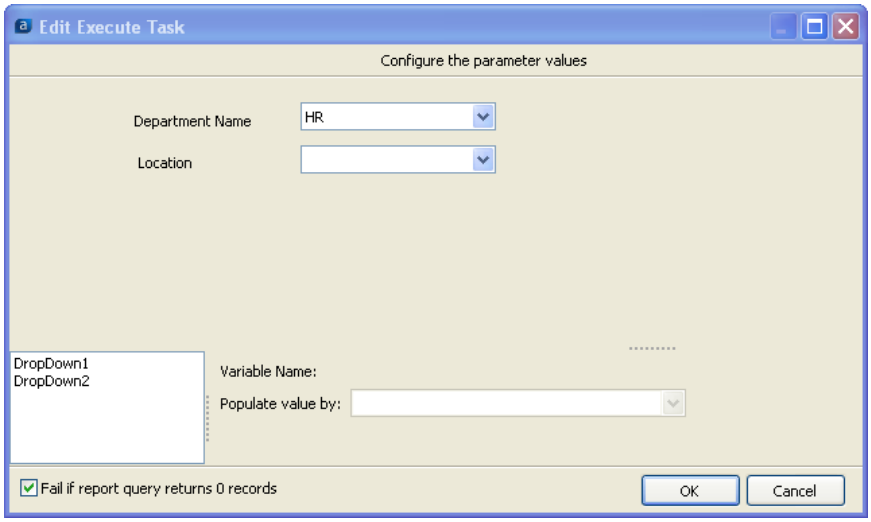
Only administrators can edit options on this tab. The API is used to call Argos reports from outside of the Argos software. This API interface provides seamless integration of Argos into various portals, web pages, and 3rd party applications.

The following sections describe each of the available tasks that you can specify on the Tasks tab, and their various configuration options.

Execute the Report Task

Argos automatically adds the "Execute the report" task to the task list. 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**.

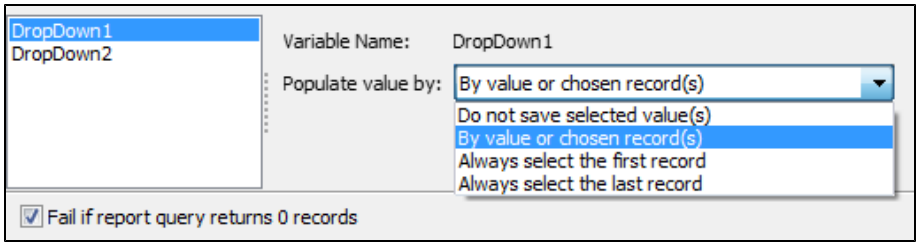
If you have set the AutoSelect property of an object to "yes", then the very first entry in the list or drop down will be used as the value when running the schedule. Otherwise, you need to enter the values to be used for each parameter on the form.



Executing the Report

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

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



Selecting method to populate the input parameters

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

The Auto Select Property for an object

Displayed when editing an object in the DataBlock

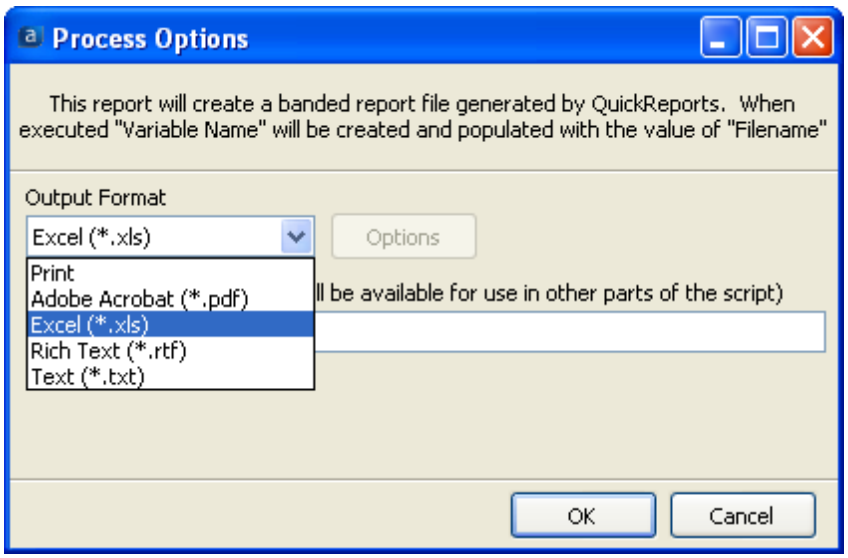
Auto Select	No
Choices	(Choices)
Columns	(Columns)
Cursor	Default
Description	
Font	(Font)
Font.Bold	No

Clicking one of the input fields (Dept. Name & Location) will activate the corresponding object name (DropDown1 & DropDown2).

The Process and Save Task

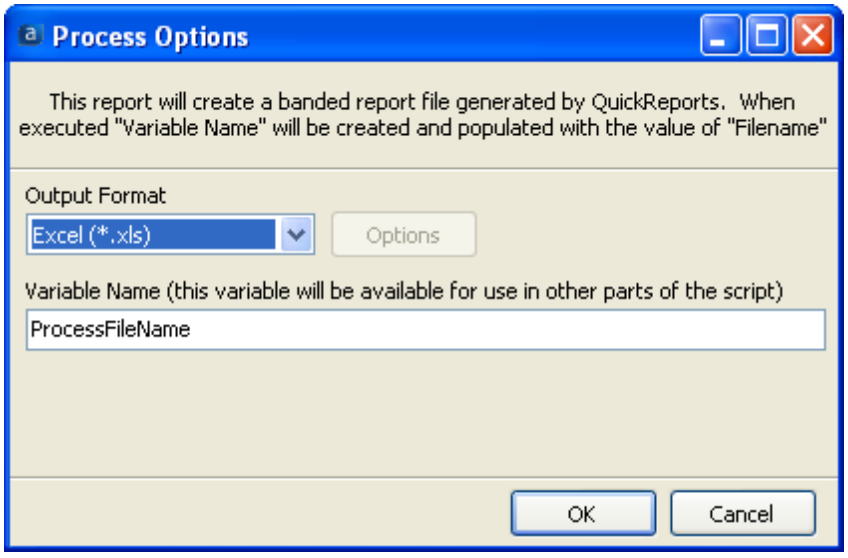
The other Task automatically added by Argos is the "Process and save" task.

The available options as shown below include "Print" in which the scheduled report will be printed, or to save the report output in one of the formats shown in the figure below.



Options for Process and Save

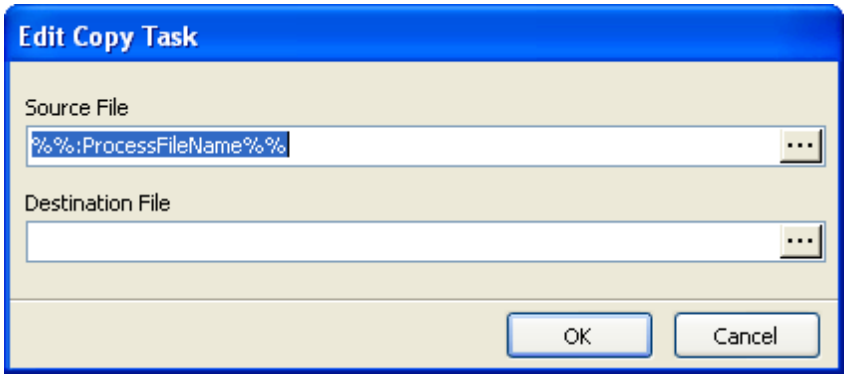
If you want to save the report output on the network, select a format. When choosing a format, the dialog box below is displayed where you specify the name of the report as an Argos variable. This variable (representing the report) can be used in other tasks as discussed later. Note the default variable name is ProcessFileName. For this example the default value is used.



The report name variable

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. Contact your MAPS Administrator to find out which destinations are available to you.

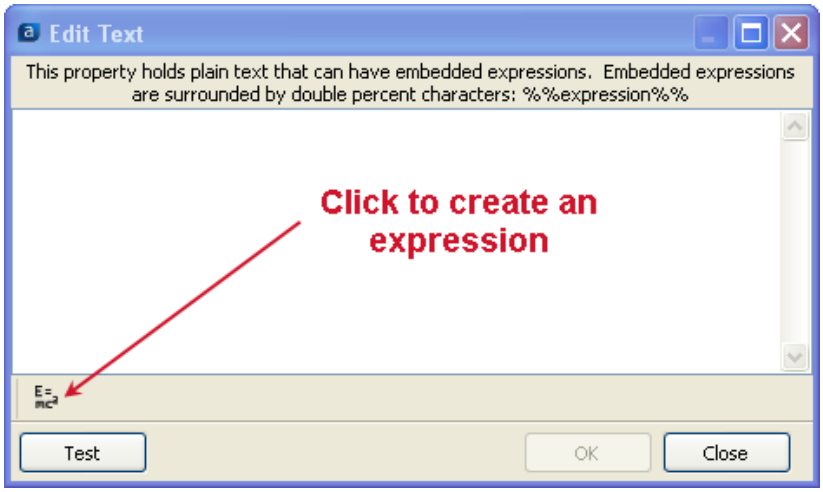


Copying the report output to a destination

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.



The Edit Text dialog box

The Destination File

See your MAPS Administrator to obtain the destination directory where you can save files.

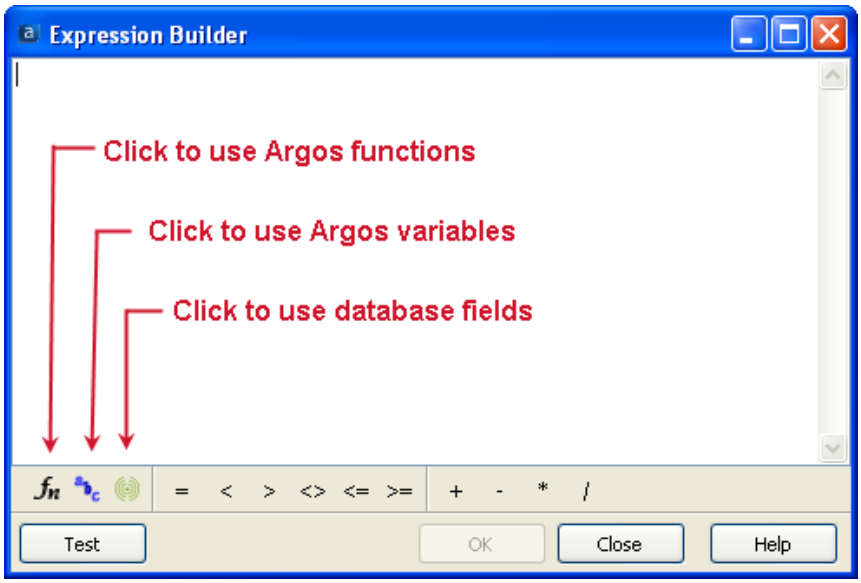
The directory on the MAP server where the MAPS applications files are to be stored is entered in the Server/File Operations option within the MAP Server Configuration Tool.

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

Destination File permission

The user account running the scheduled report must 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.

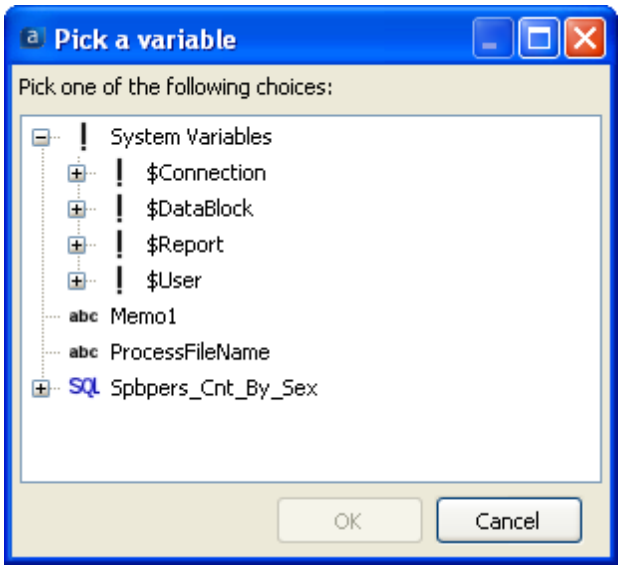
Click the **E=mc^2** icon to bring up the Expression Builder shown below. Within this dialog you can build expressions using Argos functions, Argos system variables, and database fields by clicking the icons shown below.



Click the Test button to test your expression

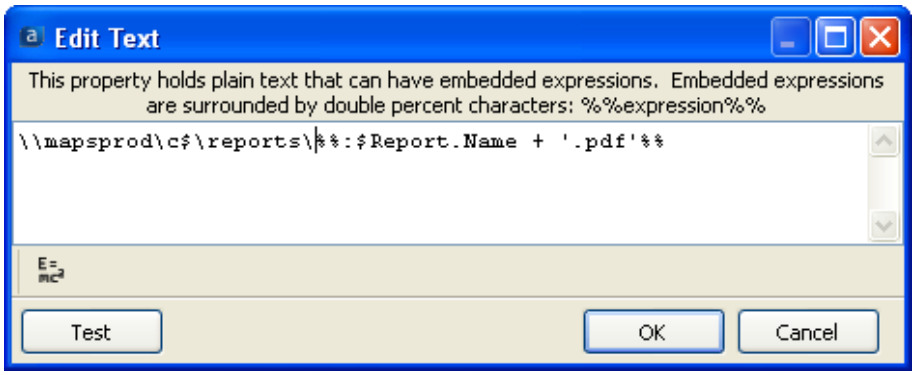
The Expression Builder dialog box

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 **abc** icon (shown in the figure above) displays the "Pick a variable" dialog where you can select System and Argos variables.



The list of Argos Variables

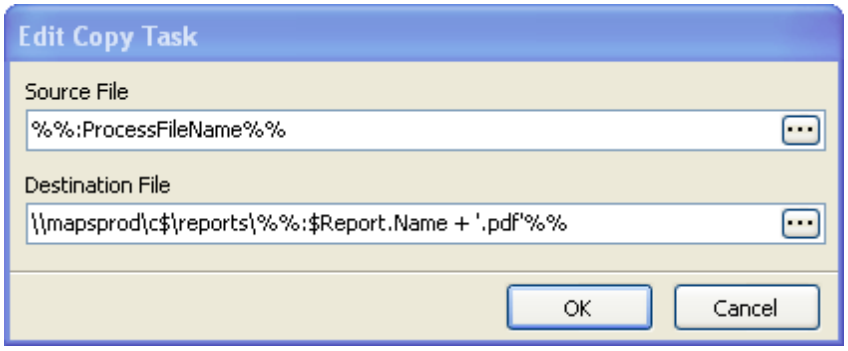
Select the **$Report** System variable, click OK, and complete the expression as shown in the figure below. The Report Name and extension is appended to the destination directory.



Click the Test button to test your completed expression

The expression which includes the Argos System Variable

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



The completed Destination file name

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:

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

The saved file name will then be named yyyymmdd.pdf

Use of Source File Name

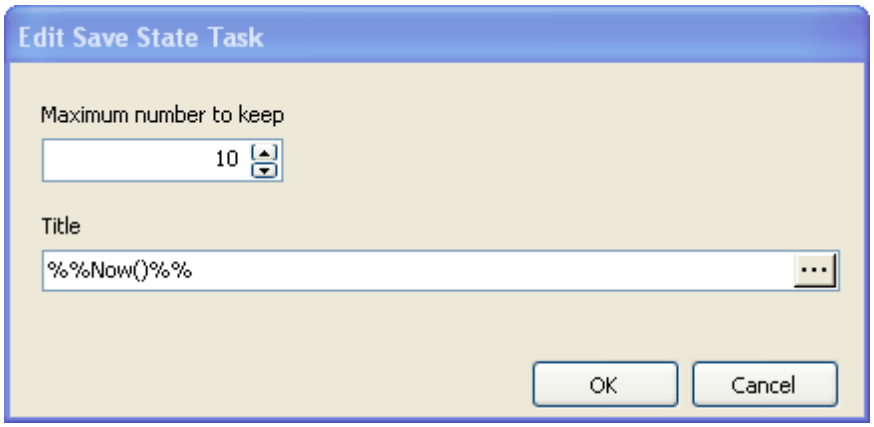
In the above examples, the default Source File name (ProcessFileName) was used throughout the example. If you are only scheduling one report it is not necessary to change the Source File name. However, if your schedule includes several reports with output to be saved, you will need to assign a Source 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.

The Saved States Task

One of the Available Tasks within the Tasks Tab is the ability to save an execution state. This allows you to save the data as it was at the time the report was run so it can be reproduced exactly as it was at that time. When you select the "Save Execution State" task, the dialog box in the figure below appears.

As shown in the figure, you can enter the maximum number of states to save for the report involved. You can also create a title for the 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.

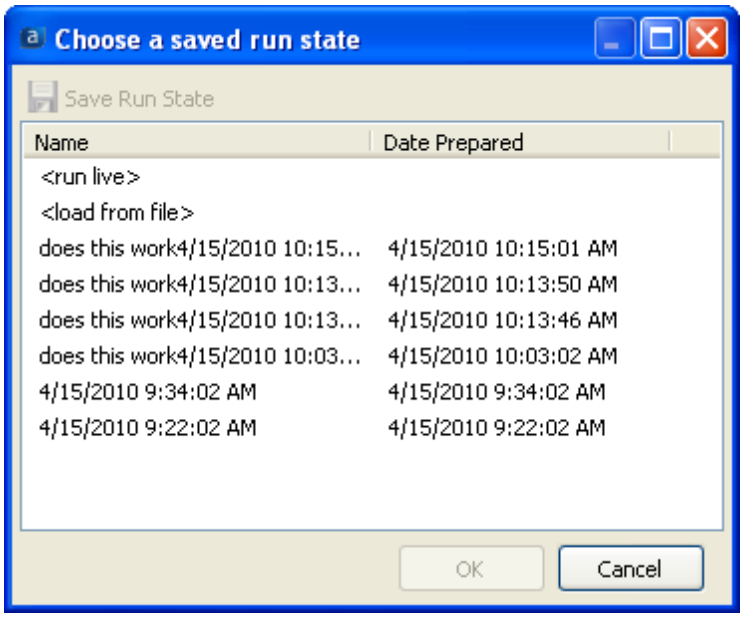
Clicking the ellipsis at the right of the Title field launches the Expression Builder where you can create an expression to be used as the Title.



Options for creating a Saved State

Executing the Saved State

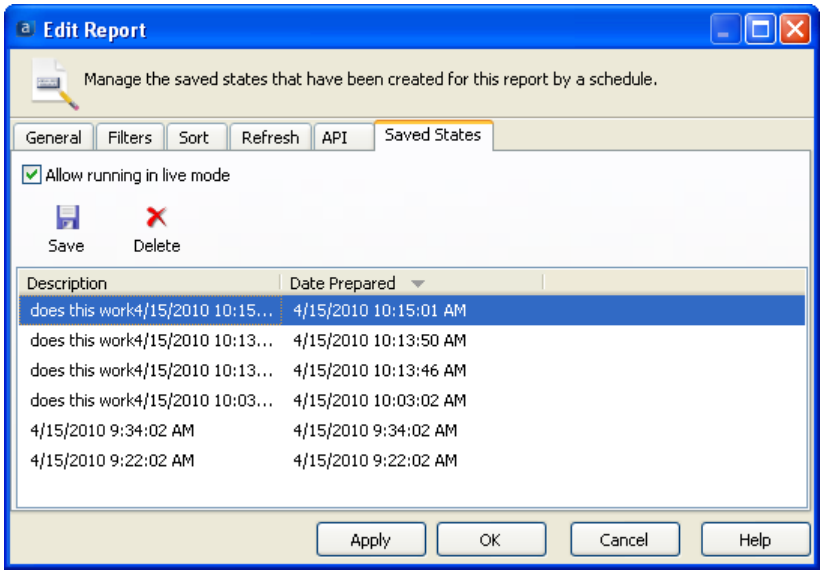
After the state is saved, when executing the report a list of saved states will listed as shown in the figure below. Executing saved states is described fully in the Argos Report Viewer Guide.



Saved States listed when executing a saved report

Managing Saved States

Saved 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 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 state to any desired location or to delete it.



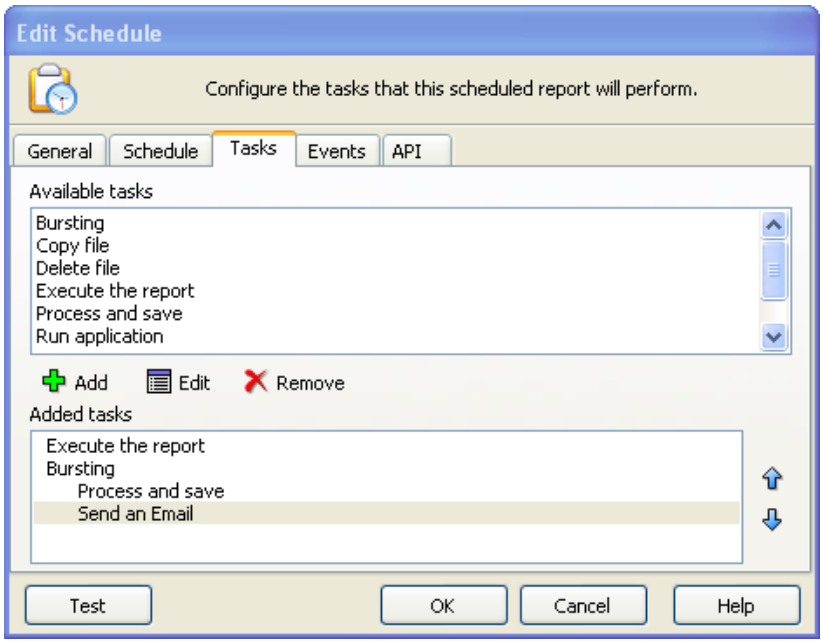
Allow running in live mode

Checking this box allows the report to be run using the current state of the database.

The Bursting Task

Bursting is typically used to separate individual records of a report. For example, a report containing information about an employee could be bursted 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:



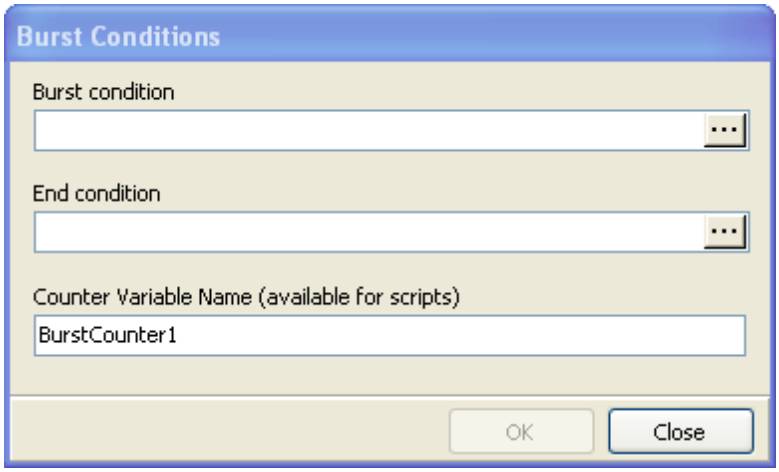
Nested tasks

The task order in the image at left means:

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

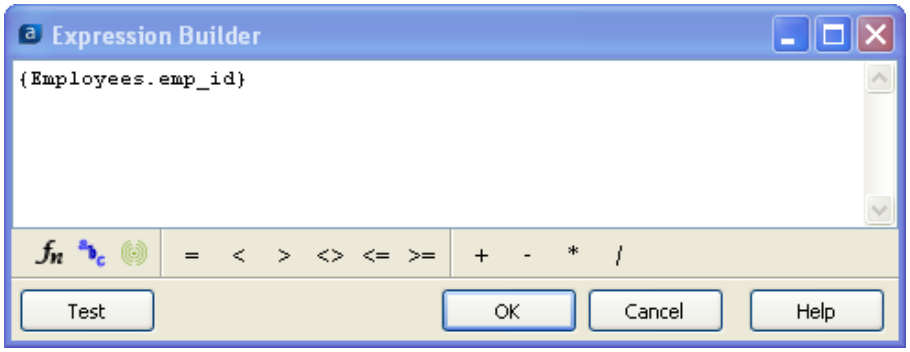
You can use the up and down arrow buttons to move tasks underneath other tasks. For example, selecting the execute task and clicking the down arrow would move it to a sub-task of the bursting task. Clicking the down arrow again would move it down the list of bursting tasks, and then to the position after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

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

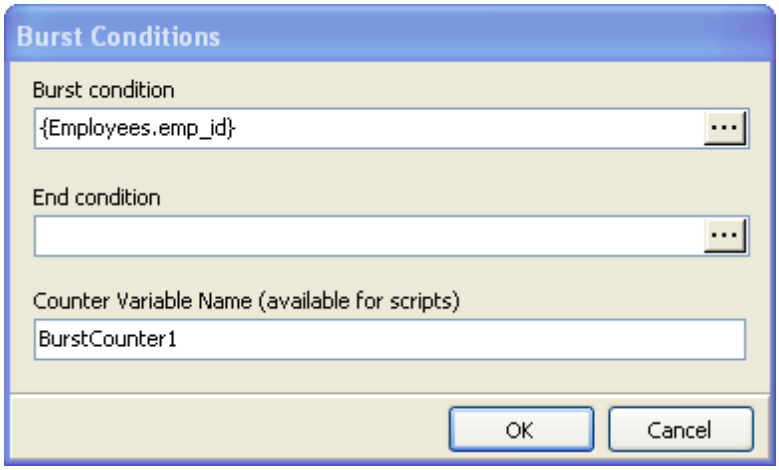


Specifying Burst Conditions

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 such that a report for each employee will be bursted.



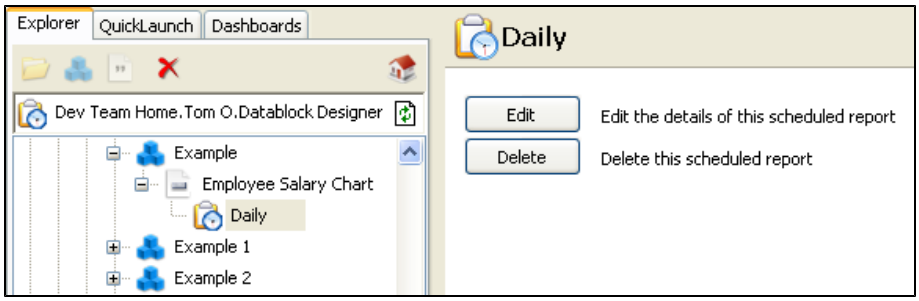
Using the Expression Builder to specify a Burst Condition



The Burst Condition field filled in

Managing the Schedule

After entering all appropriate information within the various tabs, the schedule is created and displayed under the report as shown in the figure below. The report will then executed at the specified day, time, and frequency, and will process output as specified.



The report shown in the Explorer Tree

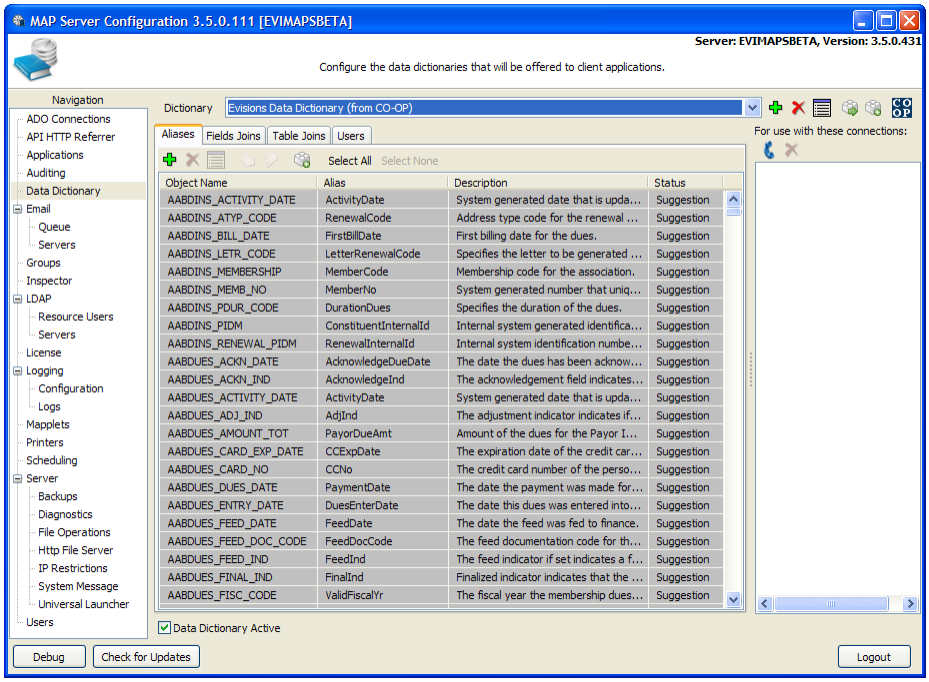
The schedule can be Edited or Deleted by highlighting the schedule then clicking the Edit or Delete button within the Design Area of Argos as shown above.

Data Dictionary

A MAPS Feature Used in Argos

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.



Data Dictionary Configuration in 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.

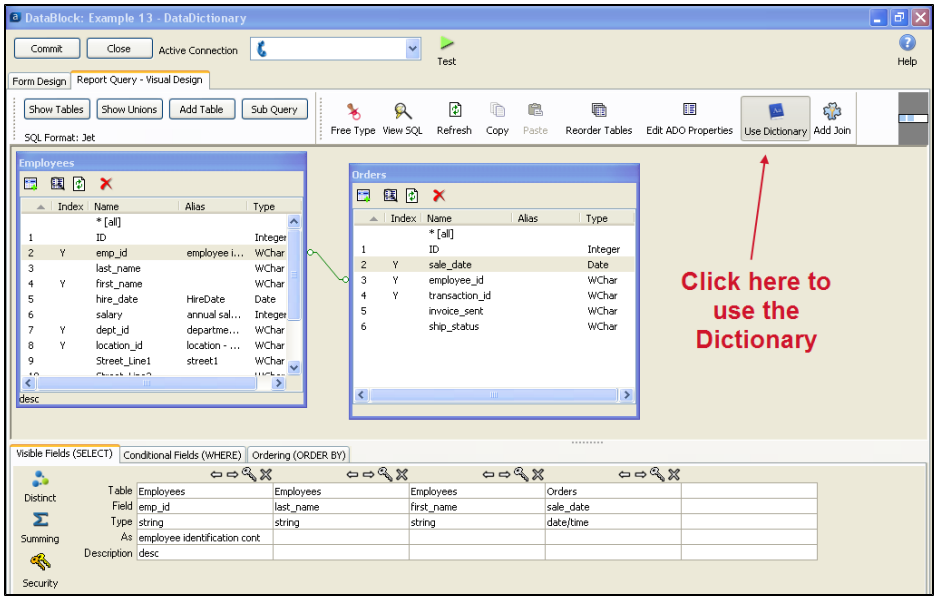
Item	Description
Aliases	Field level descriptive names that enable users to more quickly understand the tables and data they are working with and add consistency to your reports and DataBlocks.
Field Joins	Standard single field joins between two tables.
Table Joins	Custom joins between two tables that are not joined on just two fields. May contain a where clause in the join and is defined using text.
Users	Add users and the rights that they have to this dictionary.
Associated Connections	Data Dictionaries need to be assigned to a connection so that they can be used by the products (Argos) using that connection.

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

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.**



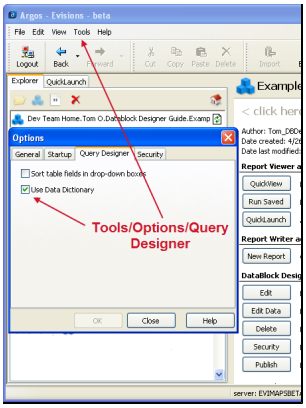
DataBlock Designer Showing Use of Data Dictionary

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

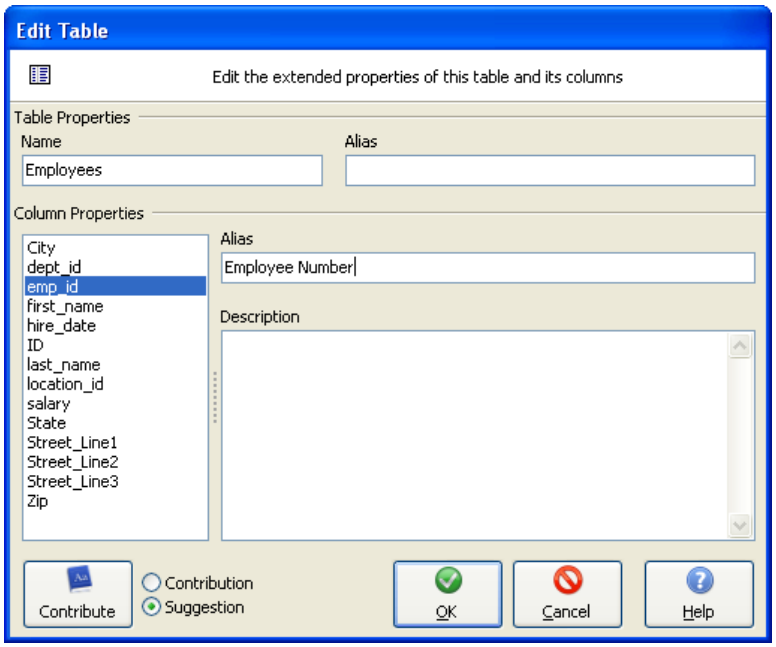
Depending upon the user permissions, suggestions and contributions can be made to the Data Dictionary using the Edit Table dialog box. Simply add aliases and descriptions to the selected field and contribute your entry.

Make sure the "Use Data Dictionary" feature is enabled via main Argos toolbar as shown below



Enabling the Data Dictionary

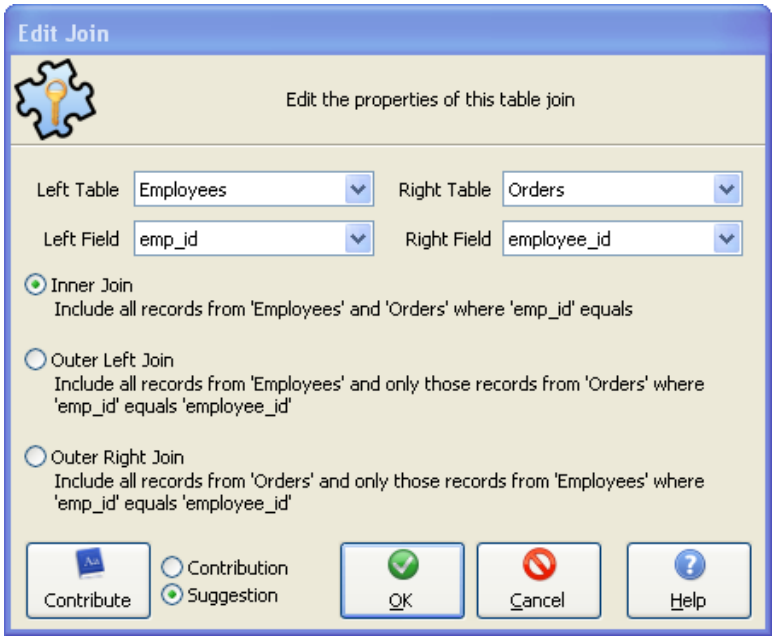
"Use Dictionary" Toggle on DataBlock Designer toolbar



Edit Table Properties

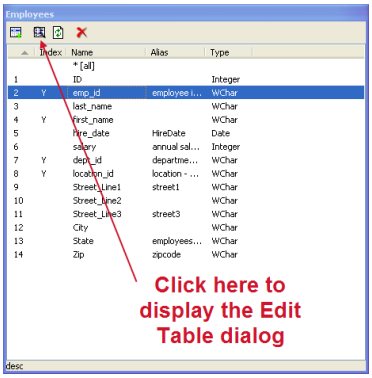
Adding Joins from within Argos

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).



Standard Field Join

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.



Display the Edit Table dialog

Contributor

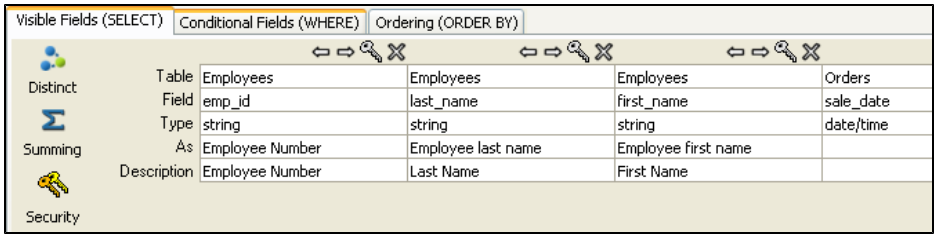
Users can contribute to the Data Dictionary by submitting alias and join information from within the Visual Query Builder. Depending upon permissions, information may be submitted as a suggestion which can then be reviewed by an administrator. The administrator can then change the suggestion to a contribution.

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

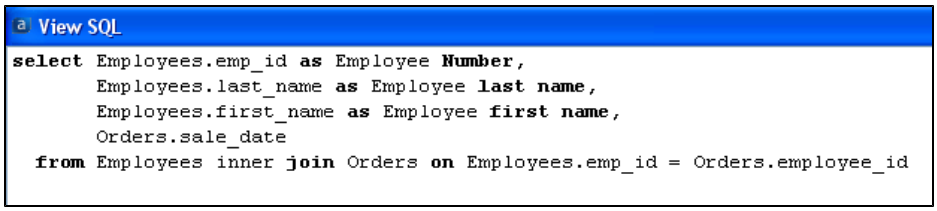
Add Join

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.



Alias added to Query

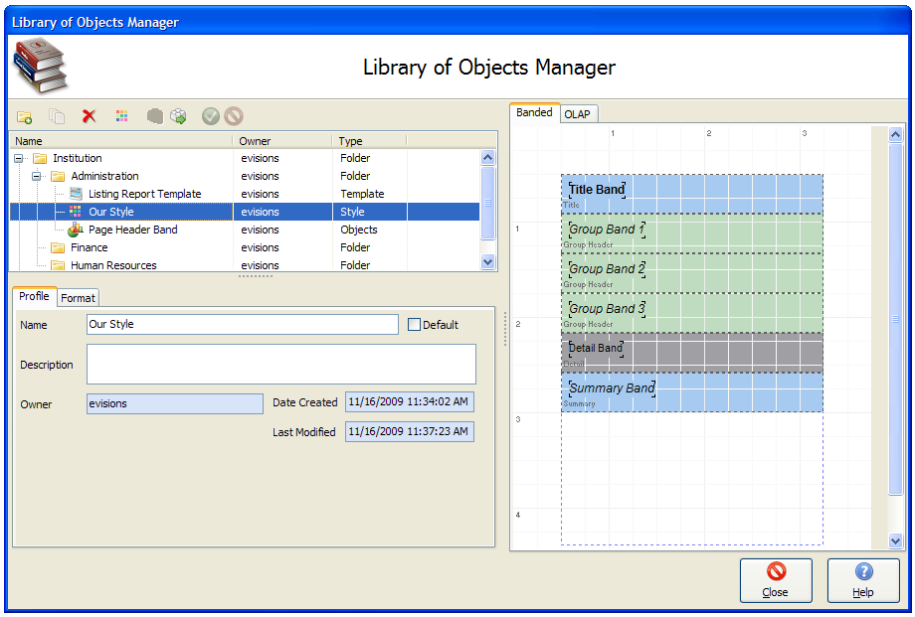


Join Info Added to Query

Library of Objects

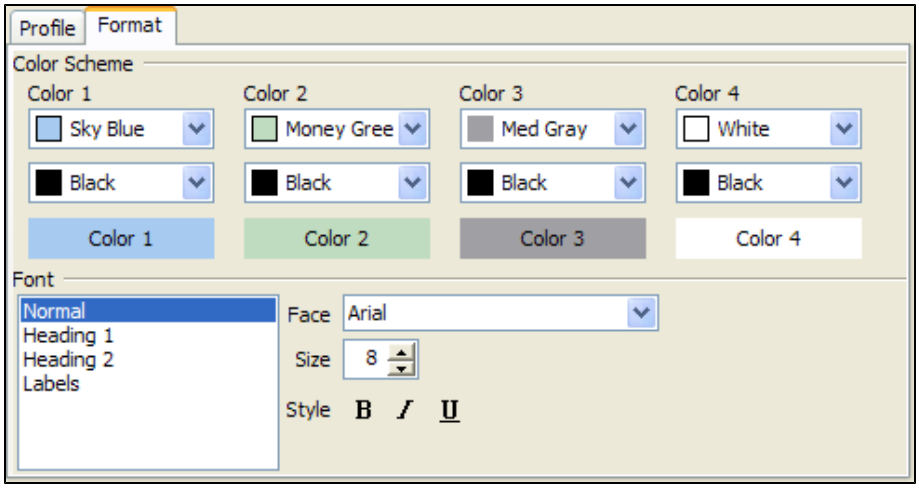
The Library of Objects (the Library) is made up of a set of features based on the concept of saving and reusing objects throughout Argos. The Library interacts with various parts of Argos in different ways, but sets the foundation for sharing objects throughout the tool. The main interface for the Library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

Library Manager



Library of Objects 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.



Edit Styles Using the Format Tab

Color and Font Settings

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:

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

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.

Add to Library

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.

Add from Library

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.

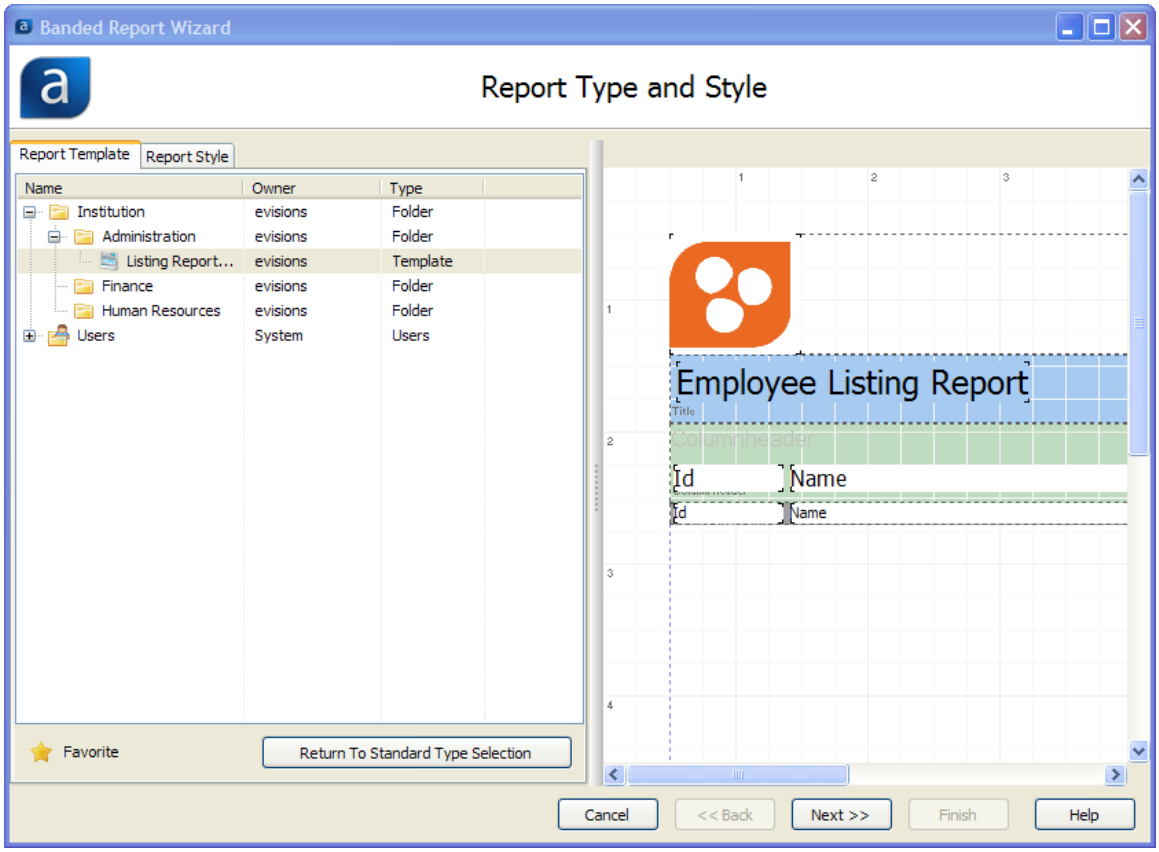
Add a Style

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.



Banded Report Wizard - Report Template

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.

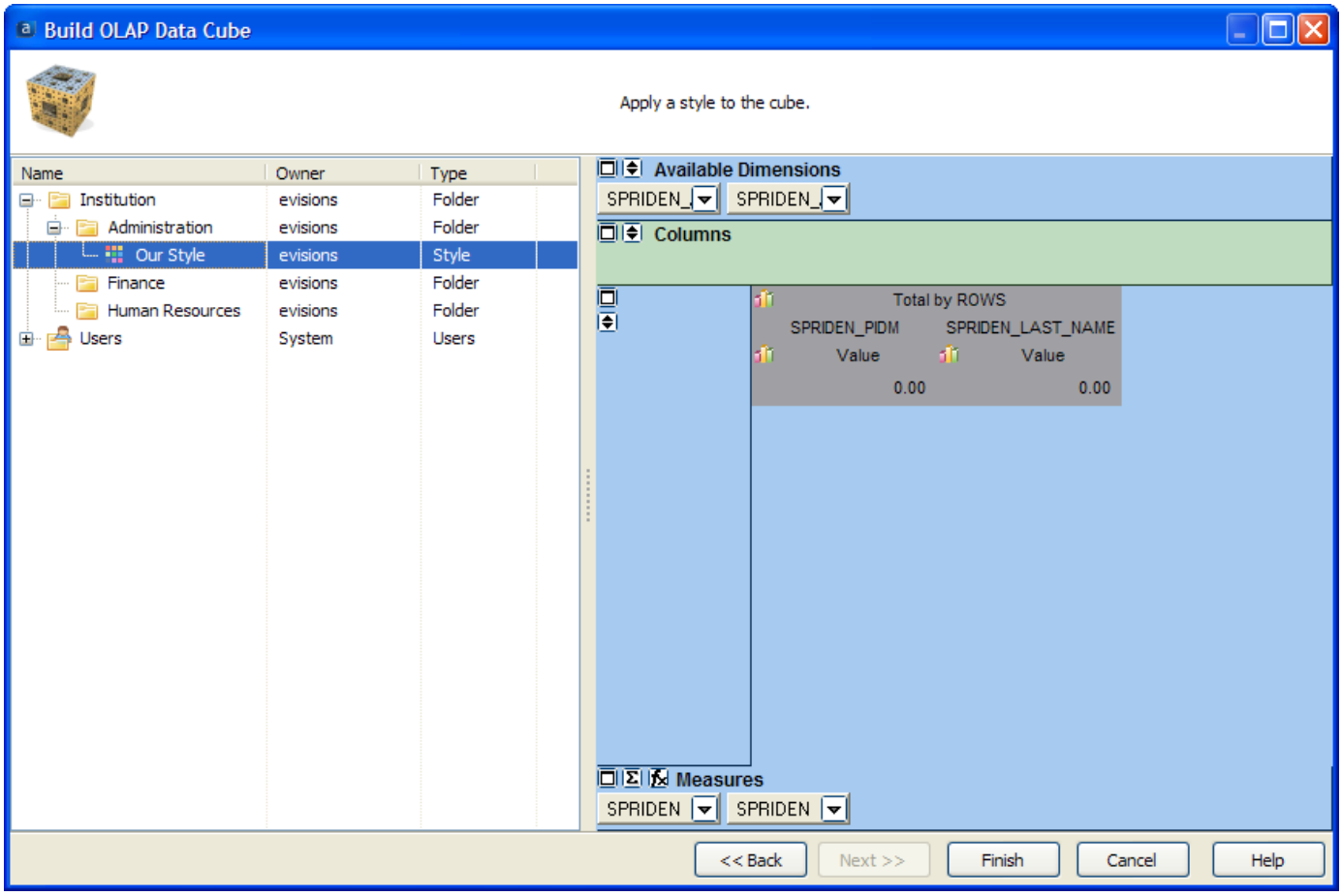
From within the Banded Report Wizard you can select a Style to use with your report. Styles are optional and can overwrite some or all style settings in the Banded Editor.

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.



Selecting a Style for an OLAP Cube

Security

Object Level Security

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

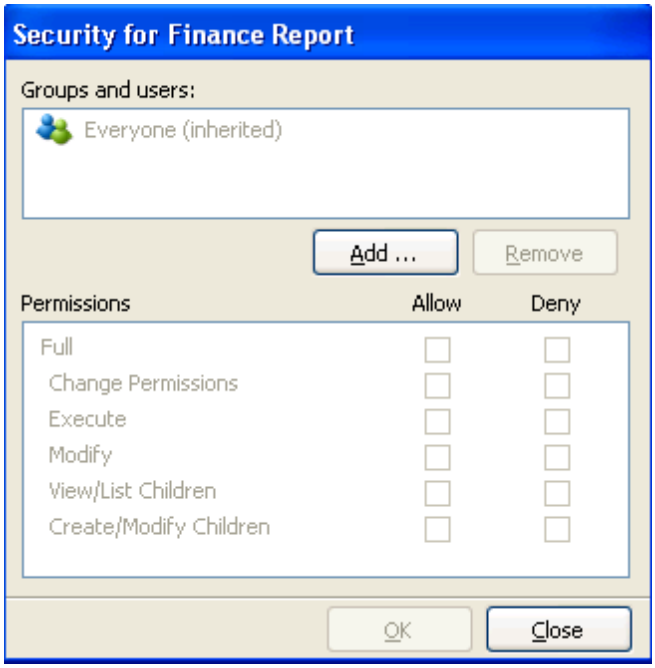
A dialog box is used to define security for each object and is accessed in two ways:

Highlighting the desired object in the Explorer Tree and either:

- Click the Security icon on the Argos toolbar, or
- Right click the highlighted object then select "Security"

Security Icon on Argos toolbar

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.



Security dialog box

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:

Full: 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.

Change Permissions: 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.

Execute: Run the selected object.

Modify: Modify the selected object.

View/List Children: 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.

Create/Modify Children: 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.

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

Direct Security: 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.

Inherited Security: 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.

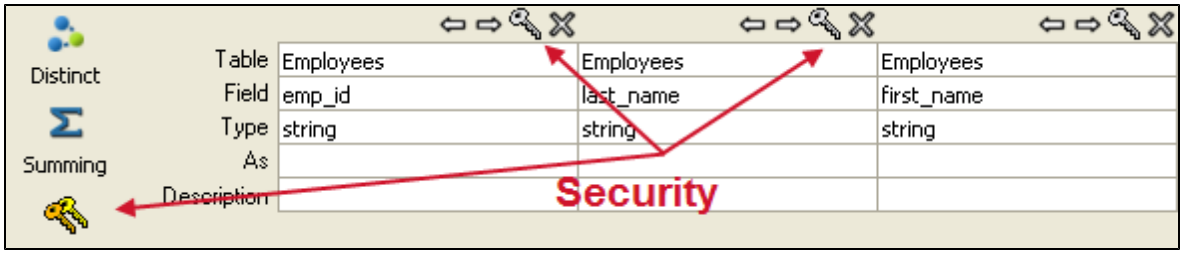
Accumulated Group Permissions: When a user attempts to access an Argos object, Argos will examine the permissions of all groups that they are a member of. Argos grants the user all accumulated positive group permissions to the object (unless that user has been directly Denied permissions - see above.) 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.

Everyone Group: 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.

Field Level Security

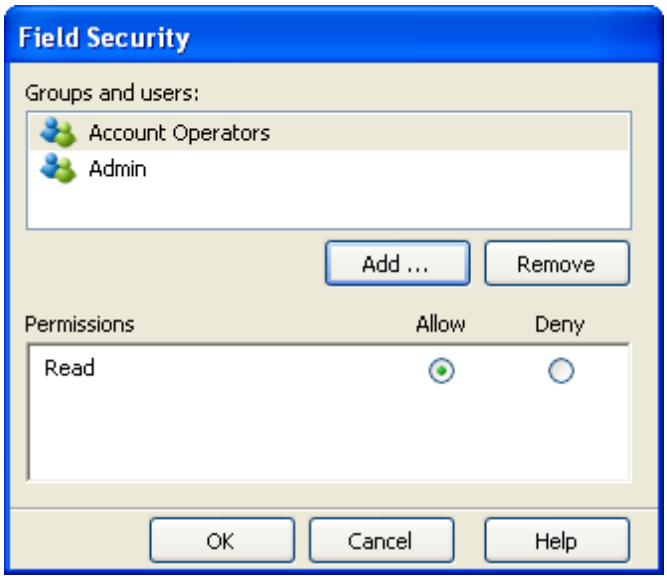
The above described how Security can be applied to entire entities such as Folders, DataBlocks, and Reports. However there are times when it is desirable to allow someone to execute a report, but limit the fields they can see.

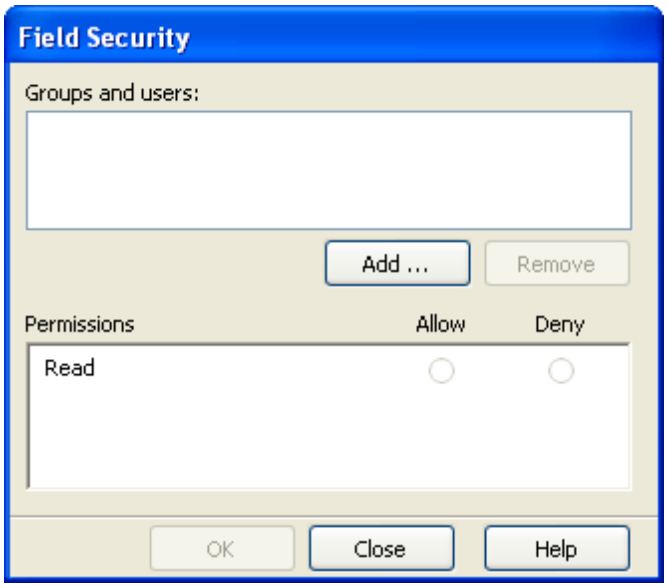
This is accomplished within the **Report Query - Visual Design** tab shown in the figure below. The key shaped icons are used to apply security to individual fields.



Security icons within the Query Designer

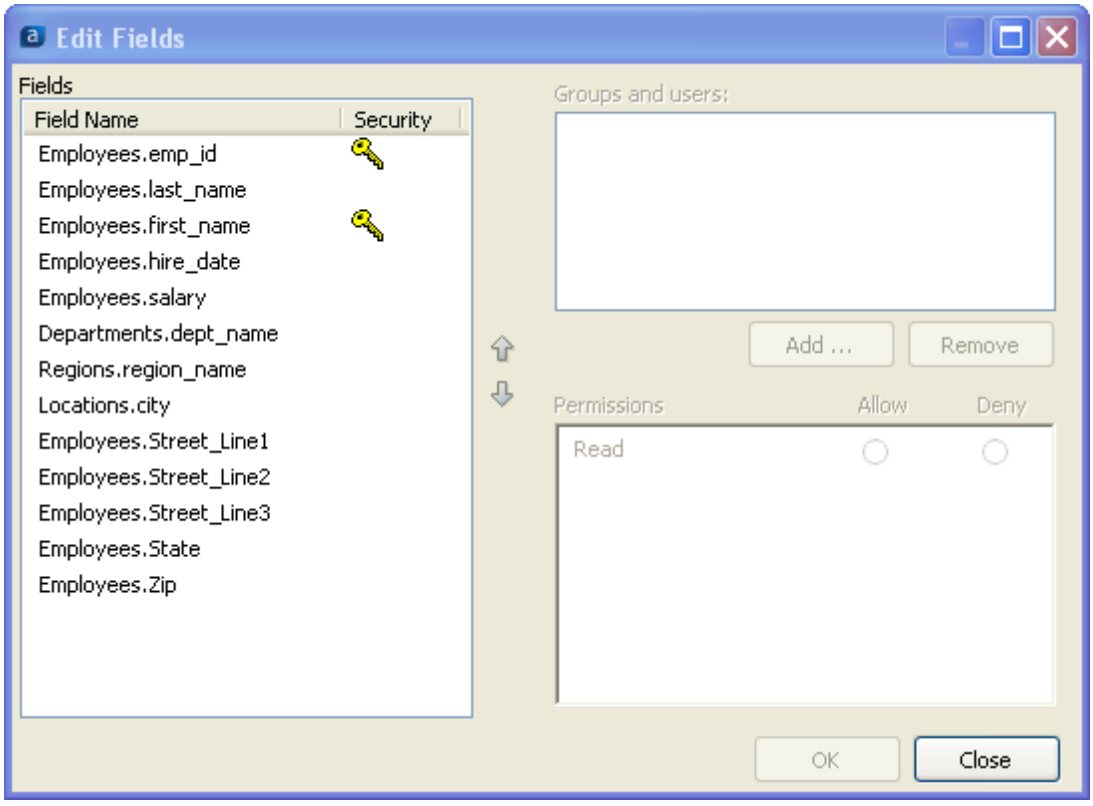
To limit access to a particular field, click on the "key" icon above the field. The Field Security dialog box shown below will be displayed. Click the Add button to display the list of Argos Groups and Users. Select the user or group then pick the Allow or Deny radio button. Click the Add button multiple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





Applying security to an individual field

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.



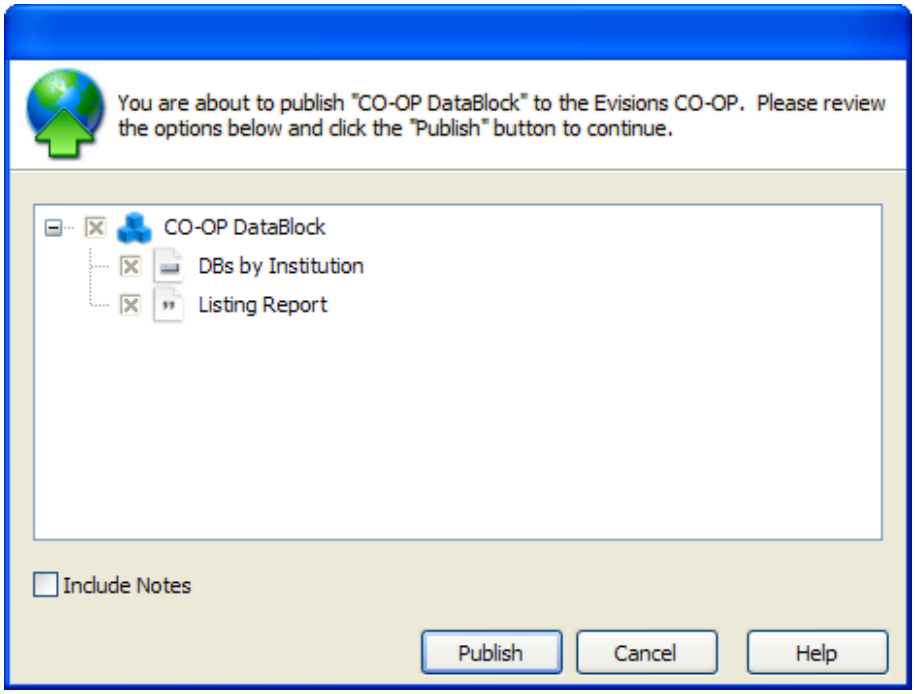
Applying security to several fields

The CO-OP User Community Share

The CO-OP was designed to be a place where you can collaborate with other Evisions clients by sharing DataBlocks and other objects. The CO-OP already houses some 900+ Evisions and client created DataBlocks.

Publishing to the CO-OP

Select the DataBlock you wish to share in the Argos Explorer then click Publish to specify the objects to include, and finally, complete the upload form, selecting the appropriate options for your DataBlock.



Publish to the CO-OP

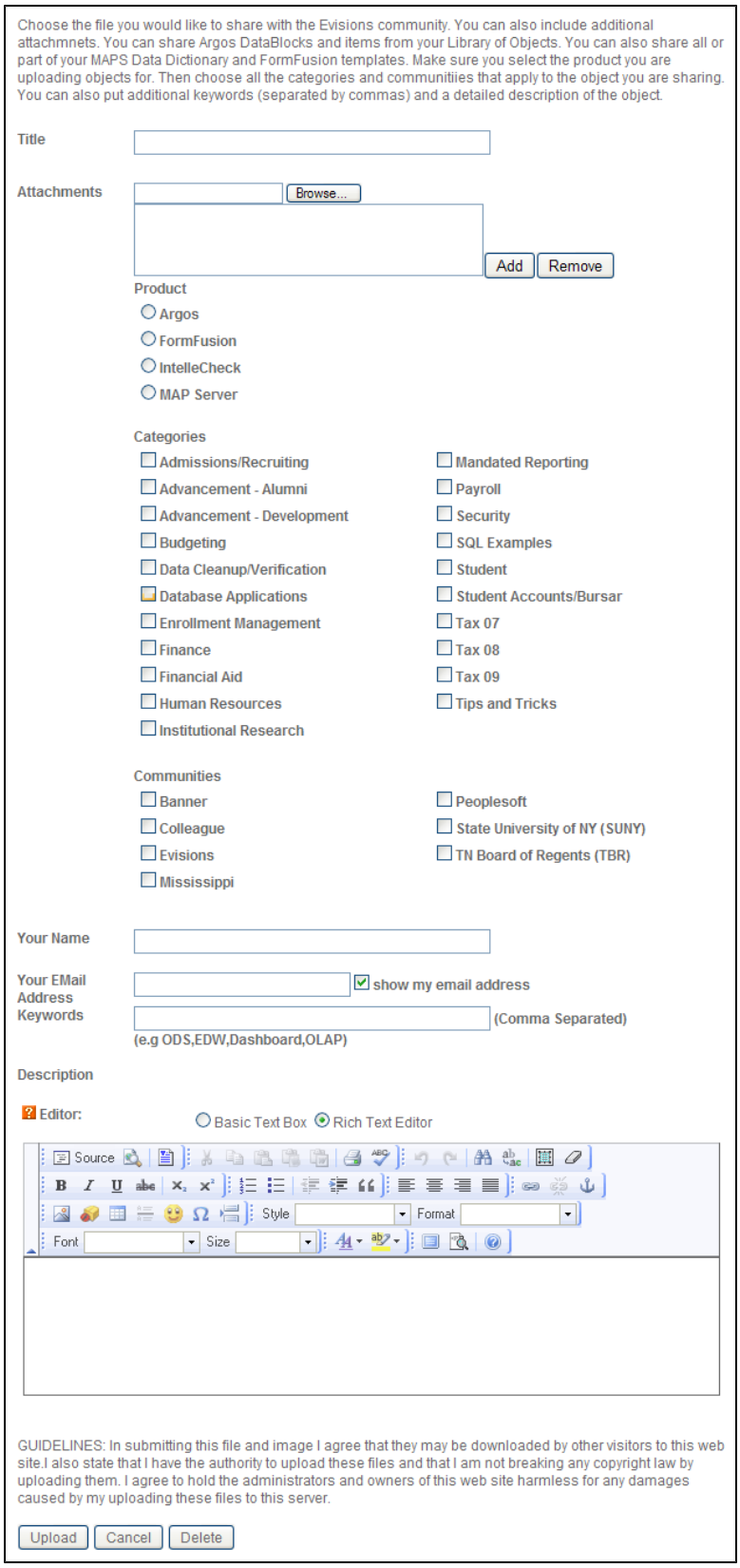
Complete the Upload Form

Complete the upload form with as much information as possible. You will need to specify the Categories and Communities for this DataBlock. Also, you can include attachments to the DataBlock. Screen images of forms, key SQL statements, report images and sample output files are all examples of attachments you might include.

Keywords

Please include several keywords that help to describe you DataBlock. Words that you would think of if you were looking for this DataBlock.

CO-OP Upload Form



Attachments

Add as many as you like

Categories

You must select at least one but you may select more.

Communities

You must select at least one but you may select more.

Keywords

Please add keywords to help others find your DataBlock

Description

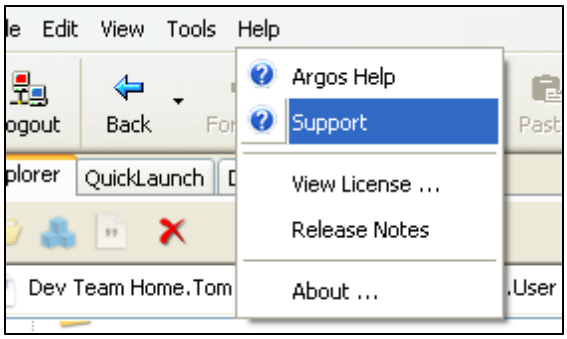
Your description will show on the main detail screen of the CO-OP Share listing.

Argos Resources

Support Site

The Evisions support site is located at http://www.evisions.com/Support/Overview.aspx.

The site requires a login to access. If you are a registered Argos user, you can create an account for yourself by going to **Help** -> **Support** in Argos.



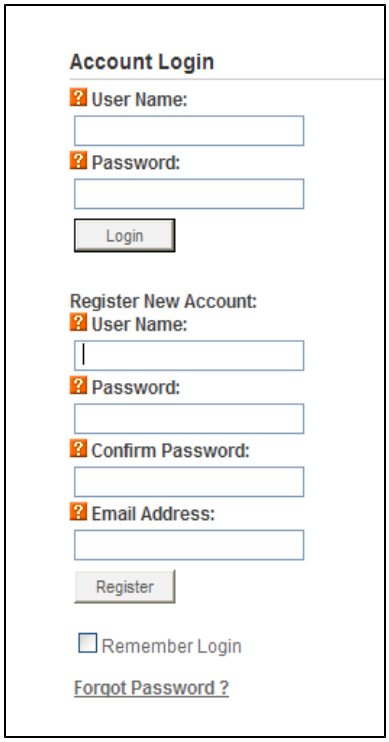
The first time you access the support site from Argos, you will be asked to log in or register a new account. Subsequent visits will use your Argos credentials to log you in automatically. The figure on the right shows the login screen that appears when accessing the support site for the first time.

Registering for a new account gives you a default set of permissions for the Evisions website. You may need to contact your administrator if you require additional permissions, such as HelpDesk access or the ability to upload items to the CO-OP Share.

After accessing the site, navigate around the site noting the available tools for Argos users.

The Argos support page contains links to many useful resources including:

- Release guides.
- Online and recorded training. Online live training is completely free and can be repeated as needed.
- Documentation such as upgrade guides and technical white papers.
- Pre-built DataBlocks provided by Evisions.
- FAQs.



Account registration screen

CO-OP User Community

The CO-OP user community is home to the Share and the product forums. The CO-OP Share contains a large number of pre-built DataBlocks that you can download and use in Argos. Evisions provides some of the DataBlocks; others have been uploaded to the Share by clients like yourself. We encourage you to take advantage of this collaborative space, and to upload any DataBlocks that you feel could benefit other Argos users.

You can access the CO-OP via the button on the Argos toolbar, or from the Support tab of the Evisions website. Again, if you do not have an account, you can register through the Argos software or your administrator can create an account for you. You may also need to contact your administrator to have upload permissions for the CO-OP added to your account.

The CO-OP icon on the Argos toolbar

Important Links

- Product Support
- Training - Free, live online training for all products.
- In-product Help - A complete description of all features in Argos. Also includes training videos and Argos user guides.
- Knowledge Base - Answers to common questions or problems.
- HelpDesk - Technical support.
- CO-OP Share - Pre-built DataBlocks and data dictionaries from Evisions and other clients.
- Forums - Collaborate with other Argos users, and receive information on product updates.

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 knowledge base of common issues and their resolutions at http://helpdesk.evisions.com. 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.