SedonaOffice

The #1 Financial Software for Security Companies

2010 Users Conference

Database Structure

Training & Setup Guide

Last Updated: December 22, 2009

About this Guide

This SedonaOffice Database Structure Training Guide is for use by SedonaOffice customers only. This guide is to be used in conjunction with an approved training class provided by SedonaOffice, and is not meant to serve as an operating or setup manual.

This training and setup guide is for experienced SedonaOffice users who have knowledge of the database setup. While this guide will review some of the basic setup necessary, this guide is not intended to teach Database Structure basics and assumes the user has knowledge of SQL and of the SedonaOffice application.

SedonaOffice reserves the right to modify the SedonaOffice product described in this guide at any time and without notice. Information in this guide is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. In no event shall SedonaOffice be held liable for any incidental, indirect, special, or consequential damages arising out of or related to this guide or the information contained herein. The information contained in this document is the property of SedonaOffice.

This guide will be updated periodically, be sure to check our website at www.sedonaoffice.com for the most current version.

Table of Contents

Overview

This guide is intended to teach you how to access data from a SedonaOffice database. Data extracted from a database can be used for many different purposes both internally and externally for an organization. While this guide will review a variety of different techniques, it is impractical to detail each and every type method that can be used to extract data.

Each Company is a Database

Each SedonaOffice company is its own unique database within the SQL server. In addition to the various company databases, there is an additional database that helps to controls access to the company databases. This access control database is named SedonaMaster.

SedonaMaster contains list of company names and the database associated with each name. All other data about a company is contained within the company database. All of the setup information, names, addresses, part numbers, service tickets, etc., for a company, are stored within the same database. The structure of the database will remain the same for all companies. The differences in how companies operate are contained in the setup tables. If a feature of SedonaOffice is not used, the data structure will still exist but may be empty of data.

Databases Contain Tables, Views, and Stored Procedures

The main structures in a database are tables, views and stored procedures. Tables contain the raw data, the actual names, addresses, etc.

Views are premade queries that will return sets of data automatically. If there is a set of data you are going to regularly extract, you may want to think about making a view. SedonaOffice uses several views in supplying data to the client. Do NOT alter these or your system may cease to operate correctly[1].

Stored procedures are routines containing SQL code. They can be created to act as a view but are usually used to manipulate the data. Stored procedures also can take parameters, values that modify how the stored procedure will operate. Most of the business logic in SedonaOffice is handled by stored procedures. They are encrypted and locked for safety and security. Do NOT delete or replace a stored procedure or your system will cease to operate correctly[2].

Tables Contain Fields

Fields contain your actual data. They are different types:

- Text including varchar, nvarchar and char. The length of the field in characters (including spaces and punctuation) is defined when the field is created.
- Numeric including integer, double and money. What range and if a fractional decimal amount is supported is defined when the field is created.

[1] Unless directed to by a SedonaOffice support person.
[2] Unless directed to by a SedonaOffice support person.

- Datetime. Microsoft SQL server does not contain a field type for date or one for time, all date and time related fields are Datetime fields.

Linking Tables

Tables are linked via fields that end in Id. In each table, the first field is the Identity field for that table. Identity fields are not editable nor should you try. Identity fields are unique. This number is automatically assigned by the SQL server. Once assigned, a number is never reused, not even if it was previously deleted. This Identity field is the “Address” of the record. Other tables that point to this table will have an Id that matches the “Address” of the record. IE Customer_Id in the AR_Customer_Bill record will point to the Customer_Id field in the AR_Customer table. The Customer_Id in the AR_Customer table is the Identity or “Address” of that record. Notice that the name of an Id is the same as the table name in our example. This is true of all ID’s with very few exceptions.

Link Types

Table links are defined by the relationship of records in one table to the records in another table. There are three basic link models.

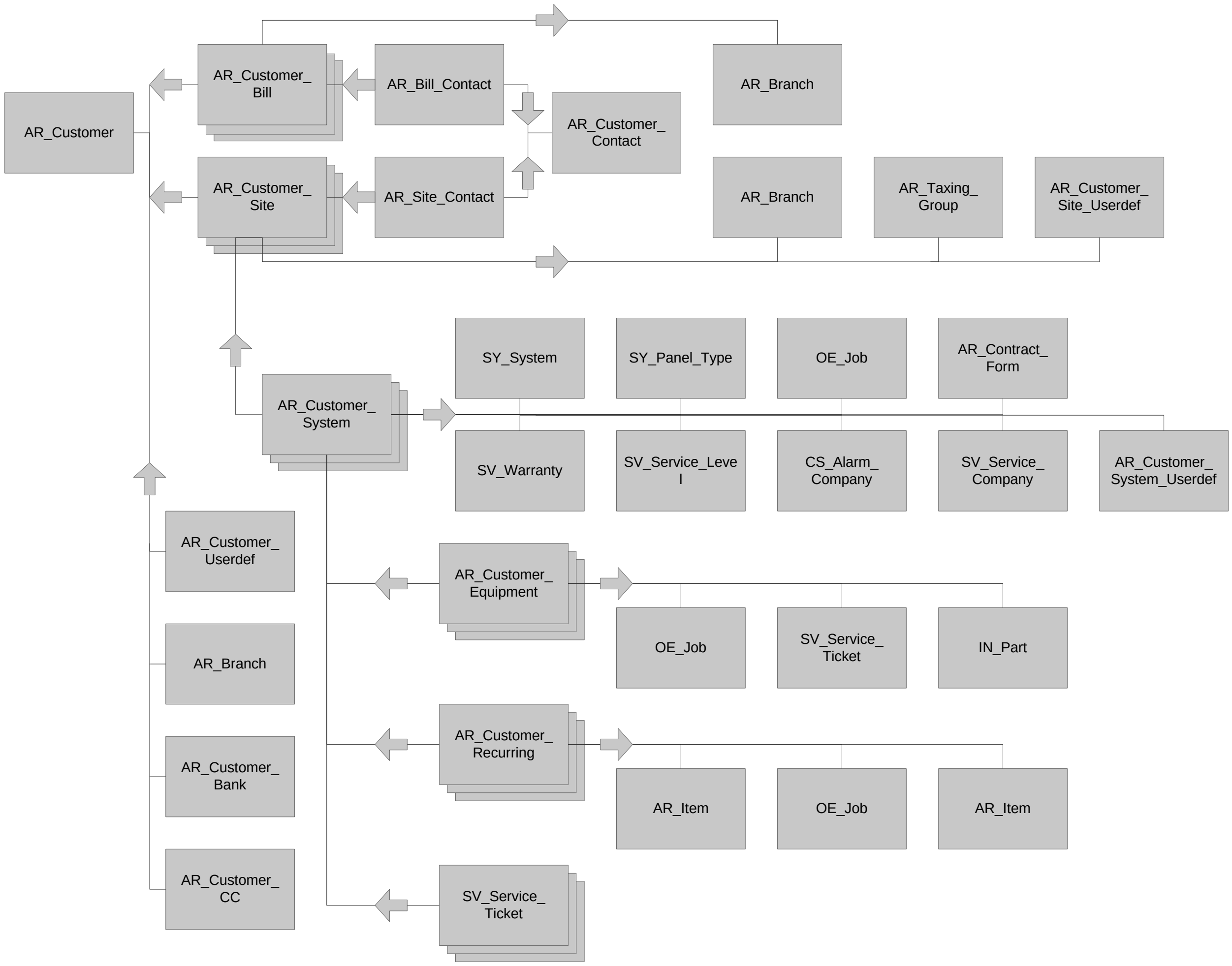
One to one: Each record in one table matches to exactly one record in the other table. IE AR_Customer and AR_Customer_Userdef

One to many: Each record in one table matches to many records in the other table. IE AR_Invoice and AR_Invoice_Item

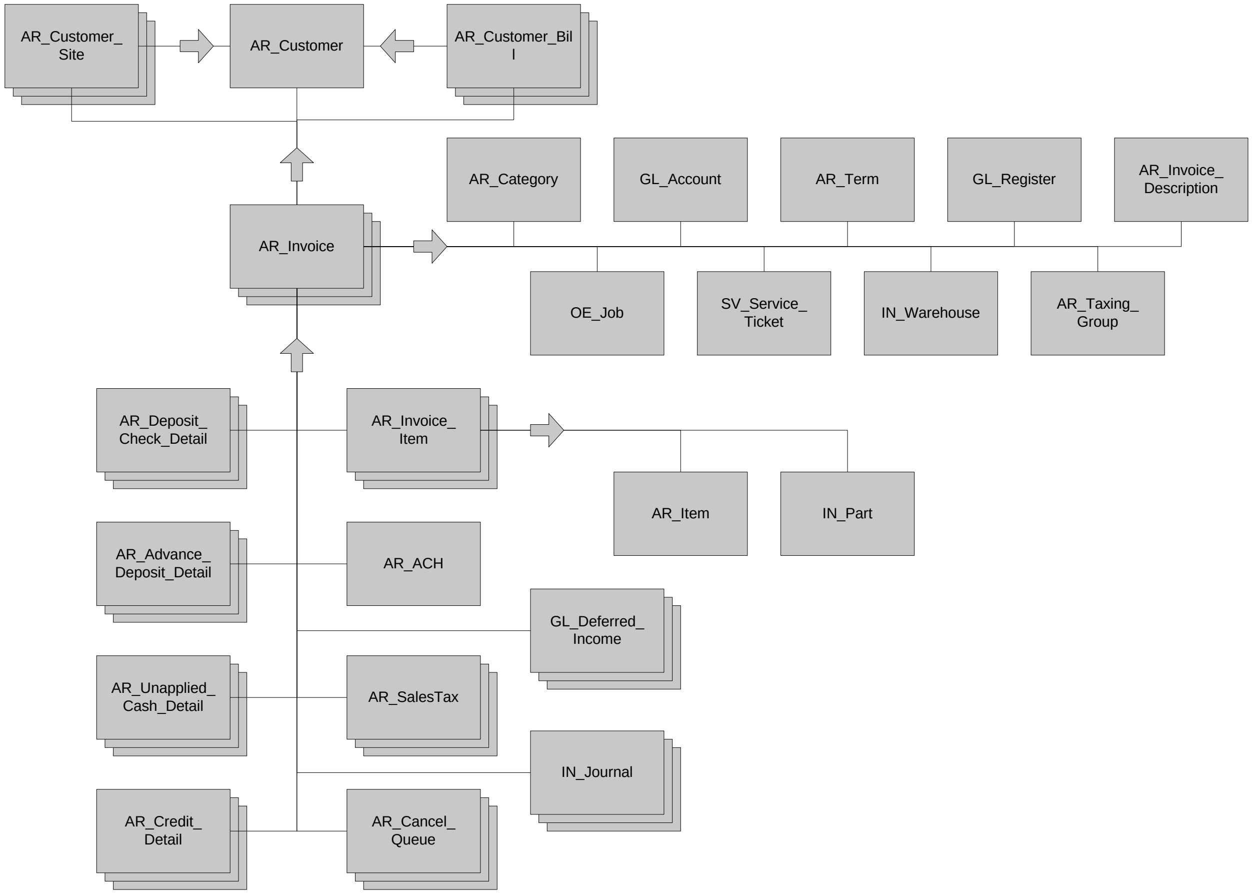
Many to one: Many records in one table match to one record in the other table. IE AR_Customer and AR_Branch

The following diagrams are not meant to be completely accurate or to be used as a definition of the database structure. They are a simplified diagram to give an outline of the relationship of the various tables that combine to make up a data structure.

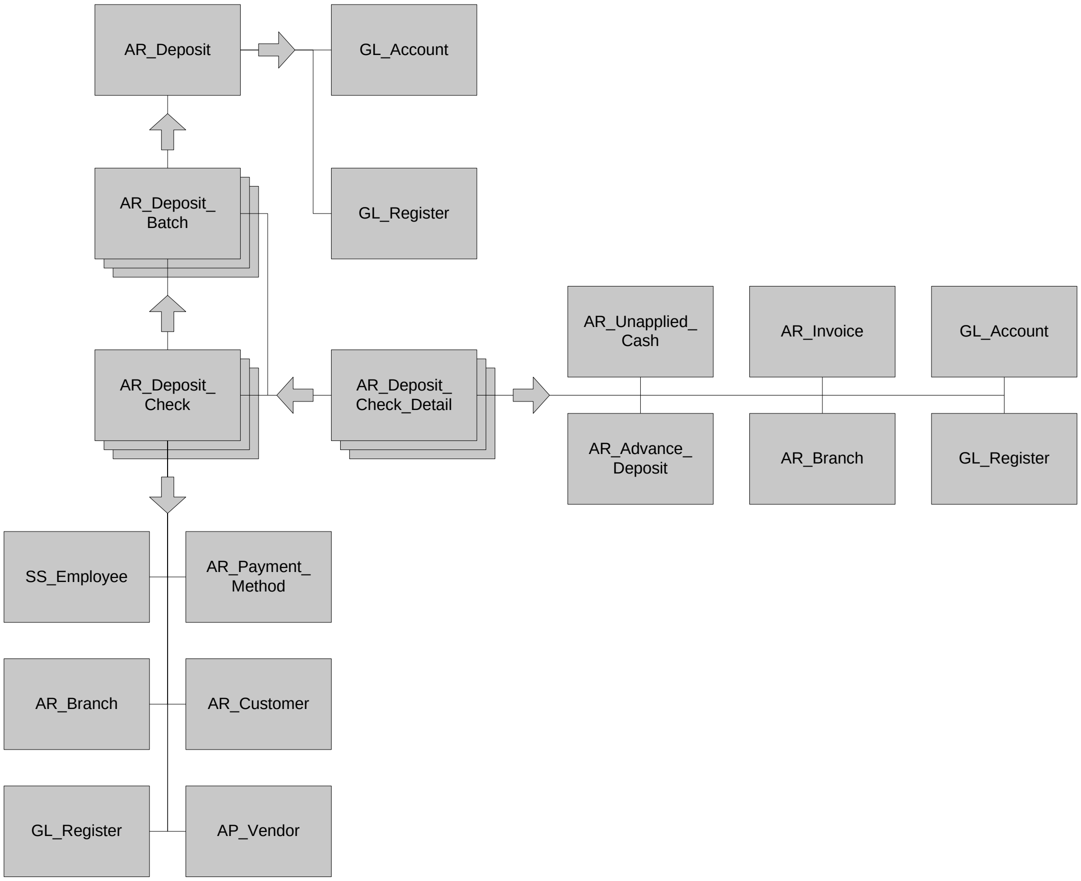
Customer Structure



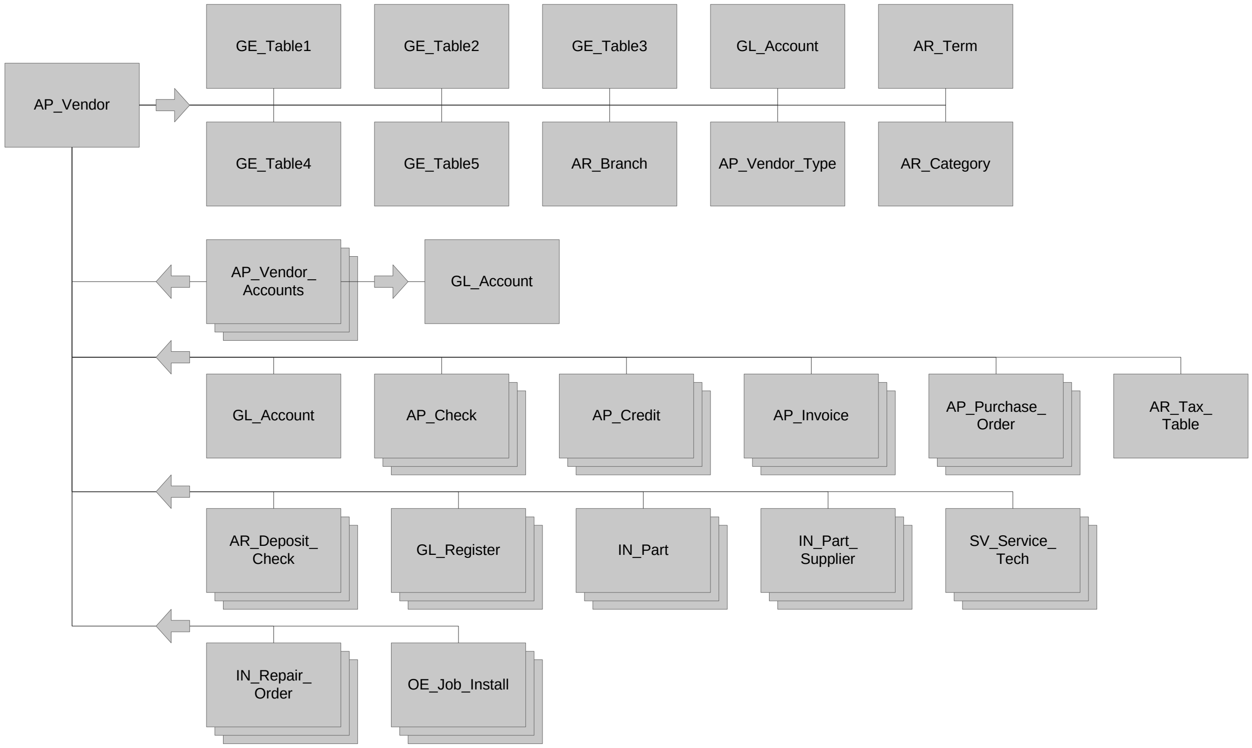
Invoice Structure



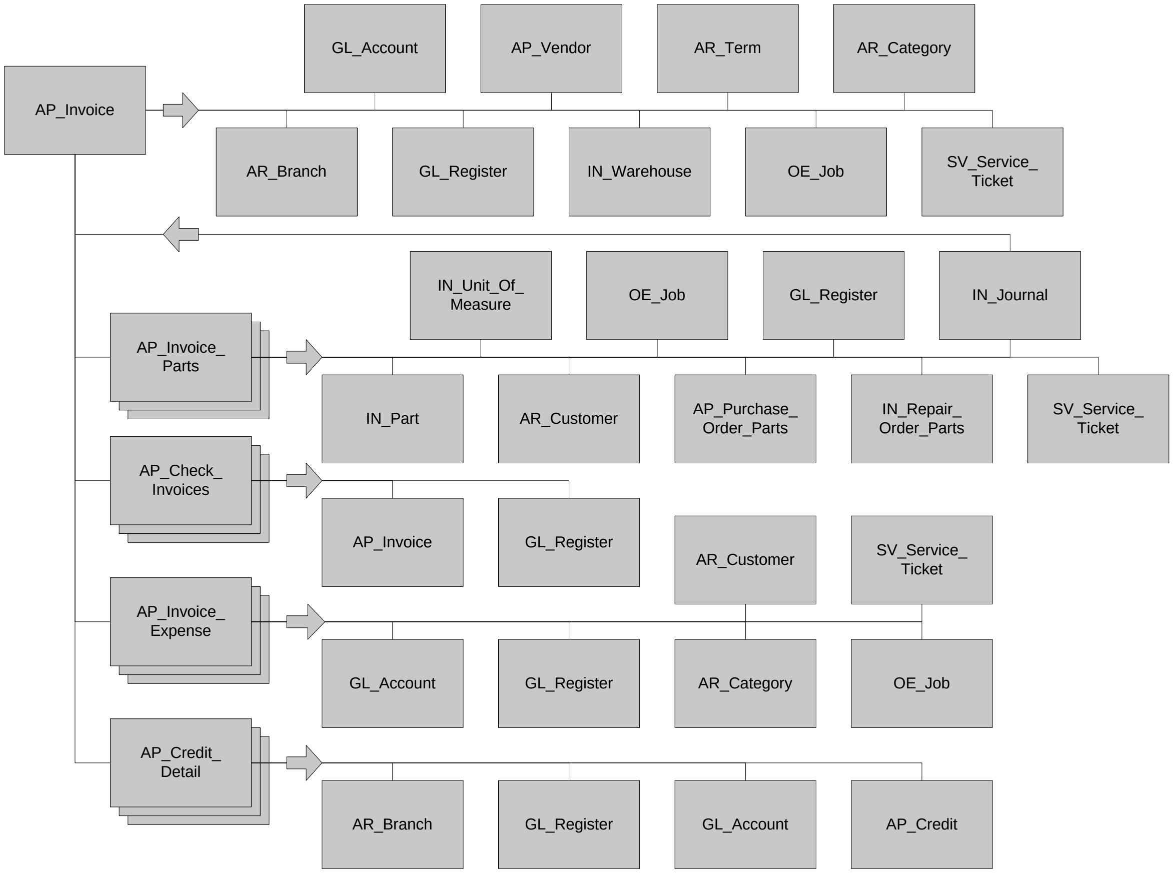
Cash Structure



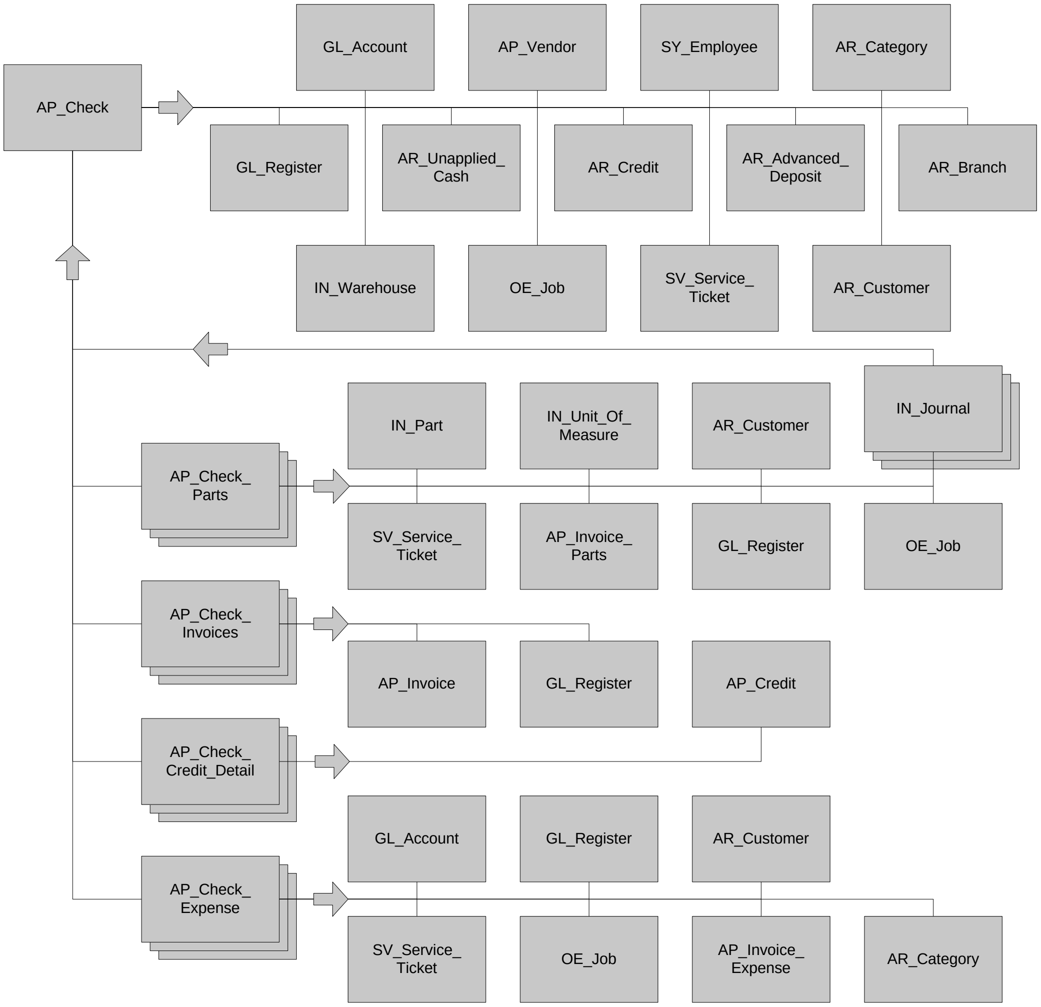
Vendor Structure



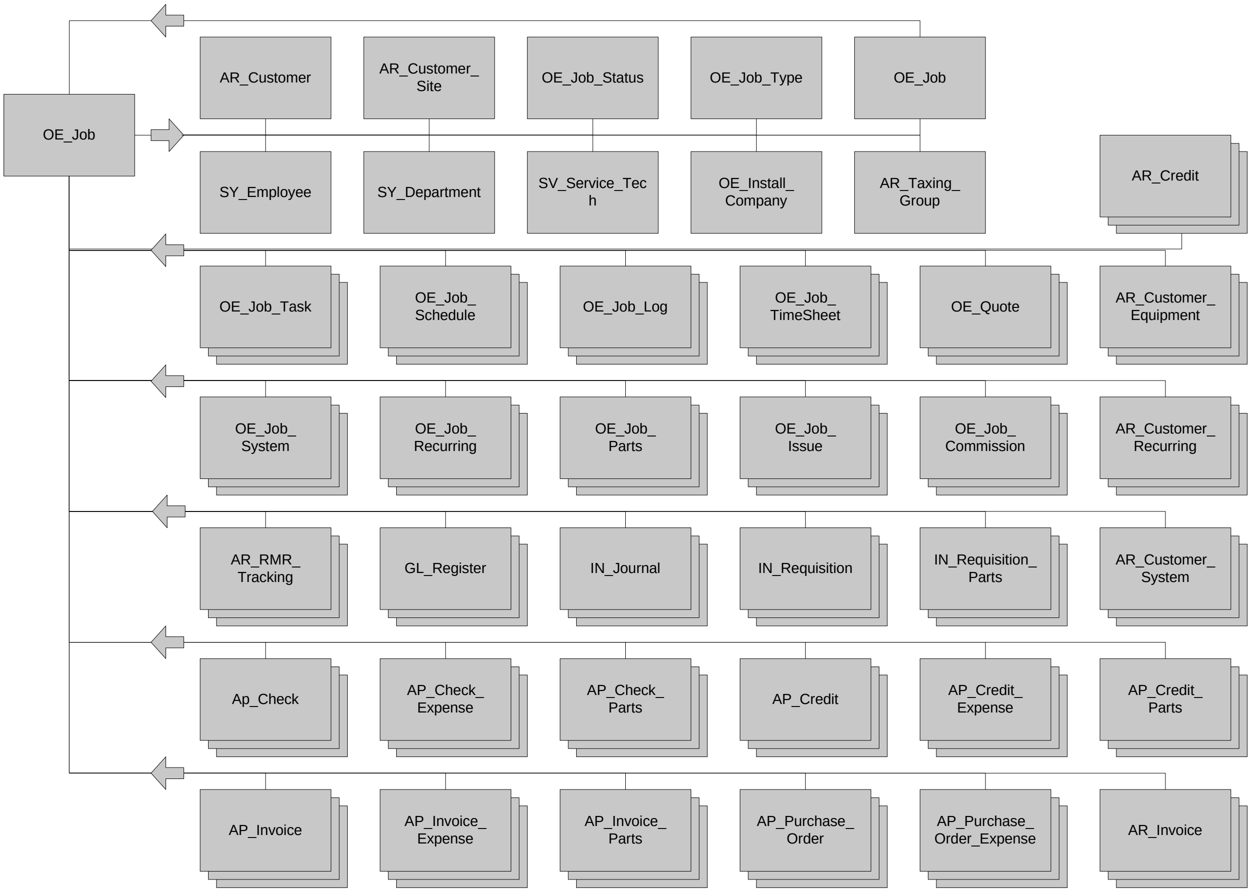
Vendor Bills Structure



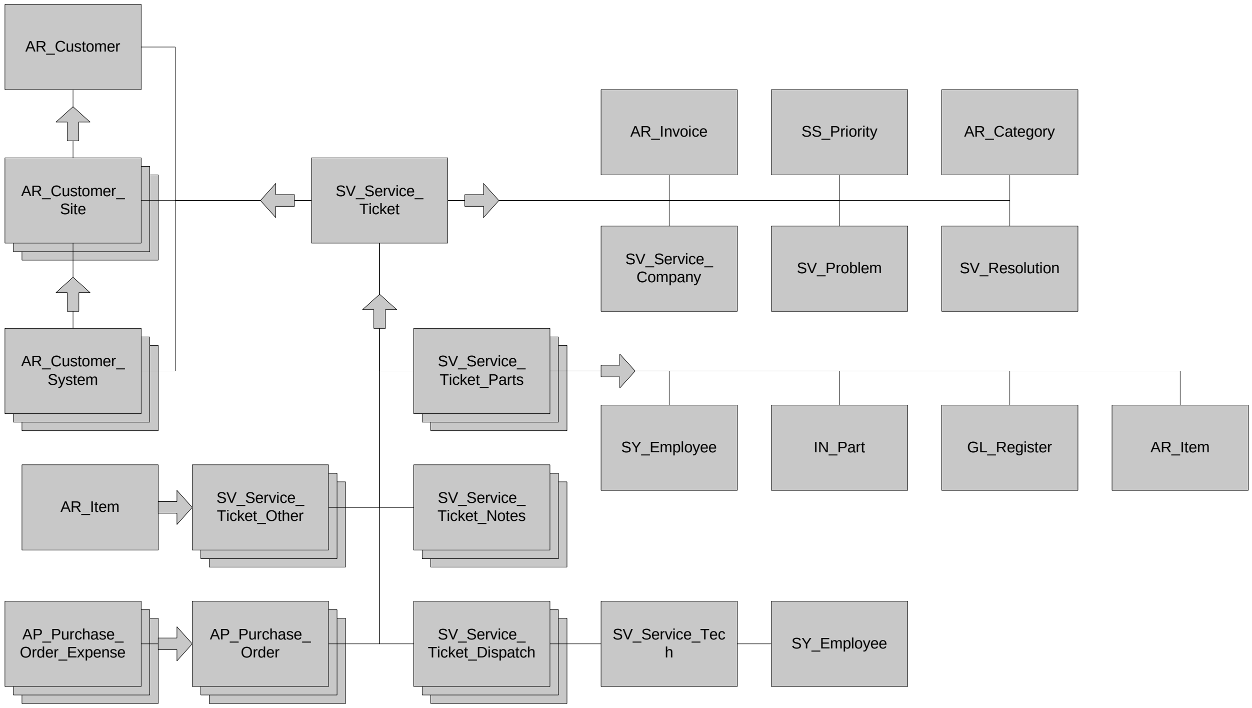
Check Structure



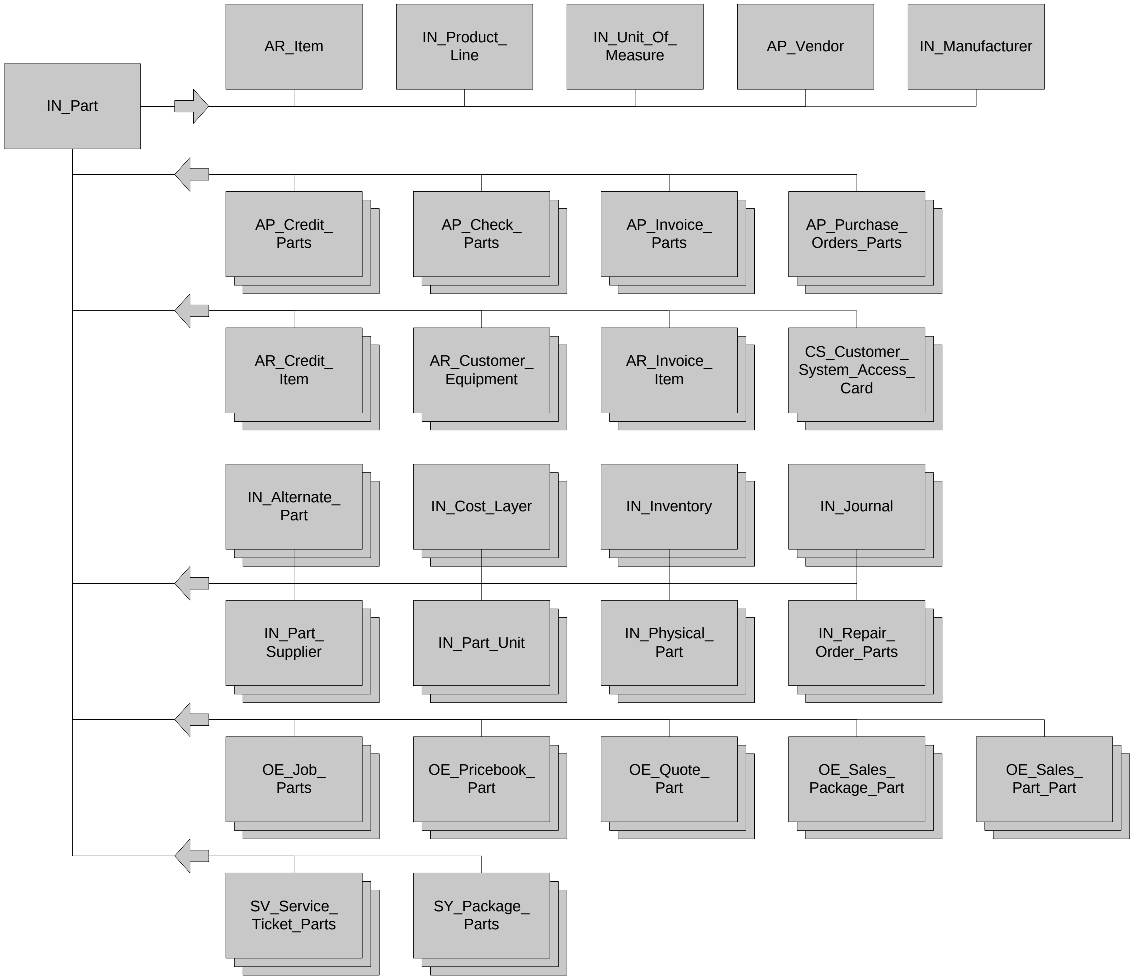
Job Structure



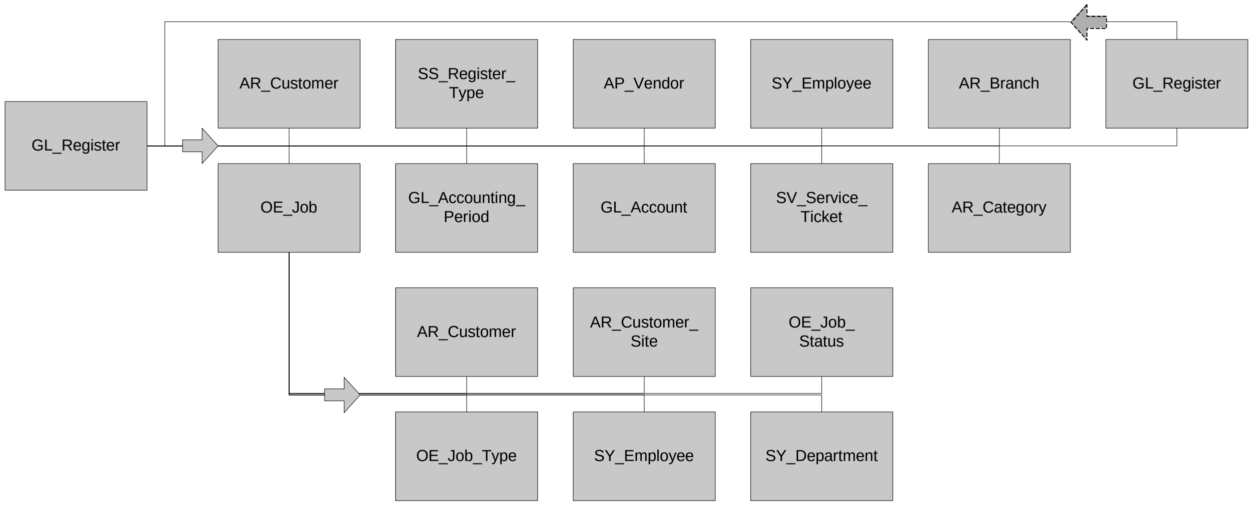
Service Ticket Structure



Inventory structure



General Ledger Structure

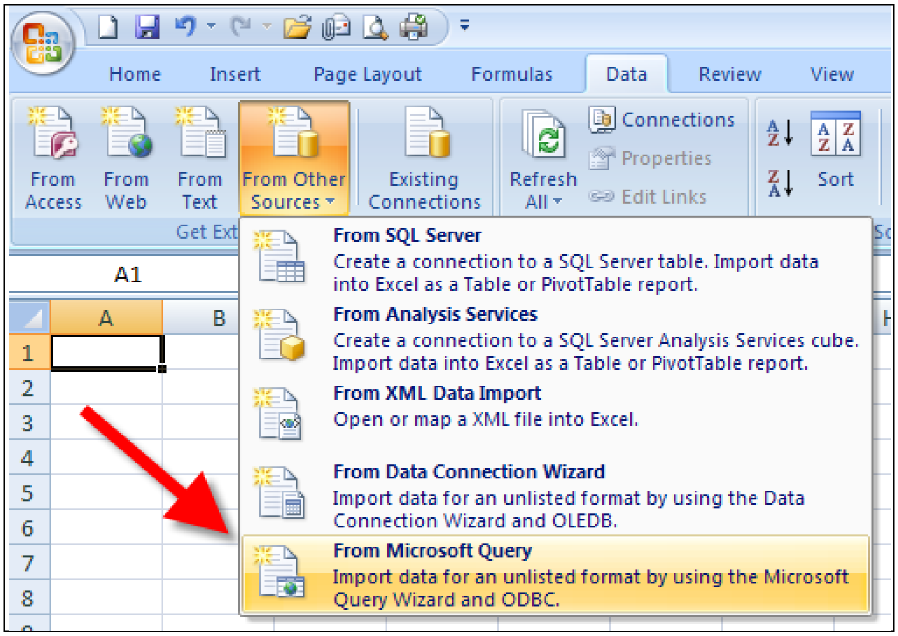


Open Database Connectivity (ODBC)

Open Database Connectivity is the methodology created by Microsoft for different applications to talk to different kinds of databases. With ODBC you can connect Excel to Microsoft SQL server or MS Word to Excel for example. The first step in connecting any application to your Microsoft SQL database is to create an ODBC connection. There is a utility for setting up ODBC connections. It is located in the Control Panel under ODBC. Many applications though contain an implementation of the ODBC Data Source Administrator. In our example we are going to use Excel to create an ODBC connection.

Creating an ODBC Connection with Excel

Let’s now review how to import Data into Microsoft Excel.   In this example we are going to use the feature in Excel to Query an External Data Source using Microsoft Query.   This feature is available in most recent versions of Excel but may needed to be installed as Excel does not install it by default in the standard install.



From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

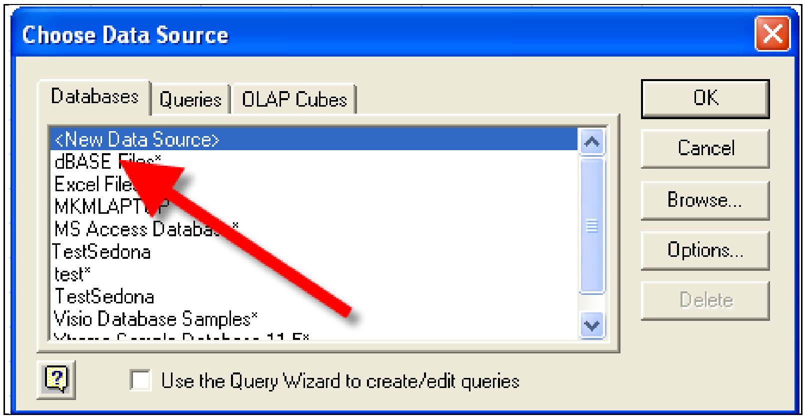
From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report.

From XML Data Import
Open or map a XML file into Excel.

From Data Connection Wizard
Import data for an unlisted format by using the Data Connection Wizard and OLEDB.

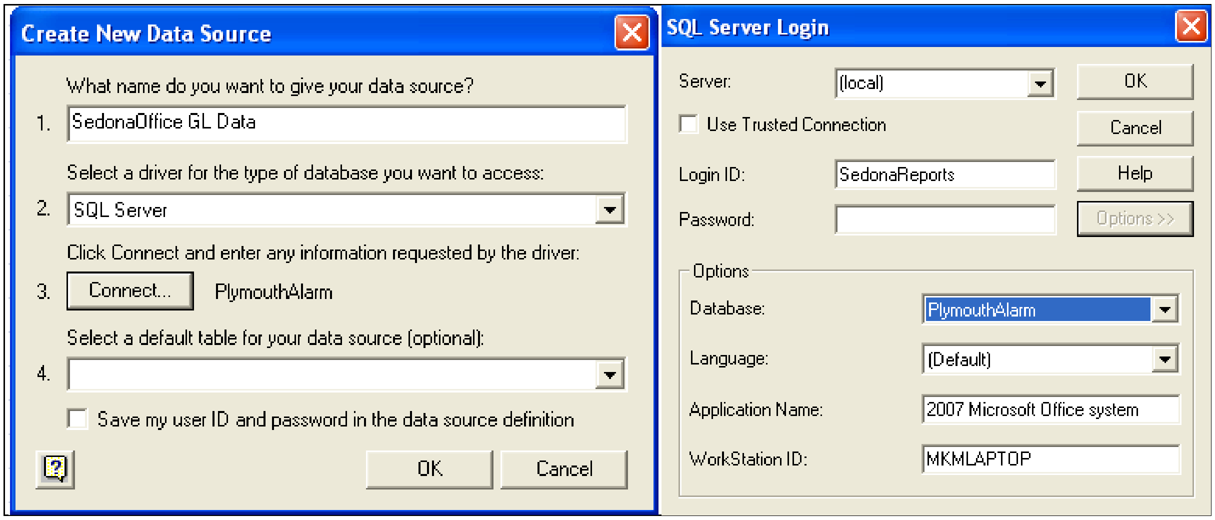
From Microsoft Query
Import data for an unlisted format by using the Microsoft Query Wizard and ODBC.

If you have not already done so, you will need to create a Data Source connection to your SedonaOffice database.



To create the new Data Source:

1) Name the data source appropriately (Here we are using “SedonaOffice GL Data” but the same connection can be used for all of your queries so you might want a more general name.)
2) Select ‘SQL Server’ as the driver to connect to the database
3) Press the Connect button
    a. On the SQL Server Login Screen select the name of the SQL Server for SedonaOffice
    b. Use “SedonaReports” as the Login ID, no password is needed
    c. Select the Options tab and select the name of your production SedonaOffice database
4) Press OK



Create New Data Source

What name do you want to give your data source?

1. SedonaOffice GL Data

Select a driver for the type of database you want to access:

2. SQL Server

Click Connect and enter any information requested by the driver:

3. Connect... PlymouthAlarm

Select a default table for your data source (optional):

4.

□ Save my user ID and password in the data source definition

SQL Server Login

Server: (local)

□ Use Trusted Connection

Login ID: SedonaReports

Password:

Options

Database: PlymouthAlarm

Language: (Default)

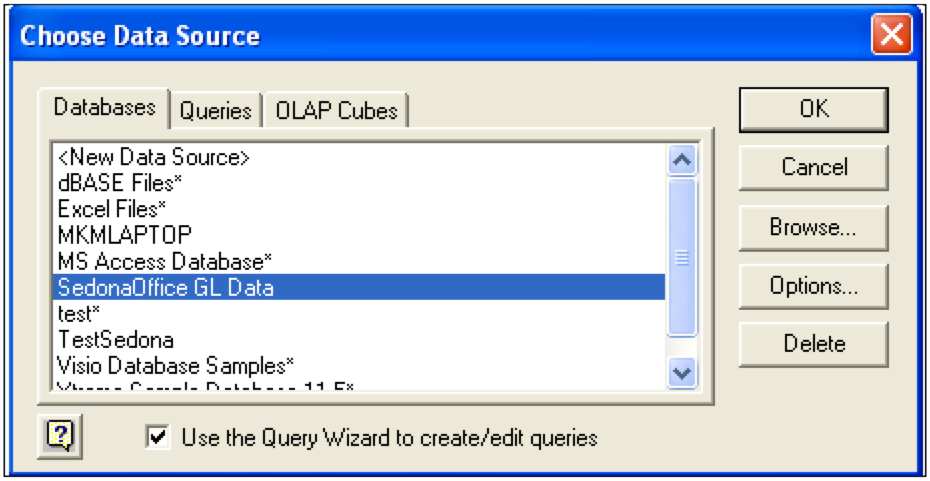
Application Name: 2007 Microsoft Office system

WorkStation ID: MKMLAPTOP

You now have an ODBC connection to your database.

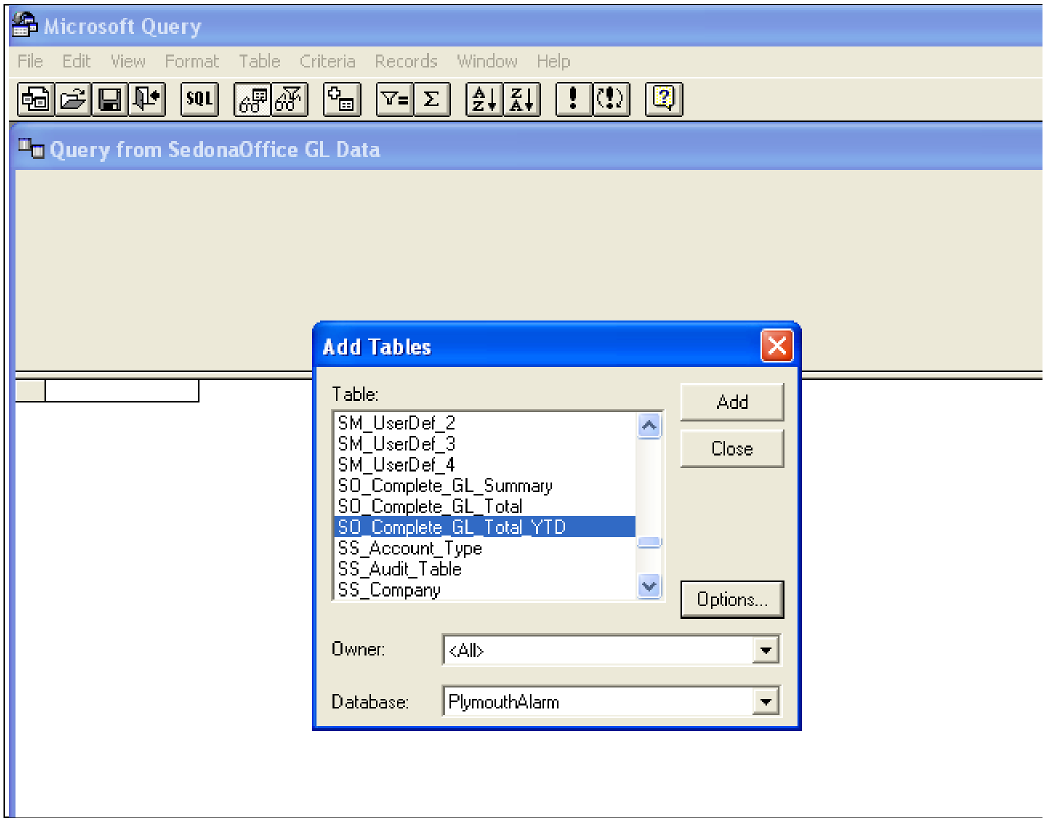
Building a Query Using Excel and MS Query

Select the data source you have previously created to create the Query. Uncheck the ‘Use the Query Wizard…” this will take you directly to Microsoft Query to create the Query.



☑ Use the Query Wizard to create/edit queries

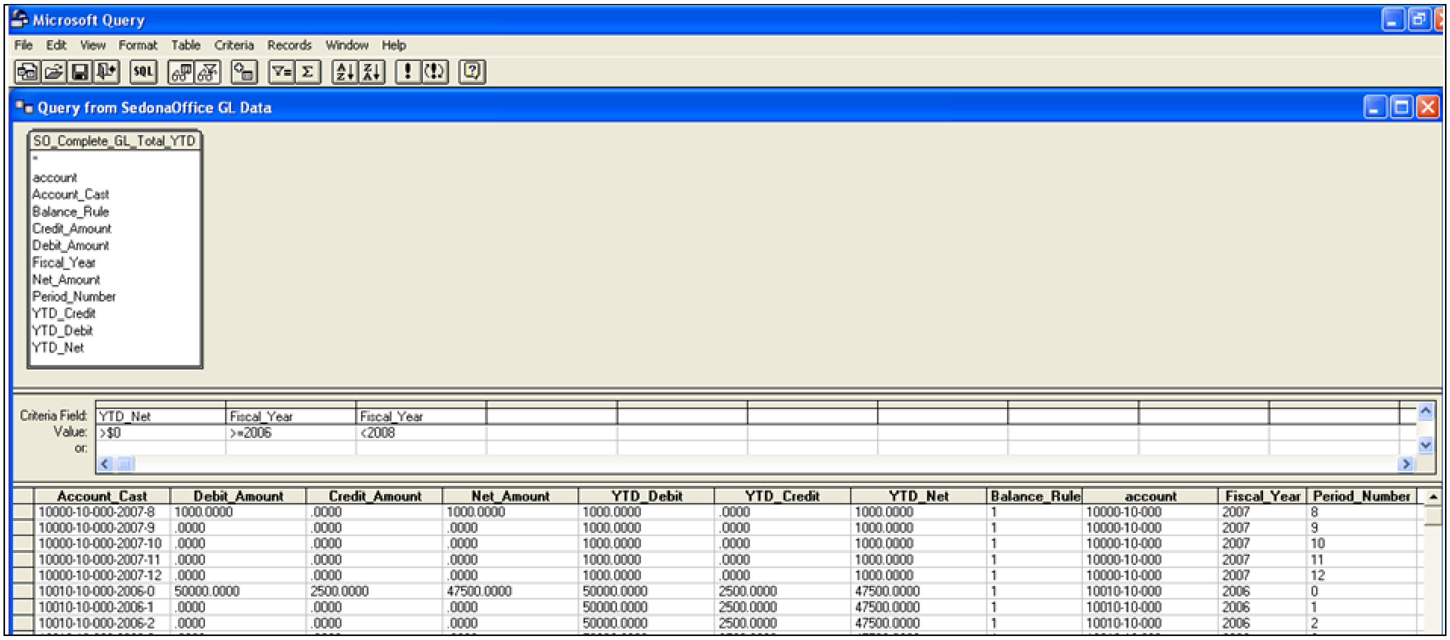
To begin with you need to select the Table file to use in the Query. Select the “SO_Complete_GL_Total_YTD” table. Then click Close.



Owner: <All>

Database: PlymouthAlarm

The next step is to select the data fields and criteria for the data to be returned. Select all the data elements in the Table. While it doesn't really matter what order to display the data fields, using the order as shown below will be more logical when viewed with Excel.



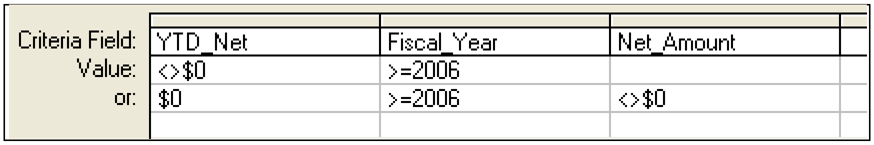
Criteria Field:	YTD_Net	Fiscal_Year	Fiscal_Year
Value:	>$0	>=2006	<2008
or:			

Account_Cast	Debit_Amount	Credit_Amount	Net_Amount	YTD_Debit	YTD_Credit	YTD_Net	Balance_Rule	account	Fiscal_Year	Period_Number
10000-10-000-2007-8	1000.0000	.0000	1000.0000	1000.0000	.0000	1000.0000	1	10000-10-000	2007	8
10000-10-000-2007-9	.0000	.0000	.0000	1000.0000	.0000	1000.0000	1	10000-10-000	2007	9
10000-10-000-2007-10	.0000	.0000	.0000	1000.0000	.0000	1000.0000	1	10000-10-000	2007	10
10000-10-000-2007-11	.0000	.0000	.0000	1000.0000	.0000	1000.0000	1	10000-10-000	2007	11
10000-10-000-2007-12	.0000	.0000	.0000	1000.0000	.0000	1000.0000	1	10000-10-000	2007	12
10010-10-000-2006-0	50000.0000	2500.0000	47500.0000	50000.0000	2500.0000	47500.0000	1	10010-10-000	2006	0
10010-10-000-2006-1	.0000	.0000	.0000	50000.0000	2500.0000	47500.0000	1	10010-10-000	2006	1
10010-10-000-2006-2	.0000	.0000	.0000	50000.0000	2500.0000	47500.0000	1	10010-10-000	2006	2

Since this table can contain thousands (hundreds of thousands of records) it is best to use some criteria to limit the data that returns.

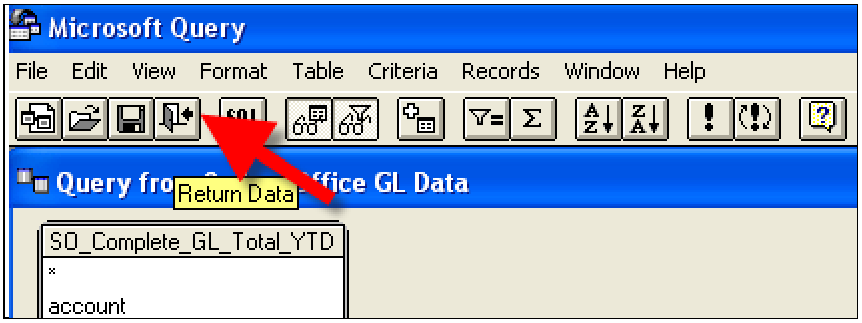
Criteria Selections:

1) YTD_Net <> $0 – By selecting this option only data with values will be returned.
2) Fiscal Year >= 2006 – In this case only years 2006 and 2007 are needed so limit the data to only these fiscal years.
3) Fiscal Year < 2008 – In this case since 2008 has been created we can remove these entries since were still reporting on 2007.
4) Net_Amount <>$0 – This is included as an 'OR' selection. This is necessary to return the Retained Earnings account (more on this later).

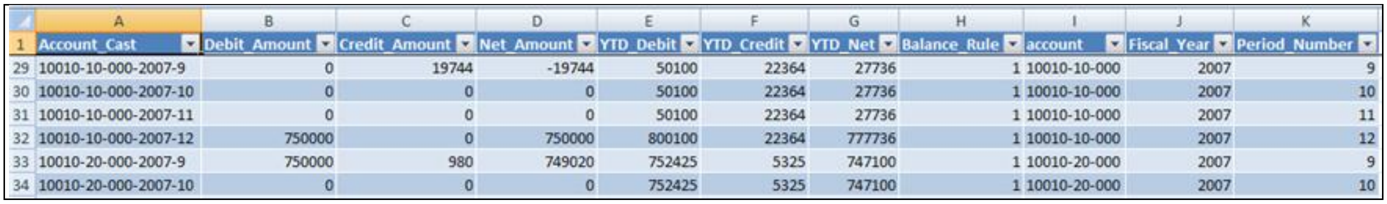


Criteria Field:	YTD_Net	Fiscal_Year	Net_Amount
Value:	<>$0	>=2006	
or:	$0	>=2006	<>$0

Now that we have completed the Query, click the Return Data icon, and the GL Data will be returned to Excel.



Your data will be returned to Excel.



	A	B	C	D	E	F	G	H	I	J	K
1	Account_Cast	Debit_Amount	Credit_Amount	Net_Amount	YTD_Debit	YTD_Credit	YTD_Net	Balance_Rule	account	Fiscal_Year	Period_Number
29	10010-10-000-2007-9	0	19744	-19744	50100	22364	27736	1	10010-10-000	2007	9
30	10010-10-000-2007-10	0	0	0	50100	22364	27736	1	10010-10-000	2007	10
31	10010-10-000-2007-11	0	0	0	50100	22364	27736	1	10010-10-000	2007	11
32	10010-10-000-2007-12	750000	0	750000	800100	22364	777736	1	10010-10-000	2007	12
33	10010-20-000-2007-9	750000	980	749020	752425	5325	747100	1	10010-20-000	2007	9
34	10010-20-000-2007-10	0	0	0	752425	5325	747100	1	10010-20-000	2007	10

Using Microsoft Access to Review Your Data

Why use Access instead of Excel? Excel has a row limit. The maximum number of rows you can have in a spreadsheet varies with what version you use from as little as 32,767 rows for older versions to 1,048,576 rows in Excel 2007. This may seem like a lot of rows and for most queries it will be sufficient. But, queries involving the GL_Register for a company that has several years of history can easily exceed these limitations.

Excel treats all fields containing only numeric characters (0-9) as numbers unless prefaced with a ' character. By treating things like postal codes as numbers postal codes starting with a 0 are truncated. Thus a postal code of 01234 becomes 1234.

Finally, Access has a built in report generator. With Access you can make complex reports with groups, subtotals, totals, etc.

****** Caution *** ONLY use SedonaReports for an ODBC connection to Access. Otherwise changes you make in Access can change your SQL Server data and corrupt your database.***

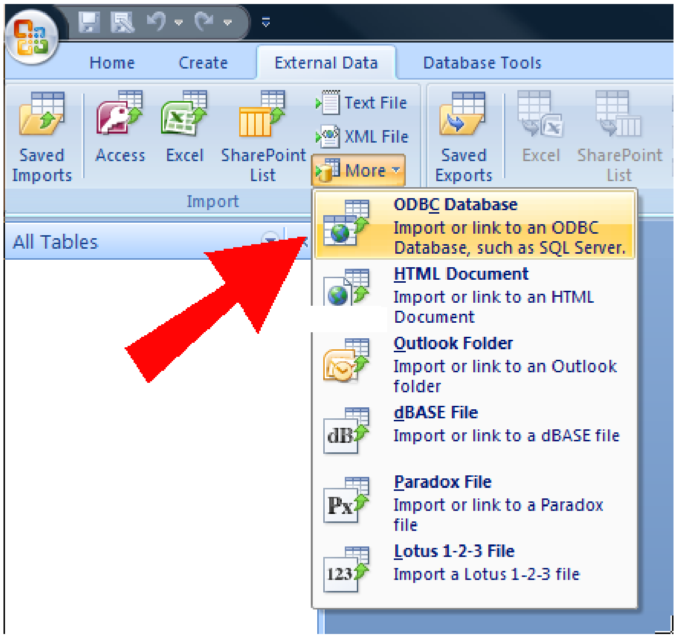
Connecting Access via ODBC

When using an ODBC connection with Access you have two options on how to connect the data, Import or Link.

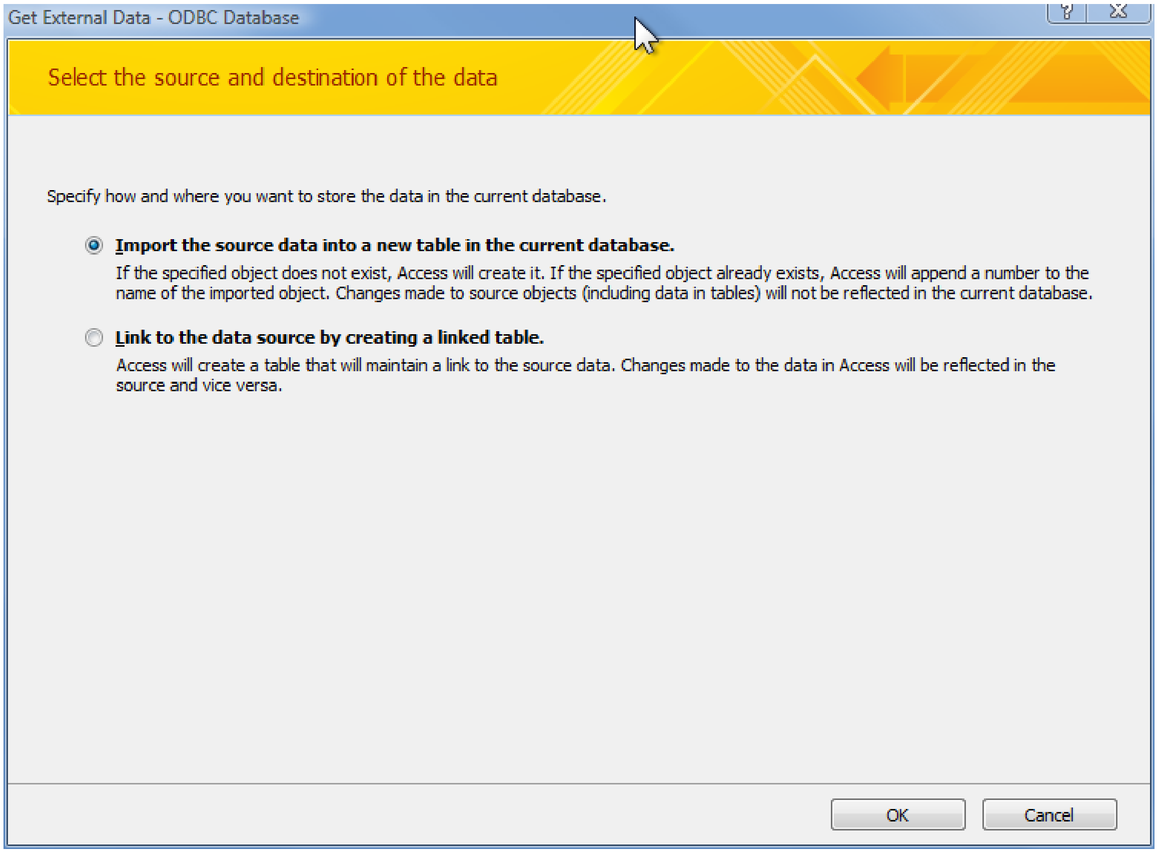
When you Import data into Access, you create a copy of the data stored within the Access database. This allows you to review the data when not connected to the database. Like Excel, you have to periodically refresh the data to keep it up to date.

When you Link data to Access, the data remains in the SQL Server but Access can use it in queries and reports. This method is constantly refreshes as the data in the SQL server changes but it will not function if it is disconnected from the SQL Server.

Choose the External Data tab. Then choose More. Finally choose ODBC Database.



Choose Import or Link and then click OK.



Get External Data - ODBC Database

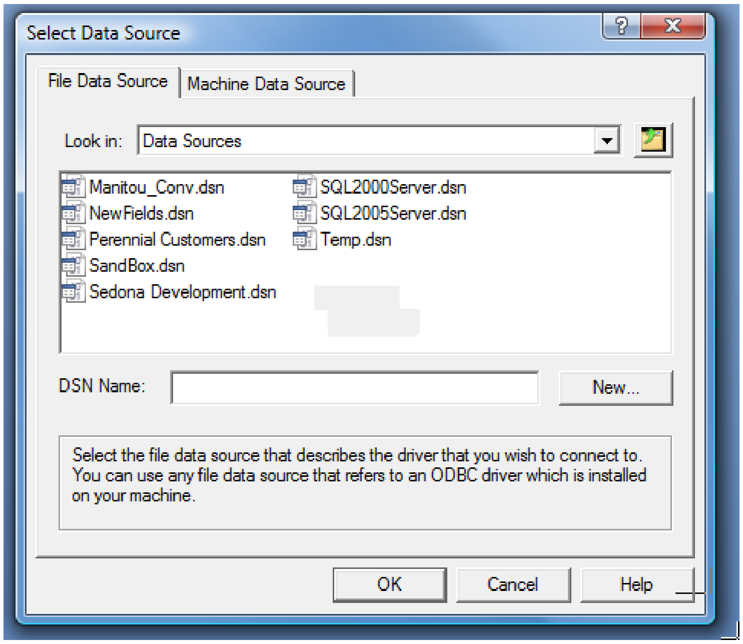
Select the source and destination of the data

Specify how and where you want to store the data in the current database.

● **Import the source data into a new table in the current database.**
If the specified object does not exist, Access will create it. If the specified object already exists, Access will append a number to the name of the imported object. Changes made to source objects (including data in tables) will not be reflected in the current database.

○ **Link to the data source by creating a linked table.**
Access will create a table that will maintain a link to the source data. Changes made to the data in Access will be reflected in the source and vice versa.

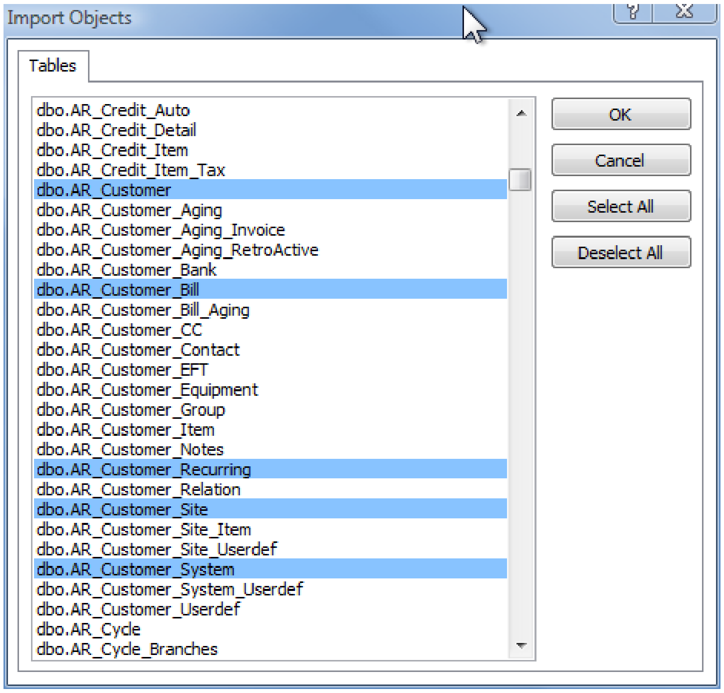
Choose your Data Source.



Manitou_Conv.dsn
NewFields.dsn
Perennial Customers.dsn
SandBox.dsn
Sedona Development.dsn
SQL2000Server.dsn
SQL2005Server.dsn
Temp.dsn

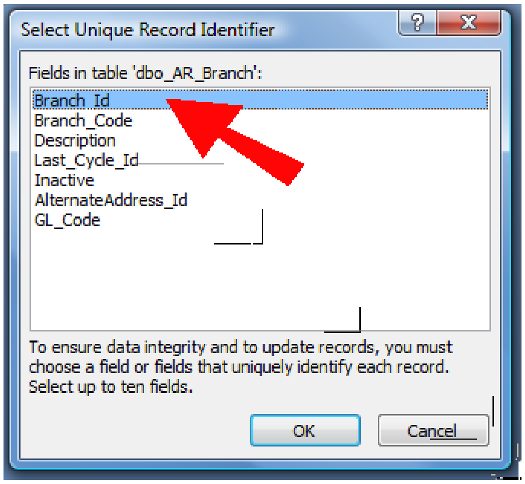
Select the file data source that describes the driver that you wish to connect to. You can use any file data source that refers to an ODBC driver which is installed on your machine.

Then choose the tables you wish to Import or Link and click OK. You can choose multiple tables but do not select all. Access is not as large or as powerful as SQL Server. Choosing all will probably crash Access.



dbo.AR_Credit_Auto
dbo.AR_Credit_Detail
dbo.AR_Credit_Item
dbo.AR_Credit_Item_Tax
dbo.AR_Customer
dbo.AR_Customer_Aging
dbo.AR_Customer_Aging_Invoice
dbo.AR_Customer_Aging_RetroActive
dbo.AR_Customer_Bank
dbo.AR_Customer_Bill
dbo.AR_Customer_Bill_Aging
dbo.AR_Customer_CC
dbo.AR_Customer_Contact
dbo.AR_Customer_EFT
dbo.AR_Customer_Equipment
dbo.AR_Customer_Group
dbo.AR_Customer_Item
dbo.AR_Customer_Notes
dbo.AR_Customer_Recurring
dbo.AR_Customer_Relation
dbo.AR_Customer_Site
dbo.AR_Customer_Site_Item
dbo.AR_Customer_Site_Userdef
dbo.AR_Customer_System
dbo.AR_Customer_System_Userdef
dbo.AR_Customer_Userdef
dbo.AR_Cycle
dbo.AR_Cycle_Branches

If you chose to Link you will be asked to Select Unique Record Indicator. This is always the top item in SedonaOffice.



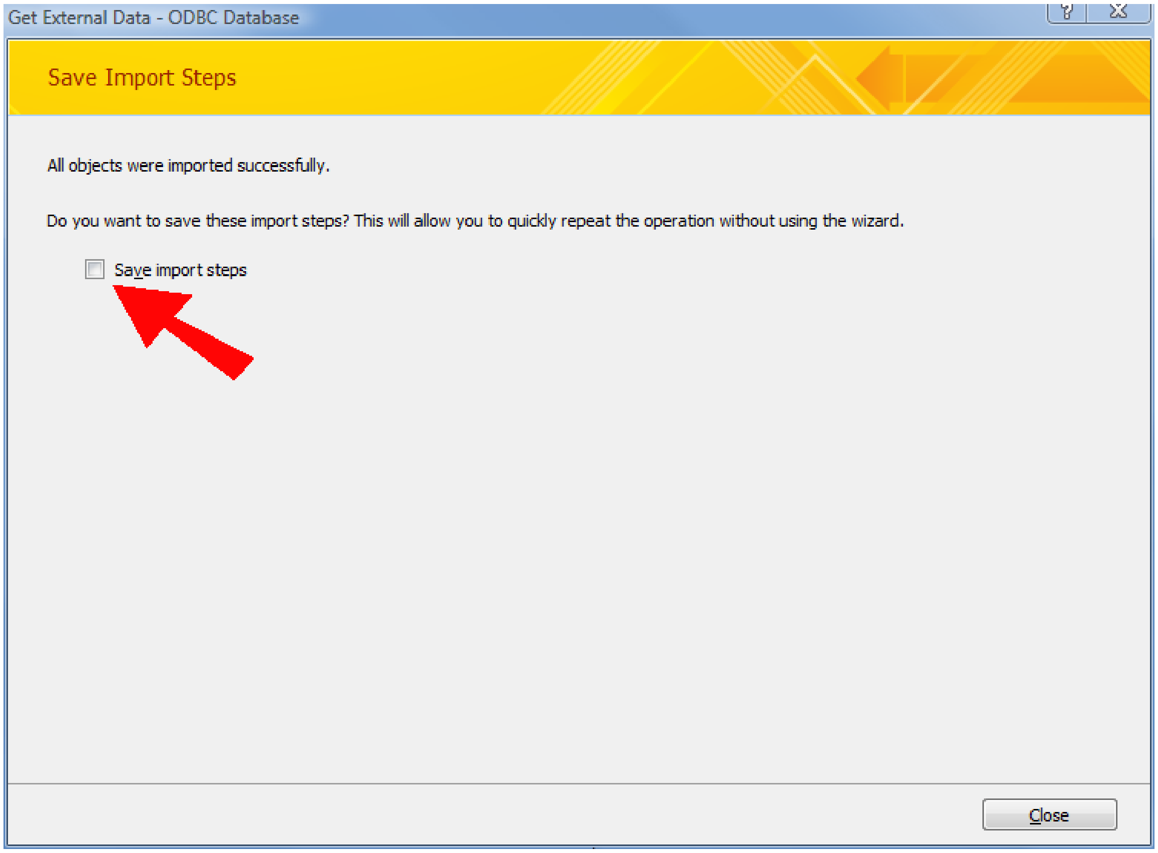
Select Unique Record Identifier

Fields in table 'dbo_AR_Branch':

Branch_Id
Branch_Code
Description
Last_Cycle_Id
Inactive
AlternateAddress_Id
GL_Code

To ensure data integrity and to update records, you must choose a field or fields that uniquely identify each record. Select up to ten fields.

If you chose Import, when the operation is complete a window will be displayed showing the success of the operation. Here you can also Save the steps you just did so refreshing the data will be easier.



Get External Data - ODBC Database

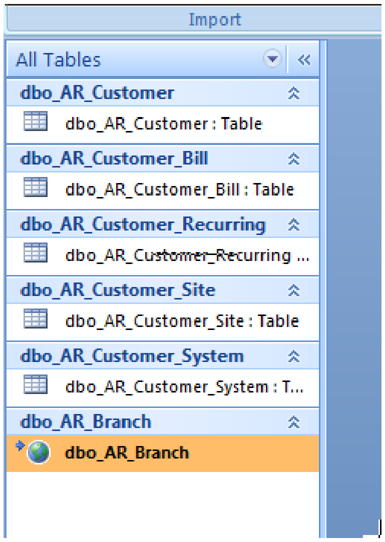
Save Import Steps

All objects were imported successfully.

Do you want to save these import steps? This will allow you to quickly repeat the operation without using the wizard.

□ Save import steps

Your tables will then be accessible in Access. You may mix Import and Link in the same Access database. In the example I have Imported several customer tables and linked the branch table. Notice the different icons for imported versus linked tables. The highlighted table is the linked branch table.



Import

All Tables

dbo_AR_Customer
dbo_AR_Customer : Table

dbo_AR_Customer_Bill
dbo_AR_Customer_Bill : Table

dbo_AR_Customer_Recurring
dbo_AR_Customer_Recurring ...

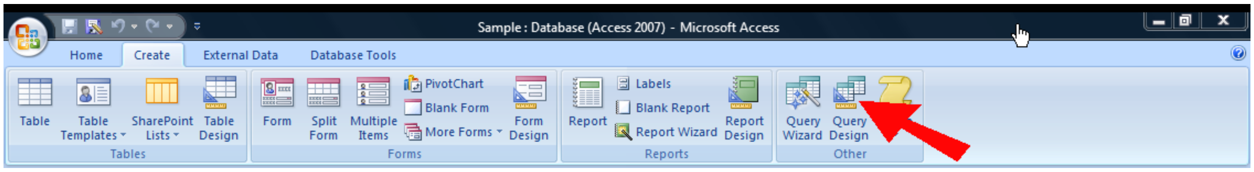
dbo_AR_Customer_Site
dbo_AR_Customer_Site : Table

dbo_AR_Customer_System
dbo_AR_Customer_System : T...

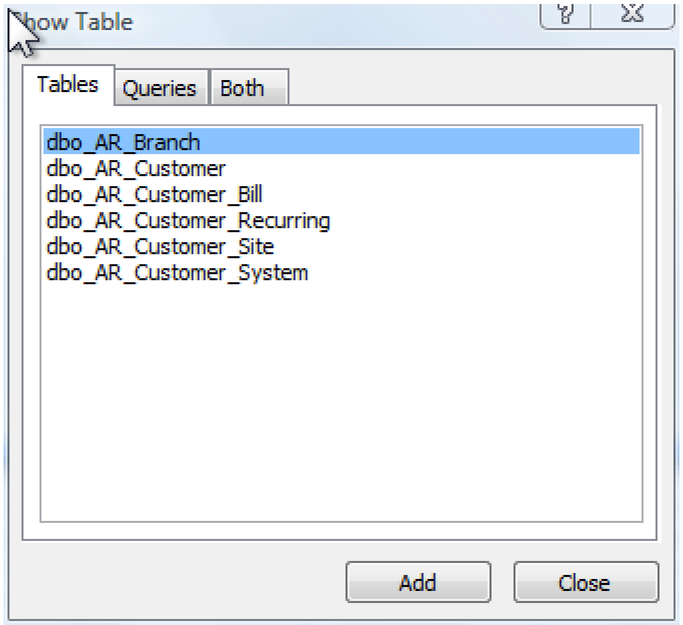
dbo_AR_Branch
dbo_AR_Branch

Writing a query with Access

Click on the Create tab and then on Query design.



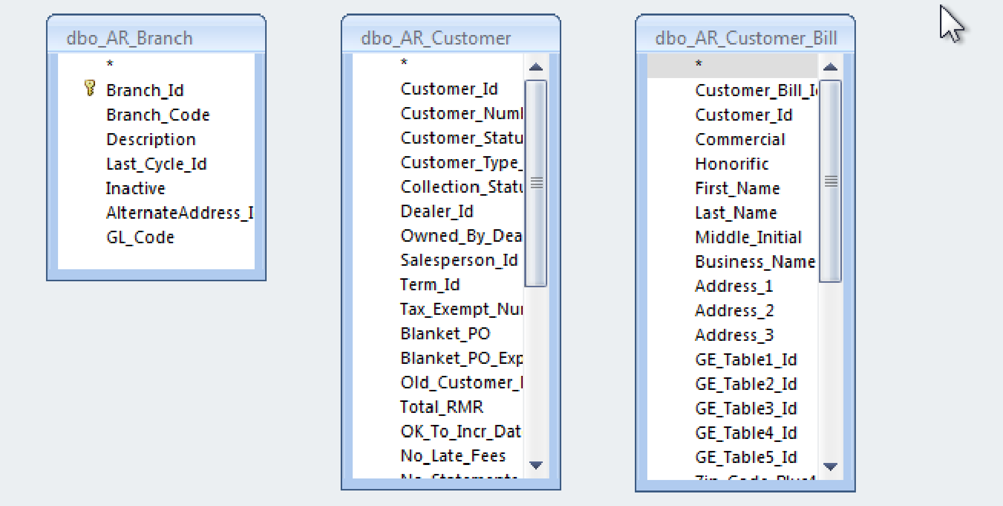
Choose the tables you wish to include in your query. A table can be selected more than once if you need to join it to more than one Id. For our example we are going to choose all of the tables.



Show Table

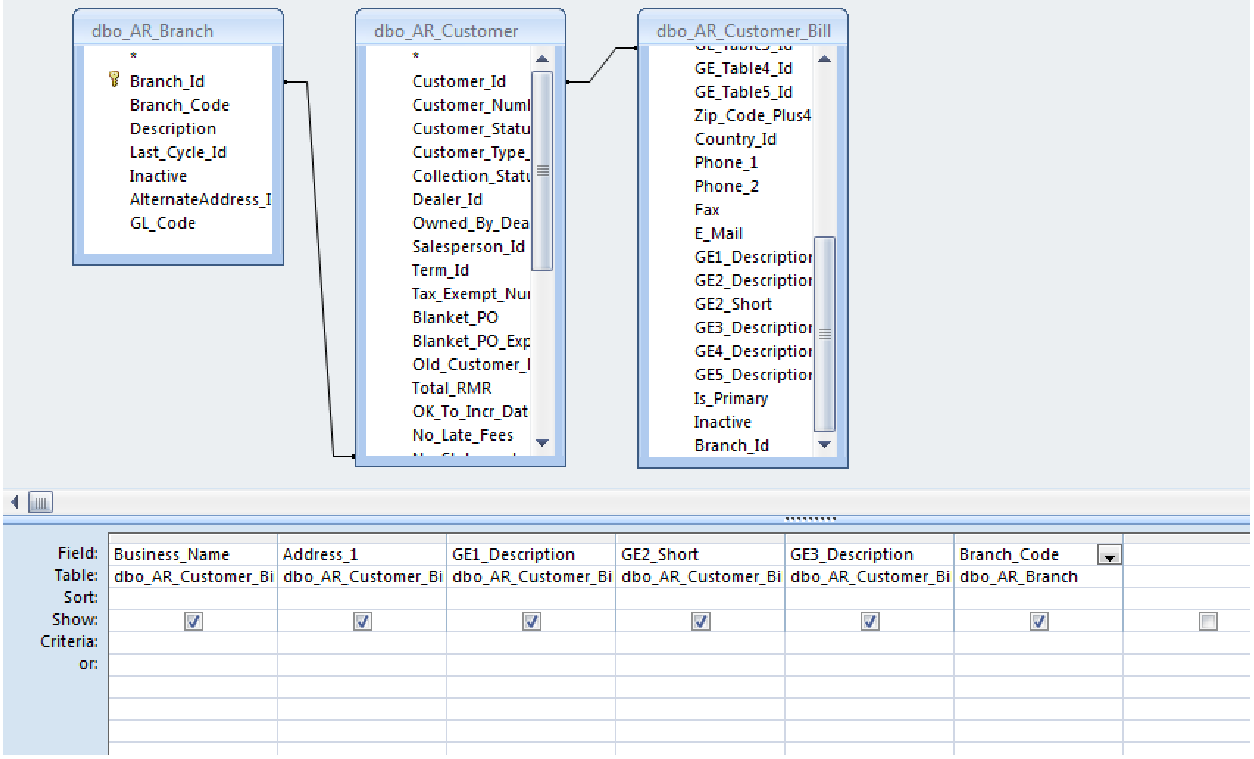
dbo_AR_Branch
dbo_AR_Customer
dbo_AR_Customer_Bill
dbo_AR_Customer_Recurring
dbo_AR_Customer_Site
dbo_AR_Customer_System

Delete all of the joins that access automatically creates.



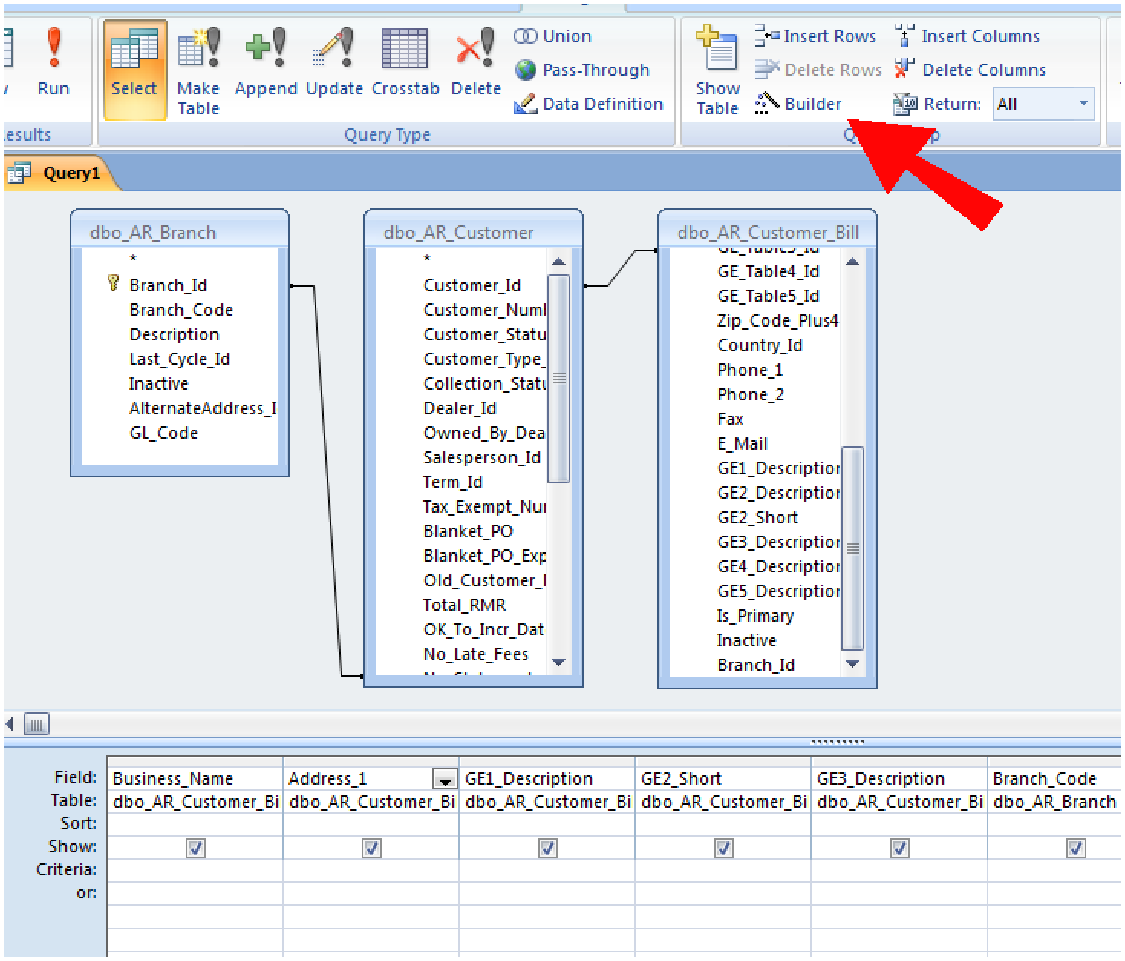
Create the joins according to the structure of SedonaOffice. In this case AR_Branch.Branch_Id to AR_Customer.Branch_Id and AR_Customer.Customer_Id to AR_Customer_Bill.Customer_id.

We are going to create a mailing list so we need to drag the name and address information to the lower pane. We are also going to drag down the branch code so we can sort on branch.



Field:	Business_Name	Address_1	GE1_Description	GE2_Short	GE3_Description	Branch_Code	
Table:	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Branch	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	☐
Criteria:							
or:							

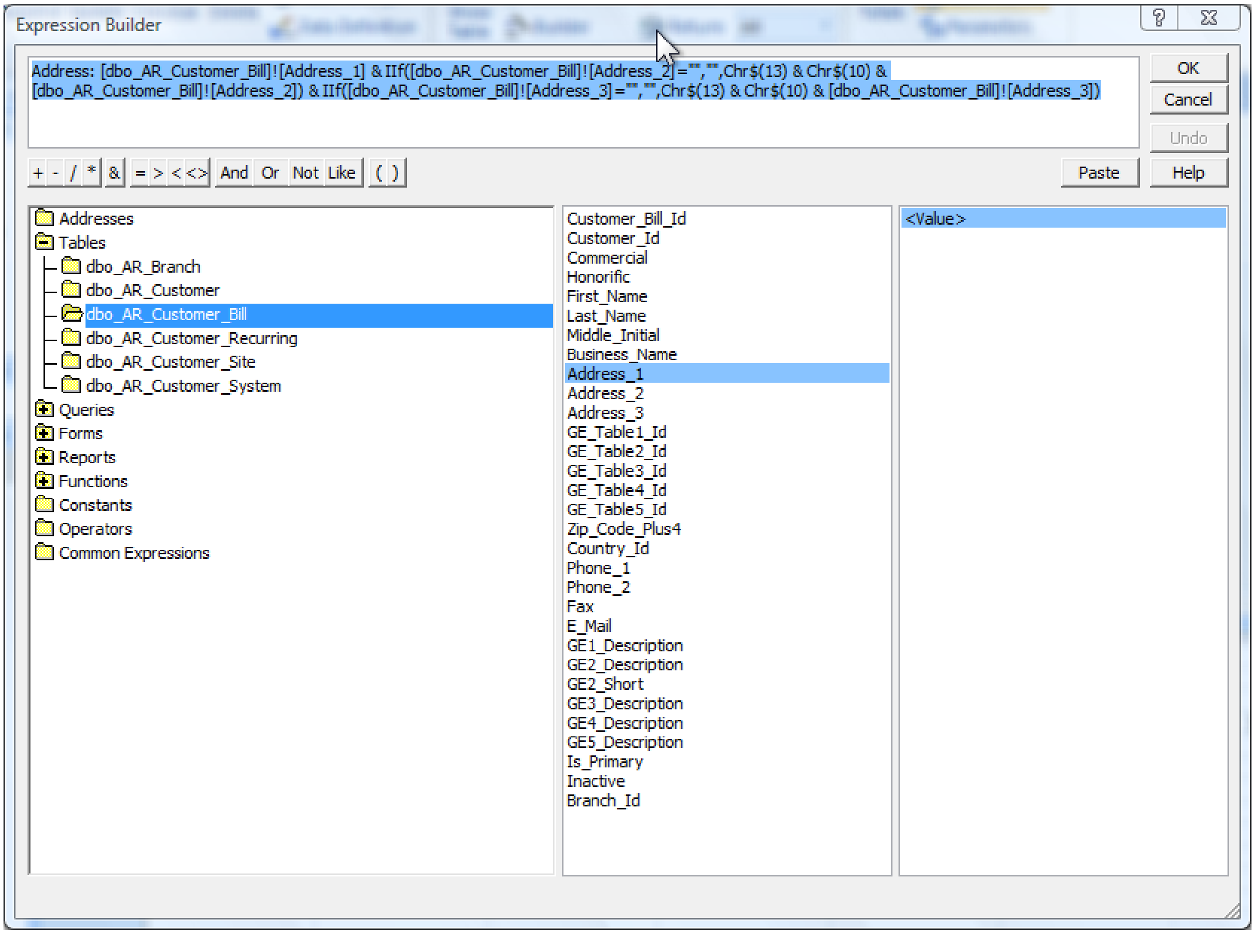
Address_1 may not be all of the address information needed but if there is no address_2 we don't want to add a blank line. So we create a formula. Click in the Address_1 cell and then click 0n the formula button.



Field:	Business_Name	Address_1	GE1_Description	GE2_Short	GE3_Description	Branch_Code
Table:	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Branch
Sort:						
Show:	☑	☑	☑	☑	☑	☑
Criteria:						
or:						

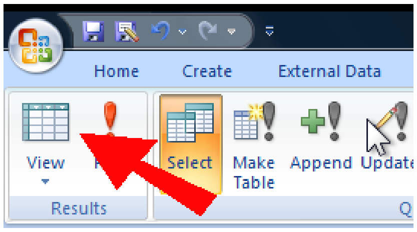
Enter the following into the builder window.

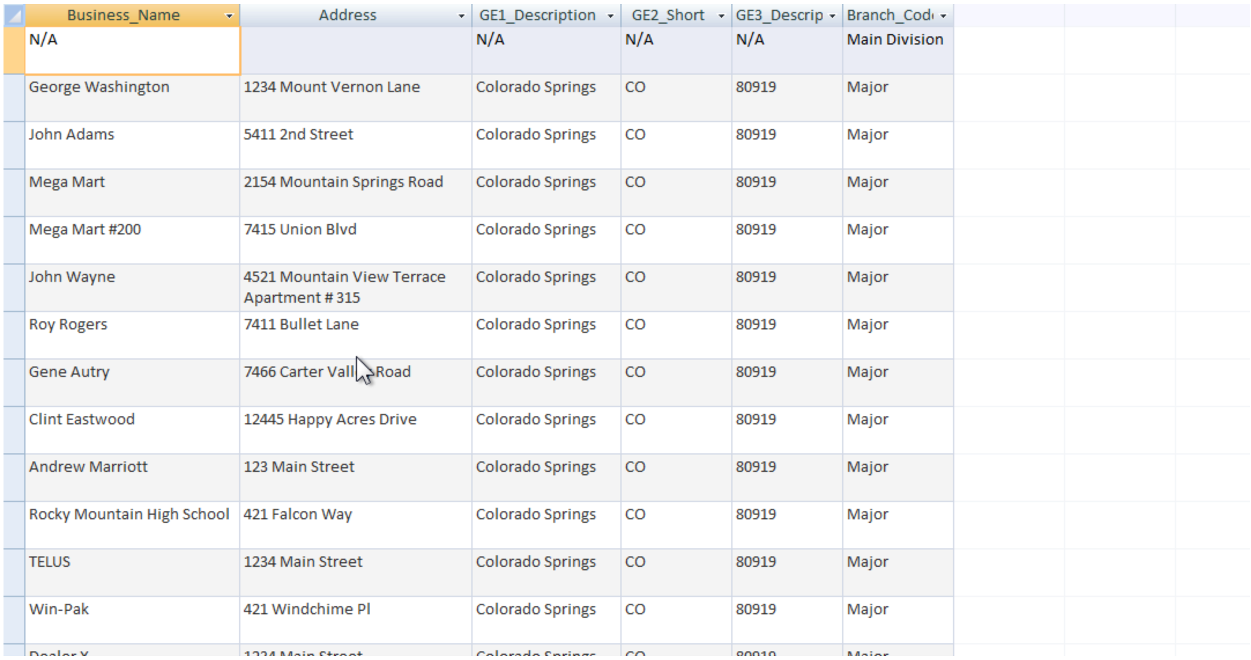
Address: [dbo_AR_Customer_Bill]![Address_1] & IIf([dbo_AR_Customer_Bill]![Address_2]= "","",Chr$(13) & Chr$(10) & [dbo_AR_Customer_Bill]![Address_2]) & IIf([dbo_AR_Customer_Bill]![Address_3]= "","",Chr$(13) & Chr$(10) & [dbo_AR_Customer_Bill]![Address_3])



```
Address: [dbo_AR_Customer_Bill]![Address_1] & IIf([dbo_AR_Customer_Bill]![Address_2]="","",Chr$(13) & Chr$(10) & [dbo_AR_Customer_Bill]![Address_2]) & IIf([dbo_AR_Customer_Bill]![Address_3]="","",Chr$(13) & Chr$(10) & [dbo_AR_Customer_Bill]![Address_3])
```

Click view to test our results.

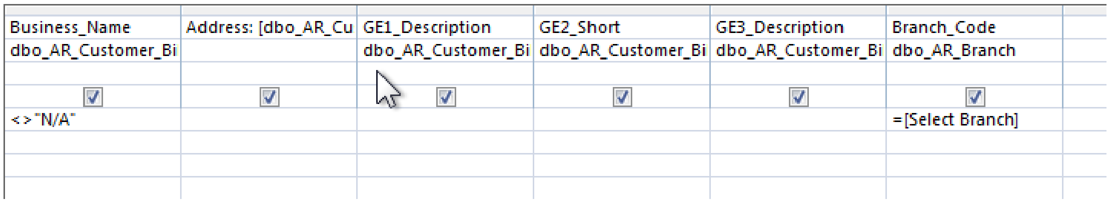




Business_Name	Address	GE1_Description	GE2_Short	GE3_Descrip	Branch_Cod
N/A		N/A	N/A	N/A	Main Division
George Washington	1234 Mount Vernon Lane	Colorado Springs	CO	80919	Major
John Adams	5411 2nd Street	Colorado Springs	CO	80919	Major
Mega Mart	2154 Mountain Springs Road	Colorado Springs	CO	80919	Major
Mega Mart #200	7415 Union Blvd	Colorado Springs	CO	80919	Major
John Wayne	4521 Mountain View Terrace Apartment # 315	Colorado Springs	CO	80919	Major
Roy Rogers	7411 Bullet Lane	Colorado Springs	CO	80919	Major
Gene Autry	7466 Carter Vall Road	Colorado Springs	CO	80919	Major
Clint Eastwood	12445 Happy Acres Drive	Colorado Springs	CO	80919	Major
Andrew Marriott	123 Main Street	Colorado Springs	CO	80919	Major
Rocky Mountain High School	421 Falcon Way	Colorado Springs	CO	80919	Major
TELUS	1234 Main Street	Colorado Springs	CO	80919	Major
Win-Pak	421 Windchime Pl	Colorado Springs	CO	80919	Major
Dealer X	1234 Main Street	Colorado Springs	CO	80919	Major

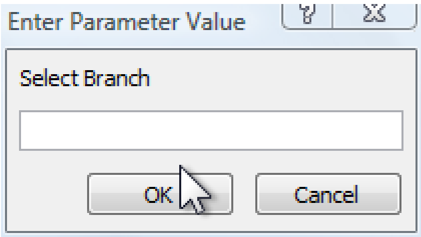
Now, let’s remove the N/A row and add a method to select which branch we want.

Under business_Name add <>”N/A”. Then under Branch_Code add =[Select Branch].



Business_Name	Address: [dbo_AR_Cu	GE1_Description	GE2_Short	GE3_Description	Branch_Code
dbo_AR_Customer_Bi		dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi	dbo_AR_Branch
☑	☑	☑	☑	☑	☑
<>"N/A"					=[Select Branch]

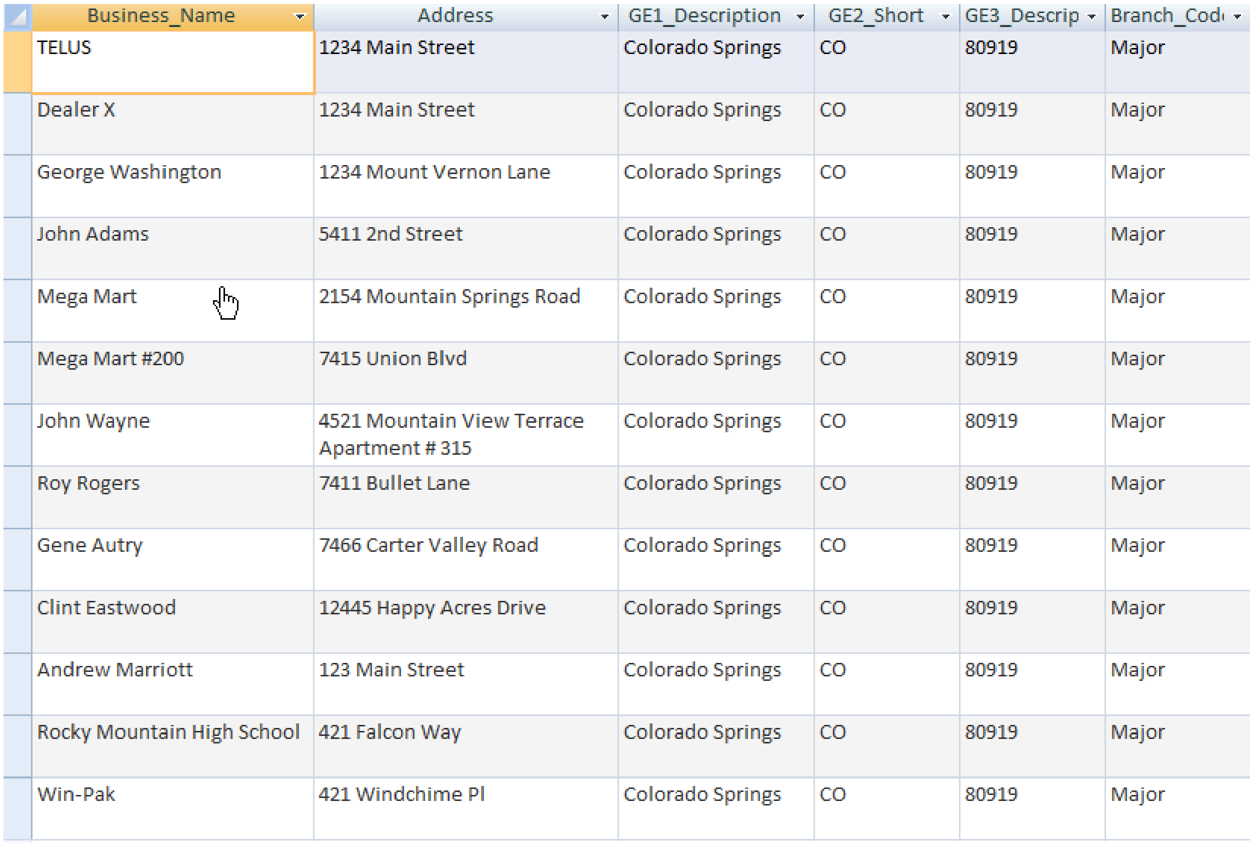
Now when we return the results we are asked to select a Branch.



Enter Parameter Value

Select Branch

Entering a branch we get results with no N/A.

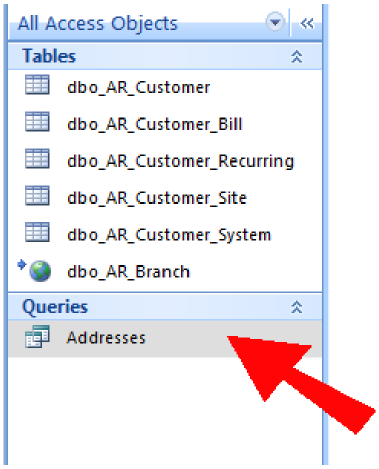


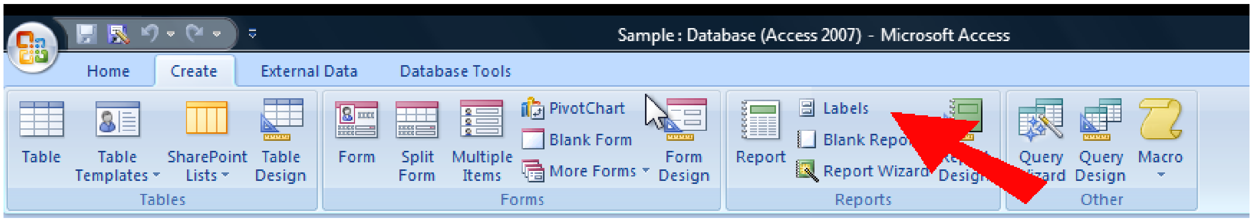
Business_Name	Address	GE1_Description	GE2_Short	GE3_Descrip	Branch_Cod
TELUS	1234 Main Street	Colorado Springs	CO	80919	Major
Dealer X	1234 Main Street	Colorado Springs	CO	80919	Major
George Washington	1234 Mount Vernon Lane	Colorado Springs	CO	80919	Major
John Adams	5411 2nd Street	Colorado Springs	CO	80919	Major
Mega Mart	2154 Mountain Springs Road	Colorado Springs	CO	80919	Major
Mega Mart #200	7415 Union Blvd	Colorado Springs	CO	80919	Major
John Wayne	4521 Mountain View Terrace Apartment # 315	Colorado Springs	CO	80919	Major
Roy Rogers	7411 Bullet Lane	Colorado Springs	CO	80919	Major
Gene Autry	7466 Carter Valley Road	Colorado Springs	CO	80919	Major
Clint Eastwood	12445 Happy Acres Drive	Colorado Springs	CO	80919	Major
Andrew Marriott	123 Main Street	Colorado Springs	CO	80919	Major
Rocky Mountain High School	421 Falcon Way	Colorado Springs	CO	80919	Major
Win-Pak	421 Windchime Pl	Colorado Springs	CO	80919	Major

Creating a Report with Access

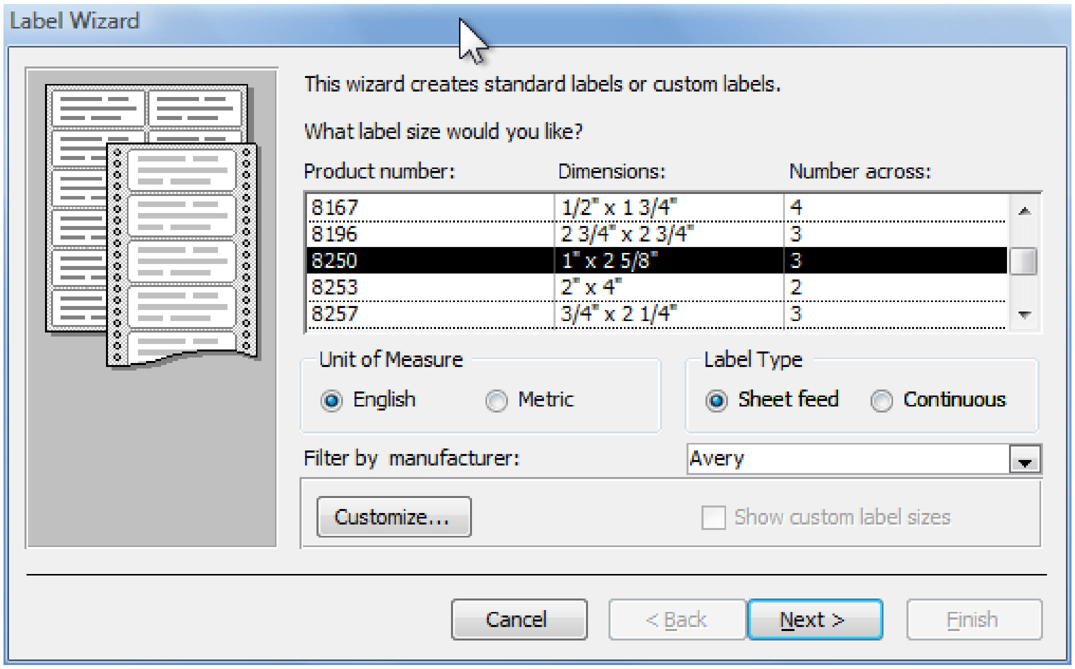
Displaying the results on the screen is useful but Access allows us to create reports. The report we are going to create will be to create mailing labels.

First make sure the new query you created is selected and then launch the label wizard.





Choose your label. You can choose by the form number if you bought labels from a major manufacturer or just choose a label of the same size as the ones you are using.



Label Wizard

This wizard creates standard labels or custom labels.

What label size would you like?

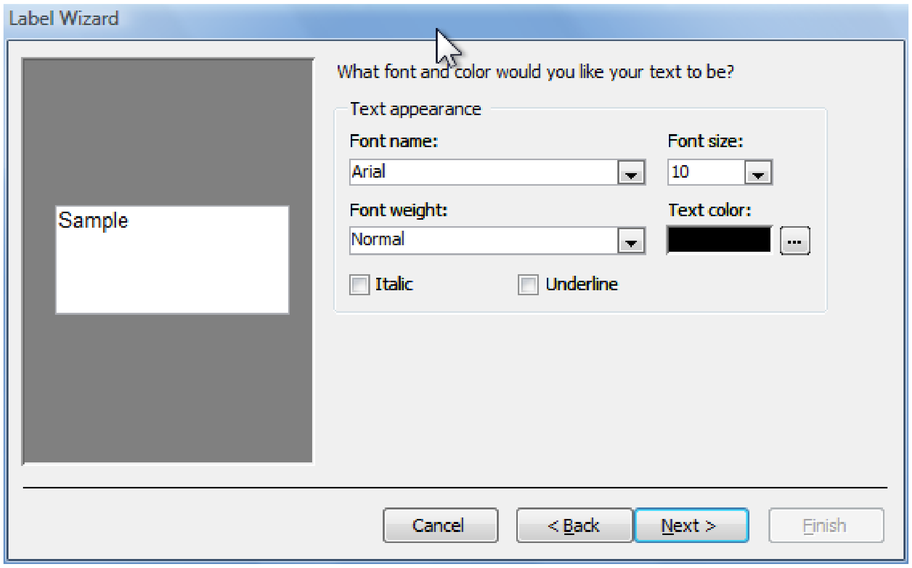
Product number:	Dimensions:	Number across:
8167	1/2" x 1 3/4"	4
8196	2 3/4" x 2 3/4"	3
8250	1" x 2 5/8"	3
8253	2" x 4"	2
8257	3/4" x 2 1/4"	3

Unit of Measure: ● English ○ Metric

Label Type: ● Sheet feed ○ Continuous

Filter by manufacturer: Avery

Choose your font.



What font and color would you like your text to be?

Text appearance

Font name: Arial

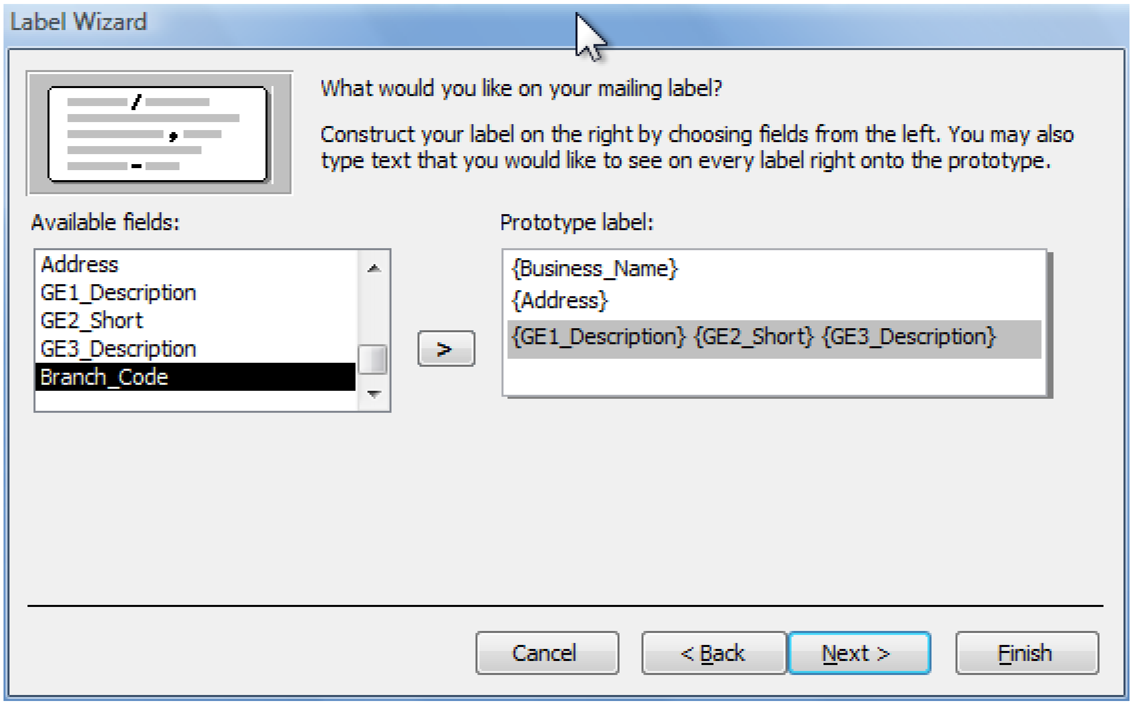
Font size: 10

Font weight: Normal

Text color:

☐ Italic ☐ Underline

Setup the fields how you want them to appear on the label.



What would you like on your mailing label?

Construct your label on the right by choosing fields from the left. You may also type text that you would like to see on every label right onto the prototype.

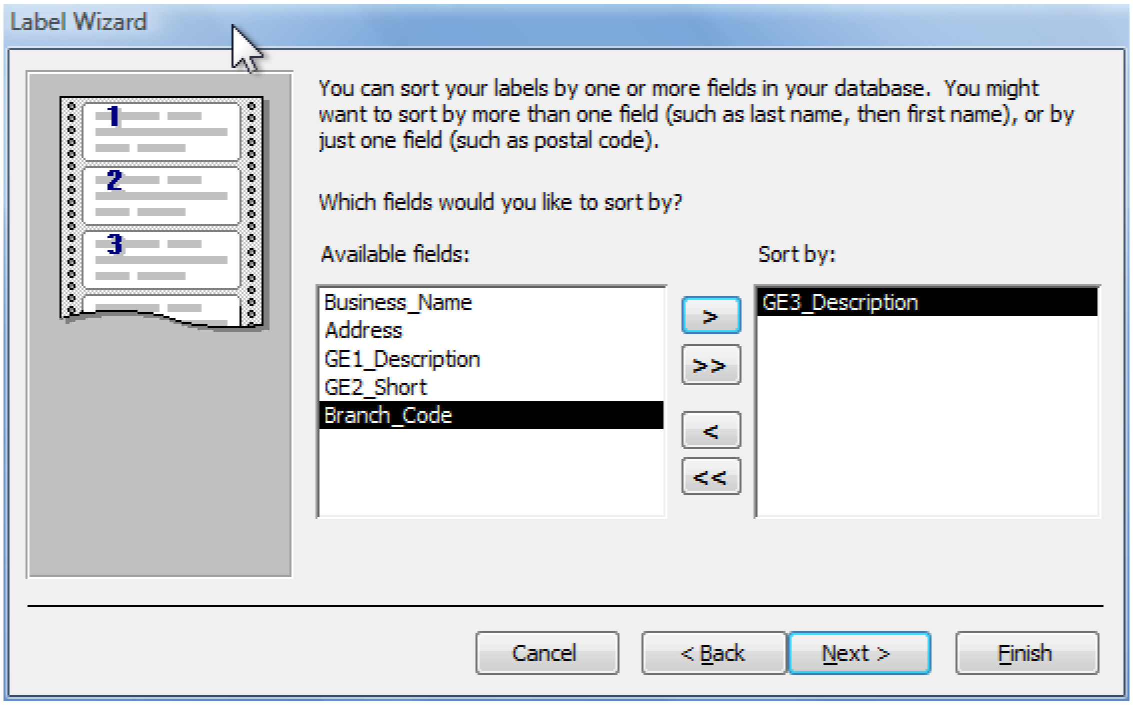
Available fields:

- Address
- GE1_Description
- GE2_Short
- GE3_Description
- Branch_Code

Prototype label:

{Business_Name}
{Address}
{GE1_Description} {GE2_Short} {GE3_Description}

Select any fields you want to sort on. Here I've selected the GE3_Description so we can get a presorted discount from the post office.



You can sort your labels by one or more fields in your database. You might want to sort by more than one field (such as last name, then first name), or by just one field (such as postal code).

Which fields would you like to sort by?

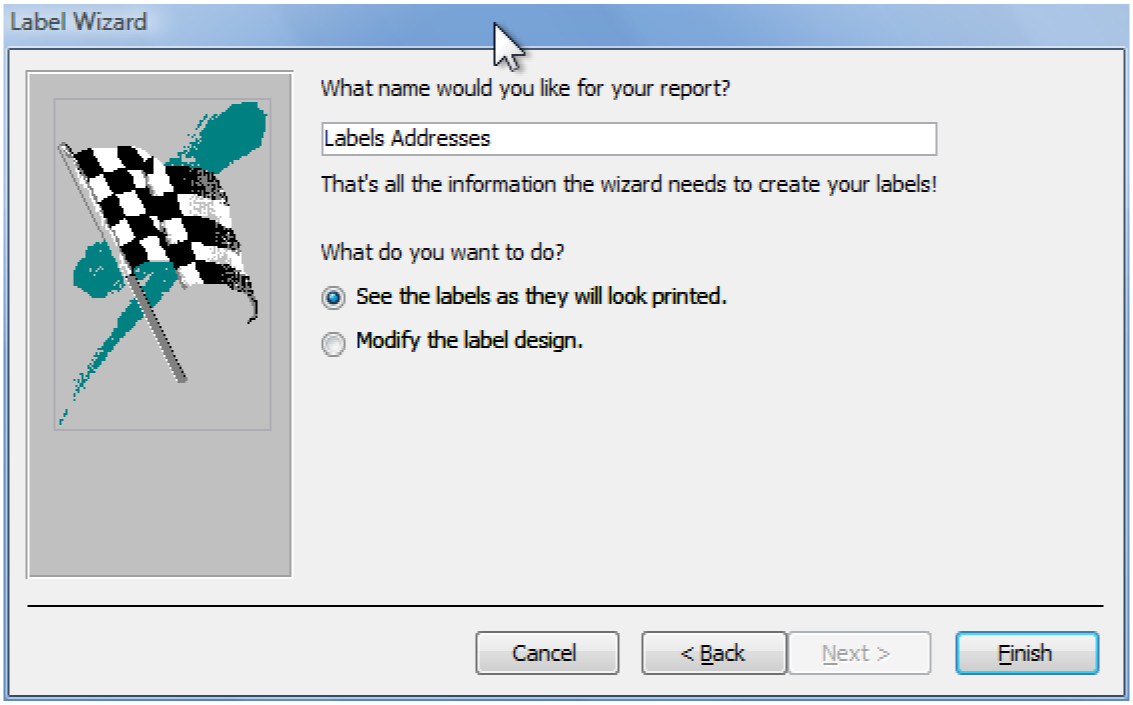
Available fields:

- Business_Name
- Address
- GE1_Description
- GE2_Short
- Branch_Code

Sort by:

- GE3_Description

Give your report a name and save it.



What name would you like for your report?

Labels Addresses

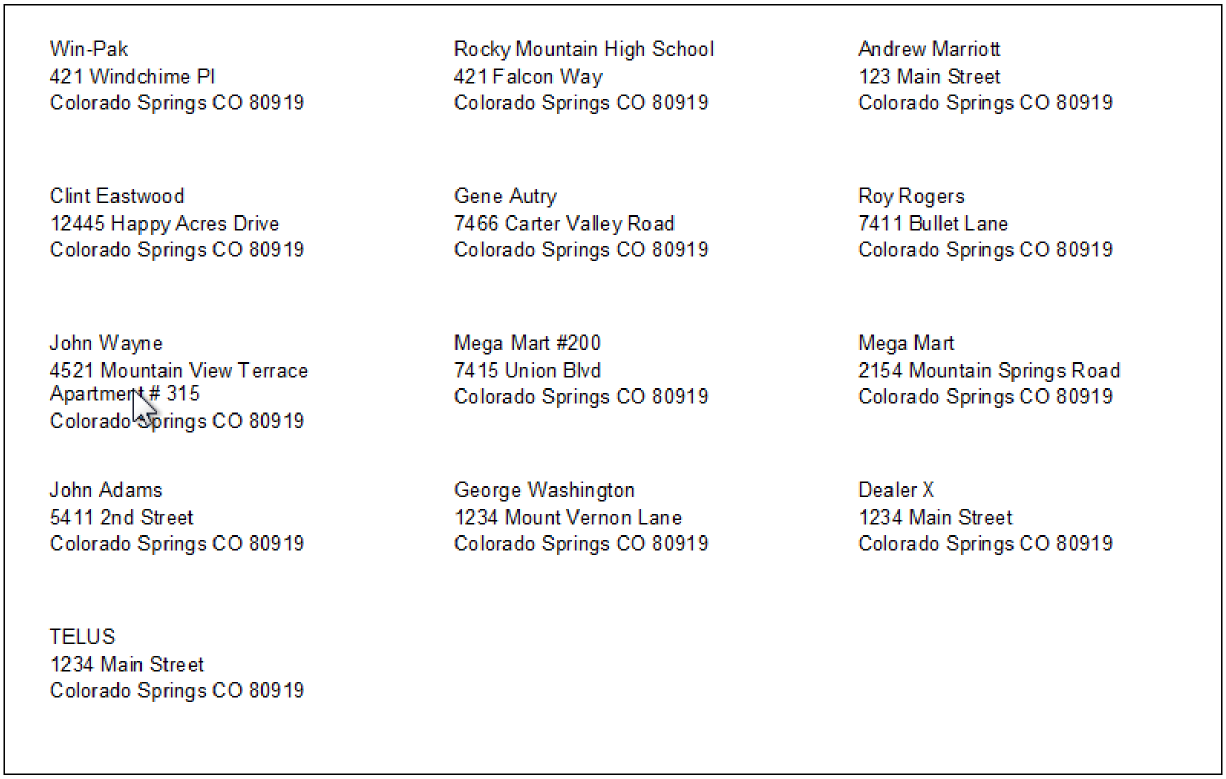
That's all the information the wizard needs to create your labels!

What do you want to do?

● See the labels as they will look printed.

○ Modify the label design.

Click Finish and see a preview.



Win-Pak
421 Windchime Pl
Colorado Springs CO 80919

Rocky Mountain High School
421 Falcon Way
Colorado Springs CO 80919

Andrew Marriott
123 Main Street
Colorado Springs CO 80919

Clint Eastwood
12445 Happy Acres Drive
Colorado Springs CO 80919

Gene Autry
7466 Carter Valley Road
Colorado Springs CO 80919

Roy Rogers
7411 Bullet Lane
Colorado Springs CO 80919

John Wayne
4521 Mountain View Terrace
Apartment # 315
Colorado Springs CO 80919

Mega Mart #200
7415 Union Blvd
Colorado Springs CO 80919

Mega Mart
2154 Mountain Springs Road
Colorado Springs CO 80919

John Adams
5411 2nd Street
Colorado Springs CO 80919

George Washington
1234 Mount Vernon Lane
Colorado Springs CO 80919

Dealer X
1234 Main Street
Colorado Springs CO 80919

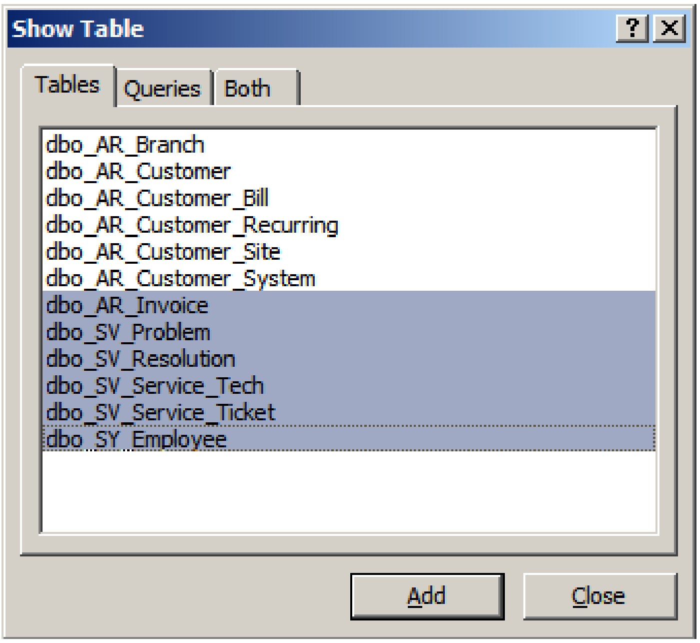
TELUS
1234 Main Street
Colorado Springs CO 80919

Creating a Grouped and Sub Totaled Report

First we will need some additional data. Again select the ODBC database import item. Add these additional tables:

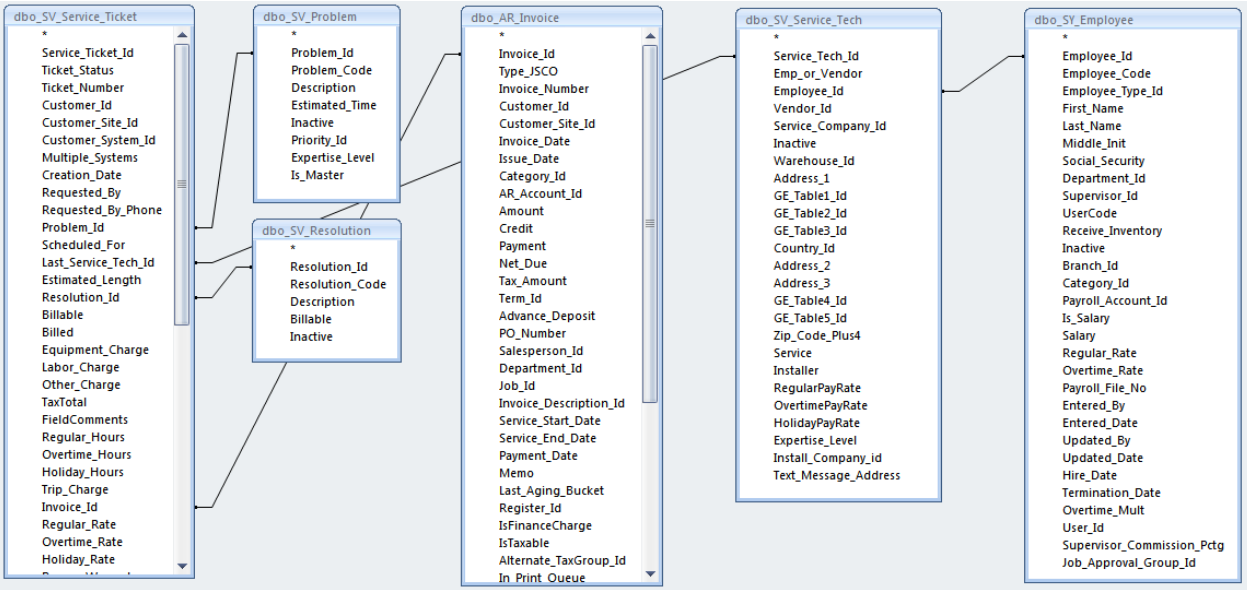
- SV_Service_Ticket
- SV_Problem
- SV_Resolution
- AR_Invoice
- SV_Service_Tech
- SY_Employee

Create a new Query and add these tables.



dbo_AR_Branch
dbo_AR_Customer
dbo_AR_Customer_Bill
dbo_AR_Customer_Recurring
dbo_AR_Customer_Site
dbo_AR_Customer_System
dbo_AR_Invoice
dbo_SV_Problem
dbo_SV_Resolution
dbo_SV_Service_Tech
dbo_SV_Service_Ticket
dbo_SY_Employee

Link the tables as shown.



dbo_SV_Service_Ticket
*
Service_Ticket_Id
Ticket_Status
Ticket_Number
Customer_Id
Customer_Site_Id
Customer_System_Id
Multiple_Systems
Creation_Date
Requested_By
Requested_By_Phone
Problem_Id
Scheduled_For
Last_Service_Tech_Id
Estimated_Length
Resolution_Id
Billable
Billed
Equipment_Charge
Labor_Charge
Other_Charge
TaxTotal
FieldComments
Regular_Hours
Overtime_Hours
Holiday_Hours
Trip_Charge
Invoice_Id
Regular_Rate
Overtime_Rate
Holiday_Rate

dbo_SV_Problem
*
Problem_Id
Problem_Code
Description
Estimated_Time
Inactive
Priority_Id
Expertise_Level
Is_Master

dbo_SV_Resolution
*
Resolution_Id
Resolution_Code
Description
Billable
Inactive

dbo_AR_Invoice
*
Invoice_Id
Type_JSCO
Invoice_Number
Customer_Id
Customer_Site_Id
Invoice_Date
Issue_Date
Category_Id
AR_Account_Id
Amount
Credit
Payment
Net_Due
Tax_Amount
Term_Id
Advance_Deposit
PO_Number
Salesperson_Id
Department_Id
Job_Id
Invoice_Description_Id
Service_Start_Date
Service_End_Date
Payment_Date
Memo
Last_Aging_Bucket
Register_Id
IsFinanceCharge
IsTaxable
Alternate_TaxGroup_Id
In_Print_Queue

dbo_SV_Service_Tech
*
Service_Tech_Id
Emp_or_Vendor
Employee_Id
Vendor_Id
Service_Company_Id
Inactive
Warehouse_Id
Address_1
GE_Table1_Id
GE_Table2_Id
GE_Table3_Id
Country_Id
Address_2
Address_3
GE_Table4_Id
GE_Table5_Id
Zip_Code_Plus4
Service
Installer
RegularPayRate
OvertimePayRate
HolidayPayRate
Expertise_Level
Install_Company_id
Text_Message_Address

dbo_SY_Employee
*
Employee_Id
Employee_Code
Employee_Type_Id
First_Name
Last_Name
Middle_Init
Social_Security
Department_Id
Supervisor_Id
UserCode
Receive_Inventory
Inactive
Branch_Id
Category_Id
Payroll_Account_Id
Is_Salary
Salary
Regular_Rate
Overtime_Rate
Payroll_File_No
Entered_By
Entered_Date
Updated_By
Updated_Date
Hire_Date
Termination_Date
Overtime_Mult
User_Id
Supervisor_Commission_Pctg
Job_Approval_Group_Id

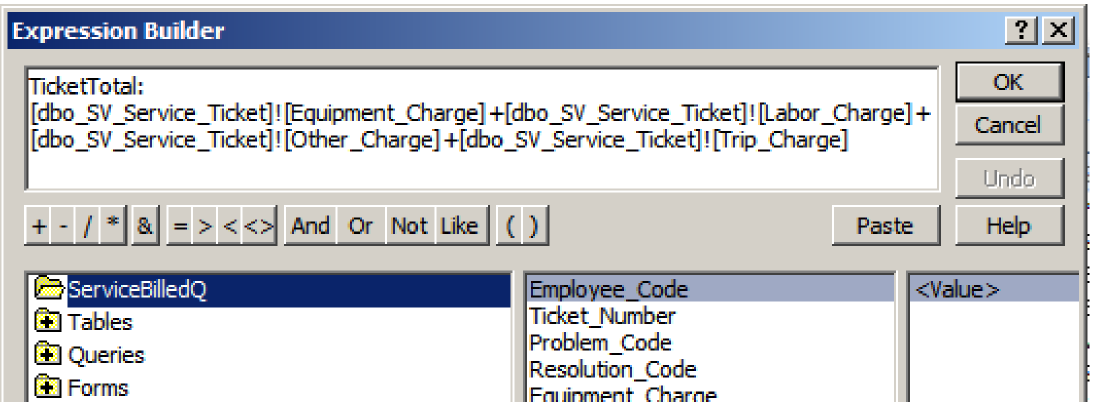
Add these fields.

- dbo_SY_Employee.Employee_Code
- dbo_SV_Service_Ticket.Ticket_Number
- dbo_SV_Problem.Problem_Code
- dbo_SV_Resolution.Resolution_Code
- dbo_SV_Service_Ticket.Equipment_Charge

- dbo_SV_Service_Ticket.Labor_Charge
- dbo_SV_Service_Ticket.Other_Charge
- dbo_SV_Service_Ticket.Trip_Charge
- dbo_AR_Invoice.Invoice_Number
- dbo_AR_Invoice.Amount
- dbo_SV_Service_Ticket.Ticket_Status
- dbo_SV_Service_Ticket.Service_Ticket_Id

Now we need to create a calculated field. We want a field that will be the sum of all of the charges on the ticket. So open the Build dialog and enter these fields.

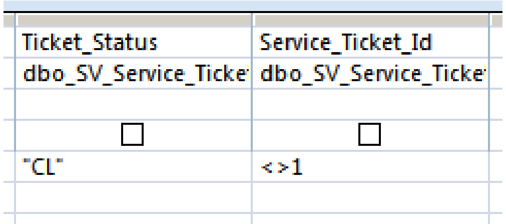
- dbo_SV_Service_Ticket.Equipment_Charge
- dbo_SV_Service_Ticket.Labor_Charge
- dbo_SV_Service_Ticket.Other_Charge
- dbo_SV_Service_Ticket.Trip_Charge



TicketTotal:
[dbo_SV_Service_Ticket]![Equipment_Charge]+[dbo_SV_Service_Ticket]![Labor_Charge]+
[dbo_SV_Service_Ticket]![Other_Charge]+[dbo_SV_Service_Ticket]![Trip_Charge]

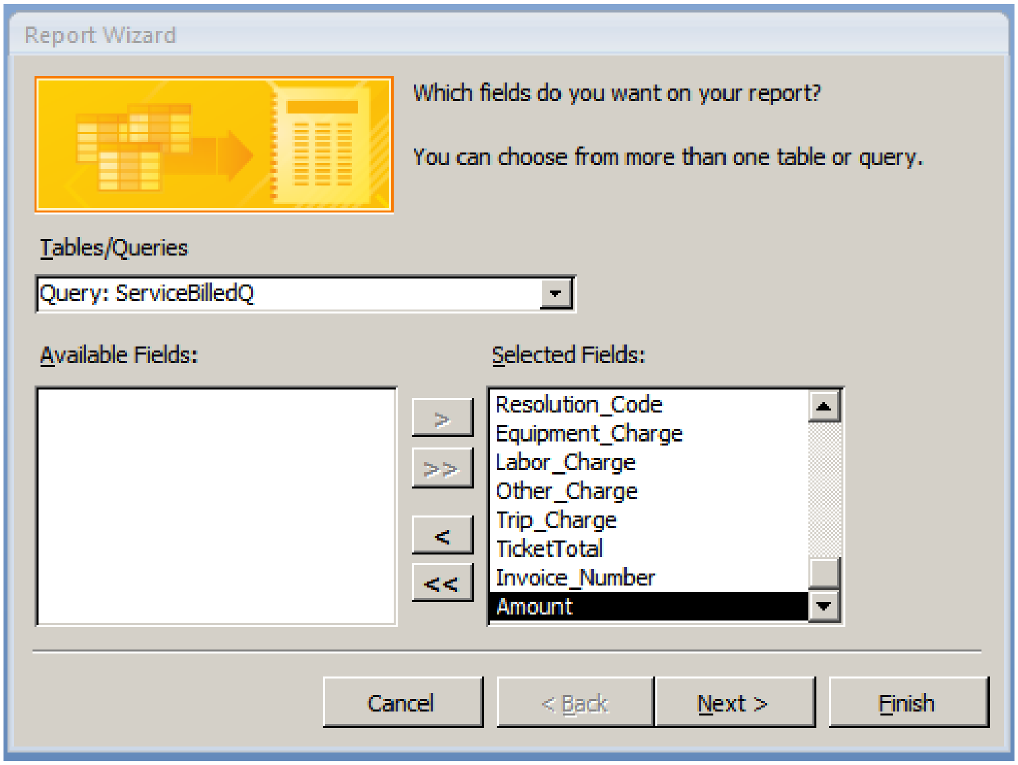
And select OK.

Also add criteria for Ticket_Status and Service_Ticket_Id.



Ticket_Status	Service_Ticket_Id
dbo_SV_Service_Ticke	dbo_SV_Service_Ticke
□	□
"CL"	<>1

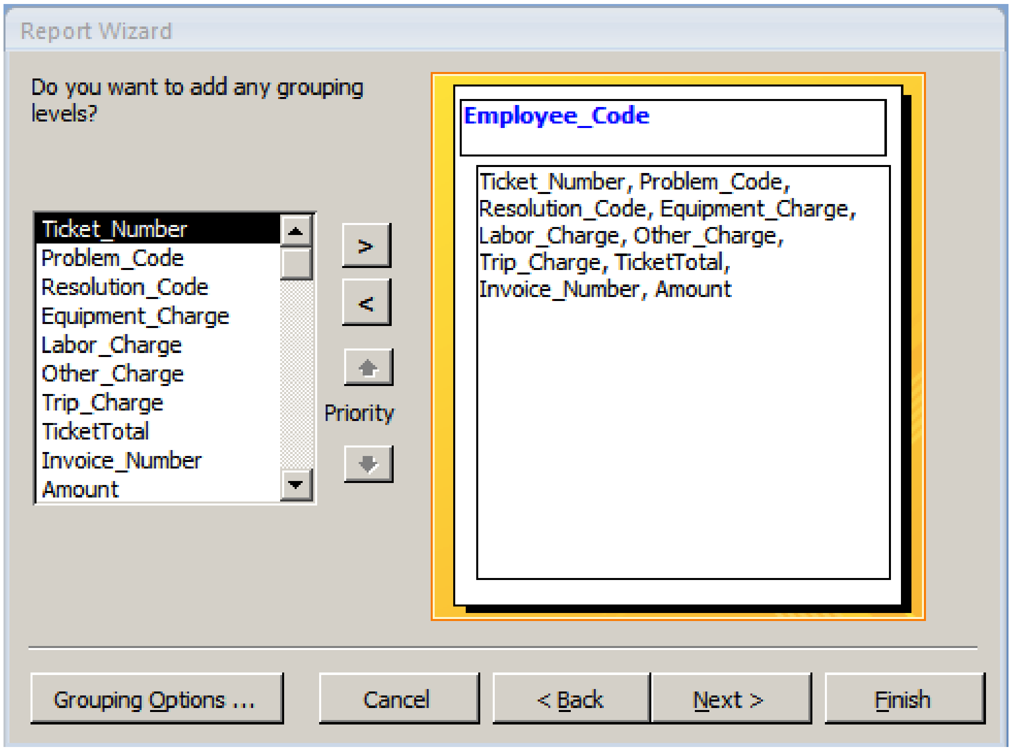
Save the query as ServiceBilledQ. Create a new report and select ServiceBilledQ as the data source, select all of the fields and press next.



Selected Fields:

- Resolution_Code
- Equipment_Charge
- Labor_Charge
- Other_Charge
- Trip_Charge
- TicketTotal
- Invoice_Number
- Amount

We are going to group by Employee_Code. Select the right pointing arrow while Employee_Code is highlighted. The result should look like the image below.

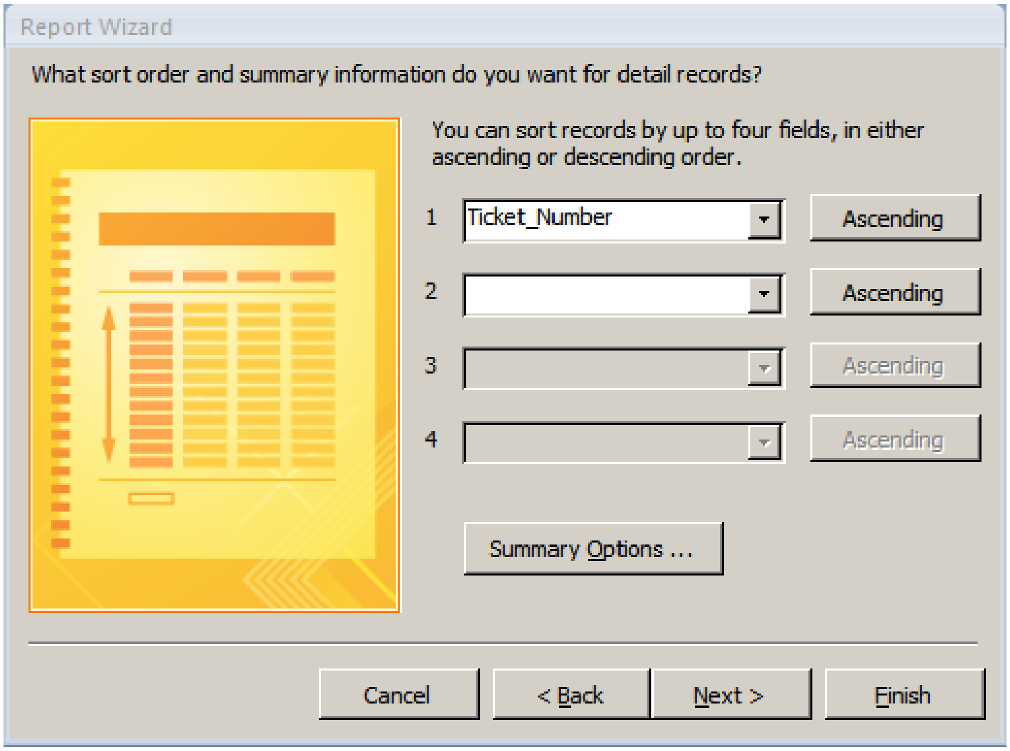


- Ticket_Number
- Problem_Code
- Resolution_Code
- Equipment_Charge
- Labor_Charge
- Other_Charge
- Trip_Charge
- TicketTotal
- Invoice_Number
- Amount

Employee_Code

Ticket_Number, Problem_Code, Resolution_Code, Equipment_Charge, Labor_Charge, Other_Charge, Trip_Charge, TicketTotal, Invoice_Number, Amount

Press Next. We want to sort the tickets by Ticket_Number for each Service_Tech which we can do on the next page of the wizard.



Report Wizard

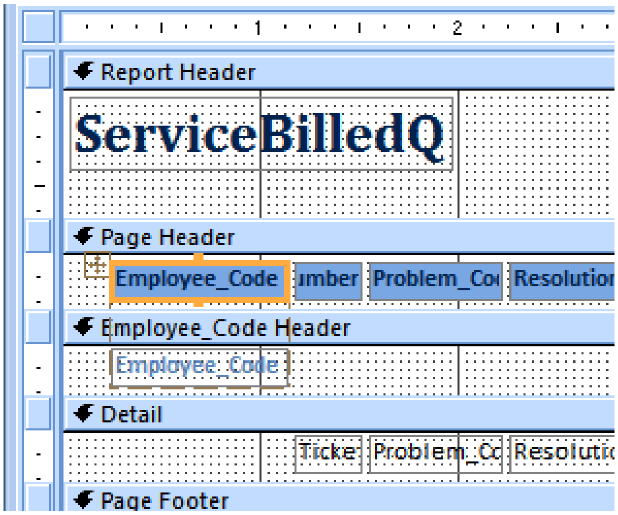
What sort order and summary information do you want for detail records?

You can sort records by up to four fields, in either ascending or descending order.

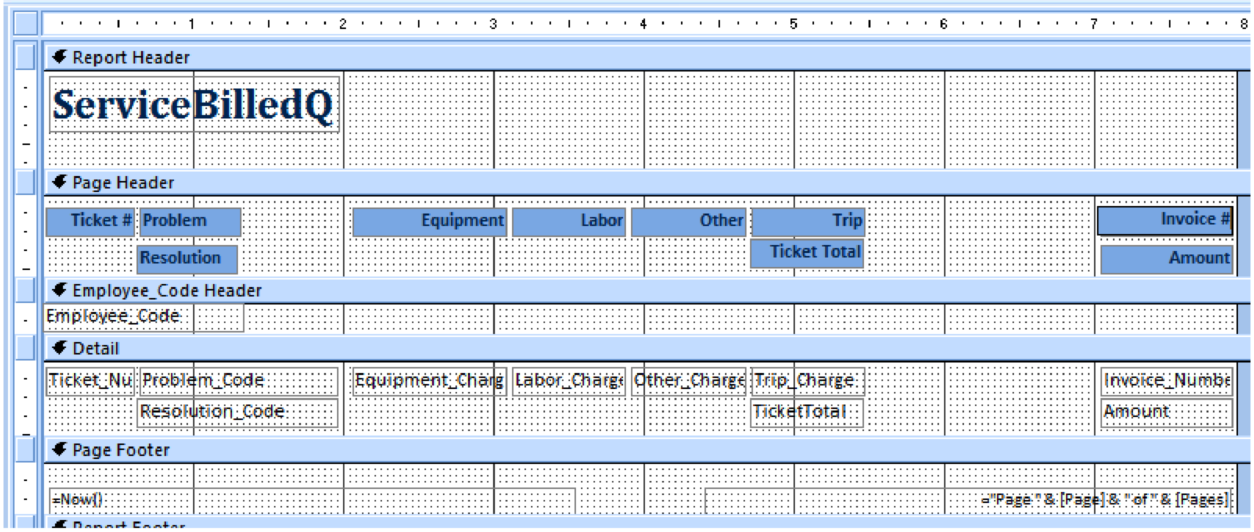
1 Ticket_Number Ascending
2 Ascending
3 Ascending
4 Ascending

Select Finish. A preview of the report will display. Close the preview for now and you will get the report designer. We can now make changes to the basic report to suit our needs.

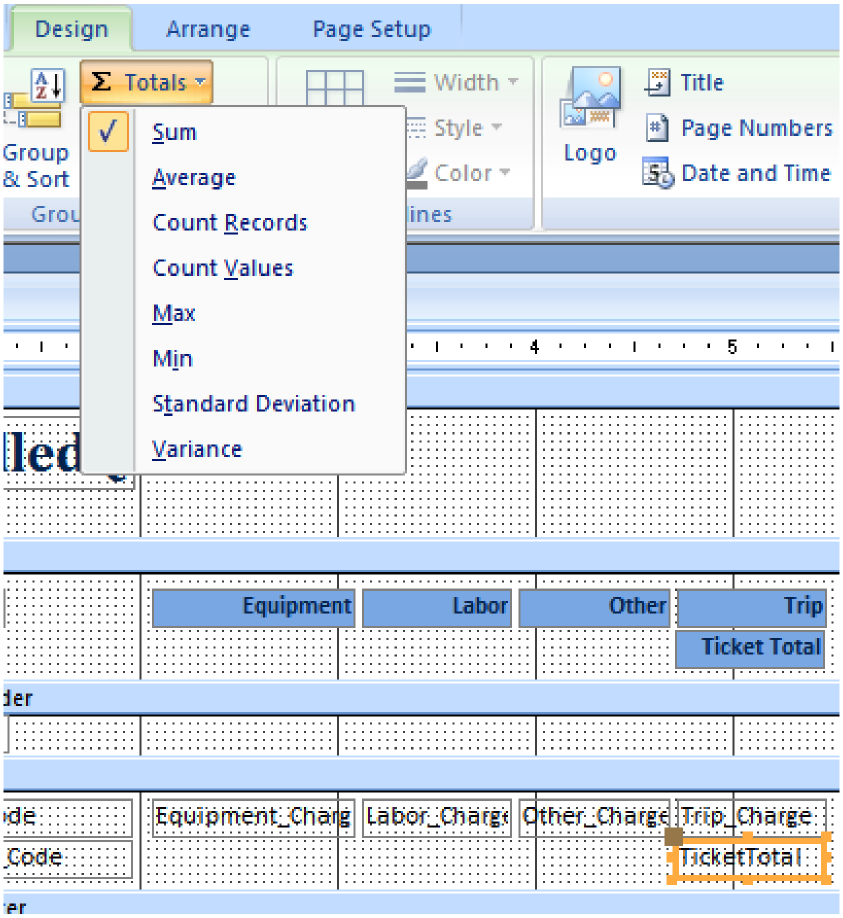
Click on any blank spot on the report to deselect everything. Now select the Employee_Code label.



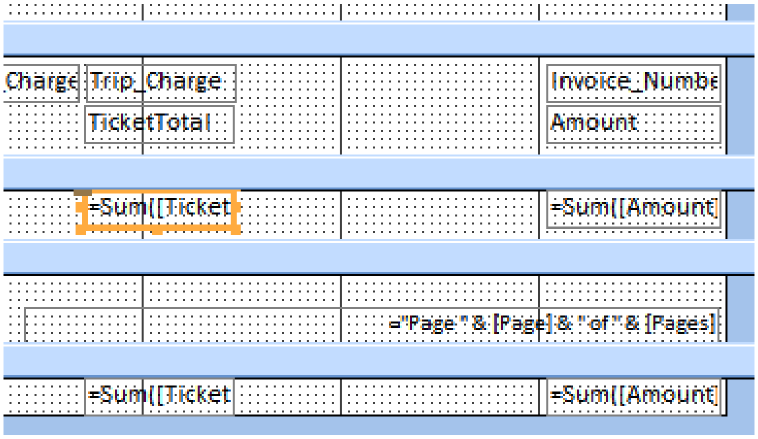
Delete the label. If you delete the Field just drag it back from the fields list. Continue moving fields till your layout looks like the image below.



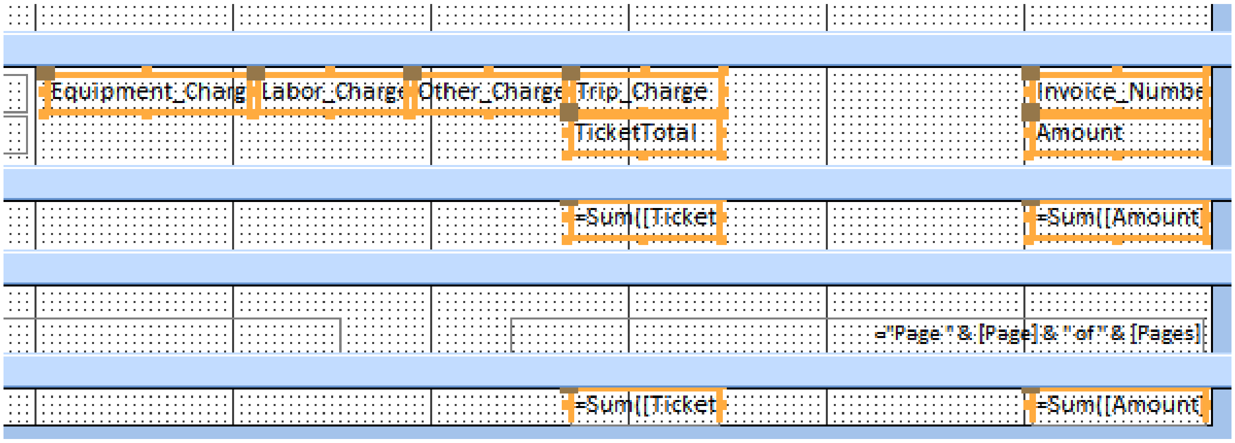
Now we are going to add subtotals. Select the TicketTotal field in the report detail section then choose the Totals menu and Sum in the Design bar.



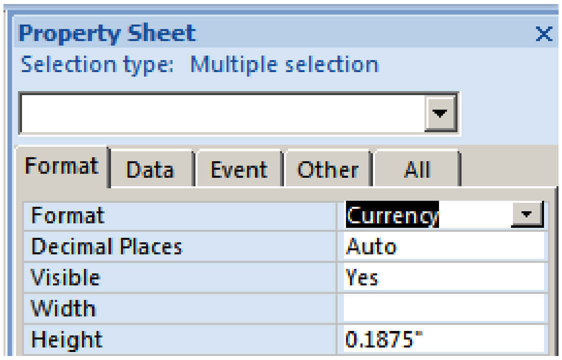
Do the same for the Amount field from the invoice. We will now have a subtotal by service tech and a grand total for all service techs.



We are almost done. Choose all of the currency fields.

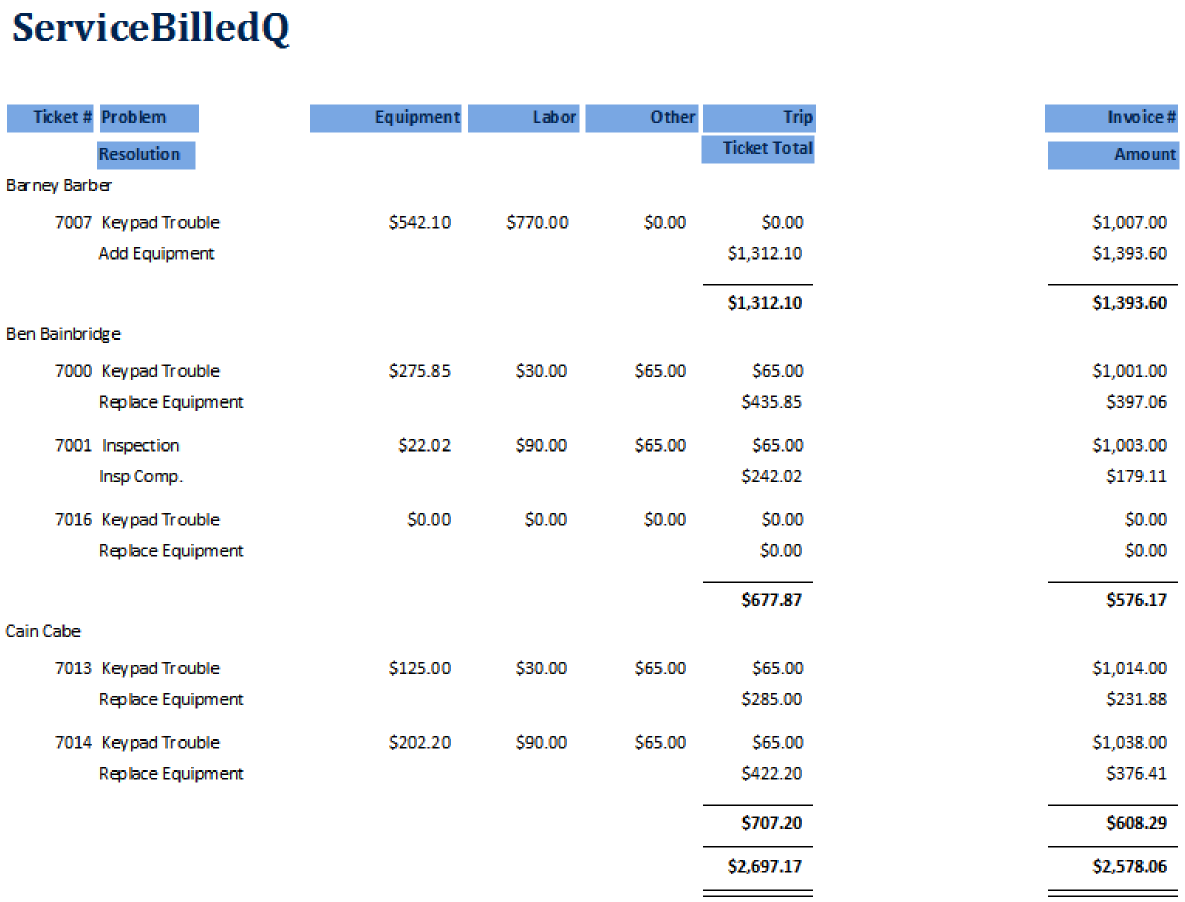


And then in the Properties dialog, choose “Currency” for the format.


Property Sheet
Selection type: Multiple selection

Format	Currency
Decimal Places	Auto
Visible	Yes
Width	
Height	0.1875"

Add a line above the subtotals and set it to black. Add one line above and two lines below the grand total and set their color to black. Save the report and preview it.



ServiceBilledQ

Ticket #	Problem Resolution	Equipment	Labor	Other	Trip Ticket Total	Invoice # Amount
Barney Barber						
7007	Keypad Trouble	$542.10	$770.00	$0.00	$0.00	$1,007.00
	Add Equipment				$1,312.10	$1,393.60
					$1,312.10	**$1,393.60**
Ben Bainbridge						
7000	Keypad Trouble	$275.85	$30.00	$65.00	$65.00	$1,001.00
	Replace Equipment				$435.85	$397.06
7001	Inspection	$22.02	$90.00	$65.00	$65.00	$1,003.00
	Insp Comp.				$242.02	$179.11
7016	Keypad Trouble	$0.00	$0.00	$0.00	$0.00	$0.00
	Replace Equipment				$0.00	$0.00
					$677.87	**$576.17**
Cain Cabe						
7013	Keypad Trouble	$125.00	$30.00	$65.00	$65.00	$1,014.00
	Replace Equipment				$285.00	$231.88
7014	Keypad Trouble	$202.20	$90.00	$65.00	$65.00	$1,038.00
	Replace Equipment				$422.20	$376.41
					$707.20	**$608.29**
					$2,697.17	**$2,578.06**

There are a number of improvements that could be added to the report. The title should be changed. Dates could be added. Perhaps some Customer information.

In this report we have learned how to group and total. We have learned how to expand the detail section to show more data than will fit on one line.

Basic SQL Language

The majority of SQL queries can be created with just four commands; Select, From, Where and Order By.

Select Keyword

The "Select" keyword prefaces the list of data to return. This data can be; fields, calculated fields, constants or sub queries. Each of these data items must be separated by a comma.

Fields are the individual fields from the records in the tables. They are collected and displayed how they are in the database. This is the most common use for the select keyword; examples might be Quantity, Rate, Part_Code, Business_Name, etc.

Calculated fields are fields that have had a process applied to them. For instance a calculated field might be Quantity * Rate to get the extended price. Another example would be GE3_Description + '-' + ZipPlus4 to get a complete U.S. zipcode.

Constants are numbers or characters that you want to show in your query and will be the same for the entire column. The '-' above is a constant. So that GE3_Description, '-' and ZipPlus4 could be in separate columns if separated by commas instead of being joined together with plus signs.

Sub queries are queries within parenthesis that return a value that is not directly related to the main query. A common use for sub queries is to return totals IE (Select Sum(Amount) From AR_Invoice).

An example of a Select clause would be:

Select cu.Customer_Number, cb.Business_Name, cb.Address_1, cb.GE1_Description, cb.GE2_Short, cb.GE3_Description

From Keyword

The "From" keyword prefaces the list of tables and how they are joined. IE

From AR_Customer cu Inner Join AR_Customer_Bill cb On cu.Customer_Id = cb.Customer_Id

From is the keyword, followed by the first table name and an optional short nickname or alias. Next comes the type of Join, which will be discussed below. Then the second table is added, also followed by an optional alias. After the two tables are named comes the "On" keyword. After the "On" keyword are the conditions of how the tables relate to one and another, in this case only return rows where for each Customer_Id in AR_Customer there is a matching Customer_Id in AR_Customer_Bill.

Join Keyword

Joins come in different types. The most common type and the type that is used by default if no other type is specified is the Inner Join.

Inner Join

Inner joins only return rows where both tables are equal. Using the example below, records 2, 4 and 6 are returned because those are the only records present in both tables.

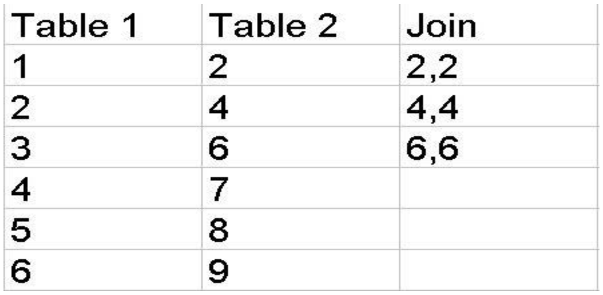


Table 1	Table 2	Join
1	2	2,2
2	4	4,4
3	6	6,6
4	7	
5	8	
6	9	

Left Outer Join

Leeft and Right Outer joins return rows containing all of the records from one table and only the matching records from the other table. So in our example below, all of the records from Table 1 are returned but only 2, 4, and 6 are returned from Table 2 as they are the only records that match. The Left Outer Join and Right Outer Join differ only in which table is on the left side of the Join keyword and which is on the right side of the Join keyword. Our Left Outer Join example would look like this:

Table 1 Left Outer Join Table 2

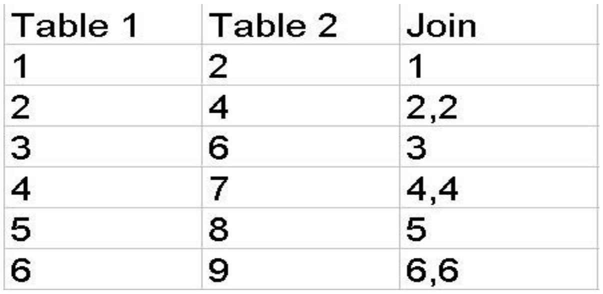


Table 1	Table 2	Join
1	2	1
2	4	2,2
3	6	3
4	7	4,4
5	8	5
6	9	6,6

Right Outer Join

Using the same example data, a Right Outer Join would return rows containing all of the records from Table 2 and only 2, 4 and 6 from Table 1. Again which side of the Join Keyword a table is on is the determining factor. Our Right Outer Join example would look like this:

Table 1 Right Outer Join Table 2

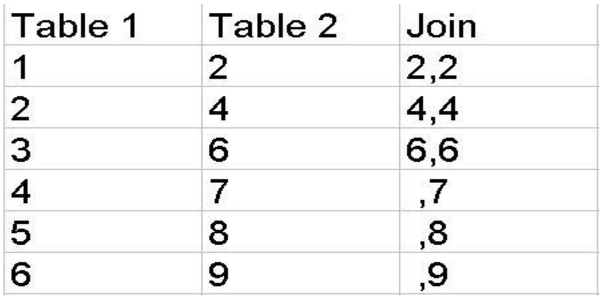


Table 1	Table 2	Join
1	2	2,2
2	4	4,4
3	6	6,6
4	7	,7
5	8	,8
6	9	,9

Full Outer Join

Full Outer Joins result in all of the records from both tables. It would be as if you added a Left Outer Join and a Right Outer Join together. A Full outer Join would look like this:

Table 1	Table 2	Join
1	2	1,
2	4	2,2
3	6	3,
4	8	4,4
5	10	5,
6	12	6,6

		,8
		,10
		,12

Where Keyword

Where clauses control what rows are returned by matching the records against a set of conditions or filters connected by logical operators. Each condition or filter results in a “True” or “False” condition.

Examples of “True” conditions are:

5 = 5
‘A’ < ‘B’
3 + 4 = 7
4 <> 9

Examples of “False” conditions are:

5 <> 5
‘A’ > ‘B’
4 + 4 = 7
4 = 9

Of course these examples would not do us much good, but we can substitute Fields for the numbers and characters in the conditions, for example:

Amount = 5
BusinessName < ‘B’
InvoiceTot - CreditTot = 7
The query will return every row in which the conditions of the Where clause are true. For example, the following query will return only invoices for $5.00. No other value invoice would be included in the returned rows.

```
Select
Invoice_Number,
Amount,
Net_Due
From
AR_Invoice
Where
Amount = 5
```

Notice we have included the Net_Due. The value of Net_Due will not affect what rows are returned. It will only be displayed. If we wanted to include only invoices with an outstanding balance we would change query to look like this:

```
Select
Invoice_Number,
Amount,
Net_Due
From
```

AR_Invoice
Where
Amount = 5
And
Net_Due > 0

This brings up the next concept, logical operators.

Logical Operators

The most common logical operators are And, Or, Not, Xor, Nand and Nor.

And

Value 1	Value 2	Result
False	False	False
False	True	False
True	False	False
True	True	True

Xor (Exclusive or)

Value 1	Value 2	Result
False	False	False
False	True	True
True	False	True
True	True	False

Or

Value 1	Value 2	Result
False	False	False
False	True	True
True	False	True
True	True	True

Nand (the same as Not(Value 1 And Value 2))

Value 1	Value 2	Result
False	False	True
False	True	True
True	False	True
True	True	False

Not (Reverses any result)

Value 1	Result
False	True
True	False

Nor (the same as Not(Value 1 Or Value 2))

Value 1	Value 2	Result
False	False	True
False	True	False
True	False	False
True	True	False

Another thing to know is the precedence of logical operators. Everyone knows that 2 + 5 * 3 is 17 and not 21 because we know that you multiply before you add, this is the precedence of arithmetic operators. Take for example the following data:

ID	Amount	City
1	5.00	Flint

2	5.00	Detroit
3	0.00	Flint
4	0.00	Detroit

If our Where clause is Amount = 5 And City = ‘Flint’ Or City = ‘Detroit’ we might expect to get rows 1 and 2. In reality we would get rows 1, 2, and 4. Just as 2 + 5 * 3 should be thought of being written as 2 + (5 * 3) so the 5 * 3 is done first, Our Where clause should be thought of as being written as (Amount = 5 And City = ‘Flint’) Or City = ‘Detroit’. The And operator is processed first just like the multiplication operator in arithmetic. If our where clause were written as Amount = 5 And (City = ‘Flint’ Or City = ‘Detroit’) we would get rows 1 and 2. So the order of precedence is Not, And then Or but the order of precedence should not be relied on. Like the example, to be sure, use parentheses.

Other Where Clause Filters

So far we have looked at filters, the true false statements that use the simple comparators listed below:

Comparator	True Examples	False Examples
=	5 = 5, ‘A’ = ‘A’	5 = 7, ‘X’ = ‘R’
<>	5 <> 6, ‘AB’ <> ‘CD’	5 <> 5, ‘A’ <> ‘A’
<	5 < 6, ‘A’ < ‘G’	5 < 5, 6 < 5, ‘A’ < ‘A’, ‘G’ < ‘A’
>	6 > 5, ‘G’ > ‘A’	5 > 5, 5 > 6, ‘A’ > ‘G’, ‘A’ > ‘A’
<=	5 <= 6, ‘A’ <= ‘G’, 5 <= 5, ‘A’ <= ‘A’	6 <= 5, ‘G’ <= ‘A’
>=	6 >= 5, ‘G’ >= ‘A’, 5 >= 5, ‘A’ >= ‘A’	5 >= 6, ‘A’ >= ‘G’

Now we will look at Is Null, In, Like and Between. Is Null returns a true if the Field being examined is a Null. Remember, a Null is not the same as a blank “” or a space “ “. A Null means not defined or never entered. So using our Right Outer Join example:

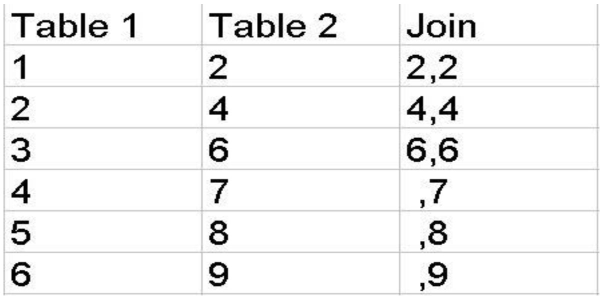


Table 1	Table 2	Join
1	2	2,2
2	4	4,4
3	6	6,6
4	7	,7
5	8	,8
6	9	,9

And a where clause something like this:

Where Table1.Field Is Null

We would get the following records returned:

,7

,8
,9

Is Null should not be confused with IsNull. IsNull is a function that allows you to replace nulls with a default value. It replaces only the Nulls otherwise it uses the value of the Field. If we wanted the nulls to be replaced with a 0 we would write a function like this:

IsNull(FieldName,0)

In compares a Field to a list of values for example:

Where City In (‘Flint’, ‘Detroit’, ‘Cleveland’)

The list of values can be either a comma separated list of literals (as above) or a sub query like the next example:

```
Select
Invoice_Number,
Amount
From
AR_Invoice
Where
Service_Ticket_Id In (Select Service_Ticket_Id From SV_Service_Ticket Where Service_Ticket_Id <> 1)
```

The sub query must return only one field per row. The returned row must match the type of the field on the left of the In Keyword.

Like uses characters and wild cards to create a pattern matching filter. An example of a Like filter:

```
Select
Customer_Number,
Customer_Name
From AR_Customer
Where Customer_Name Like 'A_a%'
```

This would return all customers whose name started with an “A” then contained another character of any sort including spaces, contained an “a” in the third spot followed by zero or more characters of any kind. Below is a chart of the wild cards and what they mean.

Wildcard character	Description	Example
%	Any string of zero or more characters.	WHERE title LIKE '%computer%' finds all book titles with the word 'computer' anywhere in the book title.
_ (underscore)	Any single character.	WHERE au_fname LIKE '_ean' finds all four-letter first

		names that end with ean (Dean, Sean, and so on).
[]	Any single character within the specified range ([a-f]) or set ([abcdef]).	WHERE au_lname LIKE '[C-P]arsen' finds author last names ending with arsen and starting with any single character between C and P, for example Carsen, Larsen, Karsen, and so on. In range searches, the characters included in the range may vary depending on the sorting rules of the collation.
[^]	Any single character not within the specified range ([^a-f]) or set ([^abcdef]).	WHERE au_lname LIKE 'de[^l]%' all author last names starting with de and where the following letter is not l.

Between takes two parameters and does exactly as one would expect. Here is an example:

```
Select
Invoice_Number,
Amount
From AR_Invoice
Where Amount Between 0.00 AND 15.00
```

This returns all invoices where the amount is 0.00 through 15.00 inclusive. If you run this it will even return the 1 record which should be the only invoice with a 0.00 Amount. Please notice the "AND" portion of the Between filter. This is not the same as a normal And. It is NOT evaluated with the other And's, and Or's. It is just part of the Between filter and should be considered only as part of the Between. Between also works with character values IE:

Where Customer_Name Between 'A' AND 'D'

Again this would include all names starting with "A" through name of "D", not starting with "D" as there are no wild cards here. If you want all of the "D"s, use something like this:

Where Customer_Name Between 'A' AND 'Dzzz'

Order By Keywords

Order By controls what order the records are returned in. If no Order By is included, then often the record set will be in the order of the first field in the Select list, often, but not always. If it is important the order records are returned in, use a Order By clause. Here is an example:

```
Select
Customer_Number,
Customer_Name
From AR_Customer
Where Customer_Name Like 'A_a%'
Order By Customer_Name
```

Order By can be either ascending (asc) or descending (desc). You can also mix fields and directions. For example:

Order By Customer_Number asc, Amount desc, Invoice_Type asc

This Order By would take the records and put them in order by Customer_Number from least to greatest, then the invoices for those customers in order by the amount from Largest to least and finally by Invoice_Type from first to last. Fields included in the Order By do NOT need to be in the Select clause.

Union Keyword

Sometimes, we need to create a list of records that combines two separate queries. For example, we want a list of open invoices and open credits in order by customer_number and then date meshed into one recordset.

```
Select
Customer_Number,
Customer_Name,
Invoice_Date,
Net_Due,
'I'
From AR_Customer C Inner Join
AR_Invoice I On C.Customer_Id = I.Customer_Id
Where net_Due > 0
Union
Select
Customer_Number,
Customer_Name,
Credit_Date,
-1 * (Amount - Used_Amount),
'C'
From AR_Customer C Inner Join
AR_Credit I On C.Customer_Id = I.Customer_Id
Where Amount - Used_Amount > 0
Order By Customer_Number, Invoice_Date
```

Let's look at this query from the top. First we have a the select clause. Notice the last item in the list is 'I'. This literal will place a column in our recordset that has an "I" in every row that is an invoice. In the from clause we use an Inner Join to connect the two tables. Notice the use of aliases here, the C and I. This is done just to make the lines a bit more manageable in length. Next we have the Where clause that returns only records that still have a Net_Due. Now we get to the new clause, the Union keyword. A Union keyword joins the Select query above it with the Select query below it. The number and order of columns must be the same and the data types for the columns of each query must be compatible. Below the Union we have another query that returns the open credits. In order to find open credits we had to take the Amount – Used_Amount. We also multiply the result time a negative one so that the credits will be negative compared to the invoices. Also notice that the 'I' field from the top query is now a 'C' and that the Where clause contains a calculated value. Lastly we have an Order By clause. The order by clause must exist after the two queries but the fields must be named from the first query. Also, You may Union as many queries as you want as long as you follow the rules about number, order and type of columns.

There is one option to the Union keyword. If you use the Union key word between two queries and some of the records returned by the first query exactly match some of the records returned by the second query, the final recordset will have only one copy of any record. In other words, all duplicates are reported only one. If you want to see the duplicates, use the All keyword after the Union keyword. IE.

Union All

In our example query, this would not be a problem for two reasons. First, we are returning records where the invoices are marked by an 'I' and credits by a 'C'. Secondly, all of the invoices will be positive amounts and all of the credits will be negative amounts.

Notes: