12TH ANNUAL

SedonaOffice® USERS CONFERENCE

MARCO ISLAND, FLORIDA
JANUARY 27 - 29, 2014

Data Mining 1

Presented By:

Matt Howe

Table of Contents

Overview 4
Each Company is a Database 4
Databases Contain Tables, Views, and Stored Procedures 4
Tables Contain Fields 4

Linking Tables 5
Link Types 5

Customer Structure 6

Invoice Structure 7

Cash Structure 8

Vendor Structure 9

Vendor Bills Structure 10

Check Structure 11

Job Structure 12

Service Ticket Structure 13

Inventory structure 14

General Ledger Structure 15

Open Database Connectivity (ODBC) 16
Creating an ODBC Connection with Excel 16

Building a Query Using Excel and MS Query 18

Using Microsoft Access to Review Your Data 21
Connecting Access via ODBC 21
Writing a query with Access 25
Creating a Report with Access 30
Creating a Grouped and Sub Totaled Report 34

Using SedonaOffice Data in a Microsoft Mail Merge 41
Creating the List of Customers 41
Creating the Letter 44
Merging and Creating the Letters 50

Basic SQL Language 50
Select Keyword 50
From Keyword 51
Join Keyword 51
Inner Join *51*
Left Outer Join *51*
Right Outer Join *52*
Full Outer Join *52*

Where Keyword 53
Logical Operators *54*

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 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 a 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 can also 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. There 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 and a separate field for time. All date and time related fields are Datetime fields.

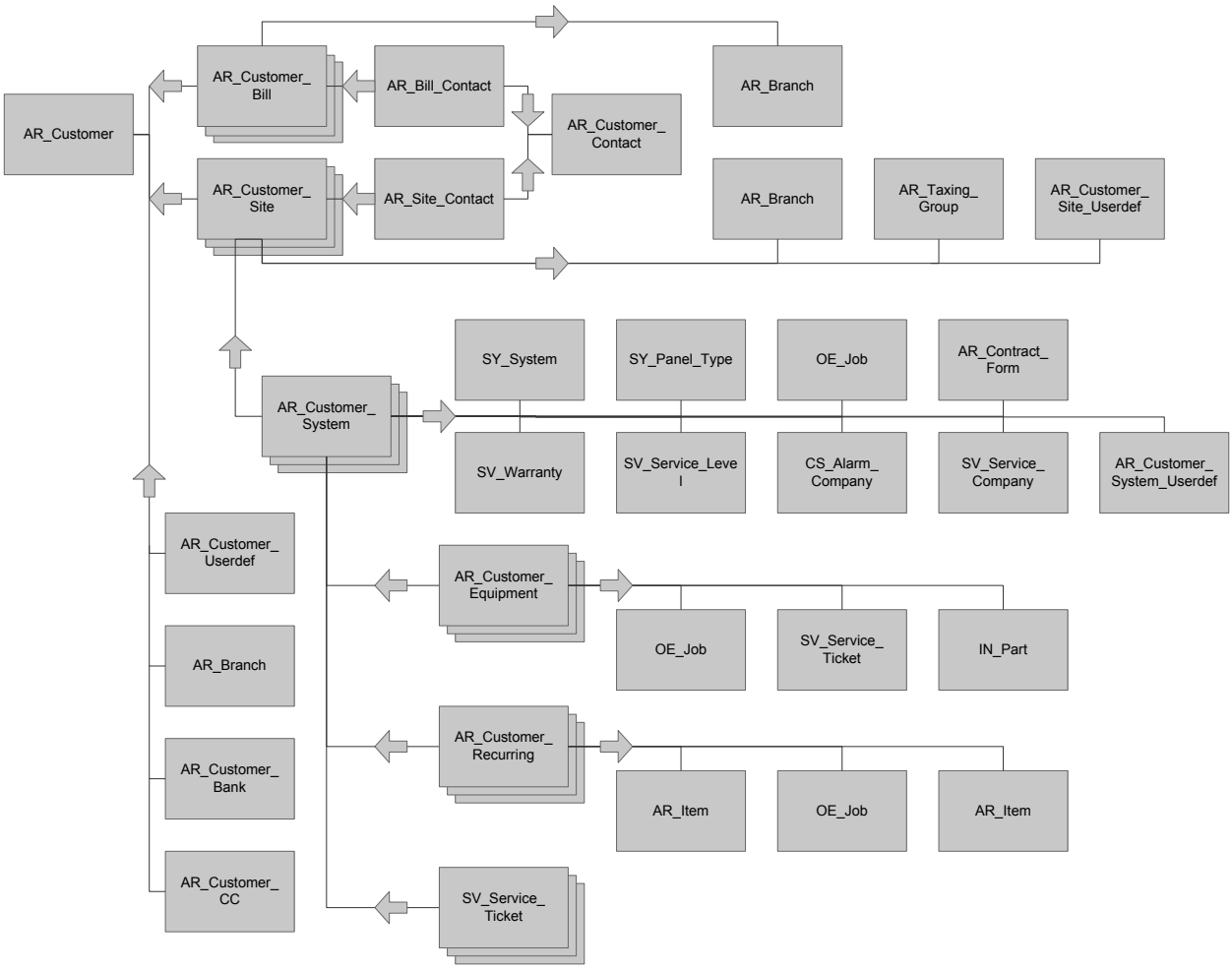
Linking Tables

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.

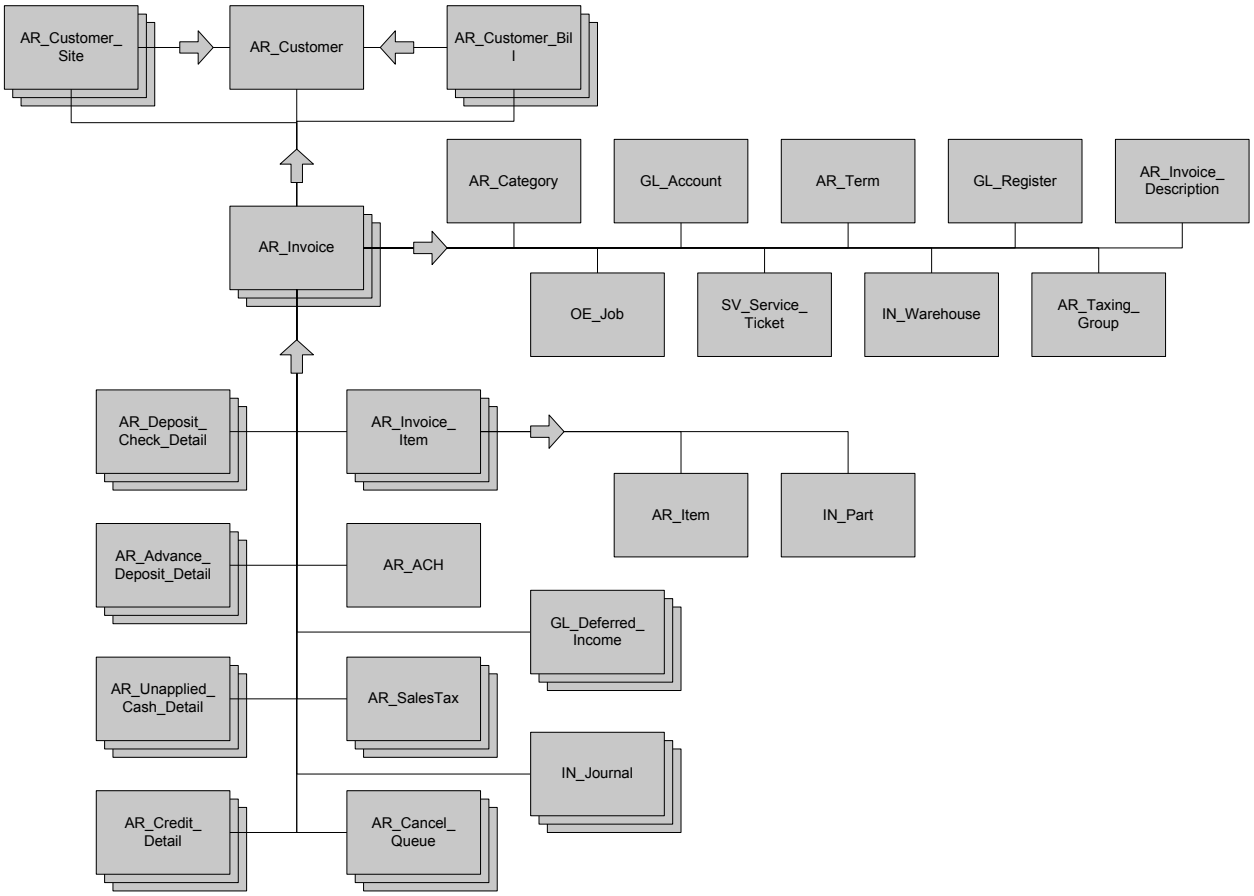
Link Types

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 simplified diagrams to give an outline of the relationship of the various tables that combine to make up a data structure.

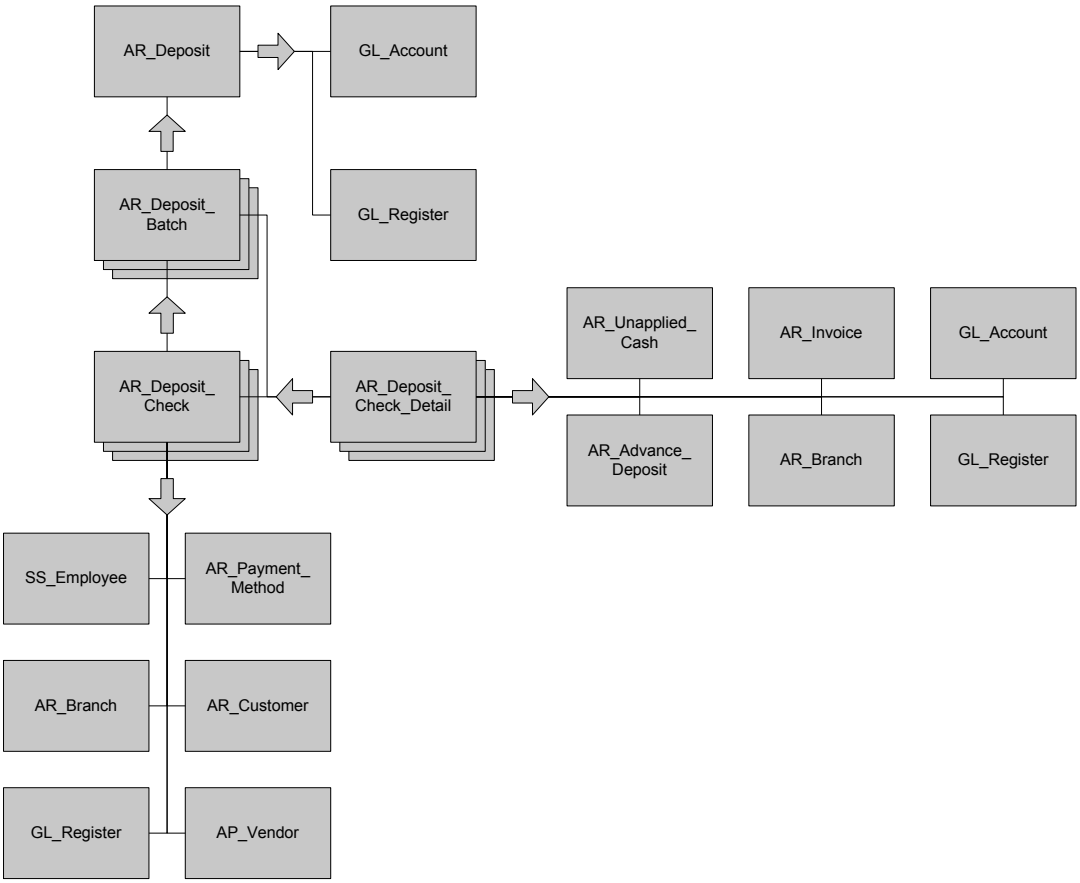
Customer Structure



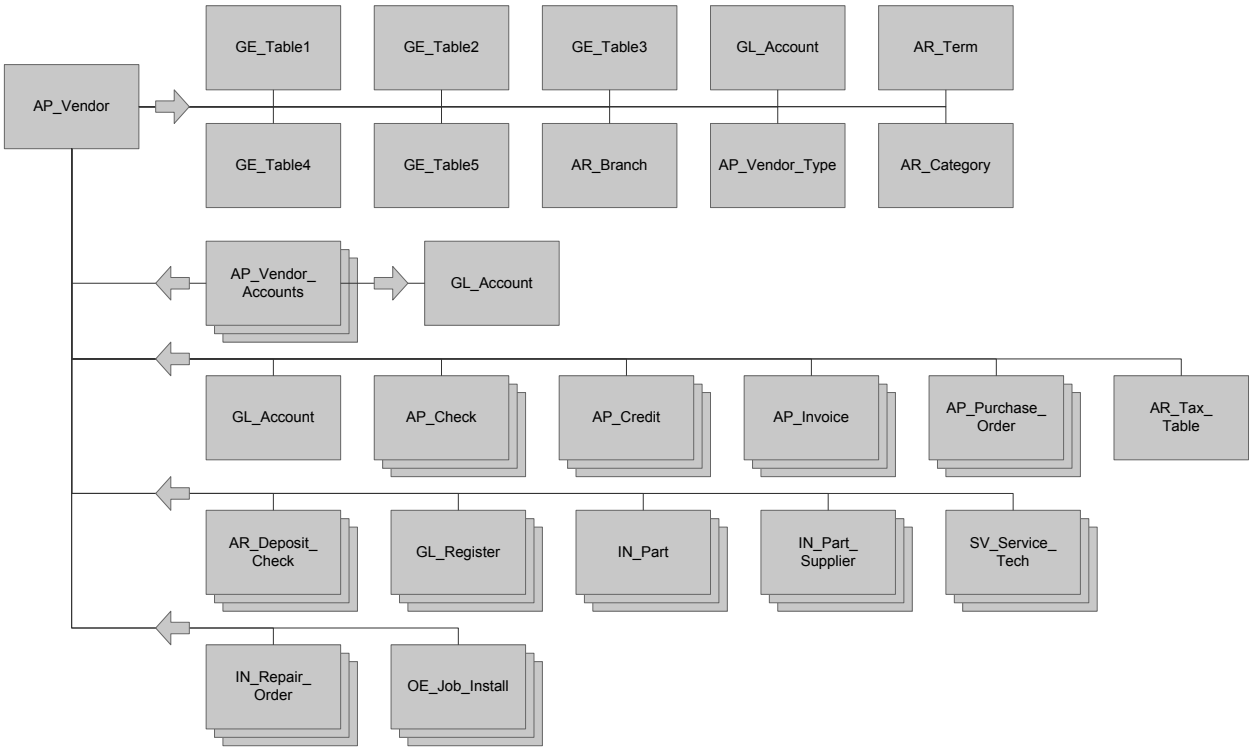
Invoice Structure



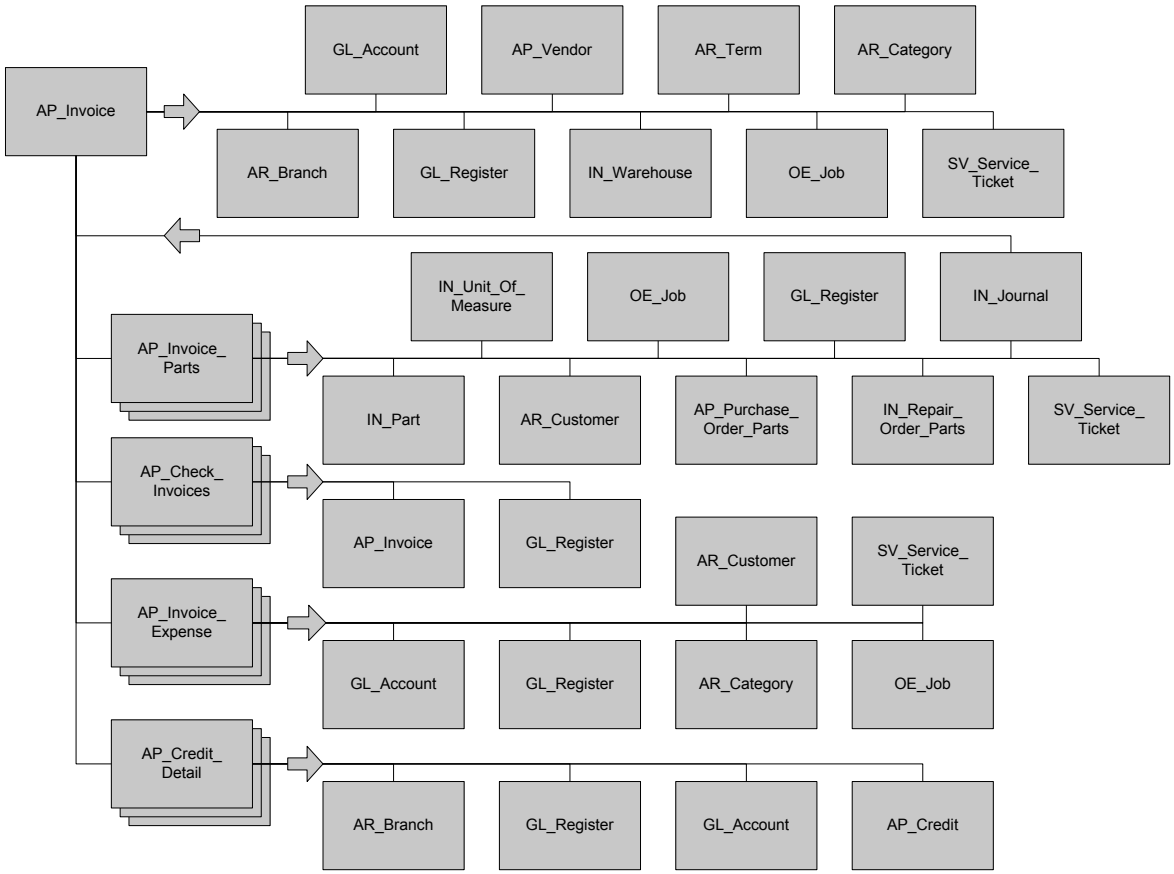
Cash Structure



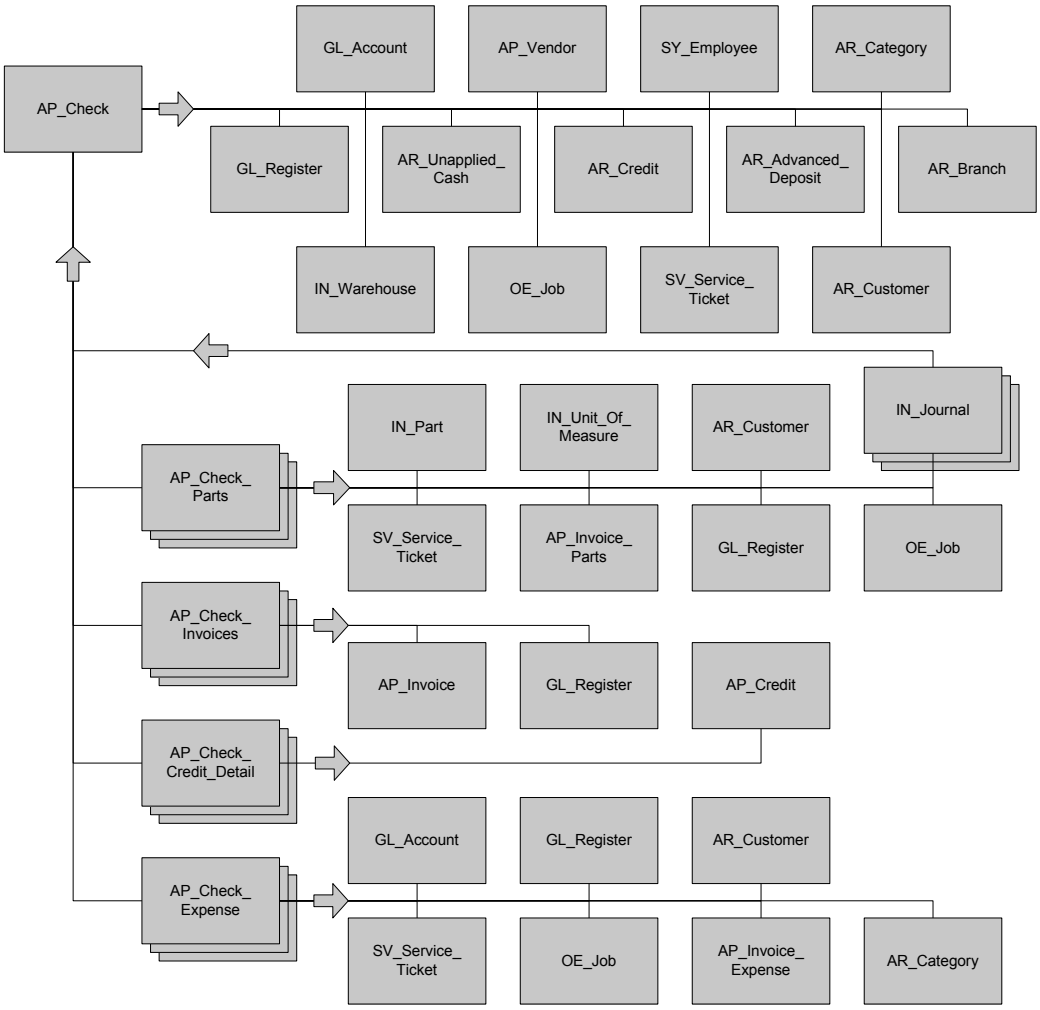
Vendor Structure



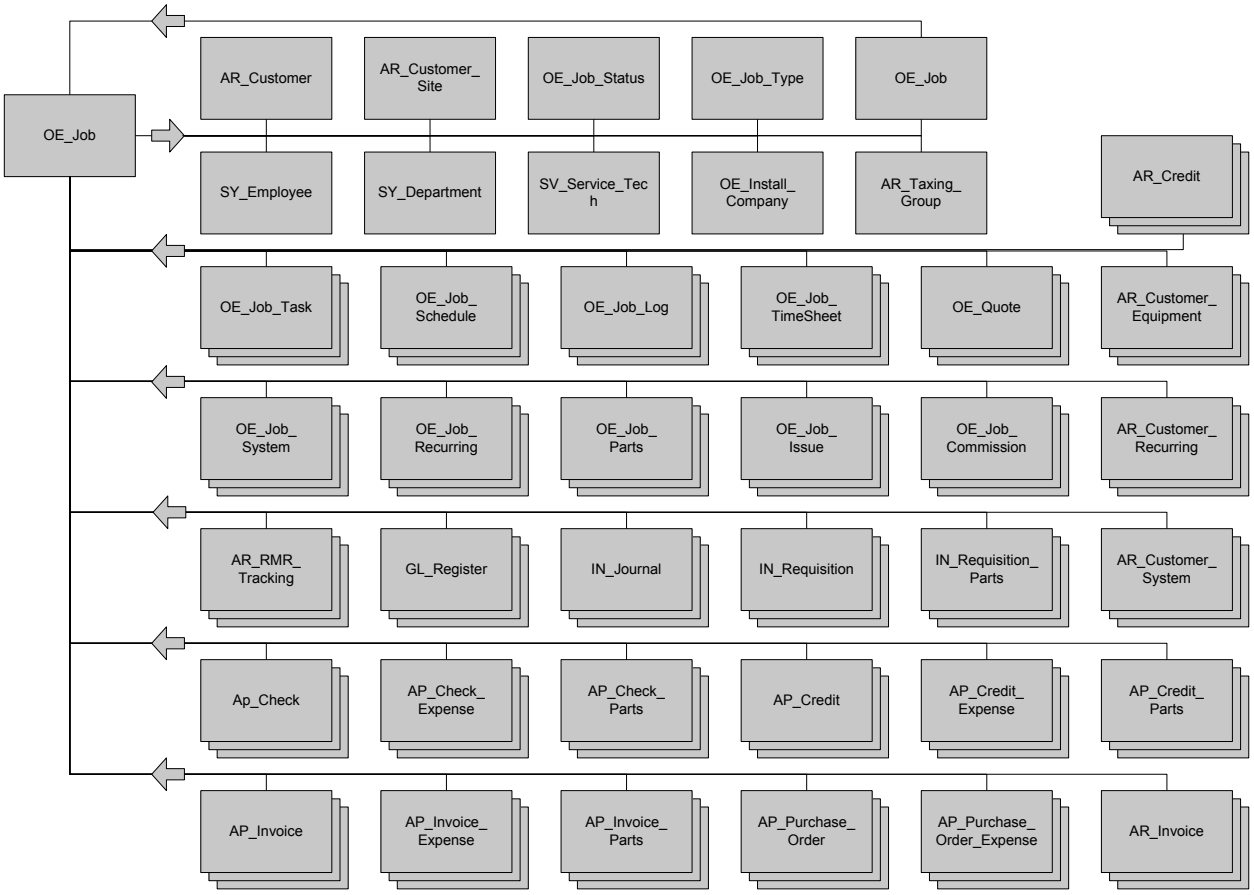
Vendor Bills Structure



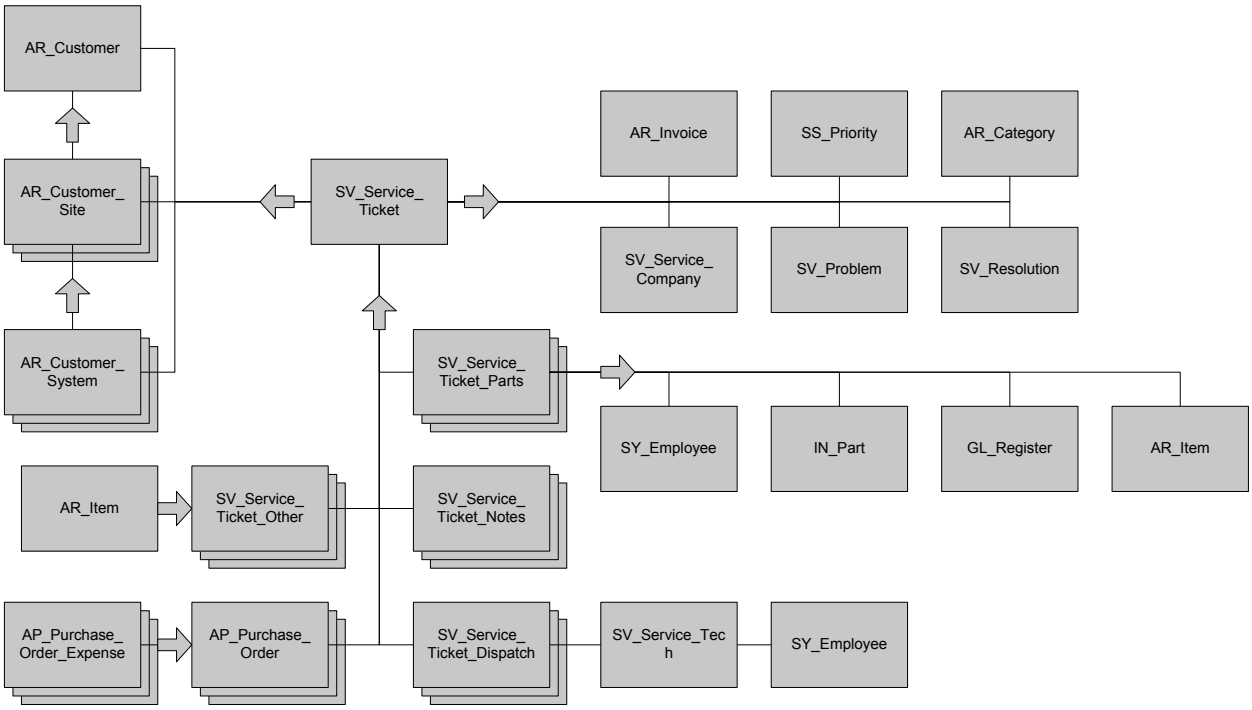
Check Structure



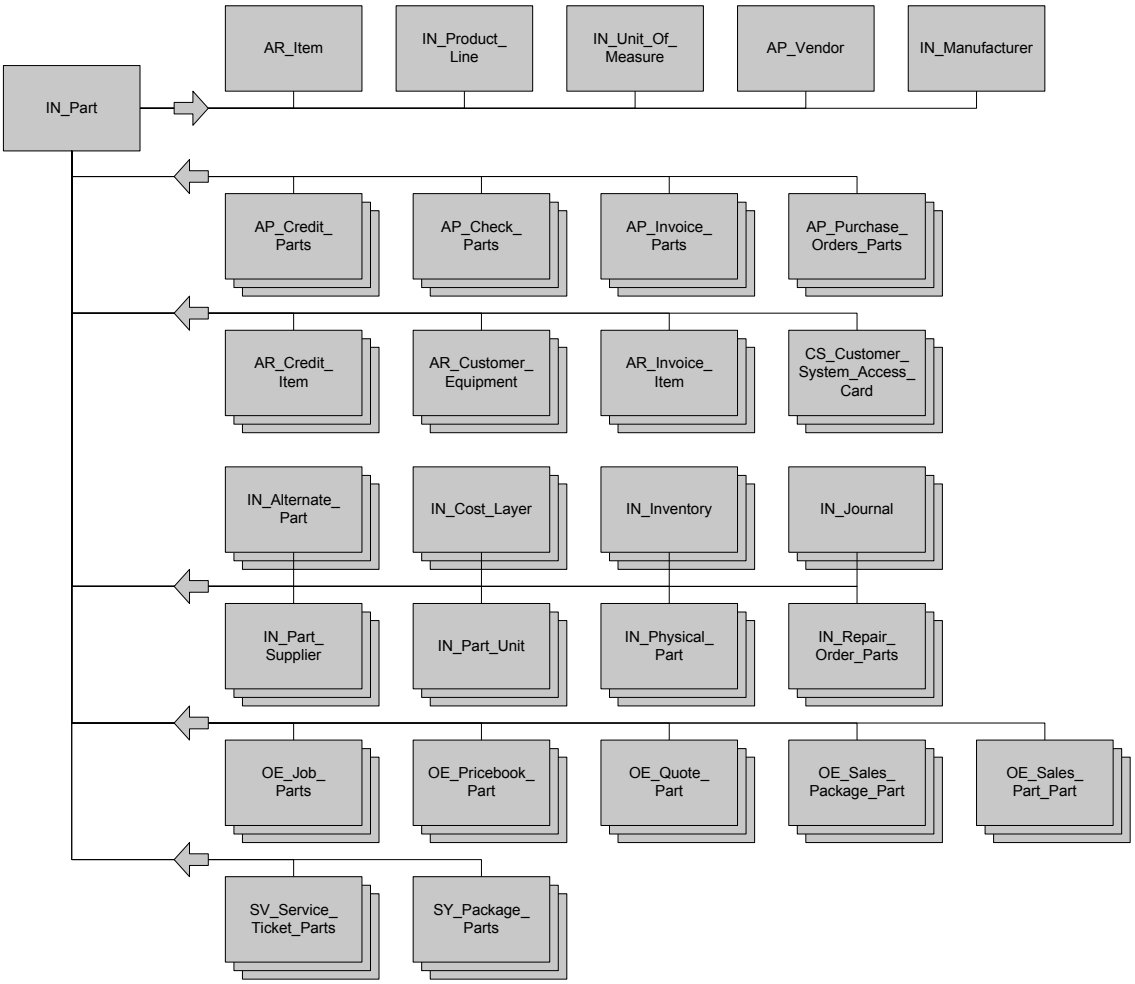
Job Structure



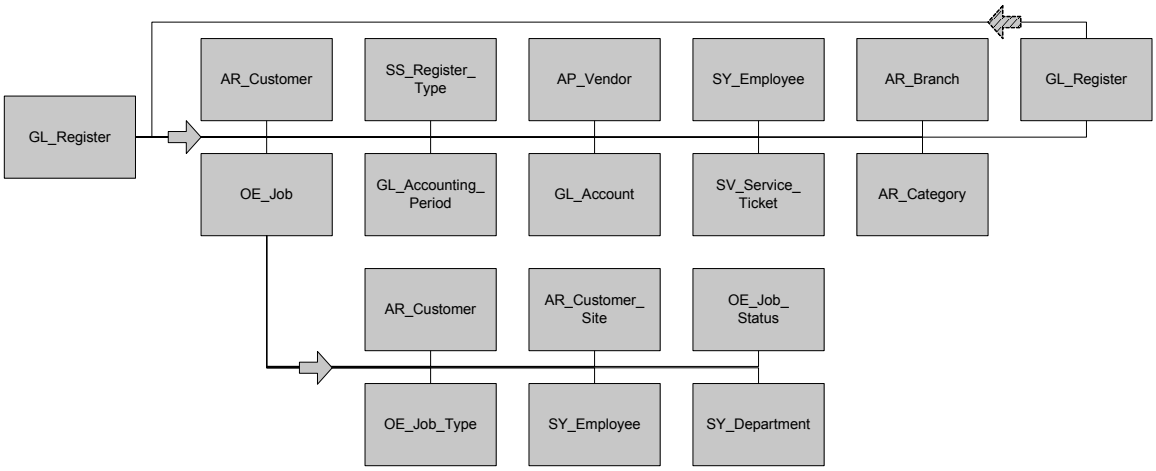
Service Ticket Structure



Inventory structure



General Ledger Structure

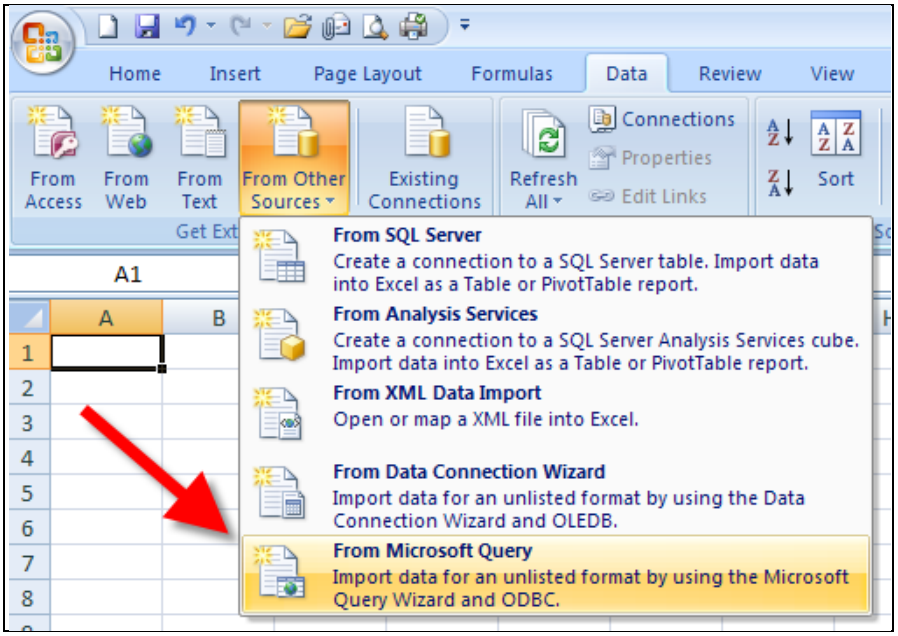


Open Database Connectivity (ODBC)

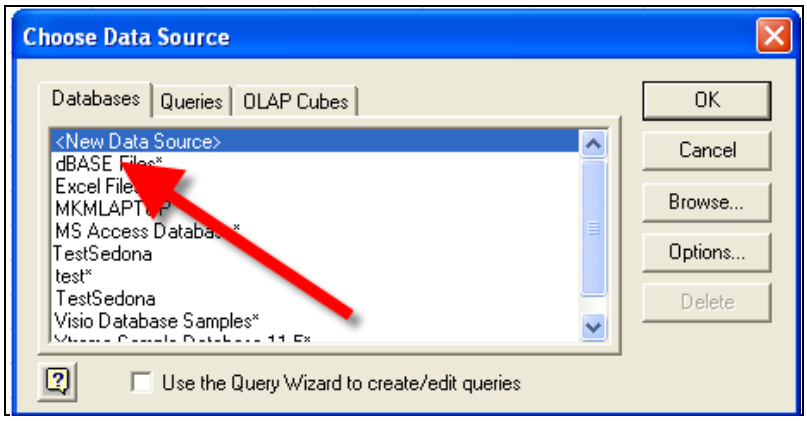
Open Database Connectivity is the methodology created by Microsoft for different applications to talk to different kinds of databases. With ODBC you can connect Excel to Microsoft SQL server or MS Word to Excel for example. The first step in connecting any application to your Microsoft SQL database is 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 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 need to be installed as Excel does not install it by default in the standard install.

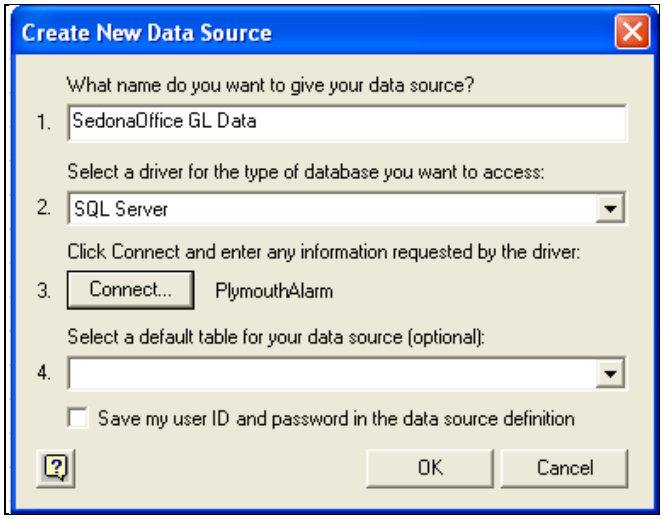


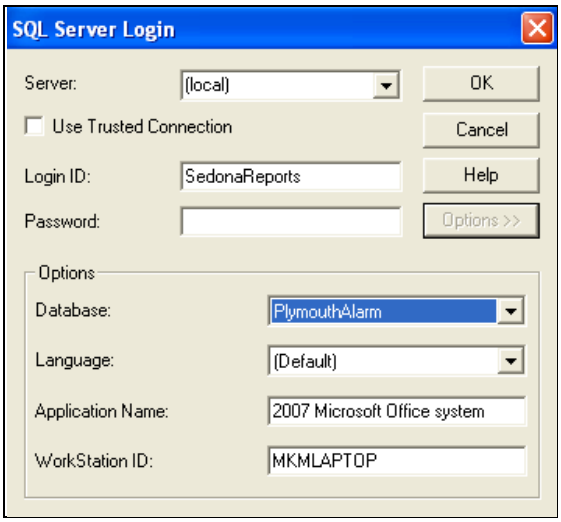
If you have not already done so, you will need to create a Data Source connection to your SedonaOffice database.



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.

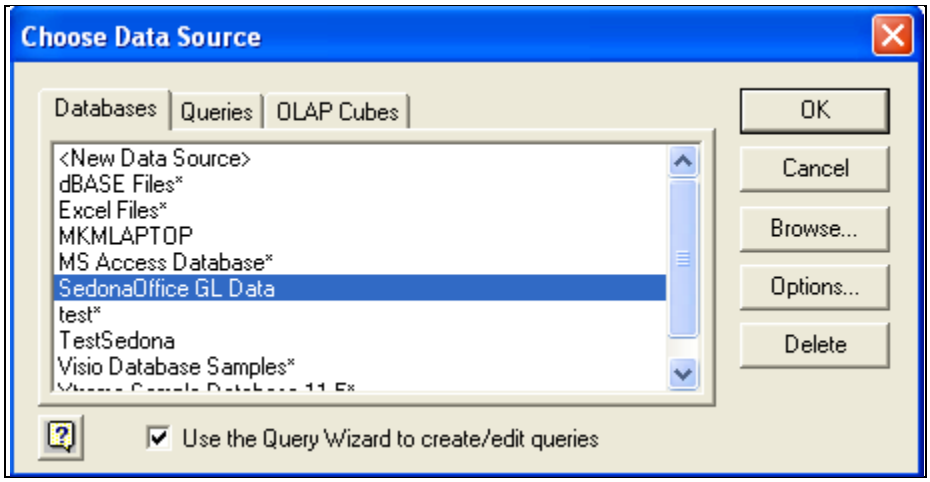




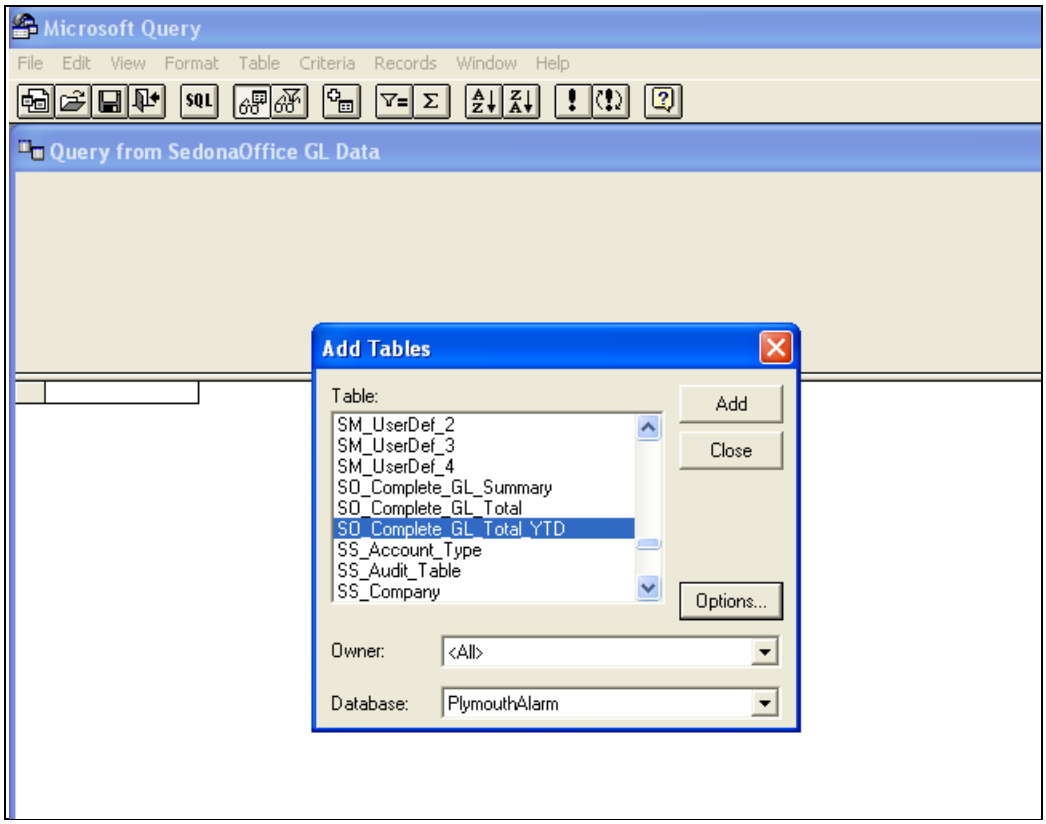
You now have an ODBC connection to your database.

Building a Query Using Excel and MS Query

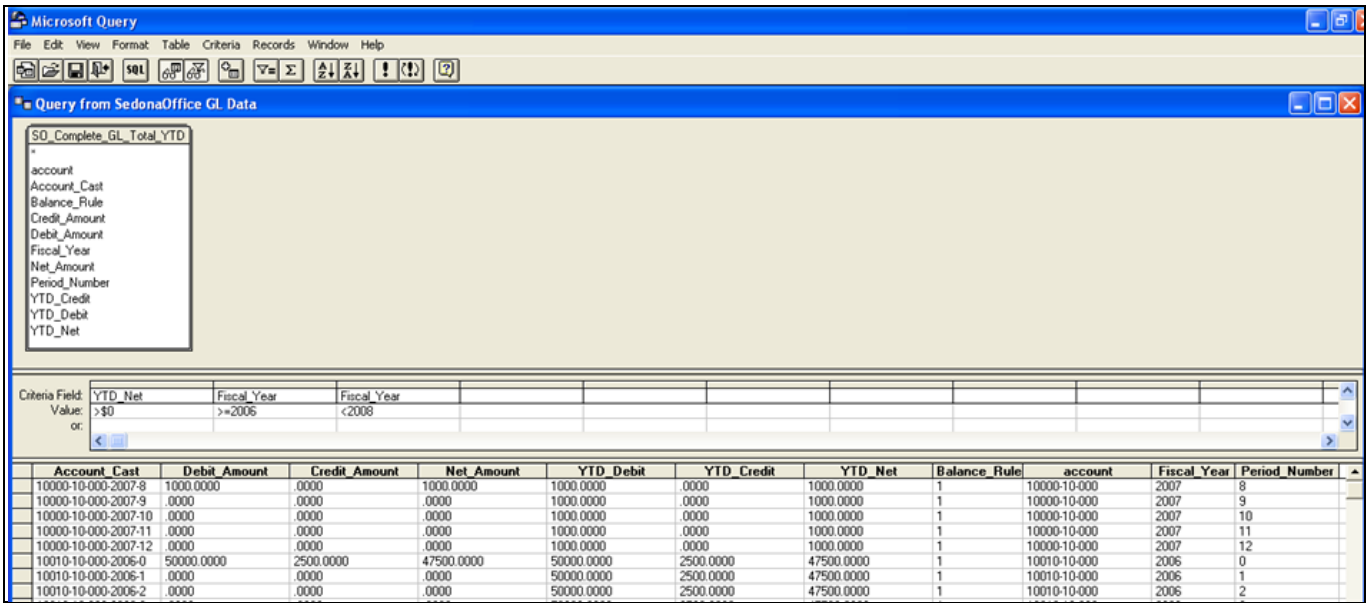
Select the data source you have previously created to create the Query. Uncheck the "Use the Query Wizard..." this will take you directly to Microsoft Query to create the Query.



To begin, you need to select the Table file to use in the Query. Select the "SO_Complete_GL_Total_YTD" table. Then click Close.



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.

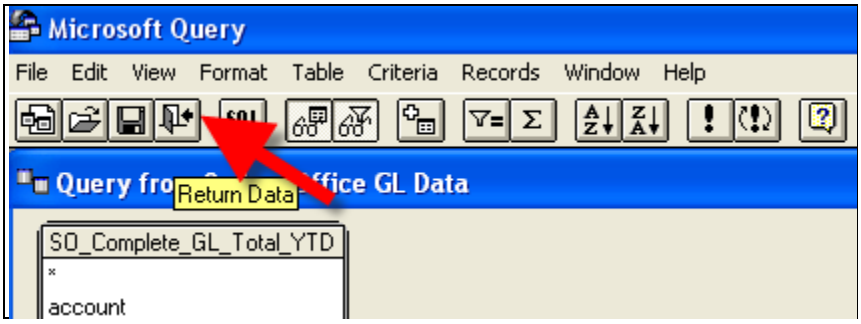


Since this table can contain thousands (or hundreds of thousands) of records, it is best to use some criteria to limit the data that is returned.
Criteria Selections:

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 we are only reporting on 2006 and 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.



	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.***

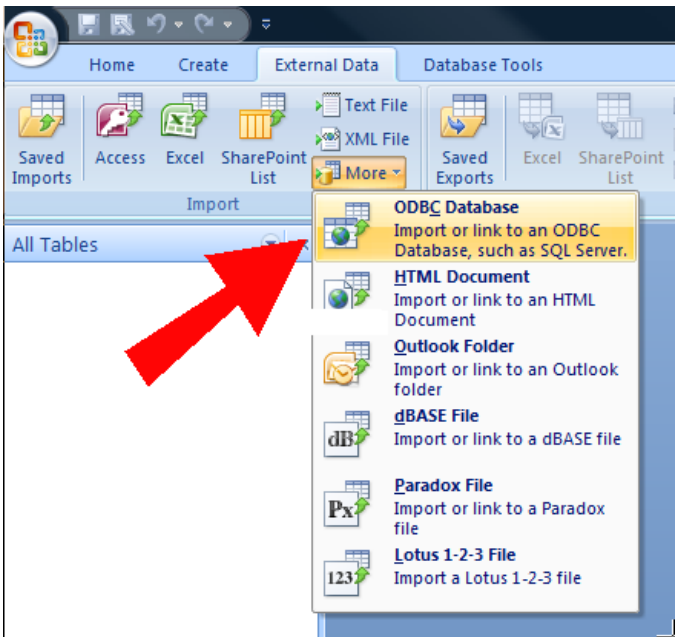
Connecting Access via ODBC

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

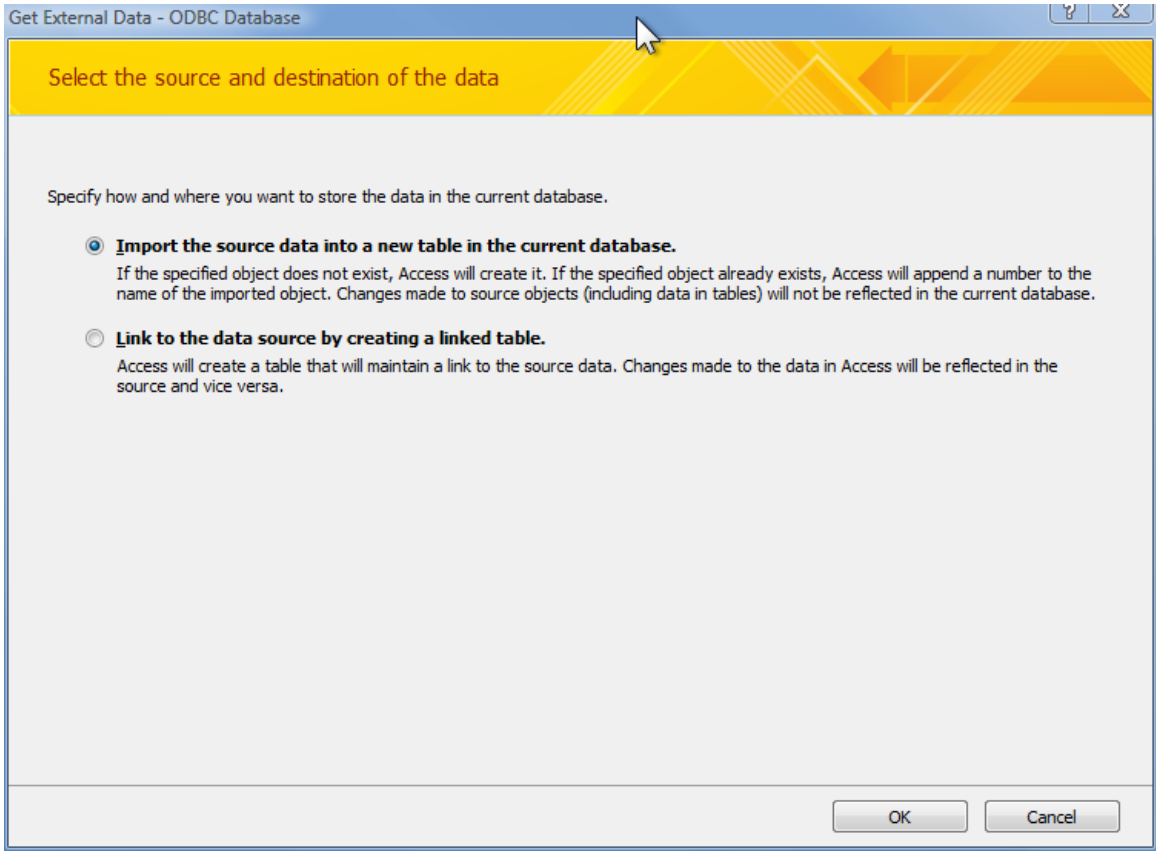
When you Import data into Access, you create a copy of the data stored within the Access database. This allows you to review the data when not connected to the database. Like Excel, you have to periodically refresh the data to keep it up to date.

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

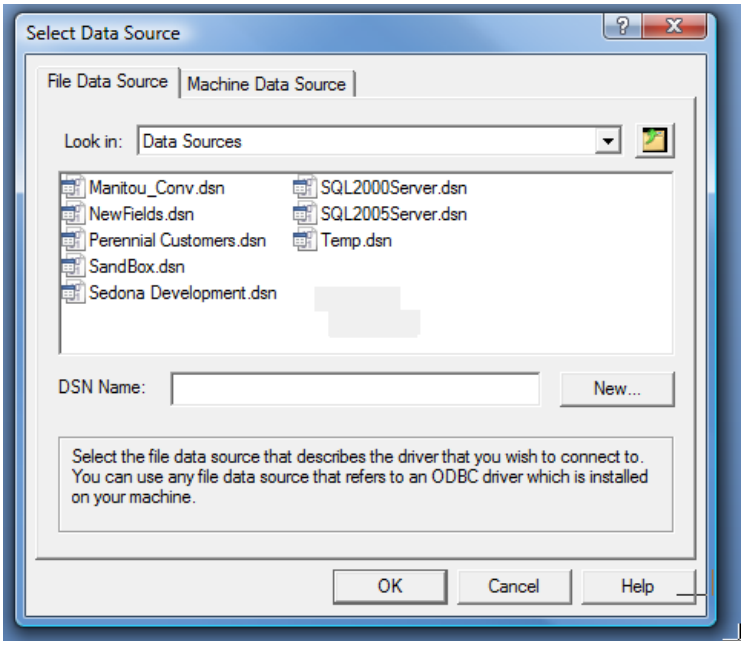
Choose the External Data tab. Then choose More. Finally choose ODBC Database.



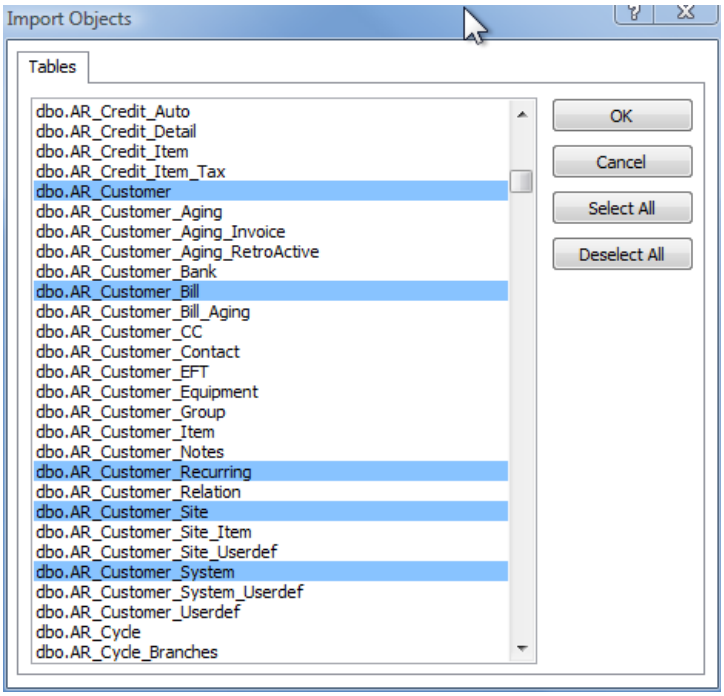
Choose Import or Link and then click OK.



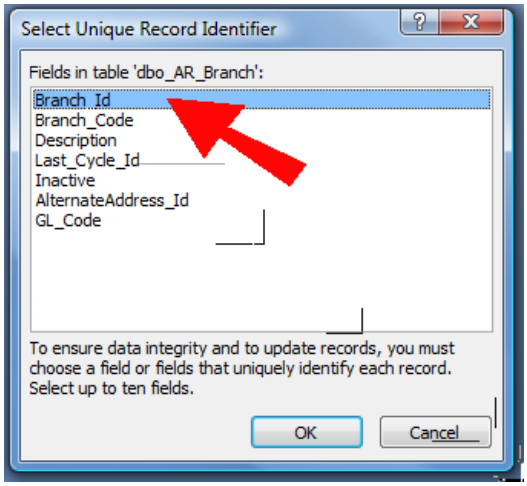
Choose your Data Source.



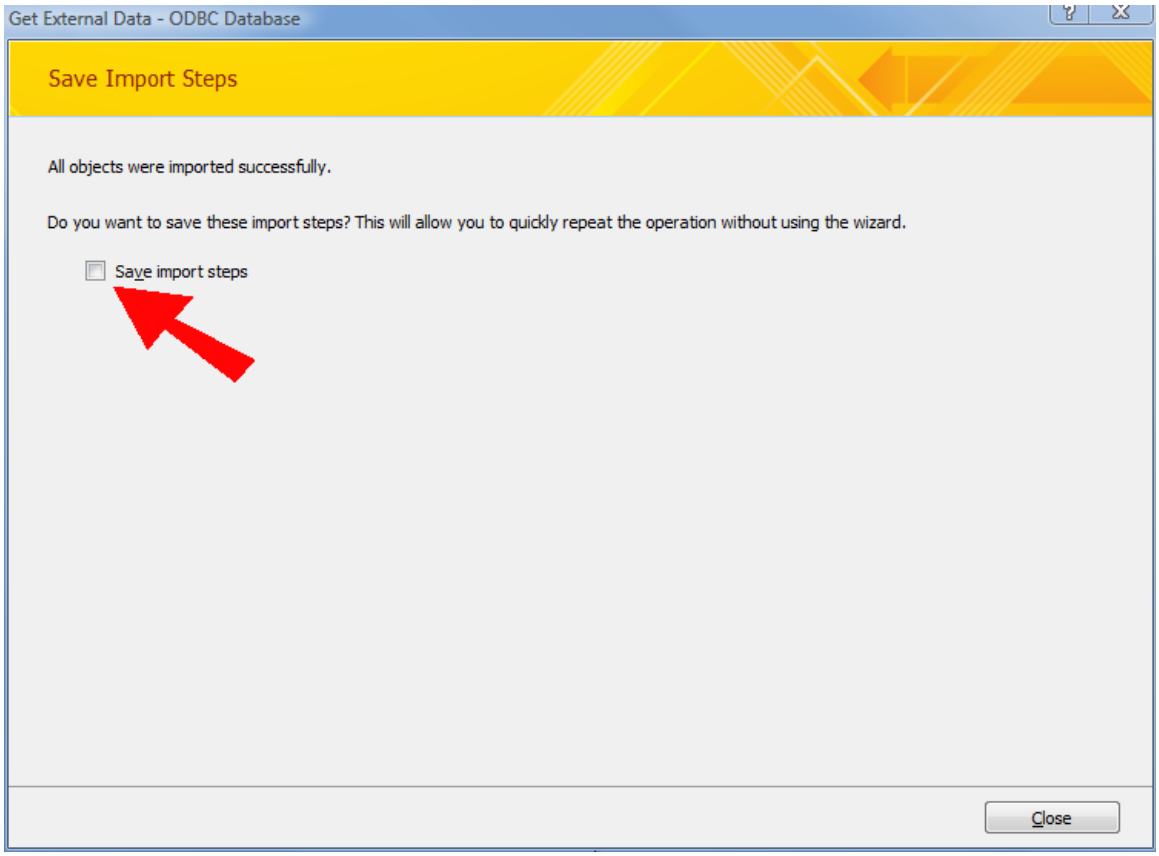
Then choose the tables you wish to Import or Link and click OK. You can choose multiple tables but do not select all. Access is not as large or as powerful as SQL Server. Choosing all will probably crash Access.



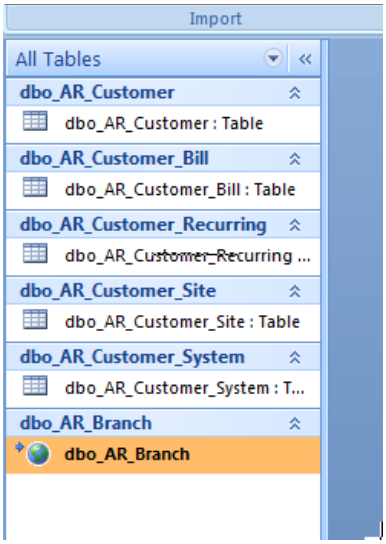
If you chose to Link, you will be asked to "Select Unique Record Identifier". This is always the top item in SedonaOffice.



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 completed so that refreshing the data will be easier.

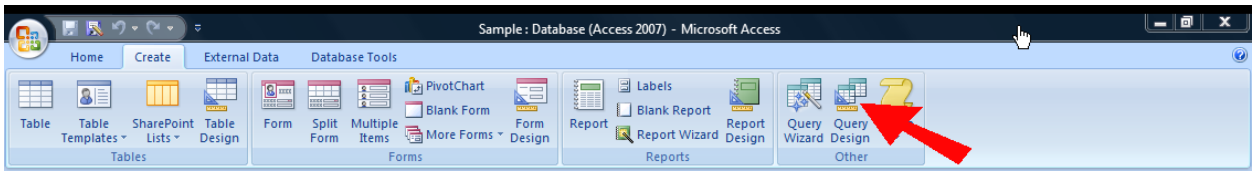


Your tables will then be accessible in Access. You may mix Import and Link in the same Access database. In the example, I have Imported several customer tables and linked the branch table. Notice the different icons for imported versus linked tables. The highlighted table is the linked branch table.

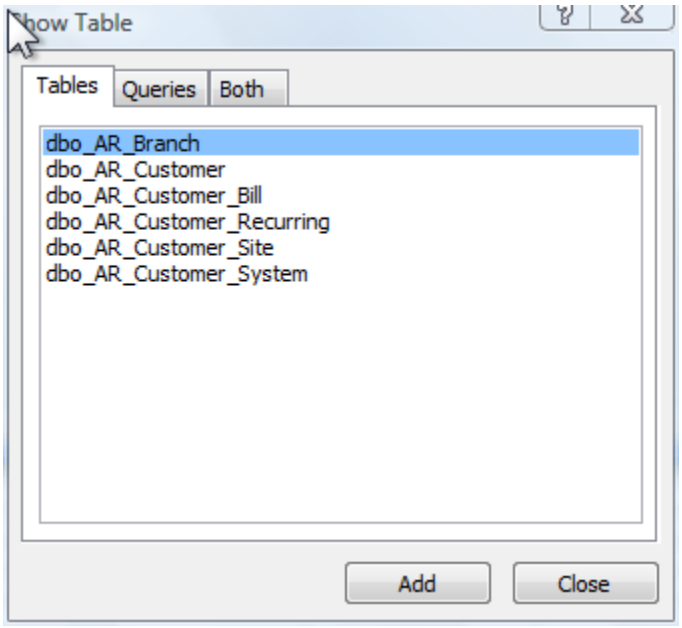


Writing a query with Access

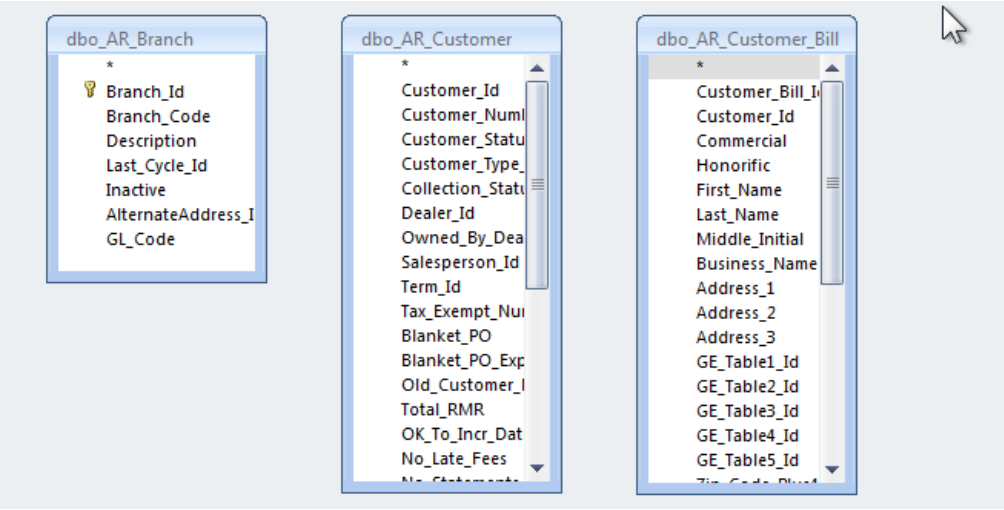
Click on the Create tab and then on Query design.



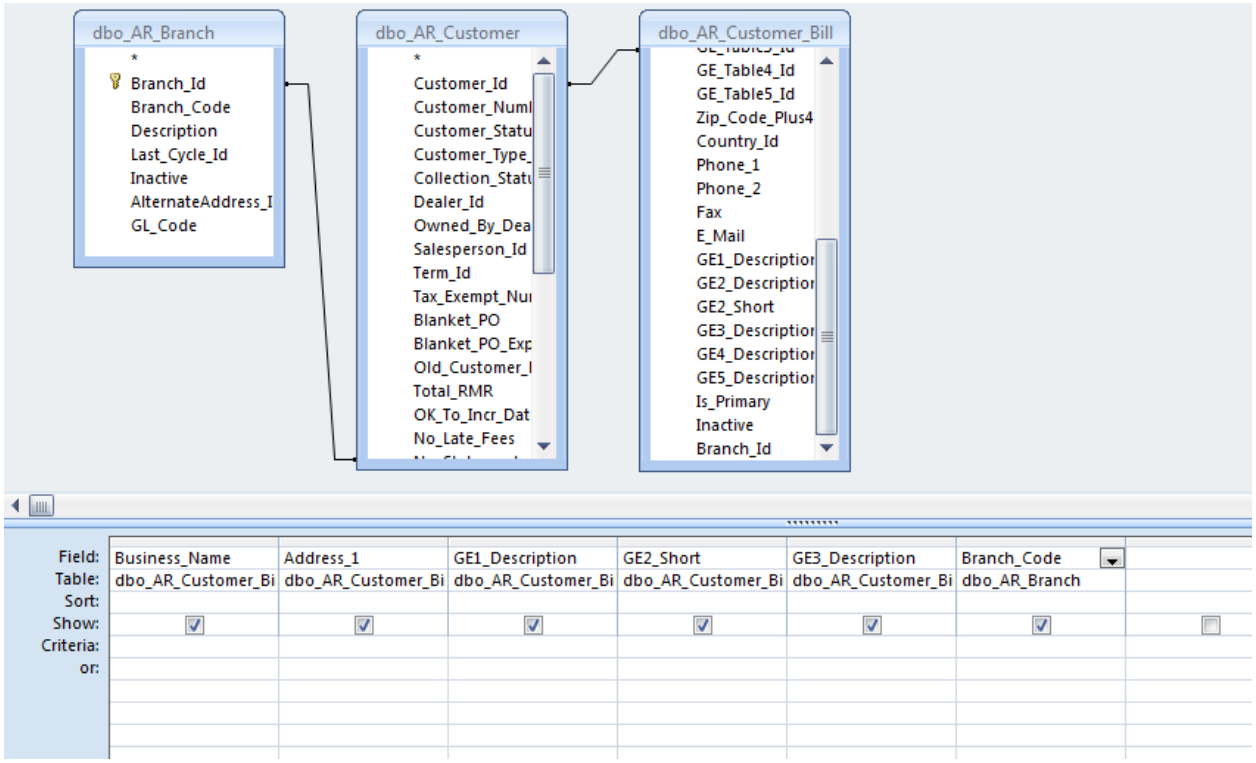
Choose the tables you wish to include in your query. A table can be selected more than once if you need to join it to more than one Id. For our example we are going to choose all of the tables.



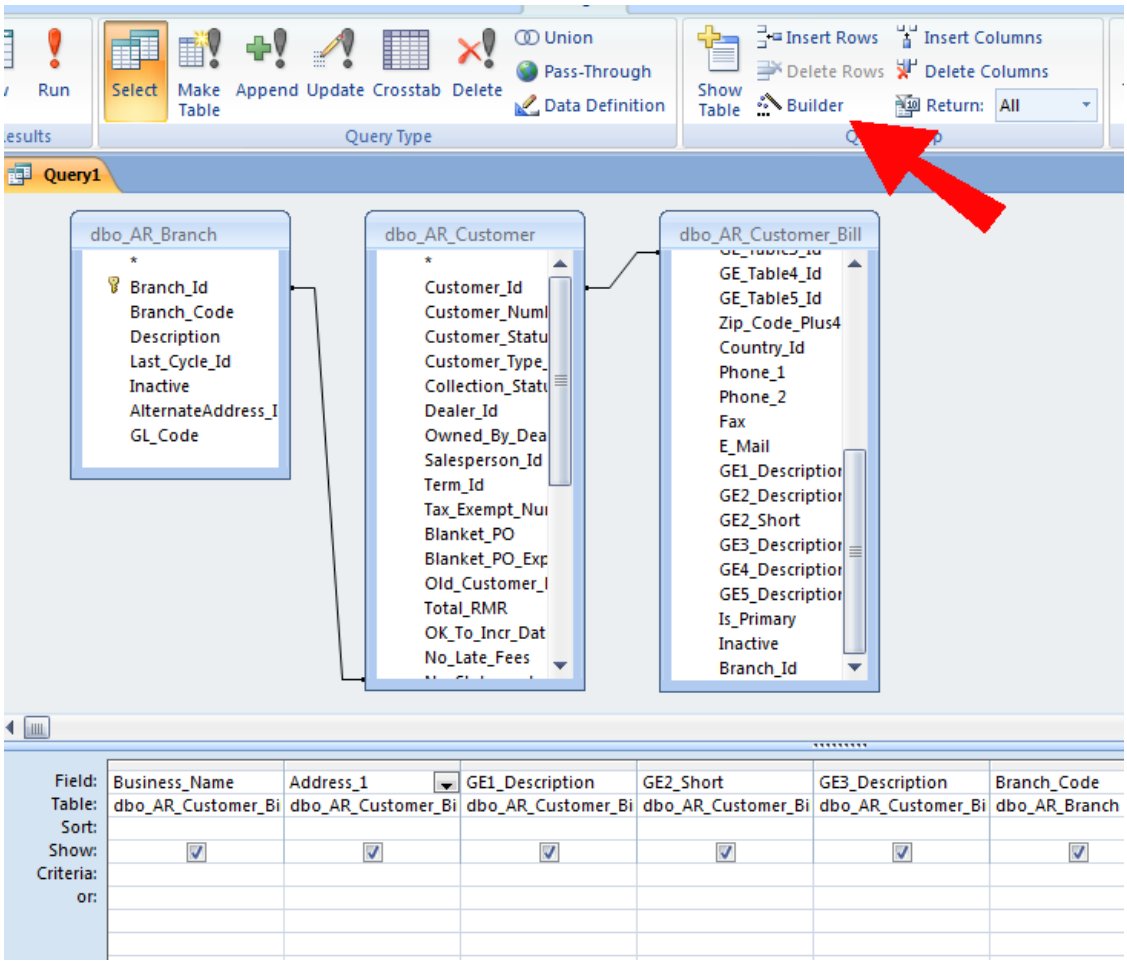
Delete all of the joins that access automatically creates.



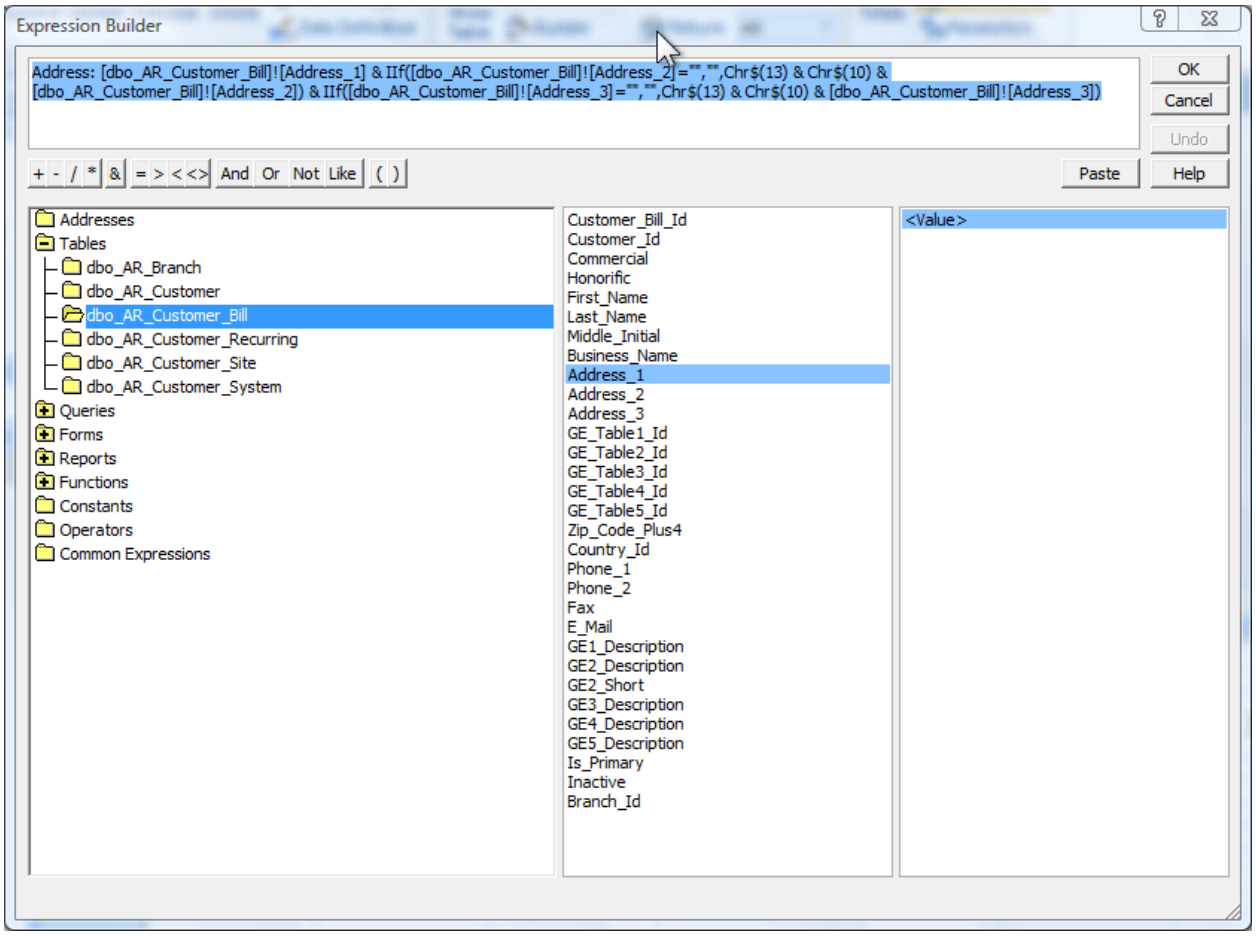
Create the joins according to the structure of SedonaOffice. In this case AR_Branch.Branch_Id to AR_Customer.Branch_Id and AR_Customer.Customer_Id to AR_Customer_Bill.Customer_id.
We are going to create a mailing list, so we need to drag the name and address information to the lower pane. We are also going to drag down the branch code so we can sort on branch.



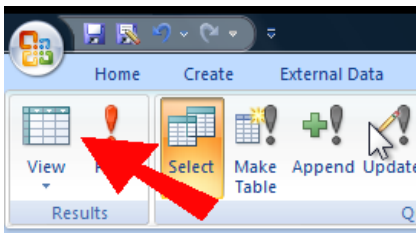
Address_1 may not be all of the address information needed, but if there is no address_2 we don't want to add a blank line. So we create a formula. Click in the Address_1 cell and then click 0n the formula button.



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])



Click view to test our results.

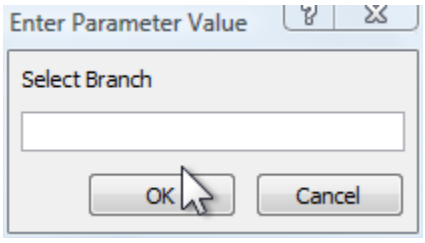


Business_Name	Address	GE1_Description	GE2_Short	GE3_Descrip	Branch_Cod
N/A		N/A	N/A	N/A	Main Division
George Washington	1234 Mount Vernon Lane	Colorado Springs	CO	80919	Major
John Adams	5411 2nd Street	Colorado Springs	CO	80919	Major
Mega Mart	2154 Mountain Springs Road	Colorado Springs	CO	80919	Major
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

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.

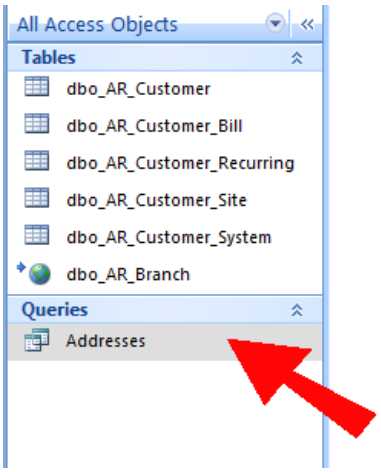


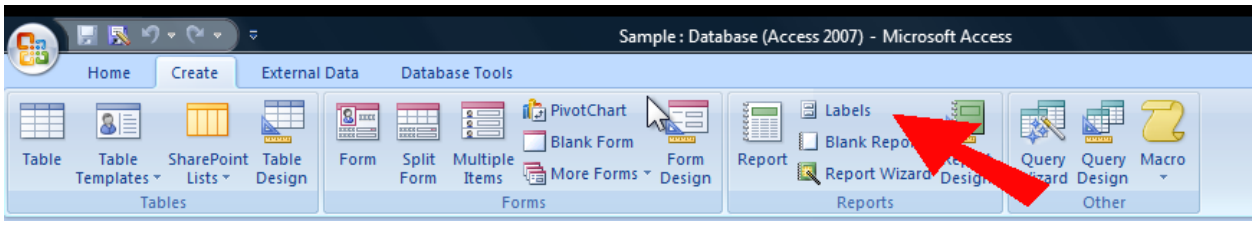
Entering a branch we get results with no N/A.

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

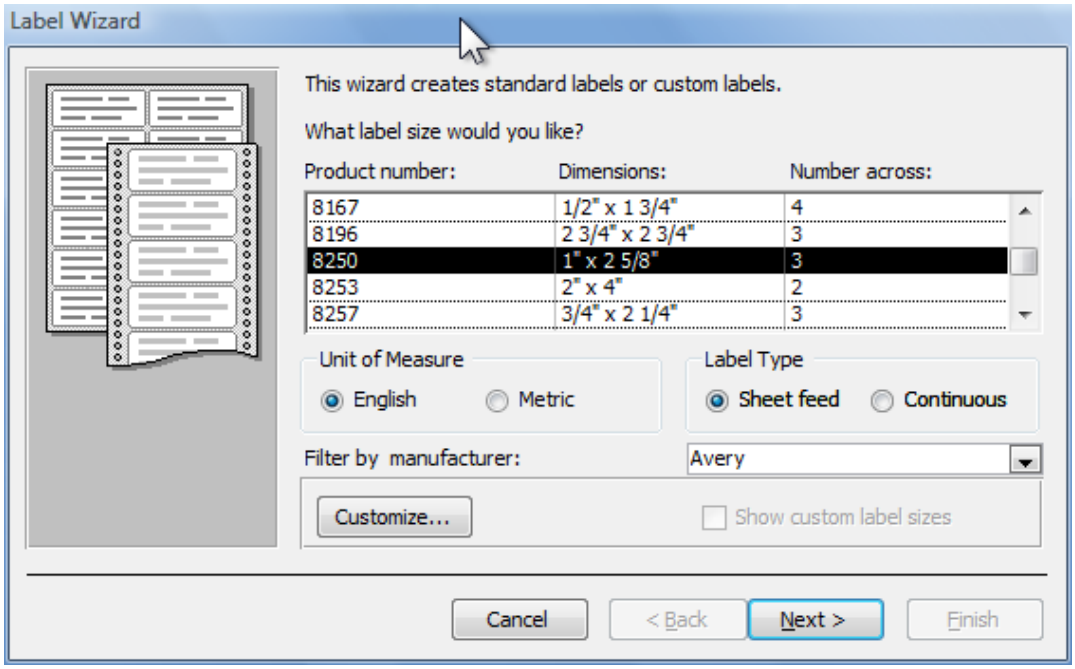
Creating a Report with Access

Displaying the results on the screen is useful but Access allows us to create reports. The report we are going to create will be used to create mailing labels.
First make sure the new query you created is selected and then launch the label wizard.

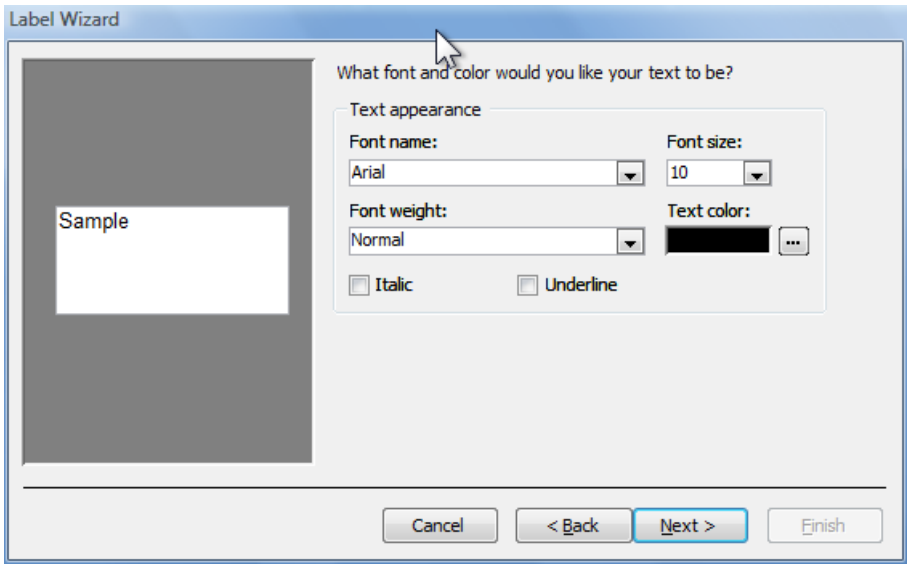




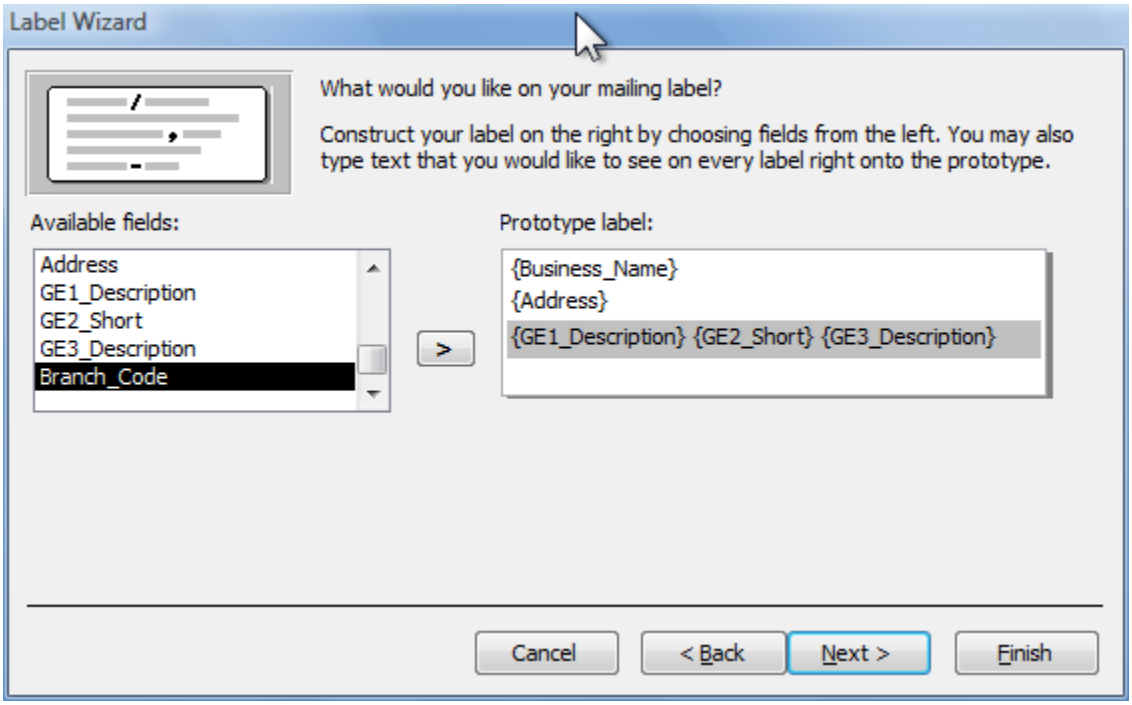
Choose your label. You can choose by the form number if you bought labels from a major manufacturer or just choose a label of the same size as the ones you are using.



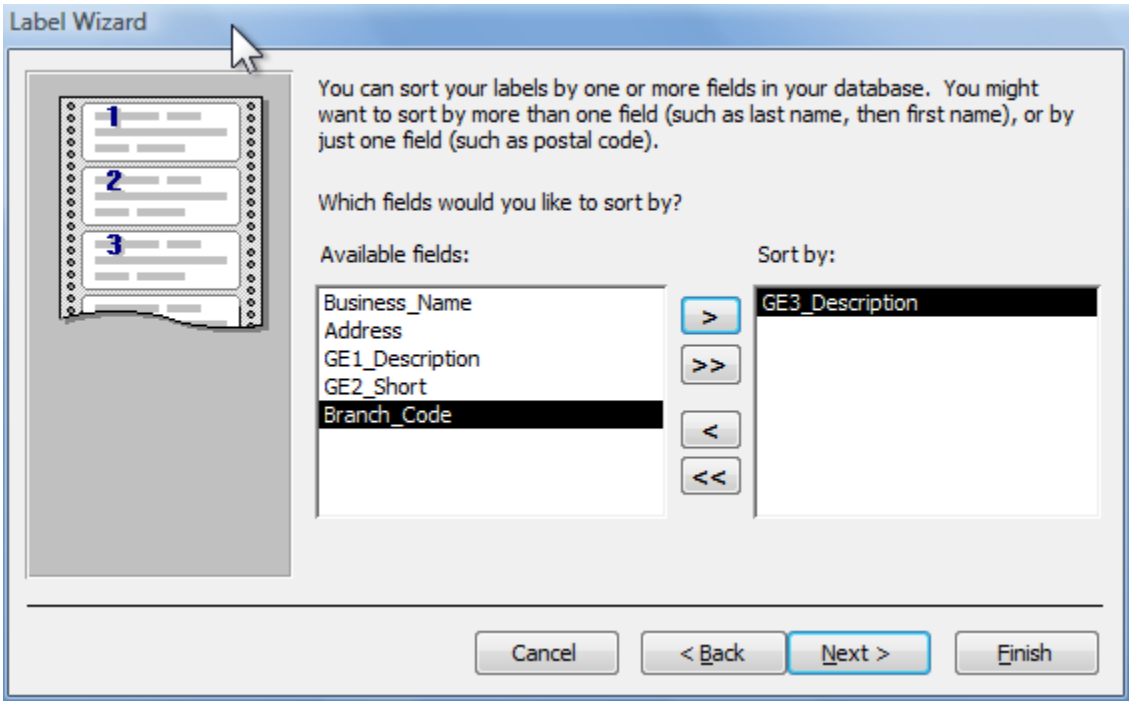
Choose your font.



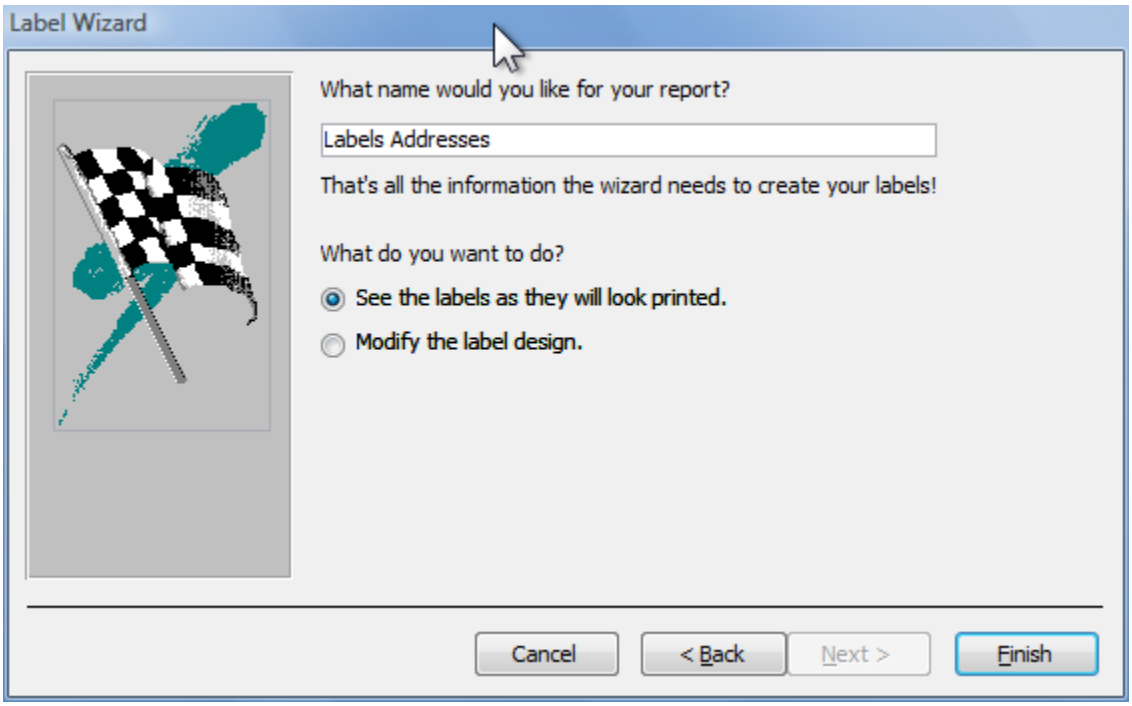
Setup the fields how you want them to appear on the label.



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.



Give your report a name and save it.



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

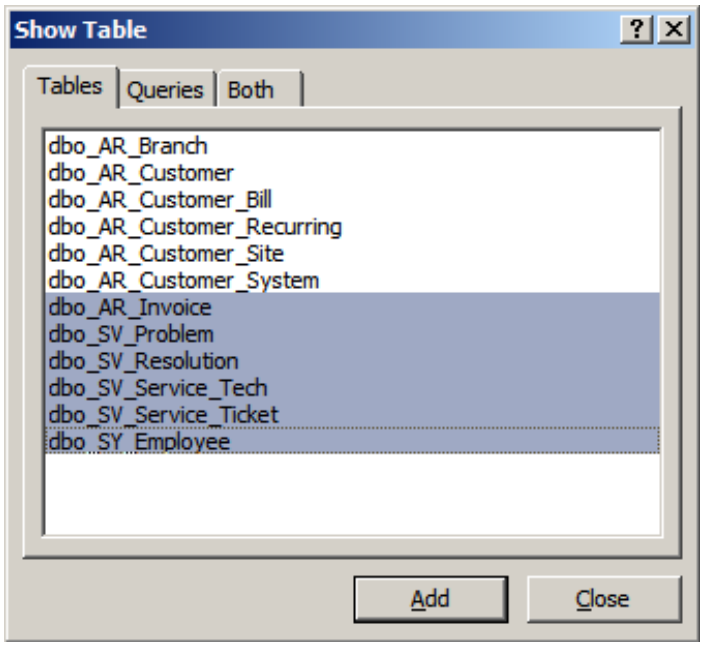
TELUS
1234 Main Street
Colorado Springs CO 80919

Creating a Grouped and Sub Totaled Report

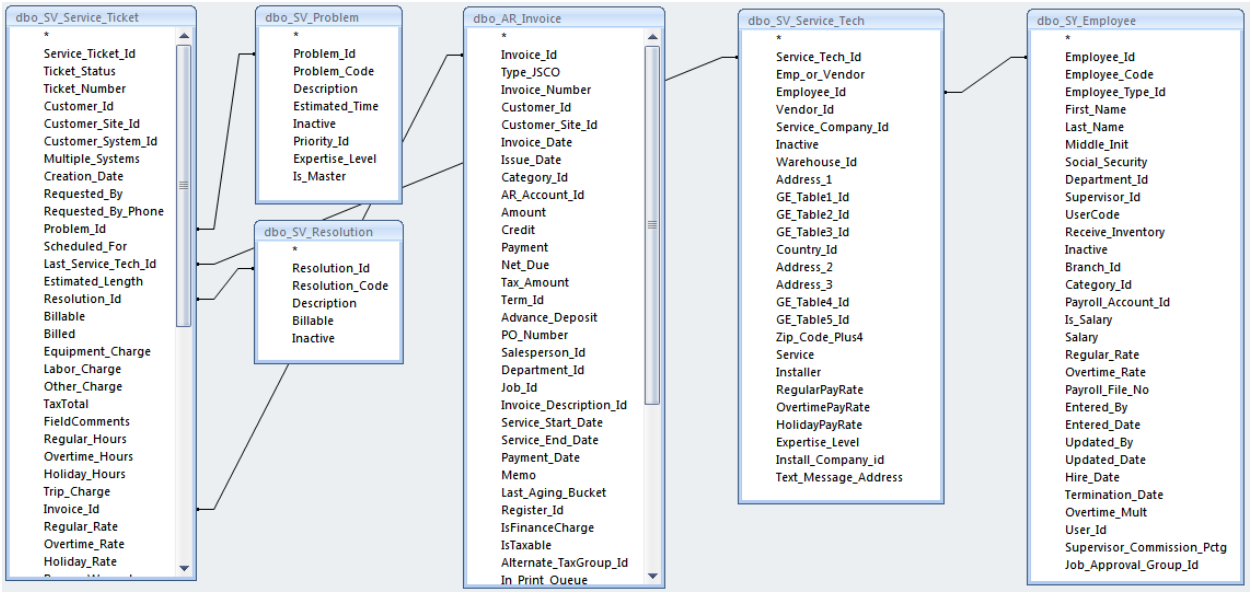
First we will need some additional data. Again select the ODBC database import item. Add these additional tables:

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

Create a new Query and add these tables.



Link the tables as shown.



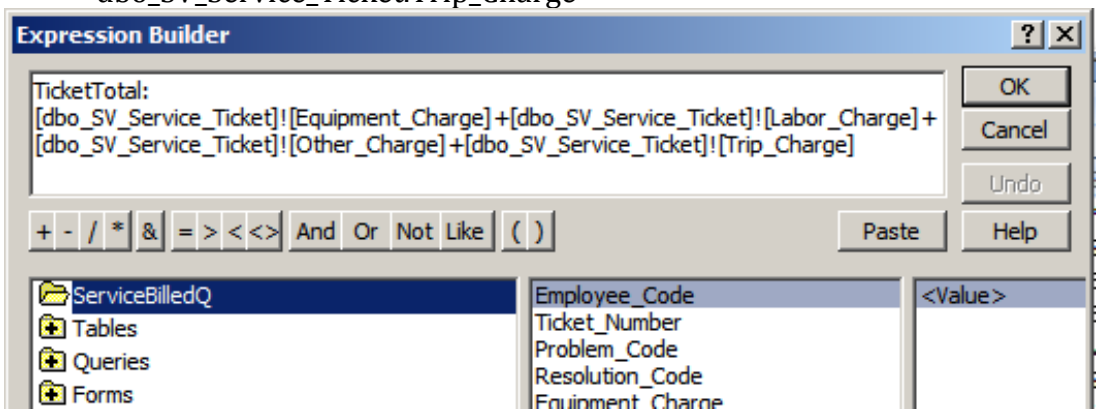
Add these fields.

- 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

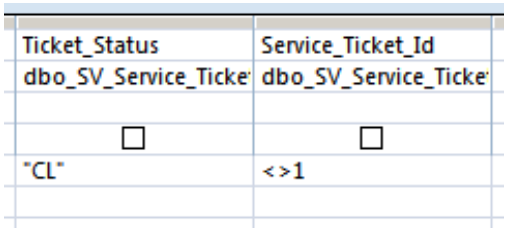
- dbo_SV_Service_Ticket.Other_Charge
- dbo_SV_Service_Ticket.Trip_Charge
- dbo_AR_Invoice.Invoice_Number
- dbo_AR_Invoice.Amount
- dbo_SV_Service_Ticket.Ticket_Status
- dbo_SV_Service_Ticket.Service_Ticket_Id

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.

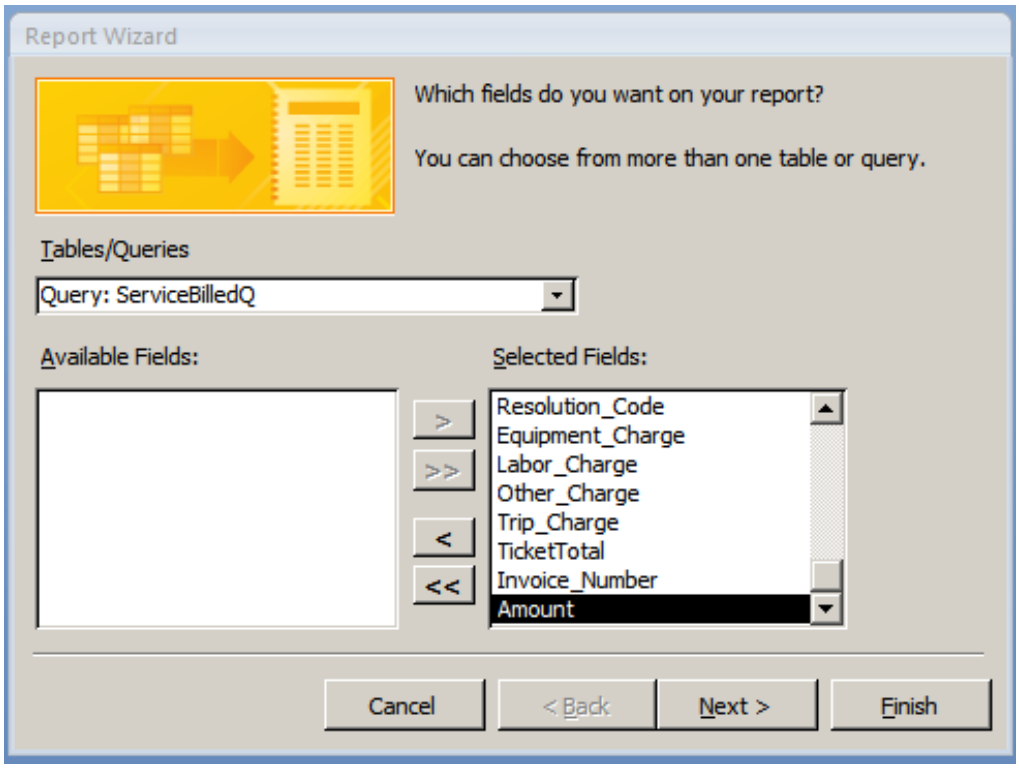
- dbo_SV_Service_Ticket.Equipment_Charge
- dbo_SV_Service_Ticket.Labor_Charge
- dbo_SV_Service_Ticket.Other_Charge
- dbo_SV_Service_Ticket.Trip_Charge



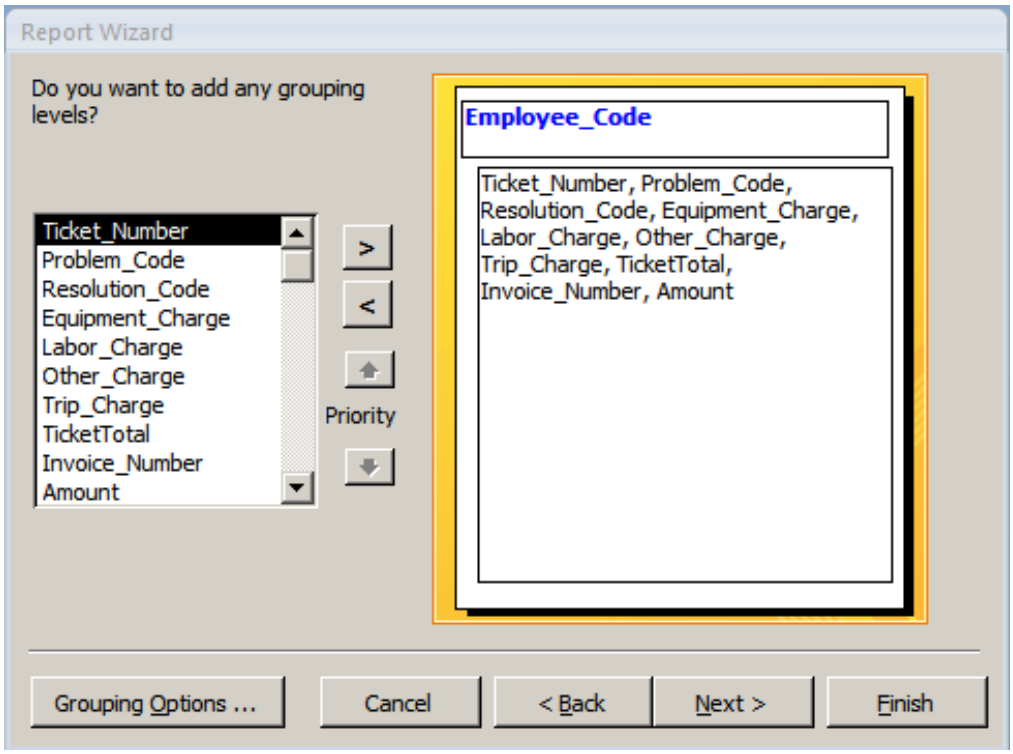
And select OK.
Also add criteria for Ticket_Status and Service_Ticket_Id.



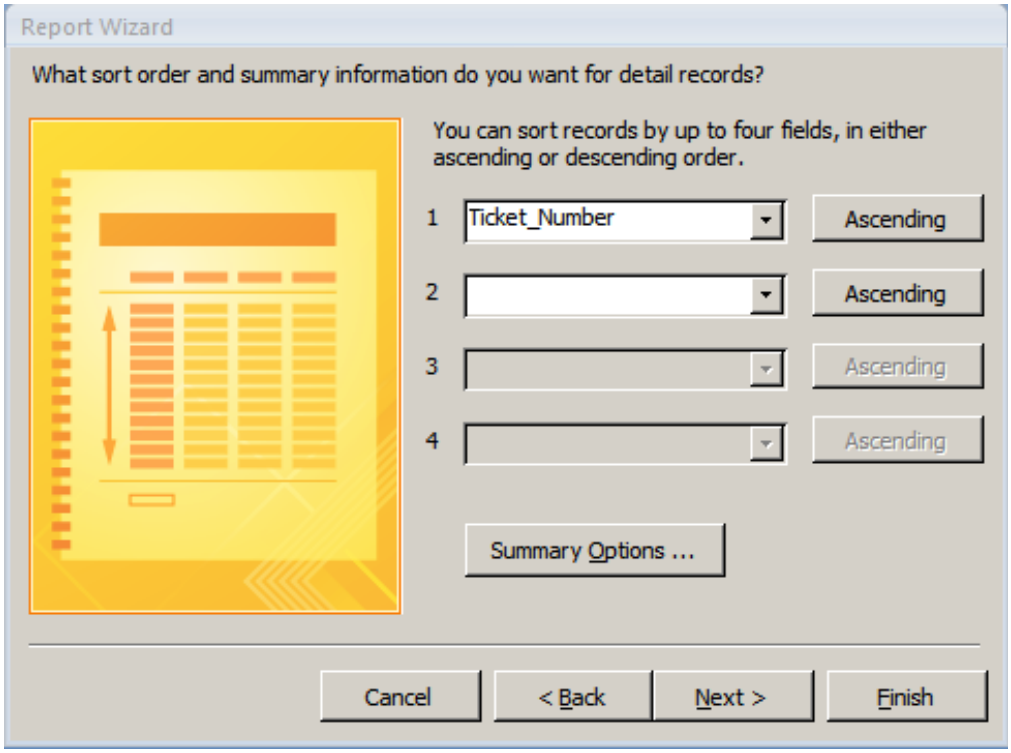
Save the query as ServiceBilledQ. Create a new report and select ServiceBilledQ as the data source, select all of the fields and press next.



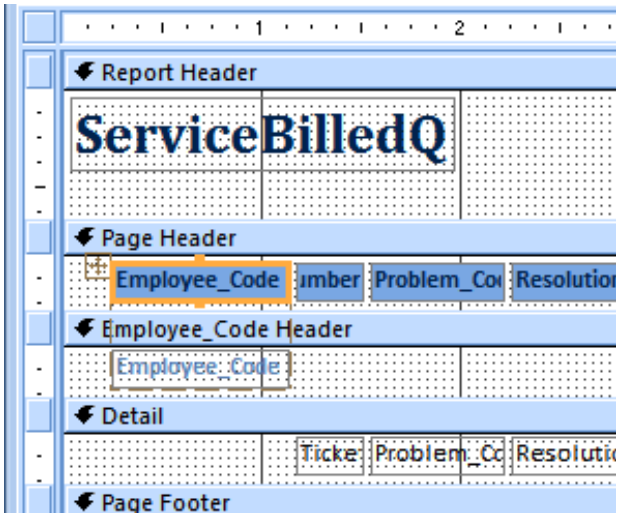
We are going to group by Employee_Code. Select the right pointing arrow while Employee_Code is highlighted. The result should look like the image below.



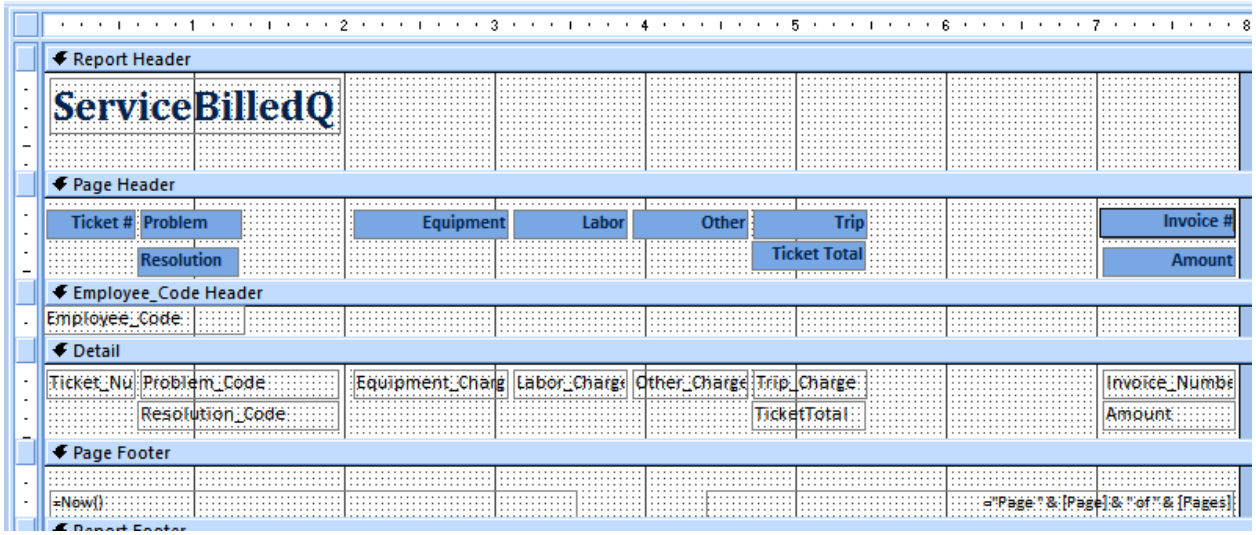
Press Next. We want to sort the tickets by Ticket_Number for each Service_Tech which we can do on the next page of the wizard.



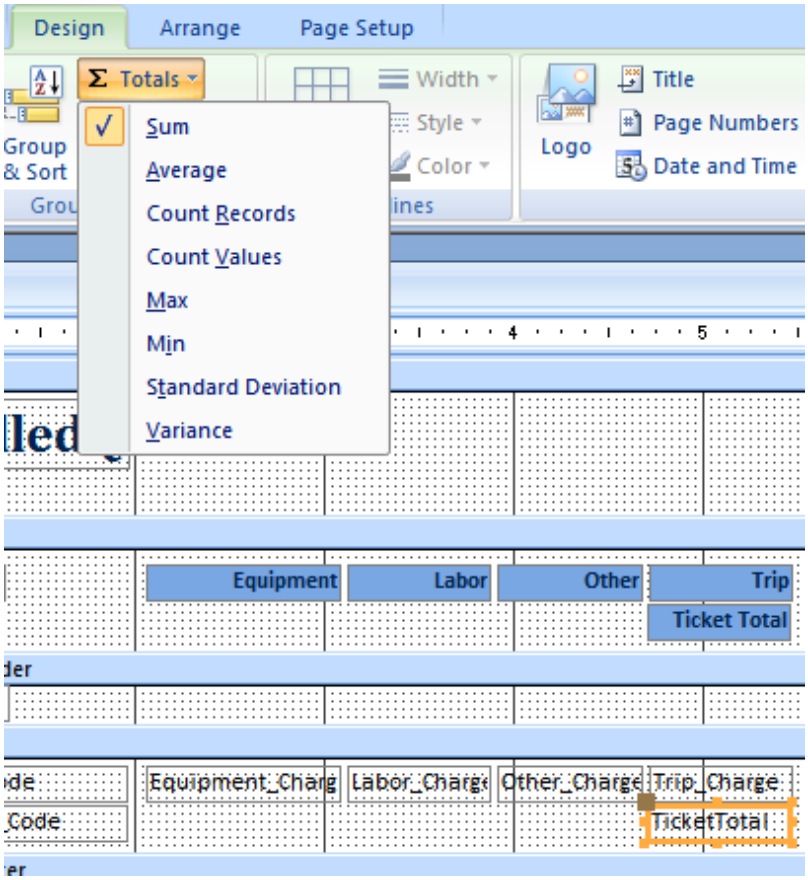
Select Finish. A preview of the report will display. Close the preview for now and you will get the report designer. We can now make changes to the basic report to suit our needs. Click on any blank spot on the report to deselect everything. Now select the Employee_Code label.



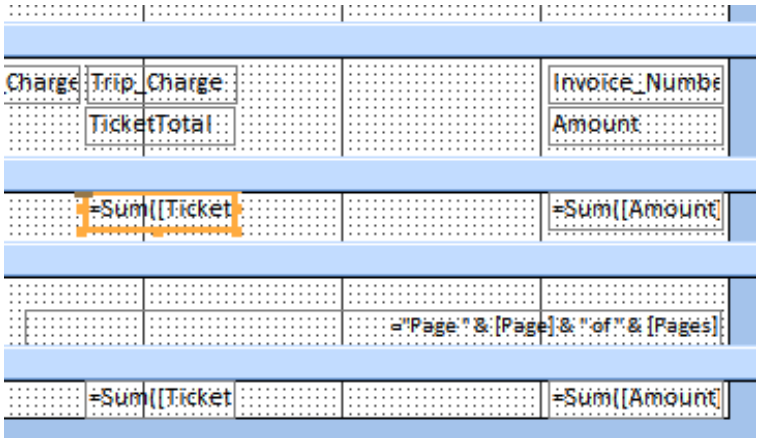
Delete the label. If you delete the Field just drag it back from the fields list. Continue moving fields till your layout looks like the image below.



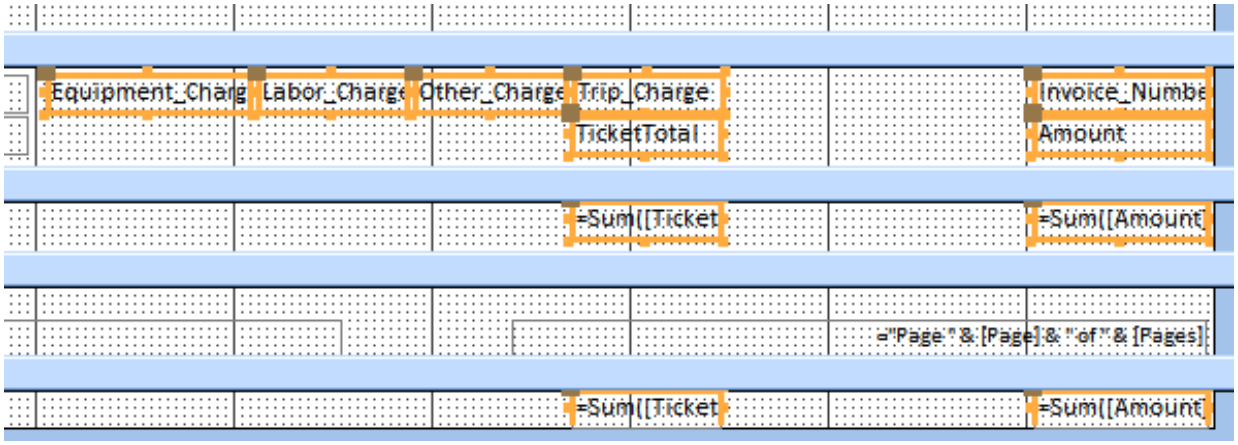
Now we are going to add subtotals. Select the TicketTotal field in the report detail section, then choose the Totals menu and Sum in the Design bar.



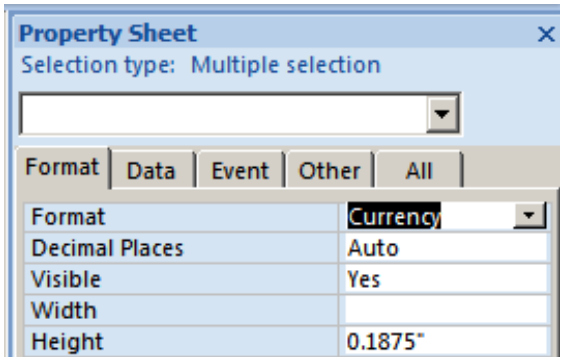
Do the same for the Amount field from the invoice. We will now have a subtotal by service tech and a grand total for all service techs.



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



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



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**

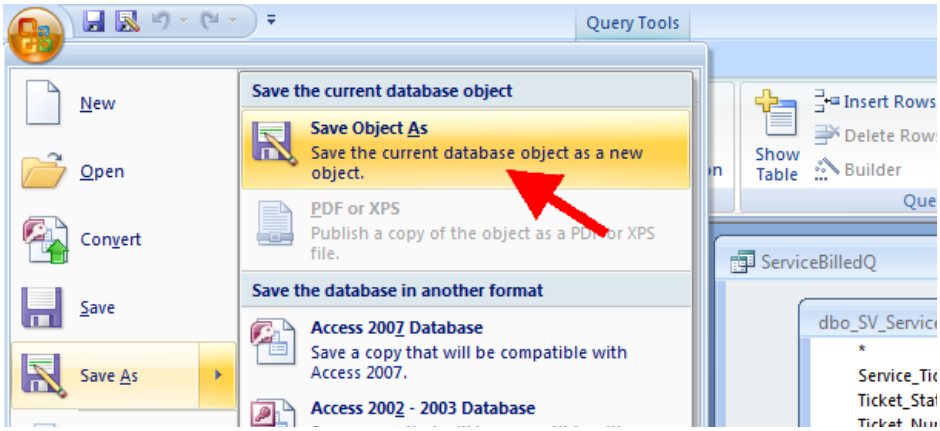
There are a number of improvements that could be added to the report. The title should be changed. Dates could be added. Perhaps some additional Customer information could be included.
In this report we have learned how to group and total. We have learned how to expand the detail section to show more data than will fit on one line.

Using SedonaOffice Data in a Microsoft Mail Merge

Now we will create a mail merge using Access and Word. For our example, let's create a query of customers that do not have a service contract and were charged for a service call last year. From this data we will create letters offering these customers a service contract.

Creating the List of Customers

We will use the query we created in our last example as a starting point. Open the ServiceBilledQ query from the last example and save it as MailMergeQ.



Again select the ODBC database import item. Add these additional tables:

- AR_Customer_Recurring
- AR_Item

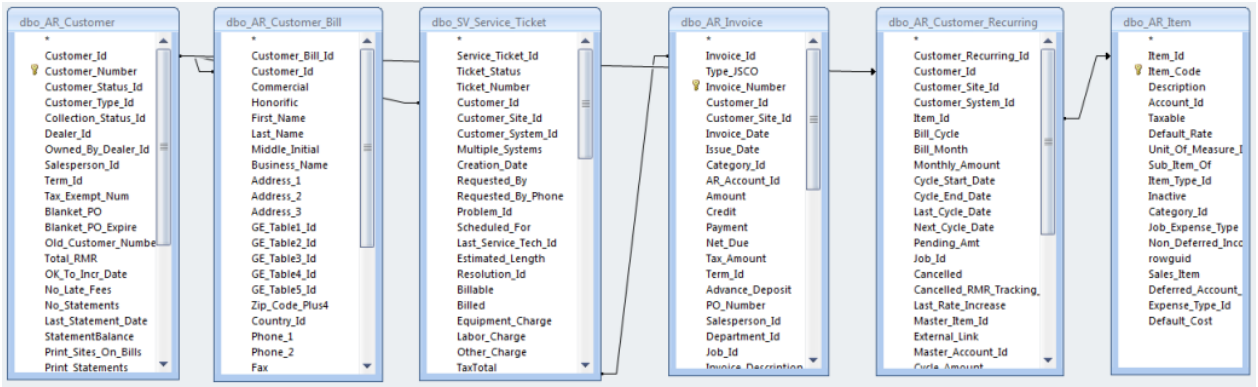
Then, using the table list, add these tables to the new query:

- AR_Customer
- AR_Customer_Bill
- AR_Customer_Recuurring
- AR_Item

Remove these tables:

- SV_Problem
- SV_Resolution
- SV_Service_Tech
- SY_Employee

Link the tables as shown:

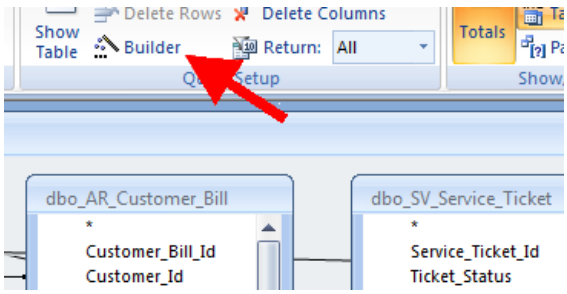


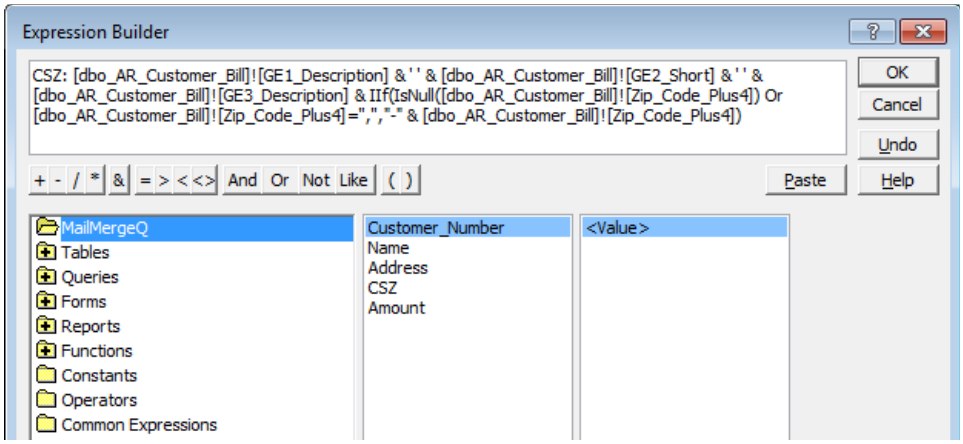
Add and remove fields till you have this list:

- AR_Customer.Customer_Number
- AR_Customer_Bill.Business_Name
- AR_Customer_Bill.Address_1
- CSZ

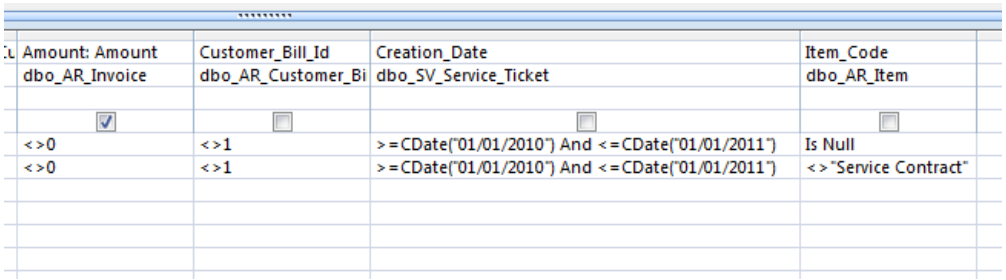
- AR_Invoice.Amount
- AR_Customer_Bill.Customer_Bill_Id
- SV_ServiceTicket.Creation_Date
- AR_Item,Item_Code

Name the Business_Name field “Name” and the Address_1 field “Address”. The CSZ field is a calculated or formula field. You create it with the Builder function.



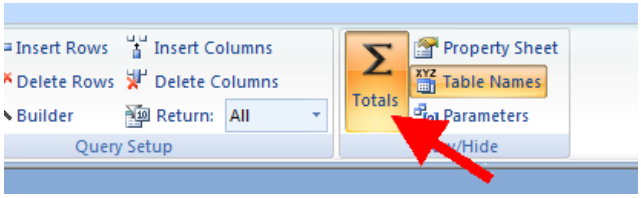


This formula could be simpler but I added an IIf function to format the City State and Zip differently if there was a Zip Plus 4. Add the following “Where” entries:



Amount: Amount	Customer_Bill_Id	Creation_Date	Item_Code
dbo_AR_Invoice	dbo_AR_Customer_Bi	dbo_SV_Service_Ticket	dbo_AR_Item
☑	☐	☐	☐
<>0	<>1	>=CDate("01/01/2010") And <=CDate("01/01/2011")	Is Null
<>0	<>1	>=CDate("01/01/2010") And <=CDate("01/01/2011")	<>"Service Contract"

Under the item code we have the “Is Null” check which is true if they have no AR_Customer_Recurring records and the <> “Service Contract” which is only true if they do not have a Service Contract. Of course this will have to be changed to the code or codes that you use for service contracts. Finally turn on grouping.



Unlike the last report, here we want to use some of the special features of grouping. So for each column, select the grouping actions as follows:

Field:	Customer_Number	Name: Business_Nam	Address: Address_1	CSZ: First([dbo_AR_Cu	Amount: Amount	Customer_Bill_Id	Creation_Date	Item_Code
Table:	dbo_AR_Customer	dbo_AR_Customer_Bi	dbo_AR_Customer_Bi		dbo_AR_Invoice	dbo_AR_Customer_Bi	dbo_SV_Service_Ticket	dbo_AR_Item
Total:	Group By	First	First	Expression	Sum	Where	Where	Where

Let's discuss these options. "Group By" is the default and basically it says create a record for each unique row. "First" means to not create a record for each unique row, but create a record only for the first row generated for that group. Notice that "First" can be either in the Total row or in the formula as it is in the CSZ column. When this appears in the formula, then "Expression" is used on the total line to show that the option is in the Formula. "Sum" means to create a sum of the values rather than a separate record or row for each value. This allows us to get a record with the sum of all of the invoices rather than a record for each invoice. Finally we have the "Where" option. This tells the query not to create a row based on these values but to use these values to see if it is included at all. Thus we can exclude invoices from other years, people with service contracts or people that did not receive an invoice. Once this is completed, save your query, and open Word.

Creating the Letter

I like to start my mail merges by creating my letter as if it were not a mail merge. I used a template downloaded from Microsoft as my base letter. Below is the basic letter after I modified the Microsoft template:

Sandbox Alarm Company
1234 Main St
Plymouth, MI 48170
January 8, 2011

[**Company Name**]
[**Street Address**]
[**City, ST ZIP Code**]

[**Recipient Name**]:

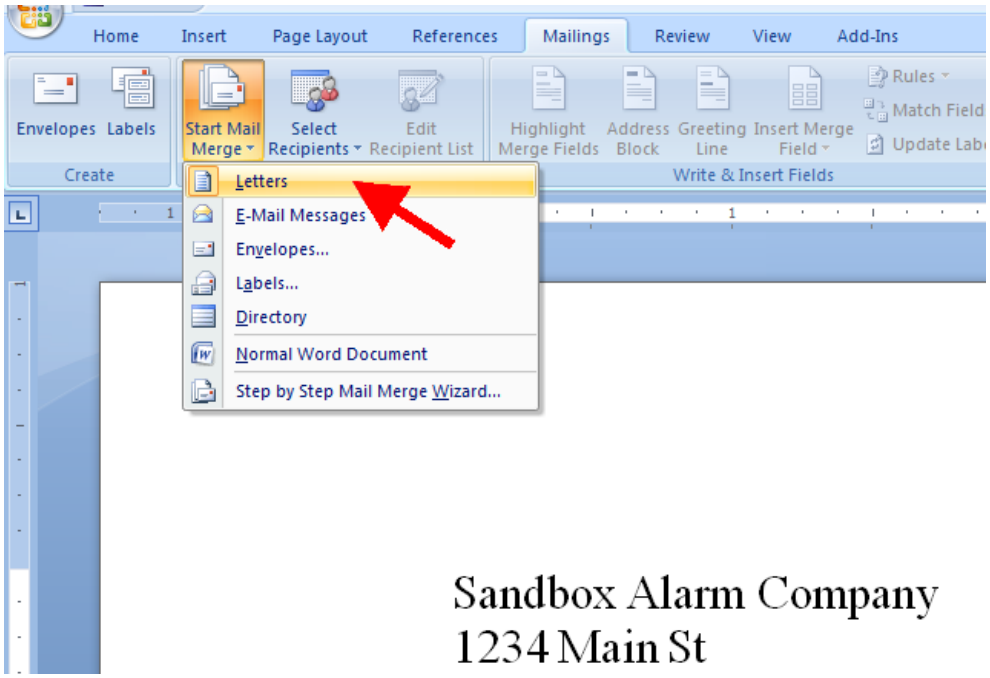
Our records show that last year you were invoiced $0.00 for service calls. You might like to know more about our service contract, which covers the cost of parts and labor for normal repairs. The actual cost varies, depending on your alarm system, but it is usually less than the time and materials costs of a service call without a service contract.

Mr. Jonathan Haas, the professional responsible for your alarm installation, will be happy to meet with you and give you a free estimate.

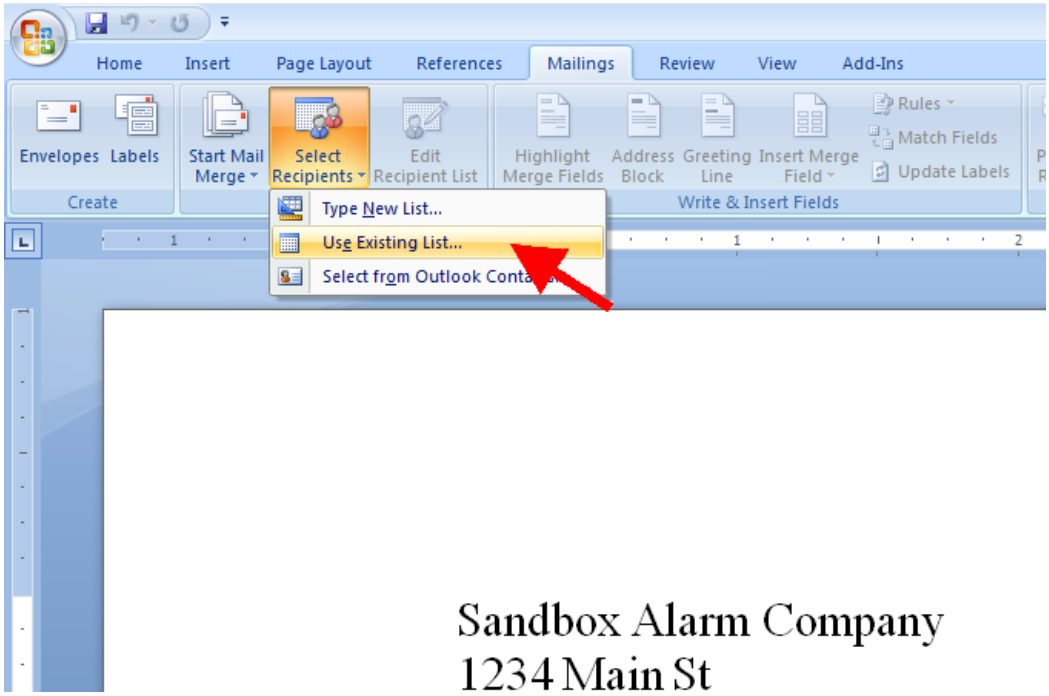
Sincerely,

Mathew S. Howe
General Manager

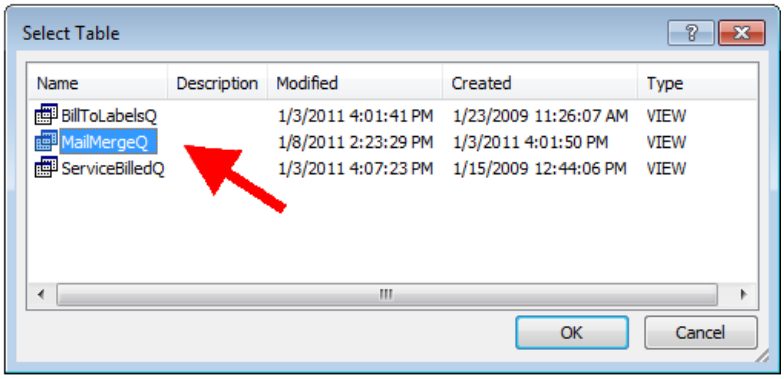
Now, go to the mailings tab and choose “Start Mail Merge” and “Letters”:



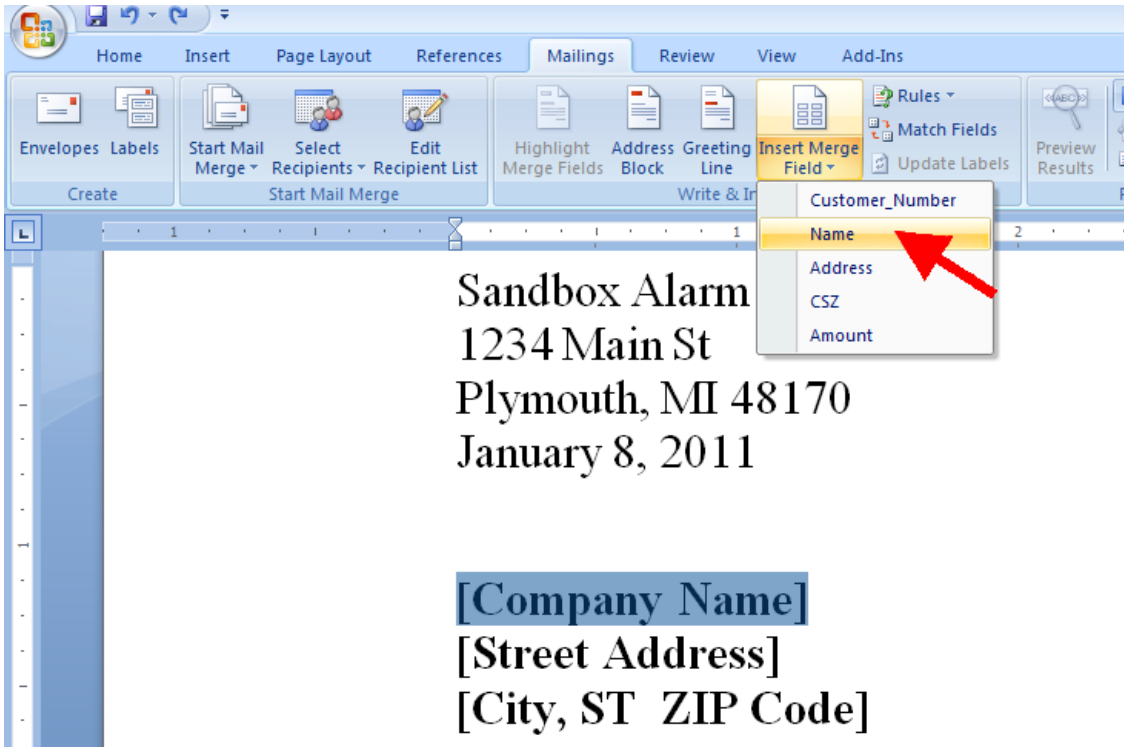
Next choose your recipients:



Locate and select your Access database and choose the MailMergeQ query:



Highlight the text you wish to replace with a field from your query, click on "Insert Merge Field" and choose the field you want:



The selected text will be replaced with the name of the field:

«Name»
[Street Address]
[City, ST ZIP Code]

[Recipient Name]:

Our records show that last year you

Insert the rest of the fields till your letter looks like this:

Sandbox Alarm Company
1234 Main St
Plymouth, MI 48170
January 8, 2011

«Name»
«Address»
«CSZ»

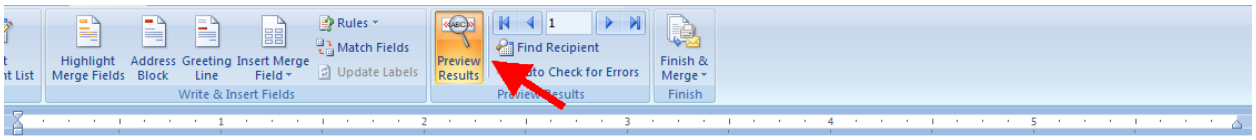
«Name»:

Our records show that last year you were invoiced $«Amount» for service calls. You might like to know more about our service contract, which covers the cost of parts and labor for normal repairs. The actual cost varies depending on your alarm system, but it is usually less than the time and materials costs of a service call without a service contract.

Mr. Jonathan Haas, the professional responsible for your alarm installation, will be happy to meet with you and give you a free estimate.

Sincerely,

Notice I replaced the 0.00 with the Amount field but left the dollar sign in place. Now we can preview our letter:



Sandbox Alarm Company
1234 Main St
Plymouth, MI 48170
January 8, 2011

A & L Renna Service
66 N Park St
E Orange NJ 07017

A & L Renna Service:

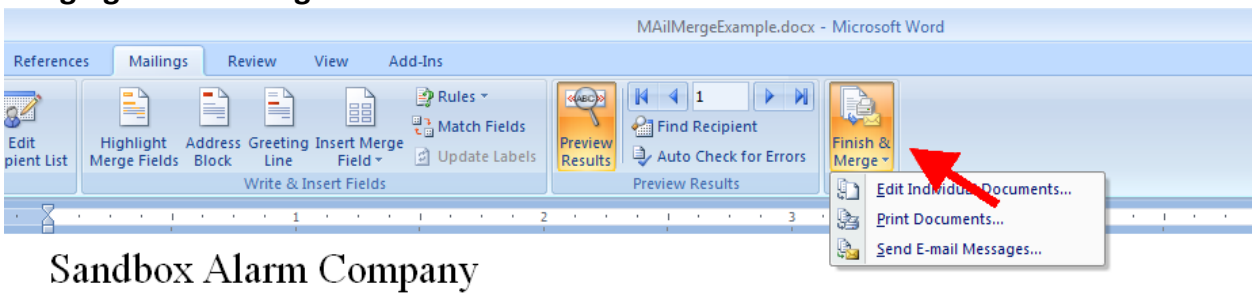
Our records show that last year you were invoiced $606.24 for service calls. You might like to know more about our service contract, which covers the cost of parts and labor for normal repairs. The actual cost varies depending on your alarm system, but it is usually less than the time and materials costs of a service call without a service contract.

Mr. Jonathan Haas, the professional responsible for your alarm installation, will be happy to meet with you and give you a free estimate.

Sincerely,

We are now ready to print our form letters. You have three choices. If you want to keep a copy of the letters on your computer choose the first choice. That will create a document containing all of the merged letters. Or you can just print them using the second option. Finally there is a function to Email the letters but we won't be covering that here.

Merging and Creating the Letters



Sandbox Alarm Company
1234 Main St
Plymouth, MI 48170
January 8, 2011

A & L Renna Service
66 N Park St
E Orange NJ 07017

In this section we have learned more about grouping in queries. We have also learned how to use our queries in mail merges to create form letters.

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 as 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. Calculated fields will have no name unless you assign one with the "As" keyword. An example of using the "As" keyword would be Quantity * Rate As Extended_Price. Notice the underscore in Extended_Price. Names must be one word unless

you place them in single quotes, 'Extended Price' The "As" keyword will also work with regular fields for example Business_Name As Name.

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

Left 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

Outer joins of any type are slower than inner joins. Replacing an inner join with a full outer join can change a query that runs in two minutes to one that takes 20 or 30 minutes to run. Only use outer joins when it is necessary.
Alias's can be used to shorten the join clause, for example:

```
From AR_Customer_System
Inner Join AR_Customer_System_Userdef On
AR_Customer_System.Customer_System_Id =
AR_Customer_System_Userdef.Customer_System_Id
```

Can be shortened to:

```
From AR_Customer_System s
Inner Join AR_Customer_System_Userdef u  On s.Customer_System_Id =
u.Customer_System_Id
```

Aliases are needed in order to refer to a table in more than one join.

```
From AR_Customer_Recurring r
Inner Join AR_Item i on r.Item_Id = i.Item_Id
Inner Join AR_Item m on r.Master_Item_Id = m.Item_Id
```

Allowing you to see both the recurring item and the master recurring item for a recurring record.

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 the 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 last concept to be covered here, logical operators.

Logical Operators

The most common logical operators are And and Or.

And

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

Or

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

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 And then Or but the order of precedence should not be relied on. Like the example, to be sure, use parentheses.