

# SedonaOffice®

The #1 Financial Software for Security Companies

---

## **Database Structure**

### **Training & Setup Guide**

**2013 SedonaOffice**

**Users Conference**

Last Updated: January 3, 2011

### 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](http://www.sedonaoffice.com) for the most current version.

Copyright 2011

---

### Table of Contents

About this Guide.....	2
Overview .....	5
Each Company is a Database .....	5
Databases Contain Tables, Views, and Stored Procedures.....	5
Tables Contain Fields .....	5
Linking Tables .....	6
Link Types .....	6
Customer Structure .....	7
Invoice Structure .....	8
Cash Structure .....	9
Vendor Structure.....	10
Vendor Bills Structure.....	11
Check Structure .....	12
Job Structure .....	13
Service Ticket Structure.....	14
Inventory structure .....	15
General Ledger Structure .....	16
Open Database Connectivity (ODBC) .....	16
Creating an ODBC Connection with Excel.....	16
Building a Query Using Excel and MS Query .....	19
Using Microsoft Access to Review Your Data .....	22
Connecting Access via ODBC .....	22
Writing a query with Access .....	26

Creating a Report with Access .....	31
Creating a Grouped and Sub Totaled Report.....	35
<b>Using SedonaOffice Data in a Microsoft Mail Merge .....</b>	<b>43</b>
Creating the List of Customers .....	44
Creating the Letter.....	47
Merging and Creating the Letters.....	52
<b>Basic SQL Language .....</b>	<b>52</b>
Select Keyword .....	52
From Keyword .....	53
Join Keyword .....	53
Inner Join .....	53
Left Outer Join .....	53
Right Outer Join .....	54
Full Outer Join.....	54
Where Keyword.....	54
Logical Operators.....	55
Other Where Clause Filters.....	57
Order By Keywords .....	59
Union Keyword .....	59

### 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 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<sup>1</sup>.

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<sup>2</sup>.

### 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.

---

<sup>1</sup> Unless directed to by a SedonaOffice support person.

<sup>2</sup> 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.

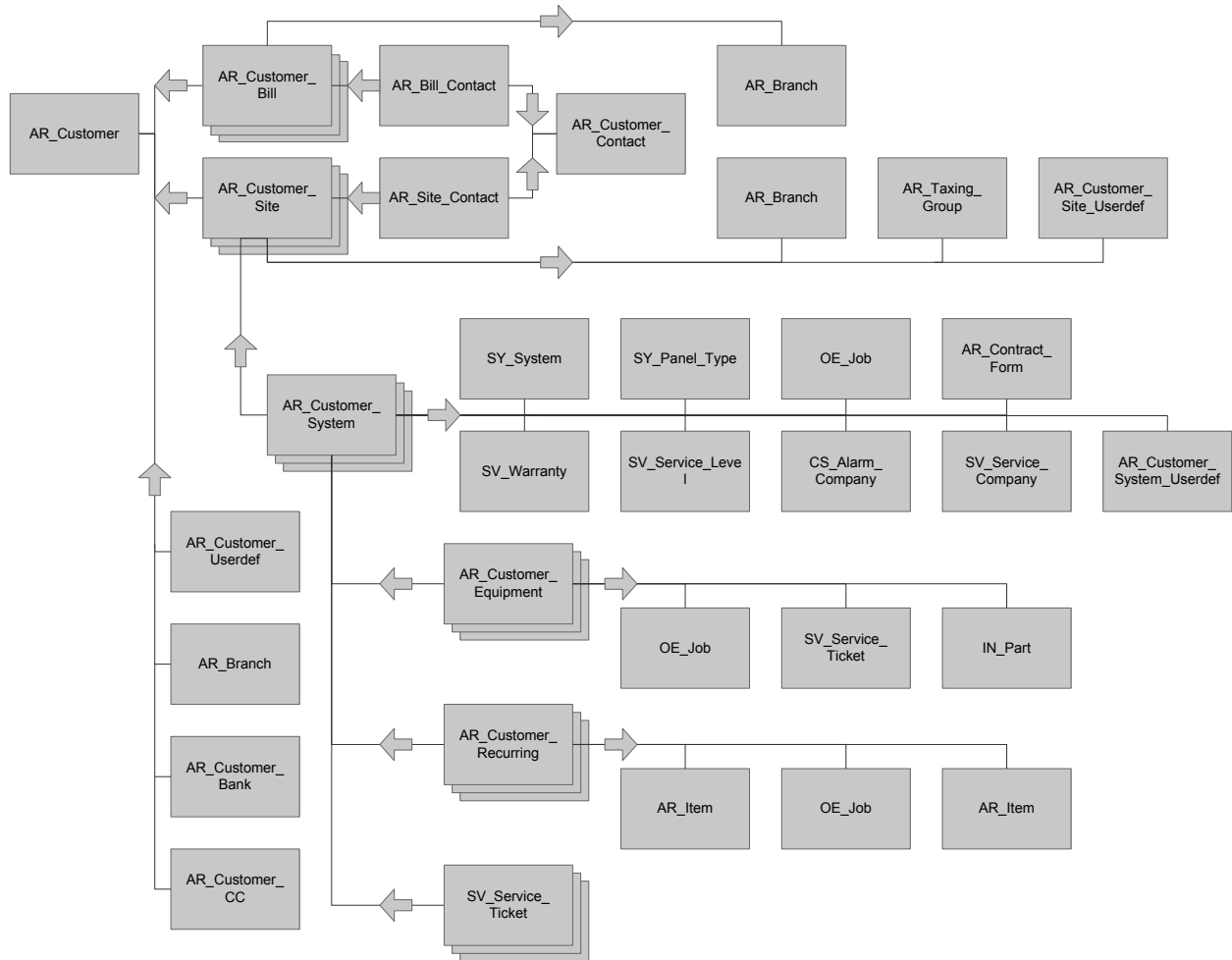
One to one: Each record in one table matches to exactly one record in the other table. IE AR\_Customer and AR\_Customer\_Userdef

One to many: Each record in one table matches to many records in the other table. IE AR\_Invoice and AR\_Invoice\_Item

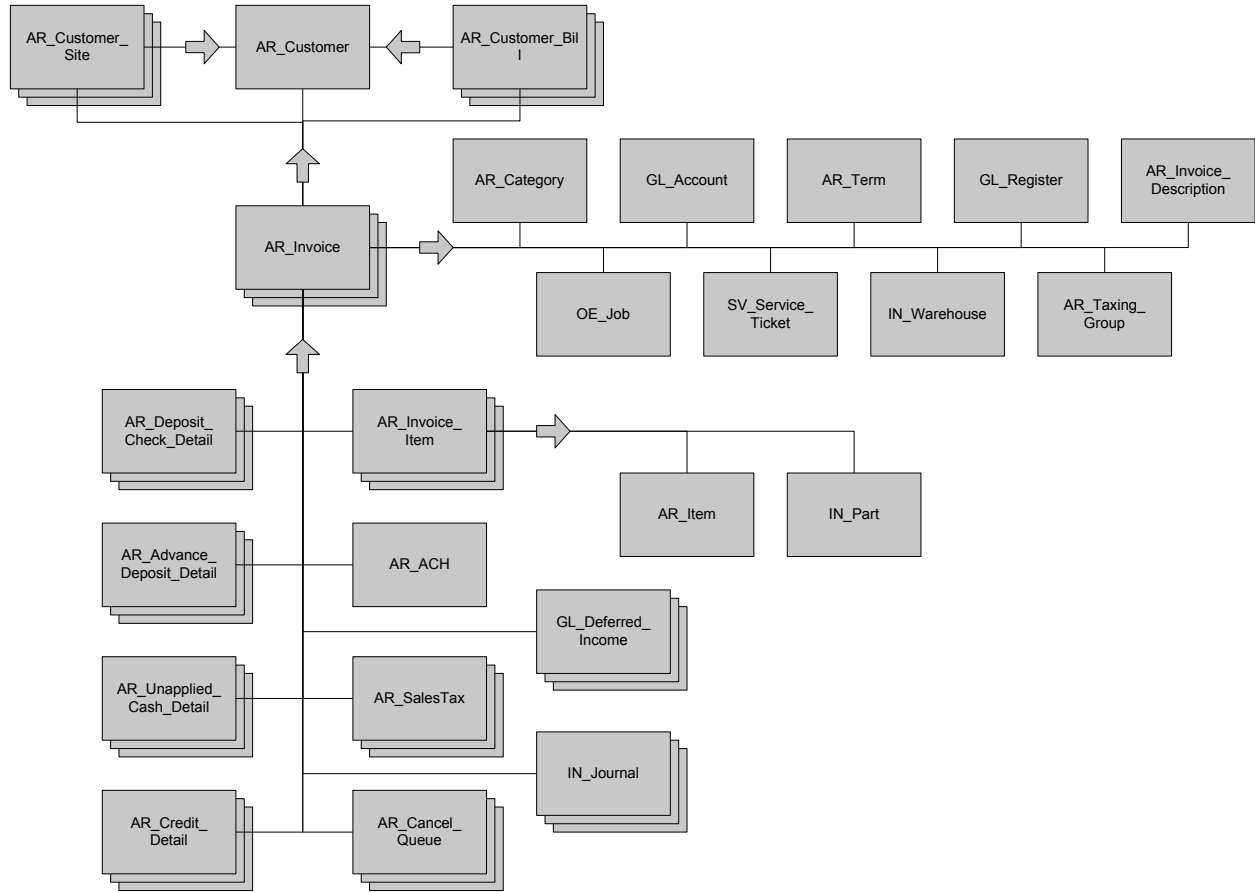
Many to one: Many records in one table match to one record in the other table. IE AR\_Customer and AR\_Branch

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

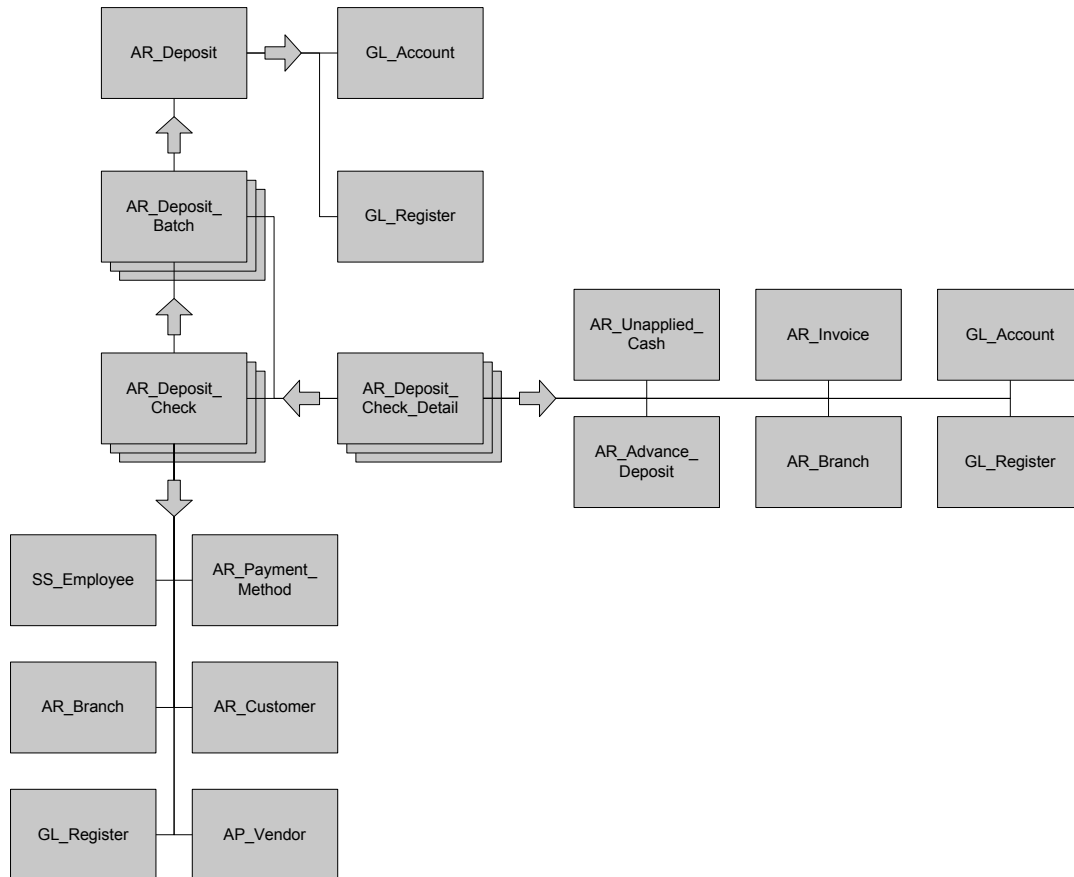


### Invoice Structure

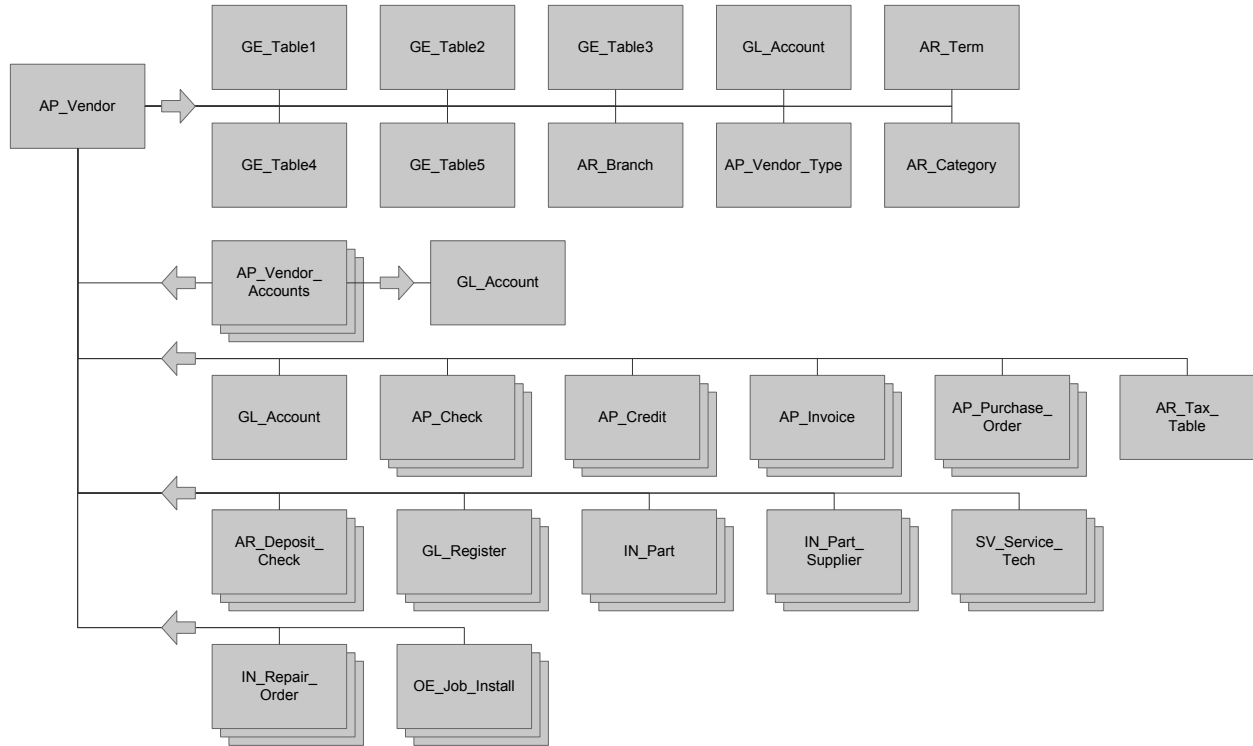




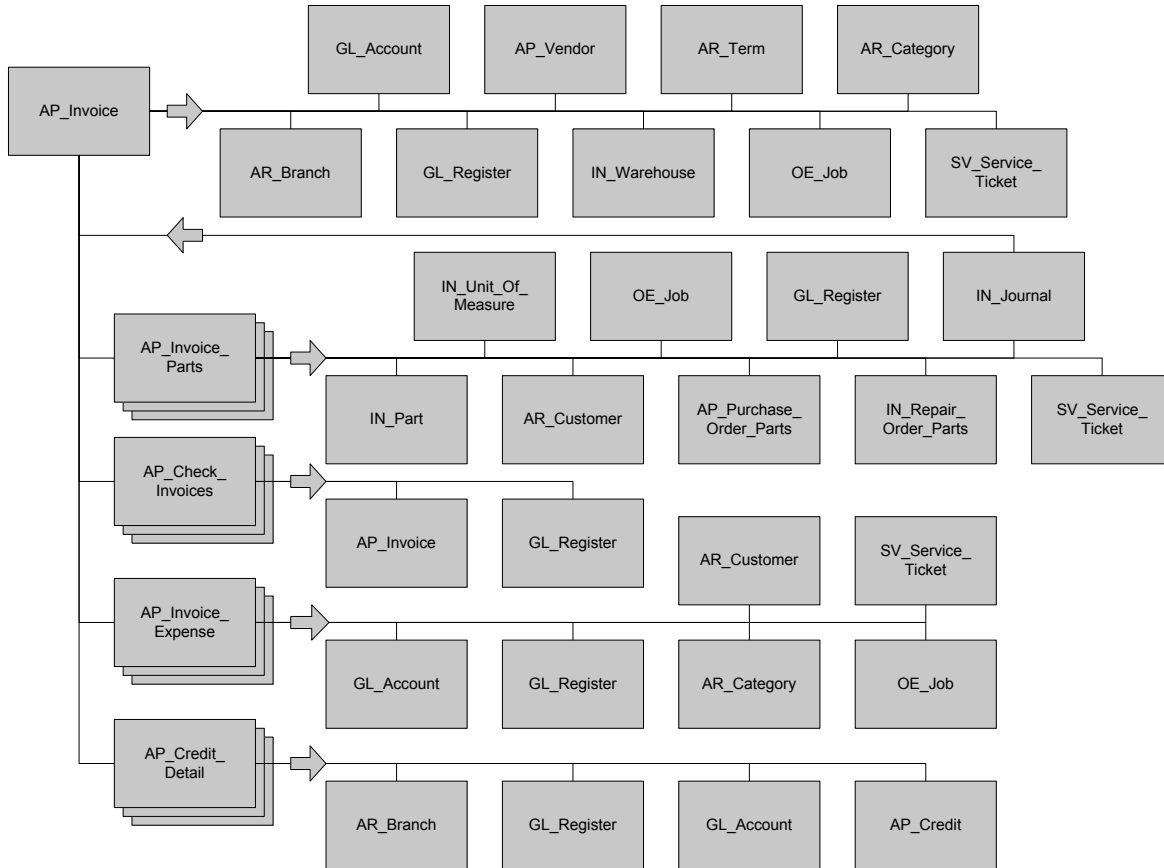
### Cash Structure



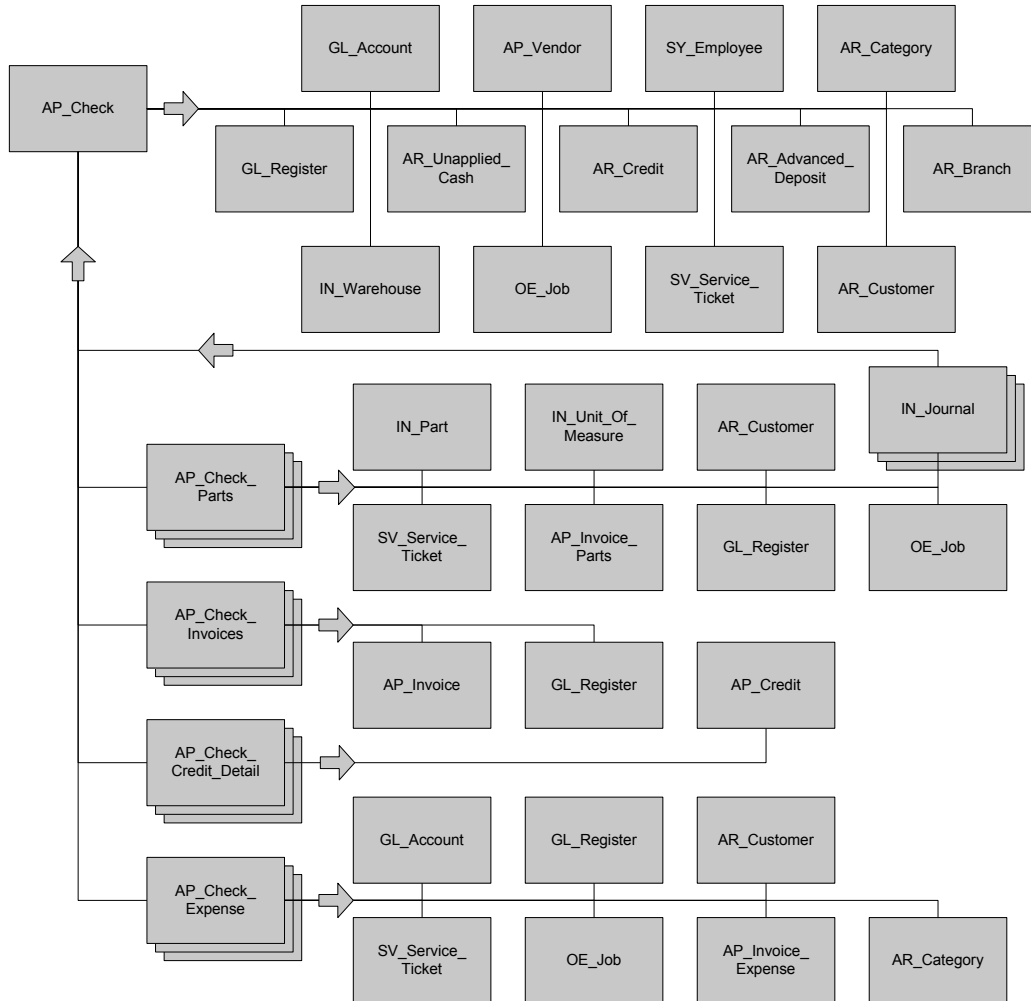
### Vendor Structure



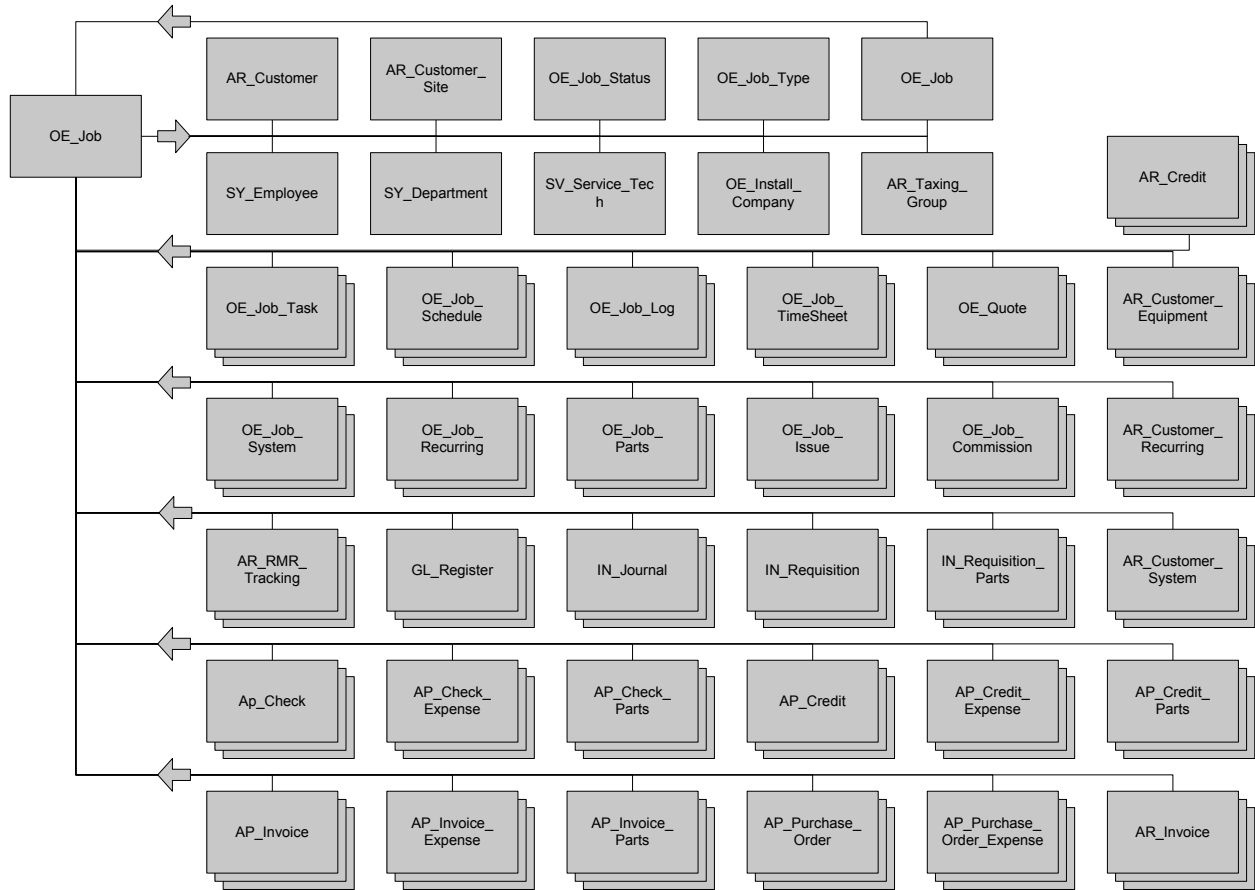
### Vendor Bills Structure



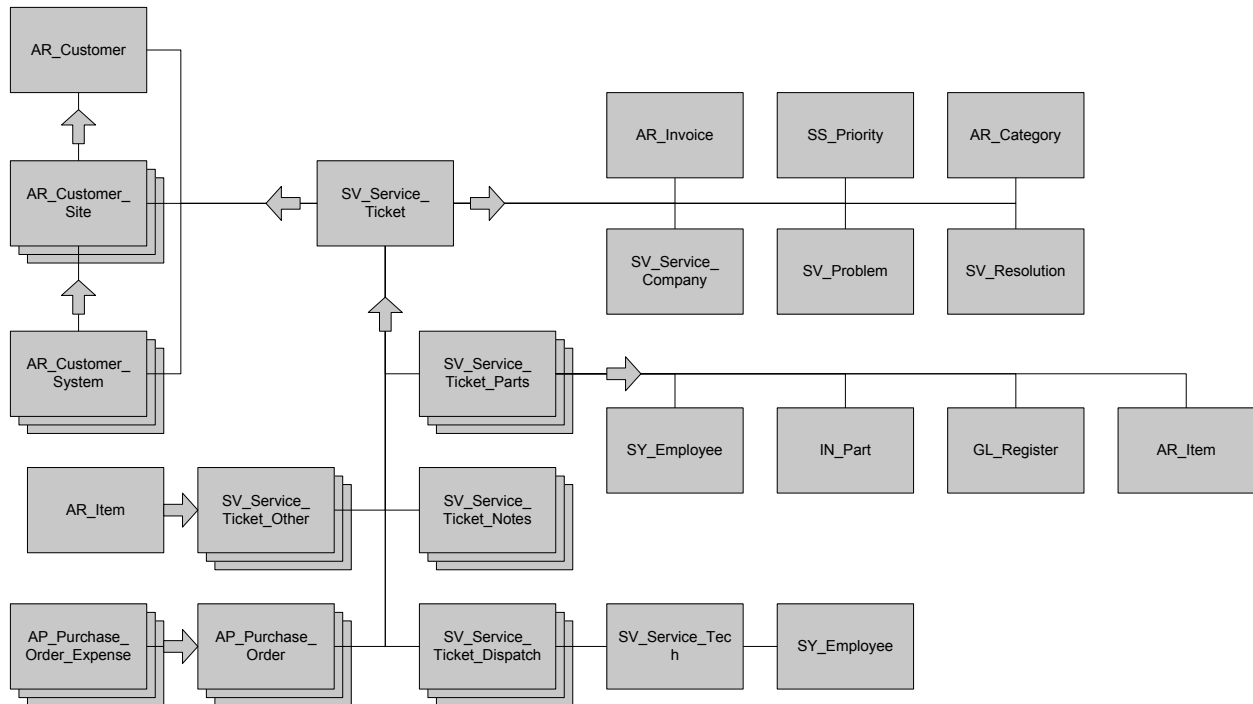
### Check Structure



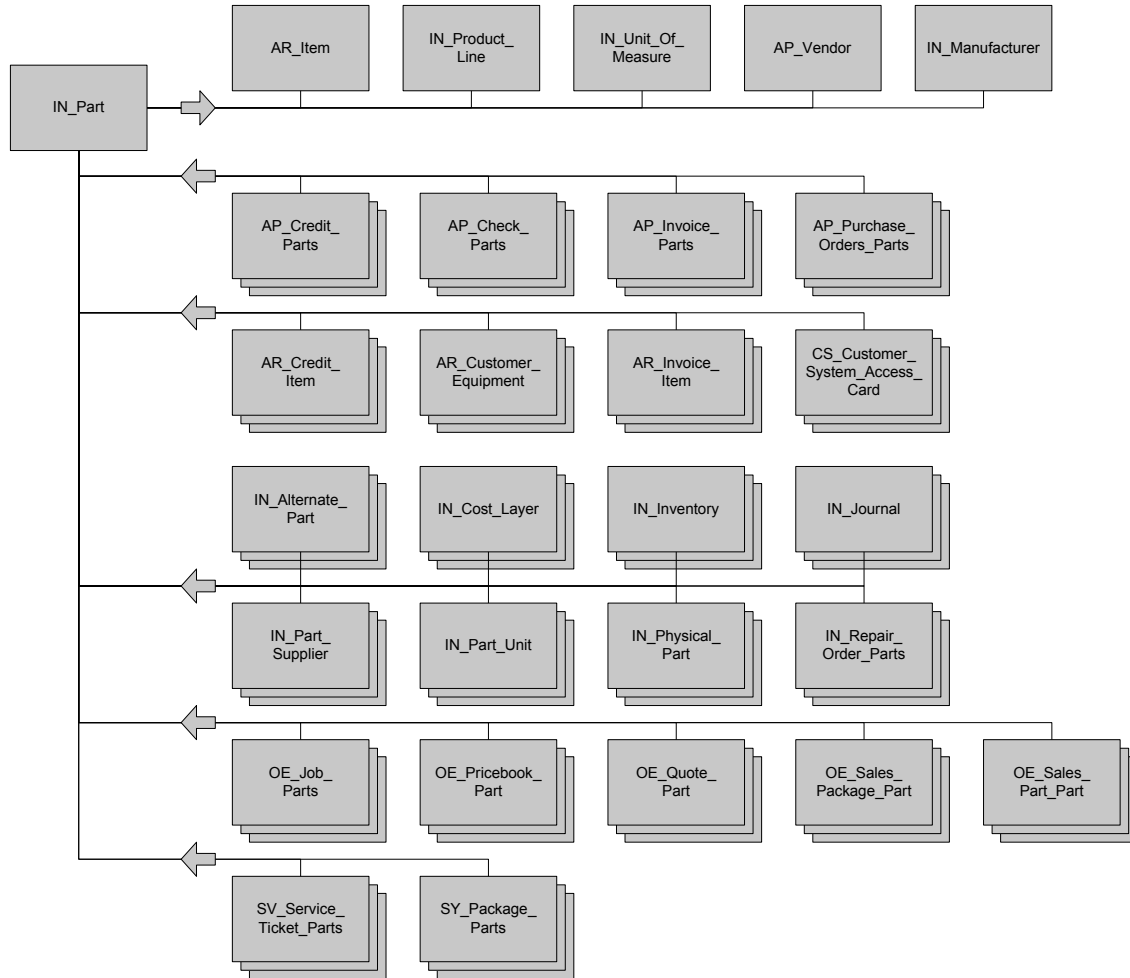
### Job Structure



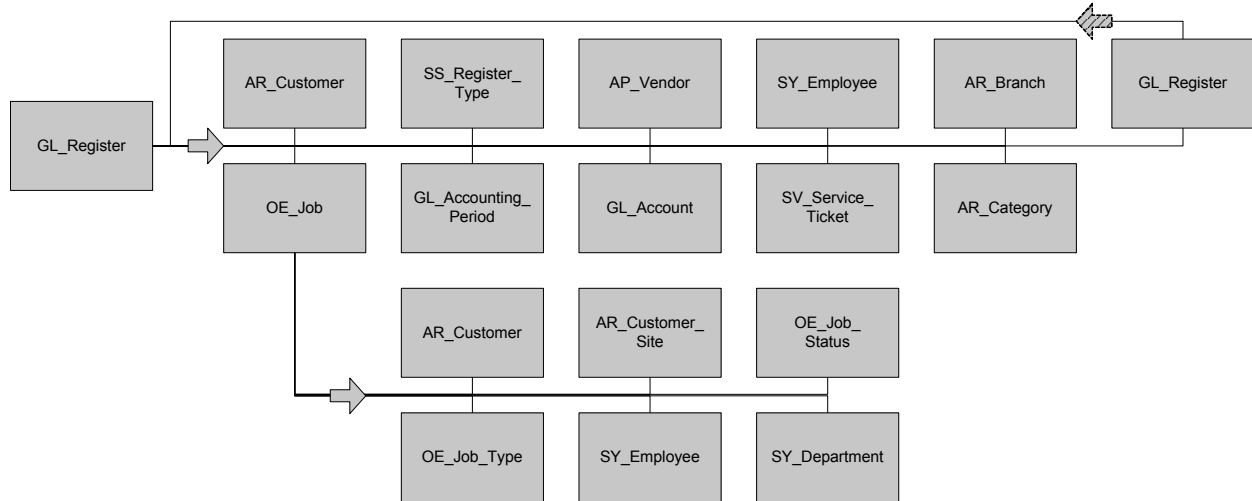
### Service Ticket Structure



### Inventory structure



### General Ledger Structure



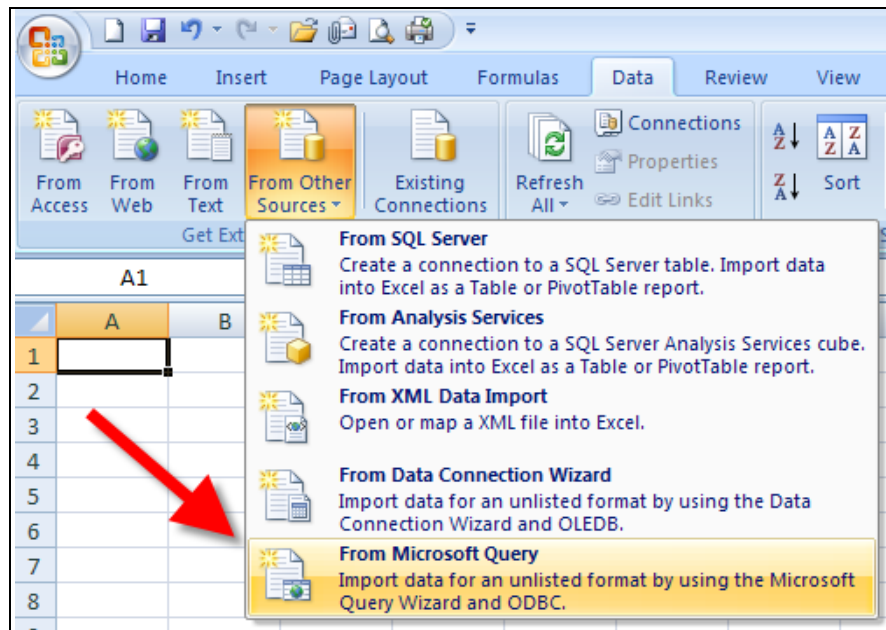
### Open Database Connectivity (ODBC)

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.

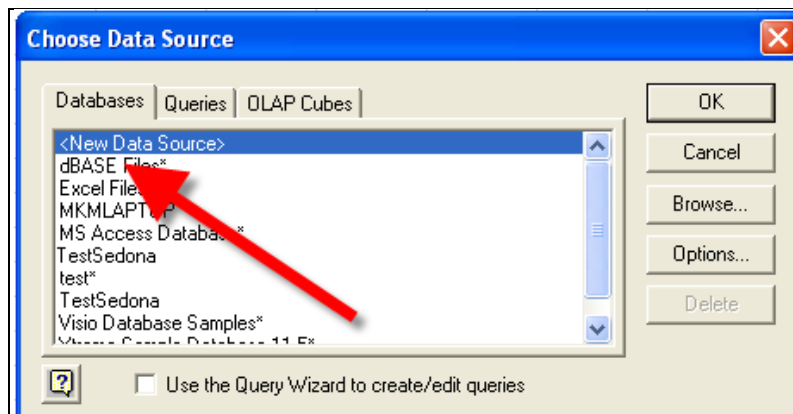
### Creating an ODBC Connection with Excel

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 need to be installed as Excel does not install it by default in the standard install.



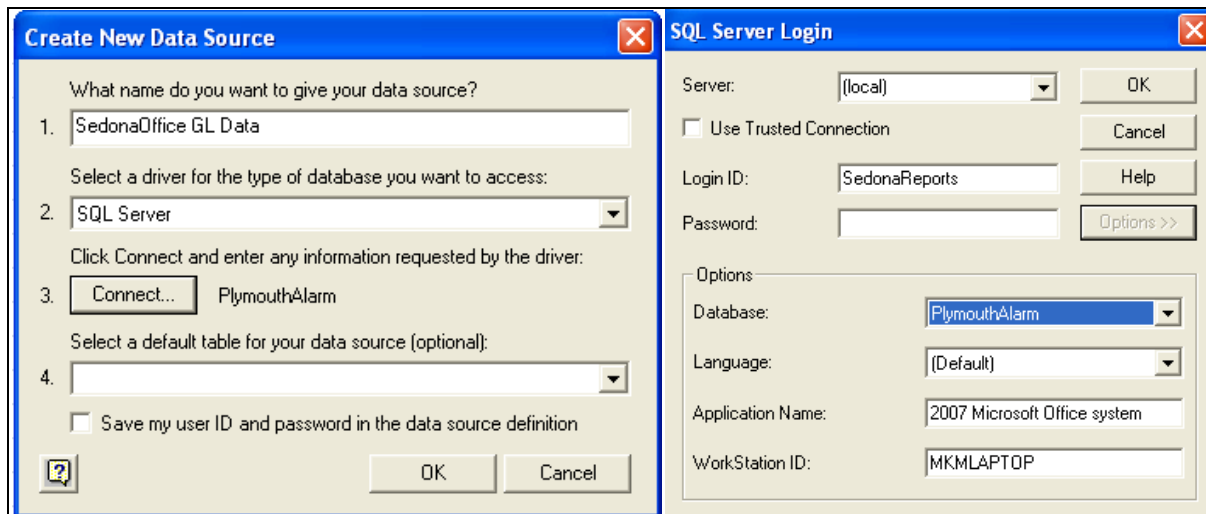


If you have not already done so, you will need to create a Data Source connection to your SedonaOffice database.



To create the new Data Source:

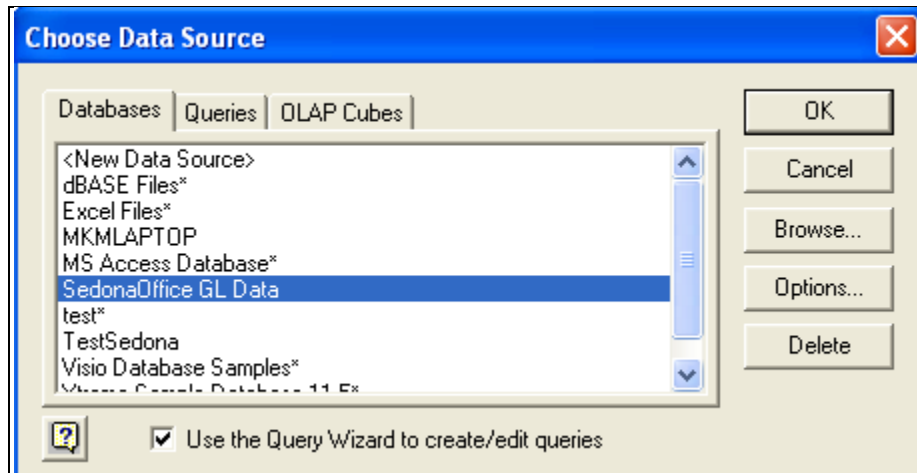
- 1) Name the data source appropriately (Here we are using "SedonaOffice GL Data" but the same connection can be used for all of your queries so you might want a more general name.)
- 2) Select 'SQL Server' as the driver to connect to the database
- 3) Press the Connect button
  - a. On the SQL Server 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



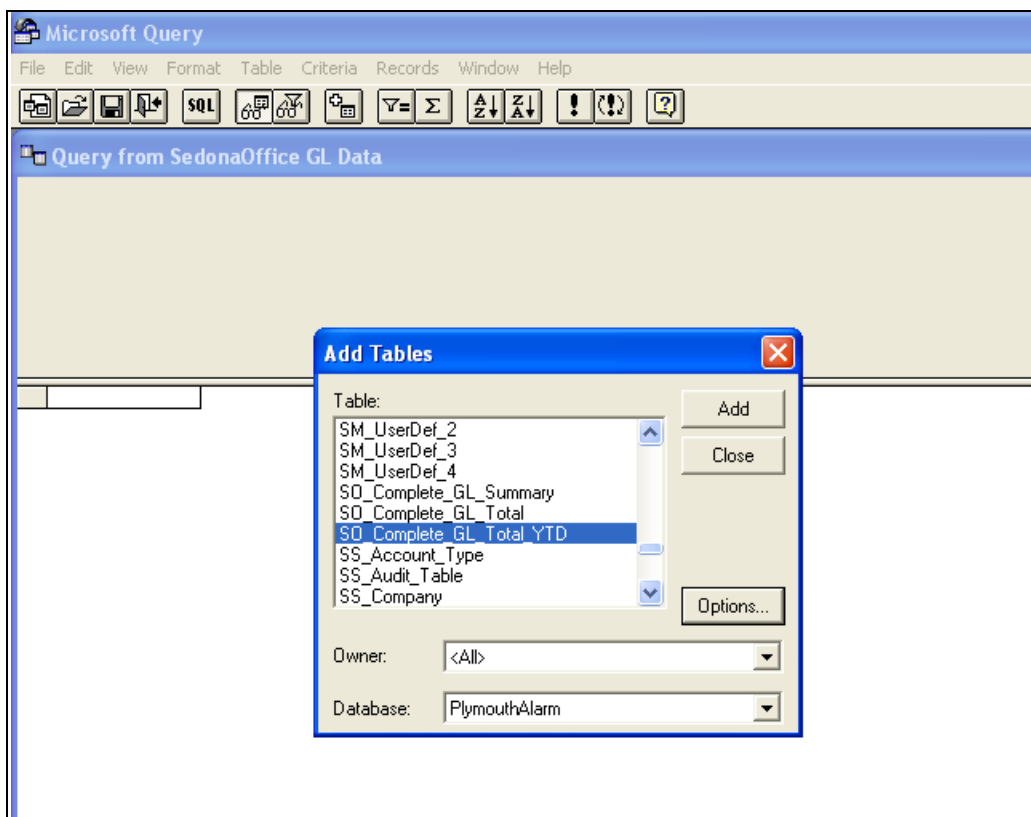
You now have an ODBC connection to your database.

### Building a Query Using Excel and MS Query

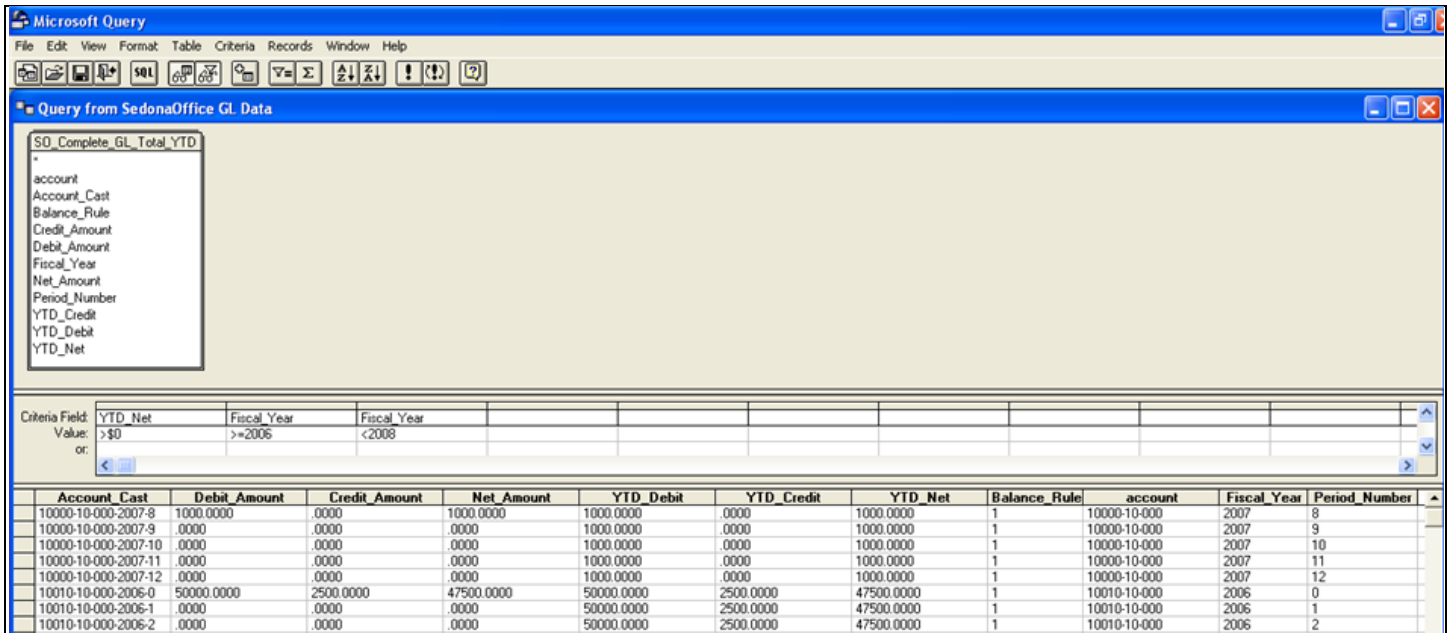
Select the data source you have previously created to create the Query. Uncheck the 'Use the Query Wizard...' this will take you directly to Microsoft Query to create the Query.



To begin with 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** **which** order **you choose** to display the data fields, using the order as shown below will be more logical when viewed with Excel.



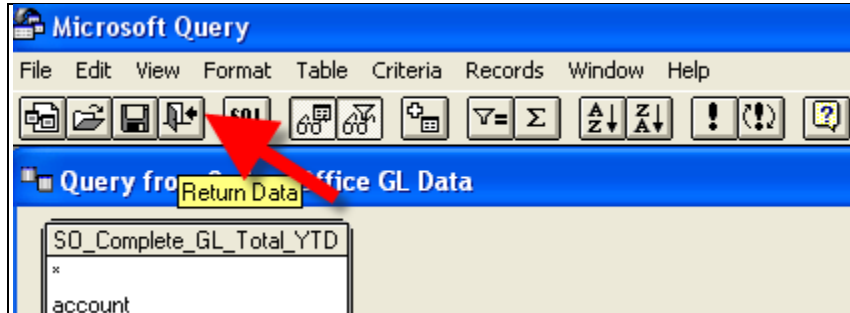
Since this table can contain thousands (hundreds of thousands of records) it is best to use some criteria to limit the data that returns.

### 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.**

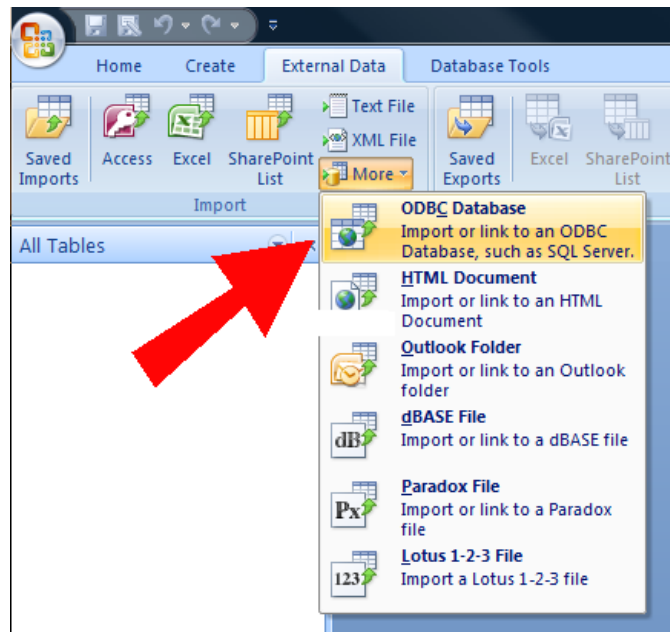
### Connecting Access via ODBC

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

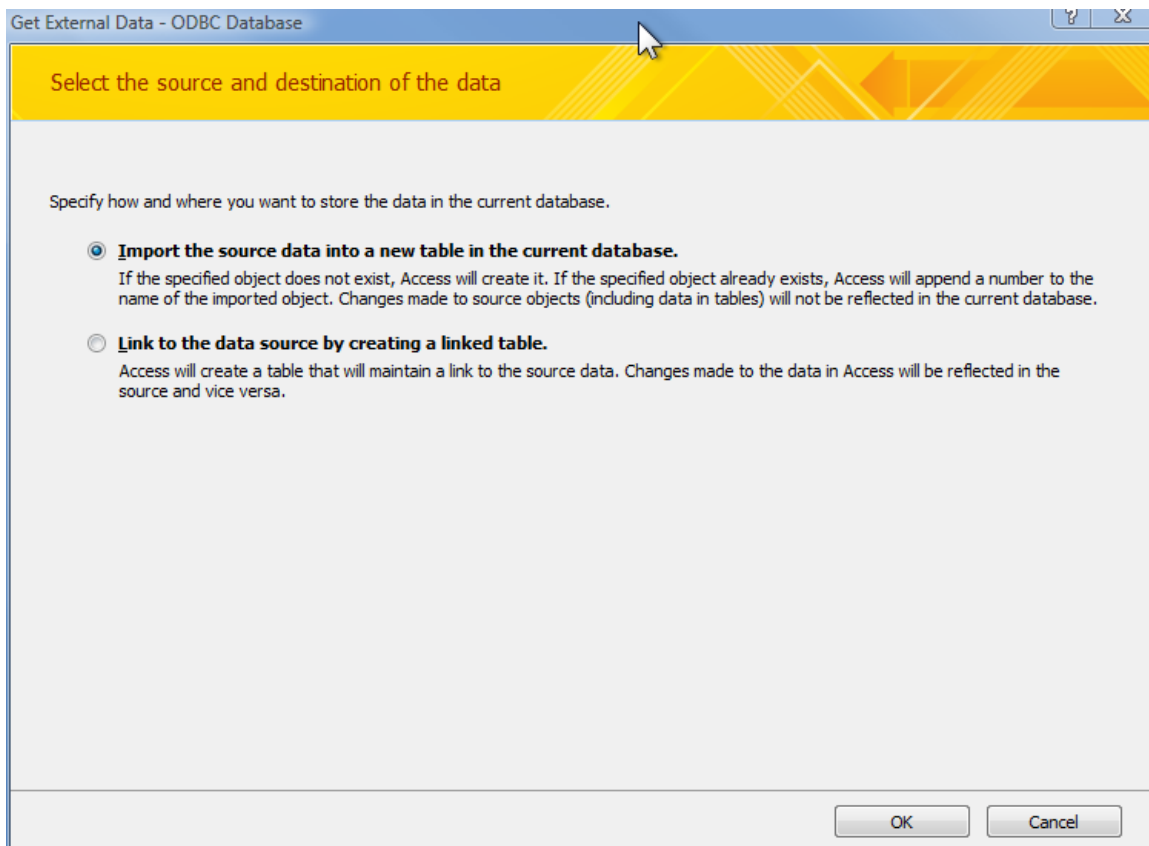
When you Import data into Access, you create a copy of the data stored within the Access database. This allows you to review the data when not connected to the database. Like Excel, you have to periodically refresh the data to keep it up to date.

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.

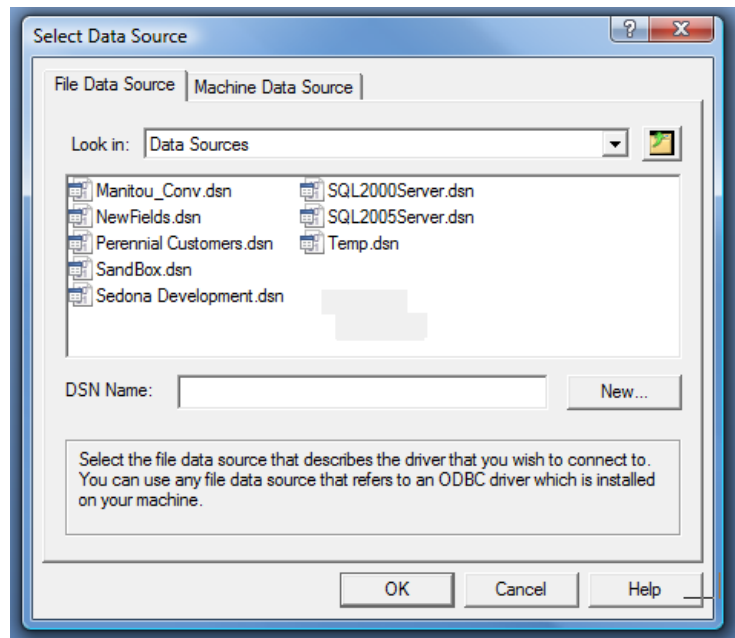
Choose the External Data tab. Then choose More. Finally choose ODBC Database.



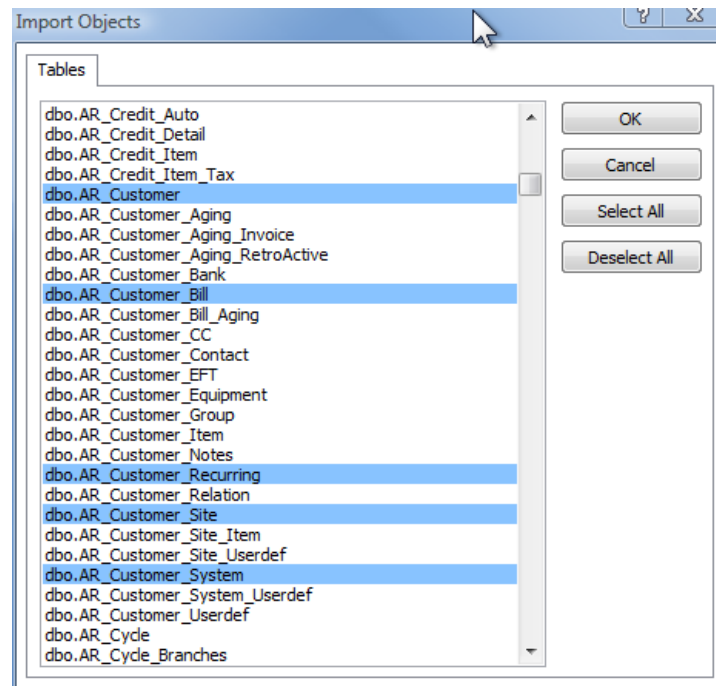
Choose Import or Link and then click OK.



Choose your Data Source.

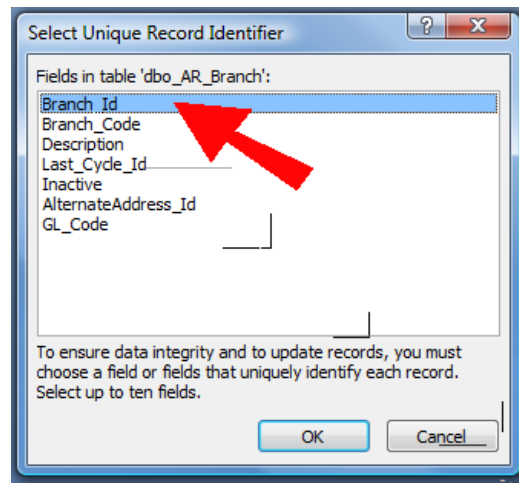


Then choose the tables you wish to Import or Link and click OK. You can choose multiple tables but do not select all. Access is not as large or as powerful as SQL Server. Choosing all will probably crash Access.

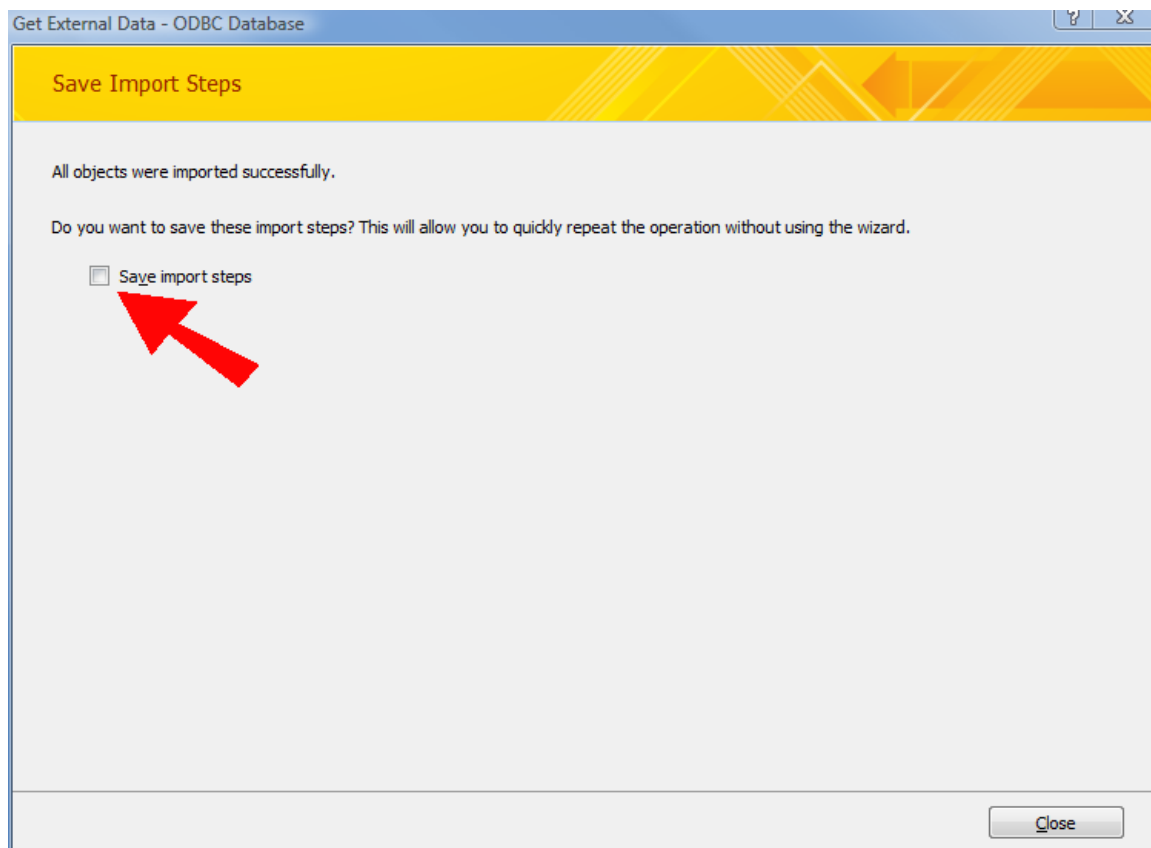


If you chose to Link, you will be asked to Select Unique Record Indicator. This is always the top item in SedonaOffice.

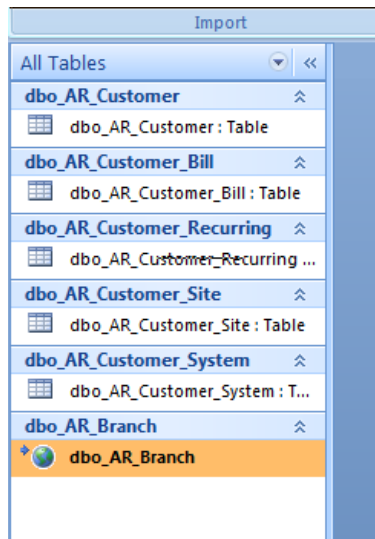




If you chose Import, when the operation is complete a window will be displayed showing the success of the operation. Here you can also Save the steps you just did so that refreshing the data will be easier.

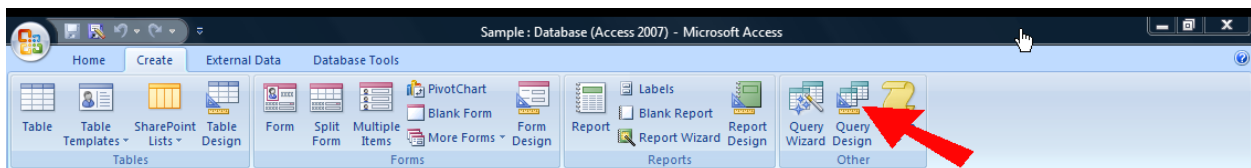


Your tables will then be accessible in Access. You may mix Import and Link in the same Access database. In the example I have Imported several customer tables and linked the branch table. Notice the different icons for imported versus linked tables. The highlighted table is the linked branch table.

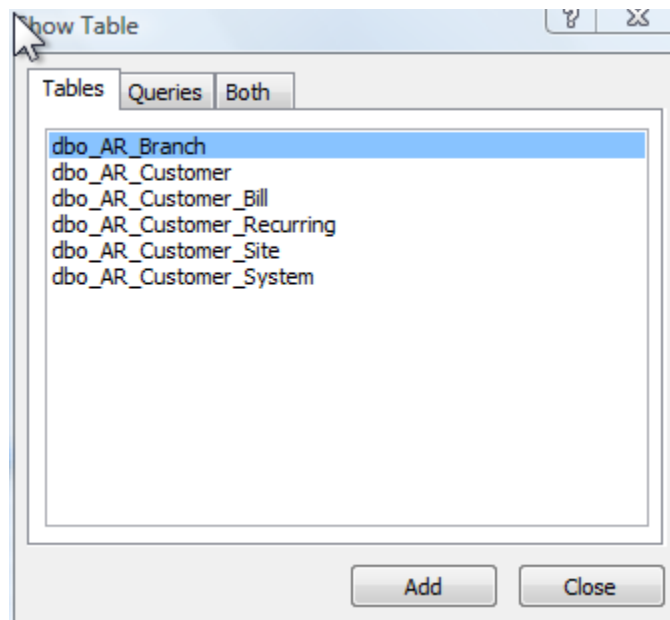


### Writing a query with Access

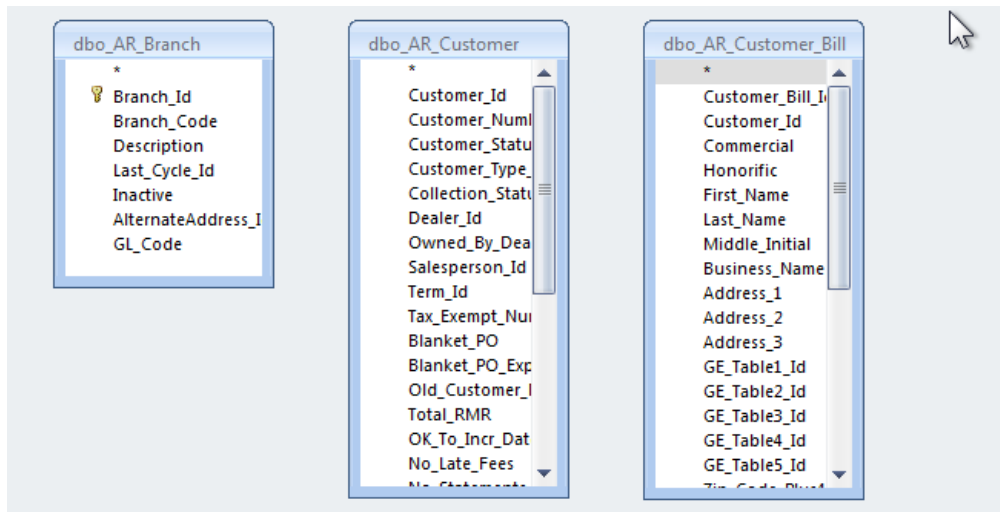
Click on the Create tab and then on Query design.



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.

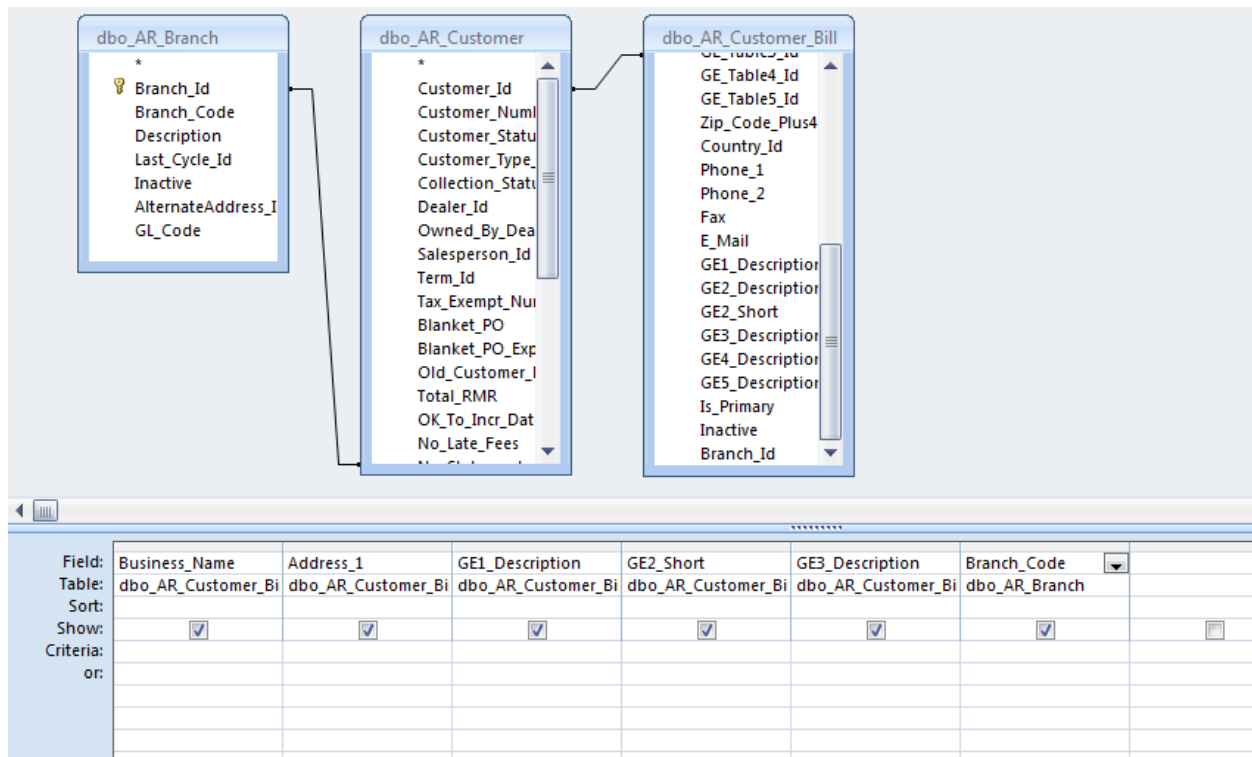


Delete all of the joins that access automatically creates.



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.

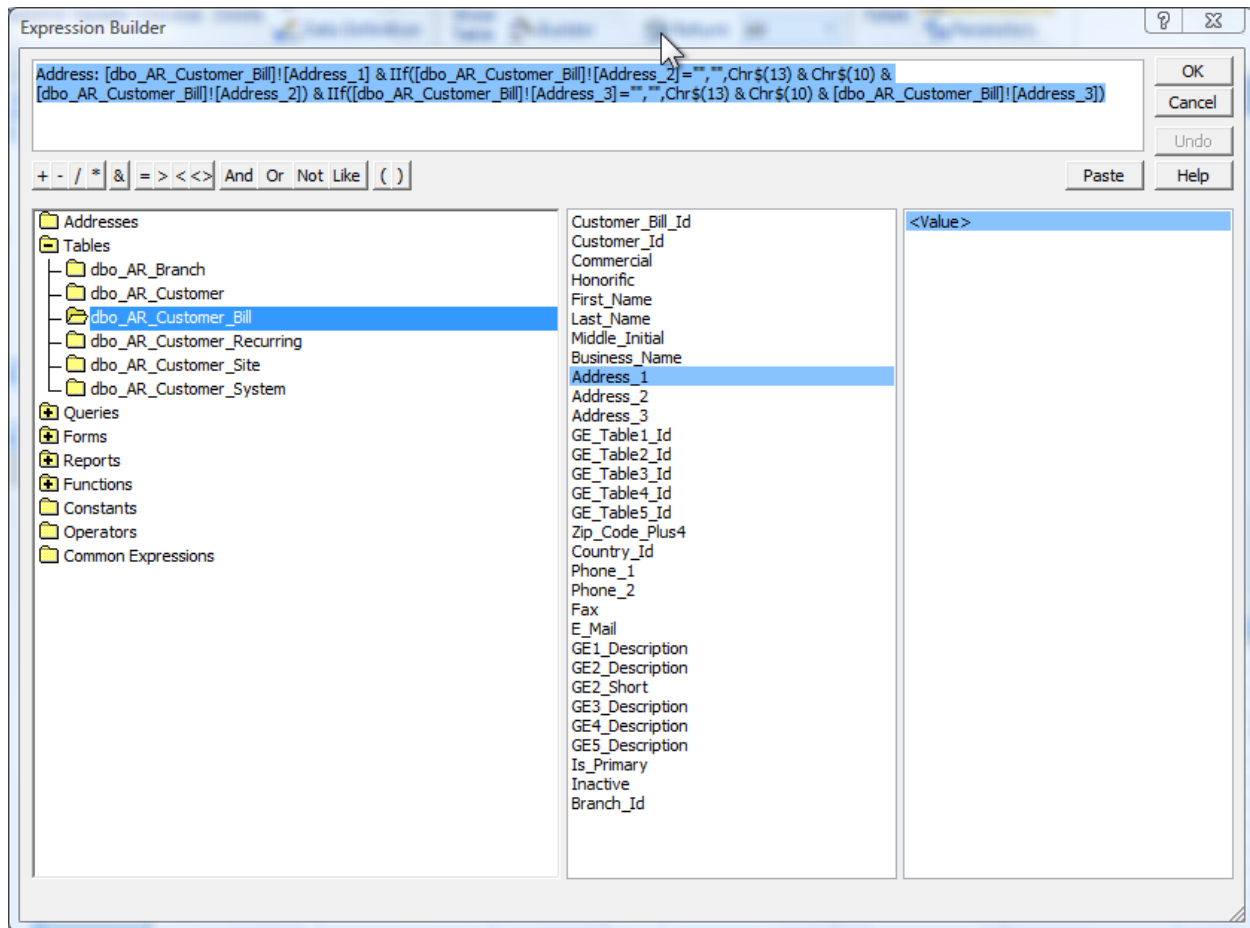
We are going to create a mailing list so we need to drag the name and address information to the lower pane. We are also going to drag down the branch code so we can sort on branch.



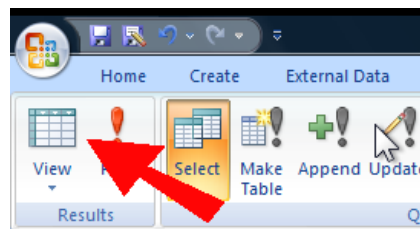
Address\_1 may not be all of the address information needed but if there is no address\_2 we don't want to add a blank line. So we create a formula. Click in the Address\_1 cell and then click On the formula button.

Enter the following into the builder window.

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



Click view to test our results.



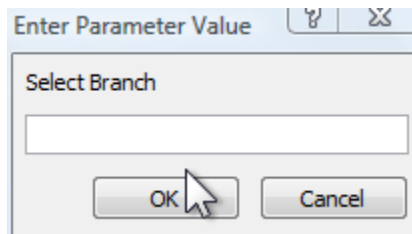
Business_Name	Address	GE1_Description	GE2_Short	GE3_Description	Branch_Code
N/A		N/A	N/A	N/A	Main Division
George Washington	1234 Mount Vernon Lane	Colorado Springs	CO	80919	Major
John Adams	5411 2nd Street	Colorado Springs	CO	80919	Major
Mega Mart	2154 Mountain Springs Road	Colorado Springs	CO	80919	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 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
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
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<> "N/A"					=[Select Branch]

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



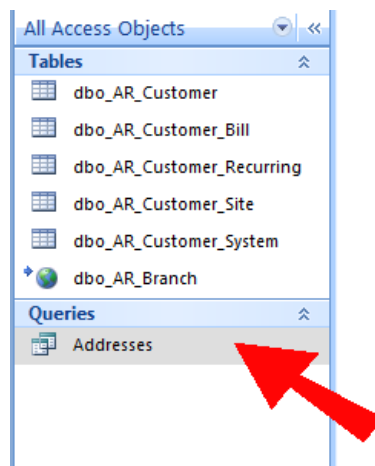
Entering a branch we get results with no N/A.

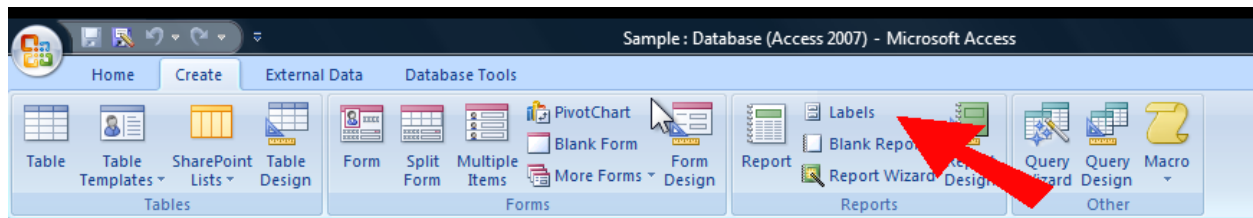
Business_Name	Address	GE1_Description	GE2_Short	GE3_Descrip	Branch_Cod
TELUS	1234 Main Street	Colorado Springs	CO	80919	Major
Dealer X	1234 Main Street	Colorado Springs	CO	80919	Major
George Washington	1234 Mount Vernon Lane	Colorado Springs	CO	80919	Major
John Adams	5411 2nd Street	Colorado Springs	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

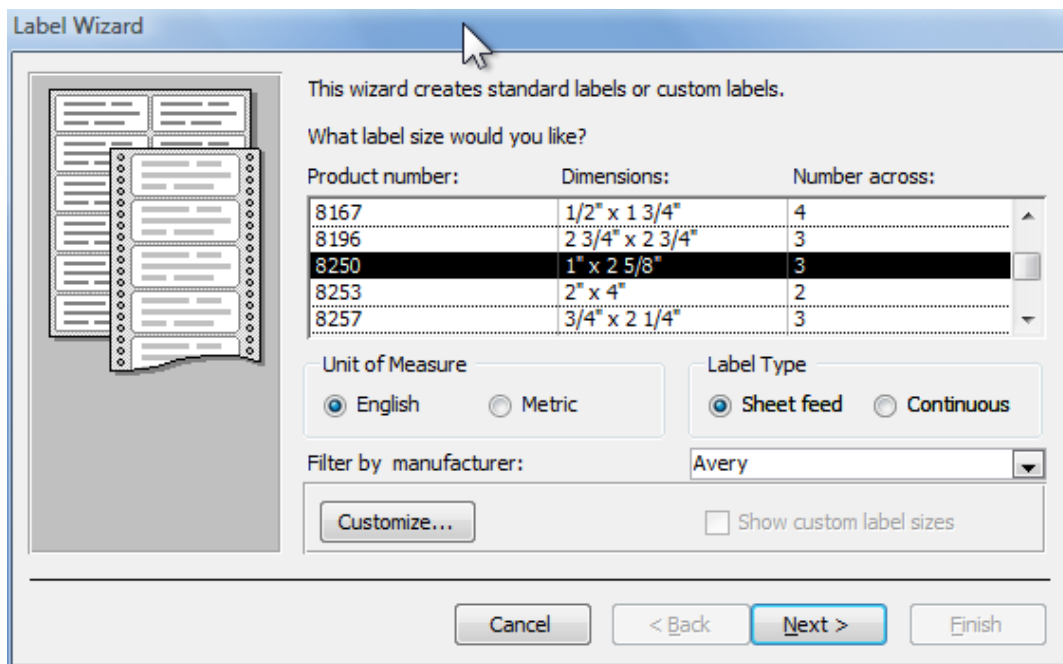
Displaying the results on the screen is useful but Access allows us to create reports. The report we are going to create will be to create mailing labels.

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



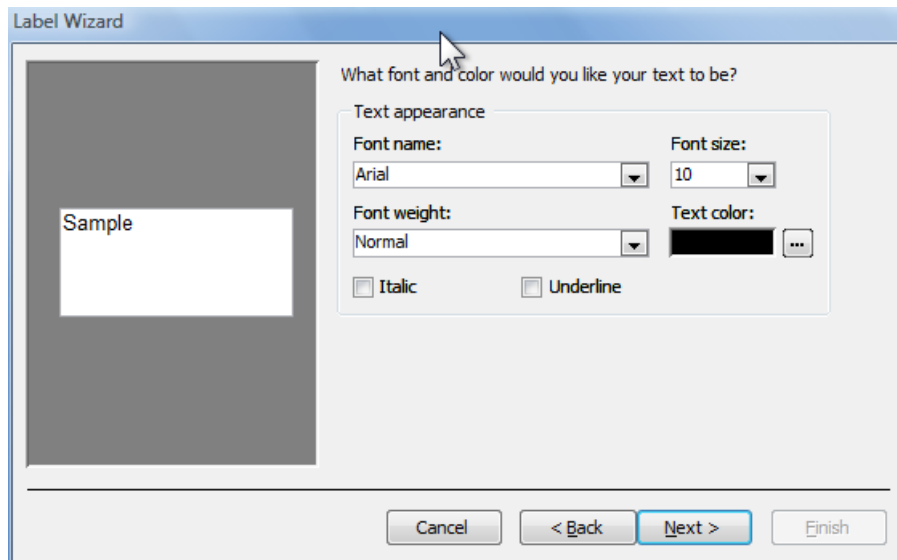


Choose your label. You can choose by the form number if you bought labels from a major manufacturer or just choose a label of the same size as the ones you are using.

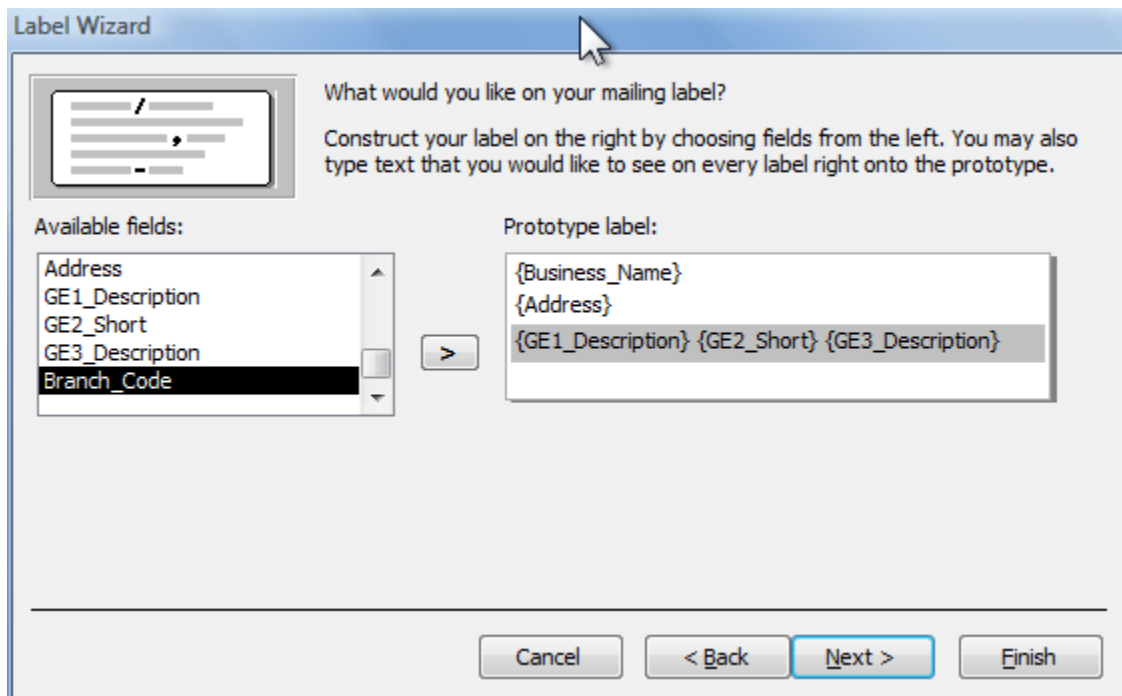


Choose your font.

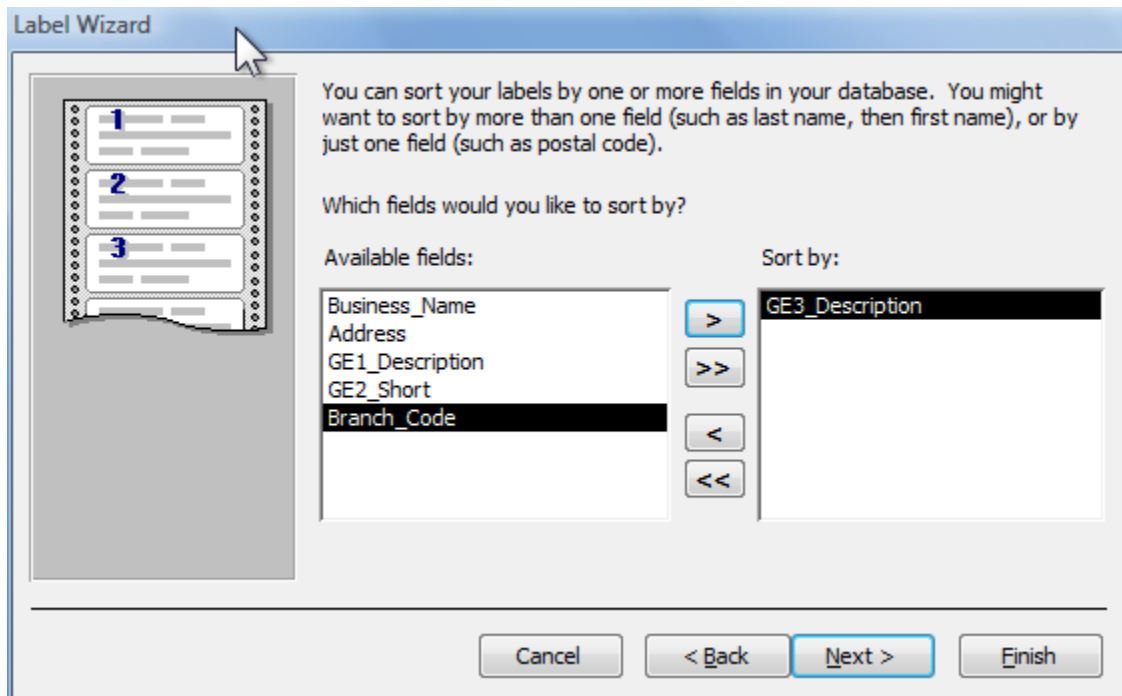




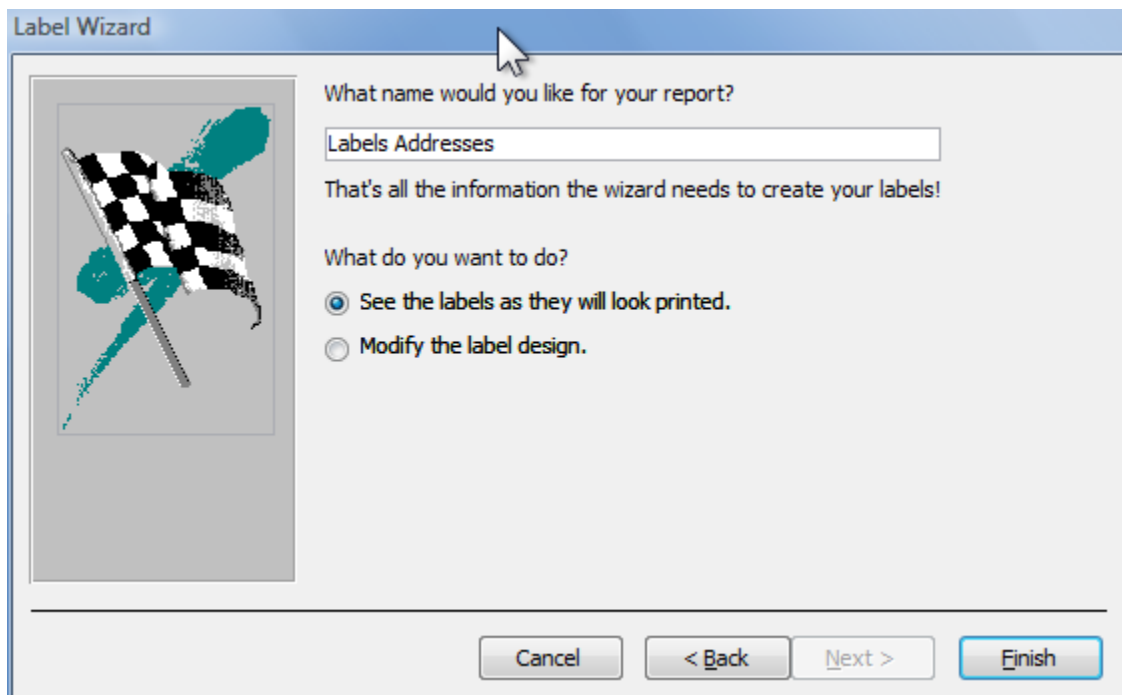
Setup the fields how you want them to appear on the label.



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.

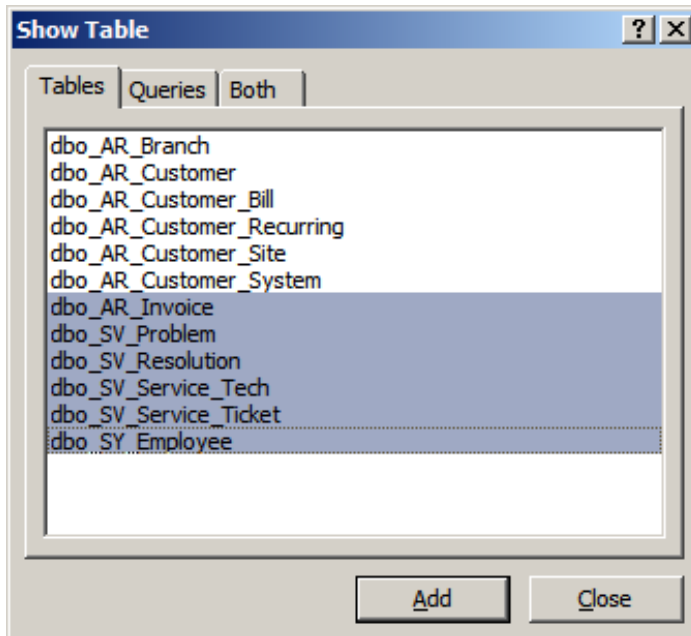
Win-Pak 421 Windchime Pl Colorado Springs CO 80919	Rocky Mountain High School 421 Falcon Way Colorado Springs CO 80919	Andrew Marriott 123 Main Street Colorado Springs CO 80919
Clint Eastwood 12445 Happy Acres Drive Colorado Springs CO 80919	Gene Autry 7466 Carter Valley Road Colorado Springs CO 80919	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

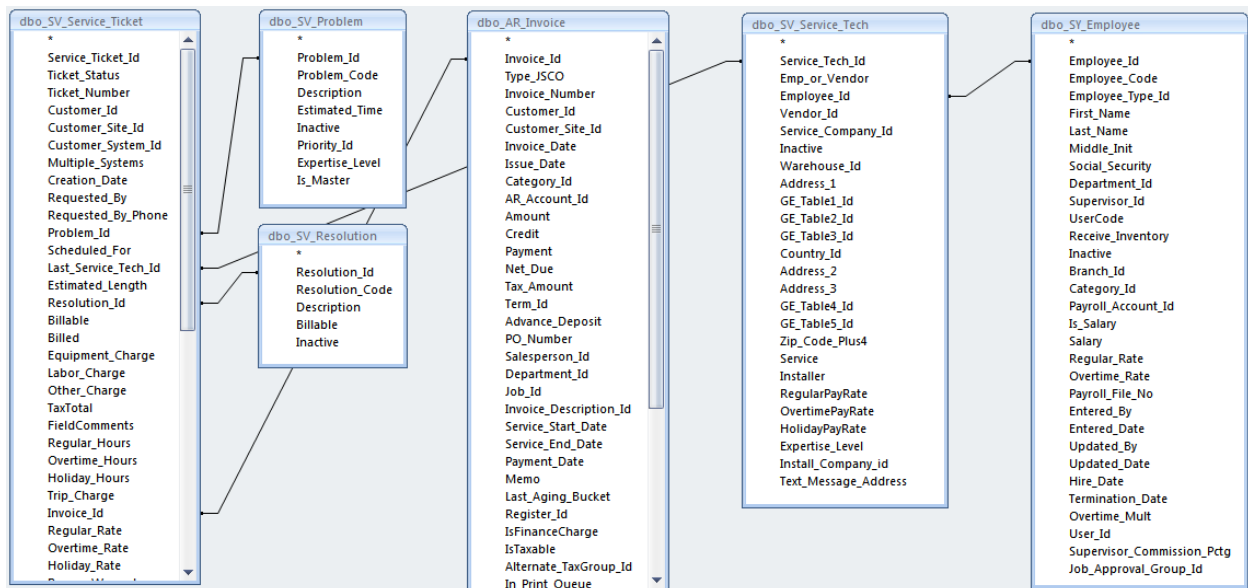
First we will need some additional data. Again select the ODBC database import item. Add these additional tables:

- 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.



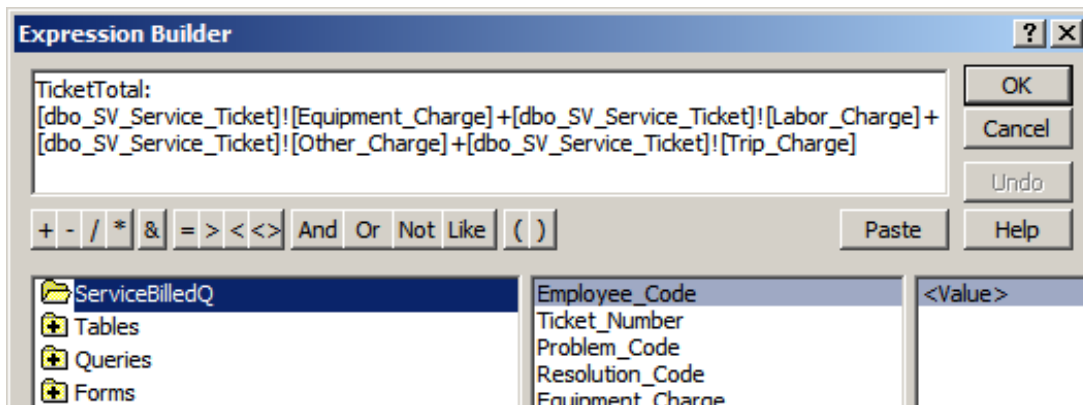
Add these fields.

- 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

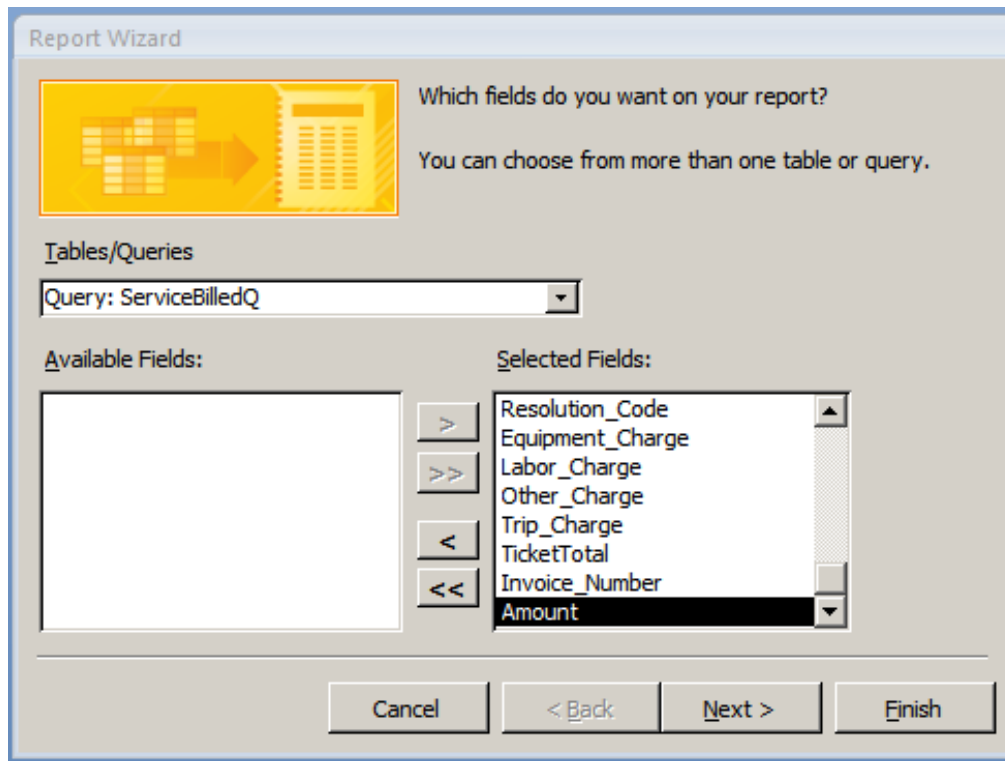


And select OK.

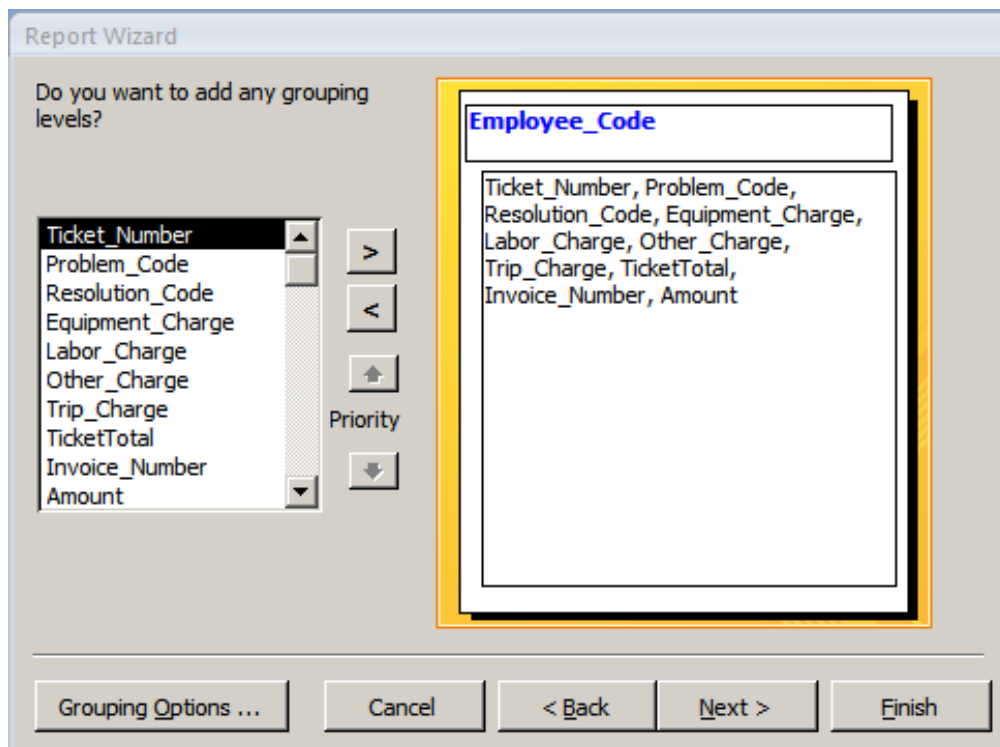
Also add criteria for Ticket\_Status and Service\_Ticket\_Id.

Ticket_Status	Service_Ticket_Id
dbo_SV_Service_Ticke	dbo_SV_Service_Ticke
<input type="checkbox"/>	<input type="checkbox"/>
"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.



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.

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

Summary Options ...

Cancel < Back Next > Finish

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.

Report Header

ServiceBilledQ

Page Header

Employee\_Code Number Problem\_Co Resolution

Employee\_Code Header

Employee\_Code

Detail

Ticke Problem\_Co Resolutio

Page Footer

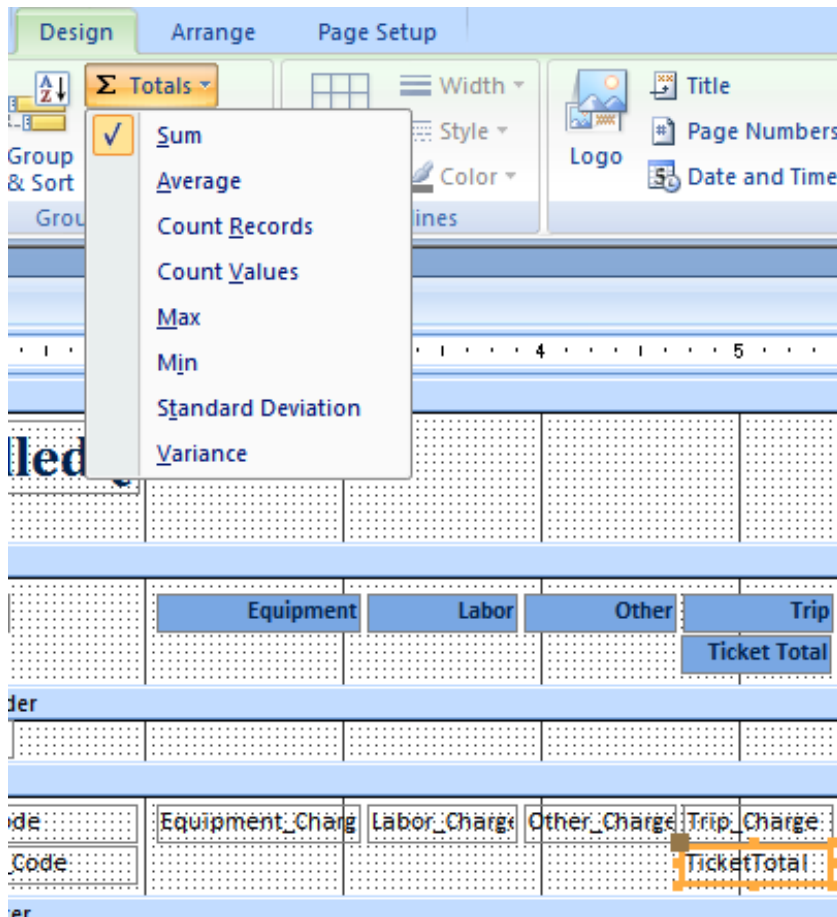
Delete the label. If you delete the Field just drag it back from the fields list. Continue moving fields till your layout looks like the image below.

The screenshot shows a report design view for a report titled "ServiceBilledQ". The report is organized into several sections:

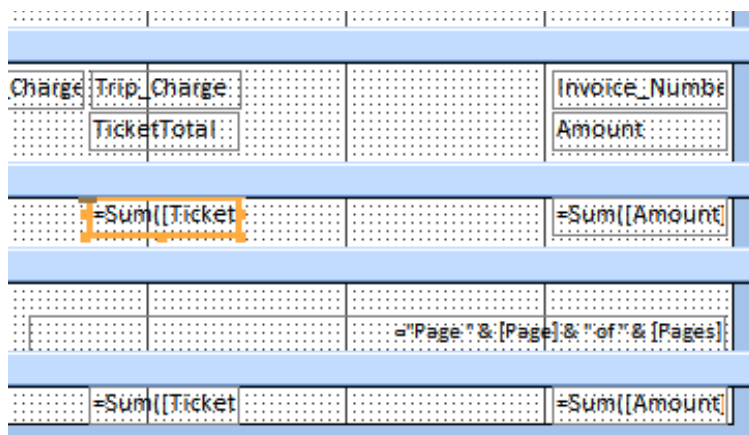
- Report Header:** Contains the report title "ServiceBilledQ".
- Page Header:** Contains fields for "Ticket #", "Problem", "Resolution", "Equipment", "Labor", "Other", "Trip", "Ticket Total", "Invoice #", and "Amount".
- Employee\_Code Header:** Contains the field "Employee\_Code".
- Detail:** Contains fields for "Ticket\_Nu", "Problem\_Code", "Resolution\_Code", "Equipment\_Charg", "Labor\_Charge", "Other\_Charge", "Trip\_Charge", "TicketTotal", "Invoice\_Numbe", and "Amount".
- Page Footer:** Contains the field "=New()" and a page number field "=Page." & [Page] & " of " & [Pages]."

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.





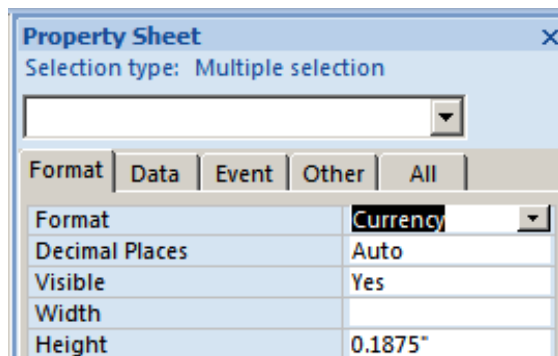
Do the same for the Amount field from the invoice. We will now have a subtotal by service tech and a grand total for all service techs.



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

	Equipment_Charg	Labor_Charg	Other_Charg	Trip_Charg		Invoice_Numbe
				TicketTotal		Amount
				=Sum([Ticket		=Sum([Amount
						a"Page "& [Page] & " of " & [Pages]
				=Sum([Ticket		=Sum([Amount

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	Equipment	Labor	Other	Trip	Invoice #
	Resolution				Ticket Total	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
					<b>\$1,312.10</b>	<b>\$1,393.60</b>
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
					<b>\$677.87</b>	<b>\$576.17</b>
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
					<b>\$707.20</b>	<b>\$608.29</b>
					<b>\$2,697.17</b>	<b>\$2,578.06</b>

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.

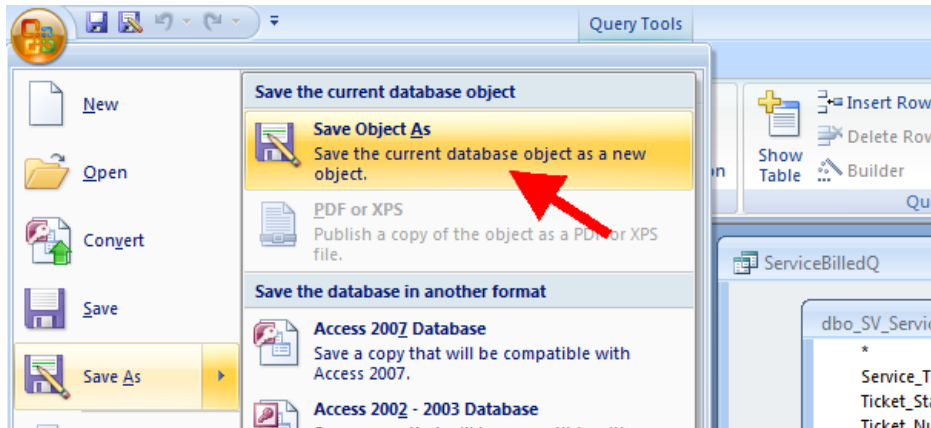
In this report we have learned how to group and total. We have learned how to expand the detail section to show more data than will fit on one line.

### 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

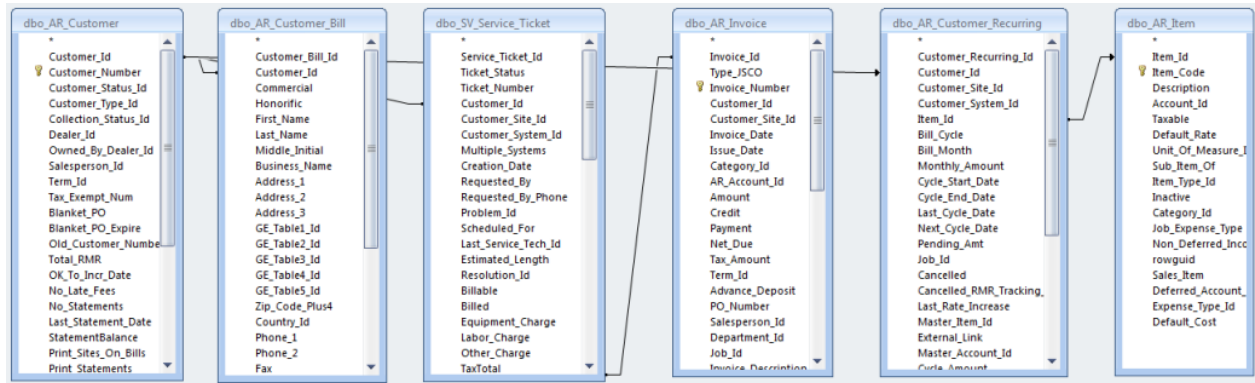
Then, using the table list, add these tables to the new query:

- AR\_Customer
- AR\_Customer\_Bill
- AR\_Customer\_Recurring
- AR\_Item

Remove these tables:

- SV\_Problem
- SV\_Resolution
- SV\_Service\_Tech
- SY\_Employee

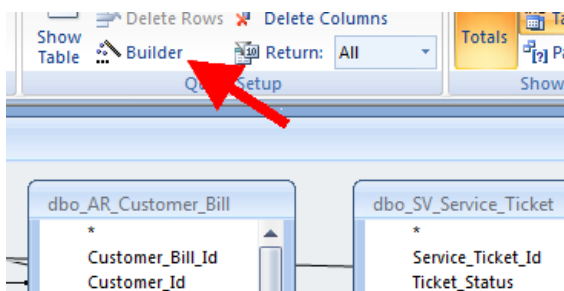
Link the tables as shown:

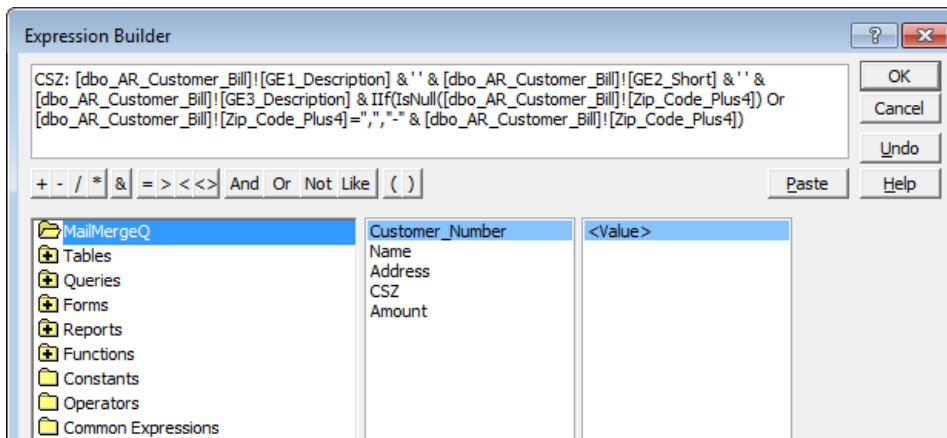


Add and remove fields till you have this list:

- 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.

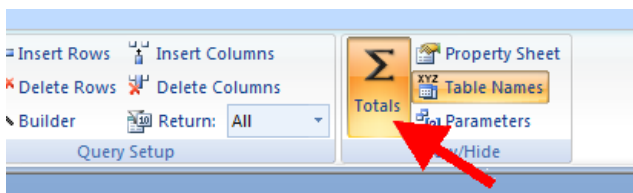




This formula could be simpler but I added an IIf function to format the City State and Zip differently if there was a Zip Plus 4. Add the following “Where” entries:

Field	Table	Group By	Expression	Where
Amount: Amount	dbo_AR_Invoice			
Customer_Bill_Id	dbo_AR_Customer_Bi			
Creation_Date	dbo_SV_Service_Ticket			
Item_Code	dbo_AR_Item			
		<input checked="" type="checkbox"/>	<>0	
		<input type="checkbox"/>	<>1	
		<input type="checkbox"/>	>=CDate("01/01/2010") And <=CDate("01/01/2011")	Is Null
		<input type="checkbox"/>	>=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	Table	Group By	Expression	Where
Customer_Number	dbo_AR_Customer	First		
Name: Business_Nam	dbo_AR_Customer_Bi	First		
Address: Address_1	dbo_AR_Customer_Bi	First		
CSZ: First([dbo_AR_C			Expression	
Amount: Amount	dbo_AR_Invoice		Sum	
Customer_Bill_Id	dbo_AR_Customer_Bi		Where	
Creation_Date	dbo_SV_Service_Ticket		Where	
Item_Code	dbo_AR_Item		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 done, 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]:

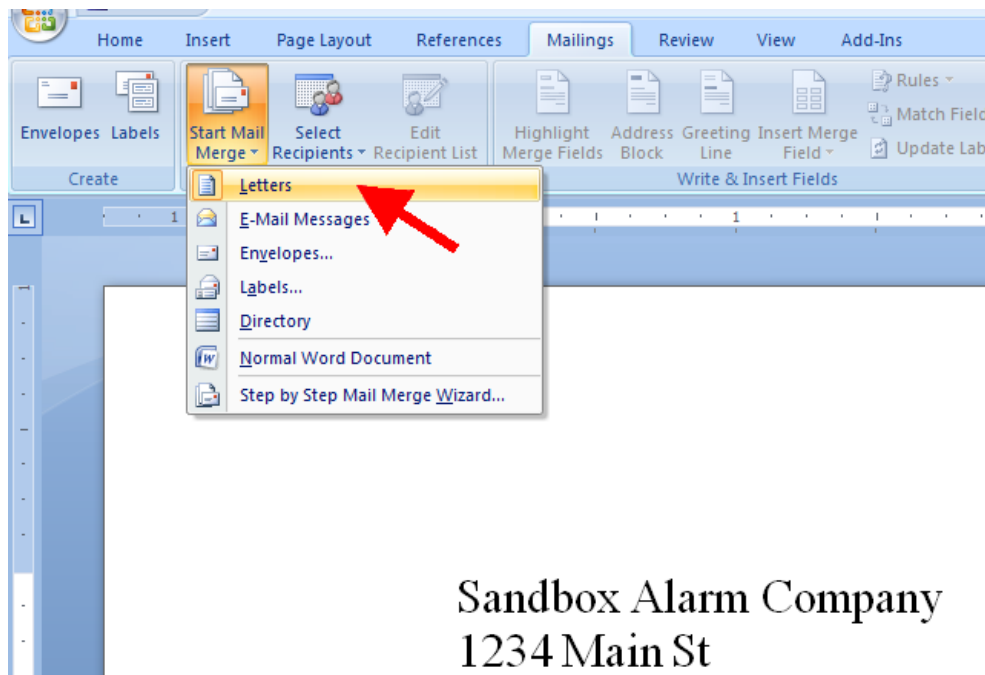
Our records show that last year you were invoiced \$0.00 for service calls. You might like to know more about our service contract, which covers the cost of parts and labor for normal repairs. The actual cost varies depending on your alarm system, but it is usually less than the time and materials costs 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.

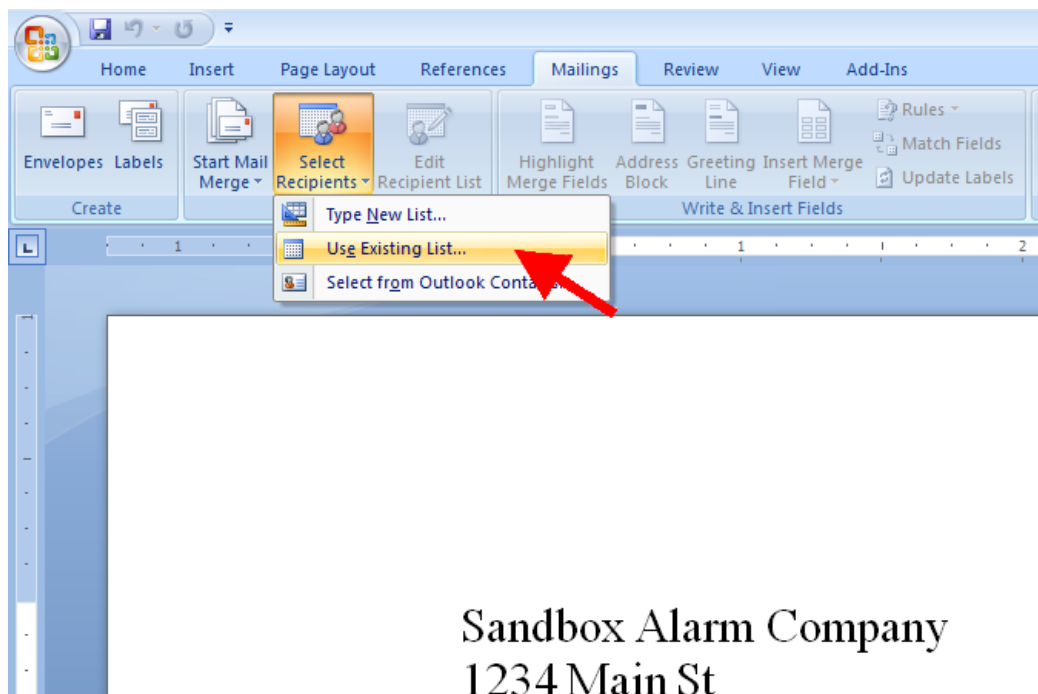
Sincerely,

Mathew S. Howe  
General Manager

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

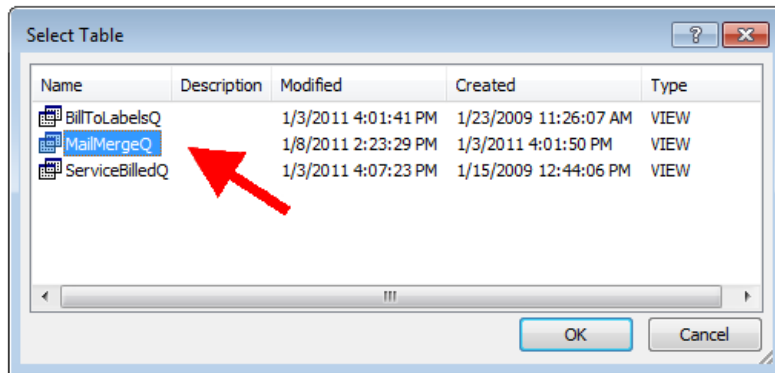


Next choose your recipients:

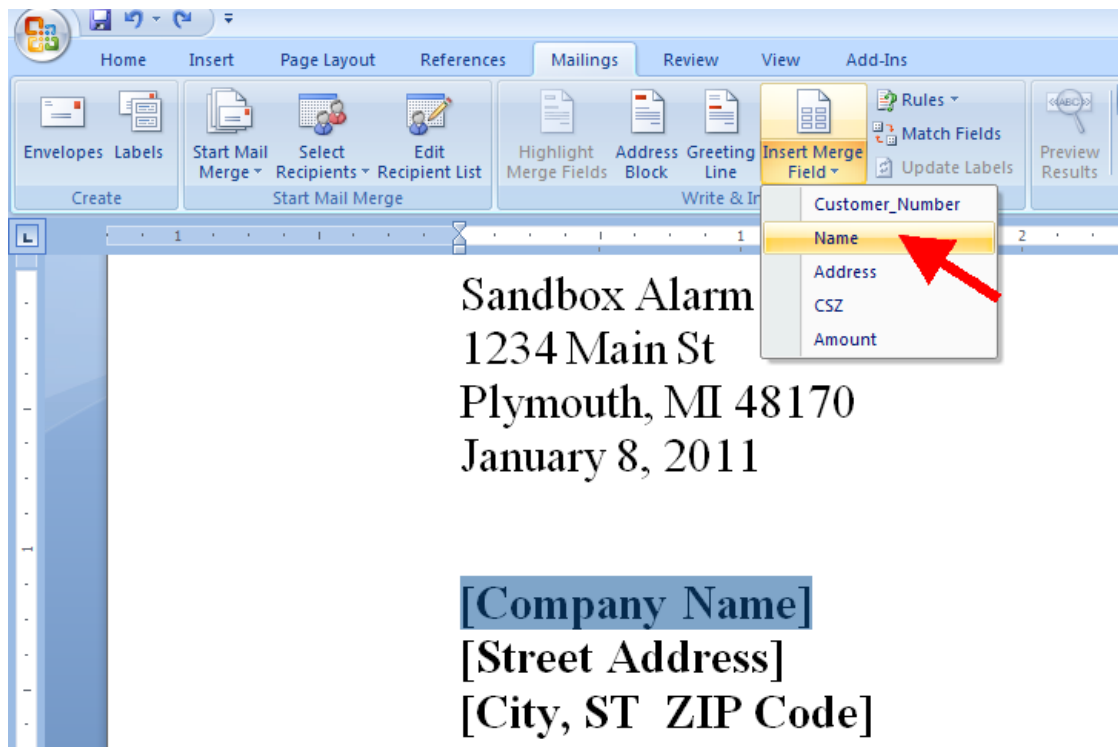


Locate and select your Access database and choose the MailMergeQ query:





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»

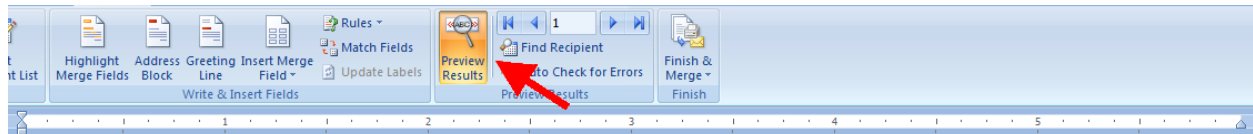
«Name»:

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:

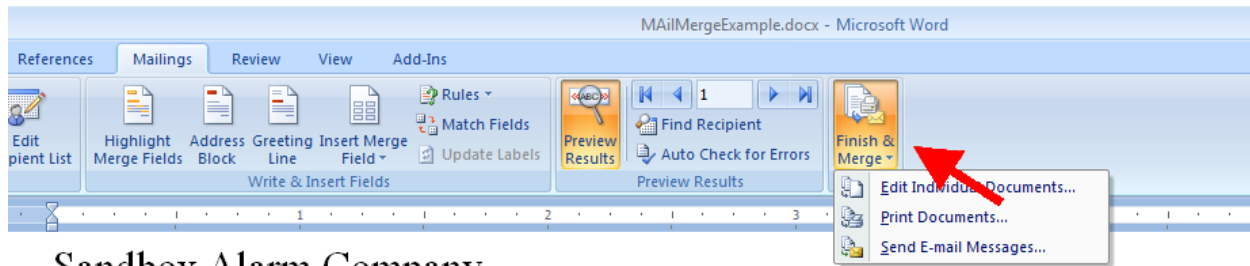
Our records show that last year you were invoiced \$606.24 for service calls. You might like to know more about our service contract, which covers the cost of parts and labor for normal repairs. The actual cost varies depending on your alarm system, but it is usually less than the time and materials 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 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

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

### 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  $\text{Amount} = 5 \text{ And } \text{City} = \text{'Flint'} \text{ Or } \text{City} = \text{'Detroit'}$  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  $(\text{Amount} = 5 \text{ And } \text{City} = \text{'Flint'}) \text{ Or } \text{City} = \text{'Detroit'}$ . The And operator is processed first just like the multiplication operator in arithmetic. If our where clause were written as  $\text{Amount} = 5 \text{ And } (\text{City} = \text{'Flint'} \text{ Or } \text{City} = \text{'Detroit'})$  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 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: