How to create, refresh and distribute Excel PowerPivot documents without the need for SharePoint

Contents

Background

With Excel 2010 Microsoft introduced PowerPivot which is a powerful tool for analysis and presentation of large amount of data with Excel. But as a start PowerPivot didn't had an API (Application Program Interface) which allowed you to automatically update the database. To present and refresh information you had to use SharePoint Enterprise. Not all organizations want to invest in SharePoint which makes PowerPivot less useful as corporate reporting tool. Read "Is SharePoint Overkill for BI" blog http://prologika.com/CS/blogs/blog/archive/2014/01/23/is-sharepoint-overkill-for-bi.aspx . But in Excel 2013 the API has been updated with a refresh function which makes it possible to refresh the PowerPivot database with a simple VBA command. But there are still limitations in the PowerPivot API. You cannot change a PowerPivot connection there a password is used on the run. You can change a connection to an Excel sheet on the run but if a password is used in the connection that password is stored in plain text in the Excel document and it will take less when a coffee brake to find the pass word. If you use a Windows Server authentication you will be prompted for the password each time the connection is opened so you need a windows (domain) account to access the database if the process is to be automated.

Situation and preferences

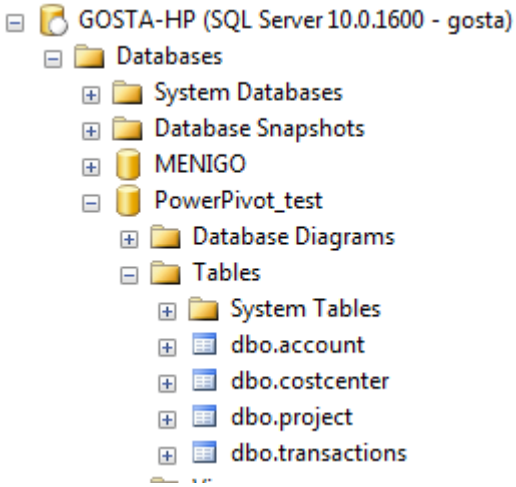
You are responsible for following up the finances in a small to middle sized company. There are a number of cost centers responsible. There are a lot of tools or systems which are feasible to solve your needs.

The purpose of this document is not to guide you throw an evaluation of all available tools but to focus on a very pragmatic solution with a very low price tag. The company has an accounting system with a SQL-server database which allows you to extract information on a regular basis. You don't want to invest in a SharePoint solution. The number of transactions may be more than 10 000 000 so a "cube of data with many dimensions" can be big and time consuming to create. If everybody needs access to that cube you need process power in a server. Also you can expect peak access needs short after bookkeeping closing dates and during working hours. For security reasons you don't want to make all finance information available to everybody but to provide subsets of data to each individual cost center responsible.

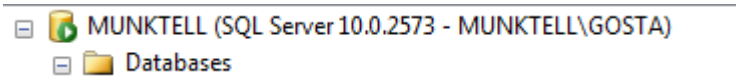
Solution

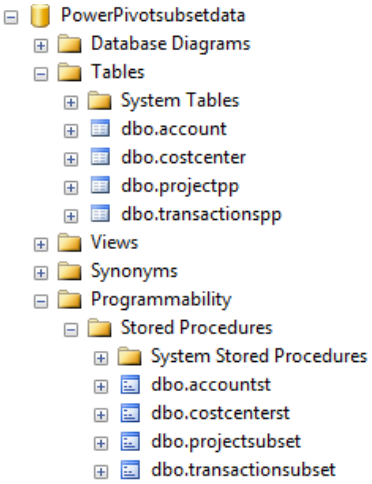
This document describes a simplified solution. You need to have experience in VBA programming, PowerPivot, SQL programming and of course knowledge about the database structure of your finance system. There is no ambition in this document to describe how you can use PowerPivot.

Starting point:
The heart of the solution is a template which is an Excel document. This template consists of a PowerPivot database which is loaded with data by a VBA script. The template also includes a number of sheets where one sheet is a “control sheet”. In that sheet you as responsible can manage the receivers of readymade Excel documents. In this simple example only a list of cost centers is included. Information about where to save the documents per cost centers is probably needed.
The template will be common for all users. The only difference is the content in terms of information from the finance system. Here you have two options. The obvious option is to extract only transactions relevant for each cost centers. But you can also prune the dimensions so for example only projects relevant for each cost center like in this example will be extracted.
You will probably run the finance system with the SQL database on one separate server and this Excel application on another server. The Excel application needs access to the database. As mentioned in the background you must use a Windows account to access the SQL server to run this application.
If your organization has a Windows server the best way is to create a Windows domain user account with the necessary privilege to the SQL Server database. But there is a way around this step. At first you need a SQL Server instance installed on the computer there you run this application.
Create an SQL Server account with the necessary privilege to get data on the server there the finance system is running. The server is GOSTA-HP the name of the database is PowerPivot_test with only 4 tables for example

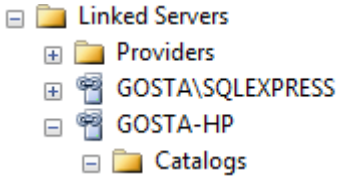


In the local server (PC) there you run this Excel application you create a SQL database PowerPivotsubsetdata:

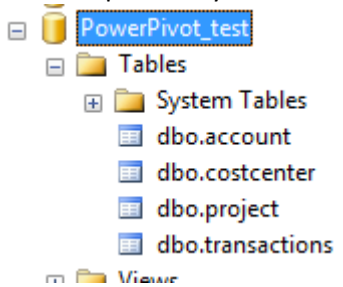




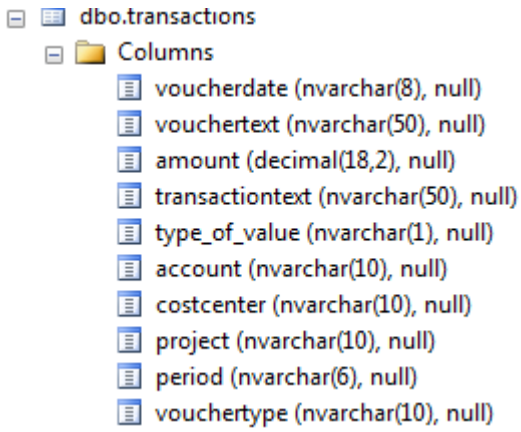
You also have to create a linked server to the server Gosta-HP!



With the possibility to access the database:



This makes it possible to transfer data from the server Gosta-HP (with the finance system) to the local database using a SQL-Server account.
Example of a table:



The transaction table is simplified. There is also a table transactionspp with the same layout.
With PowerPivot you can connect to a database and tables in that database or create queries to extract data from a database. You can also execute a stored procedure. But it is not possible to include parameters which changes on the run with the built in API. This is a limitation which has to be walked around. One way to do that is to use “temporary” tables as source for the PowerPivot database. Those tables can be truncated and reloaded on the run by a VBA script.

In this case those tables transactionspp and projectpp are to be find in the local database PowerPivot_subsetdata. The tables account and costcenter are also refreshed by the VBA script. Those tables are refreshed by the stored procedures:

```
USE [PowerPivotsubsetdata]
GO
/****** Object:  StoredProcedure [dbo].[accountst]    S
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
ALTER proc [dbo].[accountst]
AS

truncate table account
INSERT INTO account
SELECT *
--Linked server [gosta-hp]
FROM          [gosta-hp].[powerpivot_test].dbo.account
```

And

```
USE [PowerPivotsubsetdata]
GO
/****** Object:  StoredProcedure [dbo].[costcenterst]    Sc
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
ALTER proc [dbo].[costcenterst]
AS

truncate table costcenter
INSERT INTO costcenter
SELECT *
--Linked server [gosta-hp]
FROM          [gosta-hp].[powerpivot_test].dbo.costcenter
```

Table Transactionspp is refresh by a stored procedure:

```
USE [PowerPivotsubsetdata]
GO
/****** Object:  StoredProcedure [dbo].[transactionsubset]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
ALTER proc [dbo].[transactionsubset]
        @costcenter nvarchar(10)
        as
begin

   SET NOCOUNT ON
truncate  table transactionspp
if @costcenter ='all'
insert into transactionspp
select *
--Loaded from the linked server
from [gosta-hp].[powerpivot_test].dbo.transactions
else

insert into transactionspp
select *
--Loaded from the linked server
from [gosta-hp].[powerpivot_test].dbo.transactions
where costcenter = @costcenter

end
```

The procedure has the parameter costcenter.
The first step is to truncate table transactionspp
If costcenter is blank all transaction in transaction will load transactionspp from the table transactions in the linked server [gosta-hp]
(This is a special case which may be is not relevant as the template for the complete finance system will be another PP application) from the table transactions in the linked server [gosta-hp]
If costcenter is not blank only transactions which matches the actual costcenter will load Transactionspp from the table transactions in the linked server [gosta-hp]

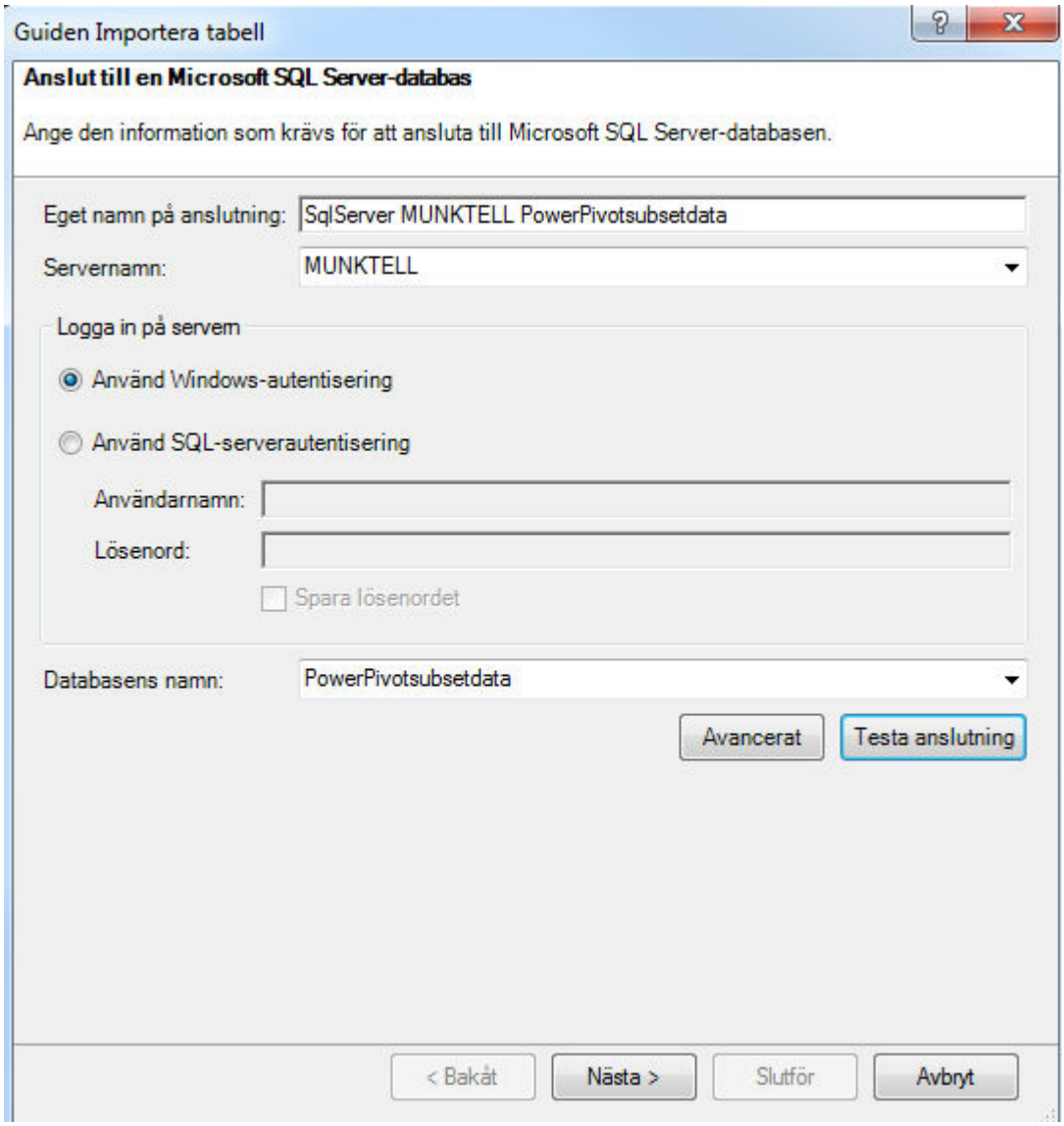
StoredProcedure projectsubset will run after StoredProcedure transactionsubset:

```
USE [PowerPivotsubsetdata]
GO
/****** Object:  StoredProcedure [dbo].[projectsubset]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
ALTER proc [dbo].[projectsubset]
AS

truncate table projectpp
INSERT INTO projectpp
SELECT DISTINCT A.project, A.[project description]
FROM
--Linked server [gosta-hp]
[gosta-hp].[powerpivot_test].dbo.project as A INNER JOIN
transactionspp ON a.project = transactionspp.project
```

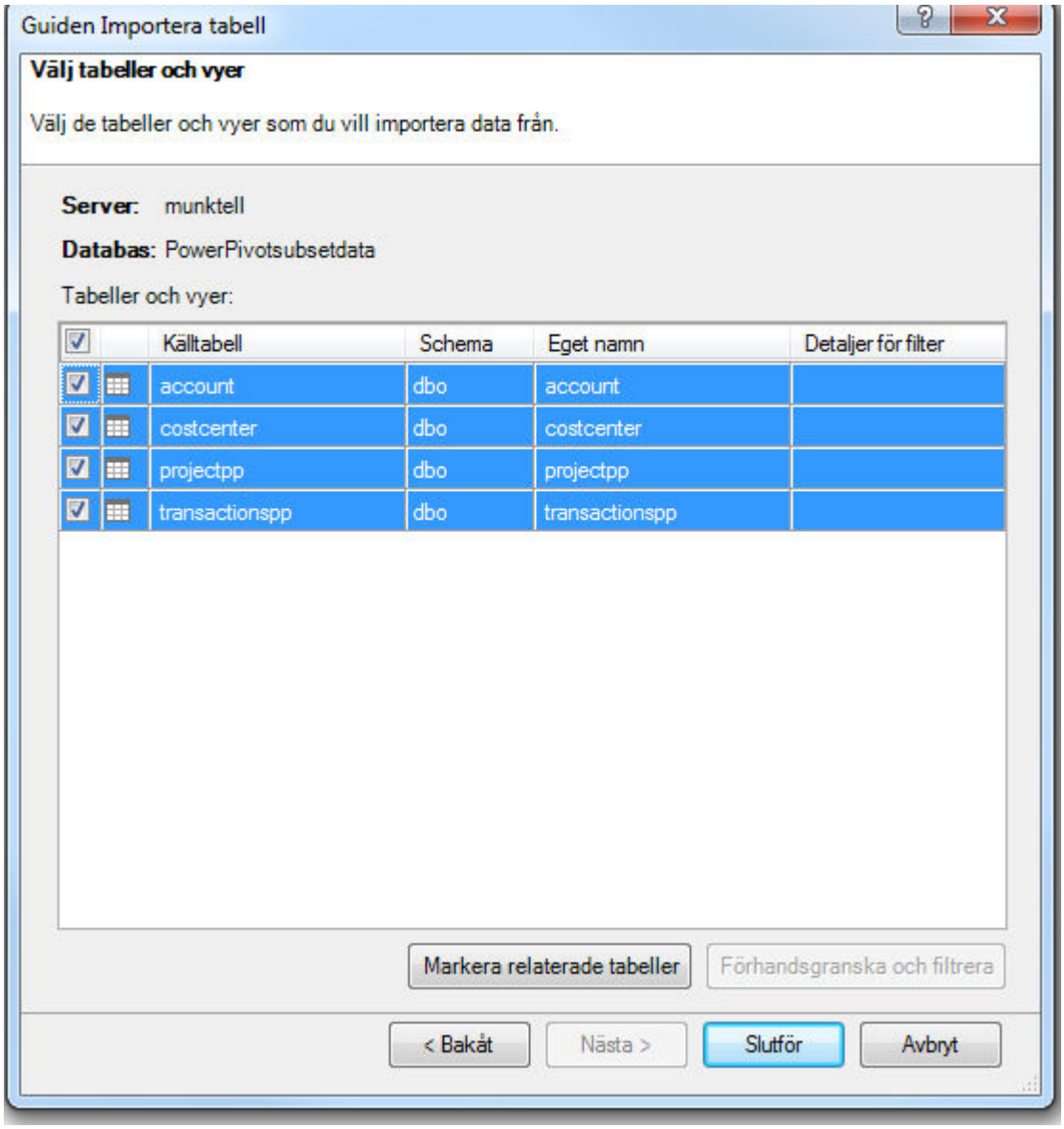
Only projects within the actual costcenter will be inserted in table projectpp.
Now you can start to configure the PowerPivot database

Step 1
Open a blank Excel workbook. Open the PowerPivot database. Create a new connection:

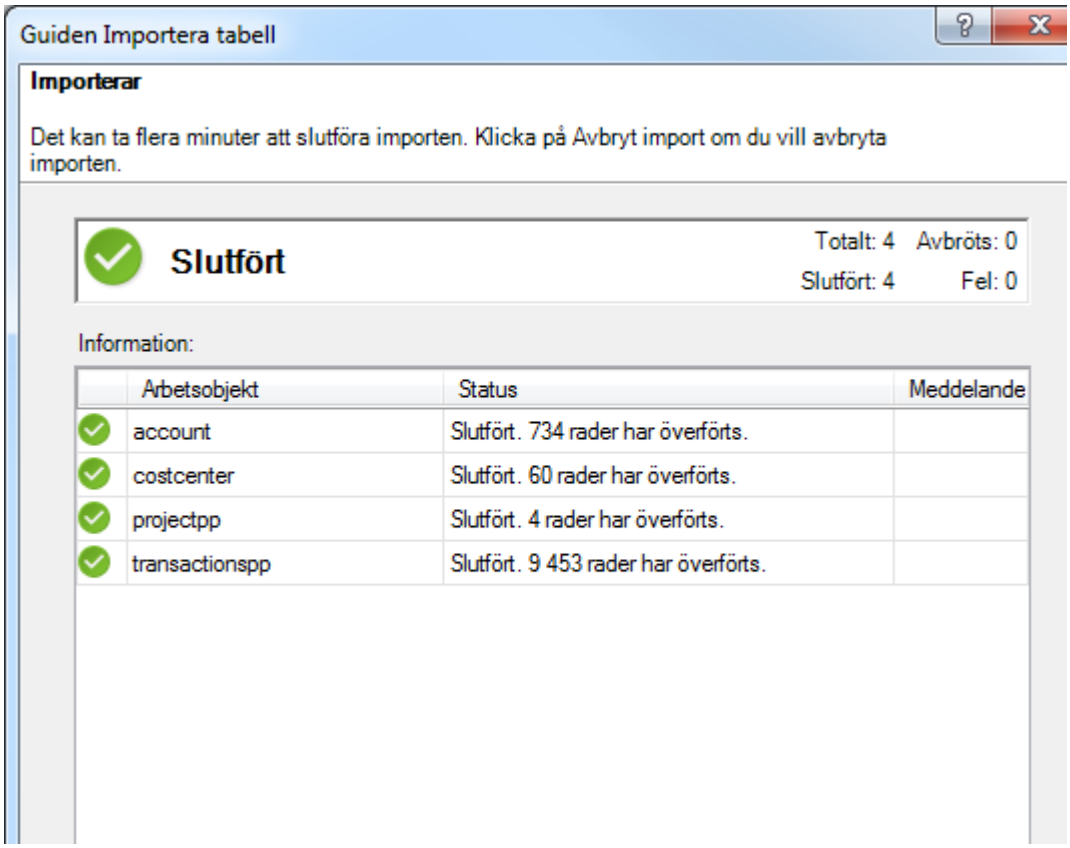
(Sorry for Swedish). Enter the name of the server. Use Windows-authentication.
Database name: PowerPivotsubsetdata


Guiden Importera tabell
Anslut till en Microsoft SQL Server-databas
Ange den information som krävs för att ansluta till Microsoft SQL Server-databasen.
Eget namn på anslutning:
SqlServer MUNKTELL PowerPivotsubsetdata
Servernamn:
MUNKTELL
Logga in på servern
Använd Windows-autentisering
Använd SQL-serverautentisering
Användarnamn:
Lösenord:
Spara lösenordet
Databasens namn:
PowerPivotsubsetdata
Avancerat
Testa anslutning
< Bakåt
Nästa >
Slutför
Avbryt

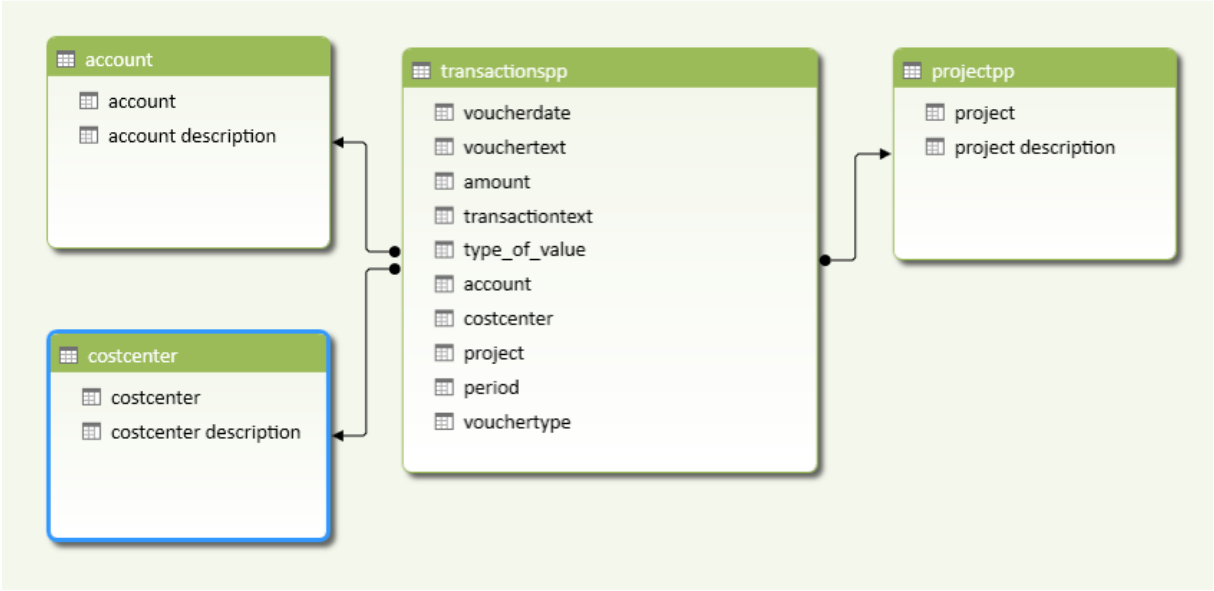
Press Next:



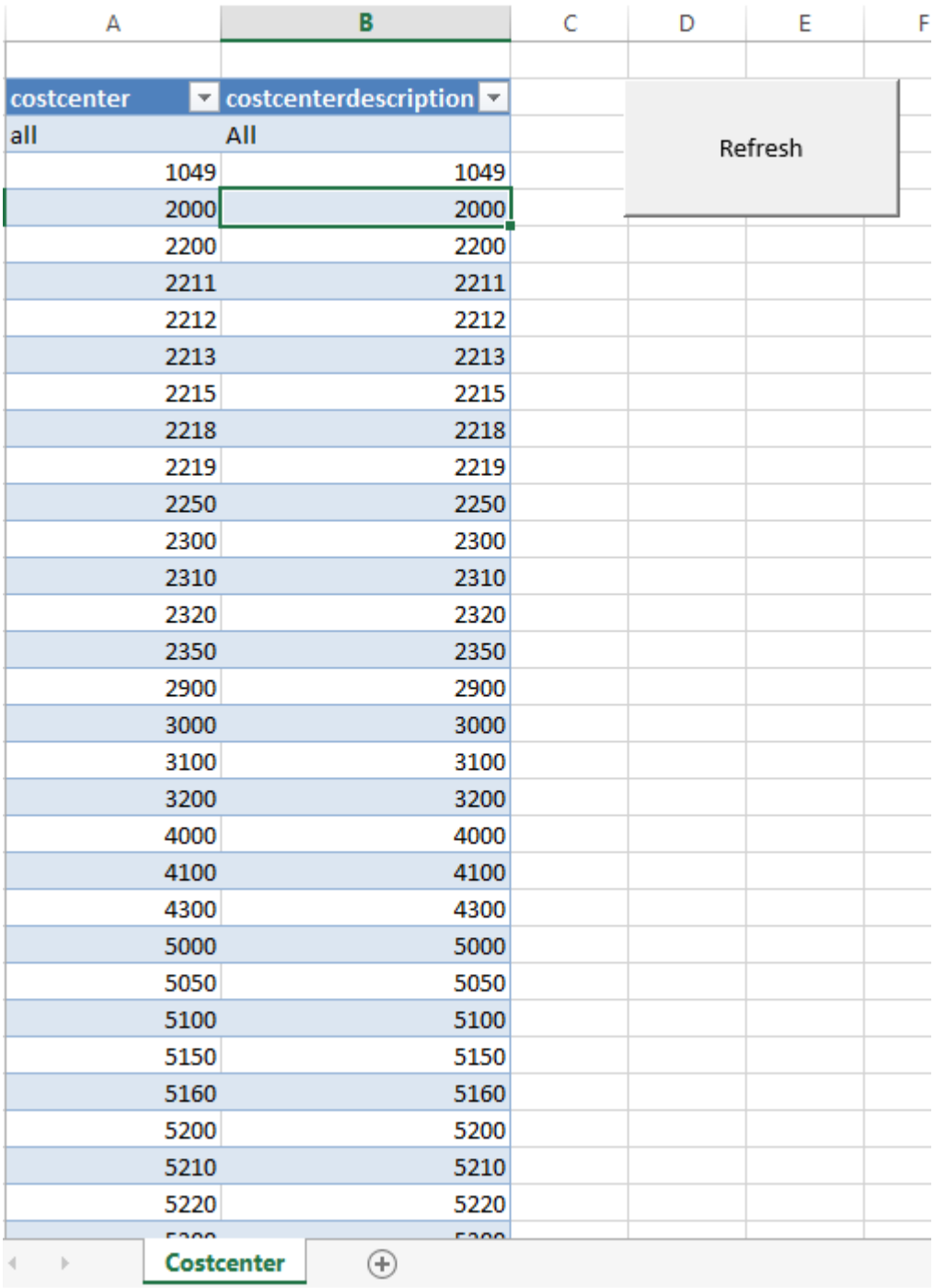
Import all tables in the database. Press finish.
You should get something like:



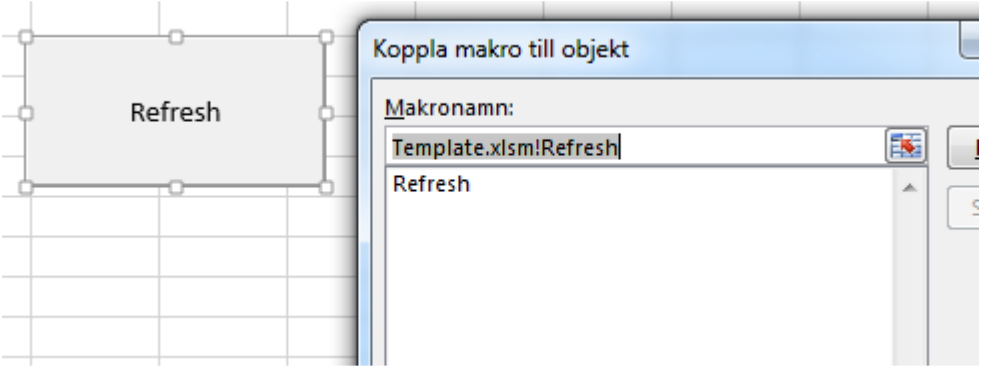
(It doesn't matter if a table is empty). Press close.
In the diagram view you should be able to see the tables and make relations.



The next step is to create a sheet where you can tell which concenters which are to be produced and may be where to be delivered (individual catalogues.



The columns, rows costcenter, costcenterdescription is a table name costcenter.
The button refer to a Macro “Refresh”



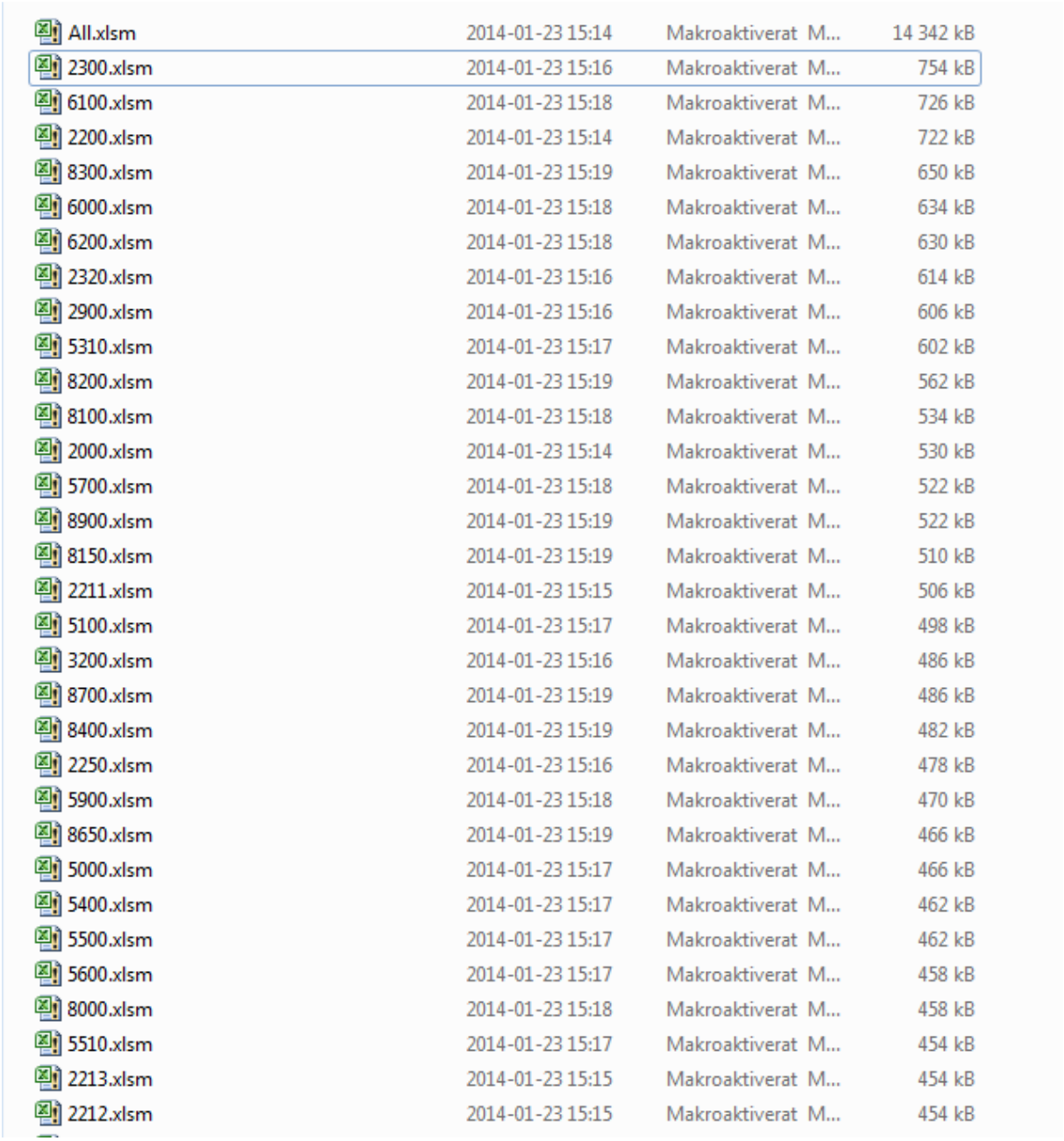
which contains the VBA script:

```
Option Explicit

Sub Refresh()

       Dim conn As New ADODB.Connection
        Dim cmd As New ADODB.Command
        Dim costcenter, description As String
        Dim L, number_of_rows As Long

        'Connection to the local SQL database
        'Don't forget set a reference to a Microsoft ActiveX Data Object library
        conn.ConnectionString = "Provider=SQLNCLI;Server=(local);" _
& "Database=PowerPivotsubsetdata;DataTypeCompatibility=80;Integrated security=SSPI;"
        conn.Open
        cmd.ActiveConnection = conn
        cmd.CommandTimeout = "10000"

'update account table
cmd.CommandText = "Execute accountst "
cmd.Execute

'update costcenter table
cmd.CommandText = "Execute costcenterst "
cmd.Execute


number_of_rows = ActiveWorkbook.Worksheets("costcenter").range("costcenter").Rows.Count

For L = 1 To number_of_rows

costcenter = range("costcenter").Cells(L, 1)
'update  transactionspp table
cmd.CommandText = "Execute transactionsubset '" & costcenter & "'"
cmd.Execute

cmd.CommandText = "Execute projectsubset "
cmd.Execute
description = range("costcenter").Cells(L, 2)
ActiveWorkbook.RefreshAll
'The Exceldocument are in this simple case stored in the catalogue
'Reslutat
ActiveWorkbook.SaveCopyAs (ActiveWorkbook.Path & "\resultat\" & description & ".xlsm")


Next L
```

The macro Refresh can be executed by a scheduled task.

The result will look like where all files are subsets of the All.xlsm file.
There are about 1 900 000 transactions in the All.xlsm file.

(Of course the refresh macro itself should be omitted in the result files).



All.xlsm	2014-01-23 15:14	Makroaktiverat M...	14 342 kB
2300.xlsm	2014-01-23 15:16	Makroaktiverat M...	754 kB
6100.xlsm	2014-01-23 15:18	Makroaktiverat M...	726 kB
2200.xlsm	2014-01-23 15:14	Makroaktiverat M...	722 kB
8300.xlsm	2014-01-23 15:19	Makroaktiverat M...	650 kB
6000.xlsm	2014-01-23 15:18	Makroaktiverat M...	634 kB
6200.xlsm	2014-01-23 15:18	Makroaktiverat M...	630 kB
2320.xlsm	2014-01-23 15:16	Makroaktiverat M...	614 kB
2900.xlsm	2014-01-23 15:16	Makroaktiverat M...	606 kB
5310.xlsm	2014-01-23 15:17	Makroaktiverat M...	602 kB
8200.xlsm	2014-01-23 15:19	Makroaktiverat M...	562 kB
8100.xlsm	2014-01-23 15:18	Makroaktiverat M...	534 kB
2000.xlsm	2014-01-23 15:14	Makroaktiverat M...	530 kB
5700.xlsm	2014-01-23 15:18	Makroaktiverat M...	522 kB
8900.xlsm	2014-01-23 15:19	Makroaktiverat M...	522 kB
8150.xlsm	2014-01-23 15:19	Makroaktiverat M...	510 kB
2211.xlsm	2014-01-23 15:15	Makroaktiverat M...	506 kB
5100.xlsm	2014-01-23 15:17	Makroaktiverat M...	498 kB
3200.xlsm	2014-01-23 15:16	Makroaktiverat M...	486 kB
8700.xlsm	2014-01-23 15:19	Makroaktiverat M...	486 kB
8400.xlsm	2014-01-23 15:19	Makroaktiverat M...	482 kB
2250.xlsm	2014-01-23 15:16	Makroaktiverat M...	478 kB
5900.xlsm	2014-01-23 15:18	Makroaktiverat M...	470 kB
8650.xlsm	2014-01-23 15:19	Makroaktiverat M...	466 kB
5000.xlsm	2014-01-23 15:17	Makroaktiverat M...	466 kB
5400.xlsm	2014-01-23 15:17	Makroaktiverat M...	462 kB
5500.xlsm	2014-01-23 15:17	Makroaktiverat M...	462 kB
5600.xlsm	2014-01-23 15:17	Makroaktiverat M...	458 kB
8000.xlsm	2014-01-23 15:18	Makroaktiverat M...	458 kB
5510.xlsm	2014-01-23 15:17	Makroaktiverat M...	454 kB
2213.xlsm	2014-01-23 15:15	Makroaktiverat M...	454 kB
2212.xlsm	2014-01-23 15:15	Makroaktiverat M...	454 kB

Summary and conclusion

With PowerPivot in Excel 2013 it is possible to make a simple application which can produce subsets of data from a central database and deliver individual Excel files without the need to set up a SharePoint solution.