Automatically Creating UML Database Diagrams for SQL Server

03 May 2016

*by* [*Phil Factor*](https://www.simple-talk.com/author/phil-factor/)

FROM: <https://www.simple-talk.com/sql/sql-tools/automatically-creating-uml-database-diagrams-for-sql-server/>

SQL Server database developers seem reluctant to use diagrams when documenting their databases. It is probably because it has, in the past, been difficult to automatically draw precisely what you want, other than  a vast Entity-relationship  diagram. However, you can do it without buying any third-party tool, just using some existing Java-based open-source tools; and can even automate it entirely, using SQL and PowerShell.

Diagrams are often a great help in understanding how databases work. I’ve always wanted to generate various database diagrams automatically to go with my database documentation. The problem is that, once you’ve drawn them nicely in a drawing package, you feel reluctant to alter the database because you’d then have to re-draw the diagrams to reflect those changes. This article explains a way of creating easily ‘refreshed’ automated diagrams based only on open-source, or free, tools.

**PlantUML**

We’ll be using PlantUML. PlantUML is an open-source project for writing Unified Modelling Language (UML) diagrams for Sequence, Use-Case, Class, Activity, Component, State and Objects. It uses GraphViz for its graphical output. Its use, like GraphViz, is to render quite complex diagrams from a simple intuitive language based on text.

For example, this will give you a simple UML sequence diagram:

@startuml

actor Phil #red

actor Editor #green

Phil -> Editor: suggestion for an article

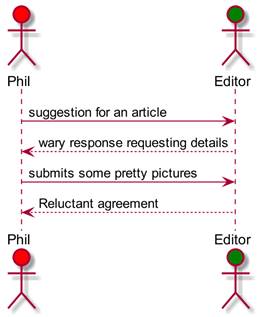
Editor --> Phil: wary response requesting details

Phil -> Editor: submits some pretty pictures

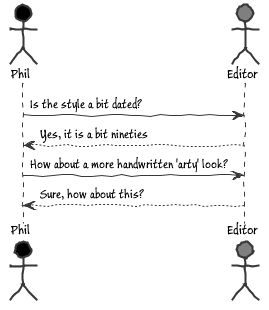
Editor --> Phil: Reluctant agreement

@enduml

If this code is then processed by PlantUML, would provide this .PNG image:



Don't like the style? You can specify it how you like. With a little extra tweaking with the **skinparam**, you can get this instead



the code changes are simple

@startuml  
skinparam handwritten true  
skinparam monochrome true  
skinparam packageStyle rect  
skinparam defaultFontName Buxton Sketch  
skinparam shadowing false  
  
actor Phil #black  
actor Editor #grey  
Phil -> Editor: Is the style a bit dated?  
Editor --> Phil: Yes, it is a bit nineties  
  
Phil -> Editor: How about a more handwritten 'arty' look?  
Editor --> Phil: Sure, how about this?  
  
@enduml

Although you may have little urge to use UML diagrams in your everyday work, PlantUML is versatile enough to provide for a range of requirements. For a relational database person who winces at the terms ‘class’ and ‘persistence’, even UML modelling has its uses as a way of communicating ideas and designs.

In this article I’m going to show how it can be used to easily generate class diagrams for database objects. I will use TSQL code to create the PlantUML code directly.

**Getting up and running**

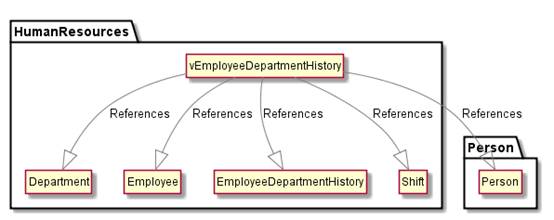
To get you started, there is a little [browser application here](http://plantuml.com/plantuml/uml) on the PlantUML site  that allows you to type in PlantUML code and view the results. With that, and the PlantUML manual, you should be well away with using this interesting application. Chrome has an Add-in that runs PlantUML called the UML Diagram Editor.

There are many ways of using PlantUML. There are several Atom packages that add PlantUML integration. If you use DocuWiki for documentation, there is an add-in for PlantUML, which is handy for teams since there is nothing to install on your machine then. If you want to use PlantUML from the command-line, there are [install instructions here](http://plantuml.com/starting.html). I use AsciiDocFx, which installs the prerequisites for you (though you may need to set the path to your GraphViz install. You can use your favourite editor via a command-line interface as well. PlantUML uses simple text-based instructions to render UML diagrams and these can be generated from SQL to show you such things as the intricacies of your database permission system, the foreign-key dependencies, or the details of your indexing strategy for a table.

We’ll just stick to showing a subset of your database objects and the dependencies between them.

**Data Modelling With Unified Modelling Language (UML)**

Let’s look at AdventureWorks, just so we can try things out. I’ve always been caught out by views so it would be nice to see those objects in the database that a view references. Here is a simple diagram.



Note that schemas have been represented as folders. All I did to do this was to use the text:

@startUML

hide empty members

hide circle

skinparam classarrowcolor gray

HumanResources.vEmployeeDepartmentHistory --|> HumanResources.Department:References

HumanResources.vEmployeeDepartmentHistory --|> HumanResources.Employee:References

HumanResources.vEmployeeDepartmentHistory --|> HumanResources.EmployeeDepartmentHistory:References

HumanResources.vEmployeeDepartmentHistory --|> Person.Person:References

HumanResources.vEmployeeDepartmentHistory --|> HumanResources.Shift:References

@enduml

I didn’t type this in. I just executed the SQL query that finds the soft references and foreign key references to and from the object that you name…

DECLARE @object\_ID INT

SELECT @Object\_ID=object\_id('HumanResources.vEmployeeDepartmentHistory')

SELECT coalesce(object\_schema\_name(referencing\_ID)+'.','')

+ object\_name(referencing\_ID) +' --|> '

+ referenced\_schema\_name+'.'+Referenced\_Entity\_name

+ ':References'

--AS reference

FROM sys.sql\_expression\_dependencies

WHERE (referencing\_id =@object\_ID

OR referenced\_ID = @object\_ID)

AND is\_schema\_bound\_reference =0

and referenced\_ID is not null

UNION ALL

SELECT coalesce(object\_schema\_name(parent\_object\_ID)+'.','')

+ object\_name(parent\_object\_ID) + ' --|> '

+ coalesce(object\_schema\_name(referenced\_object\_ID)+'.','')

+ object\_name(referenced\_object\_ID)+':FK'

FROM sys.foreign\_keys

WHERE parent\_object\_ID = @object\_ID

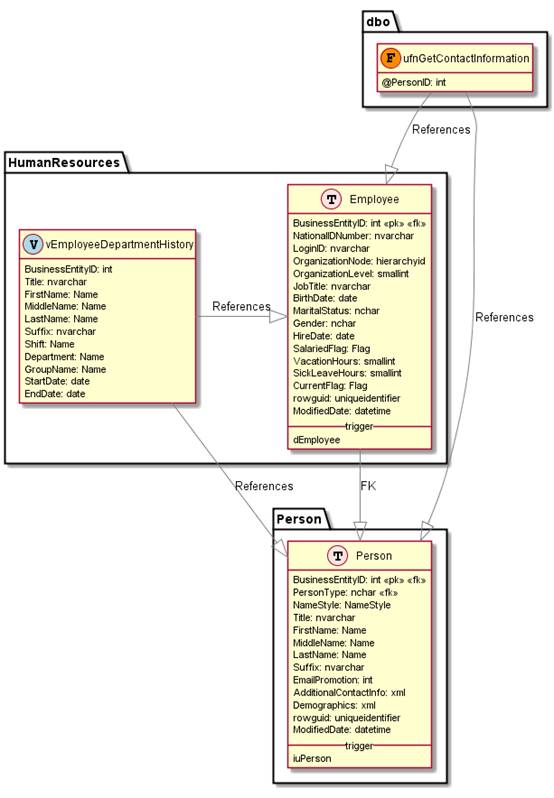
OR referenced\_object\_ID = @object\_ID

…and copied the result from the results pane (use text mode and set the ‘*Query*’ ‘*Options*’ *-> ‘Results’ -> ‘Text’ -> ‘Maximum number of characters displayed in each column*’ to 8192 or some other generous figure)

…but you can of course use it with any other database object such as **dbo.ufnGetContactInformation.** Obviously, there is plenty more you can include but there is something to be said for having just one clear message to every diagram.Be wary about including too much in a diagram. Any network diagram quickly morphs into a birds-nest diagram if you try to do too much. This type of diagram was never intended to provide an entire map of your database.

Once you’ve got some confidence, you can use PowerShell to update diagrams automatically. You just execute the code, save the result to disk, and pass the result to PlantUML, which has a command-line interface.(see http://plantuml.com/command\_line.html)

Sometimes we would like to do a bit more than this. Now UML never had a stereotype to deal with a relational database, but there [is an established database profile](http://www.agiledata.org/essays/umlDataModelingProfile.html) that includes a logical and physical model. As I have no strong feelings about the way that a table is represented, I’ve not deviated much from the standard class.



We have drilled in to the detail from the first diagram here to look more closely at the tables. PlantUML is able to give you immense diagrams which are probably useless for our purpose here. If you want to do this, I’d advise you just to open the wallet a bit and use SQL Dependency Tracker instead. Here we want to keep things simpler, and just want to document tables, views, and functions and their dependencies in detail.

The PlantUML source code that we used is this, generated from SQL code...

!define table(x) class x << (T,mistyrose) >>

!define view(x) class x << (V,lightblue) >>

!define table(x) class x << (T,mistyrose) >>

!define tr(x) class x << (R,red) >>

!define tf(x) class x << (F,darkorange) >>

!define af(x) class x << (F,white) >>

!define fn(x) class x << (F,plum) >>

!define fs(x) class x << (F,tan) >>

!define ft(x) class x << (F,wheat) >>

!define if(x) class x << (F,gaisboro) >>

!define p(x) class x << (P,indianred) >>

!define pc(x) class x << (P,lemonshiffon) >>

!define x(x) class x << (P,linen) >>

hide methods

hide stereotypes

skinparam classarrowcolor gray

table(HumanResources.Employee) {

BusinessEntityID: int <<pk>> <<fk>>

NationalIDNumber: nvarchar

LoginID: nvarchar

OrganizationNode: hierarchyid

OrganizationLevel: smallint

JobTitle: nvarchar

BirthDate: date

MaritalStatus: nchar

Gender: nchar

HireDate: date

SalariedFlag: Flag

VacationHours: smallint

SickLeaveHours: smallint

CurrentFlag: Flag

rowguid: uniqueidentifier

ModifiedDate: datetime

\_\_ trigger \_\_

dEmployee

}

table(Person.Person) {

BusinessEntityID: int <<pk>> <<fk>>

PersonType: nchar <<fk>>

NameStyle: NameStyle

Title: nvarchar

FirstName: Name

MiddleName: Name

LastName: Name

Suffix: nvarchar

EmailPromotion: int

AdditionalContactInfo: xml

Demographics: xml

rowguid: uniqueidentifier

ModifiedDate: datetime

\_\_ trigger \_\_

iuPerson

}

view(HumanResources.vEmployeeDepartmentHistory) {

BusinessEntityID: int

Title: nvarchar

FirstName: Name

MiddleName: Name

LastName: Name

Suffix: nvarchar

Shift: Name

Department: Name

GroupName: Name

StartDate: date

EndDate: date

}

tf(dbo.ufnGetContactInformation) {

@PersonID: int

}

dbo.ufnGetContactInformation -|> HumanResources.Employee:References

dbo.ufnGetContactInformation -|> Person.Person:References

HumanResources.vEmployeeDepartmentHistory -|> HumanResources.Employee:References

HumanResources.vEmployeeDepartmentHistory -|> Person.Person:References

HumanResources.Employee -|> Person.Person:FK

All we have done is to give a list of the tables that we want a diagram for.

Here is the code

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @ObjectsToShow='HumanResources.employee person.person HumanResources.vEmployeeDepartmentHistory dbo.ufnGetContactInformation',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

I have only shown how to execute the stored procedure I wrote. I have specified the four objects whose dependencies and relationships I’m interested in. You may want to see all the dependencies and relationship of any object but even just the directly linked objects can overwhelm the diagram. It is better to be selective. PlantUML will do its best to oblige but you are soon at the wallchart scale of activity.

This stored procedure can do other things. What if you want to see a UML diagram of all the objects that reference person.address Here, it is

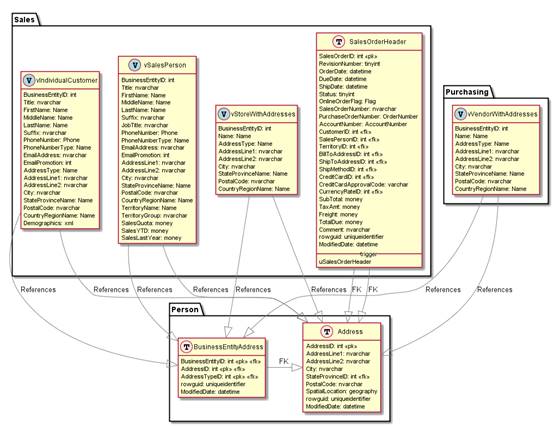
Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @DependsOn='person.address',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

…giving you this, once the output is passed to PlantUML. …



… or this …

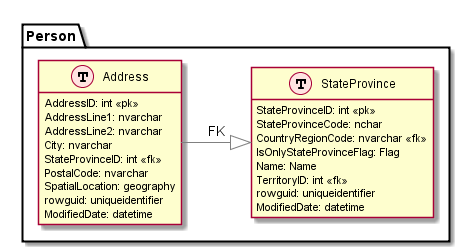
Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @DependenciesOf='person.address',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

ThePlantUMLCode

… giving you this … 

I’ve given the SQL code at the end of the article.

**Modifying your diagrams**

I have taken quite a few design decisions which you may not like. Fortunately, PlantUML has an excellent PDF manual which explains that you can change almost everything. It is so good that I’m not going to attempt to repeat anything from it. However, be assured that diagrams can be altered to suit your requirements: Related objects can be placed side by side, or vertically;, You can use dotted arrows or arrows with a different head; You can opt to avoid shadows, and change the colour of anything. My instinct is to leave GraphViz and PlantUML to do as much as possible by default.

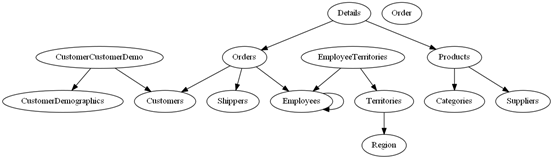
PlantUML supports a flavour of markdown called Creole. This allows you to have formatted notes and descriptions. There is even a full set of icons from OpenIconic. Because PlantUML is based on GraphViz, you can use Graphviz raw. This allows you to do more general network visualisation diagrams for general software-engineering purposes. For database people, it is great for plotting out chains of dependencies in database data or in databases themselves.

**GraphViz Dot Language**

PlantUML can execute graphviz dot language files as well. We can therefore use PlantUML to draw ‘network’ diagrams, which is something thet GraphViz does extraordinarily well. Most of the time, you’d probably want to do it with data but you can do it for your SQL Server metadata. Here, for example is Northwind’s table dependency diagram based on foreign key relationships

@startdot  
digraph G {   
CustomerCustomerDemo -> CustomerDemographics;  
CustomerCustomerDemo -> Customers;  
Employees -> Employees;  
EmployeeTerritories -> Employees;  
EmployeeTerritories -> Territories;  
Order Details -> Orders;  
Order Details -> Products;  
Orders -> Customers;  
Orders -> Employees;  
Orders -> Shippers;  
Products -> Categories;  
Products -> Suppliers;  
Territories -> Region;}   
@enddot

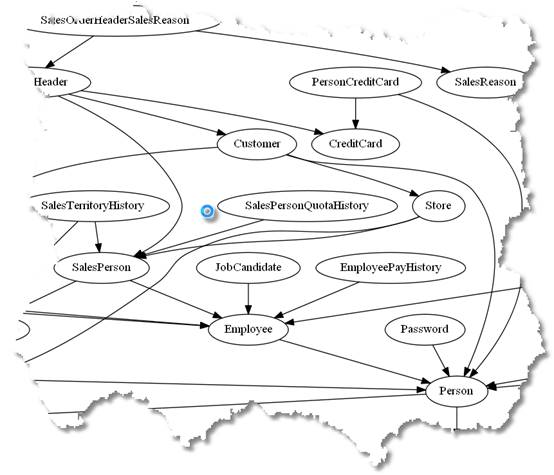
This code produced this diagram (you can click on it to enlarge it)

[](https://www.simple-talk.com/iwritefor/articlefiles/2405-ninthdot-diagram.png)

..but you’ll see that I’m cheating a bit by using a very small sample database. Even AdventureWorks produces a big mess.

[](https://www.simple-talk.com/iwritefor/articlefiles/2405-eleventhdot-diagram.png)

Which you can view if you have an image viewer that scrolls!



How did I do this? Just the same way as the first database UML diagrams. I just executed this in SSMS, and copied the result into a text editor, just to put in the top and tail as shown above.

Select distinct

object\_name(parent\_object\_ID) + ' -> '

+ object\_name(referenced\_object\_ID)+';'

FROM sys.foreign\_keys

This can easily be refined into a stored procedure that takes out those illegal characters that make graphviz crash. GraphViz has a lot of power and it is worth looking around for examples and information.

**References**

* <http://www.graphviz.org/> This is the main site for Graphviz. There is plenty of information here, and it is from this site that you can get the GraphViz application and documentation
* <http://plantuml.com/> This is the PlantUML site. It has a large number of resources including downloads and documentation.
* <http://plantuml.com/PlantUML_Language_Reference_Guide.pdf> The essential book of PlantUML. It is much less confusing to read than the PlantUML site that has some distracting google adverts.
* <http://www.graphviz.org/Documentation/dotguide.pdf> The equivalent book of GraphViz. It was written some time ago but still seems relevant.
* <http://asciidocfx.com/> The site for ASCIIdocFX, which is a very cute and useful editor for ASCIIdoc (Markdown for grown-ups) which is intended for creating technical books with mixed text and diagrams. The editor is based on Atom, and you can use PlantUML, GraphViz and DITAA for drawing your diagrams.
* [https://www.dokuwiki.org/dokuwiki#](https://www.dokuwiki.org/dokuwiki) DocuWiki is a wiki that is geared towards technical documentation. The real power of Docuwiki is in the add-ins, one of which gives you PlantUML diagrams embedded in your documentation.
* <http://www.agiledata.org/essays/umlDataModelingProfile.html> A specification for using UML diagrams for databases by Simple-Talk author Scott W Ambler
* <http://plantuml.com/command_line.html> How to use PlantUML from the command line. You’ll need this for when you automate your diagrams with PowerShell

**The Code**

/\* we are just using this for chopping up a space-delimited list of database objects. If you

have embedded spaces in your object names then tough. You'll have to do it differently\*/

if exists (Select \* from sys.xml\_schema\_collections where name like 'ObjectListParameter')

drop XML SCHEMA COLLECTION ObjectListParameter

go

create xml schema collection ObjectListParameter as '

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Object">

<xs:simpleType>

<xs:list itemType="xs:string" />

</xs:simpleType>

</xs:element>

</xs:schema>'

go

/\*

\*/

go

-- does a particular procedure exist

IF EXISTS ( SELECT 1 FROM sys.objects

WHERE object\_ID= object\_ID('dbo.CreatePlantUMLCode'))

SET NOEXEC ON

GO

-- if the routine exists this isn't executed

CREATE PROCEDURE dbo.CreatePlantUMLCode

AS Select 'created, but not implemented yet.'--just anything will do

GO

-- the following section will be always executed

SET NOEXEC OFF

GO

Alter procedure CreatePlantUMLCode

@ObjectsToShow NVarchar(400)=null, -- space-delimited list of database objects

@dependenciesOf NVarchar(400)=null, --show the first order objects that reference or otherwise depend on it

@dependsOn NVarchar(400)=null, --show the objects that it depends on

@MyPlantUMLStatement varchar(max) output --the code to use for the diagram

/\*

Examples of use:

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @ObjectsToShow='HumanResources.employee person.person',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @DependsOn='person.address',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @DependenciesOf='person.address',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @ObjectsToShow='HumanResources.employee person.person HumanResources.vEmployeeDepartmentHistory dbo.ufnGetContactInformation',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

execute CreatePlantUMLCode @ObjectsToShow='dbo.ufnGetContactInformation',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

Declare @ThePlantUMLCode Varchar(max)

execute CreatePlantUMLCode @ObjectsToShow='HumanResources.employee person.person HumanResources.vEmployeeDepartmentHistory ',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

execute CreatePlantUMLCode @ObjectsToShow='HumanResources.vEmployeeDepartmentHistory dbo.ufnGetContactInformation',

@MyPlantUMLStatement=@ThePlantUMLCode output

select @ThePlantUMLCode

\*/

as

--has the user given us a list of objects?

Declare @ObjectsToDo table(Object\_ID int primary key) --check only specified once!

declare @xml\_data xml(ObjectListParameter)

set @xml\_data='<Object>'+ @ObjectsToShow +'</Object>'

Declare @ii int

Select @ii=0 --what has been specified in the parameters

if (@ObjectsToShow is not null) set @ii=@ii+1

if (@DependenciesOf is not null) set @ii=@ii+1

if (@DependsOn is not null) set @ii=@ii+1

if (@ii<1)

begin raiserror('Sorry, but you''ll need to specify what to draw!', 16,1);return;end

if (@ii>1)

begin raiserror('Sorry, only one parameter can be used at a time', 16,1);return;end

/\* if he has given a list, then we need to parse the list and find the IDs of reach

database object that has been specified \*/

if @ObjectsToShow is not null

begin

insert into @ObjectsToDo

select object\_ID(T.ref.value('.', 'sysname'))

from (Select @xml\_data.query('

for $i in data(/Object) return

element item { $i }

')) A(list)

cross apply A.List.nodes('/item') T(ref)

end

--does the user want to see the dependencies

if @DependenciesOf is not null

begin --get all the foreign key references

insert into @ObjectsToDo

Select referenced\_object\_ID

from sys.foreign\_keys

where parent\_object\_id=object\_ID(@DependenciesOf)

union all --and all the objects that refer to it

Select referenced\_ID from sys.sql\_expression\_dependencies

where referencing\_id=object\_ID(@DependenciesOf)

and referenced\_ID is not null

and is\_schema\_bound\_reference =0

union --and insert the object itself

Select object\_ID(@DependenciesOf)

end

if @DependsOn is not null

begin --does the user want a diagram of all the objects that the object depends on?

insert into @ObjectsToDo --insert all the foreign key relationships

Select parent\_object\_ID

from sys.foreign\_keys

where referenced\_Object\_id=object\_ID(@DependsOn)

union all --and all the references this object makes

Select referencing\_ID from sys.sql\_expression\_dependencies

where referenced\_id=object\_ID(@DependsOn)

and is\_schema\_bound\_reference =0

union

Select object\_ID(@DependsOn) --and add the object itself

end

Select @MyPlantUMLStatement='!define table(x) class x << (T,mistyrose) >>

!define view(x) class x << (V,lightblue) >>

!define table(x) class x << (T,mistyrose) >>

!define tr(x) class x << (R,red) >>

!define tf(x) class x << (F,darkorange) >>

!define af(x) class x << (F,white) >>

!define fn(x) class x << (F,plum) >>

!define fs(x) class x << (F,tan) >>

!define ft(x) class x << (F,wheat) >>

!define if(x) class x << (F,gaisboro) >>

!define p(x) class x << (P,indianred) >>

!define pc(x) class x << (P,lemonshiffon) >>

!define x(x) class x << (P,linen) >>

hide methods

hide stereotypes

skinparam classarrowcolor gray

'

/\* firstly, we'll create all the UML table diagrams. \*/

Select @MyPlantUMLStatement=@MyPlantUMLStatement+ 'table('+object\_schema\_name(AllTables.object\_ID)+'.'+Name+') {

' +(Select c.name+': '+t.name+ case when PrimaryKeyColumns.Object\_ID is not null

then ' <<pk>>' else '' end

+ case when fk.parent\_Object\_ID is not null

then ' <<fk>>' else '' end+'

'

from sys.columns c --give the column names and the data types but no dimensions

inner join sys.types t

on c.user\_type\_id= t.user\_type\_id

Left outer join [sys].[foreign\_key\_columns] fk

on parent\_object\_ID=c.object\_ID

and parent\_column\_ID=c.column\_ID

Left outer join --the primary keys are a bit awkward to get

(Select i.object\_ID, column\_ID

from sys.indexes i

inner join sys.index\_columns ic

on ic.object\_ID=i.object\_ID

and ic.index\_ID=i.index\_ID

inner join sys.key\_constraints k

on k.parent\_object\_id=ic.object\_ID

and i.index\_ID=k.unique\_index\_ID

where ic.object\_ID = AllTables.object\_ID

and k.type='pk')PrimaryKeyColumns(Object\_ID,Column\_ID)

on c.object\_ID=PrimaryKeyColumns.Object\_ID

and c.column\_ID=PrimaryKeyColumns.column\_ID

where c.object\_ID = AllTables.object\_ID

FOR XML PATH(''), TYPE).value(N'(./text())[1]',N'varchar(max)')

/\* so now we can add any triggers. We could do indexes as well

but I somehow felt this wasn't appropriate\*/

+ coalesce('\_\_ trigger \_\_

'+ (Select name +'

'from sys.triggers where parent\_ID=AllTables.object\_ID

FOR XML PATH(''), TYPE).value('.', 'varchar(max)'),'')+'}

'

from sys.tables allTables

inner join @ObjectsToDo ObjectsToDo

on alltables.object\_ID = ObjectsTodo.object\_id

/\* now let's do the views \*/

Select @MyPlantUMLStatement=@MyPlantUMLStatement

+ 'view('+object\_schema\_name(Allviews.object\_ID)+'.'+Name+') {

' +(Select c.name+': '+t.name +'

'

from sys.columns c

inner join sys.types t

on c.user\_type\_id= t.user\_type\_id

where c.object\_ID = AllViews.object\_ID

FOR XML PATH(''), TYPE).value(N'(./text())[1]',N'varchar(max)')+'}

'

from sys.views allViews

inner join @ObjectsToDo ObjectsToDo

on allViews.object\_ID = ObjectsTodo.object\_id

/\* now we do anything that is capable of having parameters \*/

Select @MyPlantUMLStatement=@MyPlantUMLStatement+

rtrim(lower(AllRoutines.type))+'('+object\_schema\_name(AllRoutines.object\_ID)+'.'+AllRoutines.Name+') {

'

/\* note, a routine can exist without a parameter \*/

+ coalesce((Select p.name+': '+TYPE\_NAME(p.user\_type\_id) +'

'

FROM sys.objects AS o

INNER JOIN sys.parameters AS p ON o.object\_id = p.object\_id

where o.object\_ID = AllRoutines.object\_ID

FOR XML PATH(''), TYPE).value(N'(./text())[1]',N'varchar(max)'),'')+'}

'

from sys.objects Allroutines

inner join @ObjectsToDo ObjectsToDo

on Allroutines.object\_ID = ObjectsTodo.object\_id

and type in ('AF','FN','FS','FT','IF','P','PC','TF','X');

/\* just the types that can have parameters \*/

/\* and now that we have a class diagram for every object,

we now do the arrows.\*/

Select @MyPlantUMLStatement= @MyPlantUMLStatement+

coalesce((Select

coalesce(object\_schema\_name(referencing\_ID)+'.','')

+ object\_name(referencing\_ID) +' -|> '

+ referenced\_schema\_name+'.'+Referenced\_Entity\_name

+ ':References

'

--AS reference

FROM sys.sql\_expression\_dependencies

inner join @ObjectsToDo ObjectsToDo

on referencing\_ID = ObjectsTodo.object\_id

inner join @ObjectsToDo ObjectsToDo2

on referenced\_ID = ObjectsTodo2.object\_id

where is\_schema\_bound\_reference =0

FOR XML PATH(''), TYPE).value(N'(./text())[1]',N'varchar(max)'),'')

Select @MyPlantUMLStatement= @MyPlantUMLStatement+

coalesce((Select Object\_schema\_name(parent\_object\_ID)+'.'+object\_name(parent\_object\_ID)+' -|> '

+ Object\_schema\_name(referenced\_object\_ID)+'.'+object\_name(referenced\_object\_ID)+':FK

'

from sys.foreign\_keys

inner join @ObjectsToDo ObjectsToDo

on parent\_object\_ID = ObjectsTodo.object\_id

inner join @ObjectsToDo ObjectsToDo2

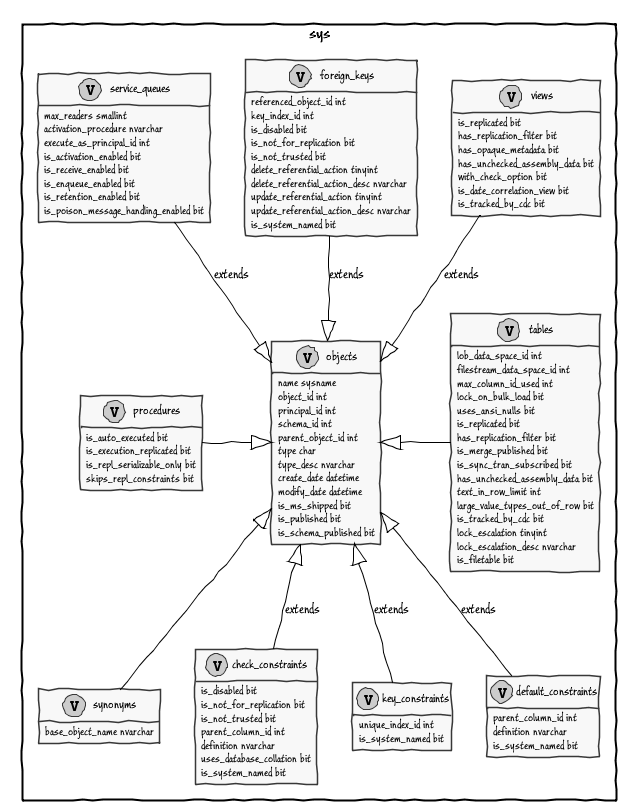
on referenced\_object\_ID = ObjectsTodo2.object\_id

FOR XML PATH(''), TYPE).value(N'(./text())[1]',N'varchar(max)'),'')

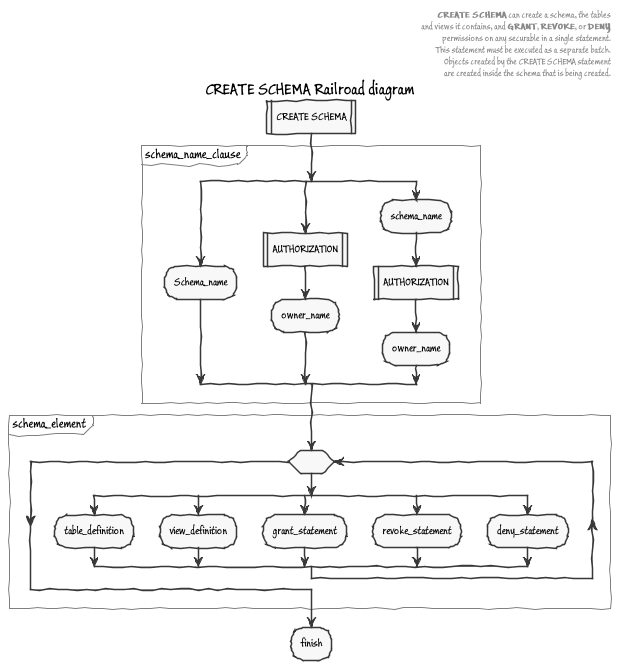
go

**The Coda**

Just to show that, with some care with the way that you do the arrows, you can get some very satisfactory effects.



... and PlantUML can even do syntax railroad diagrams for SQL.



Thank this author by sharing:

[8](https://www.simple-talk.com/sql/sql-tools/automatically-creating-uml-database-diagrams-for-sql-server/)

This article has been viewed 6047 times.

[[](https://www.simple-talk.com/author/phil-factor/)](https://www.simple-talk.com/author/phil-factor/)

**Author profile:** [Phil Factor](https://www.simple-talk.com/author/phil-factor/)

Phil Factor (real name withheld to protect the guilty), aka Database Mole, has 30 years of experience with database-intensive applications. Despite having once been shouted at by a furious Bill Gates at an exhibition in the early 1980s, he has remained resolutely anonymous throughout his career. See also :

* [The Phrenetic Phoughts of Phil Factor](http://www.simple-talk.com/community/blogs/philfactor/default.aspx)
* [Phil on Twitter](http://twitter.com/Phil_Factor)
* [Phil on SQL Server Central](http://www.sqlservercentral.com/blogs/philfactor/)
* [Robyn and Phil's Workbenches](http://www.simple-talk.com/author/robyn-page-and-phil-factor/)
* [Phil's Editorials on SQL Server Central](http://www.sqlservercentral.com/Authors/Articles/Phil_Factor/322525/)

[Google +](https://plus.google.com/106425589704685208213?rel=author) To translate this article...

[Search for other articles by Phil Factor](https://www.simple-talk.com/author/phil-factor/)

Top of Form



**Rate this article:**   Avg rating: https://www.simple-talk.com/images/interface/icon_plain_5stars.giffrom a total of 32 votes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Poor | OK | Good | Great | Must read |  |

Have Your Say

Do you have an opinion on this article? Then [add your comment below](https://www.simple-talk.com/sql/sql-tools/automatically-creating-uml-database-diagrams-for-sql-server/#commentform):

You must be logged in to post to this forum  
  
[Click here to log in](https://www.simple-talk.com/community/login.aspx?ReturnUrl=%2fsql%2fsql-tools%2fautomatically-creating-uml-database-diagrams-for-sql-server%2f).

|  |  |
| --- | --- |
| **Subject:** | **Fantastic** |
| **Posted by:** | *Sheldon* (not signed in) |
| **Posted on:** | *Tuesday, May 10, 2016 at 2:22 PM* |
| **Message:** | This is awesome! Going to try this out for sure. I did some work on diagramming in this post, but required manual work. I like the potential this offers for automated diagramming.  http://sheldonhull.com/documenting-your-database-with-diagrams/  The problem is that diagramming becomes more manual work, easy to get out of sync, whereas this approach offers ab automatedv approach. I wish graphviz was this intuitive |

|  |  |
| --- | --- |
| **Subject:** | **Wonderful tool** |
| **Posted by:** | *sterbalr* ([view profile](https://www.simple-talk.com/community/user/Profile.aspx?UserID=213351)) |
| **Posted on:** | *Monday, May 23, 2016 at 1:52 PM* |
| **Message:** | Thanks for posting this. I think I'll have a lot of use for this going forward.  I've moved my wikis to MediaWiki once I found a one click installer at Bitnami:  https://bitnami.com/stack/mediawiki  (running two types of wikis became too much of a drag) |

|  |  |
| --- | --- |
| **Subject:** | **Very useful tool** |
| **Posted by:** | *DMaletin* ([view profile](https://www.simple-talk.com/community/user/Profile.aspx?UserID=32087)) |
| **Posted on:** | *Friday, May 27, 2016 at 2:35 AM* |
| **Message:** | I've been used it for latest couple of years.  Combined with other tools it allows to generate nice documentation. |

|  |  |
| --- | --- |
| **Subject:** | **Very useful tool** |
| **Posted by:** | *DMaletin* ([view profile](https://www.simple-talk.com/community/user/Profile.aspx?UserID=32087)) |
| **Posted on:** | *Friday, May 27, 2016 at 4:43 AM* |
| **Message:** | I've been used it for latest couple of years.  Combined with other tools it allows to generate nice documentation. |

Bottom of Form