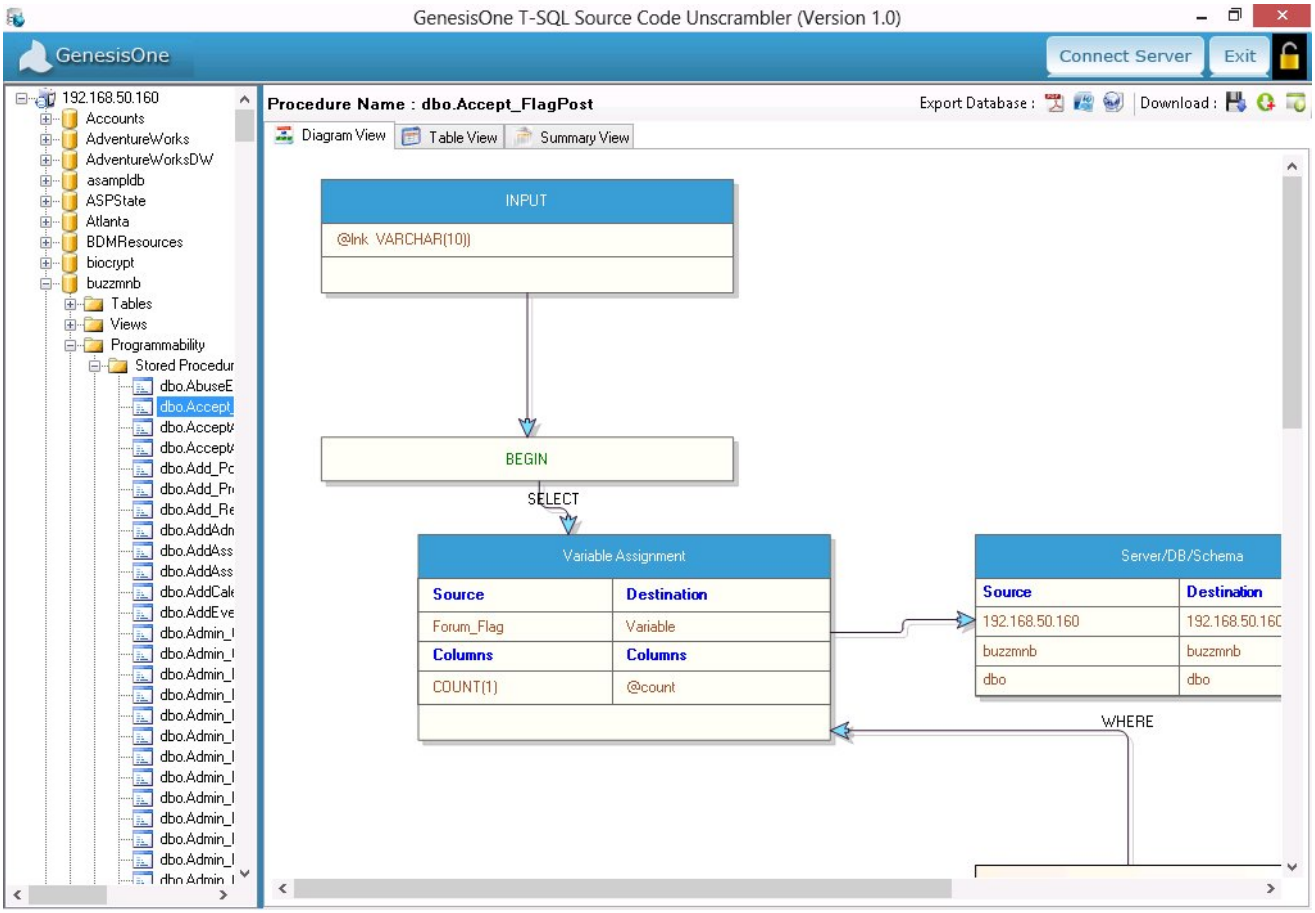
GenesisOne Use Case Example

Larry has just been brought on to ABC company as a database developer to take over advanced development and maintenance of the database side of a major internal application. The database is quite large and the company is already behind schedule after the database developer who built the system suddenly resigned and left prior to Larry's arrival. The documentation on the database is very thin and what is there is out-of-date.

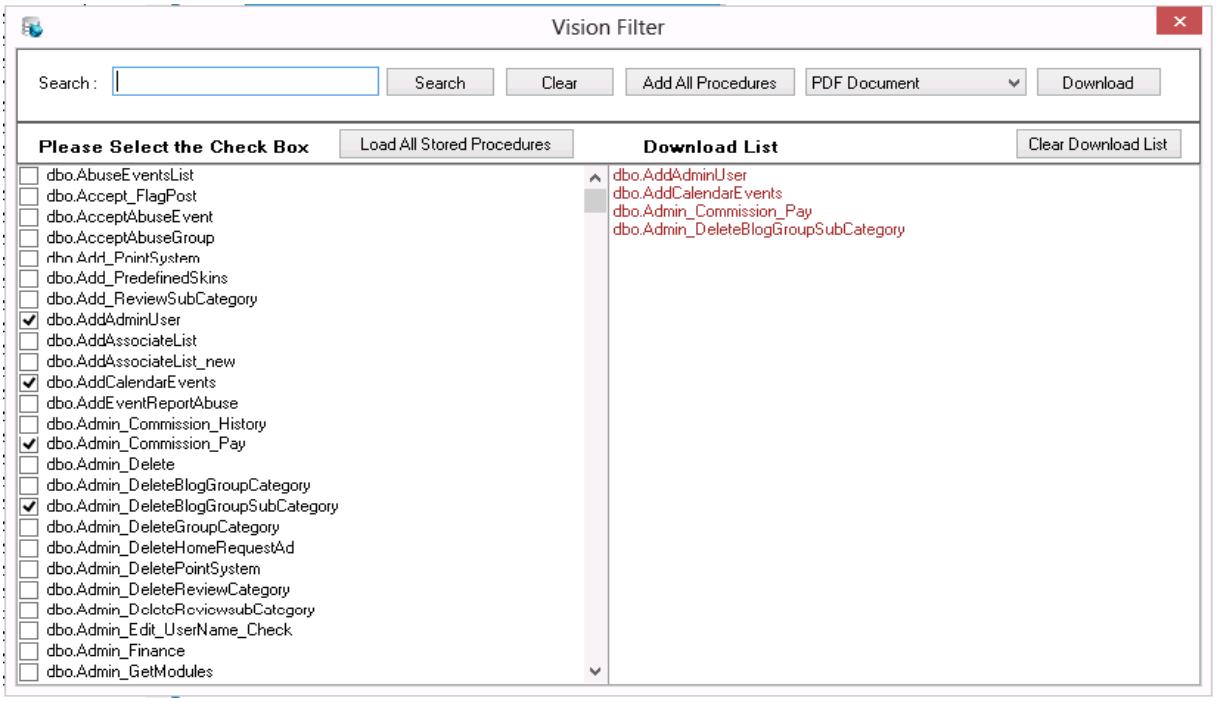
Faced with a rapid learning curve and no one to assist, Larry knows he can't spend the time he wants reading through code, so he launches his GenesisOne Source Code Unscrambler™ to get a visual representation on what the code is doing.

After logging in to the server, he starts exploring the objects one at a time to start getting a good understanding

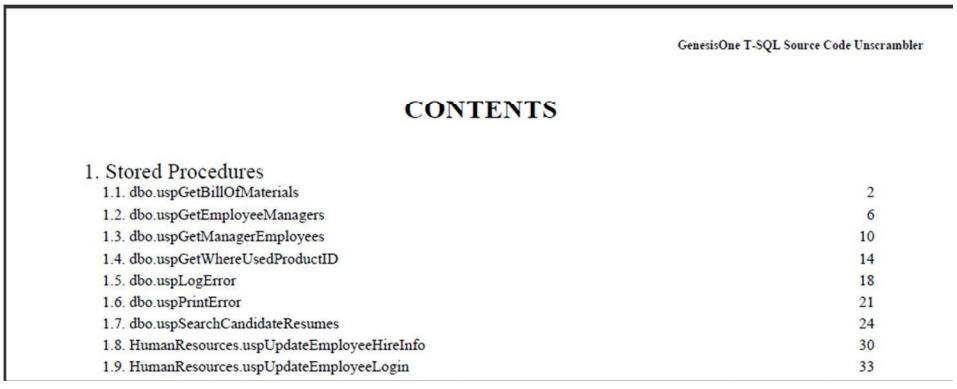


There are quite a few, so rather than take the piecemeal approach, he decides to view all the objects and put the diagrams and supporting information in a repository where both he and the DBA's can access and study any of the objects.

By clicking on the “Download List” button, Larry is then able to select all the objects for which he wants to generate documentation and chooses the output format of PDF, then clicks “Download”.



In less than a minute, Larry now has professional-grade documentation that can be used by developers, managers, operations personnel and anyone else who wants to understand the code logic without reading the code. The documentation includes a table of contents and data flow diagrams, English language explanation of each selected code object.



GenesisOne T-SQL Source Code Unscrambler

CONTENTS

Summary Definition :

Step : 1

This Procedure has @StartProductID - INT, @CheckDate - DATETIME as Inputs

Step : 2

It Selects ProductAssemblyID,ComponentID,PerAssemblyQty,BOMLevel,0 Column(s) from the table BillOfMaterials and insert the result into the CTE - table [BOM_cte]. The CTE has the following columns ProductAssemblyID,ComponentID,ComponentDesc,PerAssemblyQty,StandardCost,ListPrice,BOMLevel,RecursionLevel where the table BillOfMaterials satisfy the following condition BillOfMaterials.ProductAssemblyID = @StartProductID AND @CheckDate >= BillOfMaterials.StartDate AND @CheckDate and it inner join with the table Product on BillOfMaterials.ComponentID = Product.ProductID. The used columns from this joined table are Name,StandardCost,ListPrice

Step : 3

Then UNION ALL

Step : 4

It Selects RecursionLevel+ 1 Column(s) from the table BOM_cte and insert the result into the CTE - table where the table BOM_cte satisfy the following condition @CheckDate >= BillOfMaterials.StartDate AND @CheckDate and it inner join with the table BillOfMaterials on BillOfMaterials.ProductAssemblyID = BOM_cte.ComponentID. The used columns from this joined table are ProductAssemblyID,ComponentID,PerAssemblyQty,BOMLevel and it inner join with the table Product on BillOfMaterials.ComponentID = Product.ProductID. The used columns from this joined table are Name,StandardCost,ListPrice

As is always the case, Larry not only has to learn the code in the database, but he has also been tasked with making some programming changes and eliminating unnecessary objects.

After completing his assigned tasks, Larry starts looking through the data and notices that one of the columns in a table has started showing some odd data that doesn't quite make any sense. To find out why, he first has to know what code is either inserting or updating that column. Larry goes to the main screen on the console and clicks "Show XML Batcher". This then brings up a screen similar to the download screen where he can select all the objects in the database for analysis.

By searching the XML for DestinationTable/DestinationColumn, Larry is able to isolate the exact code objects that either insert into or update the column. In knowing this, he is also able to determine what code is acting on this column and make the necessary changes to correct it.