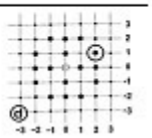
Softworkbench.SPGenerator

User Manual

Stored procedures generator

Softworkbench, Inc.

www.softworkbench.com



Softworkbench, Inc.

www.softworkbench.com

Table of Contents

Introduction

Overview

This guide provides information on the Softworkbench.SPGenerator architecture and working with the stored procedures generator.

Audience

The intended audience for this guide is database application developers and DBAs.

Document organization

This document is divided into the following chapters:

- Chapter 1 provides an overview of the Softworkbench.SPGenerator.
- Chapter 2 provides information on working Softworkbench.SPGenerator.

Chapter 1. Overview of SoftworkbenchSPGenerator

Softworkbench.SPGenerator is a XML/XSLT template-based stored procedures generator that you use to generate basic CRUD stored procedures.

This chapter provides an overview of Softworkbench.SPGenerator, including:

- What is SoftworkbenchSPGenerator?
- What can Softworkbench.SPGenerator do?
- Architecture overview of Softworkbench.SPGenerator.

1. 1 What is the Softworkbench.SPGenerator?

Softworkbench.SPGenerator is a stored procedure generator that supports Microsoft SQL Server and Oracle. Softworkbench.SPGenerator will process each table and view in the database which you selected and created stored procedures to insert, update, delete, select (single or multiple records) using the fields you choose as well as using stored procedure name you desire.

Softworkbench.SPGenerator supports the rapid development of database stored procedures by using XML as metadata input and XML Style sheet Language Transformations (XSLT) as syntax-tree scripting mechanism, this XML/XSLT – base code generation approach give you a great amount of feasibility to customize the output format, you can modify the XSLT script to meet your special requirement.

Below is how it works:

- Collect metadata from different data source (Oracle and SQL Server)
- Build xml metadata input definition file
- Generate stored procedures using xslt with different xsl stlysheets.

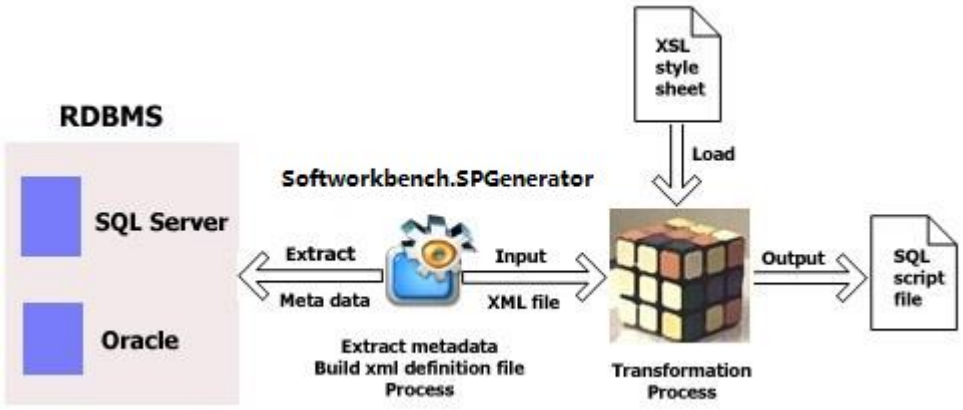
1. 2 What the Softworkbench.SPGenerator can do

Using Softworkbench.SPGenerator, you can performance the following tasks:

- Generate stored procedures to insert data rows and return identity id for SQL server and sequence number for Oracle.
- Generate stored procedures to select, update, delete single data row based on primary key definitions.
- Generate stored procedures to select all rows based on foreign key definitions.
- Generate stored procedures to select all rows of tables and views.
- Generate one big sql script file or each file per table
- Generate XML metadata definition file
- Control output stored procedures format in runtime by modifying XSL style sheet.
- Control output stored procedure's name and columns in the run-time.
- Generate package header and body for Oracle.
- Support composite primary key
- Support Windows Authentication and SQL Server Authentication

1. 3 Architecture overview of Softworkbench.SPGenerator

Softworkbench.SPGenerator stored procedures generator consists of a number of distinct processing phases as show in the following figure.



Collect metadata process Softworkbench.SPGenerator use a powerful database meta-data API collecting metadata from different database management system, such as MS SQL server, Oracle and store in the in-memory database object for building xml definition file process.

Build XML metadata definition file process Softworkbench.SPGenerator process each table and view you selected and build the XML definition file as input for XSLT transformation process.

XSLT Transform process Softworkbench.SPGenerator load the XSL style sheet, read the source XML definition file to produce the output source SQL script file and write it to the file system.

Chapter 2. Get Started

Softworkbench.SPGenerator will process each table and view in the database which you selected and output SQL script files per table and view or one big file per database.

You can find sql scripts example for MS SQL Server and Oracle in Appendix A and B.

Before You Begin

Download SPGenerator.zip from www.softworkbench.com.

This chapter provides information on working with Softworkbench.SPGenerator, including:

- Prerequisites
- General information
- Walk through
- Compile SQL script files

2. 1 Prerequisites

Ensure the following prerequisites are met prior to installing the Softworkbench.SPGenerator:

- Microsoft .NET framework 2.0 or later version is required.

2.2 General information

Softworkbench.SPGenerator will process each table and view in the database which is selected in wizard step 4. Per table and view, each field definition is read and stored in a ColumnCollection object. Softworkbench.SPGenerator tries to determine if a field is part of the Primary Key, if it's a Foreign Key, or if it has a UNIQUE constraint. This information, besides the type, length, precision and other field information, is used to determine which stored procedures should be generated and which fields should be passed as parameters.

Softworkbench.SPGenerator wizard is very user-friendly and takes you five steps to generate all necessary insert, update, delete, select (single or multiple records) stored procedures for MS SQL Server and ORACLE. Unlike many other stored procedures generator, Softworkbench.SPGenerator let you select tables or views, change each stored procedure's name, exclude columns you do not need from stored procedure you selected.

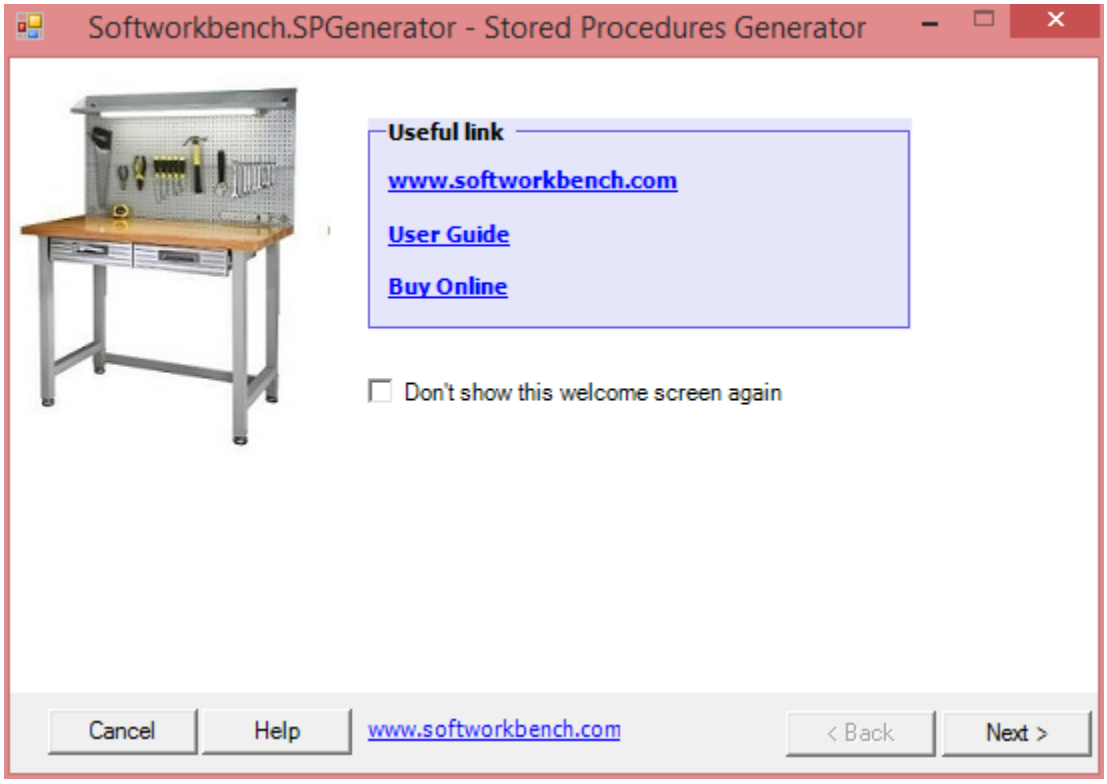
2.3 Walk through

Softworkbench.SPGenerator uses a Configuration.xml file to store database server name, database name, login name, password, stored procedure's prefix and suffix as well as format of stored procedure name. You can edit this xml file to meet your requirement before run the Softworkbench.SPGenerator. For example, you can change the "Insert" procedure format as "Insert_%tableName%, so the name of insert procedure will look like "Insert_Employee" instead of "Employee_Insert".

Here is Configuration.xml file layout:

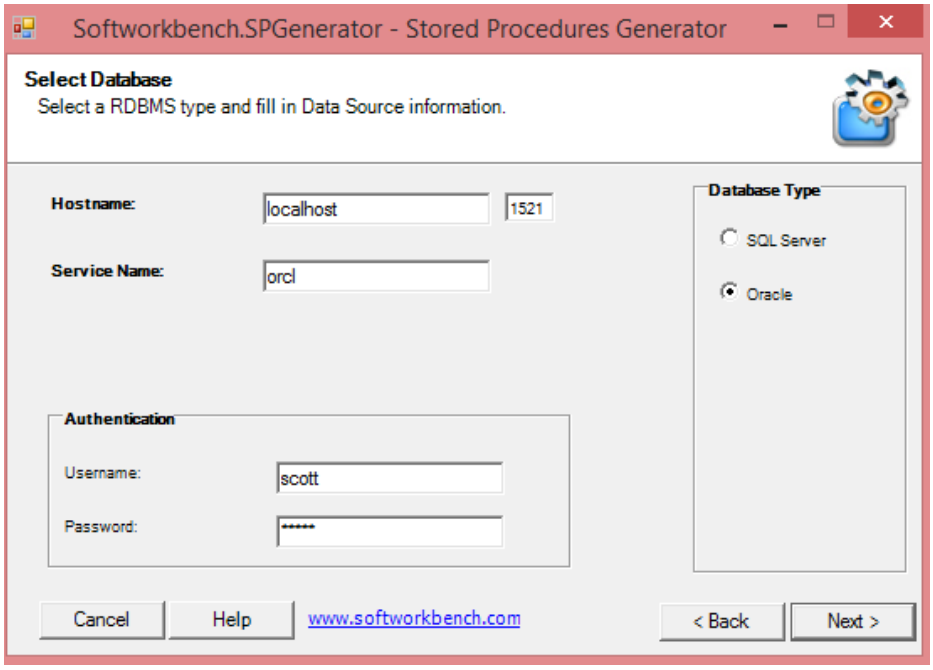
```
<?xml version="1.0"?>
<databases>
  <database type="SQLServer">
    <servername>localhost</servername>
    <databasename>Northwind</databasename>
    <loginname></loginname>
    <password></password>
    <windowsauthentication>true</windowsauthentication>
    <prefix>proc_</prefix>
    <suffix>_suffix</suffix>
    <procedure type="Insert">%tableName%_Insert</procedure>
    <procedure type="Update">%tableName%_Update</procedure>
    <procedure type="Delete">%tableName%_Delete</procedure>
    <procedure type="Read">%tableName%_GetSingle</procedure>
    <procedure type="ReadAll">%tableName%_GetAll</procedure>
    <procedure type="ReadAllByFK">%tableName%_GetAllBy_%columnName%</procedure>
    <procedure type="View">View_%tableName%_GetAll</procedure>
  </database>
  <database type="Oracle">
    <servername>localhost</servername>
    <port>1521</port>
    <databasename>orcl</databasename>
    <loginname>scott</loginname>
    <password>tiger</password>
    <prefix>proc_</prefix>
    <suffix>suffix</suffix>
    <procedure type="Insert">prInsert</procedure>
    <procedure type="Update">prUpdate</procedure>
    <procedure type="Delete">prDelete</procedure>
    <procedure type="Read">prGetSingle</procedure>
    <procedure type="ReadAll">prGetAll</procedure>
    <procedure type="ReadAllByFK">prGetAllBy_%columnName%</procedure>
    <procedure type="View">vwGetAll</procedure>
  </database>
</databases>
```

Step 1: Welcome screen

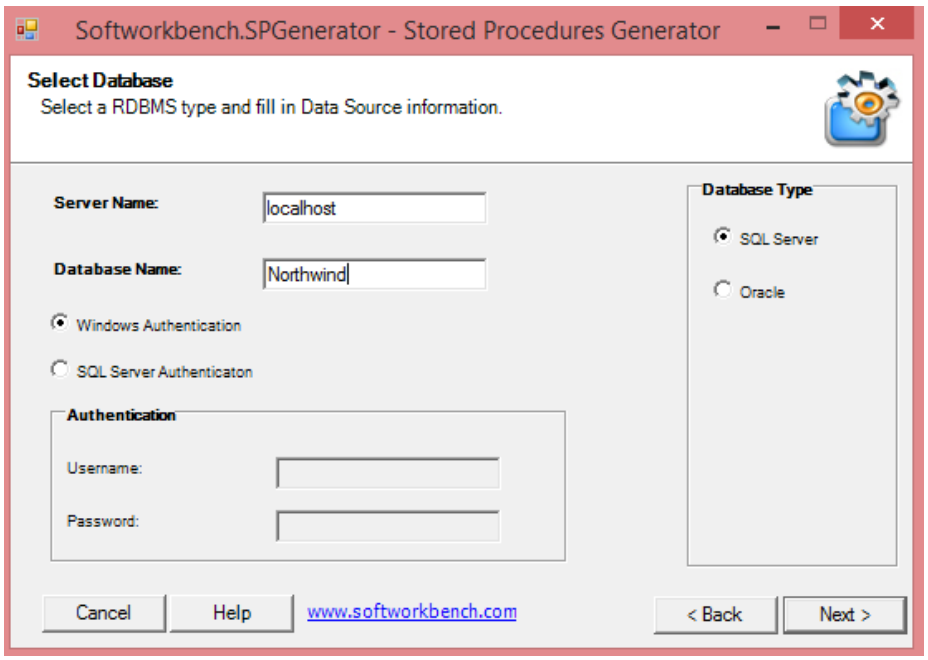


- This step introduces you to the wizard and some useful link
- Check “Don’t show this welcome screen again” check box will not show this welcome screen when you select re-run wizard in last step.
- Click Next button go to next step

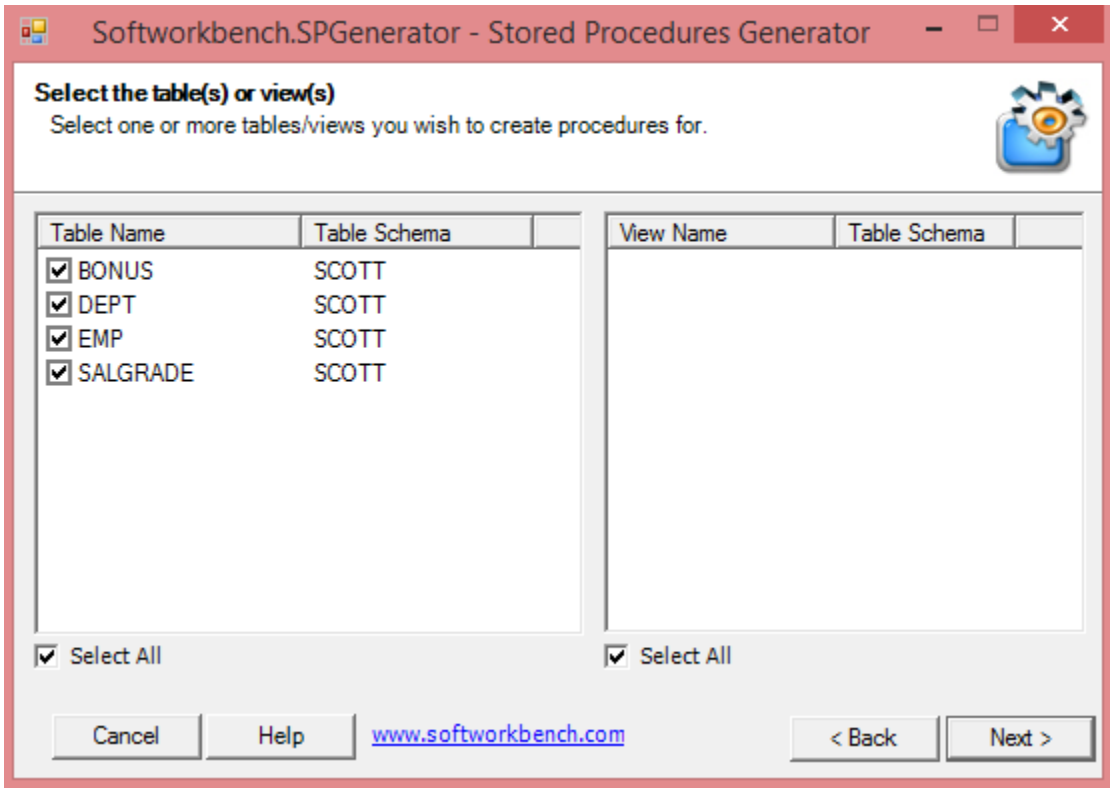
Step 2: Select database screen



- This step let you select RDBMS.
- Option to use either Windows Authentication or SQL server Authentication
- Click Next button to go to next step.

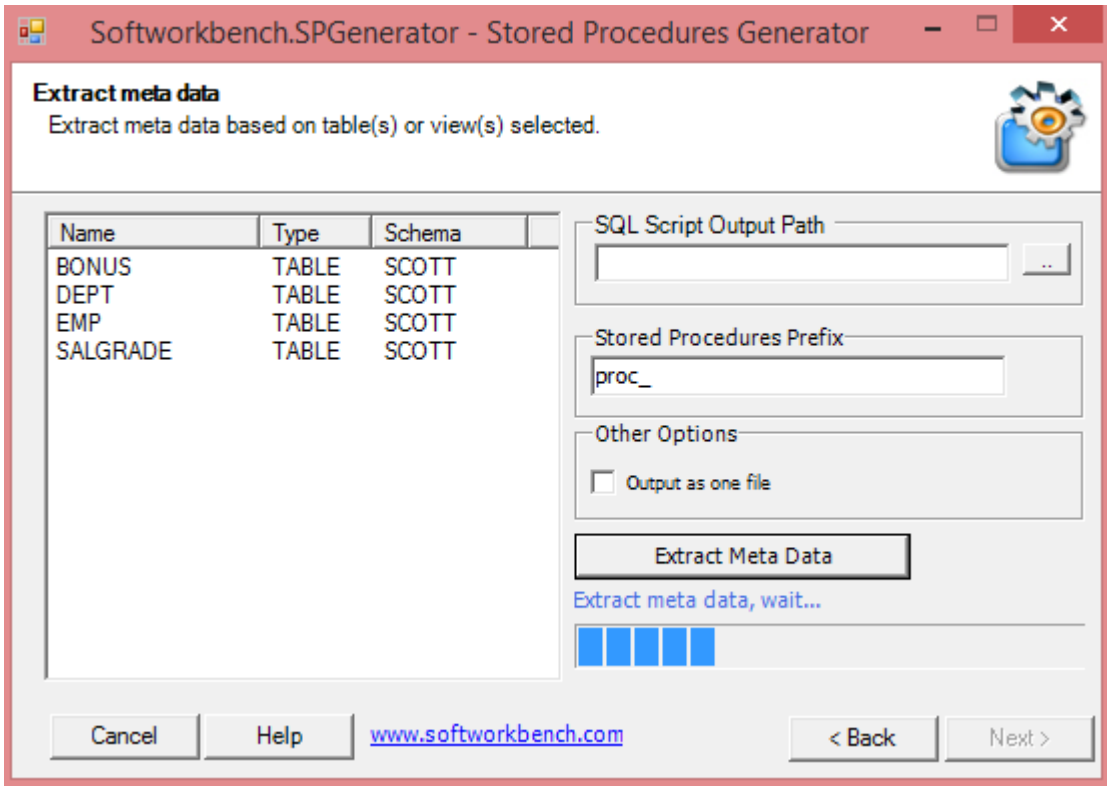


Step 3: Select table(s) and view(s)



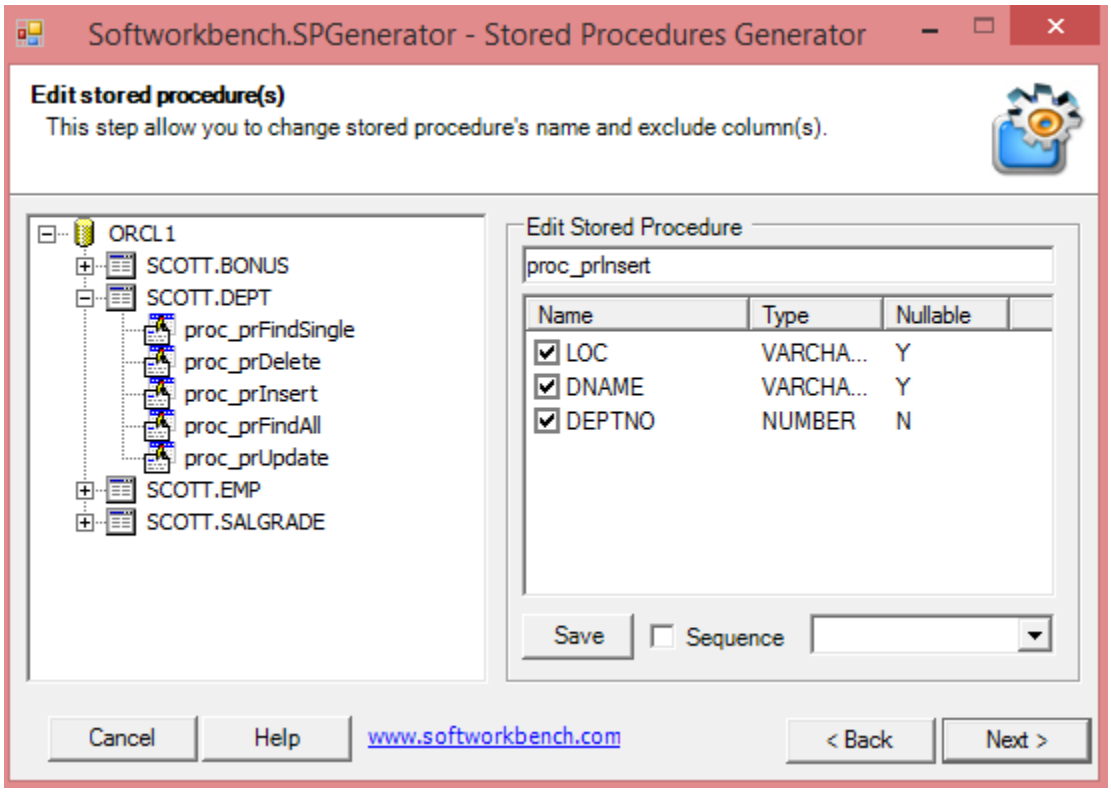
- This step let you select the table(s) and view(s).
- Check “Select All” check box to select all tables or views.
- Uncheck “Select All” check box to un-select all tables or views.
- Exclude table or views by uncheck it from list view.
- Click Next button go to next step.

Step 4: Extract Meta data screen



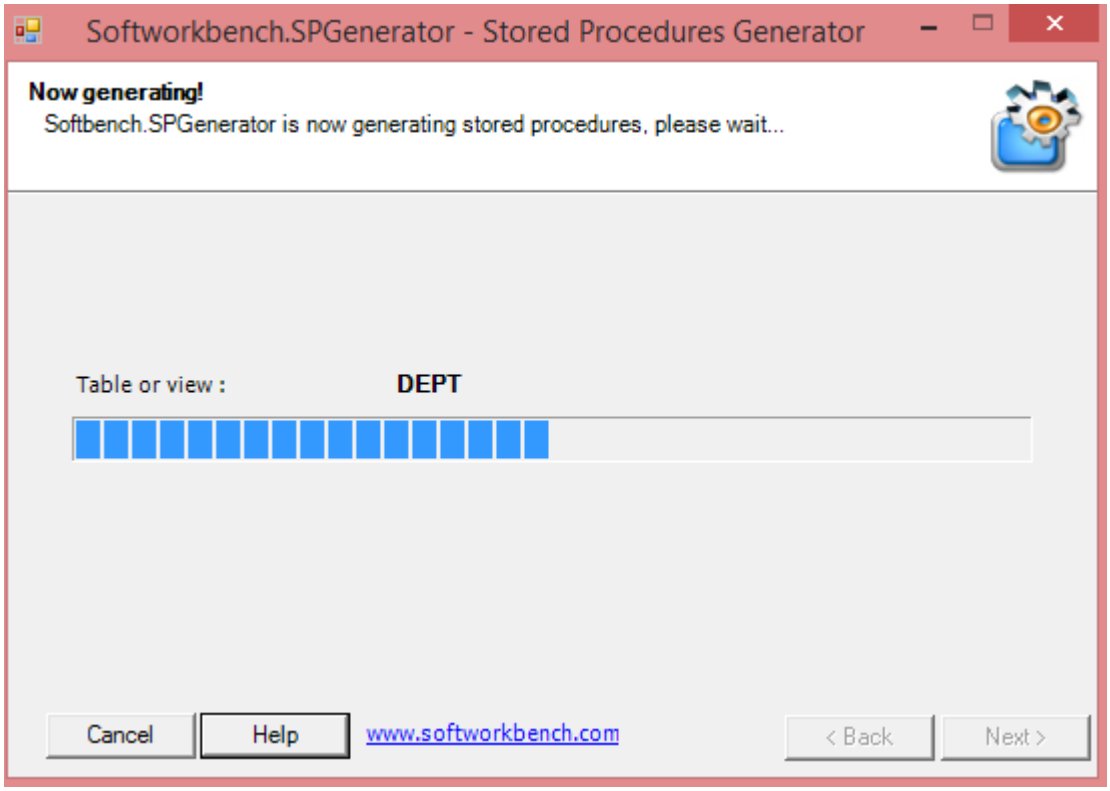
- This step extract metadata for all tables and views show on the left.
- Fill stored procedures prefix text field if you need it
- Click browser button to pick up SQL script output directory
- Option to output generated script in one big file or one file per table
- Click Extract Meta Data to begin extract metadata from database, when finish it will go to next step.

Step 5: Edit stored procedure screen



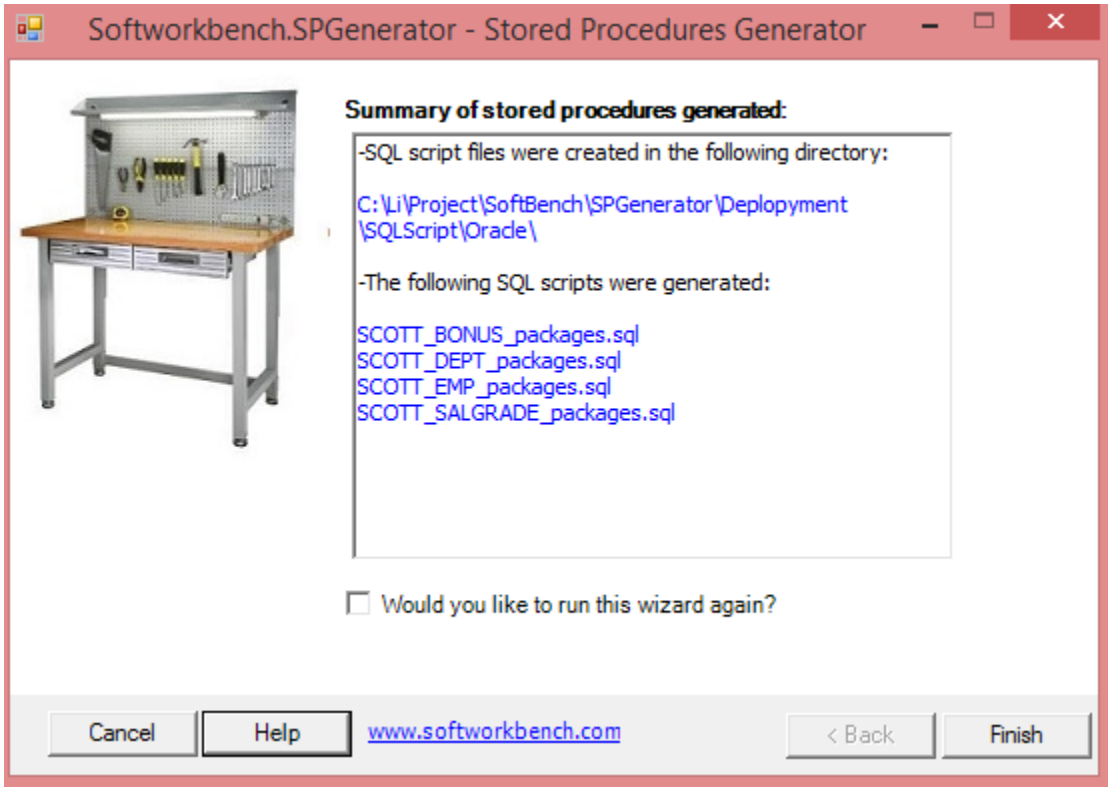
- This step let you change stored procedure's name and exclude columns.
- Select the stored procedure in the left hand tree view, change the stored procedure's name and exclude columns by uncheck them from list view in the right hand, then click Save button.
- For Oracle, check Sequence checkbox and select sequence name from dropdown box, Softworkbench.SPGenerator will generate code to get sequence number and return it as out put parameter. (Note: you need create sequence first).
- Click Next button to go to next step.

Step 6: Generating screen



- This step shows the process generating the stored procedures.
- It will go to next step when finished

Step 7: Completing screen



- This step show you the location and SQL script files generated.
- Click Finish button to complete the generating the stored procedures.
- Softworkbench.SPGenerator wizard will run again, if you checked “Would you like to run this wizard again?” checkbox.

APPENDIX A: SQL SERVER SQL SCRIPT SAMPLE (NORTHWIND)

```
-------------------------------------------------------------------------------------------------
-------------
--Stored procedures were auto-generated by Softworkbench.SPGenerator v1.1
--Generated time: Thursday, May 12, 2015, 9:36:12 PM
--Web: http://www.softworkbench.com
--Email: info@softworkbench.com
-------------------------------------------------------------------------------------------------
-------------

GO
USE [northwind]
GO

--================================================================
--[Stored Procedure generated for table: Employees]


-- //// Update Stored Procedure based on primary keys
if exists(select * from dbo.sysobjects where id = object_id(N'[dbo].[Update_Employees]')
and OBJECTPROPERTY(id, N'IsProcedure') = 1)
       drop procedure [dbo].[Update_Employees]
GO

CREATE PROCEDURE [dbo].[Update_Employees]
       @EmployeeID int,
       @LastName nvarchar(40),
       @FirstName nvarchar(20),
       @Title nvarchar(60),
       @TitleOfCourtesy nvarchar(50),
       @BirthDate datetime,
       @HireDate datetime,
       @Address nvarchar(120),
       @City nvarchar(30),
       @Region nvarchar(30),
       @PostalCode nvarchar(20),
       @Country nvarchar(30),
       @HomePhone nvarchar(48),
       @Extension nvarchar(8),
       @Photo image,
       @Notes ntext,
       @ReportsTo int,
       @PhotoPath nvarchar(510)
AS
SET NOCOUNT ON

UPDATE  [Employees]
SET

       [LastName] = @LastName,
       [FirstName] = @FirstName,
       [Title] = @Title,
       [TitleOfCourtesy] = @TitleOfCourtesy,
       [BirthDate] = @BirthDate,
       [HireDate] = @HireDate,
       [Address] = @Address,
       [City] = @City,
       [Region] = @Region,
       [PostalCode] = @PostalCode,
       [Country] = @Country,
       [HomePhone] = @HomePhone,
       [Extension] = @Extension,
       [Photo] = @Photo,
       [Notes] = @Notes,
       [ReportsTo] = @ReportsTo,
```

```
	[PhotoPath] = @PhotoPath
WHERE
	[EmployeeID] = @EmployeeID

GO

-- //// Delete Stored Procedure based on primary keys
if exists(select * from dbo.sysobjects where id = object_id(N'[dbo].[Delete_Employees]')
and OBJECTPROPERTY(id, N'IsProcedure') = 1)
	drop procedure [dbo].[Delete_Employees]
GO

CREATE PROCEDURE [dbo].[Delete_Employees]
	@EmployeeID int
AS
SET NOCOUNT ON

DELETE FROM [Employees]
WHERE
	[EmployeeID] = @EmployeeID
GO

-- //// Select All Stored Procedures
if exists(select * from dbo.sysobjects where id = object_id(N'[dbo].[ReadAll_Employees]')
and OBJECTPROPERTY(id, N'IsProcedure') = 1)
	drop procedure [dbo].[ReadAll_Employees]
GO

CREATE PROCEDURE [dbo].[ReadAll_Employees]
AS
SET NOCOUNT ON

SELECT
		EmployeeID,
		LastName,
		FirstName,
		Title,
		TitleOfCourtesy,
		BirthDate,
		HireDate,
		Address,
		City,
		Region,
		PostalCode,
		Country,
		HomePhone,
		Extension,
		Photo,
		Notes,
		ReportsTo,
		PhotoPath
FROM Employees
GO

-- //// Read Stored Procedure based on primary keys
if exists(select * from dbo.sysobjects where id = object_id(N'[dbo].[Read_Employees]')
and OBJECTPROPERTY(id, N'IsProcedure') = 1)
	drop procedure [dbo].[Read_Employees]
GO

CREATE PROCEDURE [dbo].[Read_Employees]
	@EmployeeID int
AS
SET NOCOUNT ON

SELECT
	[EmployeeID],
	[LastName],
	[FirstName],
	[Title],
	[TitleOfCourtesy],
```

```
        [BirthDate],
        [HireDate],
        [Address],
        [City],
        [Region],
        [PostalCode],
        [Country],
        [HomePhone],
        [Extension],
        [Photo],
        [Notes],
        [ReportsTo],
        [PhotoPath]
FROM [Employees]
WHERE
        [EmployeeID] = @EmployeeID
GO

-- //// Read all Stored Procedure based on foreign key
if exists(select * from dbo.sysobjects where id =
object_id(N'[dbo].[ReadALL_BY_ReportsTo_Employees]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
        drop procedure [dbo].[ReadALL_BY_ReportsTo_Employees]
GO

CREATE PROCEDURE [dbo].[ReadALL_BY_ReportsTo_Employees]
        @ReportsTo int
AS
SET NOCOUNT ON

SELECT
        [EmployeeID],
        [LastName],
        [FirstName],
        [Title],
        [TitleOfCourtesy],
        [BirthDate],
        [HireDate],
        [Address],
        [City],
        [Region],
        [PostalCode],
        [Country],
        [HomePhone],
        [Extension],
        [Photo],
        [Notes],
        [ReportsTo],
        [PhotoPath]
FROM [Employees]
WHERE   [ReportsTo] = @ReportsTo
GO


-- //// Insert Stored Procedure
if exists(select * from dbo.sysobjects where id = object_id(N'[dbo].[Insert_Employees]')
and OBJECTPROPERTY(id, N'IsProcedure') = 1)
        drop procedure [dbo].[Insert_Employees]
GO

CREATE PROCEDURE [dbo].[Insert_Employees]
        @EmployeeID int output,
        @LastName nvarchar(40),
        @FirstName nvarchar(20),
        @Title nvarchar(60),
        @TitleOfCourtesy nvarchar(50),
        @BirthDate datetime,
        @HireDate datetime,
        @Address nvarchar(120),
        @City nvarchar(30),
        @Region nvarchar(30),
```

```
        @PostalCode nvarchar(20),
        @Country nvarchar(30),
        @HomePhone nvarchar(48),
        @Extension nvarchar(8),
        @Photo image,
        @Notes ntext,
        @ReportsTo int,
        @PhotoPath nvarchar(510)
AS
SET NOCOUNT ON

INSERT INTO  [Employees]
(
                        [LastName],
                        [FirstName],
                        [Title],
                        [TitleOfCourtesy],
                        [BirthDate],
                        [HireDate],
                        [Address],
                        [City],
                        [Region],
                        [PostalCode],
                        [Country],
                        [HomePhone],
                        [Extension],
                        [Photo],
                        [Notes],
                        [ReportsTo],
                        [PhotoPath]
)
VALUES
(
                        @LastName,
                        @FirstName,
                        @Title,
                        @TitleOfCourtesy,
                        @BirthDate,
                        @HireDate,
                        @Address,
                        @City,
                        @Region,
                        @PostalCode,
                        @Country,
                        @HomePhone,
                        @Extension,
                        @Photo,
                        @Notes,
                        @ReportsTo,
                        @PhotoPath
)

SET @EmployeeID = @@IDENTITY

GO


--[End of Stored Procedure for table: Employees]
--================================================================
```

APPENDIX B: ORACLE SQL SCRIPT SAMPLE (SCOTT)

```
-------------------------------------------------------------------------------------------------
-------------
--Package was auto-generated by Softworkbench.SPGenerator v1.1
--Generated time: Friday, May 12, 2015, 3:33:27 PM
--Web: http://www.softworkbench.com
--Email: info@softworkbench.com
-------------------------------------------------------------------------------------------------
-------------

--============================================================
--[Package generated for table: EMP]

--Begin Package header
CREATE OR REPLACE PACKAGE EMP_PKG AS

        TYPE cursor_type IS REF CURSOR;
        PROCEDURE Insert_EMP
        (
                p_EMPNO OUT EMP.EMPNO%TYPE,
                p_ENAME IN EMP.ENAME%TYPE,
                p_JOB IN EMP.JOB%TYPE,
                p_MGR IN EMP.MGR%TYPE,
                p_HIREDATE IN EMP.HIREDATE%TYPE,
                p_SAL IN EMP.SAL%TYPE,
                p_COMM IN EMP.COMM%TYPE,
                p_DEPTNO IN EMP.DEPTNO%TYPE
        );
        PROCEDURE Update_EMP
        (
                p_EMPNO IN EMP.EMPNO%TYPE,
                p_ENAME IN EMP.ENAME%TYPE,
                p_JOB IN EMP.JOB%TYPE,
                p_MGR IN EMP.MGR%TYPE,
                p_HIREDATE IN EMP.HIREDATE%TYPE,
                p_SAL IN EMP.SAL%TYPE,
                p_COMM IN EMP.COMM%TYPE,
                p_DEPTNO IN EMP.DEPTNO%TYPE
        );
        PROCEDURE ReadAll_EMP
        (
                p_cur OUT cursor_type
        );
        PROCEDURE Delete_EMP
        (
                p_EMPNO IN EMP.EMPNO%TYPE
        );
        PROCEDURE ReadALL_BY_DEPTNO_EMP
        (
                p_cur OUT cursor_type,
                p_DEPTNO IN EMP.DEPTNO%TYPE

        );
        PROCEDURE Read_EMP
        (
                p_cur OUT cursor_type,
                p_EMPNO IN EMP.EMPNO%TYPE
        );

END EMP_PKG;
--End Package header
/

--Begin Package body
CREATE OR REPLACE PACKAGE BODY EMP_PKG AS

        -- //// Insert Stored Procedure
```

```
        PROCEDURE Insert_EMP
        (

                p_EMPNO OUT EMP.EMPNO%TYPE,
                p_ENAME IN EMP.ENAME%TYPE,
                p_JOB IN EMP.JOB%TYPE,
                p_MGR IN EMP.MGR%TYPE,
                p_HIREDATE IN EMP.HIREDATE%TYPE,
                p_SAL IN EMP.SAL%TYPE,
                p_COMM IN EMP.COMM%TYPE,
                p_DEPTNO IN EMP.DEPTNO%TYPE
        )
        AS
                num_EMPNO NUMBER;
        BEGIN
                select EMPNO_SEQ.NEXTVAL
                into num_EMPNO
                from DUAL;

                insert into EMP
                (
                        EMPNO,
                        ENAME,
                        JOB,
                        MGR,
                        HIREDATE,
                        SAL,
                        COMM,
                        DEPTNO
                )
                values
                (

                        num_EMPNO,
                        p_ENAME,
                        p_JOB,
                        p_MGR,
                        p_HIREDATE,
                        p_SAL,
                        p_COMM,
                        p_DEPTNO
                );

                p_EMPNO := num_EMPNO;

        END;

        -- //// Update Stored Procedure based on primary keys
        PROCEDURE Update_EMP
        (

                p_EMPNO IN EMP.EMPNO%TYPE,
                p_ENAME IN EMP.ENAME%TYPE,
                p_JOB IN EMP.JOB%TYPE,
                p_MGR IN EMP.MGR%TYPE,
                p_HIREDATE IN EMP.HIREDATE%TYPE,
                p_SAL IN EMP.SAL%TYPE,
                p_COMM IN EMP.COMM%TYPE,
                p_DEPTNO IN EMP.DEPTNO%TYPE
        )
        AS
        BEGIN
                update EMP set
                ENAME = p_ENAME, JOB = p_JOB, MGR = p_MGR, HIREDATE = p_HIREDATE, SAL =
p_SAL, COMM = p_COMM, DEPTNO = p_DEPTNO
                WHERE EMPNO = p_EMPNO;
        END;

        -- //// Select All Stored Procedures
        PROCEDURE ReadAll_EMP (p_cur OUT cursor_type)
        AS
```

```
	BEGIN
		open p_cur for
		select  EMPNO,
				ENAME,
				JOB,
				MGR,
				HIREDATE,
				SAL,
				COMM,
				DEPTNO
		from    EMP;
	END;

	-- //// Delete Stored Procedure based on primary keys
	PROCEDURE Delete_EMP
	(
		p_EMPNO IN EMP.EMPNO%TYPE
	)
	AS
	BEGIN
		delete from    EMP
		WHERE EMPNO = p_EMPNO;
	END;

	-- //// Read all Stored Procedure based on foreign key
	PROCEDURE ReadALL_BY_DEPTNO_EMP
	(
		p_cur OUT cursor_type,
		p_DEPTNO IN EMP.DEPTNO%TYPE
	)
	AS
	BEGIN
		open p_cur for
		select  EMPNO,
				ENAME,
				JOB,
				MGR,
				HIREDATE,
				SAL,
				COMM,
				DEPTNO
		from    EMP
		WHERE DEPTNO = p_DEPTNO;
	END;

	-- //// Read Stored Procedure based on primary keys
	PROCEDURE Read_EMP
	(
		p_cur OUT cursor_type,
		p_EMPNO IN EMP.EMPNO%TYPE
	)
	AS
	BEGIN
		open p_cur for
		select  EMPNO,
				ENAME,
				JOB,
				MGR,
				HIREDATE,
				SAL,
				COMM,
				DEPTNO
		from    EMP
		WHERE EMPNO = p_EMPNO;
	END;

END EMP_PKG;
--End Package body
/
--[End of Package for table: EMP]
--================================================================
```