



SQLPortal User Manual



Reference:	SQLPortalUM.doc
Version:	2.1
Date last changed:	05-06-2003

Copyright

Copyright © 2003 by Consonant Solutions B.V.

All rights reserved. No part of the contents of this publication may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Consonant Solutions B.V.

Jupiterstraat 96

2132 HE HOOFFDORP

The Netherlands

Phone : +31-(0)23-5556364

Fax : +31-(0)23-5553218

Website : www.sqlportal.nl

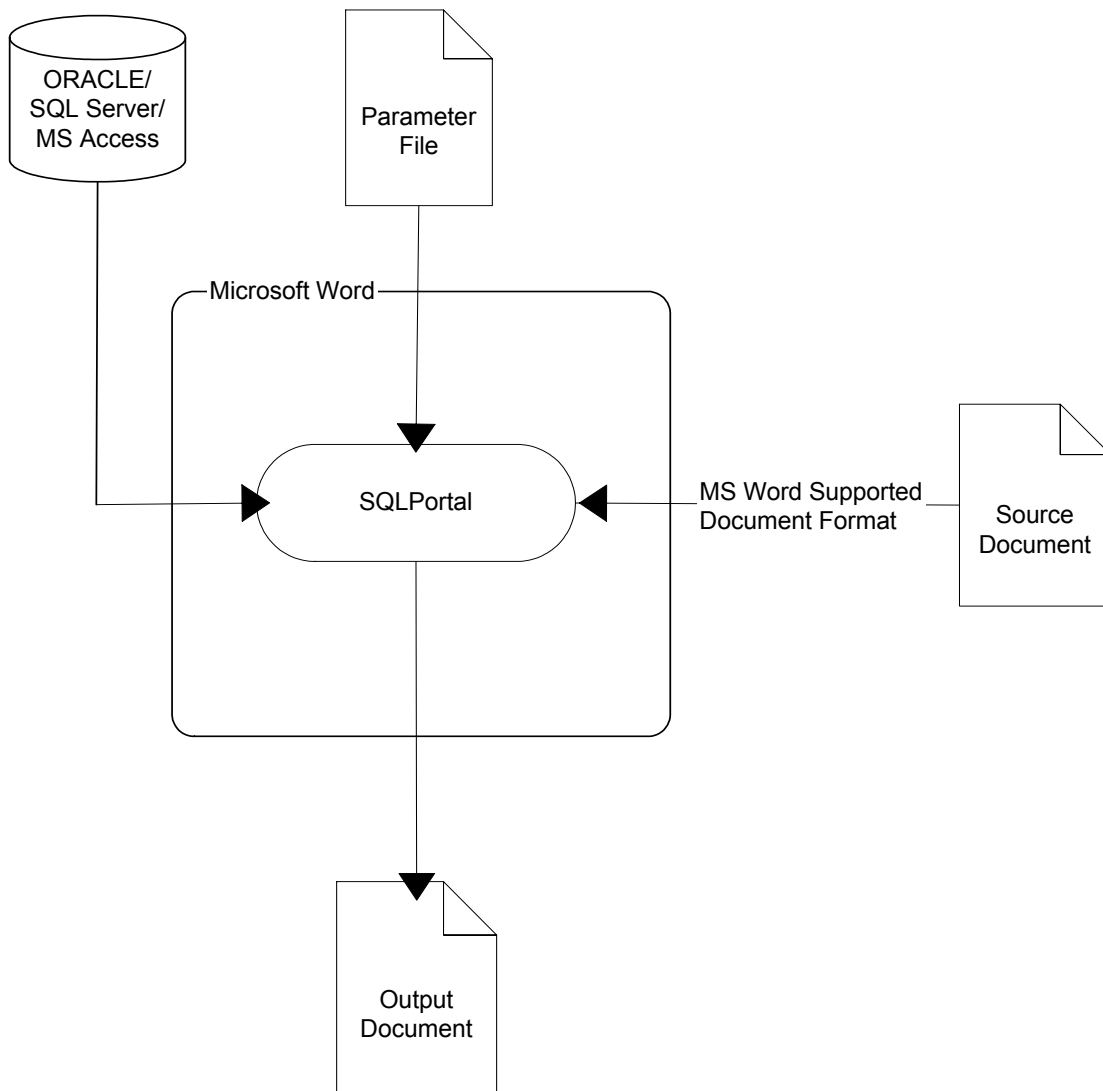
Table of Contents

Introduction	4
Starting the program	5
Starting SQLPortal Professional.....	5
Starting SQLPortal for Microsoft Access.....	5
Command line parameters	5
Using command line parameters correctly.....	8
Parameter file	9
Using the parameter file in combination with parameters on the command line	9
Graphical User Interface (GUI).....	9
Keyboard shortcuts in the GUI	10
GUI related to the command line parameters	11
Providing Oracle specific information.....	11
Providing SQL Server specific information.....	12
Providing Access specific information	13
SQLPortal Tips	15
The basics of SQLPortal	15
Keywords.....	15
General document structure.....	16
Example 1: single select query with a hard coded variable value.....	17
Example 2: single select query with a soft coded variable value	18
Example 3: multi select query	18
Nested blocks and bind variables.....	19
Example 4: nested blocks	20
Example 5: multiple nested blocks.....	21
Creating a source document.....	23
Block definition function.....	24
SQL definition function	24
Using Word tables.....	25
Performance / tips and tricks	26
Avoid using the asterisk (*) in your select clause.....	26
Use the single quote ' SQL query string delimiter	26
How to type in the straight quote ' instead of the smart quote ‘ in Word	26
Avoid using reserved words in your database query	27
How to make use of a command file	27
Learning the program's basic functions.....	27
Troubleshooting	27
What to do with an illegal SQL statement error message.....	27
What to do with the error message: given command-line arguments (..) are incorrect	28
What to do when a command line parameter does not appear in the GUI.....	28
What to do when a database field has not been substituted at all.....	28
What to do when a database field has not been substituted correctly.....	28
What to do when a parameter gets substituted with an incorrect value	29
What to do when the output document is empty	29
Getting Help	29
For more information or comments, please contact us	29
Service and Support.....	29
Appendix A: Example database.....	29
Contents of the database	30
Setting up the Oracle Example Database	30
Setting up the SQL Server Database.....	31
Oracle scripts.....	31
SQL Server scripts	32

Introduction

SQLPortal is an easy way to incorporate all relevant information from databases into your Microsoft Word documents. End users can create their own letters, contracts, reports and mailings containing data from the database. All they have to do is to add database fields and database queries to their Word document. ICT Departments can effortlessly satisfy company's reporting needs in a cost effective way.

SQLPortal currently supports Oracle, Microsoft SQL Server and Microsoft Access databases.



Starting the program

SQLPortal comes in several versions: SQLPortal Professional and SQLPortal for Microsoft Access. SQLPortal Professional supports Oracle, Microsoft SQL Server and Microsoft Access databases. The product is targeted at advanced users and professional software developers. SQLPortal for Microsoft Access is a low cost version specially developed for Microsoft Access users.



Starting SQLPortal Professional

SQLPortal Professional can be started in either the command line mode or the graphical user interface mode. The Graphical User Interface (GUI) is described below in detail and will be shown if the following command has been specified:

```
sqlportal.exe showgui=y
```

or if “SQLPortal” in the Windows Start Menu has been selected.



Starting SQLPortal for Microsoft Access

SQLPortal for Microsoft Access can be started in either the command line mode or the graphical user interface mode. The Graphical User Interface (GUI) is described below in detail and will be shown if the following command has been specified:

```
sqlportal_access.exe showgui=y
```

or if “SQLPortal for Access” in the Windows Start Menu has been selected.

Command line parameters

SQLPortal recognizes the following command line parameters.

Parameter	Description
userid = <username> ^P	<p>Passes the username for logging on to the database.</p> <p><i>Oracle:</i> Oracle interprets the userid as the schema name to which the connection will be made. Please see your Oracle administrator manual for more information regarding the login options of Oracle.</p> <p><i>SQL Server:</i> SQL Server interprets the userid as a user who is allowed to access a database. It is not necessarily the same name as the database to which the connection is being made. When the parameter is being omitted, Windows authentication will be used.</p> <p><i>Access:</i> This parameter is not used.</p>

^P The parameter is not supported in SQLPortal for Microsoft Access.

Parameter	Description
pwd = <password>	Passes the password for the login
dbtype = <type> ^P	<p>Passes the database type to which the connection will be made. The following values can be passed:</p> <p>ORACLE MSSQL MSACCESS</p> <p>The values are not case sensitive.</p>
infile = <path and file.ext>	<p>Passes the input filename. The input file name is the source document in which the relevant database fields and SQL statements are specified.</p> <p>The <path and file.ext> can either be a fully quantified path or an UNC path. The format is respectively:</p> <p>c:\mypath\myinfile.ext or \\myserver\myshare\mypath\myinfile.ext</p> <p>The infile cannot be the same as the outfile. The file extension can be any document extension that Microsoft Word supports</p>
outfile = <path and file.ext>	<p>Passes the output filename. The output file name is the destination document in which the result will be stored.</p> <p>The <path and file.ext> can either be a fully quantified path or an UNC path. The format is respectively:</p> <p>c:\mypath\myoutfile.ext or \\myserver\myshare\mypath\myoutfile.ext</p> <p>The outfile cannot be the same as the infile. The file extension is not of consequence. The outfile will always be a Word Document.</p>
server = <server name> ^P	<p>Passes the computer name of where the database is located.</p> <p><i>Oracle:</i> The server parameter is the TNSName to the Oracle Server.</p> <p><i>SQL Server:</i> The server parameter is either the name of the server of where the SQL Server database is running (SQL 6.5 or SQL2000 with one database instance) or <code>servername\instance_name</code> (SQL 2000) when there are multiple instances of the SQL Server running on the same computer.</p> <p><i>Access:</i> This parameter is not used.</p>
database = <database name>	Passes the name of the database.

^P The parameter is not supported in SQLPortal for Microsoft Access.

Parameter	Description
	<p><i>Oracle:</i> The parameter is not used.</p> <p><i>SQL Server:</i> The Database name is the actual name of the database.</p> <p><i>Access:</i> If the dbtype = MSACCESS then the value of database must be set to the path and filename of the MSACCESS database file. The path can either be a fully quantified path or an UNC path. E.g. c:\mypath\mydatabase.mdb or \\myserver\myshare\mypath\mydatabase.mdb</p>
parfile = <path and file.ext>	<p>Optional parameter which passes the path and file name to the parameter file. The structure of the parameter file is as follows:</p> <pre>parameter name 1=parameter value 1 parameter name 2=parameter value 2</pre> <p>The parfile is a plain text file. The parameters within the parameter file can be applied to the infile.</p> <p>The <path and file.ext> can either be a fully quantified path or an UNC path. The format is respectively:</p> <pre>c:\mypath\myparfile.ext or \\myserver\myshare\mypath\myparfile.ext</pre> <p>Note 1: Instead of using the parfile parameter, you can also use the 'parameter' parameter. Note 2: The parfile parameter can also be used in combination with the 'parameter' parameter. See below for details. Note 3: SQLPortal will not interpret the command line parameters when they are placed in the parfile as being commands for SQLPortal.</p>
showeditor = Y / N	<p>By default SQLPortal will exit without showing the outfile in Word. By explicitly setting <code>showeditor=Y</code> you will force SQLPortal to exit and leave Word running with the result document still open. This parameter is only relevant if the GUI is not used. When the <code>showgui</code> parameter is set to yes the <code>showeditor</code> parameter will automatically be set to Y.</p>
showgui = Y / N	<p>By default the GUI (Graphical User Interface) will not be shown to the user in command line mode and SQLPortal relies completely on the provided command line arguments. By explicitly setting <code>showgui=y</code> SQLPortal will show the GUI before processing the infile.</p>
parameter <name 1>=<value 1> <name 2>=<value 2> ... <name n>=<value n>	<p>Optional parameter that passes a list of parameters which will be applied to the infile.</p> <p>An example of the list of parameters :</p>

Parameter	Description
	emp_id=1209 color=red db="Oracle and Access"
	Note 1: Instead of using the 'parameter' parameter, you can also use the parfile parameter. Note 2: The 'parameter' parameter can also be used in combination with the parfile parameter. See below for details.
service = Y / N ^P	By setting the service parameter you will prevent SQLPortal from generating user interaction. as best as possible. If the service parameter has been set, then SQLPortal will always ignore the logging parameter and will not perform any logging.
logging = Y / N ^P	By default SQLPortal will write SQL commands and error messages to the log file. If you do not want SQLPortal to append data to the log file, you have to set the parameter logging=N.

Using command line parameters correctly

The first command line parameter is placed after the program name with a single space between the program name and the first parameter.

The command line parameters must be separated from each other with a ; (semicolon).

A closing semicolon at the end of the command line parameter list is optional.

Spaces are allowed between the command line parameter and its value. For example `userid = Bill` is the same as `userid=Bill`.

The sequence of the command line parameters is irrelevant to the working of SQLPortal.

The command line parameter names are not case sensitive.

The next example shows how you can startup SQLPortal Professional in command line mode for a particular Oracle database.

```
"C:\Program Files\SQLPortal\sqlportal.exe"
infile=c:\Program
Files\SQLPortal\examples\ExampleSource.doc;outfile=C:\Pro
gram
Files\SQLPortal\examples\Target.doc;userID=koesen;pwd=ko3
6e45y;dbtype=ORACLE;server=database;ShowEditor=Y;showgui=
Y
```

^P The parameter is not supported in SQLPortal for Microsoft Access.

Parameter file

The parameter file contains the variables that will be used when SQLPortal processes the infile. It is a plain text file that cannot be used as a substitute for passing command line parameters. Each parameter has to be placed on a separate line in the parameter file. For example, the parameter file can have the following contents:

```
UserName = Bill
userage = 20
```

To access a variable from a parameter file just place within the infile an ampersand in front of the parameter name. For example, *The name is* &UserName will result in &UserName being replaced with the text *Bill*.

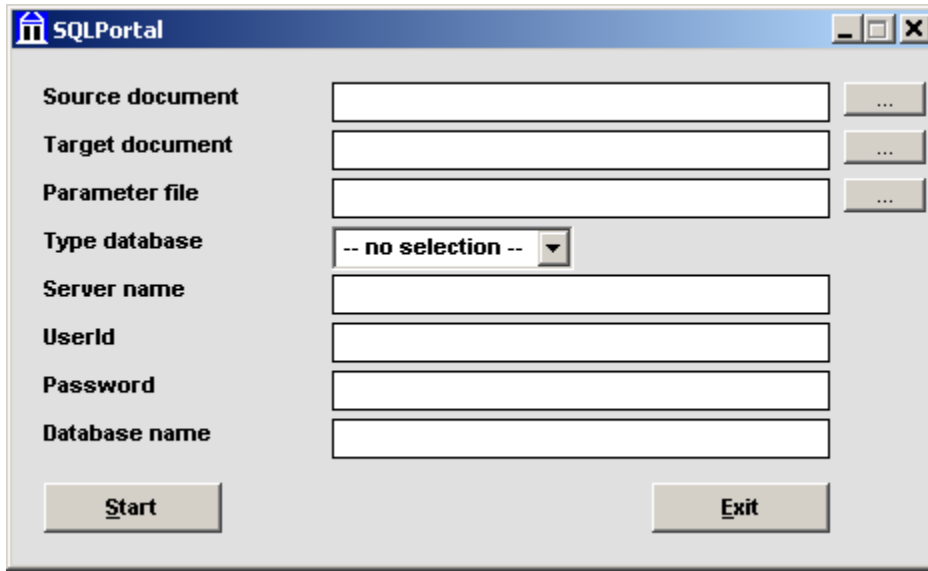
Using the parameter file in combination with parameters on the command line

Starting SQLPortal in the command line mode, you can use the parameter file in combination with parameters specified directly on the command line. SQLPortal will read the parameters on the command line at first and subsequently read the parameters specified inside the parameter file. If a parameter is indicated more than once, then SQLPortal will always process only the first occurrence of the parameter.

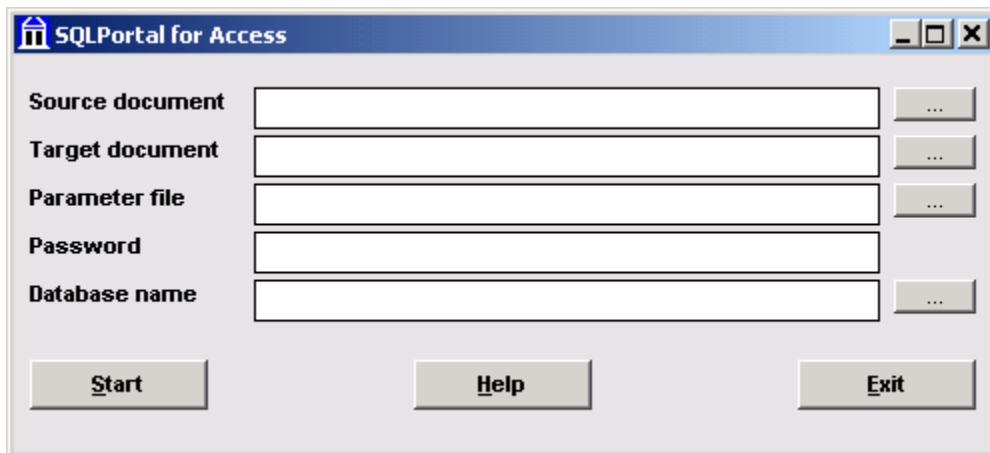
Graphical User Interface (GUI)

To start SQLPortal in the graphical user mode just pass the parameter `showgui=y`. All other parameters can be omitted because the graphical user interface will provide the means for giving the values for the other parameters. When additional parameters are provided, the GUI will read the parameters from the command line and set the GUI fields accordingly.

The SQLPortal Professional Graphical User Interface has the following layout.



In SQLPortal for Microsoft Access the layout of the Graphical User Interface looks as follows:



Keyboard shortcuts in the GUI

The list below shows the SQLPortal keyboard shortcuts.

Keyboard shortcut	Description
TAB	Move forward through the fields
SHIFT + TAB	Move back through the fields
HOME	Move to the beginning of a field
END	Move to the end of a field

Keyboard shortcut	Description
UP ARROW	Scroll towards the beginning of a field
DOWN ARROW	Scroll towards the end of a field
ALT + S	Start generating a SQLPortal target document
ALT + E	Exit the application
ALT + F4	Close active window
F1	Help

GUI related to the command line parameters

The table below shows how the GUI fields are related to the command line parameters. SQLPortal Professional supports several databases in contrast to SQLPortal for Microsoft Access. The columns Oracle, SQL Server and Access show whether the parameter is optional or mandatory for the corresponding database type.

GUI field	Command line parameter	Oracle	SQL Server	Access
Source document	infile ^P	x	x	x
Target document	outfile	x	x	x
Parameter file	parfile	optional	optional	optional
Type Database	dbtype ^P	x	x	x
Server name	server ^P	x	x	
UserId	userid	x	optional	
Password	pwd	x	optional	optional
Database name	database		x	x

When starting SQLPortal in the graphical user interface mode, the showeditor parameter will be ignored. SQLPortal automatically sets the value for showeditor to Y causing SQLPortal, after completing its tasks, to leave Word running with the target document open for further editing.



Providing Oracle specific information

The following parameter combination is specific to Oracle.

```
dbtype      = ORACLE
server      = <TNSNAME>
userid      = <Schema name to log on to>
pwd         = <Corresponding password>
```

For instance:

^P The parameter is not supported in SQLPortal for Microsoft Access.

The screenshot shows the SQLPortal application window. It has a title bar with the icon and text 'SQLPortal'. The window contains several input fields and buttons. The 'Source document' field is 'ExampleSource.doc', 'Target document' is 'Target.doc', and 'Parameter file' is empty. The 'Type database' dropdown is set to 'ORACLE'. The 'Server name' is 'database', 'UserId' is 'koesen', 'Password' is masked with '#####', and 'Database name' is empty. At the bottom are 'Start' and 'Exit' buttons.

Source document	ExampleSource.doc	...
Target document	Target.doc	...
Parameter file		...
Type database	ORACLE	
Server name	database	
UserId	koesen	
Password	#####	
Database name		

Start Exit



Providing SQL Server specific information

SQLPortal gains access to the SQL Server database using either SQL authentication or Windows authentication. The following parameter combination is specific using SQL Authentication

```
dbtype      = MSSQL
server      = <SQLSERVER> or <SQLSERVER\Instance>
userid      = <Username used to log on to the database>
pwd         = <Corresponding password>
database    = <Database name>
```

For example:

The screenshot shows the SQLPortal application window with the 'Type database' dropdown set to 'MSSQL'. The 'Server name' is 'srv-prod2-venus', 'Database name' is 'exampledb', and 'UserId' is 'koesen'. The 'Source document' is 'ExampleSource.doc', 'Target document' is 'Target.doc', and 'Parameter file' is empty. The 'Password' is masked with '#####'. At the bottom are 'Start' and 'Exit' buttons.

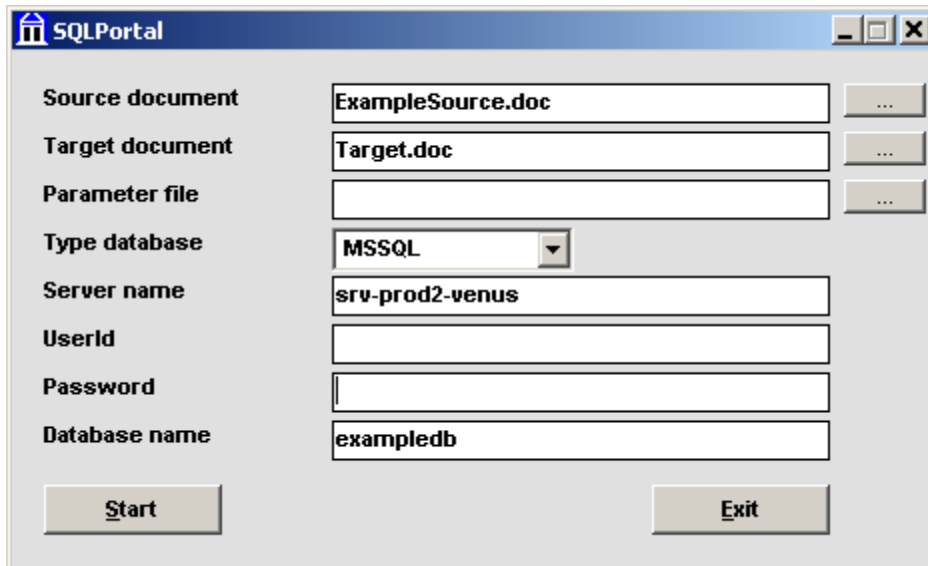
Source document	ExampleSource.doc	...
Target document	Target.doc	...
Parameter file		...
Type database	MSSQL	
Server name	srv-prod2-venus	
UserId	koesen	
Password	#####	
Database name	exampledb	

Start Exit

Using Windows authentication:

```
dbtype      =      MSSQL
server      =      <SQLSERVER> or <SQLSERVER\Instance>
database    =      <Database name>
```

For example:



Providing Access specific information

An Access database can be accessed with or without a password. The following parameter combination is specific for access without a password:

```
dbtype = MSACCESS
database = <Path and file name to the database>
```

Access the database with a password:

```
dbtype = MSACCESS
database = <Path and file name to the database>
pwd = <Password>
```

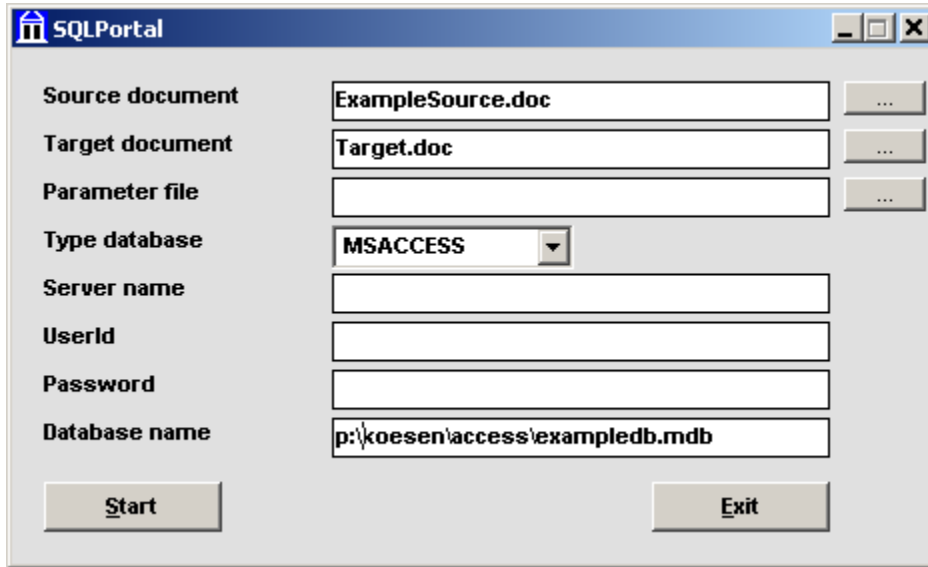
The database value is either in the format

c:\mypath\mydatabase.mdb

or

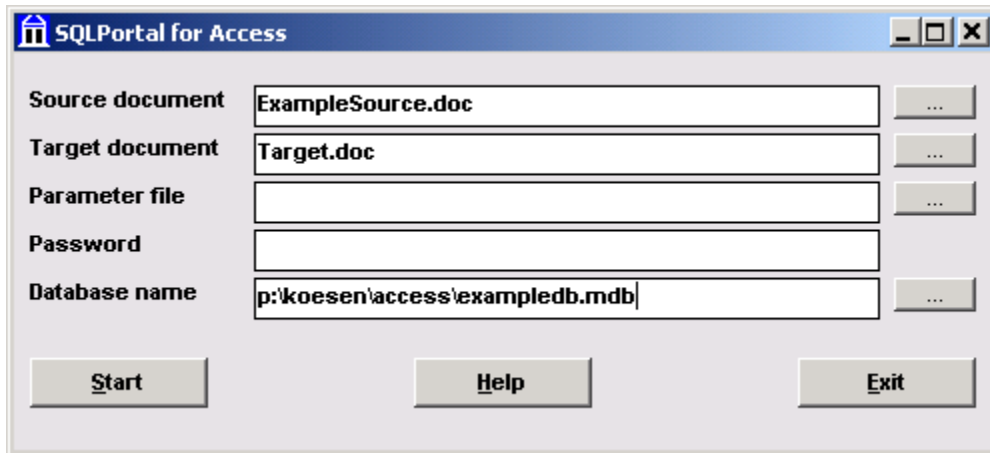
\\myserver\myshare\mypath\mydatabase.mdb

An example, when accessing the database without a password in SQLPortal Professional:



The screenshot shows the 'SQLPortal' dialog box. It has a title bar with the application icon and name. The main area contains several labeled text boxes and a dropdown menu. The 'Source document' box contains 'ExampleSource.doc', 'Target document' contains 'Target.doc', 'Parameter file' is empty, 'Type database' is set to 'MSACCESS' with a dropdown arrow, 'Server name' is empty, 'UserId' is empty, 'Password' is empty, and 'Database name' contains 'p:\koesen\access\exampledb.mdb'. At the bottom, there are two buttons: 'Start' and 'Exit'.

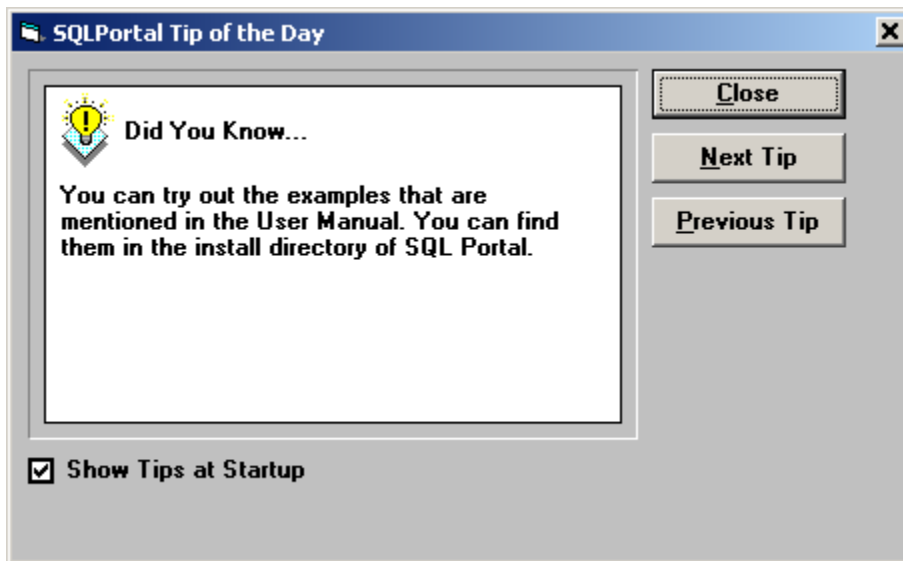
The same example in SQLPortal for Microsoft Access:



The screenshot shows the 'SQLPortal for Access' dialog box. It has a title bar with the application icon and name. The main area contains several labeled text boxes. The 'Source document' box contains 'ExampleSource.doc', 'Target document' contains 'Target.doc', 'Parameter file' is empty, 'Password' is empty, and 'Database name' contains 'p:\koesen\access\exampledb.mdb'. At the bottom, there are three buttons: 'Start', 'Help', and 'Exit'.

SQLPortal Tips

When you first start SQLPortal, you will see the SQLPortal Tip of the Day screen similar to the following:



The tip screen contains information on the use of SQLPortal. If you no longer wish the tip screen to display when opening SQLPortal, click on the **Show tips at startup** box to remove the checkmark. The tip screen will no longer display the next time SQLPortal is opened.

The basics of SQLPortal

Keywords

The starting-point is always a Word document in which you indicate the data you want to use. Subsequently you start up SQLPortal for this source document. In order to query the database SQLPortal needs to know which data has to be employed and which data has to be presented. For this SQLPortal uses the following keywords.

Keyword	Description
STARTBLOCK: <name>;	Signals the beginning of a section in which data from the corresponding database query can be used.
ENDBLOCK: <name>;	Signals the end of the section defined by STARTBLOCK. The <name> must correspond to the <name> given at a STARTBLOCK
STARTSQL: <name>;	Signals the beginning of a database query. The query result can be used in the STARTBLOCK section with the corresponding name.
ENDSQL: <name>;	Signals the end of a database query.

For every STARTBLOCK there must be a corresponding ENDBLOCK, STARTSQL and ENDSQL. You have to connect the keywords by using the same name. The colon is the delimiter between the keyword and the name. No spaces are allowed between the last letter of the keyword and the colon and between the colon and the name.

In case more data sections and database queries are specified all database queries should be preceded by all data sections. In other words the database queries are always at the end of the source document.

In the source document, you can always refer to a parameter that is specified in the parameter file. Furthermore, database queries can be connected to one another by so-called bind variables. The following keywords are provided for this purpose:

Reference parameter	Description
&	The ampersand is used for reference to a variable in the parameter file, e.g. &projectid.
FIELDNAME=:BINDVARIABLE	The bind variable acts like a placeholder for a value in a SQL statement. The placeholder is identified by a colon, and represents a value that you supply when the statement is executed. It can be used to create a child relationship to the master select statement.

General document structure

To retrieve data from a database, place at the end of your source document the STARTSQL and ENDSQL keywords. For example,

```
STARTSQL:myquery;
Select Field1
,      Field2
From Mytable
Where Field1 = 'apples'
ENDSQL:myquery;
```

To use the retrieved data in the source document place anywhere above the STARTSQL keyword the keywords STARTBLOCK and ENDBLOCK. Place between the STARTBLOCK and ENDBLOCK a field from the select statement between brackets []. For example,

```
STARTBLOCK:myquery;
Data from Field no1: [FIELD1]
Data from Field no2: [FIELD2]
ENDBLOCK:myquery;
```


The complete source document in the above mentioned example has the following content.

```
STARTBLOCK:myquery;
Data from Field no1: [FIELD1]
Data from Field no2: [FIELD2]
ENDBLOCK:myquery;

STARTSQL:myquery;
Select Field1
,      Field2
From Mytable
Where Field1 = 'apples'
ENDSQL:myquery;
```

When you present this source document to SQLPortal, SQLPortal will replace [FIELD1] and [FIELD2] with the corresponding values from the select statement and will delete all keywords and all database queries. It produces the following target document:

```
Data from Field no1: apples
Data from Field no2: 234
```

Example 1: single select query with a hard coded variable value

An employee report for employee number 2 is depicted the source document below. It's an example of a so-called single select query. A single select query is a query that returns a single row from the database. When this example is presented to SQLPortal, it will retrieve the name and id of employee with id 2 from the database.

To run the example, specify the source document, the target document and the database specific information.

The contents of the source documents:

```
STARTBLOCK:emp;
Reference number: [ID]
Name: [NAME]
ENDBLOCK:emp;

STARTSQL:emp;
SELECT id, name
FROM employees
WHERE id = 2
ENDSQL:emp;
```

SQLPortal will put the following contents in the target document specified:

```
Reference number: 2
Name: Mark Kok
```

Example 2: single select query with a soft coded variable value

The example above showed the employee report for employee number 2. The employee number was hard coded in the source document. Instead of hard coding the number, you can also specify the number in a parameter file or specify the number directly on the command line. In the following example, SQLPortal will retrieve the number from a specified parameter file.

To run the example, specify the parameter file, source document, the target document and the database specific information.

The contents of the parameter file:

```
empid=3
```

The contents of the source documents:

```
STARTBLOCK:emp;  
Reference number: [ID]  
Name: [NAME]  
ENDBLOCK:emp;  
  
STARTSQL:emp;  
SELECT id, name  
FROM employees  
WHERE id = &empid  
ENDSQL:emp;
```

SQLPortal will put the following contents in the target document specified:

```
Reference number: 3  
Name: Hanny Smit
```

Starting SQLPortal in the command mode, you can directly specify the parameter on the command line. An example of such a command is:

```
"C:\Program Files\SQLPortal\sqlportal.exe"  
infile=c:\Program  
Files\SQLPortal\examples\example2;outfile=C:\Program  
Files\SQLPortal\examples\out2.doc;dbtype=MSACCESS;databas  
e=C:\Program  
Files\SQLPortal\examples\database\samledb.mdb;ShowEditor  
=Y;showgui=y;parameter empid=3
```

Example 3: multi select query

An employee report for all employees is depicted the source document below. The report will show a list of all employees with their name, social security number and department number.

To run the example, specify the source document, the target document and the database specific information.

The contents of the source documents:

```
STARTBLOCK:emp;
ID: [ID]
Name: [Name]
Social Security nr: [SSNr]
Department number: [Depno]

ENDBLOCK:emp;

STARTSQL:emp;
Select id
, name
, ssnr
, prj
, depno
From employees
ENDSQL:emp;
```

SQLPortal will put the following contents in the target document specified:

```
ID: 1
Name: Johan Keizer
Social Security nr: 767101343
Department number: 3

ID: 2
Name: Mark Kok
Social Security nr: 557101343
Department number: 3

ID: 3
Name: Hanny Smit
Social Security nr: 235701343
Department number: 3

ID: 4
Name: Henk Tan
Social Security nr: 201999343
Department number: 2
```

Nested blocks and bind variables

You can create nested blocks in SQLPortal. Each block corresponds to exactly one database query and the database queries are connected by the use of bind variables. An example of bind variables is shown below:

```
STARTSQL:proj;
Select id as p_id
```

```

,      name as p_name
From projects
ENDSQL:proj;

STARTSQL:emp;
Select id   as e_id
,      name as e_name
,      ssnr
From employees
Where prj = :p_id
ENDSQL:emp;

```

The two queries can be used to obtain the employees for each project. The bind variable :p_id in the second query is used to link it to the first SQL statement. The variable refers to a column of the first query.

Example 4: nested blocks

The source document below describes a list of all employees per project. The document contains a database query to retrieve all projects and a second database query to get the employees of each project. The bind variable :p_id in the second query is used to link the projects.

To run the example, specify the source document, the target document and the database specific information.

The contents of the source documents:

List of employees per project

```

STARTBLOCK:proj;
On project [P_NAME] ([P_ID]) are working:
STARTBLOCK:emp;
    ID: [E_ID]
    Name: [E_NAME]
    SSNR: [SSNR]

```

```

ENDBLOCK:emp;
ENDBLOCK:proj;
end of list

```

```

STARTSQL:proj;
Select id   as p_id
,      name as p_name
From projects
ENDSQL:proj;

```

```

STARTSQL:emp;
Select id   as e_id
,      name as e_name
,      ssnr
From employees

```

```
Where prj = :p_id
ENDSQL:emp;
```

SQLPortal will put the following contents in the target document specified:

List of employees per project

On project Webshop (1) are working:

ID: 1
Name: Johan Keizer
SSNR: 767101343

ID: 3
Name: Hanny Smit
SSNR: 235701343

ID: 4
Name: Henk Tan
SSNR: 201999343

On project SQLPortal (2) are working:

ID: 2
Name: Mark Kok
SSNR: 557101343

end of list

Example 5: multiple nested blocks

The following sample code takes the nested blocks concept a step further by linking four tables together. The result will be an overview of all projects. Per project all people who are working on that project will be listed with their skills and in which department they are working.

To run the example, specify the source document, the target document and the database specific information.

The contents of the source documents:

Extensive list of employees per project

```
STARTBLOCK:proj;
On project [P_NAME] ([P_ID]) are working:
STARTBLOCK:emp;
    ID: [EMP_ID]
    Naam: [EMP_NAME]
    SSNr: [SSNR]
    Department nr.: [DEPNO]
STARTBLOCK:DEP;
    Department name: [DEP_NAME]
ENDBLOCK:DEP;
```

```

        Skills:
STARTBLOCK:Skill;
        [S_NAME]: [DESCRIPTION]
ENDBLOCK:Skill;

ENDBLOCK:emp;
ENDBLOCK:proj;

STARTSQL:proj;
Select id      as p_id
,              name as p_name
From projects
ENDSQL:proj;
STARTSQL:emp;
Select id      as emp_id
,              name as emp_name
,              ssnr
,              prj
,              depno
From employees
Where prj =:p_id
ENDSQL:emp;

STARTSQL:DEP;
Select id      as dep_id
,              name as dep_name
From departments
Where id =:depno
ENDSQL:DEP;

STARTSQL:Skill;
Select s.id      as s_id
,              s.name      as s_name
,              s.description as description
From  skills s, emp_skill es
Where s.id = es.skill_id
And    es.emp_id =:emp_id
ENDSQL:Skill;

```

SQLPortal will put the following contents in the target document specified:

Extensive list of employees per project

On project Webshop (1) are working:

```

ID: 1
Naam: Johan Keizer
SSNr: 767101343
Department nr.: 3
Department name: System Development
Skills:
English: reading and speaking
programming: programming in 3GL and 4GL

```

ID: 3
Naam: Hanny Smit
SSNr: 235701343
Department nr.: 3
Department name: System Development
Skills:
programming: programming in 3GL and 4GL

ID: 4
Naam: Henk Tan
SSNr: 201999343
Department nr.: 2
Department name: Marketing
Skills:

On project SQLPortal (2) are working:

ID: 2
Naam: Mark Kok
SSNr: 557101343
Department nr.: 3
Department name: System Development
Skills:
programming: programming in 3GL and 4GL
databases: dba knowledge of Oracle and SQL Server

Creating a source document

A source document contains references to database fields and one or more SQL statements that are enclosed with keywords. You can either type in the keywords yourself or you can make use of the SQLPortal keyword generator. The keyword generator is provided to you as a Microsoft Word template file also known as a dot file.

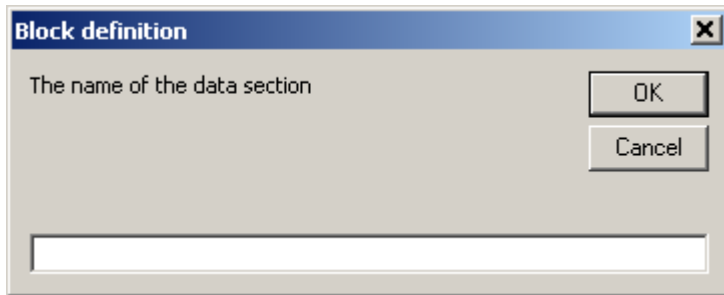
To access this keyword generator you have to create a new document through the Windows Start Menu option "Programs" > "SQLPortal" > "New Source Document" or option "Programs" > "SQLPortal for Access" > "New Source Document".

You can also open directly the SQLPortal dot file, which is located in the install directory of SQLPortal. This dot file makes not only the SQLPortal keyword generator functions available to you, but also the SQLPortal Help and SQLPortal Tutorial.

Since SQLPortal has a few keywords, only two keyword generator functions are needed. These functions are located in the SQLPortal Toolbar and are named "Block Definition" and "SQL Definition". From the SQLPortal Toolbar you can also access the SQLPortal Help and SQLPortal Tutorial.

Block definition function

After you have activated the Block Definition function, the following dialog box will appear in which you have to type in the name of the data section.



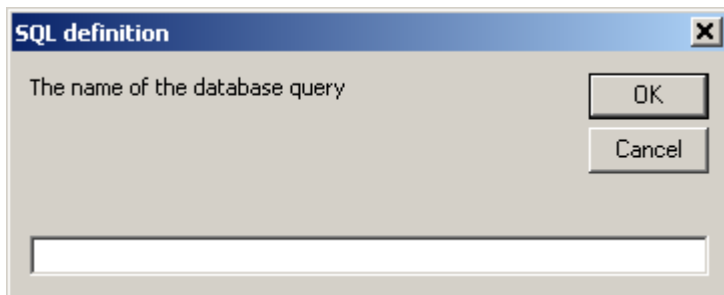
The STARTBLOCK and ENDBLOCK keywords will be inserted at the cursor position when the function was activated. If you typed in the text `emp` in the dialog box, the following text will be inserted into your document:

```
STARTBLOCK : emp ;
```

```
ENDBLOCK : emp ;
```

SQL definition function

As soon as you have activated the SQL Definition function, the following dialog box will appear in which you have to enter the name of the database query.



The STARTSQL and ENDSQL keywords will be inserted at the current cursor position. Typing in the text `emp` will result in the insertion of the following text into your document:

```
STARTSQL : emp ;
```

```
ENDSQL : emp ;
```


Using Word tables

SQLPortal has the unique feature that it can make use of Microsoft Word tables without the aid of additional keywords. The example below illustrates how the use of two tables will result into a single table with a header and detail information.

To run the example, specify the source document, the target document and the database specific information.

The contents of the source documents:

```
STARTBLOCK:proj;
```

On Project [P_ID], [P_NAME] are working:

ID	NAME	SSNR	Department	Project
STARTBLOCK:emp;				
[e_id]	[e_name]	[ssnr]	[depno]	[p_name]
ENDBLOCK:emp;				
ENDBLOCK:proj;				

```
STARTSQL:proj;
select id as p_id
,      name as p_name
from   projects p
ENDSQL:proj;
```

```
STARTSQL:emp;
select id as e_id
,      name as e_name
,      ssnr
,      prj
,      depno
from   employees
WHERE prj =:p_id
ENDSQL:emp;
```

SQLPortal will put the following contents in the target document specified:

On Project 1, Webshop are working:

ID	NAME	SSNR	Department	Project
1	Johan Keizer	767101343	3	Webshop
3	Hanny Smit	235701343	3	Webshop
4	Henk Tan	201999343	2	Webshop

On Project 2, SQLPortal are working:

ID	NAME	SSNR	Department	Project
2	Mark Kok	557101343	3	SQLPortal

To duplicate this example:

- In your source document, create a Word table with five columns and name them: ID, NAME, SSNR, Dept and Project.
- Place directly beneath this single row table the keyword `STARTBLOCK:emp;`.
- Place directly below it the second Word table also with five columns. Each column will contain a reference to the corresponding field by placing the field name between the brackets.
- Place directly beneath this single row table the keyword `ENDBLOCK:emp;`.
- Now place above the first table the keyword `STARTBLOCK:proj;` and place after the `ENDBLOCK:emp;` the keyword `ENDBLOCK:proj;`. Notice that the `emp` block is encapsulated in the `proj` block. This will give a master-detail result: for each project the employees will be shown.
- To finish this example off, place at the end of the document the database queries `proj` and `emp`. See above for the exact select statement that needs to be placed between the `STARTSQL` en `ENDSQL` statements. Notice the bind variable `:p_id` wich is used in the database query `emp`. The value of `p_id` comes from the database query `proj`.

Performance / tips and tricks

Avoid using the asterisk (*) in your select clause

You can use a 'select *' in your SQL statement. The asterisk (*) means 'give me all columns'. Consonant recommends that you avoid using asterisk in favor of explicitly coding the exact list of columns that you want to retrieve. The 'select *' will lead to a decreased performance of SQLPortal.

Use the single quote ' SQL query string delimiter

In SQL queries, strings are enclosed within a pair of single quotes. You cannot use double quotes.

How to type in the straight quote ' instead of the smart quote ‘ in Word

Single quote can be part of a SQL query. If you type in a straight quote in Word and you do not want Word to autocorrect it with a smart quote, turn off this feature as follows:

1. Open **Word**
2. On the **Tools** menu, click **AutoCorrect**
3. In the **Autocorrect** dialog box, use the second tab **AutoFormat As You Type**
4. In section **Replace as you type**, uncheck the "straight quotes" with "smart quotes" box.

Avoid using reserved words in your database query

Reserved words have special meaning to a database management system, so they can't be used in your SQL statement. Consult your database documentation for the list of reserved words.

How to make use of a command file

By following the steps below you will start SQLPortal using the command line parameters and specify only the infile and outfile.

1. Create a command file (.bat or .cmd) in which you specify the access to the database.
2. Add the infile parameter to the file and give it the value %1
3. Add the outfile parameter to the file and give it value %2.

Under here an example of the contents of such a command file is shown. In the example a database connection is made with a Microsoft Access database.

```
"C:\Program Files\SQLPortal\sqlportal.exe"  
infile=%1;outfile=%2;dbtype=MSACCESS;database=C:\Program  
Files\SQLPortal\examples\sampladb.mdb;showeditor=Y;showgu  
i=n
```

4. Next start up the command file with two parameters. The first parameter will specify the infile and the second parameter the outfile.

If the command file has been stored as C:\my_cmds\sqlportal.cmd, then the call could be as follows:

```
C:\my_cmds\sqlportal.cmd C:\my_files\example10.doc  
C:\my_files\out.doc
```

The %1 will be substituted by the first parameter (that is C:\my_files\example10.doc) and the %2 will be substituted by the second parameter (that is C:\my_files\out.doc). Next the SQLPortal application will be started.

Learning the program's basic functions

An multimedia tutorial is available that aids in rapid learning of the program's basic functions. The tutorial is not part of packaged product and has to be downloaded separately from the website www.sqlportal.nl.

Troubleshooting

What to do with an illegal SQL statement error message

If you encounter an illegal SQL statement error, a database query that you have written down is not correct. Consult the log file for detailed information about the

error. The indications in the log file will help you to locate the part of the SQL statement likely to contain the flaw.

One of the causes of an illegal SQL statement error could be that you are making use of a parameter in your document and that you either did not specify the parameter file or the parameter file does not contain the referenced parameter.

Depending upon the database type used, the log file will show one of the following similar error messages:

```
Oracle message      :ORA-01008: not all variables bound
Access message      :Syntax error (missing operator) in
                     query expression 'id = &empid'.
SQL Server message  :Line 1: Incorrect syntax near '&'
```

What to do with the error message: given command-line arguments (..) are incorrect

If you come upon this error message, then check you command line arguments. One of next situations could occur:

- You did not apply the correct separator. Command line arguments must be separated with a semicolon.
- You put a new line character in your command. The text between brackets indicates only the first part of the specified command.

What to do when a command line parameter does not appear in the GUI

If a parameter does not show up in the GUI, then you most likely misspelled the name of the parameter or you enclosed the command line parameter with quotes. In the last case, remove the quotes.

Notice that the 'parameter' parameter will not be shown in the GUI.

What to do when a database field has not been substituted at all

One of the following situations could occur:

- a. The database field is not part of a STARTBLOCK / ENDBLOCK section.
- b. The database field had been placed in a STARTBLOCK / ENDBLOCK section and the database field has not been mentioned in the corresponding STARTSQL / ENDSQL section or in the surrounding STARTSQL / ENDSQL sections.

What to do when a database field has not been substituted correctly

In a document that contains instructions for SQLPortal all database fields should have a unique name. If database fields located in different SQL statements have the same name, than an correct substitution is not guaranteed. SQLPortal will substitute a database field in a STARTBLOCK / ENDBLOCK section with the first occurrence in the corresponding SQL statement or surrounding SQL statements.

You can make database fields unique by using column aliases.

What to do when a parameter gets substituted with an incorrect value

If an incorrect value has been substituted for a parameter, then probably you have specified this parameter several times. SQLPortal will always process only the first occurrence of the parameter. Therefore all subsequent occurrences will be ignored by SQLPortal.

Check your parameter file and / or the parameters that are specified on the command line for duplicates.

What to do when the output document is empty

If the file produced by SQLPortal is empty, then the database queries you specified is likely to return no rows. In other words, no rows in the tables queried satisfy the selection you specified in your database query. Check whether your selection criteria are too strict. If you referred to a variable in the parameter file, check the value you specified in the parameter file.

Getting Help

For more information or comments, please contact us

Consonant Solutions BV
Jupiterstraat 96
2132 HE HOOFFDORP
The Netherlands
Phone: +31 (0)23 5556364
Fax: +31 (0)23 5553218
Email: Info@consonant.nl

Service and Support

Email: support@consonant.nl
Phone: +31 (0)23 5556364
Fax: +31 (0)23 5553218

Appendix A: Example database

For your convenience the SQL scripts for the Oracle database and SQL Server database are provided below so that it is possible to try out the given examples. The SQL scripts are also packaged with SQLPortal Professional. The packaged product too contains an Access Example Database. The Access Example Database is also packaged with SQLPortal for Microsoft Access. You do not need a license to access the Access Example Database through SQLPortal.

Contents of the database

The databases consists of the tables departments, projects, skills, employees and emp_skill.

Departments:

ID	NAME
1	Sales
2	Marketing
3	System Development

Projects:

ID	NAME
1	Webshop
2	SQLPortal

Skills:

ID	NAME	Description
1	English	reading and speaking
2	programming	programming in 3GL and 4GL
3	databases	dba knowledge of Oracle and SQL Server

Employees:

ID	NAME	SSNR	Depno	Prj
1	Johan Keizer	767101343	3	1
2	Mark Kok	557101343	3	2
3	Hanny Smit	235701343	3	1
4	Henk Tan	201999343	2	1

Emp_skill:

EMP_ID	SKILL_ID
1	1
1	2
2	2
2	3
3	2

Setting up the Oracle Example Database

Point of departure:

- you have an Oracle account at your disposal
- your account has been granted create table privileges
- the install directory of SQLPortal contains the script directory in which the create_tables_oracle.sql and the fill_tables_oracle.sql scripts reside

To set up the Oracle database:

1. Start SQL*Plus
2. Log on to your Oracle account

3. Execute the SQL statements from file create_tables_oracle.sql by issuing the following command in SQL*Plus: @ c:\Program Files\SQLPortal\script\create_tables_oracle.sql
4. Execute the SQL statements from file fill_tables_oracle.sql in the same manner.

Setting up the SQL Server Database

Point of departure:

- you have an SQL Server account at your disposal
- your account has been granted create table privileges
- the install directory of SQLPortal contains the script directory in which the create_tables_mssql.sql and the fill_tables_mssql.sql scripts reside

To set up the SQL Server database:

1. Start the SQL Query Analyzer
2. Log on to your account
3. Execute the SQL statements from file create_tables_mssql.sql by issuing the following commands:
 - On the File menu, click Open
 - Open file create_tables_mssql.sql
 - On the Query menu, click Execute
4. Execute the SQL statements from file fill_tables_mssql.sql in the same manner

Oracle scripts

Create table scripts

```
drop table departments;
create table departments
(
    id      integer      not null,
    name varchar2(40) not null,
    constraint pk_dep primary key (id)
);
commit;

drop table projects;
create table projects
(
    id      integer      not null,
    name varchar2(40) not null,
    constraint pk_prj primary key (id)
);
commit;

drop table skills;
create table skills
(
    id            integer      not null,
    name          varchar2(40) not null,
    description varchar2(512) not null,
    constraint pk_skill primary key (id)
);
commit;
```

```

drop table employees;
create table employees
(
    id      integer      not null,
    name    varchar2(40) not null,
    ssnr    number(9)     null,
    depno   integer      not null,
    prj     integer      null,
    constraint pk_emp primary key (id)
);
commit;

drop table emp_skill;
create table emp_skill
(
    emp_id   integer not null,
    skill_id integer not null,
    constraint pk_empskill primary key (emp_id, skill_id)
);
commit;

```

Fill table scripts

```

insert into departments (id, name) values (1, 'Sales');
insert into departments (id, name) values (2, 'Marketing');
insert into departments (id, name) values (3, 'System
Development');
COMMIT;

insert into projects (id, name) values (1, 'Webshop');
insert into projects (id, name) values (2, 'SQLPortal');
COMMIT;

insert into skills (id, name, description) values
(1, 'English', 'reading and speaking');
insert into skills (id, name, description) values
(2, 'programming', 'programming in 3GL and 4GL');
insert into skills (id, name, description) values
(3, 'databases', 'dba knowledge of Oracle and SQL Server');
COMMIT;

insert into employees (id, name, ssnr, depno, prj) values
(1, 'Johan Keizer', 767101343, 3, 1);
insert into employees (id, name, ssnr, depno, prj) values
(2, 'Mark Kok', 557101343, 3, 2);
insert into employees (id, name, ssnr, depno, prj) values
(3, 'Hanny Smit', 235701343, 3, 1);
insert into employees (id, name, ssnr, depno, prj) values
(4, 'Henk Tan', 201999343, 2, 1);
COMMIT;

insert into emp_skill (emp_id, skill_id) values (1, 1);
insert into emp_skill (emp_id, skill_id) values (1, 2);
insert into emp_skill (emp_id, skill_id) values (2, 2);
insert into emp_skill (emp_id, skill_id) values (2, 3);
insert into emp_skill (emp_id, skill_id) values (3, 2);
COMMIT;

```

SQL Server scripts

Create table scripts.


```

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name = 'departments')
create table departments
(id          smallint      not null,
 name       varchar(40) not null,
 PRIMARY KEY (id))
GO

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name = 'projects')
create table projects
(id          smallint      not null,
 name       varchar(40) not null,
 PRIMARY KEY (id))
GO

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name = 'skills')
create table skills
(id          smallint      not null,
 name       varchar(40) not null,
 description varchar(512) not null,
 PRIMARY KEY (id))
GO

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name = 'employees')
create table employees
(
 id          smallint      not null,
 name       varchar(40) not null,
 ssnr       int            null,
 depnosmallint not null,
 prj        smallint      null,
 PRIMARY KEY (id))
GO

IF NOT EXISTS (SELECT * FROM sysobjects WHERE name = 'emp_skill')
create table emp_skill
(emp_id      smallint not null,
 skill_id    smallint not null,
 PRIMARY KEY (emp_id, skill_id))
GO

```

Fill table script

```

insert into departments (id, name) values (1, 'Sales')
go
insert into departments (id, name) values (2, 'Marketing')
go
insert into departments(id, name) values (3, 'System Development')
go
insert into projects (id, name) values (1, 'Webshop')
go
insert into projects (id, name) values (2, 'SQLPortal')
go
insert into skills (id, name, description) values
(1, 'English', 'reading and speaking')
go
insert into skills (id, name, description) values
(2, 'programming', 'programming in 3GL and 4GL')
go
insert into skills (id, name, description) values
(3, 'databases', 'dba knowledge of Oracle and SQL Server')
go
insert into employees (id, name, ssnr, depno, prj) values
(1, 'Johan Keizer', 767101343, 3, 1)
go

```

```
insert into employees (id, name, ssnr, depno, prj) values
(2, 'Mark Kok', 557101343, 3, 2)
go
insert into employees (id, name, ssnr, depno, prj) values
(3, 'Hanny Smit', 235701343, 3, 1)
go
insert into employees (id, name, ssnr, depno, prj) values
(4, 'Henk Tan', 201999343, 2, 1)
go
insert into emp_skill (emp_id, skill_id) values (1, 1)
go
insert into emp_skill (emp_id, skill_id) values (1, 2)
go
insert into emp_skill (emp_id, skill_id) values (2, 2)
go
insert into emp_skill (emp_id, skill_id) values (2, 3)
go
insert into emp_skill (emp_id, skill_id) values (3, 2)
go
```