SQL Prompt 3.9

July 2008

Note: these pages apply to a version of
this product that is not the current released version.

For the latest support documentation, please see
http://documentation.red-gate.com

redgate®
ingeniously simple tools

Contents

Getting started

SQL Prompt provides intelligent auto-completion for SQL Server Management Studio, SQL Server Management Studio Express, Query Analyzer, and Visual Studio®, enabling you to build accurate SQL scripts much more quickly.

SQL Prompt displays a candidate list as you type, when you press CTRL+SPACEBAR, or when you type a trigger word. The candidate list shows items based on the context of what you type. Candidates are grouped by object type for you to select and insert into your query editor.

To see how you use SQL Prompt's main interface, see Using the candidate list (page 21).

SQL Prompt is compatible with SQL Server 2008, SQL Server 2005, and SQL Server 2000. SQL Prompt supports SQL Server 2008 keywords and syntax.

You can use SQL Prompt to:

- expand column lists in SELECT * and SELECT *table.** statements by pressing TAB
- select columns using a column picker
- apply keyword formatting as you type
- lay out existing SQL code according to a range of options (Pro version only)
- view schema information to see the definition of an object
- qualify object names and column names
- execute stored procedures
- create snippets of code for queries you run regularly
- assign aliases automatically, including custom aliases
- customize auto-completion behaviour by specifying options

SQL Prompt supports cross-database queries as well as distributed queries with linked SQL Server instances.

Worked examples

Learn more about SQL Prompt by following the detailed examples (page 5).

Installing SQL Prompt

To install SQL Prompt, your Windows® user must have administrator privileges. Close any open query windows before you start.

If you are upgrading from a previous version the installation program does all the preparation for you.

For details of how to uninstall SQL Prompt, see Uninstalling SQL Prompt.

SQL Prompt in SQL Server Management Studio

When you install SQL Prompt, add-ins to SQL Server Management Studio and SQL Server Management Studio Express Edition are automatically installed for you. SQL Prompt features are available from the **SQL Prompt** menu.

SQL Prompt in Query Analyzer

The SQL Prompt 3 icon in the notification area of your Windows desktop is for Query Analyzer support. When you have installed SQL Prompt 3, SQL Prompt runs automatically whenever you start Query Analyzer.

By default, SQL Prompt starts automatically whenever you start Query Analyzer. To switch off support for SQL Prompt in Query Analyzer, right-click the SQL Prompt 3 icon in the notification area, and click **Start SQL Prompt when you start Query Analyzer** so that the menu item is not ticked; to switch on support for SQL Prompt, click **Start SQL Prompt when you start Query Analyzer** so that the menu item is ticked.

To remove the SQL Prompt icon from the notification area, right-click the SQL Prompt 3 icon, and click **Exit**; to add the SQL Prompt 3 icon so that you can switch on or switch off support for SQL Prompt with Query Analyzer, click the Windows **Start** menu, and on the **SQL Prompt 3** menu, click **SQL Prompt Query Analyzer Integration**.

SQL Prompt in Visual Studio

When you install SQL Prompt, an add-in to Microsoft Visual Studio 2005 is automatically installed for you. The SQL Prompt features are available from the **SQL Prompt** menu.

Worked examples

This topic shows you how you can use SQL Prompt. The following examples are provided:

- Typing a simple query (page 5)
- Expanding column lists (page 7)
- Selecting columns (page 10)
- Generating full INSERT statements (page 10)
- Executing stored procedures (page 12)
- Using snippets (page 13)
- Cross-database queries and queries using linked SQL Servers (page 14)
- Common table expressions (page 19)

You can follow the examples on your own system. Most examples in this topic use SQL Server 2005 and you will need access to a SQL Server on which the AdventureWorks example database has been installed. In addition, the cross-database queries and queries using linked SQL Servers (page 14) examples use SQL Server 2000 on which the Northwind and pubs example databases have been installed.

To ensure that you see the same results as those illustrated, set SQL Prompt options as follows:

- On the **SQL Prompt** menu, click **Options**, and then click **Restore All Defaults**.
- Click the **Listed Candidates** tab and on the **Performance** page, select **Search entire batch/GO block**.

Typing a simple query

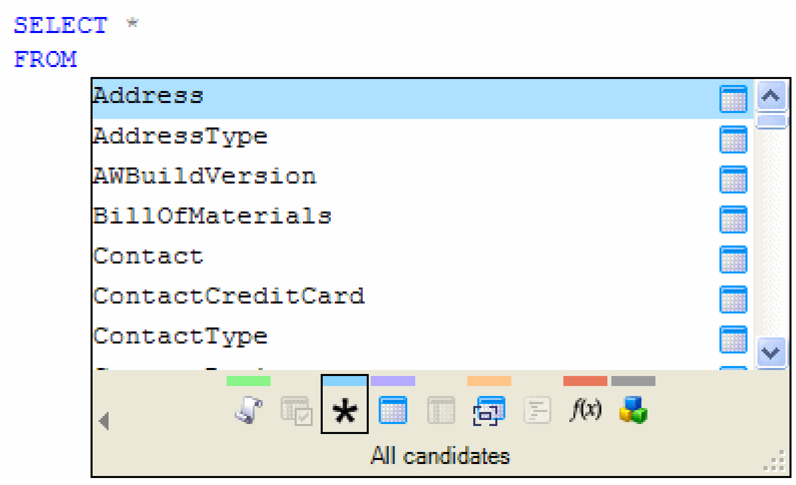
This example using AdventureWorks shows you how you can use SQL Prompt to assist you when you are typing a query.

1. On the **Options** dialog box, click the **Inserted Candidates** tab, click the **Formatting** page, and then select the **Qualify object names** check box and click **OK**.
2. In your query editor window, type:

   ```
   SELECT *
   FROM
   ```

   and press SPACEBAR.

The candidate list displays the ✱ **All candidates** category; tables are displayed at the top of the list.

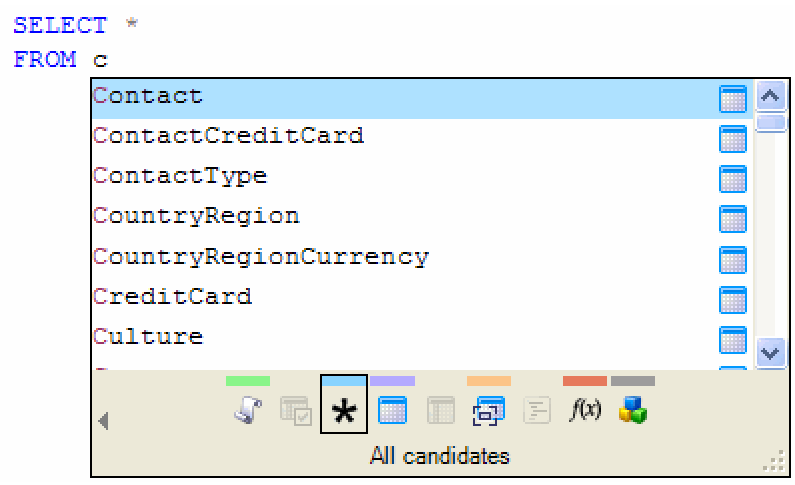


```
SELECT *
FROM
```

Address
AddressType
AWBuildVersion
BillOfMaterials
Contact
ContactCreditCard
ContactType

3. Type `c` to filter the list by candidates that begin with c.

   The Contact table is highlighted.



```
SELECT *
FROM c
```

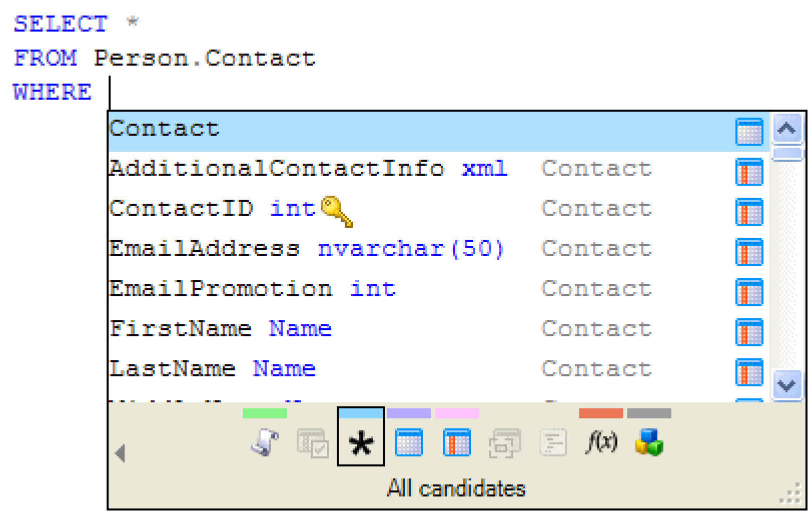
Contact
ContactCreditCard
ContactType
CountryRegion
CountryRegionCurrency
CreditCard
Culture

4. Press ENTER to insert the Contact table.

   SQL Prompt inserts the table name. Because you selected the **Qualify object names** option, SQL Prompt qualifies the table name with the schema (owner) name *Person*.

5. Type `WHERE` and press SPACEBAR.

The candidate list displays column names for the Contact table and associated data types.



```
SELECT *
FROM Person.Contact
WHERE
```

```
Contact
AdditionalContactInfo xml   Contact
ContactID int               Contact
EmailAddress nvarchar(50)   Contact
EmailPromotion int          Contact
FirstName Name              Contact
LastName Name               Contact
```

6. Type `L` and press ENTER to insert the LastName column.
7. Type `LIKE 'FR%'` so that your query returns rows where *LastName* begins with *Fr*.

   By default, when you type an opening quotation mark, SQL Prompt automatically inserts the closing quotation mark. If you type the closing quotation mark (for example, because you have not noticed the automatic insertion), SQL Prompt overtypes the auto-inserted quotation mark. You can change this setting and other auto-completion options, on the **Auto Insert**, **Closing Characters** options page.
8. Delete the * (after `SELECT`) and press CTRL+SPACEBAR to display the candidate list.

   The candidate list displays columns from the Contact table.

   Note that you must always type * after the SELECT keyword, even when you want use SQL Prompt to insert the column list later. If you do not type * the candidate list may display incorrect candidates. For more information, see Troubleshooting.
9. Type `f` and press ENTER to insert the FirstName column.
10. Press COMMA, type `L`, and then press ENTER to insert the LastName column.

```
SELECT FirstName,LastName
FROM Person.Contact
WHERE LastName LIKE 'FR%'
```

Expanding column lists

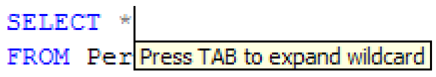
These examples using AdventureWorks show you how SQL Prompt can expand a column list when you are using a SELECT statement.

In the first example, you type SELECT * and you specify the tables or views in the FROM clause; you can then complete the column list by placing the insertion point after the * and pressing TAB:

1. In your query editor window, type:

   ```
   SELECT *
   FROM Person.Contact
   ```

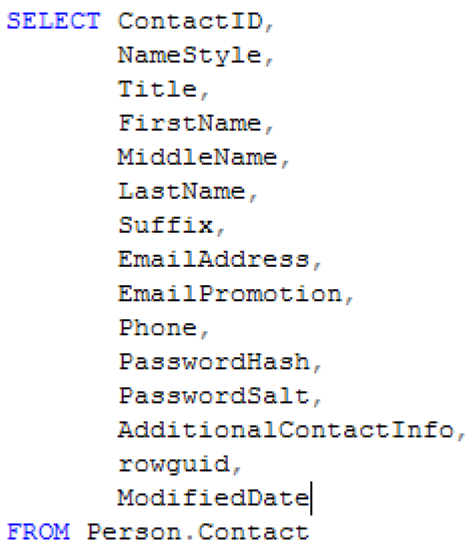
2. Place the insertion point after the *

   ```
   SELECT *
   FROM Per Press TAB to expand wildcard
   ```

3. Press TAB.

   SQL Prompt expands the column list.

   ```
   SELECT ContactID,
          NameStyle,
          Title,
          FirstName,
          MiddleName,
          LastName,
          Suffix,
          EmailAddress,
          EmailPromotion,
          Phone,
          PasswordHash,
          PasswordSalt,
          AdditionalContactInfo,
          rowguid,
          ModifiedDate
   FROM Person.Contact
   ```

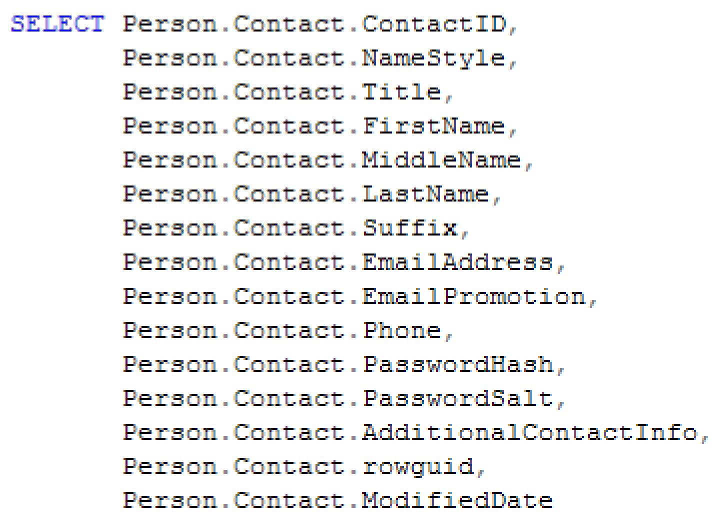
The next examples show how SQL Prompt can complete the column list for you when you type a SELECT *table*.* fragment and press TAB. The difference in the option **Qualify object names** is shown.

1. Ensure that **Qualify object names** is selected on the **Inserted Candidates**, **Formatting** options page.
2. Type:

   ```
   SELECT Person.Contact.*
   ```

3. Press TAB.

SQL Prompt expands the column list. SQL Prompt qualifies each table with the schema (owner) name.



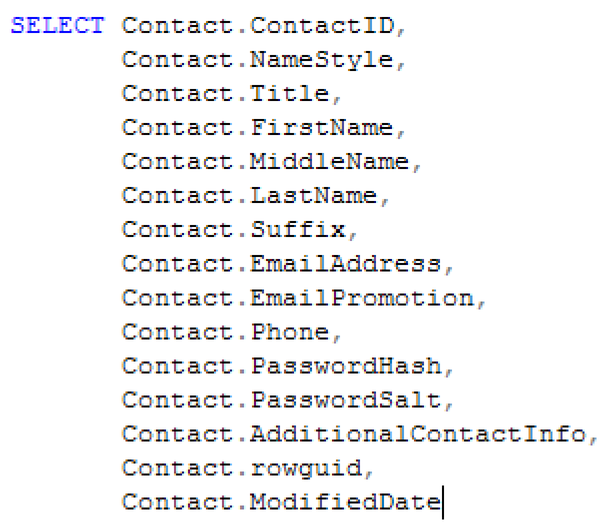
```
SELECT Person.Contact.ContactID,
       Person.Contact.NameStyle,
       Person.Contact.Title,
       Person.Contact.FirstName,
       Person.Contact.MiddleName,
       Person.Contact.LastName,
       Person.Contact.Suffix,
       Person.Contact.EmailAddress,
       Person.Contact.EmailPromotion,
       Person.Contact.Phone,
       Person.Contact.PasswordHash,
       Person.Contact.PasswordSalt,
       Person.Contact.AdditionalContactInfo,
       Person.Contact.rowguid,
       Person.Contact.ModifiedDate
```

4. Ensure that **Qualify object names** is not selected on the **Inserted Candidates**, **Formatting** options page.
5. Type:

   ```
   SELECT Person.Contact.*
   ```

6. Press TAB.

   SQL Prompt expands the column list. SQL Prompt does not qualify each table with the schema (owner) name.



```
SELECT Contact.ContactID,
       Contact.NameStyle,
       Contact.Title,
       Contact.FirstName,
       Contact.MiddleName,
       Contact.LastName,
       Contact.Suffix,
       Contact.EmailAddress,
       Contact.EmailPromotion,
       Contact.Phone,
       Contact.PasswordHash,
       Contact.PasswordSalt,
       Contact.AdditionalContactInfo,
       Contact.rowguid,
       Contact.ModifiedDate
```

Selecting columns

This example using AdventureWorks shows how you can use the SQL Prompt column picker to select columns for the column list in a SELECT statement:

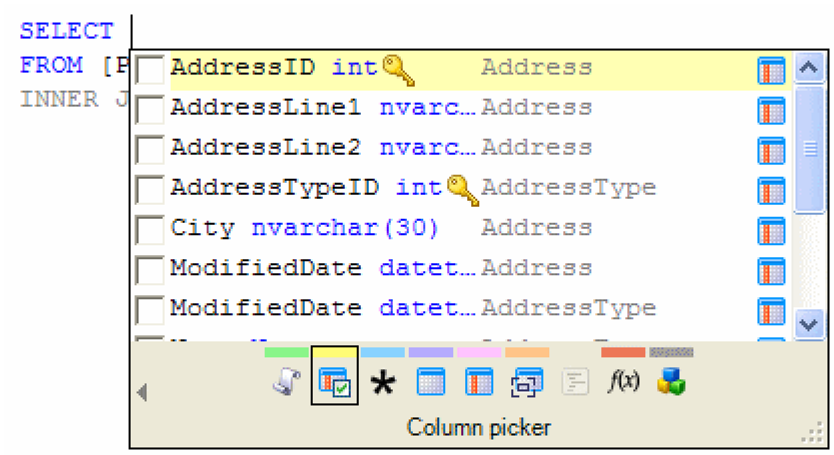
1. Type:

   ```
   SELECT *
   FROM [Person].[Address]
   INNER JOIN [Person].[AddressType]
   ```

2. Delete the * after `SELECT` and press CTRL+SPACEBAR to display the candidate list.

   The candidate list displays columns from the Address table and the AddressType table.

3. Press CTRL+LEFT ARROW or click to display the **Column picker** category.



```
SELECT
FROM [P
INNER J
```

```
AddressID int          Address
AddressLine1 nvarc...  Address
AddressLine2 nvarc...  Address
AddressTypeID int      AddressType
City nvarchar(30)      Address
ModifiedDate datet...  Address
ModifiedDate datet...  AddressType
```

4. Select columns in the order you want them to appear in the SQL code.

   To select a column, click the check box, or highlight the column and press SPACEBAR. You can use SHIFT and CTRL to select multiple columns.

5. Press ENTER to insert the selected columns into your query editor.

For more information about how to use the column picker, see Using the candidate List (page 21).

Generating full INSERT statements

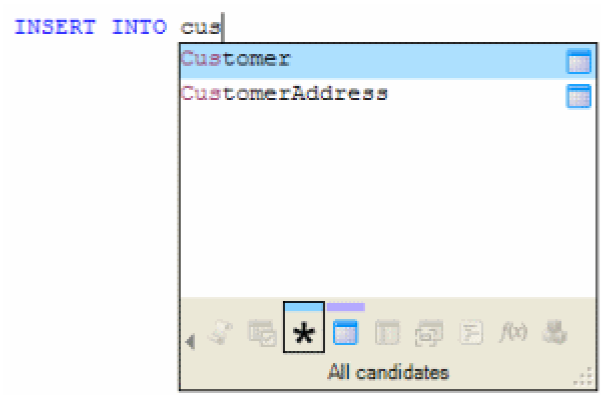
These examples using AdventureWorks show how SQL Prompt generates full INSERT statements.

1. On the **Options** dialog box, click the **Inserted Candidates** tab, click the **SQL** page, and ensure that the following check boxes are selected:

   - **Insert full INSERT statement**
   - **Insert default values for data types**
   - **Show hints for data types**

- **Include column name hints**

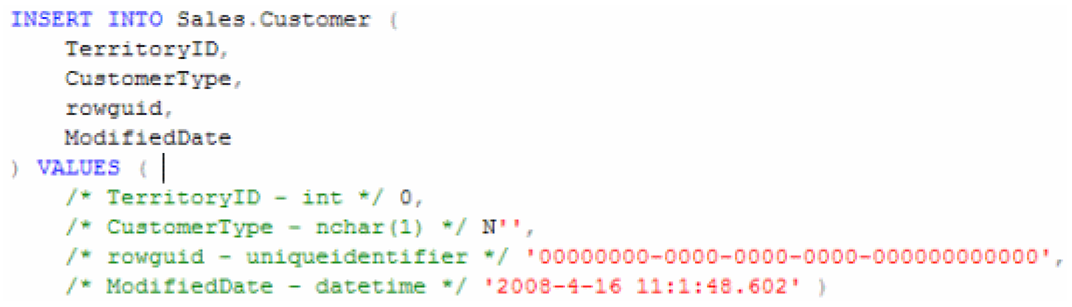
2. In your query editor window, type `INSERT INTO cus`



```
INSERT INTO cus
```

3. Select *Customer*.

   SQL Prompt inserts the INSERT statement. The insertion point is positioned for you to enter the values.



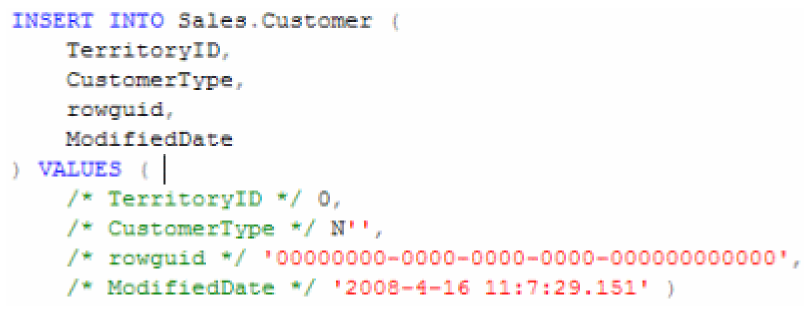
```
INSERT INTO Sales.Customer (
    TerritoryID,
    CustomerType,
    rowguid,
    ModifiedDate
) VALUES ( 
    /* TerritoryID - int */ 0,
    /* CustomerType - nchar(1) */ N'',
    /* rowguid - uniqueidentifier */ '00000000-0000-0000-0000-000000000000',
    /* ModifiedDate - datetime */ '2008-4-16 11:1:48.602' )
```

   In SQL Server 2008, you can insert multiple records into the same table in a single INSERT statement.

In the following example, hints are not shown.

1. On the **Options** dialog, click the **Inserted Candidates** tab, click the **SQL** page, and clear the **Show hints for data types** check box.
2. In your query editor window, type **INSERT INTO cus**, and select Customer from the candidate list.

   SQL Prompt inserts the full INSERT statement. The insertion point is positioned for you to enter the values.

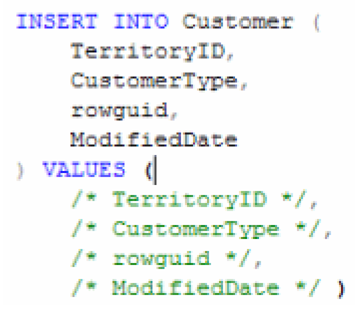


```
INSERT INTO Sales.Customer (
    TerritoryID,
    CustomerType,
    rowguid,
    ModifiedDate
) VALUES ( |
    /* TerritoryID */ 0,
    /* CustomerType */ N'',
    /* rowguid */ '00000000-0000-0000-0000-000000000000',
    /* ModifiedDate */ '2008-4-16 11:7:29.151' )
```

In the following example, default values are not shown for data types.

1. On the **Options** dialog, click the **Inserted Candidates** tab, click the **SQL** page, and clear the **Insert default values for data types** check box.
2. In your query editor window, type **INSERT INTO cus** and select Customer from the candidate list.

   SQL Prompt inserts the INSERT statement. The insertion point is positioned for you to enter the values.

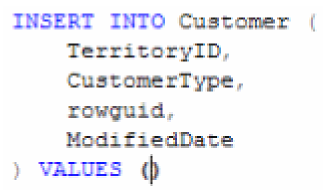


```
INSERT INTO Customer (
    TerritoryID,
    CustomerType,
    rowguid,
    ModifiedDate
) VALUES (|
    /* TerritoryID */,
    /* CustomerType */,
    /* rowguid */,
    /* ModifiedDate */ )
```

In the following example, column name hints are not shown.

1. On the **Options** dialog box, click the **Inserted Candidates** tab, click the **SQL** page, and clear the **Include column name hints** check box.
2. In your query editor window, type **INSERT INTO cus** and select Customer from the candidate list.

   SQL Prompt inserts the INSERT statement. The insertion point is positioned for you to enter the values.



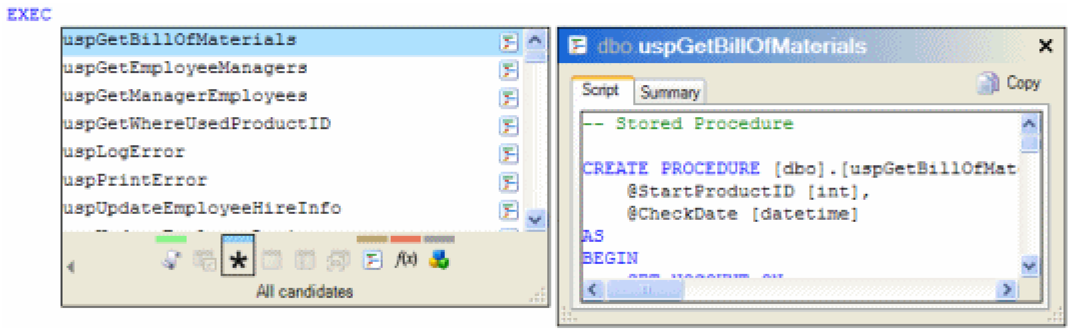
```
INSERT INTO Customer (
    TerritoryID,
    CustomerType,
    rowguid,
    ModifiedDate
) VALUES (|)
```

Executing stored procedures

This example using AdventureWorks shows how SQL Prompt inserts the parameters for stored procedures when you type EXEC and you select a stored procedure.

1. On the **Options** dialog box, click the **Inserted Candidates** tab, click the **SQL** page, and then ensure the **Insert parameters for functions and stored procedures** and **Show hints for data types** check boxes are selected.
2. In your query editor, type `EXEC` and then press SPACEBAR.

   SQL Prompt displays a list of stored procedures at the top of the candidate list. The schema panel shows the creation SQL for the highlighted stored procedure. For example:

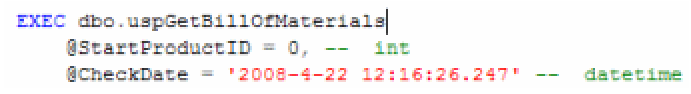


```
EXEC
uspGetBillOfMaterials
uspGetEmployeeManagers
uspGetManagerEmployees
uspGetWhereUsedProductID
uspLogError
uspPrintError
uspUpdateEmployeeHireInfo
```

```
-- Stored Procedure

CREATE PROCEDURE [dbo].[uspGetBillOfMat
    @StartProductID [int],
    @CheckDate [datetime]
AS
BEGIN
```

3. Select a stored procedure by pressing UP ARROW or DOWN ARROW to highlight it, and then press ENTER to insert it.

   SQL Prompt inserts the parameters for the stored procedure and inserts the associated data types as comments. For example:



```
EXEC dbo.uspGetBillOfMaterials
    @StartProductID = 0, --  int
    @CheckDate = '2008-4-22 12:16:26.247' --  datetime
```

Using snippets

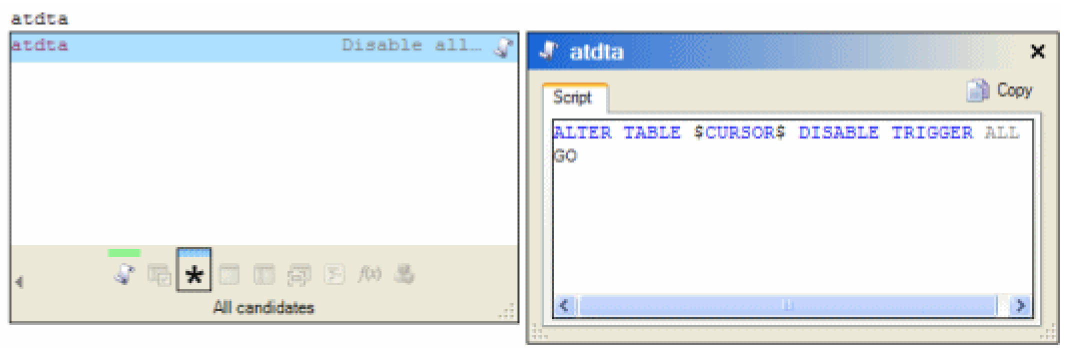
SQL Prompt is pre-configured with default snippets. To see the snippets, press CTRL+SPACEBAR and then CTRL+LEFT ARROW to display the **Snippets** category.

This example shows you how to insert the snippet code for disabling all triggers on a table.

1. Type:

```
atdta
```

The candidate list displays a description of the snippet and the schema panel displays the snippet code.



```
ALTER TABLE $CURSOR$ DISABLE TRIGGER ALL
GO
```

2. Press ENTER to insert the snippet code.

   SQL Prompt places the insertion point after TABLE for you to type the table name.

```
ALTER TABLE | DISABLE TRIGGER ALL
GO
```

Cross-database queries and queries using linked SQL Servers

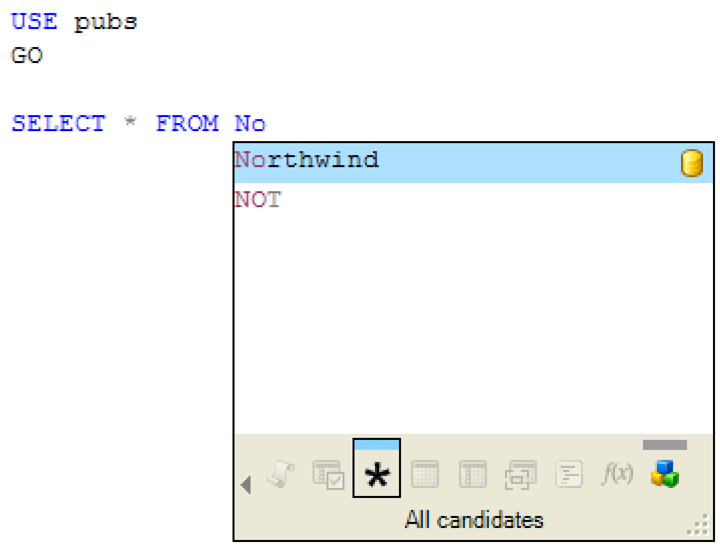
These examples use SQL Server 2000 on which the Northwind and pubs example databases have been installed.

The first example begins using the pubs database, and then accesses a table in the Northwind database.

1. On the **Options** dialog box, click the **Listed Candidates** tab, and on the **Cross-Database Support** page ensure the **Enable cross-database and linked server support** check box is selected.
2. In your query editor window, type:

```
USE pubs

GO


SELECT * FROM No
```

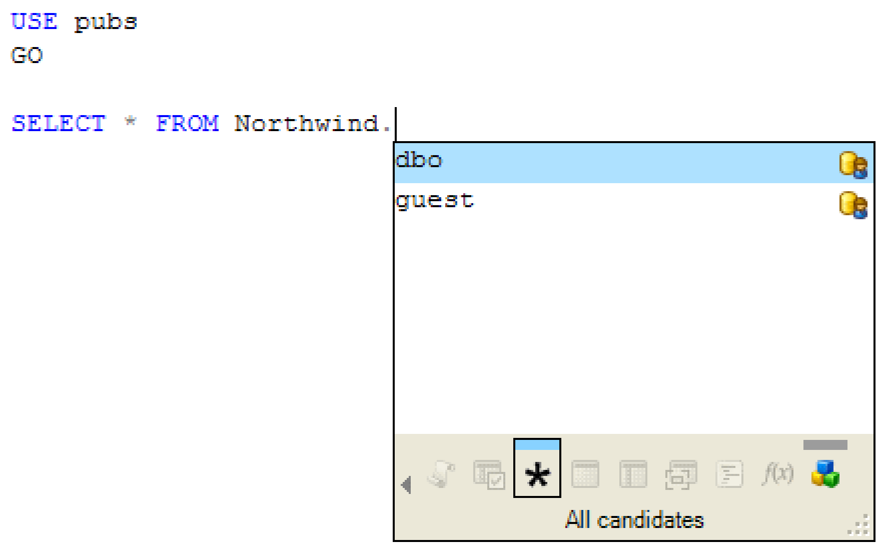
The candidate list displays the candidates. For example:



```
USE pubs
GO

SELECT * FROM No
```

3. Select Northwind.
4. Type **.** (dot).

   SQL Prompt caches the Northwind database. (Note that you may be asked to enter authentication credentials if SQL Prompt has been unable to log on to the database.)
5. Display the candidate list (CTRL+SPACEBAR)

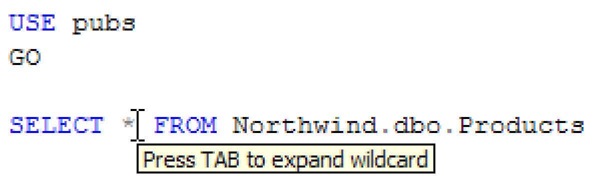


```
USE pubs
GO

SELECT * FROM Northwind.
```

6. Select dbo, and type **.** (dot) again.

   SQL Prompt displays the candidate list containing tables and views.

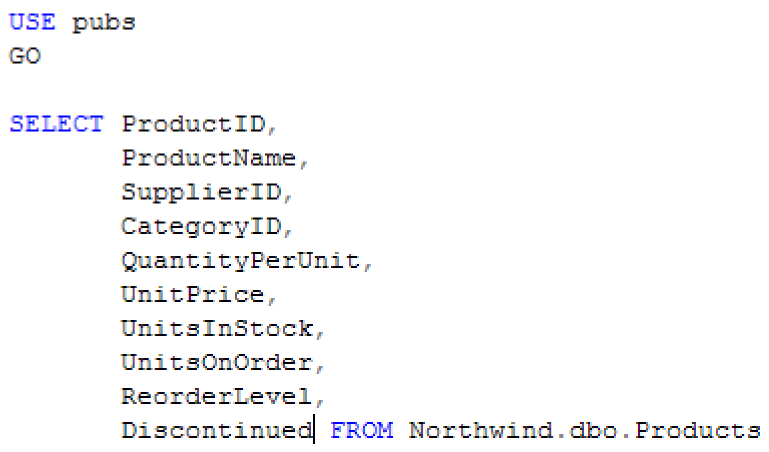
7. Select the Products table, then place the insertion point after the *.



```
USE pubs
GO

SELECT * FROM Northwind.dbo.Products
```

8. Press TAB.

   SQL Prompt inserts the columns from the dbo.Products table in the Northwind database.



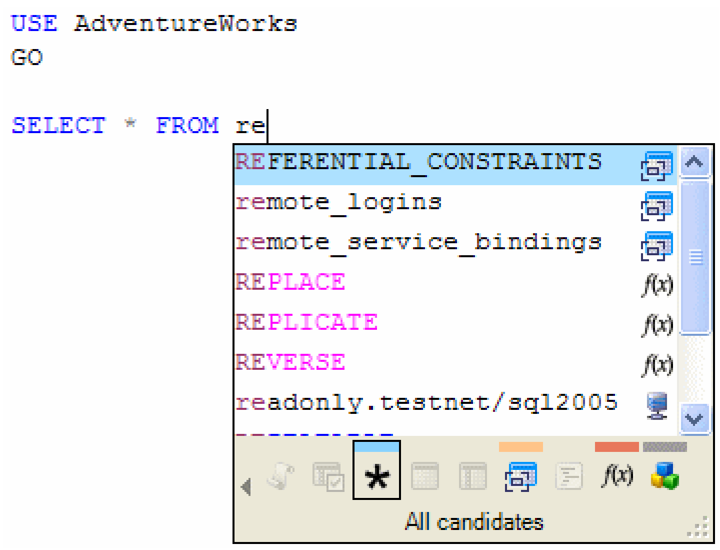
```
USE pubs
GO

SELECT ProductID,
       ProductName,
       SupplierID,
       CategoryID,
       QuantityPerUnit,
       UnitPrice,
       UnitsInStock,
       UnitsOnOrder,
       ReorderLevel,
       Discontinued FROM Northwind.dbo.Products
```

This next example shows a distributed query using a linked SQL Server 2000 instance from a SQL Server 2005 instance. The linked SQL Server 2000 instance is configured with the name READONLY.TESTNET on the SQL Server 2005 instance. For more information about linking servers, refer to SQL Server Books Online.

1. On the **Options** dialog box, click the **Listed Candidates** tab, and on the **Candidate Types and Filters** page ensure the **System objects** check box is selected.
2. On the SQL Server 2005 instance, in your query editor window, type:

```
USE AdventureWorks

GO


SELECT * FROM re
```

The candidate list displays the candidates. For example:

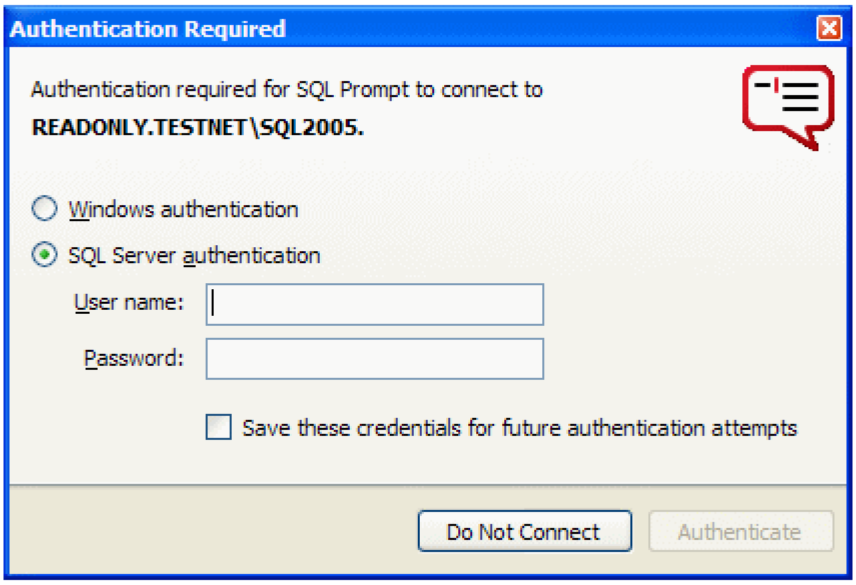


```
USE AdventureWorks
GO

SELECT * FROM re
```

```
REFERENTIAL_CONSTRAINTS
remote_logins
remote_service_bindings
REPLACE
REPLICATE
REVERSE
readonly.testnet/sql2005
```

3. Select the linked SQL Server, and then type **.** (dot).

   If the SQL Server uses SQL Server authentication and you have not saved the connection details, SQL Prompt displays a dialog box for you to enter the authentication credentials.



Authentication Required

Authentication required for SQL Prompt to connect to **READONLY.TESTNET\SQL2005.**

Windows authentication [✓ SQL Server authentication]

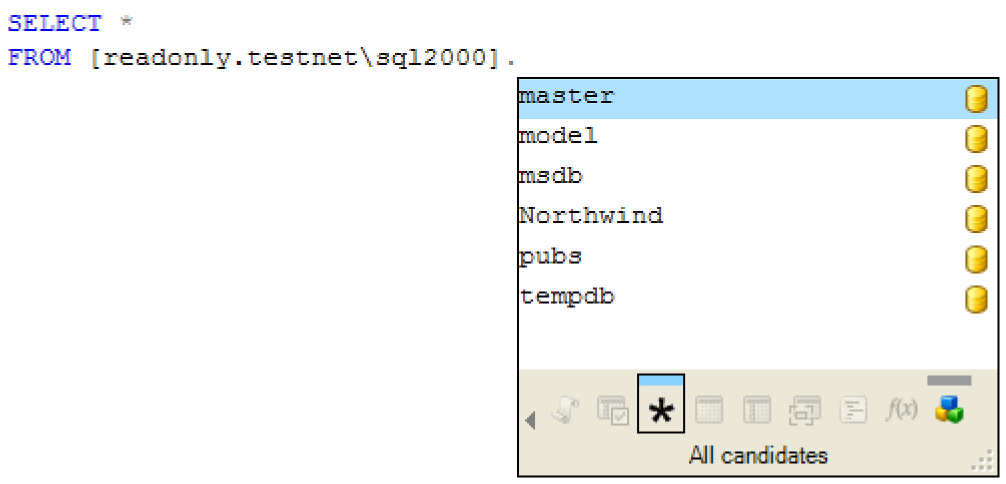
User name:

Password:

Save these credentials for future authentication attempts

If you click **Do Not Connect**, or you close the dialog box, in future SQL Prompt will not display any candidates for that SQL Server.

For this example, enter the credentials and click **Authenticate**, and then press CTRL+SPACEBAR to display the candidate list. The candidate list shows a list of databases on the linked server.



```
SELECT *
FROM [readonly.testnet\sql2000].
```

master
model
msdb
Northwind
pubs
tempdb

4. Select Northwind, and then type **.** (dot).

   You may need to press CTRL+SPACEBAR to display the candidate list. If the authentication credentials you entered allow you unrestricted access to the SQL Server, you are not prompted to enter them again.

5. Select dbo from the list of candidates, and then type **.** (dot).
6. Select the Customers table from the list of candidates, then place the insertion point after the *.
7. Press TAB.

   SQL Prompt inserts the columns from the dbo.Customers table in the Northwind database.

```
USE AdventureWorks
GO

SELECT CustomerID,
       CompanyName,
       ContactName,
       ContactTitle,
       Address,
       City,
       Region,
       PostalCode,
       Country,
       Phone,
       Fax
FROM [readonly.testnet\sql2000].Northwind.dbo.Customers
```

Common table expressions

This example shows you how you can use SQL Prompt when writing a query that contains a common table expression (CTE). A CTE is a named results set that exists only for the duration of the query.

This example uses the AdventureWorks database.

1. In your query editor window, type:

   ```
   WITH DirReps (ManagerID, DirectReports) AS
        (
         SELECT ManagerID, COUNT(*)
         FROM [HumanResources].[Employee] AS e
         WHERE [ManagerID] IS NOT NULL
         GROUP BY [ManagerID]
        )
   ```

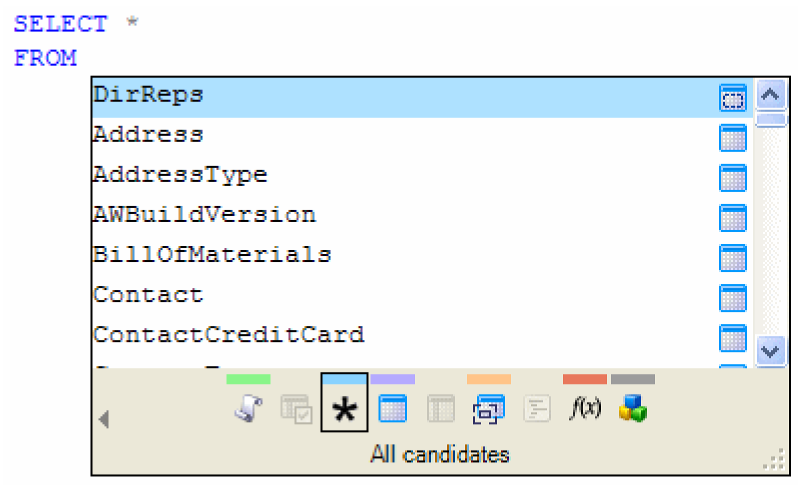
   This statement defines a CTE called DirReps that creates a simple set of results, the number of employees that report to each manager.

2. In your query editor window, type:

   ```
   SELECT *
   FROM
   ```

   and press SPACEBAR.

   The candidate list displays the * **All candidates** category. The CTE you have defined is displayed at the top of the list.



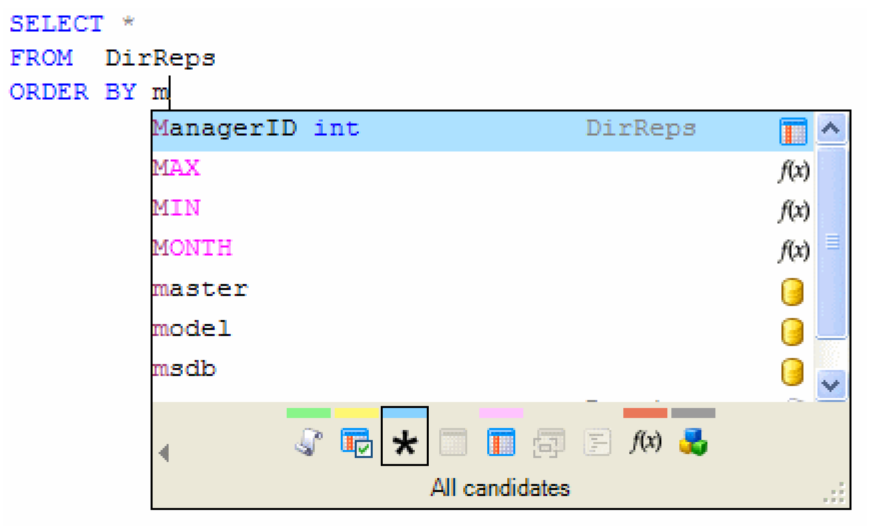
```
SELECT *
FROM
```

DirReps
Address
AddressType
AWBuildVersion
BillOfMaterials
Contact
ContactCreditCard

3. Press ENTER to insert the DirReps CTE.
4. Type:

   ```
   ORDER BY m
   ```

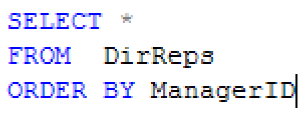
The candidate list displays the ManagerID column, the only column defined in the CTE that matches the first character typed.



```
SELECT *
FROM   DirReps
ORDER BY m
```

```
ManagerID int        DirReps
MAX
MIN
MONTH
master
model
msdb
```

5. Press ENTER to insert the column name.

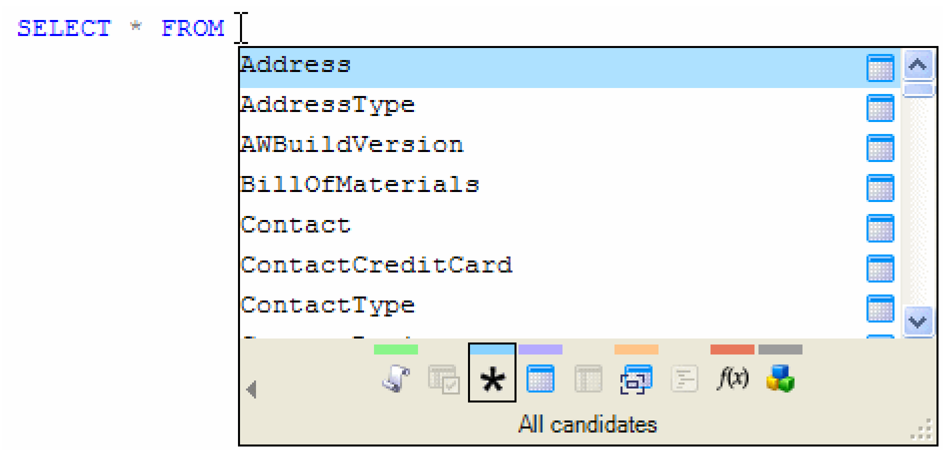


```
SELECT *
FROM   DirReps
ORDER BY ManagerID
```

Note that SQL Prompt also supports recursive CTEs, and multiple CTEs within a single WITH statement.

Using the candidate list

SQL Prompt provides assistance for SQL code completion with a *candidate list*. The candidate list displays items based on the context of what you type in your query editor. For example, when you type `SELECT * FROM`, the candidate list displays a list of suitable objects grouped by category.



```
SELECT * FROM
```

You can customize the appearance and behavior of the candidate list in your SQL Prompt options; for more information about the **Options** dialog box, see Setting options.

Some examples of how you use the candidate list are provided in Worked examples (page 5).

Displaying the candidate list

You can display the candidate list:

- automatically as you type

  SQL Prompt displays the candidate list automatically when you type the first character of a keyword or identifier.

  SQL Prompt also displays the candidate list automatically when you finish typing a *trigger* word and press SPACEBAR. Some trigger words are already defined for you, but if required, you can specify additional trigger words on the **Behavior**, **Triggering** options page.

  You can specify how long SQL Prompt will wait before displaying the candidate list. If you do not want the candidate list to display automatically, on the **Triggering** options page, click **Manual**.

- at any time, by pressing CTRL+SPACEBAR

To close the candidate list, press ESC.

To switch off the candidate list, on the **Behavior**, **Availability** options page, click **Layout only**.

Using categories

The candidate list displays items grouped by category. The categories are a way of filtering the candidate list so that you can find the item you require more easily. To switch between categories:

- click the category icon at the bottom of the candidate list
- or press CTRL+LEFT ARROW and CTRL+RIGHT ARROW

  If required, you can choose to use SHIFT+LEFT ARROW and SHIFT+RIGHT ARROW to switch between categories, using the **Behavior**, **Category Navigation** options page. You can also turn off this feature if necessary; for example, you may want to do this if you use these shortcut keys for a different purpose.

Note that if there are no candidates to display for a particular category, you cannot select that category.

The following categories are shown:

Snippets lists shortcuts for pre-defined SQL fragments or statements. If an insertion point has been defined, when you insert the snippet, SQL Prompt places the cursor at the relevant position in your query editor. For more information about snippets, see Managing Snippets (page 39).

Column picker lists column names, with associated data types and table names (or table aliases), for tables you have specified in a SELECT statement or for a table you have specified in an INSERT statement.

For details of how you select the columns, see Using the column picker below.

* All candidates is the default category. The candidates displayed in this category depend on the context of what you type in your query editor.

The following candidate types may be listed:

database names - database objects (for a full list, see Object types below)

linked servers - keywords (displayed in blue or gray)

data types - built-in functions (displayed in pink)

join conditions

local variables

By default, SQL Prompt lists exact matches at the top of the candidate list irrespective of their type, and the remaining candidates are grouped by type, and then sorted alphanumerically within each group; you can turn off exact matches on the Listed Candidates, Candidate Ordering options page.

Tables lists tables from the current database.

System tables are displayed if you have chosen to display system objects.

Columns lists column names, with their data types and associated table names (or table aliases). Primary key and foreign key columns are also shown.

Views lists views from the current database.

System views are listed if you have chosen to display system objects.

Stored procedures lists stored procedures from the current database.

System stored procedures are listed if you have chosen to display system objects.

Functions lists user-defined functions from the current database, and built-in functions.

System functions are displayed if you have chosen to display system objects.

Other candidate lists keywords, data types, and objects such as users and roles. For a full list of object types, see Object types below.

Object types

You may see the following types of object listed on the **All candidates** and **Other candidates** categories:

Tables	Rules
Views	Defaults
Stored Procedures	User Defined Types
Users	Functions
Roles	Full Text Catalogs
DML Triggers	

For SQL Server 2005:

Assemblies	Queues
Asymmetric Keys	Routes
Certificates	Schemas
Contracts	Services
Triggers	Service Bindings
Event Notifications	Symmetric Keys
Message Types	Synonyms
Partition Functions	XML Schema Collections

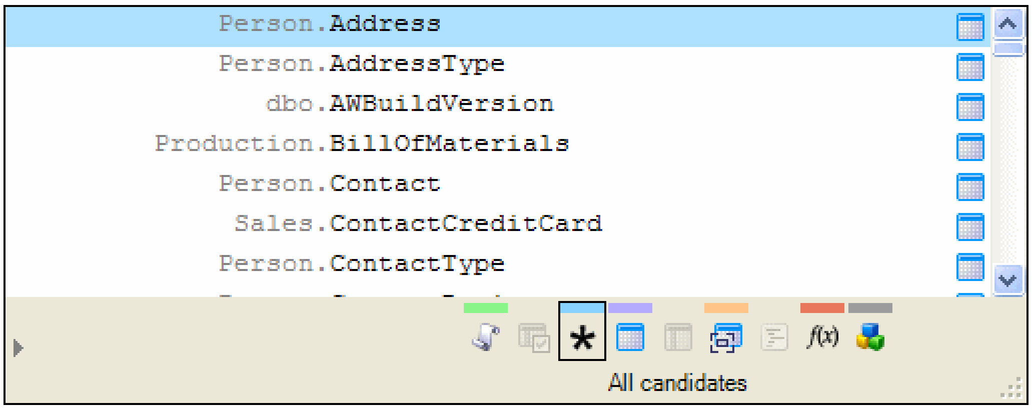
Partition Schemes

System objects

By default, SQL Prompt does not display system objects in candidate lists. If you want to display system objects, on the SQL Prompt menu, click **Options**, click the **Listed Candidates** tab, then from the list of pages on the left, click **Candidate Types and Filters**. In the **Candidate types** section, select the **System objects** check box.

Schema (owner) names

You can display schema (owner) names in the candidate list by clicking the left arrow button ◀ in the lower left corner of the candidate list.



```
      Person.Address
      Person.AddressType
         dbo.AWBuildVersion
  Production.BillOfMaterials
      Person.Contact
       Sales.ContactCreditCard
      Person.ContactType
```

To hide the list of schema (owner) names, click the right arrow button ▶

Note that when you select a candidate, the schema (owner) name is inserted in your query editor only if **Qualify object names** is selected on the **Inserted Candidates**, **Formatting** options page.

Customizing the candidate list

You can resize the candidate list by dragging the resize handle .⁝ or by specifying the height and width on the **Behavior**, **Candidate List Size and Font** options page.

You can also change the font used in the candidate list on this options page.

Moving through candidate list items

You can move through the list of items in the candidate list by pressing:

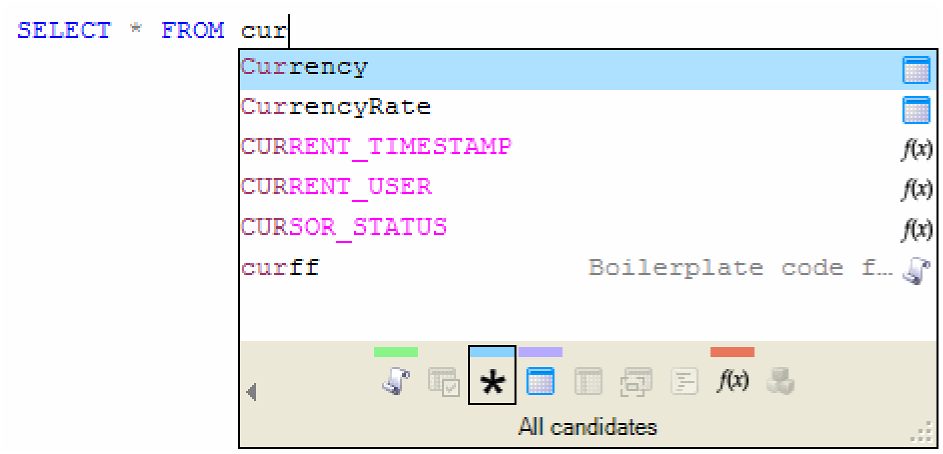
- UP ARROW or DOWN ARROW to move up or down the list one item at a time

  If you are at the top of the list, pressing UP ARROW takes you to the bottom of the list; if you are at the bottom of the list, pressing DOWN ARROW takes you to the top of the list.

- PAGE UP or PAGE DOWN to move up or down the list one page at a time
- CTRL+PAGE UP or CTRL+PAGE DOWN to move up or down the list one page at a time

Text filtering

You always filter the candidate list as you type the first few characters of a candidate. For example, typing `SELECT * FROM cur` displays all candidates relevant to SELECT that begin with *cur*.



```
SELECT * FROM cur
Currency
CurrencyRate
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR_STATUS
curff                 Boilerplate code f…
```

By default, SQL Prompt lists all possible candidates as you type and does not distinguish between upper and lower case. If you want to see only those candidates that match the case of your typing, select **Enable case-sensitive filtering of candidates** on the **Listed Candidates**, **Candidate Types and Filters** options page.

If a category does not contain any candidates that match the filter and the current context, it is shown as unavailable. In the example shown above, there are no view or stored procedure names that begin with *cur*.

Cross-database queries and distributed queries using linked SQL Servers

If you are writing cross-database distributed queries and cross-database queries using different databases on the same linked SQL Server, the candidate list can display the appropriate candidates for you. You can use this feature with databases on SQL Server 7, SQL Server 2000, and SQL Server 2005.

Some examples are shown in Worked examples (page 5).

If you do not want to use this feature, you can clear the **Enable cross-database and linked server support** check box on the **Listed Candidates**, **Cross-database Support** options page to improve performance and reduce memory usage.

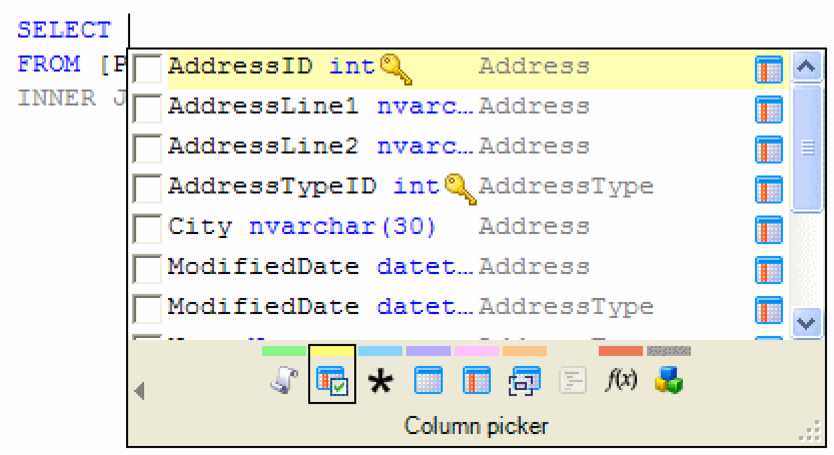
Selecting candidates for completion

When you have located the required candidate, double-click it or press ENTER to insert it into your query editor.

You can select other completion keys to insert candidates in your query editor on the **Behavior**, **Completion Keys** options page. For example, you can choose to press TAB, or SPACEBAR, as well as or instead of ENTER.

Using the column picker

The column picker enables you to select multiple columns for tables you have specified in a SELECT statement. The column picker also enables you to select columns for a table you have specified in an INSERT statement.



```
SELECT
FROM [P
INNER J
```

☐	AddressID	int	Address
☐	AddressLine1	nvarc…	Address
☐	AddressLine2	nvarc…	Address
☐	AddressTypeID	int	AddressType
☐	City	nvarchar(30)	Address
☐	ModifiedDate	datet…	Address
☐	ModifiedDate	datet…	AddressType

For an example of how to select columns for the column list in a SELECT statement, see Worked examples (page 5).

Column insertion

- The columns are inserted in the order in which you selected them.
- By default, the columns are inserted on separate lines.

  To insert the columns on a single line, clear the **When inserting multiple columns, use a new line for each** check box on the **Inserted Candidates**, **Formatting** options page.
- The columns are qualified if any of the following apply:
  - you have selected **Qualify column names** on the **Inserted Candidates**, **Formatting** options page
  - you have specified more than one table in the FROM clause

  When alias assignment (page 36) is switched off on the **Auto Insert**, **Aliases** options page, columns are qualified with the table name and the schema (owner) name. For example:

```
SELECT [Person].[AddressType].[Name],
       [Person].[Address].[AddressLine1],
       [Person].[Address].[AddressLine2],
       [Person].[Address].[City],
        [Person].[Address].[PostalCode]
FROM [Person].[Address]
INNER JOIN [Person].[AddressType]
```

When alias assignment is turned on, columns are qualified with the table alias. For example:

```
SELECT at.[Name],
       a.[AddressLine1],
       a.[AddressLine2],
       a.[City],
       a.[PostalCode]
FROM [Person].[Address] AS a,
INNER JOIN [Person].[AddressType] AS at
```

SQL Prompt assigns aliases when the table name is inserted.

- The columns are inserted according to the formatting you have specified on the **Inserted Candidates**, **Formatting** options page; by default, SQL Prompt does not surround identifiers with brackets [ ].

Note that for SELECT * or SELECT *table*.* statements, you can replace * with all of the columns in the referenced tables, for more information see Worked examples (page 5).

Inserted Candidates

When you select an item from the candidate list, SQL Prompt inserts the candidate according to the options you have set on the **Inserted Candidates** tab of the **Options** dialog box. For example, you can specify the case in which you want keywords to be inserted, and you can define options for INSERT statements.

The options on **Inserted Candidates** are only applied to text inserted by the candidate list and to certain code that you type in. These options do not configure the **Layout** feature of SQL Prompt Pro. For information on this feature, see Laying out SQL code (page 32).

Support for scripted objects created in the current query

When you create a table, view, function, or stored procedure in the query editor using the **CREATE <objectname>** statement, SQL Prompt adds these objects to the candidate list; you can then select the created objects from the candidate list as you continue to write code. For example, if you create a table, the table and its columns will appear as items in the candidate list when you create a view based on that table later in the same query.

Temporary tables are also shown in the candidate list while the query is being edited.

If the items are not shown in the candidate list, move the cursor to the section of SQL containing the script that creates the object. This forces SQL Prompt to add the items to the candidate list.

Objects that you delete from the script may still appear in the candidate list.

Comments

While you are typing comments, only the **Snippets** category is available in the candidate list.

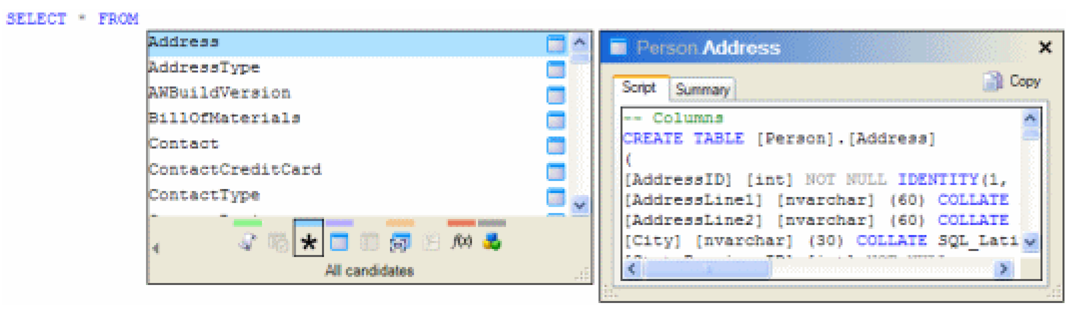
Unexpected results

The candidate list may show unexpected results in some circumstances. For example, this may occur when you type SQL statements that contain invalid syntax and SQL Prompt is unable to parse the statement.

For more information, see Troubleshooting.

Using the schema panel

SQL Prompt displays object definitions in a *schema* panel. For example, when you select a table name in the candidate list (page 21), the schema panel shows the creation SQL and a summary of the object.



```
SELECT * FROM
```

Address
AddressType
AWBuildVersion
BillOfMaterials
Contact
ContactCreditCard
ContactType

```
-- Columns
CREATE TABLE [Person].[Address]
(
[AddressID] [int] NOT NULL IDENTITY(1,
[AddressLine1] [nvarchar] (60) COLLATE
[AddressLine2] [nvarchar] (60) COLLATE
[City] [nvarchar] (30) COLLATE SQL_Lati
```

You can specify the behavior of the schema panel on the **Behavior**, **Information Panels** options page. To display schema panel only when the object definition hint is shown, select the **Hide** check box.

For tables and views, the **Summary** tab shows the column names and data types. For stored procedures and functions, it displays parameters and their data types.

For some objects, the **Summary** tab is not displayed. For example, for snippets, only the pre-defined code is shown on the **Script** tab.

Note that if you are using SQL Server 2005 or SQL Server 2008, SQL Prompt cannot display schema information for:

- encrypted objects

  SQL Prompt displays the message *Text was encrypted* in the schema panel.
- system functions
- system stored procedures
- some system views

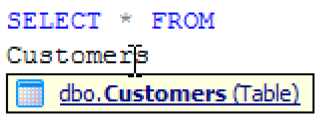
You can copy the schema information to the clipboard:

- To copy all the schema information, ensure that there is no text selected, and click **Copy to Clipboard**.
- To copy part of the schema information, select the required text, and click **Copy to Clipboard**.

To resize the schema panel, drag the resize handle .:

Using object definition hints

SQL Prompt can display object definitions in a hint. When you point to an object, the hint displays information about the object.



```
SELECT * FROM
Customers
```

To manage object definition hints, on the **SQL Prompt** menu, click **Options**, and then click the **Behavior** tab and the **Information Panels** page. Under **Pop-up hints**, you can select or clear the **Enable object definition hints** check box to switch the feature on or off.

Hints are displayed for the objects in the list below.

- tables
- views
- stored procedures
- databases
- data types
- snippets
- triggers
- local variables (type)
- local parameters (type)
- columns (table name and type)
- built-in functions (function signature)
- user-defined functions (function signature)

If the object definition is underlined, you can click it to display the schema panel (page 29) and see more detailed information on the object.

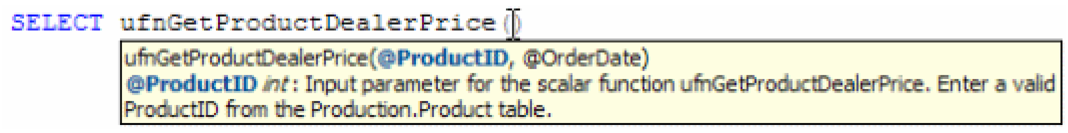
SQL Prompt also displays parameter hints (page 31) for user-defined functions.

Using parameter hints

SQL Prompt can display the parameters of a user-defined function in a hint.

To display a hint for a user-defined function that is already in your SQL code, click within the function's argument parentheses to see the parameters that the function takes.

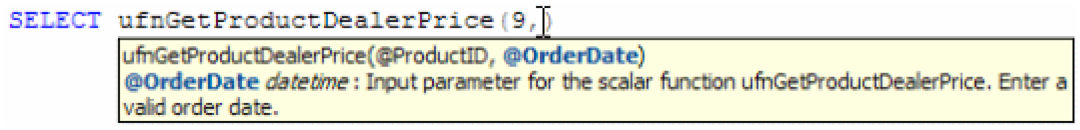
When you insert the function from the candidate list, the cursor is automatically placed within the function's argument parentheses and the hint is displayed.



```
SELECT ufnGetProductDealerPrice ()
```

ufnGetProductDealerPrice(**@ProductID**, @OrderDate)
@ProductID *int* : Input parameter for the scalar function ufnGetProductDealerPrice. Enter a valid ProductID from the Production.Product table.

As you enter information, the parameter hint changes automatically to remain in context.



```
SELECT ufnGetProductDealerPrice (9,)
```

ufnGetProductDealerPrice(@ProductID, **@OrderDate**)
@OrderDate *datetime* : Input parameter for the scalar function ufnGetProductDealerPrice. Enter a valid order date.

To manage parameter hints, on the **SQL Prompt** menu, click **Options**, and then click the **Behavior** tab and the **Information Panels** page. Under **Pop-up hints**, you can select or clear the **Enable parameter hints** check box to switch the feature on or off.

SQL Prompt also displays object definition hints (page 30) for objects.

Laying out SQL code

Lay Out SQL applies configurable refactoring rules to existing SQL code to make it more readable without changing the behavior of the code. **Lay Out SQL** is only available in SQL Prompt Pro.

Setting the layout options

To ensure layout options are correct before SQL Prompt lays out an existing script, on the **Options** dialog box, click the **Layout** tab.

The **Layout** tab has the following pages:

- **Case**
- **Schema Statements**
- **Data Statements**
- **Expressions**
- **Commas and Parentheses**
- **Tab Size and Wrapping**

Click on a page, then use the **Preview** section to see the effect changing an option has on the example script.

Note that the layout options affect only the behavior when you click **Lay Out SQL**.

To customize text that SQL Prompt inserts as you type or inserts from the candidate list, on the Options dialog box, click the **Inserted Candidates** tab and click **Formatting**, **Case** and **Tab Size** respectively.

For more information about how to customize text you insert or type, see Inserted candidates in Using the candidate list (page 21).

Laying out SQL

To lay out the SQL code in the current query editor or the highlighted SQL code fragment, on the **SQL Prompt** menu, click **Lay Out SQL**, or press CTRL+K, CTRL+Y. In Query Analyzer, press CTRL+SHIFT+Y.

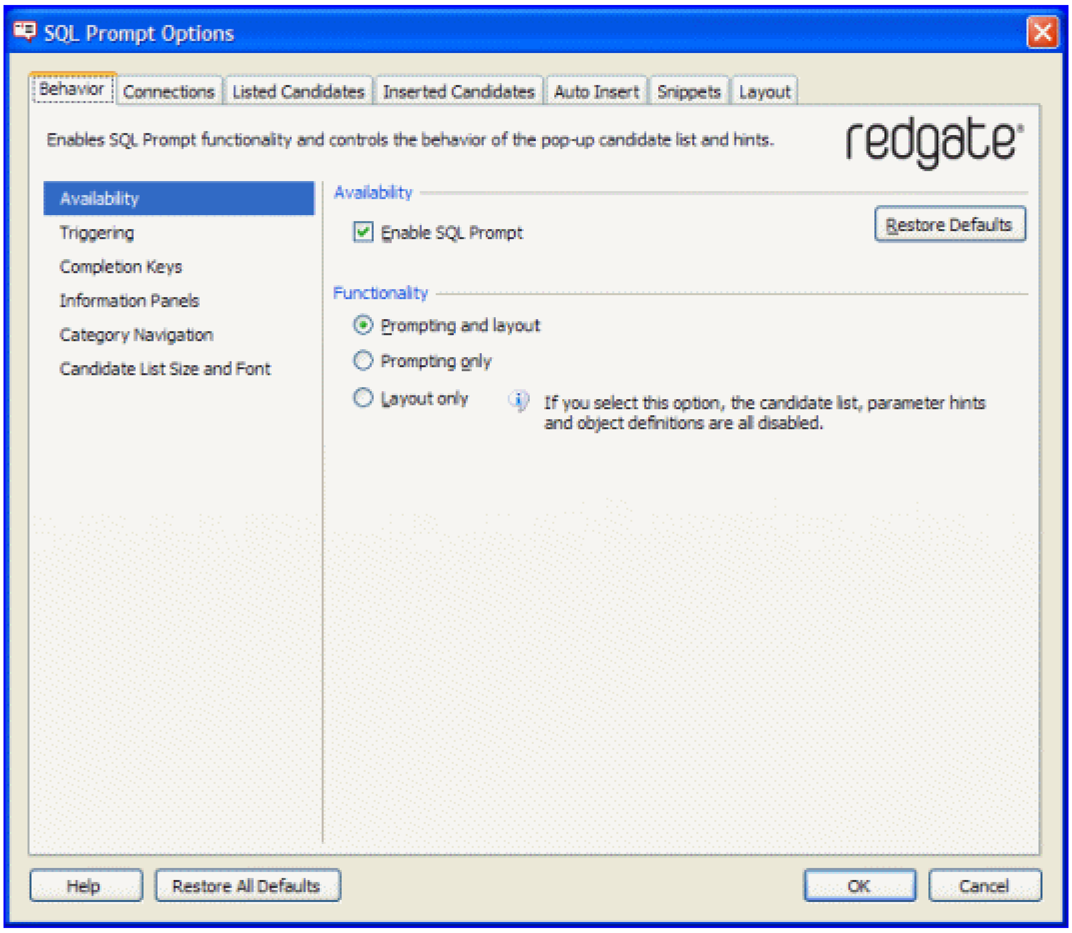
Note that not all SQL commands can be refactored by SQL Prompt. For unsupported commands only the wrapping option is applied. If there are syntax errors, a message is displayed, and the incorrect code is underlined in the query editor to help you fix the problem.

To switch off SQL Prompt's layout feature use the **Behavior**, **Availability** options page and select **Prompting only**.

Setting options

SQL Prompt offers a number of options for you to customize the behavior and appearance of SQL Prompt features.

To access the options, on the **SQL Prompt** menu, click **Options**.



Enables SQL Prompt functionality and controls the behavior of the pop-up candidate list and hints.

Availability

[✓] Enable SQL Prompt

Functionality

[✓ Prompting and layout] Prompting only Layout only

If you select this option, the candidate list, parameter hints and object definitions are all disabled.

The **Options** dialog box contains the following tabs:

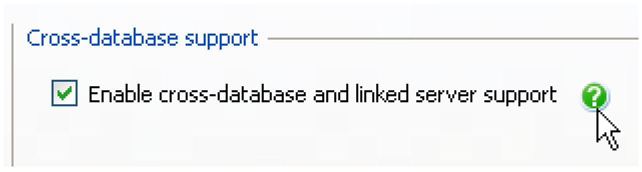
- **Behavior** provides options on SQL Prompt availability, the candidate list (page 21) and the schema panel (page 29).
- **Connections** provides options to manage your SQL Server and database connections (page 42).
- **Listed Candidates** provides options to control which candidates are displayed in the candidate list (page 21).
- **Inserted Candidates** provides options that define what SQL Prompt inserts in your query editor when you pick a candidate.
- **Auto Insert** lets you set up time-saving actions, such as aliases (page 36) and automated closing characters.
- **Snippets** provides options for you to manage code snippets (page 39).
- **Layout** (Pro version) enables you to configure formatting rules and lay out (page 32) an existing SQL script to make it more readable.

Most tabs contain a number of pages. To display a page, click the name of the page in the column on the left of the **Options** dialog box.

To reset all the options on a particular page to their default values, click **Restore Defaults** on that page.

To reset all the options on all of the pages to their default values, click **Restore All Defaults**.

You can display hints on individual items on the **Options** dialog box where you see the icon. For example:



Cross-database support

☑ Enable cross-database and linked server support

Click to display the hint; click to close the hint window.

Managing aliases

You can set SQL Prompt so that it automatically assigns an alias to each table and view that is referenced in a SQL statement. You can also define custom aliases for tables and views. If your table or view names contain prefixes, you can set SQL Prompt so that it ignores those prefixes when assigning aliases.

Where possible, SQL Prompt generates aliases using the first letter of the table or view name. SQL Prompt also takes into account:

- underscores

  TBL_Contact is assigned the alias *tc*

- hyphens

  hyphenated-tablename is assigned the alias *ht*

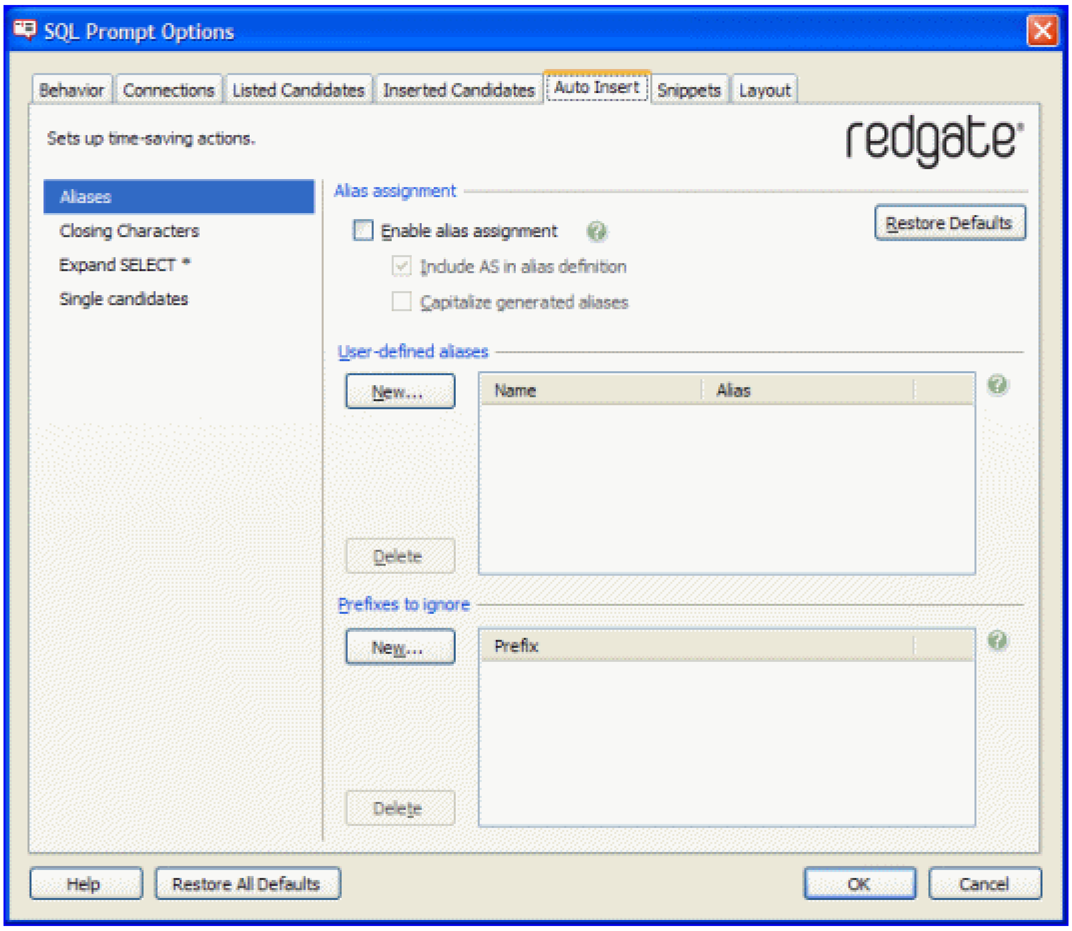
- case

  MixedCase is assigned the alias *mc*

SQL Prompt creates additional aliases where there is ambiguity, for example in self-joins:

```
SELECT DISTINCT pv.[VendorID], pv2.[ProductID], pv.[ProductID]

FROM [Purchasing].[ProductVendor] AS pv

INNER JOIN [Purchasing].[ProductVendor] AS pv2

ON pv.[ProductID] = pv2.[ProductID]

WHERE pv.[VendorID] <> pv2.[VendorID]
```

To manage alias assignment, on the **SQL Prompt** menu, click **Options**, and then click the **Auto Insert** tab and the **Aliases** page.



Alias assignment

When the **Enable alias assignment** check box is selected, SQL Prompt assigns an alias to tables and views that are referenced in a SQL statement, provided that you have specified a list of columns or * to select all columns.

For example, if you select the column *FirstName* and then the table *Contact*, SQL Prompt creates the alias *c* where *c* represents the table name *Contact*.

```
SELECT [FirstName]

FROM [Person].[Contact] AS c
```

If you do not want SQL Prompt to include the AS keyword when it assigns aliases, clear the **Include AS in alias definition** check box. For example:

```
SELECT [FirstName]

FROM [Person].[Contact] c
```

When SQL Prompt assigns an alias, it remembers it for use in subsequent queries in the current query editor window. The candidate list displays the learned aliases at appropriate points in your query, for example when you are typing a WHERE clause or adding additional columns to your query.

If you assign a different alias to a table or view name, SQL Prompt remembers the alias you assigned. For example, if SQL Prompt assigns the alias *c* to the table *Contact*, you can overwrite the *c* with a different alias; SQL Prompt remembers this alias the next time you reference the table *Contact*.

To set SQL Prompt so that it learns aliases from SQL statements that you have pasted into your query editor window or from SQL that you have loaded from a file, on the **Listed Candidates**, **Performance** page select the **Search for objects and aliases when opening files or pasting text** check box. You are not recommended to select this option if you are working with large scripts.

User-defined aliases

If the aliases that SQL Prompt automatically generates do not satisfy your naming conventions, you can specify user-defined aliases for table or view names.

For example, if you specify the user-defined alias *Con* for the table *Contact*, SQL Prompt assigns the alias as follows:

```
SELECT [FirstName]

FROM [Person].[Contact] AS Con
```

To add a user-defined alias, under **User-defined Aliases**, click **New**. Then, on the **Alias Properties** dialog box, type the name of the table or view in the **Object name** box and the alias in the **Alias** box, and then click **Save**.

To delete a user-defined alias, select the alias that you want to delete, and click **Delete**.

Prefixes to Ignore

You can specify prefixes that SQL Prompt will ignore when assigning aliases for column, table, or view names.

For example, if you specify *TBL_* as a prefix to ignore and you have a table called *TBL_Orders*, SQL Prompt considers only *Orders* when assigning an alias for the table name.

To add a prefix to the list, under **Prefixes to Ignore**, click **New**. Then, on the **Prefix to Ignore** dialog box, type the prefix in the **Prefix** box and click **Save**.

To delete a prefix from the list, select the prefix that you want to delete, and click **Delete**.

Managing snippets

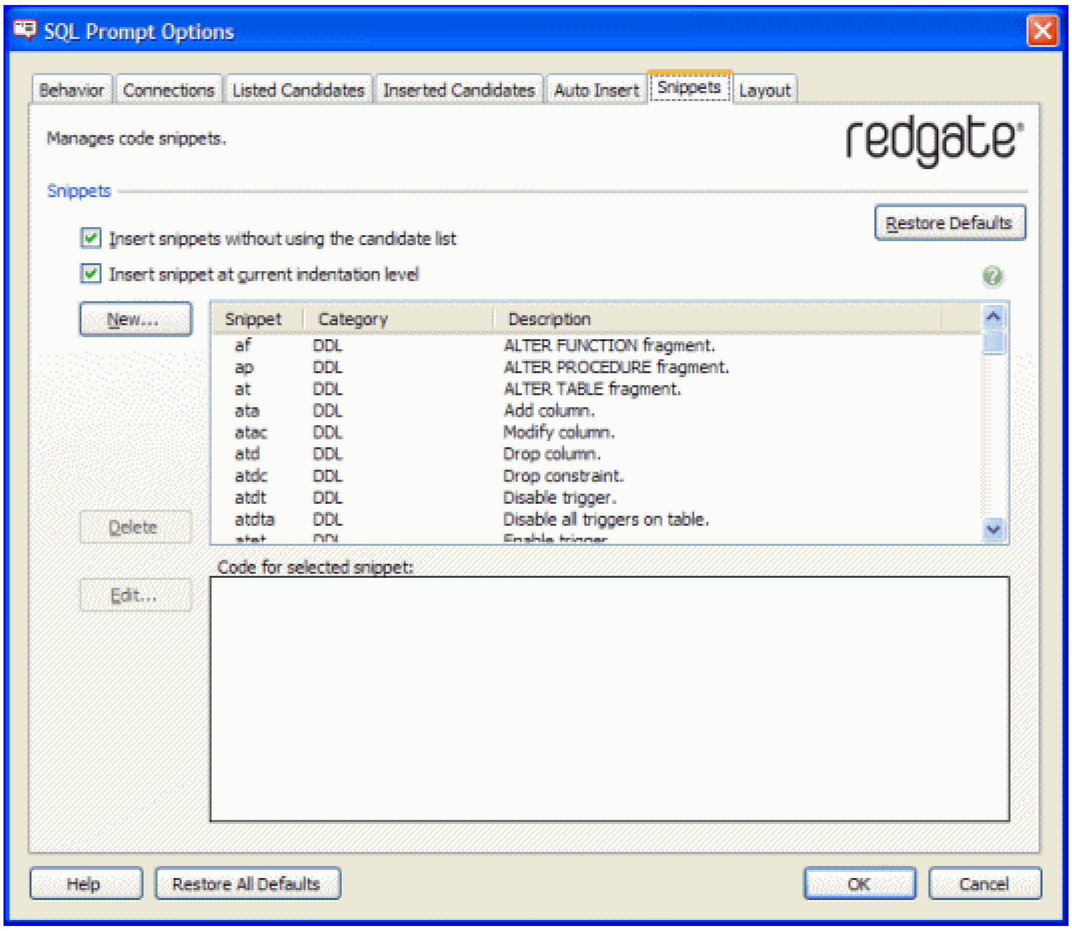
Snippets enable you to insert pre-defined code into your query editor. To insert snippet code:

- type the snippet name and press TAB
- or if you are displaying the candidate list automatically, type the snippet name and press the completion key (by default, this is ENTER)
- or to see a list of available snippets, press CTRL+SPACEBAR to display the candidate list, and select the **Snippets** category

SQL Prompt is pre-configured with default snippets. For example, the *ssf* snippet inserts a SELECT * FROM fragment; the *cdb* snippet inserts a CREATE DATABASE statement and places the insertion point after the CREATE DATABASE fragment for you to specify the name of the new database. You can edit or delete the default snippets on the **Snippets** options page.

You can create your own snippets of code, including specifying the position at which SQL Prompt will place the cursor after it has inserted the snippet of code.

To manage snippets, on the **SQL Prompt** menu, click **Options**, and then click the **Snippets** tab.



Manages code snippets.

Snippets

☑ Insert snippets without using the candidate list

☑ Insert snippet at current indentation level

Snippet	Category	Description
af	DDL	ALTER FUNCTION fragment.
ap	DDL	ALTER PROCEDURE fragment.
at	DDL	ALTER TABLE fragment.
ata	DDL	Add column.
atac	DDL	Modify column.
atd	DDL	Drop column.
atdc	DDL	Drop constraint.
atdt	DDL	Disable trigger.
atdta	DDL	Disable all triggers on table.
atet	DDL	Enable trigger

Code for selected snippet:

The **Snippets** tab displays a list of default and custom snippets, listed in alphanumeric order. To sort by category or description, click the column header. You can view the code for a snippet by clicking it.

If you do not want to use TAB to insert snippets, clear **Insert snippets without using the candidate list**.

By default, the snippet code is inserted at the indentation level of your SQL code at the point of insertion. If you do not want the snippet code to be indented, clear the **Insert snippet at current indentation level** check box.

To restore the default snippets if you have edited or deleted them, click **Restore Defaults**. This also restores the default settings for the check boxes.

Creating custom snippets

1. On the **SQL Prompt** menu, click **Options**, click the **Snippets** tab then click **New**.
2. Type a name for your snippet in the **Snippet** box.

This is the text that you will type in your query editor window when you want to use the snippet.

3. Select a **Category**:

   You can use categories to organize your snippets. Choose from the following categories or select **New** to create your own:

   - Access Control
   - Administrative
   - Analysis Services
   - Control Flow
   - DDL
   - DML
   - Miscellaneous

4. Type a short description of your snippet in the **Description** box.

   The description helps you to identify a snippet if you are unsure of the snippet name.

5. Type or paste the SQL code in the **Code** box.

   To specify the insertion point at which you want the cursor to be placed when the snippet is inserted, use the variable *$CURSOR$*. For example:

   ```
   SELECT $CURSOR$ FROM
   ```

   places the insertion point after SELECT.

6. Click **Save**.

Editing snippets

To edit a snippet, on the **SQL Prompt** menu, click **Options**, click the **Snippets** tab then select the snippet that you want to edit and click **Edit**, or double-click the snippet. Then, on the **Snippet Properties** dialog box, change the details as required and click **Save**.

Deleting snippets

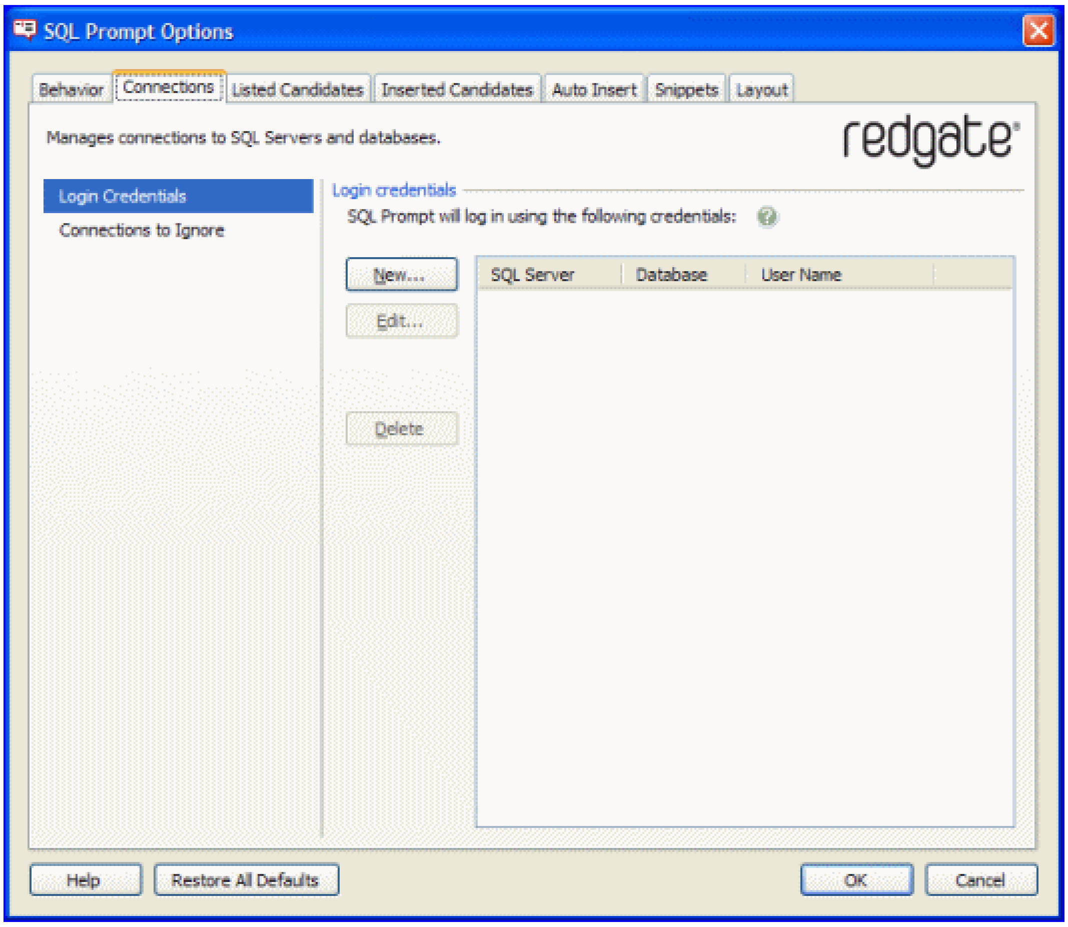
To delete a snippet, on the **SQL Prompt** menu, click **Options**, click the **Snippets** tab then select the snippet that you want to delete and click **Delete**.

Managing connections

Occasionally, it is necessary for SQL Prompt to connect to databases that you are using. For example, SQL Prompt connects to the database when you switch databases by typing USE and you select a database from the candidate list, or when you refresh a cache file. By default, SQL Prompt uses the connection details you specified when you opened your query editor. However, there may be times when you are prompted to enter login credentials, for example if SQL Prompt is denied access to a SQL Server or database.

You can save login credentials for SQL Prompt to use so that you are not prompted to enter them again. You can also specify SQL Servers and databases to which you do want SQL Prompt to connect.

To manage database connections, on the **SQL Prompt** menu, click **Options**, and then click the **Connections** tab.



Manages connections to SQL Servers and databases.

SQL Prompt will log in using the following credentials:

Login credentials

You can specify the login credentials that you want to be used whenever SQL Prompt connects to a SQL Server or database. This means that you will not be prompted if access is denied.

To add login credentials, click **New**, and on the **Login Credentials** dialog box enter the connection details as required. If you want SQL Prompt to use the same connection details for all databases on the SQL Server, select **All databases**; to specify the connection details for a single database on the SQL Server, select **This database** and type the name of the database. For **SQL Server authentication**, SQL Prompt encrypts the password and saves it on your computer. When you click **Save**, the details are summarized on the **Login Credentials** page.

To edit login credentials that you have previously saved, double-click the details in the list, or click the details and click **Edit**. For example, you may want to do this if you have changed the password for a SQL Server login.

To delete login credentials that you have previously saved, click the details in the list, and click **Delete**.

Connections to ignore

SQL Prompt stores information about the structure of a database in a cache file. If you do not want to cache information for a particular database, you can set SQL Prompt to ignore the database. For example, you may want SQL Prompt to ignore a large database to reduce memory usage.

You can also set SQL Prompt to ignore all databases for a particular SQL Server.

Note that SQL Prompt does not display information about ignored databases in the candidate list.

Click **New**, and on the **SQL Server or Database to Ignore** dialog box enter the name of the SQL Server. If you want SQL Prompt to ignore all databases on the SQL Server, select **All databases**; to ignore a single database on the SQL Server, select **This database** and type the name of the database. When you click **Save**, the details are summarized on the **Connections to Ignore** page.

To edit an ignored item, click the appropriate details in the list and click **Edit**; to delete an item, click the details and click **Delete**.

Connecting to a database in Visual Studio

When editing a SQL file in Visual Studio, you must be connected to a database for SQL Prompt to show database objects in the candidate list. How you connect depends upon the context in which you are using SQL Prompt.

Creating a database project:

- When you create a Database project, you are prompted for the database to connect to. You can connect to a different database by creating a new Database Reference and setting it as the project default, or by using **Change Connection** on the **SQL Prompt** menu.

Generating a script from Server Explorer:

- When you generate a SQL script from a database object using Server Explorer, you are connected to that database using the same connection details as specified in Server Explorer.

Opening a SQL file:

- When you open a SQL file in Visual Studio, outside of a database project, you must choose the database to connect to. From the **SQL Prompt** menu, select **Connect to Server**.

Connecting to a database in Visual Studio Team Edition for Database Professionals

In Visual Studio Team Edition for Database Professionals, you can connect to a server when editing a SQL script. In this case, SQL Prompt uses that connection and reads the database information to populate the candidate list. You can also connect directly using SQL Prompt. From the **SQL Prompt** menu, select **Connect to Server**.

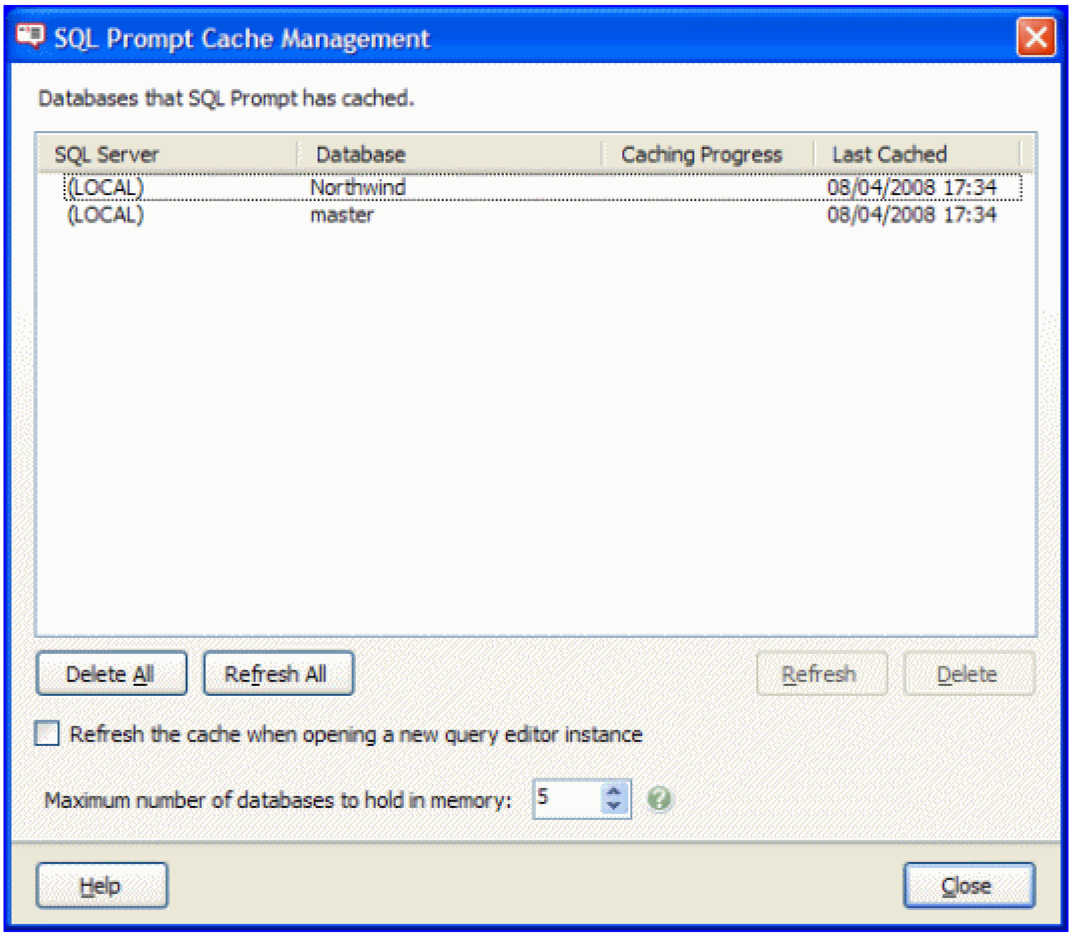
Managing the SQL Prompt cache

SQL Prompt stores information about the structure of databases in cache files. Using cache files speeds up performance when SQL Prompt retrieves information about a database.

SQL Prompt creates a cache file for each database that you use. The cache files are encrypted and stored in your local settings on your computer; SQL Prompt can read only the cache files that were created by your Windows® user.

Note that SQL Prompt does not create cache files for databases you have chosen to ignore (page 42) on the **Connections**, **Connections to Ignore** options page.

To manage the cache files, on the **SQL Prompt** menu, click **Cache Management**.



Databases that SQL Prompt has cached.

SQL Server	Database	Caching Progress	Last Cached
(LOCAL)	Northwind		08/04/2008 17:34
(LOCAL)	master		08/04/2008 17:34

☐ Refresh the cache when opening a new query editor instance

Maximum number of databases to hold in memory: 5

Refreshing cache files

If the structure of a databases has been modified, for example if a table has been added to the database, you can refresh the cache files so that the changes are shown in the candidate list. SQL Prompt updates the cache files ready for you to use immediately.

To refresh the cache file for the database you are using in your query editor, on the **SQL Prompt** menu, click **Refresh Cache**, or press CTRL+SHIFT+D.

You can set SQL Prompt so that it refreshes cache files whenever you connect to a database or open a new query editor session. You do this by selecting the check box on the **Cache Management** dialog box.

To refresh selected cache files, on the **Cache Management** dialog box, select the databases for which you want to update the cache, and then click **Refresh**.

To refresh all cache files, on the **Cache Management** dialog box, click **Refresh All**. The **Caching Progress** bar shows the progress of the refresh. This process may be time-consuming if you have many databases cached, or if you have large databases cached. However, you can close the **Cache Management** dialog box and continue working while the cache is refreshing.

Note that if a database is being restored or it is offline, you cannot refresh its cache file.

Deleting cache files

You may want to delete cache files, for example to create disk space. If you delete the cache file for a database, SQL Prompt creates a new cache file the next time you use the databases in your query editor.

To delete individual cache files, on the **Cache Management** dialog box, select the databases for which you want to delete the cache, and then click **Delete**.

To delete all the cache files, on the **Cache Management** dialog box, click **Delete All**.

Maximum number of databases to hold in memory

You can change the number of databases to be cached. For example, you may want to increase the number if you are using cross-database support and you regularly query a number of different databases.

However, note that caching databases increases memory usage, so if your memory usage is high you may want reduce this number.

The minimum value for this option is 3.

Tips

This topic provides information to help you to optimize SQL prompt for use with:

- databases that contain a large number of objects
- databases with complicated schemas
- large scripts

Using SQL Prompt with large databases

If you are using SQL Prompt with databases that contain several thousand objects and you are experiencing very high memory usage, you are recommended to set the SQL Prompt options on the **Listed Candidates**, **Candidate Types and Filters** options page as follows:

- Clear the **Filter candidate list by owner/schema prefix** check box.
- Clear the **Enable case-sensitive filtering of candidates** check box (this is cleared by default).
- Clear the **List all columns in database after SELECT** check box (this is cleared by default).

On the **Cache Management** dialog box, set the **Maximum number of databases to hold in memory** to *3*.

You may also wish to clear the **Enable cross-database and linked server support** check box on the **Listed Candidates**, **Cross-Database Support** options page.

Using SQL Prompt with databases with complicated schemas

If you use SQL Prompt with a database that has a complicated schema, you may experience some performance issues because of the amount of memory that SQL Prompt needs to use. If you have insufficient RAM installed in your PC, the performance of Windows and all your applications may degrade because the operating system has to swap data in and out of memory, onto disk and back again.

If you want to take full advantage of the features that SQL Prompt offers, you are recommended to upgrade your PC to increase the amount of physical memory to a minimum of 1 GB; for very complicated schemas, 2 GB may be required.

If you are unable to upgrade your PC, you can change your options and settings to reduce memory usage, as follows. However, this means that you will not be able to take full advantage of all of SQL Prompt's features.

- Reduce the number of databases cached in memory.

  When you reduce the number of cached databases, SQL is more likely to have to pull information for databases from disk, which is slower. Therefore, if you often work with multiple databases, you may wish to experiment with a higher number of cached databases to find the optimum performance.

On the **Cache Management** dialog box, set the **Maximum number of databases to hold in memory** to *3*.

- 'Ignore' databases for which you are not interested in seeing information in the candidate list.

  SQL Prompt does not cache information about databases you have chosen to ignore, so that memory is not used on these databases. The benefit is greater for larger (more complex) databases.

  On the **Options** dialog box, click **Connections**, and then click **Connections to Ignore**. Click **New** to specify the databases that you want to ignore. For more information about this option, see Managing connections (page 42)

- For SELECT statements, show columns only for the tables and views that you specified in the SQL statement.

  SQL Prompt provides an option to show all columns in the database when the candidate list is displayed after the SELECT keyword. By default, this option is not selected and columns are displayed after the SELECT keyword only when you have specified a table. If you are working with complex databases, you are recommended **not** to select this option.

  To ensure that the option is not selected, on the **Options** dialog box click **Listed Candidates**, and then click **Candidate Types and Filters**. If necessary, clear the **List all columns in database after SELECT** check box.

- Do not filter the candidate list by schema (owner).

  SQL Prompt provides an option to filter the candidate list by schema (owner), so that only candidates that belong to that schema (owner) are shown. This is very useful if you are working with SQL Server 2005 or SQL Server 2008 and you have a large number schemas to which objects might belong.

  However, if most of your objects belong to the same schema (owner), this option is less useful and you may want to switch it off. For example, this may be the case if you are using SQL Server 2000 and your objects belong to the owner *dbo*. Switching off this option can reduce the amount of memory that SQL Prompt uses by up to 10%.

  To ensure that the option is not selected, on the **Options** dialog box click **Listed Candidates**, and then click **Candidate Types and Filters**. If necessary, clear the **Filter candidate list by owner/schema prefix** check box.

- Do not filter the candidate list by case.

  SQL Prompt provides an option to filter the candidate list so it shows only those candidates that match the case of your typing. By default, this option is not selected and candidates are listed irrespective of case. If you have switched this option on and you have a lot of object names that use mixed case, switching it off will reduce memory usage. To ensure that the option is not selected, on the **Options** dialog box click **Listed Candidates**, and then click **Candidate Types and Filters**. If necessary, clear the **Enable case-sensitive filtering of candidates** check box.

Using SQL Prompt with large scripts

SQL Prompt is optimized for use with large scripts when you first install it. However, if you experience slow performance, for example because you have a slow processor, you can reduce the number of lines of SQL code that SQL Prompt parses when it populates the candidate list.

To do this, on the **Options** dialog box, click **Listed Candidates**, **Performance**. Ensure that **Search a fixed number of lines from the caret** is selected, and then reduce the number of lines to search for variables and parameters. Decreasing the number of lines speeds up performance. However, if you reduce this value too much you may experience problems; for details, see Troubleshooting.

When you open a SQL file or paste text into your query editor, SQL Prompt will automatically parse the new code for any scripted objects and assigned aliases. For large files and scripts, this may take a few seconds. If you do not want SQL Prompt to scan pasted text or opened SQL files, in the **Listed Candidates**, **Performance** page, clear the **Search for objects and aliases when opening or pasting text** check box.

Note that SQL Prompt stops searching an opened file or a pasted block of text after five seconds. For very large scripts, this may not be long enough to parse all the scripted objects and aliases. You can increase the length of time SQL Prompt scans the text; to do this, increase the value of the *MaximumScanTimeMilliseconds* property in the *EngineOptions.xml* file from its default value of 5000 milliseconds. The *EngineOptions.xml* file is stored by default in your local user profile's application settings, for example *C:\Documents and Settings\<username>\Local Settings\Application Data\Red Gate\SQL Prompt 3*. To specify that SQL Prompt should always scan the whole file or block of text with no time limit, set the value to 0.

Troubleshooting

This topic provides information that may help you if you are experiencing problems with SQL Prompt.

Candidate list displays incorrect candidates

When SQL Prompt parses SQL code to populate the candidate list (page 21), the number of lines of code parsed is defined by the number of lines specified in **Search a fixed number of lines from the caret**. This setting is on the **Listed Candidates**, **Performance** options page.

If you decrease this value, performance is improved because SQL Prompt parses less code. However, if you have a statement that spans more lines than are parsed, SQL Prompt cannot populate the candidate list correctly. You may also see unexpected results if SQL Prompt parses some, but not all, of a BEGIN... END block.

In general, it is good practice to refactor very large stored procedures or functions into smaller blocks. If you do this, SQL Prompt is less likely to encounter parsing problems.

If you work only with small scripts (less than 100 lines of SQL code), or with large scripts that have batch markers every 50 or 100 lines, you can select **Search entire batch/GO block** on the **Listed Candidates**, **Performance** options page. SQL Prompt can then display an accurate list of candidates without slowing performance.

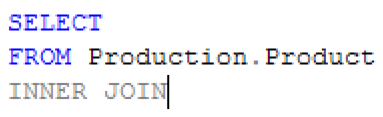
The candidate list may also display unexpected results when you type SQL statements that contain invalid syntax, because SQL Prompt may be unable to parse the statement.

When you open a script file or paste SQL into the query editor, SQL Prompt will automatically parse the code to populate the candidate list with any scripted objects (for example, tables or views that are defined using a CREATE statement) and aliases. If the candidate list does not display these objects and aliases, ensure that the **Search for objects and aliases when opening or pasting text** check box in the **Listed Candidates**, **Performance** page is selected.

Note that SQL Prompt will stop parsing the code after a fixed number of seconds. This is to reduce the impact on performance when handling very large scripts. You can increase the time SQL Prompt parses the code; for more information, see Using SQL Prompt with large scripts.

Candidate list incorrect for JOIN clause

The following SQL code snippets may cause SQL code to display incorrect candidates in the candidate list when you press CTRL+SPACEBAR at the insertion point:



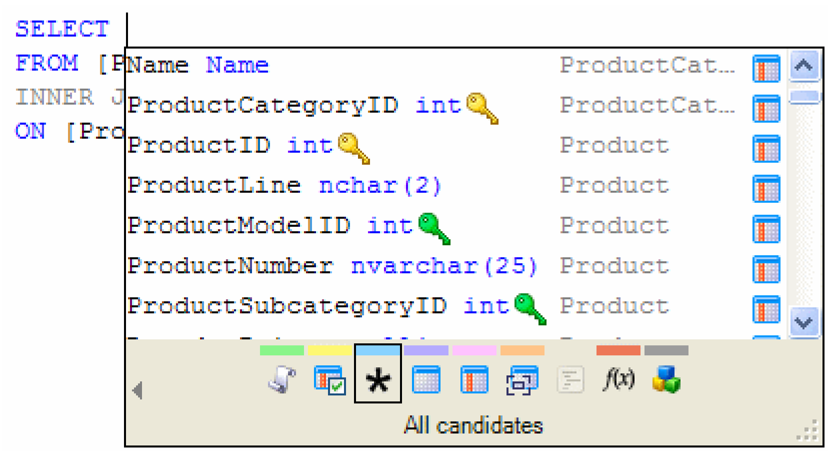
```
SELECT
FROM Production.Product
INNER JOIN
```

and:

```
SELECT
FROM Production.Product
INNER JOIN Production.ProductCategory
ON 
```

This is because you have not followed the SELECT keyword with * or a list of columns, therefore the SELECT statement is invalid.

You may have omitted the list because you intend to add it later using SQL Prompt. However, you must always type * after the SELECT keyword, even when you want use SQL Prompt to insert the column list later. You can then go back to the SELECT keyword, delete the *, and press CTRL+SPACEBAR to display the candidates.



```
SELECT
FROM [P
INNER J
ON [Pro
```

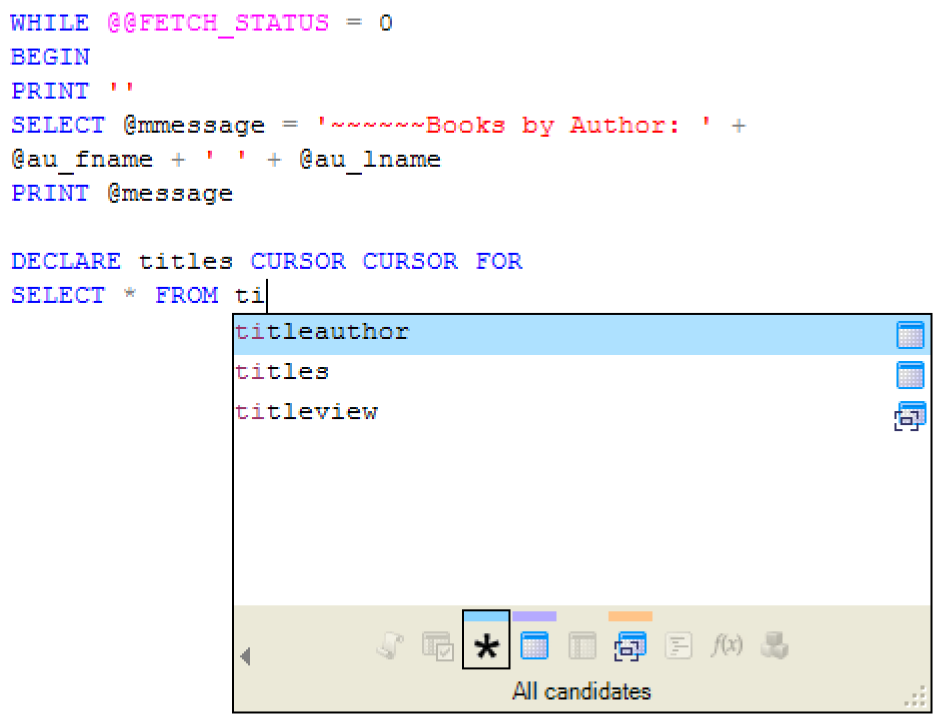
```
Name Name                       ProductCat...
ProductCategoryID int           ProductCat...
ProductID int                   Product
ProductLine nchar(2)            Product
ProductModelID int              Product
ProductNumber nvarchar(25)      Product
ProductSubcategoryID int        Product
```

For examples, see Worked examples (page 5).

Alias not assigned to table following unterminated BEGIN...END block

If you do not terminate a BEGIN... END clause in a WHILE loop, SQL Prompt does not recognize the WHILE loop and therefore may not generate aliases even though **Enable alias assignment** is selected on the **Auto Insert**, **Aliases** options page.

The following example uses the *pubs* database on SQL Server 2000. In this SQL code, the BEGIN clause is not terminated:



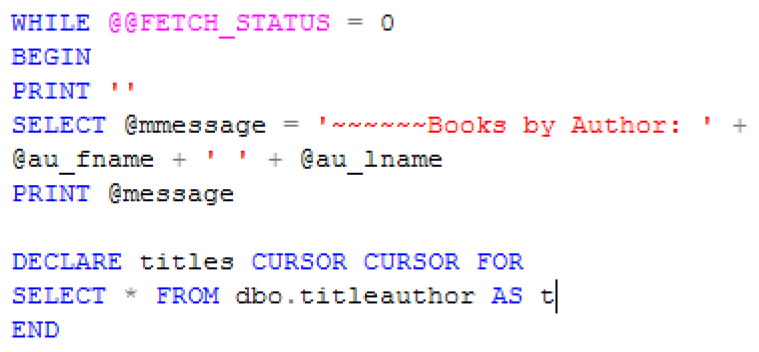
```
WHILE @@FETCH_STATUS = 0
BEGIN
PRINT ''
SELECT @mmessage = '~~~~~~Books by Author: ' +
@au_fname + ' ' + @au_lname
PRINT @message

DECLARE titles CURSOR CURSOR FOR
SELECT * FROM ti
```

```
titleauthor
titles
titleview
```

When the table is selected, SQL Prompt inserts the table name without assigning the alias:

```
WHILE @@FETCH_STATUS = 0
BEGIN
PRINT ''
SELECT @mmessage = '~~~~~~Books by Author: ' +
@au_fname + ' ' + @au_lname
PRINT @message

DECLARE titles CURSOR CURSOR FOR
SELECT * FROM dbo.titleauthor
```

If the END keyword is included in the SQL code, SQL Prompt assigns an alias to the table.



```
WHILE @@FETCH_STATUS = 0
BEGIN
PRINT ''
SELECT @mmessage = '~~~~~~Books by Author: ' +
@au_fname + ' ' + @au_lname
PRINT @message

DECLARE titles CURSOR CURSOR FOR
SELECT * FROM dbo.titleauthor AS t
END
```

SQL Prompt is showing an old version of a stored procedure

When displaying the creation SQL for an object in the schema panel (page 29), or inserting SQL code into the query editor window, SQL Prompt uses the version of the SQL in its cache file; it does not retrieve the latest version from the database. This means that the SQL code may be out of date.

Because of this, if you auto insert an object definition after the ALTER statement and you have not updated your cache, it is possible to accidentally overwrite the new version of a stored procedure with the old version stored in the cache file. Therefore, when you are editing SQL code on a database for which the schema may have changed, ensure you refresh the cache before you use the auto insertion features.

For more information about caching databases, see Managing the SQL Prompt cache.

A new table does not appear in the candidate list

If the schema of the database you are working on has changed, new objects may not appear in the candidate list until you refresh the cached information for that database.

To manually refresh the cache:

- From the **SQL Prompt** menu, select **Cache Management**. Select the database, and click **Refresh**.

To automatically refresh the cache:

- If the database you are working with changes regularly, you can select **Refresh the cache when opening a new query editor instance**. This will force SQL Prompt to update the cache file whenever you start your editing session.

SQL Prompt menu needs to be restored

If you are using SQL Server Management Studio, and you need to restore the SQL Prompt menu, run `sqlwb.exe /setup` at the command prompt. Note that this restores all the SQL Server Management Studio menus.

Quoted identifiers

SQL Prompt does not support quoted identifiers. However, if you have quoted identifiers switched on, this does not usually cause a problem with SQL Prompt.

String literals

SQL Prompt does not provide completion assistance inside string literals.

SQL Prompt auto-completes object name without including the owner or schema name

You can specify the behaviour when SQL Prompt inserts objects from the candidate list. If you want to include the owner or schema name automatically, go to the SQL Prompt Options dialog and choose the Inserted Candidates page. Select **Formatting** on the left of the dialog, then select:

- Qualify object names
- Qualify column names

SQL Prompt inserts a code snippet instead of a keyword or alias

Typing some keywords or aliases may result in a code snippet being added in place of the typed characters. This occurs when the snippet shortcut name is the same as a keyword. For example, typing IS automatically adds "INFORMATION_SCHEMA", (the default code for the snippet IS).

To fix this, edit the snippet in the Options dialog so that it no longer matches the keyword. Select **Options** from the SQL Prompt menu, then select the **Snippets** tab. In the list of snippets, scroll down to find the one that you want to change then double-click it. In the **Snippet** box enter the new name for the snippet.

Contacting SQL Prompt Technical Support

If you contact the Red Gate support team, they may ask you to send a log of SQL Prompt's activities; to do this, on the **SQL Prompt** menu, click **Show Log**. The log file is opened in your default text editor and you can save or copy the text as required.

Uninstalling SQL Prompt

To uninstall SQL Prompt:

1. Right-click the SQL Prompt 3 icon in the notification area and click **Exit**.
2. Use the Windows® **Add or Remove Programs** tool to remove **SQL Prompt**.
3. In SQL Server Management Studio, to remove the **SQL Prompt** menu from the toolbar, do the following:
   - On the **Tools** menu, click **Customize**.
   - Drag the **SQL Prompt** menu from the menu bar, or right-click the **SQL Prompt** menu and click **Delete**.
   - Close the **Customize** dialog box.

Acknowledgements

Copyright information

© Red Gate Software Ltd 1999 - 2008.

Trademarks and registered trademarks

Red Gate is a registered trademark of Red Gate Software Ltd registered in the U.S. Patent and Trademark Office. SQL Prompt is a trademark of Red Gate Software Ltd.

Microsoft, Windows, Windows 98, Windows NT, Windows 2000, Windows 2003, Windows XP, SQL Server, Visual Studio and other Microsoft products referenced herein are either registered trademarks or trademarks of Microsoft Corporation.

Snippet contributors

Red Gate would like to thank the following people for their snippet contributions: Dinesh Asanka, Jerry J. Deroo, Bart Holthuijsen, muellerbill, Alan Wood, and Mark Yuabov.