

1 Using the DB2 Web Query Adapter for Microsoft SQL Server

The DB2 Web Query Adapter for Microsoft® SQL Server® allows applications to access Microsoft SQL Server data sources. The adapter converts data or application requests into native Microsoft SQL Server statements and returns optimized answer sets to the requesting program.

Topics:

- ❑ Preparing the Microsoft SQL Server Environment
- ❑ Configuring the DB2 Web Query Adapter for Microsoft SQL Server
- ❑ Managing Microsoft SQL Server Metadata
- ❑ Reporting Against a Microsoft SQL Server Stored Procedure
- ❑ Customizing the Microsoft SQL Server Environment
- ❑ Optimization Settings
- ❑ Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru

Preparing the Microsoft SQL Server Environment

In order to take full advantage of the features available through the DB2 Web Query Adapter for Microsoft SQL Server 2005, we strongly recommend using MS SQL Server 2005 with Service Pack 1 (2005.90.2047) or higher. To determine which version of MS SQL Server you are using, please refer to the following article:
<http://support.microsoft.com/kb/321185>.

Configuring the DB2 Web Query Adapter for Microsoft SQL Server

In this section:

Declaring Connection Attributes

Controlling the Connection Scope

Before you configure the adapter, be sure to install the Microsoft SQL Server 2000 or 2005 JDBC Driver to the Java Extensions directory: `/QIBM/UserData/Java400/ext`. This eliminates the need to define classpath on the Web Query Reporting Server.

Configuring the adapter consists of specifying connection and authentication information for each of the connections you want to establish.

Declaring Connection Attributes

How to:

Declare Connection Attributes

Reference:

Connection Attributes for Microsoft SQL Server

In order to connect to a Microsoft SQL Server database server, the adapter requires connection and authentication information. You supply this information using the Web Query Metadata option. The connection and authentication information is added to the profile you select: the global server profile (`edasprof.prf`), a user profile (`user.prf`), or a group profile (if supported on your platform).

You can declare connections to more than one Microsoft SQL Server databases. The actual connection to the Microsoft SQL Server takes place when the first query that references the connection is issued. If you issue multiple CONNECTION_ATTRIBUTES commands:

- ❑ The connection named in the *first* CONNECTION_ATTRIBUTES command serves as the default connection.
- ❑ If more than one CONNECTION_ATTRIBUTES command contains the same connection name, the adapter uses the attributes specified in the *last* CONNECTION_ATTRIBUTES command.

Procedure: How to Declare Connection Attributes

To configure the adapter from the Web Query:

1. Log on to Web Query as a developer or administrator.
2. Expand the *Domains* folder, then expand a domain.
3. Expand *Reports* folder, click a *request* subfolder, and choose *Metadata* from the menu.
4. Click *New Adapter*. The Select Adapter to Add pane opens.
5. Expand the *SQL* group folder, then expand the *MS SQL Server* folder, and click *MS SQL Server*. The Add MS SQL Server to Configuration pane opens.
6. Enter values for the parameters required by the adapter.
For a description of these parameters, see *Connection Attributes for Microsoft SQL Server* on page 1-4.
7. Click *Configure*. The configured adapter is added to the Adapters list in the navigation pane.

Reference: Connection Attributes for Microsoft SQL Server

This chart describes the connection attributes for which you can supply values. To complete the attribute declaration, click the *Configure* button.

Attribute	Description
Connection name	Logical name used to identify this particular set of connection attributes.
URL <i>(UNIX, IBM i, and z/OS only)</i>	Enter the location URL for the Microsoft SQL Server data source.
Security	<p>There are three methods by which a user can be authenticated when connecting to a Microsoft SQL Server:</p> <p>Explicit. The user ID and password are explicitly specified for each connection and passed to Microsoft SQL Server, at connection time, for authentication as a standard log on.</p> <p>This option requires that SQL Server security be set to SQL Server and Windows (for Windows), or else to SQL Server and UNIX.</p> <p>Password Passthru. <i>(Windows only)</i> The user ID and password received from the client application are passed to Microsoft SQL Server, at connection time, for authentication as a standard log on.</p> <p>This option requires that SQL Server security be set to: SQL Server and Windows.</p> <p>Trusted. The adapter connects to Microsoft SQL Server as an operating system log on using the credentials of the operating system user impersonated by the server data access agent.</p> <p>This option works with either of the SQL Server security settings.</p>
User	Authorization ID by which the user is known to the instance of Microsoft SQL Server.
Password	Password associated with the authorization ID.

Attribute	Description
Driver name (UNIX, IBM i, and z/OS only)	Name for the Microsoft JDBC driver. Note that the sample should not be used as it may not be the correct driver name.
Select profile	<p>Select a profile from the drop-down list to indicate the level of profile in which to store the CONNECTION_ATTRIBUTES command. The global profile, edasprof.prf, is the default.</p> <p>If you wish to create a new profile—either a user profile (<i>user.prf</i>) or a group profile if available on your platform (using the appropriate naming convention)—choose <i>New Profile</i> from the drop-down list and enter a name in the Profile Name input box. (The extension is added automatically.)</p>

Note: Ignore the message stating CLASSPATH must be defined before the server starts. The CLASSPATH variable is not required and does not need to be set.

Controlling the Connection Scope

The AUTODISCONNECT command controls the persistence of connections when using the adapter for each of the connections you want to establish.

Syntax: How to Control the Connection Scope

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the Adapter for Microsoft SQL Server.

FIN

Disconnects automatically only after the session has been terminated. FIN is the default value.

COMMAND

Disconnects automatically after each request. Depending on how often the event occurs, the AUTODISCONNECT command may result in considerable overhead. Almost all of this overhead is not related to the server; it is related to the operating system and the data source.

Managing Microsoft SQL Server Metadata

In this section:

Creating Synonyms

Data Type Support

Enabling National Language Support

Support of Read-Only Fields

When the server accesses a data source, it needs to know how to interpret the data stored there. For each data source the server will access, you create a synonym that describes the structure of the data source and the server mapping of the Microsoft SQL Server data types.

Creating Synonyms

How to:

Create a Synonym From the Web Query Environment

Example:

Sample Generated Synonym

Reference:

Synonym Creation Parameters for Microsoft SQL Server

Mapping Microsoft SQL Table Comments Into a Synonym

Access File Keywords

Synonyms define unique names (or aliases) for each Microsoft SQL Server table or view that is accessible from the server. Synonyms are useful because they hide the underlying data source's location and identity from client applications. They also provide support for extended metadata features of the server, such as virtual fields and additional security mechanisms.

Using synonyms allows an object to be moved or renamed while allowing client applications to continue functioning without modification. The only modification required is a redefinition of the synonym on the server. The result of creating a synonym is a Master File and an Access File, which represent the server's metadata.

Note that creating a synonym for a stored procedure is described with reporting against a stored procedure, in *Generating a Synonym for a Stored Procedure* on page 1-21.

Procedure: How to Create a Synonym From the Web Query Environment

1. Expand the *Domains* folder, then expand a domain.
2. Expand the *Reports* folder, click a request subfolder, and choose *Metadata* from the menu.
3. In the left-hand Adapter navigation pane, click the database connection (MS SQL Server 200x) and choose *Create Synonym* from the menu. The first of a series of synonym creation pages opens.
4. Enter values for the parameters required by the adapter.
For information about these parameters, see *Synonym Creation Parameters for Microsoft SQL Server* on page 1-8.
5. After entering parameter values, click *Create Synonym*.
Synonyms are created and added under the specified application directory.
The Status pane indicates that the synonyms were created successfully.
6. You can click *Go to Metadata page* where you can manage synonyms from the navigation pane.

Reference: Synonym Creation Parameters for Microsoft SQL Server

This chart describes the synonym creation parameters for which you can supply values.

Parameter/Task	Description
Restrict Object Type to	<p>Restrict candidates for synonym creation based on the selected object type(s): Tables, Views, External SQL Scripts, and any other supported objects.</p> <p>Choosing <i>External SQL Scripts</i> from the drop-down list enables you to represent an SQL Query as a synonym for read-only reporting. A Synonym candidate can be any file that contains one (and only one) valid SQL Query and does not contain end-of-statement delimiters (";" or "/") and comments. For related information, see <i>Location of External SQL Scripts</i> in this chart.</p> <p>Depending on adapter, you can further restrict your search by choosing check boxes for listed objects.</p> <p>Important: If you select Stored Procedures as your object type, the input parameters will be a little different from those described here. For details, refer to the topic <i>Creating a Report Against a Stored Procedure</i> on page 1-26.</p>
Database selection	<p>To specify a database from which you can select a table or other object, do one of the following:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Check <i>Use current database</i> to use the database that has been set as the default database. <input type="checkbox"/> Select a database from the <i>Select database</i> drop-down list, which lists all databases in the current DBMS instance. <p>Before selecting a database, if <i>Use current database</i> is checked, uncheck it.</p>

Parameter/Task	Description
Filter by Owner/Schema and Object name	<p>Selecting this option adds the Owner/Schema and Object Name parameters to the screen.</p> <p>Owner/Schema. Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.</p> <p>Object name. Type a string for filtering the procedure names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all procedures whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.</p>

Parameter/Task	Description
<p>Location of External SQL Scripts</p> <p>Extension</p>	<p>If you specify <i>External SQL Scripts</i> in the <i>Restrict Object type</i> to field, these additional fields are displayed.</p> <p>The following standard naming conventions apply for UNIX, IBM i IFS, and z/OS HFS:</p> <ul style="list-style-type: none"> <input type="checkbox"/> In the <i>Location of External SQL Scripts</i> field, specify the physical directory location of the file that contains the SQL Query. <input type="checkbox"/> In the <i>Extension</i> field, enter the extension of the script files to filter the list of candidates. <p>On IBM i, you can use alternative IFS naming conventions to access library members. The following entry illustrates this method:</p> <ul style="list-style-type: none"> <input type="checkbox"/> In the <i>Location of External SQL Scripts</i> field, enter: <ul style="list-style-type: none"> <code>/QSYS.LIB/MYLIBRARY.LIB/MYSRC.FILE</code> <input type="checkbox"/> The <i>Extension</i> is understood to be MBR. You can enter this value explicitly or leave the input box blank. <p>During synonym generation, the adapter issues native API calls to obtain a list of elements in the select list and builds the Master File with a field for each element. The generated Access File references the location of the SQL script in the DATASET attribute, which contains the full path, including the name and extension of the file containing the SQL Query. For example,</p> <p><code>DATASET=/ul/home2/apps/report3.sql</code></p> <p>When a Web Query report is created, the SQL Query is used to access data.</p>
<p>Select Application</p>	<p>Select an application directory. baseapp is the default value.</p>

Parameter/Task	Description
Cardinality	<p>Select the <i>Cardinality</i> check box to reflect the current cardinality (number of rows or tuples) in the table during metadata creation. Cardinality is used for equi-joins. The order of retrieval is based on the size (cardinality) of the table. Smaller tables are read first.</p> <p>If the cardinality of the tables to be used in the application are dynamic, it may not be beneficial to choose this setting.</p>
With foreign key	<p>Select the <i>With foreign key</i> check box to include within this synonym every table related to the current table by a foreign key. The resulting multi-table synonym describes all of this table's foreign key relationships.</p>
Dynamic columns	<p>To specify that the Master File created for the synonym should not contain column information, select the <i>Dynamic columns</i> check box.</p> <p>If this option is selected, column data is retrieved dynamically from the data source at the time of the request.</p>
Prefix/Suffix	<p>If you have tables with identical table names, assign a prefix or a suffix to distinguish them. For example, if you have identically named human resources and payroll tables, assign the prefix HR to distinguish the synonyms for the human resources tables. Note that the resulting synonym name cannot exceed 64 characters.</p> <p>If all tables and views have unique names, leave prefix and suffix fields blank.</p>
Customize data type mappings	<p>To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed. For information about them, see <i>Data Type Support</i>.</p>
Overwrite Existing Synonyms	<p>To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the <i>Overwrite existing synonyms</i> check box.</p>
Default Synonym Name	<p>This column displays the name that will be assigned to each synonym. To assign a different name, replace the displayed value.</p>

Parameter/Task	Description
Select tables	<p>Select tables for which you wish to create synonyms:</p> <ul style="list-style-type: none"> <input type="checkbox"/> To select all tables in the list, select the check box to the left of the <i>Default Synonym Name</i> column heading. <input type="checkbox"/> To select specific tables, select the corresponding check boxes.

Example: Sample Generated Synonym

A DB2 Web Query Adapter for Microsoft SQL Server synonym comprises a Master File and an Access File. This is a synonym for the table nf29004.

Generated Master File nf29004.mas

```
FILE=DIVISION, SUFFIX=SQLMSS , $
SEGNAME=SEG1_4, SEGTYPE=S0 , $
FIELD=DIVISION4, DIVISION4, I9, I4, MISSING=OFF , $
FIELD=DIVISION_NA4, DIVISION_NA4, A25, A25, MISSING=ON , $
FIELD=DIVISION_HE4, DIVISION_HE4, I9, I4, MISSING=ON , $
```

Generated Access File nf29004.acx

```
SEGNAME=SEG1_4, TABLENAME=edaqa.nf29004,
CONNECTION=connmss, KEYS=1, WRITE=YES, $
```

Reference: Mapping Microsoft SQL Table Comments Into a Synonym

The adapter enables you to capture non-Unicode column descriptions from MS SQL tables.

When you generate a synonym for a Microsoft SQL table, the adapter maps MS SQL Server column comments (if present) to the DESCRIPTION attribute in the synonym’s Master File.

Reference: Access File Keywords

This chart describes the keywords in the Access File.

Keyword	Description
SEGNAME	Value must be identical to the SEGNAME value in the Master File.
TABLENAME	<p>Identifies the Microsoft SQL Server table. The table name can be fully qualified as follows:</p> <p><code>TABLENAME= [[database.]owner.] table</code></p>

Keyword	Description
CONNECTION	<p>Indicates a previously declared connection. The syntax is:</p> <p><code>CONNECTION=connection</code></p> <p><code>CONNECTION=' '</code> indicates access to the local database server.</p> <p>Absence of the CONNECTION attribute indicates access to the default database server.</p>
KEYS	<p>Indicates how many columns constitute the primary key for the table. Range is 0 to 64. Corresponds to the first <i>n</i> fields in the Master File segment.</p>
WRITE	<p>Specifies whether write operations are allowed against the table.</p>
KEYFLD IXFLD	<p>Supply the names of the primary key and foreign key fields that implement the relationships established by the multi-table Master File. Together, KEYFLD and IXFLD identify the field shared by a related table pair.</p> <ul style="list-style-type: none"> <input type="checkbox"/> KEYFLD is the FIELDNAME of the common column from the parent table. <input type="checkbox"/> IXFLD is the FIELDNAME of the common column from the related table. <p>KEYFLD and IXFLD must have the same data type. It is recommended, but not required, that their lengths also be the same.</p> <p>Note: An RDBMS index on both the KEYFLD and IXFLD columns provides the RDBMS with a greater opportunity to produce efficient joins. The columns must have the same data type. If their length is the same, the RDBMS handles the join more efficiently.</p>

Reference: Managing Synonyms

Once you have created a synonym, you can click synonym name in the Adapter navigation pane to access the following options:

Option	Description
Open in Synonym Editor	Opens the synonym's Master File component in the right pane, for viewing and editing using a graphical interface.

Option	Description
Edit as Text	<p>Enables you to view and manually edit the synonym's Master File.</p> <p>Note: To update the synonym, it is strongly recommended that you use the graphical interface provided by the Open option, rather than manually editing the Master File.</p>
Edit Access File as Text	<p>Enables you to view and manually edit the synonym's Access File.</p>
Sample Data	<p>Retrieves up to 20 rows from the associated data source.</p>
Data Profiling	<p>Data Profiling provides the data characteristics for a synonym's columns.</p> <p>Alphanumeric columns provide the count of distinct values, total count, maximum, minimum, and average length, and number of nulls.</p> <p>Numeric columns provide the count of distinct values, total count, maximum, minimum, and average value, and number of nulls.</p>
Refresh Synonym	<p>Regenerates the synonym. Use this option if the underlying object has been altered.</p>
Data Management	<p>Followed by these options, <i>if applicable</i>:</p> <p>Recreate DBMS Table. Recreate the data source tables. You are asked to confirm this selection before the tables are regenerated. (Note that the table will be dropped and recreated. During the process, data may be lost.)</p> <p>Delete All Data. Deletes all existing data. You are asked to confirm this selection before the data is deleted.</p> <p>Insert Sample Data. Inserts a specified number of sample records, populating all fields with counter values.</p>
Impact Analysis	<p>Generates reports on procedures, synonyms, and columns that provide information on the flows/stored procedures available on a particular server, and the synonyms and columns they use. These reports enable you to evaluate changes before they are made by showing which components will be affected.</p>
Drop Synonym	<p>Deletes the synonym. You are asked to confirm this selection before the synonym is deleted.</p>

Option	Description
Move Synonym	Moves the synonym to another application directory. Click the target directory from the resulting list.
Copy Synonym	Copies the synonym to another application directory. Click the target directory from the resulting list.

Data Type Support

Reference:

Data Type Support for Unicode

The following table lists how the server maps Microsoft SQL Server data types. Some data type mappings differ when the server is configured for Unicode. These mappings are described in *Data Type Support for Unicode* on page 1-19.

Microsoft SQL Server Data Type	Data Type		Remarks
	USAGE	ACTUAL	
CHAR (<i>n</i>) single-byte code page	An	An	<i>n</i> is an integer between 1 and 8000.
CHAR (<i>n</i>) double-byte code page	An	An	<i>n</i> is an integer between 1 and 8000. Unicode notes for this mapping are in <i>Data Type Support for Unicode</i> on page 1-19.
NCHAR (<i>n</i>)	An	An	<i>n</i> specifies the number of characters, and is an integer between 1 and 4000. The Unicode version of this mapping is described in <i>Data Type Support for Unicode</i> on page 1-19.
VARCHAR (<i>n</i>)	AnV	AnV	<i>n</i> is an integer between 1 and 8000. Unicode notes for this mapping are in <i>Data Type Support for Unicode</i> on page 1-19.

Microsoft SQL Server Data Type	Data Type		Remarks
	USAGE	ACTUAL	
NVARCHAR (n)	AnV	AnV	n specifies the number of characters, and is an integer between 1 and 4000. The Unicode version of this mapping is described in <i>Data Type Support for Unicode</i> on page 1-19.
TEXT	A32767	A32767	
NTEXT	TX50	TX	
BINARY(n)	Am	Am	n is an integer between 1 and 8000. $m = 2 * n$
VARBINARY(n)	AmV	AmV	n is an integer between 1 and 8000. $m = 2 * n$
SQL_VARIANT	A8000	A8000	
UNIQUEIDENTIFIER (GUID)	A38	A38	
TIMESTAMP	A16	A16	Supported as Read-only
DATETIME	HYYMDs	HYYMDs	range: 1/1/1753 to 12/31/9999
SMALLDATETIME	HYYMDI	HYYMDI	range: 1/1/1900 thru 6/6/2079
INT	I11	I4	range: -2^{31} to $2^{31} - 1$
BIGINT	P20	P10	range: 2^{63} to $2^{63} - 1$
SMALLINT	I6	I4	range: -2^{15} to $2^{15} - 1$
TINYINT	I6	I4	range: 0 to 255
BIT	I11	I4	-1 for "True" and 0 for "False"

Microsoft SQL Server Data Type	Data Type		Remarks
	USAGE	ACTUAL	
DECIMAL (<i>p,s</i>)	<i>Pn.m</i>	<i>Pk</i>	<p><i>p</i> is an integer between 1 and 38 <i>s</i> is an integer between 0 and <i>p</i></p> <p>If <i>s</i> is 0 and <i>p</i> is between 1 and 31, $n = p + 1$</p> <p>If <i>s</i> is 0 and <i>p</i> is between 32 and 38, $n = 32$</p> <p>If <i>s</i> is greater than 0 and <i>p</i> is between 1 and 31, $n = p + 2$ and $m = s$</p> <p>If <i>s</i> is greater than 0 and <i>p</i> is between 32 and 38, $n = 33$ and $m = 31$</p> <p>If <i>p</i> is between 1 and 31, $k = (p / 2) + 1$</p> <p>If <i>p</i> is between 32 and 38, $k = 16$</p> <p>Note: If the column is nullable, <i>p</i> is greater than or equal to 8.</p>
NUMERIC (<i>p,s</i>)	<i>Pn.m</i>	<i>Pk</i>	<p><i>p</i> is an integer between 1 and 38 <i>s</i> is an integer between 0 and <i>p</i></p> <p>If <i>s</i> is 0 and <i>p</i> is between 1 and 31, $n = p + 1$</p> <p>If <i>s</i> is 0 and <i>p</i> is between 32 and 38, $n = 32$</p> <p>If <i>s</i> is greater than 0 and <i>p</i> is between 1 and 31, $n = p + 2$ and $m = s$</p> <p>If <i>s</i> is greater than 0 and <i>p</i> is between 32 and 38, $n = 33$ and $m = 31$</p> <p>If <i>p</i> is between 1 and 31, $k = (p / 2) + 1$</p> <p>If <i>p</i> is between 32 and 38, $k = 16$</p> <p>Note: If the column is nullable, <i>p</i> is greater than or equal to 8.</p>
MONEY	P21.4	P10	range: -2^{63} to $2^{63} - 1$
SMALLMONEY	P12.4	P8	range: -214,748.3648 to 214,748.3647

Microsoft SQL Server Data Type	Data Type		Remarks
	USAGE	ACTUAL	
FLOAT	D20.2	D8	range: -1.79E+308 to 1.79E+308
REAL	D20.2	D8	range: -3.40E+38 to 3.40E+38
IMAGE	BLOB	BLOB	length: $2^{31} - 1$ Supported through the Reporting Server API

Reference: Data Type Support for Unicode

The following table describes how the adapter maps Microsoft SQL Server data types that support Unicode when the server is configured for Unicode.

Microsoft SQL Server Data Type	Data Type		Remarks
	USAGE	ACTUAL	
CHAR (<i>n</i>) double-byte code page	AnV	AnV	<i>n</i> is an integer between 1 and 8000. This data type does not support Unicode. However, character columns such as CHAR that had been created with different code pages can be read at the same time only if the server has been configured for Unicode.
NCHAR (<i>n</i>)	An	An	<i>n</i> specifies the number of characters, and is an integer between 1 and 4000. This data type supports Unicode.
VARCHAR (<i>n</i>)	AnV	AnV	<i>n</i> is an integer between 1 and 8000. This data type does not support Unicode. However, character columns such as VARCHAR that had been created with different code pages can be read at the same time only if the server has been configured for Unicode.
NVARCHAR (<i>n</i>)	AnV	AnV	<i>n</i> specifies the number of characters, and is an integer between 1 and 4000. This data type supports Unicode.

Enabling National Language Support

How to:

Enable National Language Support

The parameter NCHAR indicates whether the character set is single-byte, double-byte, or triple-byte. The NCHAR setting affects the mapping of NCHAR and NVARCHAR data types.

The following chart lists data type mappings based on the value of NCHAR.

Microsoft SQL Server Data Type	Remarks	NCHAR SBCS		NCHAR DBCS		NCHAR TBCS	
		USAGE	ACTUAL	USAGE	ACTUAL	USAGE	ACTUAL
NCHAR (n)	n is an integer between 1 and 4000 d = 2 * n t = 3 * n	An	An	Ad	Ad	At	At
NVARCHAR (n)	n is an integer between 1 and 4000 d = 2 * n t = 3 * n	An	An	Ad	Ad	At	At

Syntax: How to Enable National Language Support

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SQLENGINE command.

SBCS

Indicates a single-byte character set. SBCS is the default value.

DBCS

Indicates a double-byte character set.

TBCS

Indicates a triple-byte character set.

Support of Read-Only Fields

CREATE SYNONYM creates a field description with FIELDTYPE=R for Microsoft SQL Server columns created as TIMESTAMP or columns with the IDENTITY attribute. These fields are read-only.

Reporting Against a Microsoft SQL Server Stored Procedure

In this section:

Generating a Synonym for a Stored Procedure

Creating a Report Against a Stored Procedure

You can use a reporting tool—such as a SELECT statement or TABLE command—to execute Microsoft SQL Server stored procedures and report against a procedure's output parameters and answer set. Among the benefits of this method of executing a stored procedure are:

- ❑ The retrieval of output parameters—OUT parameters, and INOUT parameters in OUT mode—as well as the answer set. (Other methods of invocation retrieve the answer set only.)
- ❑ The ease with which you can process, format, and display output parameters and the answer set, using TABLE and other reporting tools.

To report against a stored procedure:

- 1. Generate a synonym** for the stored procedure answer set, as described in *Generating a Synonym for a Stored Procedure*.
- 2. Create a report procedure**, as described in *Creating a Report Against a Stored Procedure*.
- 3. Run the report.** This executes the stored procedure and reports against any output parameters (OUT and INOUT in OUT mode), and any answer set fields, specified in the report.

Generating a Synonym for a Stored Procedure

Example:

Synonym for Microsoft SQL Server Stored Procedure CustOrders

Reference:

Synonym Creation Parameters for Stored Procedures

A synonym describes a stored procedure's parameters and answer set.

An answer set's structure may vary depending on the input parameter values that are provided when the procedure is executed. Therefore, you need to generate a separate synonym for each set of input parameter values that will be provided when the procedure is executed at run time. For example, if users may execute the stored procedure using three different sets of input parameter values, you need to generate three synonyms, one for each set of values. (Unless noted otherwise, "input parameters" refers to IN parameters and to INOUT parameters in IN mode.)

There is an exception: if you know the procedure's internal logic, and are certain which range of input parameter values will generate each answer set structure returned by the procedure, you can create one synonym for each answer set structure, and for each synonym simply provide a representative set of the input parameter values necessary to return that answer set structure.

A synonym includes the following segments:

- ❑ INPUT, which describes any IN parameters and INOUT parameters in IN mode.
If there are no IN parameters or INOUT parameters in IN mode, the segment describes a single dummy field.
- ❑ OUTPUT, which describes any OUT parameters and INOUT parameters in OUT mode.
If there are no OUT parameters or INOUT parameters in OUT mode, the segment is omitted.
- ❑ ANSWERSET n , one for each answer set.
If there is no answer set, the segment is omitted.

Example: Synonym for Microsoft SQL Server Stored Procedure CustOrders

The following synonym describes a Microsoft SQL Server stored procedure with one input parameter, one output parameter, and one answer set containing four variables.

The synonym's Master File is:

```
FILENAME=CUSTORDERS, SUFFIX=SQLMSS , $
  SEGMENT=INPUT, SEGTYPE=S0, $
    FIELDNAME=CUSTOMERID, ALIAS=P0001, USAGE=A5, ACTUAL=A5,
      MISSING=ON, ACCESS_PROPERTY=(NEED_VALUE), $
  SEGMENT=OUTPUT, SEGTYPE=S0, PARENT=INPUT, $
    FIELDNAME=@RETURN_VALUE, ALIAS=P0000, USAGE=I11, ACTUAL=I4, $
  SEGMENT=ANSWERSET1, SEGTYPE=S0, PARENT=INPUT, $
    FIELDNAME=ORDERID, ALIAS=OrderID, USAGE=I11, ACTUAL=I4, $
    FIELDNAME=ORDERDATE, ALIAS=OrderDate, USAGE=HYMDS, ACTUAL=HYMDS,
      MISSING=ON, $
    FIELDNAME=REQUIREDDATE, ALIAS=RequiredDate, USAGE=HYMDS,
      ACTUAL=HYMDS, MISSING=ON, $
    FIELDNAME=SHIPPEDDATE, ALIAS=ShippedDate, USAGE=HYMDS,
      ACTUAL=HYMDS, MISSING=ON, $
```

The synonym's Access File is:

```
SEGNAME=INPUT, CONNECTION=ITarget, STPNAME=Northwind.dbo.CustOrders, $
SEGNAME=OUTPUT, STPRESORDER=0, $
SEGNAME=ANSWERSET1, STPRESORDER=1, $
```

Reference: Synonym Creation Parameters for Stored Procedures

Parameter/Task	Description
Restrict Object Type to	Select <i>Stored Procedures</i> .
Filter by Owner/Schema and Object name	<p>Selecting this option adds the Owner/Schema and Object Name parameters to the screen.</p> <p>Owner/Schema. Type a string for filtering the selection, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select tables or views whose owner/schema begin with the letters ABC; %ABC to select tables or views whose owner/schema end with the letters ABC; %ABC% to select tables or views whose owner/schema contain the letters ABC at the beginning, middle, or end.</p> <p>Object name. Type a string for filtering the procedure names, inserting the wildcard character (%) as needed at the beginning and/or end of the string. For example, enter: ABC% to select all procedures whose names begin with the letters ABC; %ABC to select all whose names end with the letters ABC; %ABC% to select all whose names contain the letters ABC at the beginning, middle, or end.</p>
Select	Select a procedure. You may only select one procedure at a time since each procedure will require unique input in the Values box on the next synonym creation pane.
Name	The name of the synonym, which defaults to the stored procedure name.
Select Application	Select an application directory. baseapp is the default value.
Prefix/Suffix	<p>If you have stored procedures with identical names, assign a prefix or a suffix to distinguish their corresponding synonyms. Note that the resulting synonym name cannot exceed 64 characters.</p> <p>If all procedures have unique names, leave the prefix and suffix fields blank.</p>
Overwrite Existing Synonyms	To specify that this synonym should overwrite any earlier synonym with the same fully qualified name, select the <i>Overwrite existing synonyms</i> check box.

Parameter/Task	Description
Customize data type mappings	To change the data type mappings from their default settings, select this check box. The customizable mappings are displayed.
Values	<p>Select the check box for every parameter displayed for the specified procedure.</p> <p>Note the following before you enter parameter values: if the procedure you selected has input parameters (IN parameters and/or INOUT parameters in IN mode), you will be prompted to enter values for them. However, the need for an explicit Value entry depends on the logic of the procedure and the data structures it produces. Therefore, while you must check the parameter box, you may not need to enter a value. Follow these guidelines:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Explicit input values (and separate synonyms) are required when input parameter values cause answer sets with different data structures, which vary depending on the input parameters provided. <input type="checkbox"/> Explicit input values are not required when you know the procedure's internal logic and are certain that it always produces the same data structure. In this situation, only one synonym needs to be created and you can leave the Value input blank for synonym-creation purposes. <p>If a Value is required, enter it without quotes. Any date, date-time, and timestamp parameters must have values entered in an ISO format. Specify the same input parameters that will be provided when the procedure is executed at run time if it is a procedure that requires explicit values.</p>

Creating a Report Against a Stored Procedure

How to:

Report Against a Stored Procedure Using the TABLE Command

Report Against a Stored Procedure Using SELECT

You can report against a stored procedure's answer set using the same facilities you use to report against a database table:

- ❑ **SQL SELECT statement.** For syntax, see *How to Report Against a Stored Procedure Using SELECT*.
- ❑ **TABLE and GRAPH commands.** For syntax, see *How to Report Against a Stored Procedure Using the TABLE Command*.

When joining from or to a stored procedure answer set, you can:

- ❑ **Join from** only OUTPUT and ANSWERSET segments in a host file.
- ❑ **Join to** only INPUT segments in a cross-referenced file.

Syntax: How to Report Against a Stored Procedure Using the TABLE Command

To execute a stored procedure using the TABLE command, use the following syntax

```
TABLE FILE synonym
PRINT [parameter [parameter] ... | *]
[IF in-parameter EQ value]
.
.
.
END
```

where:

synonym

Is the synonym of the stored procedure you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, specify an asterisk (*). This displays a dummy segment, created when the synonym is generated, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

IF

Is an IF or WHERE keyword. Use this to pass a value to an IN parameter or an INOUT parameter in IN mode.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

Note: The length of in-parameters cannot exceed 1000 characters if the adapter is configured for Unicode support.

value

Is the value you are passing to a parameter.

Syntax: How to Report Against a Stored Procedure Using SELECT

SQL

```
SELECT [parameter [,parameter] ... | *] FROM synonym
[WHERE in-parameter = value]
.
.
.
END
```

where:

synonym

Is the synonym of the stored procedure that you want to execute.

parameter

Is the name of a parameter whose values you want to display in the report. You can specify input parameters, output parameters, or input and output parameters.

If the stored procedure does not require parameters, enter an asterisk (*) in the syntax. This displays a dummy segment, created during synonym generation, to satisfy the structure of the SELECT statement.

*

Indicates that you want to display all indicated parameters, or that there are no required parameters.

WHERE

Is used to pass a value to an IN parameter or an INOUT parameter in IN mode. You must specify the value of each parameter on a separate line.

in-parameter

Is the name of an IN parameter, or INOUT parameter in IN mode, to which you want to pass a value.

value

Is the value you are passing to a parameter.

Customizing the Microsoft SQL Server Environment

In this section:

Specifying the Cursor Type

Activating NONBLOCK Mode

Obtaining the Number of Rows Updated or Deleted

Controlling Transactions

Specifying the Transaction Isolation Level

The DB2 Web Query Adapter for Microsoft SQL Server provides several parameters for customizing the environment and optimizing performance. This topic provides an overview of customization options.

Specifying the Cursor Type

You can use the SET CURSORS command to specify the type of cursors for retrieval.

Syntax: **How to Specify the Cursor Type**

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

CLIENT

Uses Microsoft SQL Server client-side cursors for retrieving data. Client-side cursors normally demonstrate the best performance for data retrieval and benefit the Microsoft SQL Server process. However, except in TRANSACTIONS AUTOCOMMITTED mode, using client-side cursors prevents a server agent from simultaneously reading more than one answer set from the same instance of Microsoft SQL Server.

SERVER

Uses Microsoft SQL Server server-side cursors for retrieving data. Server-side cursors demonstrate lower performance than client cursors. However, setting a high FETCHSIZE factor (100 is the adapter default) improves their performance dramatically making them almost as fast as client-side cursors. Client-side cursors are recommended wherever possible in order to take the load off the Microsoft SQL Server process.

blank

Uses client-side cursors in TRANSACTIONS AUTOCOMMITTED mode and server-side cursors otherwise. This value is the default.

Activating NONBLOCK Mode

The Adapter for Microsoft SQL Server has the ability to issue calls in NONBLOCK mode. The default behavior is BLOCK mode.

This feature allows the adapter to react to a client request to cancel a query while the adapter is waiting on engine processing. This wait state usually occurs during SQL parsing, before the first row of an answer set is ready for delivery to the adapter or while waiting for access to an object that has been locked by another application.

Syntax: How to Activate NONBLOCK Mode

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SQLENGINE command.

n

Is a positive numeric number. 0 is the default value, which means that the adapter will operate in BLOCK mode. A value of 1 or greater activates the NONBLOCK calling and specifies the time, in seconds, that the adapter will wait between each time it checks to see if the:

- Query has been executed.
- Client application has requested the cancellation of a query.

- ❑ Kill Session button is pressed.

Note: A value of 1 or 2 should be sufficient for normal operations.

Obtaining the Number of Rows Updated or Deleted

PASSRECS returns the number of rows affected by a successfully executed SQL Passthru INSERT, UPDATE, or DELETE command.

Syntax: How to Obtain the Number of Rows Updated or Deleted

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SQLENGINE command.

ON

Provides the number of rows affected in the application program SCB count member after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command. ON is the default value.

OFF

Provides no information after the successful execution of an SQL Passthru INSERT, UPDATE, or DELETE command.

In addition, the adapter updates the &RECORDS system variable with the number of rows affected. You can access this variable using Dialogue Manager.

Controlling Transactions

The TRANSACTIONS command controls how the adapter handles transactions.

Syntax: How to Control Transactions

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the Adapter for Microsoft SQL Server. You can omit this value if you previously issued the SQLENGINE command.

LOCAL

Indicates that the adapter implicitly starts a local transaction on each of the connections where any work is performed. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter commits or aborts the work on each connection consecutively. LOCAL is the default value.

DISTRIBUTED

Indicates that the adapter implicitly invokes Microsoft Distributed Transactions Coordinator (DTC) to create a single distributed transaction within which to perform all work on all the connections. At the time of COMMIT or ROLLBACK, or at the end of the server session, the adapter invokes DTC to execute the two-phase commit or rollback protocol. For this purpose, the DTC service must be started on the machine where the server is running and also on all the machines where involved instances of Microsoft SQL Server reside.

This mode is recommended for read-write applications that perform updates on multiple connections simultaneously.

AUTOCOMMITTED

Indicates that each individual operation with Microsoft SQL Server is immediately committed (if successful) or rolled back (in case of errors) by the SQL Server. This is recommended for read-only applications for performance considerations. It is not recommended for read-write applications because in this mode it is impossible to roll back a logical unit of work that consists of several operations.

Specifying the Transaction Isolation Level

You can specify the transaction isolation level from the Web Console or using the ISOLATION command.

Syntax: How to Specify Transaction Isolation Level

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

RU

Sets the transaction isolation level to Read Uncommitted.

RC

Sets the transaction isolation level to Read Committed.

RR

Sets the transaction isolation level to Repeatable Read.

SE

Sets the transaction isolation level to Serializable Read.

CH

Sets the transaction isolation level to Chaos.

CS

Sets the transaction isolation level to Cursor Stability, which is a synonym for Read Committed.

Optimization Settings

In this section:

Optimizing Requests

Optimizing Requests Containing IF-THEN-ELSE Virtual Fields

Optimizing Requests if a Virtual Field Contains Null Values

Adapter optimization allows the RDBMS to perform the work for which it is best suited, reducing the volume of RDBMS-to-server communication and improving response time. It also enables the RDBMS to exploit its own internal optimization techniques.

Optimizing Requests

How to:

Optimize Requests

Example:

SQL Requests Passed to the RDBMS With Optimization OFF

SQL Requests Passed to the RDBMS With Optimization ON

Reference:

SQL Generation in Optimization Examples

The adapter can optimize DML requests by creating SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities.

Syntax: How to Optimize Requests

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Is the target RDBMS. You can omit this value if you previously issued the `SQLENGINE` command.

SQLJOIN

Is a synonym for `OPTIMIZATION`.

setting

Is the optimization setting. Valid values are as follows:

OFF instructs the adapter to create SQL statements for simple data retrieval from each table. The server handles all aggregation, sorting, and joining in your address space or virtual machine to produce the report.

ON instructs the adapter to create SQL statements that take advantage of RDBMS join, sort, and aggregation capabilities. Note that the multiplicative effect may disable optimization in some cases. However, misjoined unique segments and multiplied lines in PRINT- and LIST-based report requests do not disable optimization.

Example: SQL Requests Passed to the RDBMS With Optimization OFF

This example demonstrates SQL statements generated without optimization; the report request joins tables EMPINFO and FUNDTRAN with trace components SQLAGGR and STMTRACE allocated.

When optimization is disabled, the data adapter generates two SELECT statements. The first SELECT retrieves any rows from the EMPINFO table that have the value MIS in the DEPARTMENT column. For each EMPINFO row, the second SELECT retrieves rows from the cross-referenced FUNDTRAN table, resolving the parameter marker (?, :000n, or :H, depending on the RDBMS) with the value of the host field (EMP_ID). Both SELECT statements retrieve answer sets, but the server performs the join, sort, and aggregation operations:

```
SQL SQLMSS SET OPTIMIZATION OFF
JOIN EMP_ID IN EMPINFO TO ALL WHO IN FUNDTRAN AS J1
TABLE FILE EMPINFO
  SUM AVE.CURRENT_SALARY ED_HRS BY WHO BY LAST_NAME
  IF DEPARTMENT EQ 'MIS'
END
```

In a trace operation, you will see the following output:

```
(FOC2510) FOCUS-MANAGED JOIN SELECTED FOR FOLLOWING REASON(S) :
(FOC2511) DISABLED BY USER
(FOC2590) AGGREGATION NOT DONE FOR THE FOLLOWING REASON:
(FOC2592) RDBMS-MANAGED JOIN HAS BEEN DISABLED
  SELECT T1.EID,T1.LN,T1.DPT,T1.CSAL,T1.OJT
    FROM "USER1"."EMPINFO" T1 WHERE (T1.DPT = 'MIS') FOR FETCH ONLY;
  SELECT T2.EID FROM "USER1"."FUNDTRAN" T2 WHERE (T2.EID = ?)
    FOR FETCH ONLY;
```

Example: SQL Requests Passed to the RDBMS With Optimization ON

With optimization enabled, the data adapter generates one SELECT statement that incorporates the join, sort, and aggregation operations. The RDBMS manages and processes the request; the server only formats the report.

```
SQL SQLMSS SET OPTIMIZATION ON
JOIN EMP_ID IN EMPINFO TO ALL WHO IN FUNDTRAN AS J1
TABLE FILE EMPINFO
SUM AVE.CURRENT_SALARY ED_HRS BY WHO BY LAST_NAME
IF DEPARTMENT EQ 'MIS'
END
```

In a trace operation, you will see the following output:

```
AGGREGATION DONE ...
SELECT T2.EID,T1.LN, AVG(T1.CSAL), SUM(T1.OJT)
FROM "USER1"."EMPINFO" T1,"USER1"."FUNDTRAN" T2
WHERE (T2.EID = T1.EID) AND (T1.DPT = 'MIS')
GROUP BY T2.EID,T1.LN
ORDER BY T2.EID,T1.LN FOR FETCH ONLY;
```

Both OPTIMIZATION settings produce the same report.

Reference: SQL Generation in Optimization Examples

There are minor differences in the specific SQL syntax generated for each RDBMS. However, the adapter messages are the same and the generated SQL statements are similar enough that most examples will illustrate the SQL syntax generated by any relational adapter.

Optimizing Requests Containing IF-THEN-ELSE Virtual Fields

How to:

Optimizing Requests Containing IF-THEN-ELSE Virtual Fields

Example:

Using IF-THEN_ELSE Optimization Without Aggregation

Using IF-THEN_ELSE Optimization With Aggregation

Using IF-THEN_ELSE Optimization With a Condition That is Always False

Reference:

SQL Limitations on Optimization of DEFINE Expressions

The adapter can optimize DML requests to the server that include virtual (DEFINE) fields created using IF-THEN-ELSE syntax. In certain cases, such DEFINE fields can be passed to the RDBMS as expressions, enhancing performance and minimizing the size of the answer set returned to the server.

Syntax: **How to Optimize Requests Containing IF-THEN-ELSE Virtual Fields**

You change this setting by expanding the Domains and Reports folders, then right-clicking a subfolder and choosing *Metadata*. Right-click MS SQL Server 200x and select *Change Settings*. The Change Settings pane opens.

The available parameters are:

SQLMSS

Indicates the target RDBMS. You can omit this value if you previously issued the `SQLENGINE` command.

ON

Enables IF-THEN-ELSE optimization.

OFF

Disables IF-THEN-ELSE optimization. OFF is the default setting.

There is no guarantee that the SQL that is generated will improve performance for all requests. If you find that this feature does not improve performance, set `OPTIFTHENELSE OFF` to disable the feature.

IF-THEN-ELSE optimization applies to `SELECT` statements created as a result of requests and is subject to SQL limitations on optimization of DEFINE expressions.

Example: Using IF-THEN_ELSE Optimization Without Aggregation

Consider the following request:

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF1 = IF (LAST_NAME EQ ' ') AND (FIRST_NAME EQ ' ')
        AND (DEPARTMENT EQ 'MIS') THEN 1 ELSE 0;
END

TABLE FILE EMPINFO
PRINT DEPARTMENT LAST_NAME FIRST_NAME
WHERE DEF1 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF1 EQ 1 test:

```
SELECT T1."LN",T1."FN",T1."DPT" FROM USER1."EMPINFO" T1
WHERE (((((T1."LN" = ' ') AND (T1."FN" = ' '))
        AND (T1."DPT" = 'MIS')))) FOR FETCH ONLY;
```

Example: Using IF-THEN_ELSE Optimization With Aggregation

Consider the following request:

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF2 = IF LAST_NAME EQ 'SMITH' THEN 1 ELSE IF LAST_NAME EQ 'JONES' THEN 2
        ELSE IF LAST_NAME EQ 'CARTER' THEN 3 ELSE 0;
END

TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 1
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 1 test:

```
SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1
WHERE (((T1."LN" = 'SMITH')) FOR FETCH ONLY;
```

```
TABLE FILE EMPINFO
SUM MAX.LAST_NAME IF DEF2 EQ 2
END
```

The adapter generates an SQL request that incorporates the IF-THEN-ELSE condition corresponding to the WHERE DEF2 EQ 2 test:

```
SELECT MAX(T1."LN") FROM USER1."EMPINFO" T1
WHERE (((NOT (T1."LN" = 'SMITH')) AND (T1."LN" = 'JONES'))
FOR FETCH ONLY;
```

Optimizing Requests if a Virtual Field Contains Null Values

The SET OPTNOAGGR command provides finely-tuned control of adapter behavior for optimization. Users who for any reason wish to prevent passing aggregation to the RDBMS can use this command. An example of such a reason might be where NULL values occur in aggregated data with calculations. The SET OPTNOAGGR command causes the adapter to generate SQL without passing aggregation to the DBMS. Aggregation is instead performed internally by the server while JOIN and SORT operations are handled by the RDBMS.

If any DEFINE field contains calculations with NULL fields then such operations cannot be translated to SQL and pass to DBMS because always return NULL. It has to be processed by Web Query.

This can be achieved by setting OPTIMIZATION to OFF.

However, in some cases it is preferable to use the off-load JOIN and SORT operation to DBMS for better performance while leaving AGGREGATION to Web Query.

Example: Using IF-THEN_ELSE Optimization With a Condition That is Always False

```
SQL SQLMSS SET OPTIFTHENELSE ON
DEFINE FILE EMPINFO
DEF3 = IF FIRST_NAME EQ 'RITA' THEN 1 ELSE 0;
END
```

```
TABLE FILE EMPINFO
PRINT FIRST_NAME
IF DEF3 EQ 2
END
```

Because DEF3 EQ 2 will never be true, the adapter passes the WHERE test 1=0 (which is always false) to the RDBMS, returning zero records from the RDBMS:

```
SELECT T1."FN" FROM USER1."EMPINFO" T1 WHERE (1 = 0) FOR FETCH ONLY;
```

Reference: SQL Limitations on Optimization of DEFINE Expressions

Since the Web Query reporting language is more extensive than native SQL, the data adapter cannot pass certain DEFINE expressions to the RDBMS for processing. The data adapter does not offload DEFINE-based aggregation and record selection if the DEFINE includes:

- ❑ User-written subroutines.
- ❑ Self-referential expressions such as:
`X=X+1;`
- ❑ EDIT functions for numeric-to-alpha or alpha-to-numeric field conversions.
- ❑ DECODE functions for field value conversions.
- ❑ Relational operators INCLUDES and EXCLUDES.
- ❑ Web Query subroutines ABS, INT, MAX, MIN, LOG, and SQRT.
Note: Do not confuse the Web Query user-written subroutines MAX and MIN with the MAX. and MIN. prefix operators. DEFINE fields cannot include prefix operators.
- ❑ Expressions involving fields with ACTUAL=DATE, except for the subtraction of one DATE field from another and all logical expressions on DATE fields.
- ❑ Date-time manipulation handled by the Web Query date-time functions is not converted to SQL.

In addition, IF-THEN-ELSE optimization does not support the following features:

- ❑ Any type of DECODE expression.
- ❑ STATIC SQL.
- ❑ IF/WHERE DDNAME.
- ❑ Partial date selection.

Calling a Microsoft SQL Server Stored Procedure Using SQL Passthru

How to:

Invoke a Stored Procedure

Example:

Invoking a Stored Procedure

Sample Stored Procedure

Reference:

Capturing Application Errors in Stored Procedures

Microsoft SQL Server stored procedures are supported using SQL Passthru. These procedures need to be developed within Microsoft SQL Server using the CREATE PROCEDURE command.

The adapter supports stored procedures with input, output, and in-out parameters.

The output parameter values that are returned by stored procedures are available as result sets. These values form a single-row result set that is transferred to the client after all other result sets are returned by the invoked stored procedure. The names of the output parameters (if available) become the column titles of that result set.

Note that only the output parameters (and the returned value) referenced in the invocation string are returned to the client. As a result, users have full control over which output parameters have their values displayed.

The server supports invocation of stored procedures written according to the rules of the underlying DBMS. Note that the examples shown in this section are SQL-based. See the DBMS documentation for rules, languages, and additional programming examples.

Syntax: **How to Invoke a Stored Procedure**

```
SQL [SQLMSS] EX procname [parameter_specification1]
[,parameter_specification2]...
END
```

where:

SQLMSS

Is the ENGINE suffix for Microsoft SQL Server.

procname

Is the name of the stored procedure. It is the fully or partially qualified name of the stored procedure in the native RDBMS syntax.

parameter_specification

IN, OUT, and INOUT parameters are supported. Use the variation required by the stored procedure:

IN

Is a literal (for example, 125, 3.14, 'abcde'). You can use reserved words as input. Unlike character literals, reserved words are not enclosed in quotation marks (for example, NULL). Input is required.

OUT

Is represented as a question mark (?). You can control whether output is passed to an application by including or omitting this parameter. If omitted, this entry will be an empty string (containing 0 characters).

INOUT

Consists of a question mark (?) for output and a literal for input, separated by a slash: /. (For example: ?/125, ?/3.14, ?/'abcde'.) The out value can be an empty string (containing 0 characters).

Example: Invoking a Stored Procedure

In this example, a user invokes a stored procedure, `edaqa.test_proc01`, supplies input values for parameters 1, 3, 5 and 7, and requests the returned value of the stored procedure, as well as output values for parameters 2 and 3.

Note that parameters 4 and 6 are omitted; the stored procedure will use their default values, as specified at the time of its creation.

```
SQL SQLMSS EX edaqa.test_proc01 125,?,?/3.14,, 'abc',, 'xyz'
END
```

Example: Sample Stored Procedure

This stored procedure uses out and inout parameters:

```
CREATE PROCEDURE EDAQA.PROCP3 (    OUT chSQLSTATE_OUT    CHAR(5),
                                OUT intSQLCODE_OUT    INT,
                                INOUT l_name char(20),
                                INOUT f_name char(20))

    RESULT SETS 1
    LANGUAGE SQL

-----
-- SQL Stored Procedure
-----

P1: BEGIN
    -- Declare variable
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE SQLCODE INT DEFAULT 0;
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT
            EDAQA.NF29005.SSN5 AS SSN5,
            EDAQA.NF29005.LAST_NAME5 AS LAST_NAME5,
            EDAQA.NF29005.FIRST_NAME5 AS FIRST_NAME5,
            EDAQA.NF29005.BIRTHDATE5 AS BIRTHDATE5,
            EDAQA.NF29005.SEX5 AS SEX5
        FROM
            EDAQA.NF29005
        WHERE
            (
                ( EDAQA.NF29005.LAST_NAME5 = l_name )
            AND
                ( EDAQA.NF29005.FIRST_NAME5 = f_name )
            );
    -- Cursor left open for client application
    OPEN cursor1;
    SET chSQLSTATE_OUT = SQLSTATE;
    SET intSQLCODE_OUT = SQLCODE;
    SET l_name = 'this is first name';
    SET f_name = 'this is last name';
END P1    @
```

Reference: Capturing Application Errors in Stored Procedures

You can capture application errors using the RAISERROR method. Any application error that is issued by the stored procedure is available in the server variable &MSSMSGTXT.