SQL Server 2012 Reporting Services and Teradata Database

SQL Server Technical Article

Writer: Houman Ghaemi, Simba Technologies

Contributors: Craig Guyer, Mary Lingel

Technical Reviewers: Cal Arabshahi, Aaron Myers, Rupal Shah, Andre Magni

Project Editor: Deborah Dinzes

Published: January 2014

Applies To: Microsoft SQL Server 2012

Summary: This article discusses the connectivity and usage of Microsoft SQL Server 2012 Reporting Services (SSRS) with the Teradata Database using the .NET Data Provider for Teradata. For Teradata users that are new to working with Reporting Services, this article aims to present tips and Teradata features that can help make the most of SQL Server Reporting Services. For Reporting Services users that are new to using Teradata as a data source, the article serves as an introduction to working with different data types and characteristics that are specific to the Teradata database.

Introduction: Reporting Services Connectivity with Teradata

Since the release of SQL Server 2008 Reporting Services (SSRS), Teradata users can take advantage of a rich report authoring environment. Reporting Services can interoperate with Teradata using the .NET Data Provider for Teradata and Teradata users can continue to take full advantage of Reporting Services capabilities without migrating data to another platform.

About This Document

This article is designed for customers and partners who are interested in using Reporting Services with a Teradata relational database. Topics covered in this article include:

- Prerequisites, installation, and configuration
- Teradata-specific terms and concepts
- Teradata client tools
- Report design and Teradata relational data sources
- Leveraging Teradata Database features
- Teradata Database native data types
- Troubleshooting tips
- Appendix A – Teradata Connectivity
- Appendix B – Teradata Database

This article is intended to complement the documentation available in SQL Server Books Online. This article assumes a basic understanding of Reporting Services, report server projects and Report Builder 3.0. It is recommended that the reader complete the tutorials in SQL Server Books Online, or acquire an equivalent know-how on the prerequisite topics.

Prerequisites

To use a Teradata Database as a data source with Reporting Services, you need the following components:

· Microsoft SQL Server Reporting Services

· .NET Data Provider for Teradata

The .NET Data Provider for Teradata is the ‘advocated’ interface for Reporting Services. The .NET Data Provider for Teradata is an implementation of the Microsoft ADO.NET specification. Once installed, the .NET Data Provider for Teradata supports all required ADO.NET interfaces and classes.

The following table summarizes the ‘official’ supported configurations with SQL Server 2012 and 2008 Reporting Services:

Teradata Database	14.10	14.0	13.10	13.0	12.0
SQL Server 2008		√	√	√	√
SQL Server 2008 R2		√	√	√	√
SQL Server 2012	√	√	√	√	

Below is Teradata versioning for ‘major’, ‘minor’ and ‘bug or enhancement’ release.

- 14.00.00.xx is a *‘major’* release
- 14.10.00.xx is a *‘minor’* release
- *xx* denoted intermediate releases to address issues and/or enhancement (i.e. 14.00.00.01)

The .NET Data Provider for Teradata is ‘backward’ and ‘forward’ compatible with the Teradata Database and SQL Server Reporting Services versions. For example, .NET Data Provider for Teradata version 13.10.00.xx can work with Teradata Database 14.10.00.xx and SQL Server 2008. Or .NET Data Provider for Teradata version 14.10.00.xx can work with Teradata Database 14.00.00.xx and SQL Server 2012.

Higher Versions of .NET Data Provider for Teradata

Starting with .NET Data Provider for Teradata version 12.00.01.00 and higher, you need to select the **Publisher Policy** option from the **Select Features** select dialog box when installing or upgrading the .NET Data Provider for Teradata. Publisher policy installs on default when you choose **Setup Type 'Complete'**

The publisher policy enables 'backwards' compatibility with applications built from an earlier version of the .NET Data Provider for Teradata. The publisher policy will automatically bind the higher version of the .NET Data Provider for Teradata version at run time. For example, if Reporting Services used version 13.10 of the provider, and the user installs version 14.10 and installs the publisher policy file. At runtime, CLR will load the Data Provider version 14.10 instead of looking for the Data Provider version 13.10.

Installation Overview

A default installation of Reporting Services has the capability of connecting with a Teradata database using the .NET Data Provider for Teradata. The .NET Data Provider for Teradata can be downloaded from the Teradata web site here or is now available on Teradata Tools and Utilities suite version 14.10 or higher..

For more information about installation of the .NET Data Provider for Teradata, see the Readme file that accompanies the provider or search the Teradata web site for additional help.

SQL Server Data Tools (SSDT)

The SQL Server Data Tools is one of the report authoring environments for Reporting Services 2012. You can also author reports using Report Builder 3.0. Report Builder is a click-once application that supports the full capabilities presented in this document, for more information about Report Builder click here.

This section gives a brief overview of SQL Server Data Tools, and any data source specific notes associated with using a Teradata data source in SQL Server Data Tools.

Unlike Reporting Services 2008 which had two types of report projects that can be created in Business Intelligence Development Studio. The first being, report server and the second being report model projects. In Reporting Services 2012 report model has been dropped and only report server projects can be created in SQL Server Data Tools for reporting against Teradata relational database. A report server project supports the traditional design experience in SQL Server Data Tools.

Report Server project type is discussed in more detail in paper.

Report Server Projects

The purpose of the report server project, in the context of Teradata, is to create a report by directly querying the database and using the rich report design options available in SQL Server Data Tools. The following is a summary of the steps that a typical user goes through to create and deploy a report:

1. Within a report server project, a data source is created. Data sources are representative of the database connection.

2. One or more datasets are created based on a data source. A dataset is a table of rows and columns that are returned by a query.

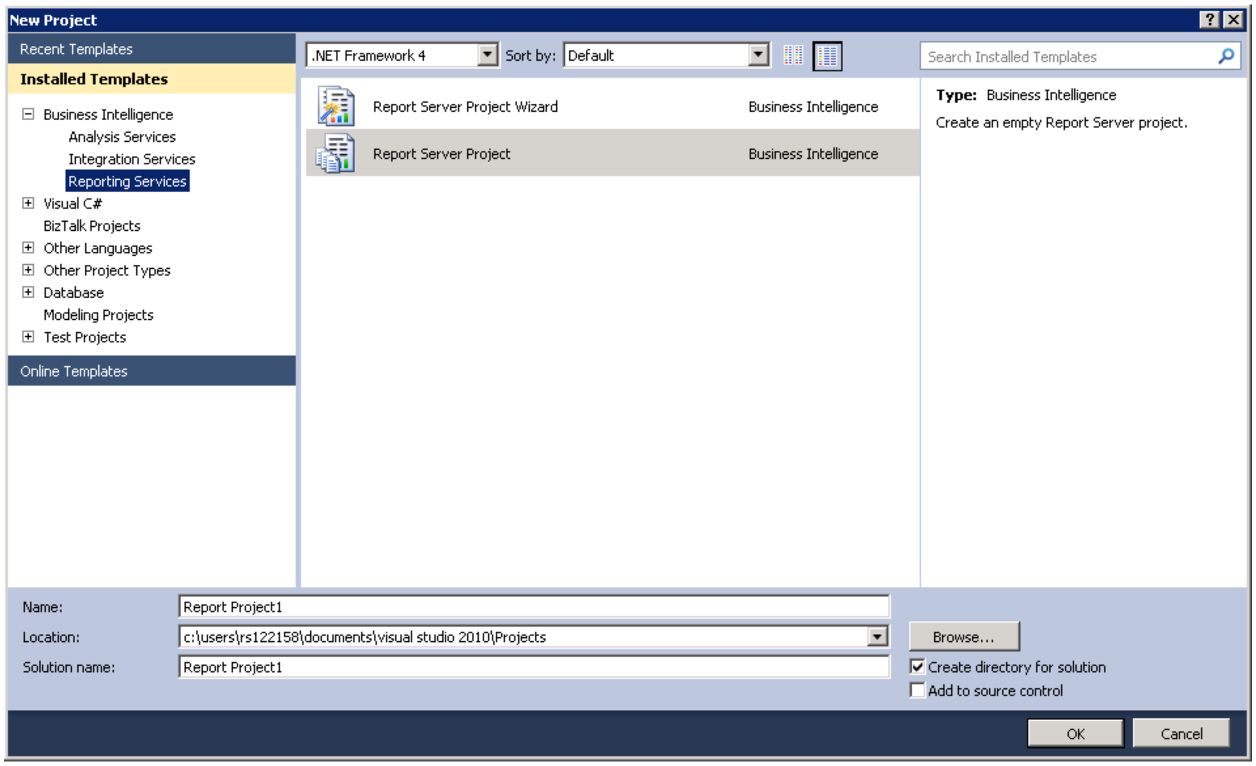
3. A report, which is a visual representation of the data returned by the data set (or query), is created from the datasets.

4. The report is deployed to the report server.

Upon deployment, end-users can request the server to process the report to see the final results. This section describes some of the steps involved in detail. Specifically, the steps are focused on nuances of working with Teradata as a data source. For more information about working with report server projects, see Reporting Services Tutorials.

Creating a Report Server Project and Data Source

1. In SQL Server Data Tools, click **File**, then **New**, and then **Project**. Select **Reporting Services** from the **Installed Templates** section and select **Report Server Projects** from the list of project templates as shown in Figure 1.


New Project

.NET Framework 4 Sort by: Default

Installed Templates: Business Intelligence > Reporting Services

Report Server Project Wizard	Business Intelligence
Report Server Project	Business Intelligence

Type: Business Intelligence
Create an empty Report Server project.

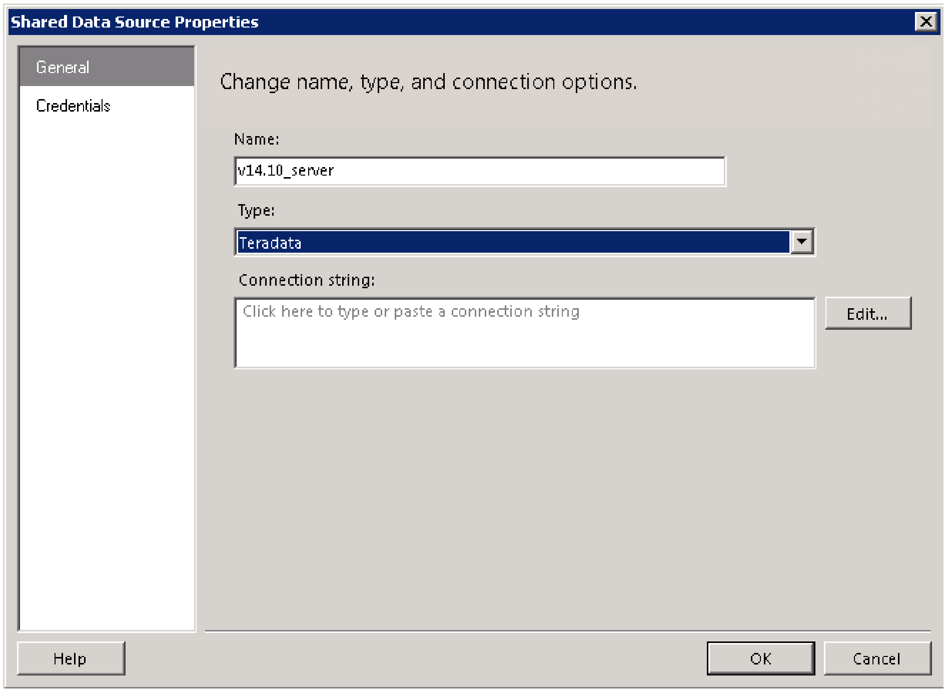
Name: Report Project1
Location: c:\users\rs122158\documents\visual studio 2010\Projects
Solution name: Report Project1
[✓] Create directory for solution
[ ] Add to source control

Figure 1: Creating a report server project

After you create a new project, you will see three nodes (or folders) in the Solution Explorer window. If Solution Explorer is not visible, you can activate it by selecting **View** then **Solution Explorer**, or alternatively, by using the keyboard shortcut **Ctrl+Alt+L**. The first node (or folder) is **Shared Data Sources**, the second is Shared Datasets and third second is **Reports**.

At this point, you have two choices. First, you can create a shared data source to be used in all your reports, or second, create a report with its private (i.e. embedded) data source.

2. Create a shared data source by right-clicking on the **Shared Data Sources** folder and selecting **Add New Data Source**. Type a name for your data source and select Teradata for **Type** as shown in Figure 2.


Shared Data Source Properties

Change name, type, and connection options.

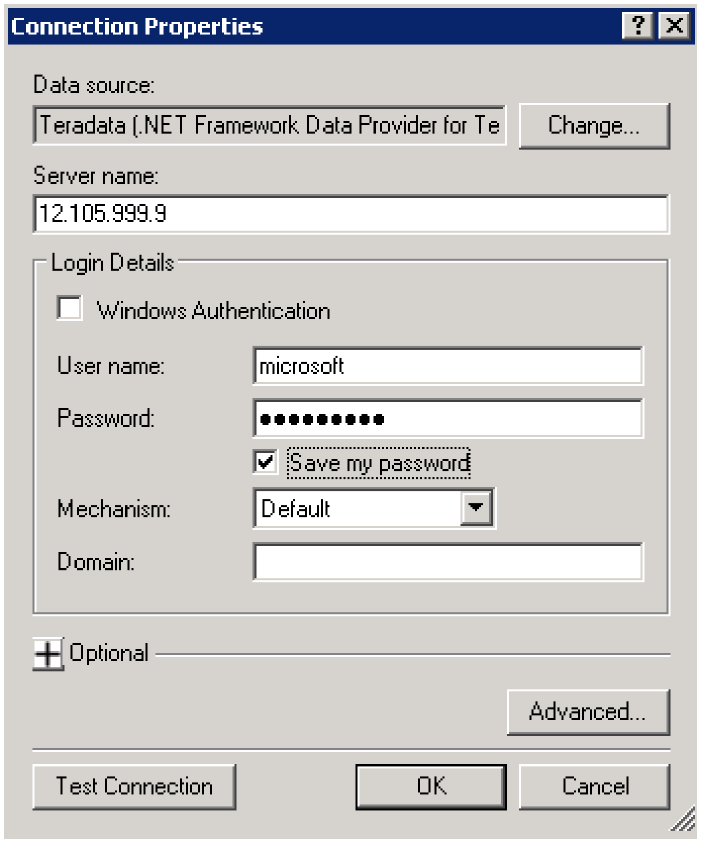
Name: v14.10_server

Type: Teradata

Connection string: Click here to type or paste a connection string

Figure 2: Creating a shared data source

3. You can either enter a connection string or click **Edit** to build one which will open the **Connection Properties** dialog box shown in Figure 3.



Data source: Teradata (.NET Framework Data Provider for Te

Server name: 12.105.999.9

Login Details

☐ Windows Authentication

User name: microsoft

Password: ●●●●●●●●●

☑ Save my password

Mechanism: Default

Domain:

Optional

Figure 3: Teradata Connection Properties dialog box

The ‘basic’ connection information to access Teradata requires Server name, User name and Password. The **Connection Properties** dialog box is customized for Teradata and was installed when the .NET Data Provider for Teradata was installed. Note that all fields are disabled until a server name or IP address is entered in the **Server name** box.

Note, the .NET Data Provider for Teradata uses the Teradata system (TDPID or DBC-Name) name. For more information, see *"Teradata Host Naming Convention"* section in this article.

After filling in your username and password, click **Test Connection** and you should see a message box noting *‘Test connection succeeded’*. If you don’t see this message, you may need to click Teradata Authentication Mechanisms and change the authentication mechanism using the **Mechanism** box (as shown in Figure 3). Otherwise, check with your administrator to ensure you have access to the Teradata Database.

The **Advanced** button enables you to edit all connection string properties. For optimal experience, see *Appendix A – Teradata Connectivity* for more information about connection string properties and recommendations or see the Teradata help files that are installed with the .NET Data Provider for Teradata.

Creating a New Report and Data Set - Report Server Project Type

1. In Solution Explorer, right-click on the **Reports** node and then select **Add New Report** as shown in Figure 4.

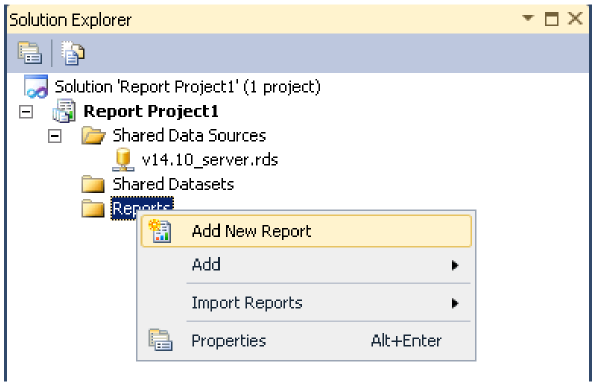
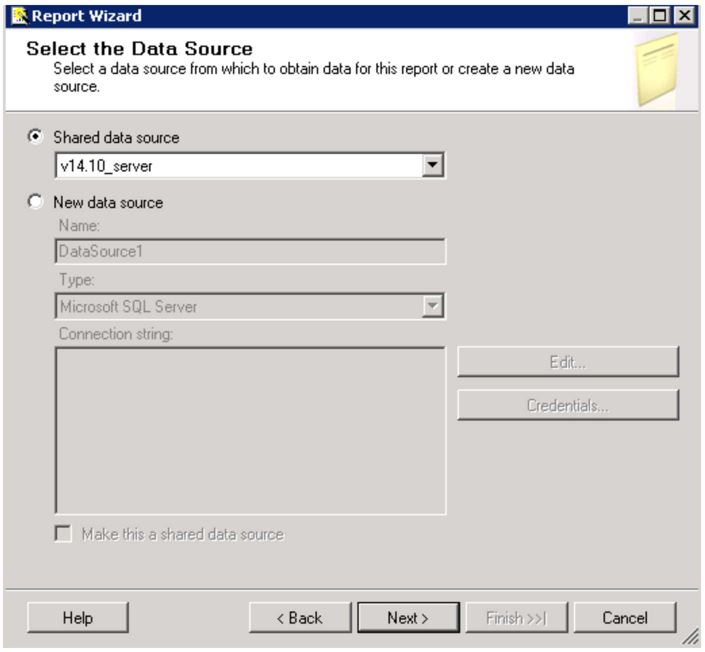


Figure 4: Adding a new report in Solution Explorer

2. The Report Wizard is now open. If you see the **Welcome** page, click **Next** to move to the **Select the Data Source** page as shown in Figure 5.


Report Wizard

Select the Data Source

Select a data source from which to obtain data for this report or create a new data source.

(●) Shared data source: v14.10_server

( ) New data source

Name: DataSource1

Type: Microsoft SQL Server

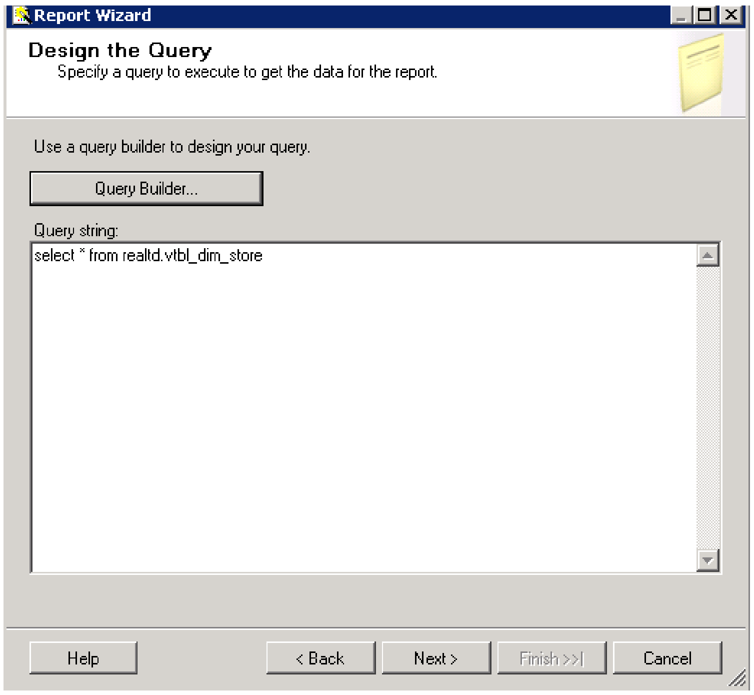
Connection string:

[ ] Make this a shared data source

Figure 5: Specifying a data source

At this point, you have the option of selecting a shared data source or creating a new data source. Click **Next** to continue.

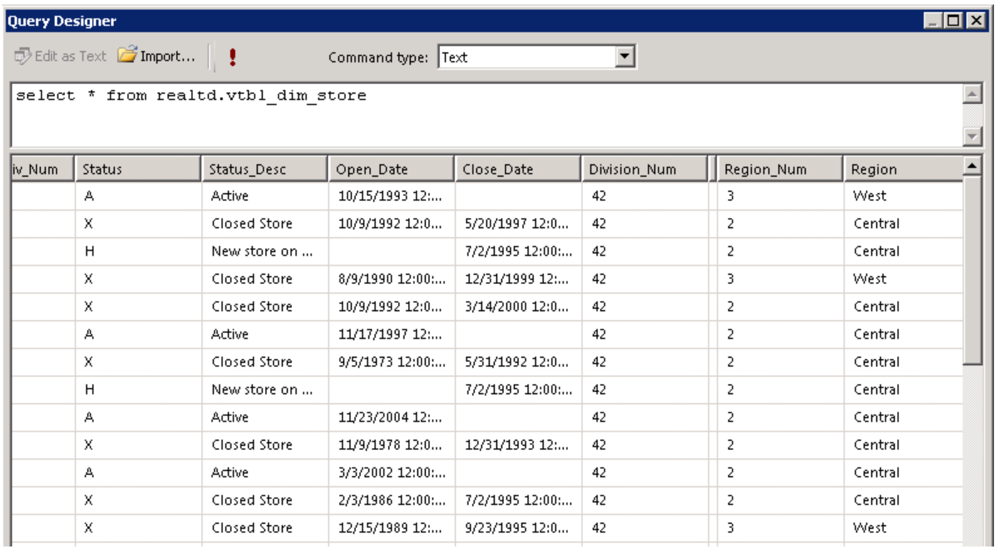
3. On the **Design the Query** page, you can type in your query, or click on **Query Builder** to create a data set as shown in Figure 6.


Report Wizard
Design the Query
Specify a query to execute to get the data for the report.
Use a query builder to design your query.
Query Builder...
Query string:
Help
< Back
Next >
Finish >>|
Cancel

```
select * from realtd.vtbl_dim_store
```

Figure 6: Report Wizard Design the Query page

4. If you click **Query Builder**, then you will see the text-based query designer as shown in Figure 7.



```
select * from realtd.vtbl_dim_store
```

iv_Num	Status	Status_Desc	Open_Date	Close_Date	Division_Num	Region_Num	Region
	A	Active	10/15/1993 12:...		42	3	West
	X	Closed Store	10/9/1992 12:0...	5/20/1997 12:0...	42	2	Central
	H	New store on ...		7/2/1995 12:00:...	42	2	Central
	X	Closed Store	8/9/1990 12:00:...	12/31/1999 12:...	42	3	West
	X	Closed Store	10/9/1992 12:0...	3/14/2000 12:0...	42	2	Central
	A	Active	11/17/1997 12:...		42	2	Central
	X	Closed Store	9/5/1973 12:00:...	5/31/1992 12:0...	42	2	Central
	H	New store on ...		7/2/1995 12:00:...	42	2	Central
	A	Active	11/23/2004 12:...		42	2	Central
	X	Closed Store	11/9/1978 12:0...	12/31/1993 12:...	42	2	Central
	A	Active	3/3/2002 12:00:...		42	2	Central
	X	Closed Store	2/3/1986 12:00:...	7/2/1995 12:00:...	42	2	Central
	X	Closed Store	12/15/1989 12:...	9/23/1995 12:0...	42	3	West

Figure 7: Text-based query designer

Note, Query Designer does not support UI for Teradata data sources ***only*** *'Text' editor.*

This query designer allows you to directly edit the SQL command text sent to the database. A graphical query designer, such as the designer used for SQL Server, is not available when using the Teradata provider. The text-only designer is used for several data source types in Reporting Services and Report Builder therefore is not optimized for any one data source type in particular.

In the Dataset Properties window, the only Query type available is Text. Hence, in the Query Designer window, the **Command Type** box only contains the Text option. The Text mode is the most commonly used and allows you to enter a standard SQL query, Teradata Stored Procedure and a Teradata Macro for a dataset. TableDirect is not supported by SQL Server Reporting Services for a Teradata data source.

5. After you enter your query and click **OK**, you will see additional steps in the wizard which allow you to format your report fields and finally you will see a report in the design view as shown in Figure 8.

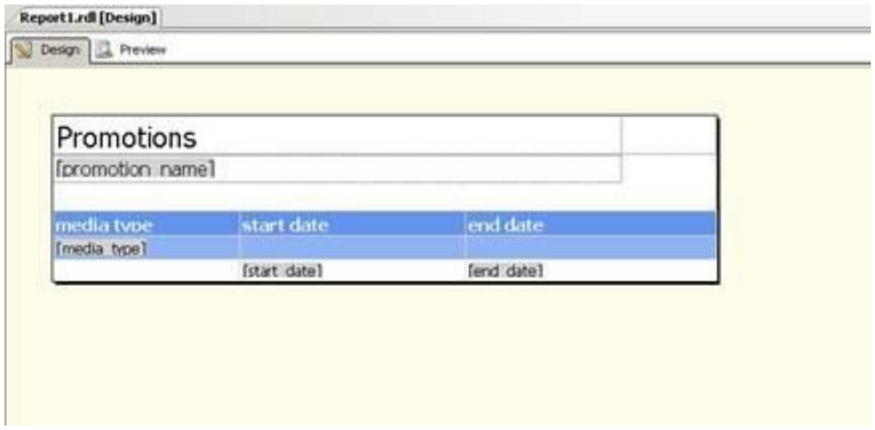


Figure 8: Finished report in design view

Previewing and Deploying the New Report

1. You can now see the dataset you created in the Report Wizard in the **Report Data** window. The **Report Data** window can be opened by selecting **View**, then **Report Data.** An example of a dataset is shown in Figure 9.

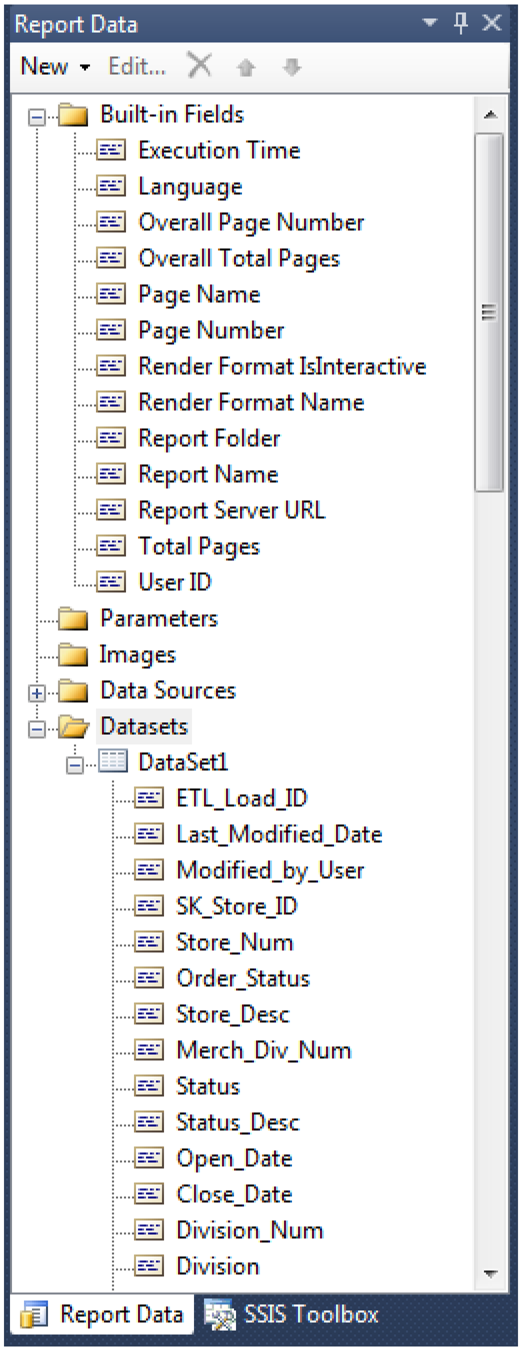
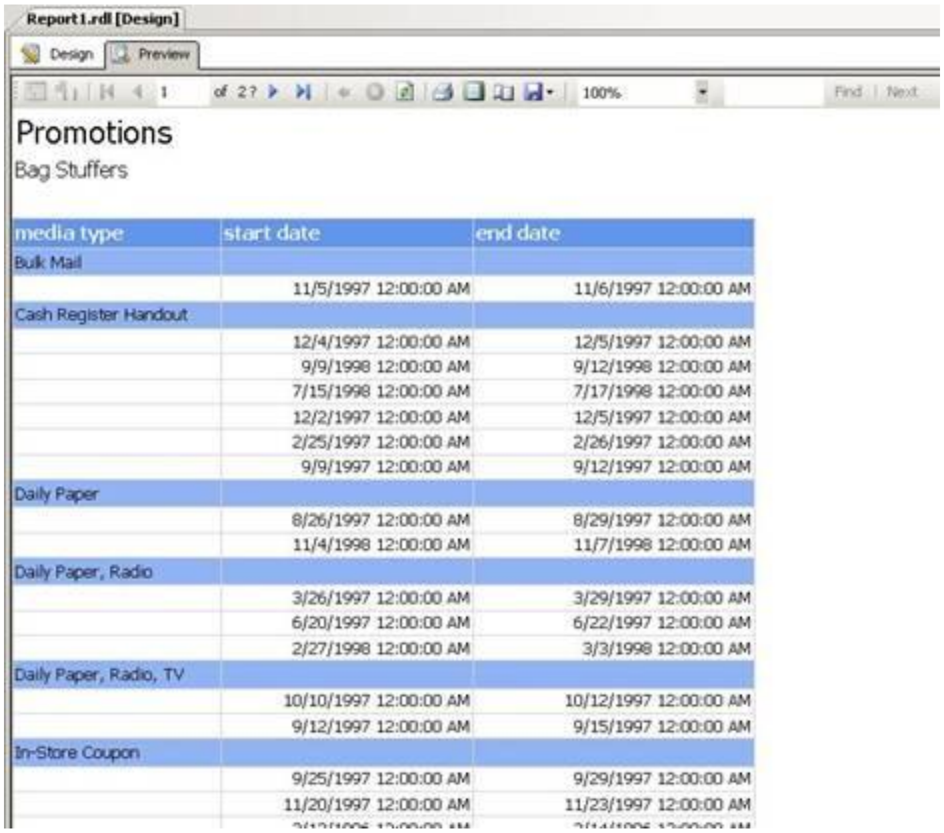


Figure 9: Report Data window showing new data sets

2. To see how your new report looks, click **Preview**. A sample report preview is shown in Figure 10.



Promotions

Bag Stuffers

media type	start date	end date
Bulk Mail		
	11/5/1997 12:00:00 AM	11/6/1997 12:00:00 AM
Cash Register Handout		
	12/4/1997 12:00:00 AM	12/5/1997 12:00:00 AM
	9/9/1998 12:00:00 AM	9/12/1998 12:00:00 AM
	7/15/1998 12:00:00 AM	7/17/1998 12:00:00 AM
	12/2/1997 12:00:00 AM	12/5/1997 12:00:00 AM
	2/25/1997 12:00:00 AM	2/26/1997 12:00:00 AM
	9/9/1997 12:00:00 AM	9/12/1997 12:00:00 AM
Daily Paper		
	8/26/1997 12:00:00 AM	8/29/1997 12:00:00 AM
	11/4/1998 12:00:00 AM	11/7/1998 12:00:00 AM
Daily Paper, Radio		
	3/26/1997 12:00:00 AM	3/29/1997 12:00:00 AM
	6/20/1997 12:00:00 AM	6/22/1997 12:00:00 AM
	2/27/1998 12:00:00 AM	3/3/1998 12:00:00 AM
Daily Paper, Radio, TV		
	10/10/1997 12:00:00 AM	10/12/1997 12:00:00 AM
	9/12/1997 12:00:00 AM	9/15/1997 12:00:00 AM
In-Store Coupon		
	9/25/1997 12:00:00 AM	9/29/1997 12:00:00 AM
	11/20/1997 12:00:00 AM	11/23/1997 12:00:00 AM
	[illegible]	[illegible]

Figure 10: Report preview

3. The final step in the cycle of report creation is to deploy the report to a Reporting Services server where it can be viewed by many users. For more information about report deployment and processing, see Publishing Data Sources and Reports.

For more information about how to work with datasets and reports, see the following: Report Design Basics and Reporting Services in SQL Server Data Tools.

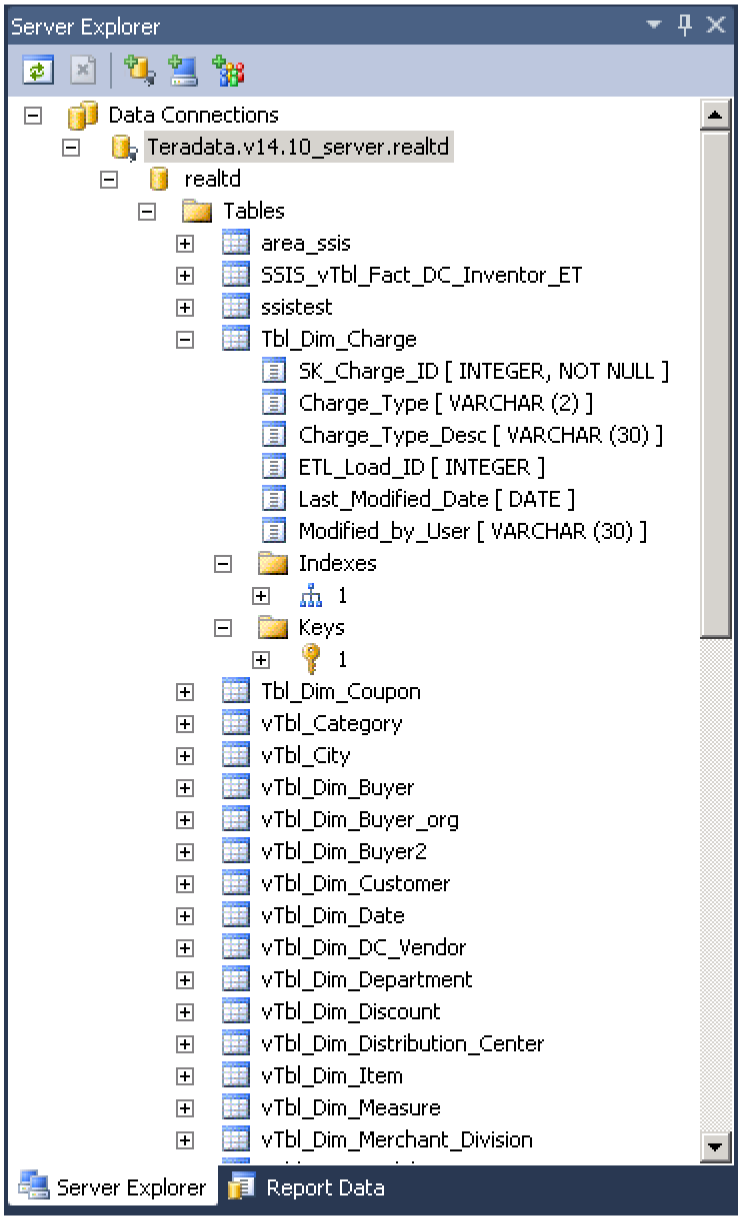
How to Explore Teradata Database Entities

As noted earlier, the Report Designer in SQL Server Data Tools provides a text-based query designer when working with a Teradata data source.

To assist in query creation, Server Explorer, found in SQL Server Data Tools, can be used to graphically explore a Teradata database structure as well as author queries which can be copied and pasted into a Reporting Services dataset.

To access Server Explorer in SQL Server Data Tools, select menu item **View**, then **Other Windows**, then **Server Explorer**. You can also use the keyboard short cut <**Ctrl>+<Alt>+<S>**. Server Explorer not only reveals the tables, views, and stored procedures; it also shows the fields, and primary and foreign keys of the tables.

Figure 11 shows an example of a Server Explorer window.



Server Explorer

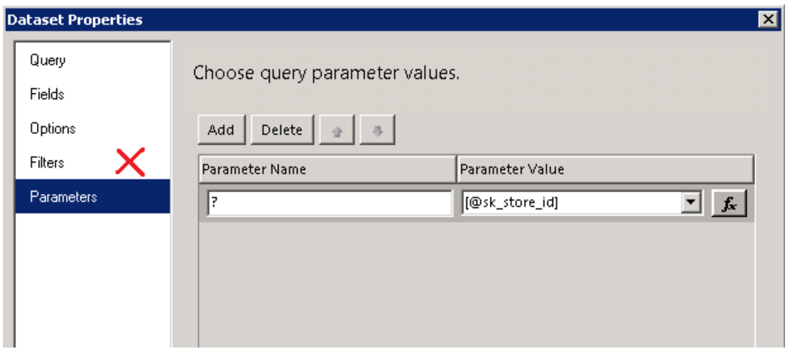
- Data Connections
  - Teradata.v14.10_server.realtd
    - realtd
      - Tables
        - area_ssis
        - SSIS_vTbl_Fact_DC_Inventor_ET
        - ssistest
        - Tbl_Dim_Charge
          - SK_Charge_ID [INTEGER, NOT NULL]
          - Charge_Type [VARCHAR (2)]
          - Charge_Type_Desc [VARCHAR (30)]
          - ETL_Load_ID [INTEGER]
          - Last_Modified_Date [DATE]
          - Modified_by_User [VARCHAR (30)]
          - Indexes
            - 1
          - Keys
            - 1
        - Tbl_Dim_Coupon
        - vTbl_Category
        - vTbl_City
        - vTbl_Dim_Buyer
        - vTbl_Dim_Buyer_org
        - vTbl_Dim_Buyer2
        - vTbl_Dim_Customer
        - vTbl_Dim_Date
        - vTbl_Dim_DC_Vendor
        - vTbl_Dim_Department
        - vTbl_Dim_Discount
        - vTbl_Dim_Distribution_Center
        - vTbl_Dim_Item
        - vTbl_Dim_Measure
        - vTbl_Dim_Merchant_Division

Figure 11: Server Explorer

Working with Report and Query Parameters

Report parameters allow end users to enter values (or use specified defaults), which are used when a report is processed. The new value replaces the parameter placeholder. For example, a report based on sales by country could support a parameter for country name. Therefore, the end user may choose the countries with which to filter the report at report execution time.

In Reporting Services, there are two types of parameters in a report: **report parameters** and **query parameters**. Report parameters (or filters) that are not tied to a query parameter are processed by the report server, while query parameters are processed on the data source server. Depending on your reporting requirements, design reports to use query parameters versus report filters to push processing to Teradata Database (Figure P-1)

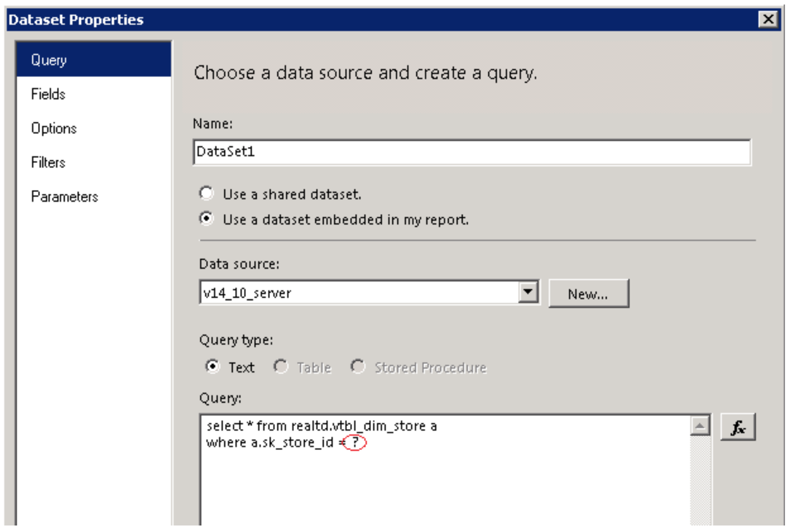


Choose query parameter values.

Parameter Name	Parameter Value
?	[@sk_store_id]

Figure P-1: Dataset Properties page for Query Parameter setting

Using hard-coded or parameterized (user prompt) queries, ensures backend database performs the heavy lifting and returns data which is required for the request as shown in Figure P-2.



Choose a data source and create a query.

Name: DataSet1

Use a shared dataset. [✓ Use a dataset embedded in my report.]

Data source: v14_10_server

Query type: [✓ Text] Table Stored Procedure

Query:

```
select * from realtd.vtbl_dim_store a
where a.sk_store_id = ?
```

Figure P-2: Dataset Properties page for Query statement

Whereas, report filters will process in the middle-tier against more data than requests may require

In some scenarios, you will want to use report parameters. For example, you have a large result set that takes a long time to process and your data will not change during the day, you might consider taking a snapshot of the report and you can use the report parameters to properly filter the report. For more information on Snapshot execution and report filtering, refer to Performance, Snapshots, Caching (Reporting Services).

In addition, there are two types of report parameters that one can create: a single-value parameter and multi-value parameter.

The report parameters can be named or unnamed.

Teradata only supports unnamed parameters. An unnamed parameter is denoted by a *'?'*, and is merely a placeholder for data that is going to be entered at report processing time. For example, the following is a dataset with a single-value unnamed query parameter:

```
Select * from A where A.aid = ?;
```

For a multi-value parameter use an IN clause with *'(?)'*, the end-user has the option to select from a list of available values. The following query uses a multi-value parameter:

```
Select * from A where A.aid in (?);
```

For more information about the Reporting Services report parameters, see the following: Adding Parameters to Your Report and Using Single-Valued and Multivalued Parameters.

Working with Teradata Macros

Teradata macros are similar concepts to stored procedures. A Teradata macro can be called using *exec <macro name>*in the **Design the Query** page as shown in Figure 6. A Teradata macro normally returns at least one result set. For example, the following macro will generate two result sets:

```
 replace macro get_promo(ID smallint) as (

      sel * from promotion where promotion_id = :ID;

      sel * from sales_fact_1997 where promotion_id = :ID;);
```

The Reporting Services query designer will only return and process the first result set. If you have macros that return more than one result set from which you need the data, then you need to create wrapper macros that join the result sets into a single result set, or modify your report design to use multiple datasets; for example, showing two tables rather than one.

Working with Teradata Query Band Feature

Though, no ‘official’ integration or support exists between Microsoft Reporting Services and Teradata Query Band feature, there are some ‘limited’ ‘out-of-the-box’ capabilities for governance and managing Teradata database workloads. To enable appropriate priority and Teradata database resources to reports and users to meet agreed upon SLA(s) requirements.

The Teradata Query Bands feature was introduced in Teradata Database v12 to provide a means to set *name/value* pairs across individual Database connections at a Session or Transaction level to provide the database with significant information about the connections originating source. This provides a mechanism for Applications to collaborate with the underlying Teradata Database in order to provide for better Workload Management, Prioritization and Accounting. Through utilizing of this feature it is expected that improvements in Supportability, Logging, Security and Notification will also be achieved[1].

At a ‘high-level’, the .NET Data Provider for Teradata supports and enables the provider to manage Query Bands for applications. Query Band can be defined within the connection string using the Query Band attribute using *name/value* pairs. Depending on your requirements, this can be achieved with Reporting Services shared or embedded data source connections. Using either ‘static/hard-coding’ Query Band attributes *key=value* pairs (Figure 12) or using Reporting Services Expression-based Connection String (ECS) editor (Figure 13) to pass ‘dynamic’ values to Query Band attributes from Reporting Services built–in fields and/or report parameters for your connection string.

Note, it is assume readers of this section are familiar with Teradata Query Band feature[2] and Reporting Services Expression-based Connection String feature.

Creating a ‘static’ or ‘hard-coded’ Query Band statement

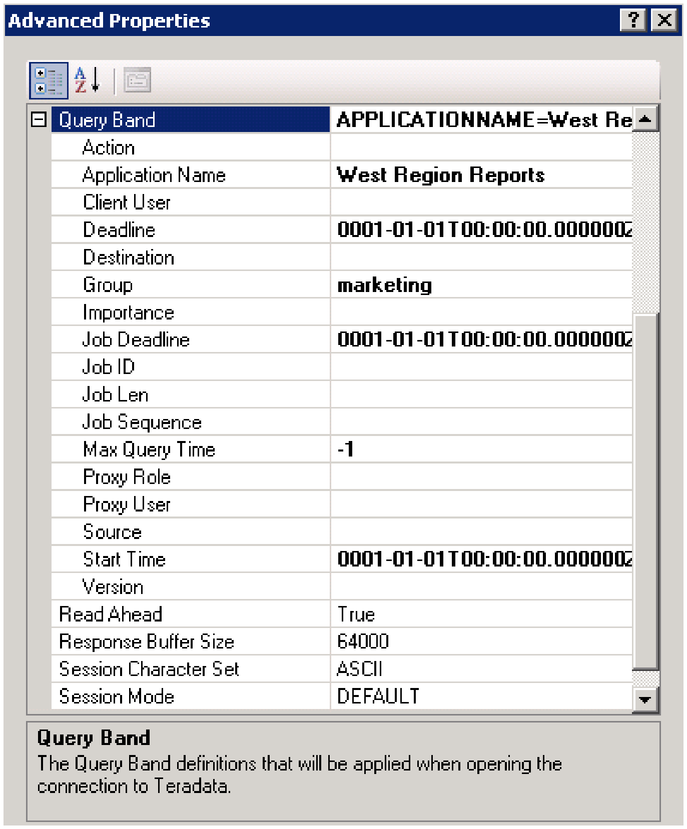
To submit a Query Band statement at an application level for a group of reports and/or user group or for a ‘single’ user, a ‘static/hard-coded’ Query Band statement might suffice. Simply open your shared or embedded data source and enter appropriate Query Band information. The following is a summary of the steps that a typical user goes through to define a static or hard-coded Query Band within the connection string using the Query Band attribute.

1. Within a report server project, create or open a **Shared** or **Embedded** data source which represents your database connection and click edit button (Figure 2 above)

2. In **Connection Properties**, click on **Advance** button to open **Advance Properties**. Expand Query Band attribute (Figure 12)

[1] Referenced Teradata Orange Book – “Reserved” Query Band Names for use by Teradata, Customer and Partner applications

[2] Referenced Teradata Orange Book – Using Query Banding in Teradata Database



Advanced Properties

Query Band	APPLICATIONNAME=West Re
Action	
Application Name	West Region Reports
Client User	
Deadline	0001-01-01T00:00:00.0000002
Destination	
Group	marketing
Importance	
Job Deadline	0001-01-01T00:00:00.0000002
Job ID	
Job Len	
Job Sequence	
Max Query Time	-1
Proxy Role	
Proxy User	
Source	
Start Time	0001-01-01T00:00:00.0000002
Version	
Read Ahead	True
Response Buffer Size	64000
Session Character Set	ASCII
Session Mode	DEFAULT

Query Band
The Query Band definitions that will be applied when opening the connection to Teradata.

Figure 12: Advanced Properties page for Query Band attributes

3. By setting values to any Query Band keys, the Query Band definition will be applied when opening a connection to Teradata. For example, setting Application Name=’West Regional Reports’ and Group=’marketing’ will generate the following connection string and Query Band statement to Teradata.

   <u>Connection String</u>
   Database=marketing; Data Source=12.105.999.9; ….
   *Query Band="**APPLICATIONNAME=West Regional Reports; GROUP=marketing**;"*

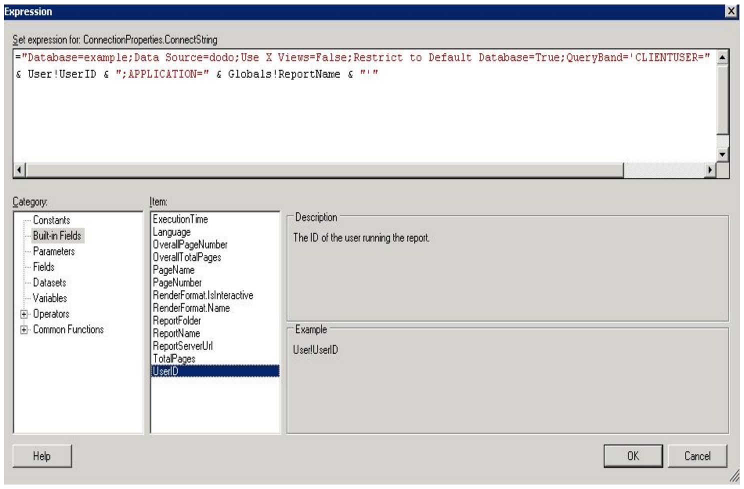
   <u>Query Band statement in Teradata DBQL logs</u>
   *SET QUERY_BAND = **‘GROUP=marketing; APPLICATION=West Regional Reports**;*
   QueryIssueTime = 2012-03-02T14:01:38.316188Z;' FOR SESSION
   SET QUERY_BAND = NONE FOR SESSION

4. Type in your query on the **Design the Query** page, click **Finish** and **Preview** your report for correctness.

5. The final step of deploying the report keeps the Query Band settings intact. Furthermore, Report Builder users can use the shared data source which will continue to leverage the Query Band settings in place.

Creating a ‘dynamic’ Query Band statement

To provide more details about the individual user and report will require creating a ‘dynamic’ Query Band definition with Reporting Services Expression-based Connection String (ECS) editor (Figure 13). This will provide the level of detail for Teradata workload management requirements and sites that require end-users to be individually identified and authorized using Teradata Trusted Sessions. Since, the .NET Data Provider for Teradata supports Query Band, it supports Trusted Sessions using specific Query Band *key=value* pair called **Proxyuser** and **Proxyrole**. When a proxy user requests database access, the application forwards the user identity and applicable role information to the database in the form of a Query Band statement.



```
="Database=example;Data Source=dodo;Use X Views=False;Restrict to Default Database=True;QueryBand='CLIENTUSER="
& User!UserID & ";APPLICATION=" & Globals!ReportName & "'"
```

Figure 13: Expression Editor

The expression must be written in Visual Basic and is processed at run time or when the report is previewed. The following requirements apply to be able to use ECS when defining a data source expression:

- Use Report Wizard
- Must use ‘Embedded’ (not Shared) data source connections for ‘each’ report
- ‘Shared’ data source connections do not support ECS definition
- Must use .NET Data Provider for Teradata
- Specify credentials separately from the connection string
- Can use a report parameter and/or built in field to specify a ‘static’, query-based value or built in value for a Query Band parameter and values
- Before publishing the report, replace the static connection string with an expression
- Updates to a report may require converting between ECS and ‘static’ connection

See Expression-based Connections String for more information.

At a ‘high-level’, design the report with a ‘static’ connection string which allows you to connect to the data source in Report Designer so you can get the query results you need to create the report. Test your report and then replace ‘static’ connection string with an expression based connection string before deploying report. The following is a summary of the steps that a user goes through to define a ‘dynamic’ Query Band to their report using Expression-based

Connection String feature to supply 'run time' values to the Query Band attribute in the connection string.

1. Within a report server project, right-click on the **Reports** node in Solution Explorer and select **Add New Report** as shown in Figure 14 or double-click on **Report1** from earlier example above and go to step 4 to change the data source from shared to embedded.

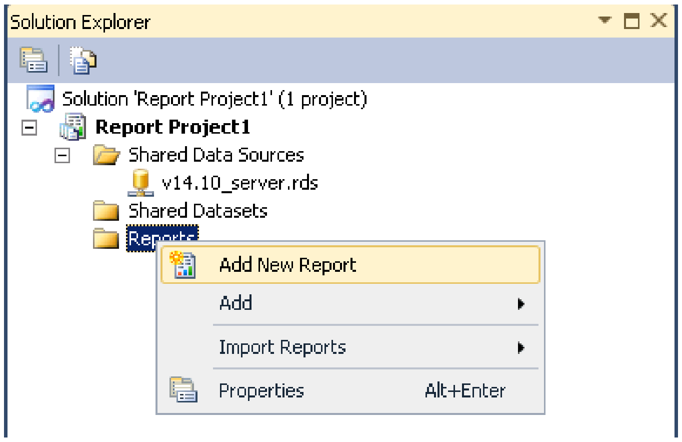
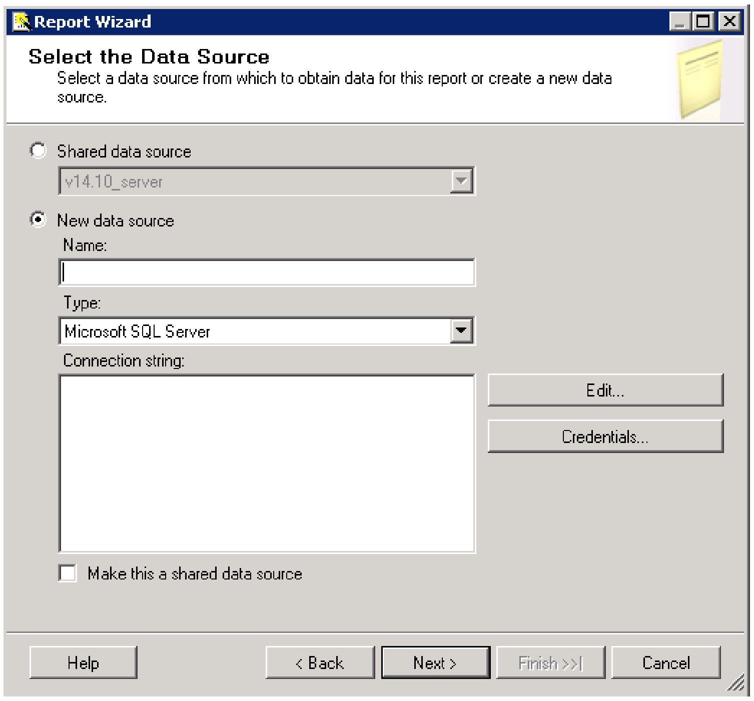


Figure 14: Adding a new report in Solution Explorer

2. The Report Wizard is now open. If you see the **Welcome** page, click **Next** to move to the **Select the Data Source** page as shown in Figure 15.


Report Wizard

Select the Data Source

Select a data source from which to obtain data for this report or create a new data source.

Shared data source: v14.10_server

[● New data source]

Name:

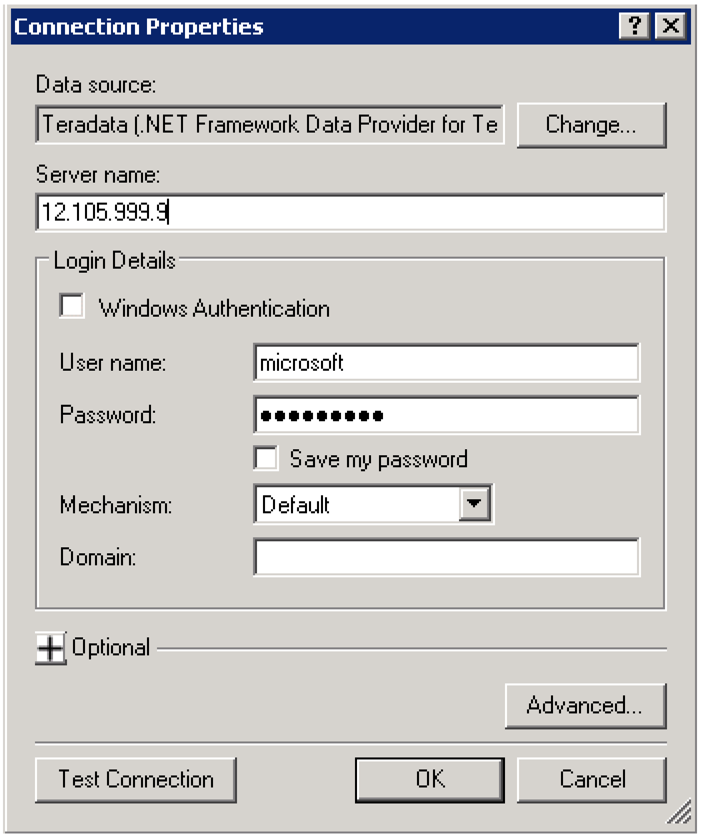
Type: Microsoft SQL Server

Connection string:

Make this a shared data source

Figure 15: Specifying a data source

3. At this point, you have the option of selecting a shared data source or creating a new data source. If you choose **New data source** for our embedded data source connection, provide a **Name** for your **New** data source connection, **Type** of data source should be Teradata for the .NET Data Provider for Teradata. Click **Edit** (Figure 16) to provide the Teradata **Server name**, **User name** and **Password**.

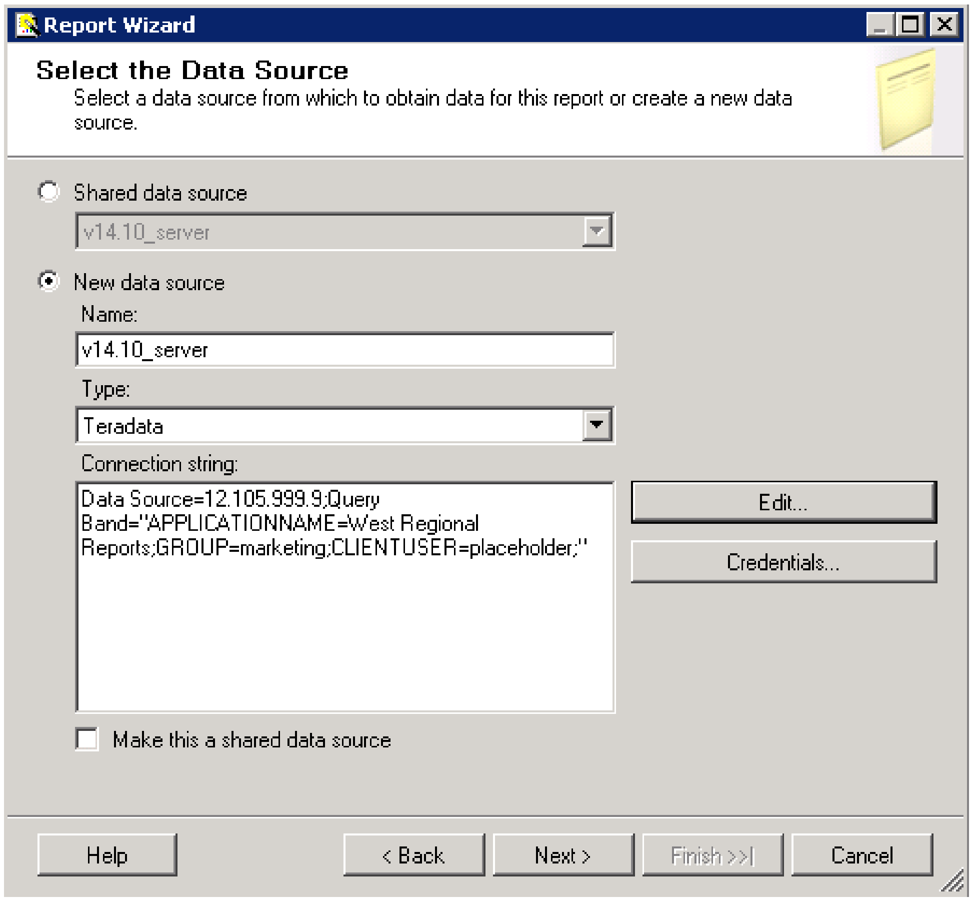


Data source: Teradata (.NET Framework Data Provider for Te
Server name: 12.105.999.9
Login Details
☐ Windows Authentication
User name: microsoft
Password: ●●●●●●●●●●
☐ Save my password
Mechanism: Default
Domain:

Figure 16: Teradata connection properties

It is also suggested to click on **Advanced** dialog to enter any Query Band 'static' and/or placeholder values you plan to use and/or derive from the expression based connection string feature (see Figure 12) above. For example, Query Band attribute GROUP='marketing' might be a 'static' value, but CLIENTUSER= or PROXYUSER='placeholder' might be a run time derived value using a built-in field (i.e. *UserID*) with Expression based Connection String editor. For Teradata Trusted Sessions sites using domain users to identify and authorize access to Teradata see *Hint* below to derive user name from *UserID* built-in field to set PROXYUSER. This will be clearer in step 7 below.

Finally, click **OK** and then click **Next** as shown in Figure 17 and continue to step 6.


Report Wizard

Select the Data Source

Select a data source from which to obtain data for this report or create a new data source.

Shared data source: v14.10_server

New data source [✓]

Name: v14.10_server

Type: Teradata

Connection string:

```
Data Source=12.105.999.9;Query
Band="APPLICATIONNAME=West Regional
Reports;GROUP=marketing;CLIENTUSER=placeholder;"
```

Make this a shared data source [ ]

Figure 17: Select the Data Source with ‘static’ connection string

4. Use steps 4 and 5 to change existing **Report1** data source from shared to an embedded data source. Expand **Data Sources** under **Report Data** pane (Figure 18) and click on **Data Source Properties** to change earlier **Shared** data source connection to **Embedded**.

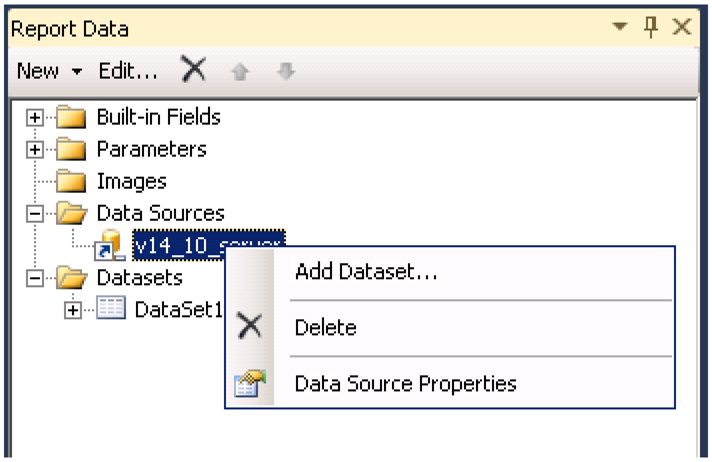
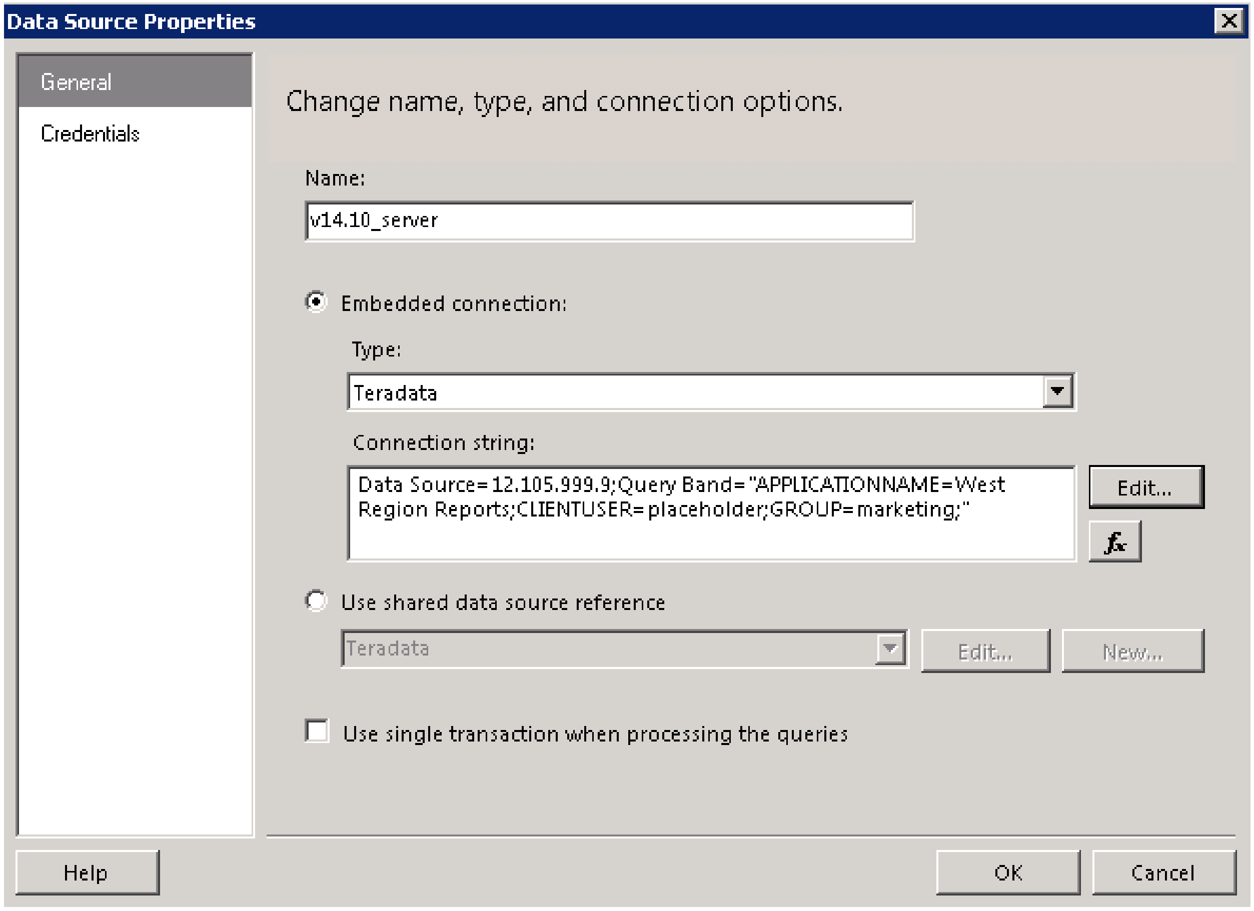


Figure 18: Data Sources under Report Data

5. Choose **Embedded** connection, Teradata for **Type** and **Edit** to re-establish connection to Teradata. Reference steps 3 and Figure 16 above to re-establish connection and Query Band statement to display Figure 19 and click **OK**


Data Source Properties

Change name, type, and connection options.

Name: v14.10_server

(●) Embedded connection:

Type: Teradata

Connection string:

```
Data Source=12.105.999.9;Query Band="APPLICATIONNAME=West Region Reports;CLIENTUSER=placeholder;GROUP=marketing;"
```

( ) Use shared data source reference: Teradata

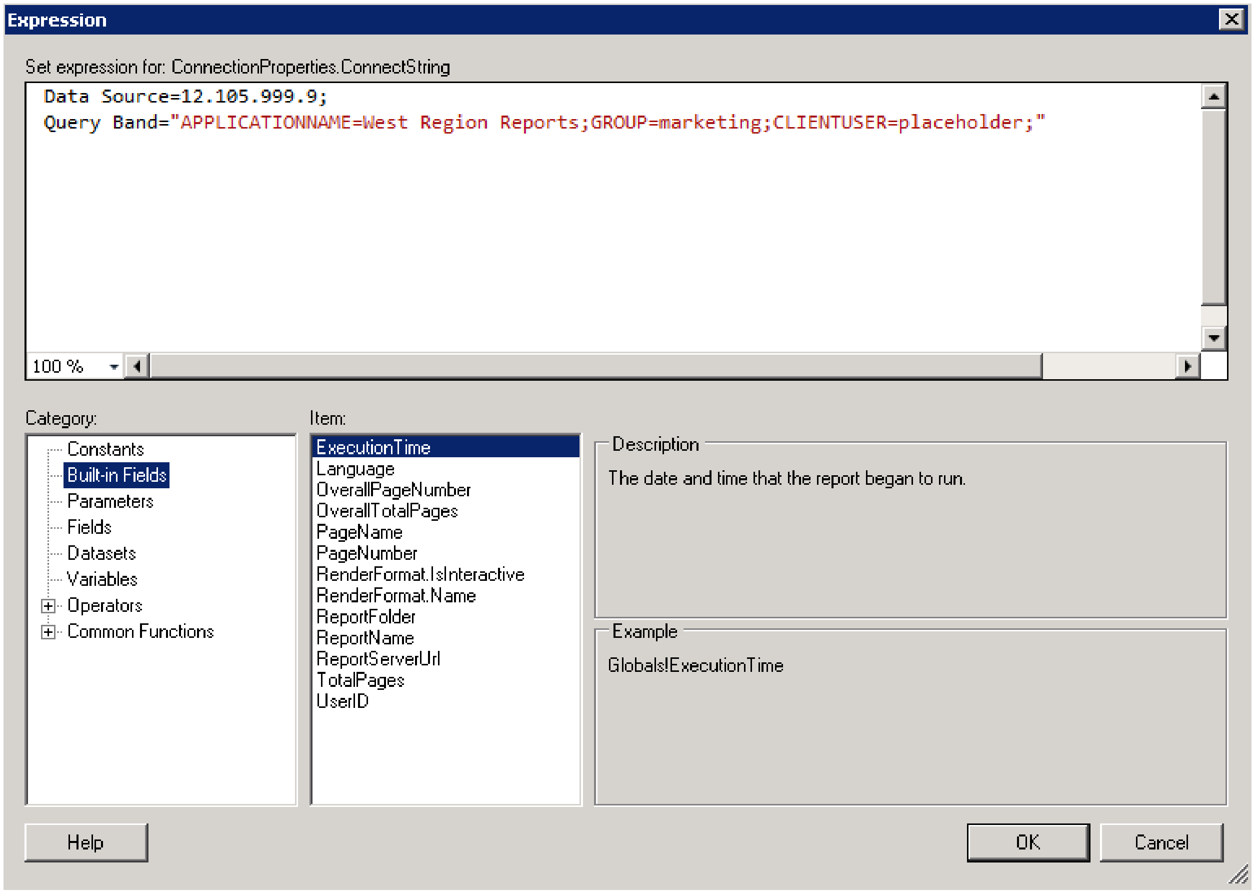
☐ Use single transaction when processing the queries

Figure 19: Data Source Properties

6. On the **Design the Query** page (Figure 6), enter your query and click **Finish** and test you 'static' embedded connection string and query by clicking on **Preview** tab before proceeding to the next step.

7. In the next few steps, we will change your embedded 'static' connection string to an expression based connection string and replace any Query Band 'placeholder' values (from step 3) with built-in field parameters for your Query Band statement for this report. Though, report parameters can also be used with an expression, this example does not warrant its use.

8. To open the Expression editor (Figure 20), click your **Data Source Properties** (Figure 19 above) and click on ***fx*** button under **Edit** button.



```
Data Source=12.105.999.9;
Query Band="APPLICATIONNAME=West Region Reports;GROUP=marketing;CLIENTUSER=placeholder;"
```

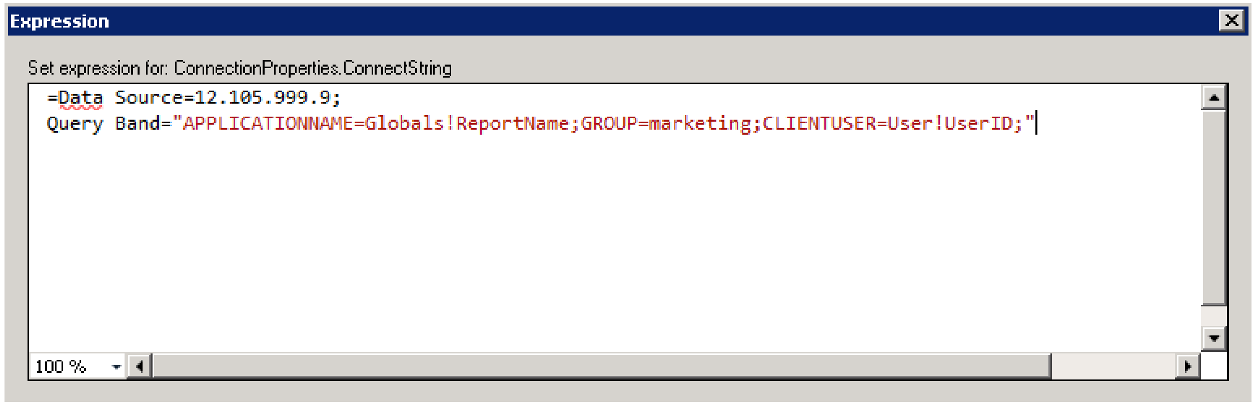
The date and time that the report began to run.

Globals!ExecutionTime

Figure 20: Expression Editor

9. In this example, we set Query Band attributes CLIENTUSER and APPLICATIONNAME with ‘static’ or placeholder values. To provide specific information about the user and actual report name, under **Category** using **Built-in Fields *UserID*** and ***ReportName*** will provide just such information.

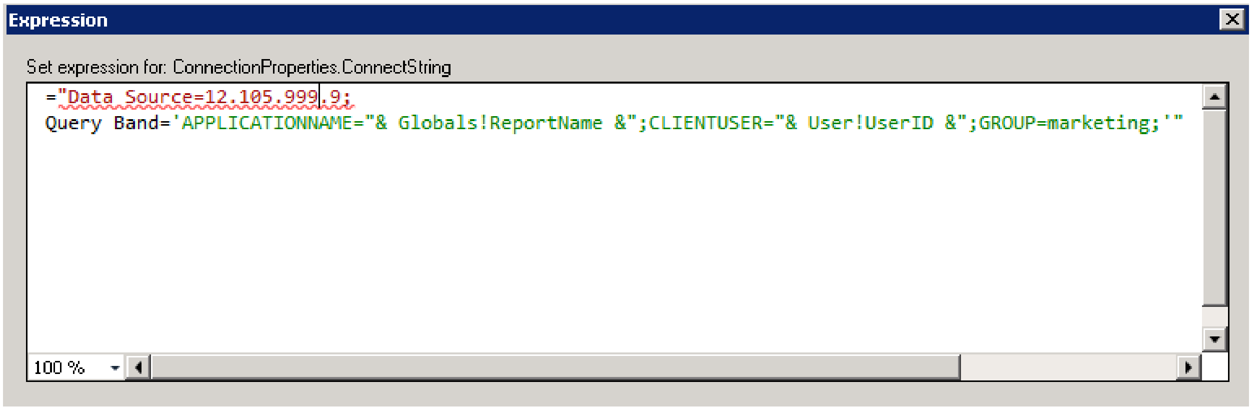
10. Before double-clicking a built-in field, insert ‘=’ before the expression, otherwise your whole connection string will disappear. Then highlight your ‘placeholder’ and ‘static’ values and double-click built-in fields UserID and then ReportName items as shown in Figure 21.



```
=Data Source=12.105.999.9;
Query Band="APPLICATIONNAME=Globals!ReportName;GROUP=marketing;CLIENTUSER=User!UserID;"
```

Figure 21: Expression based Connection String after Built-in Field replacement

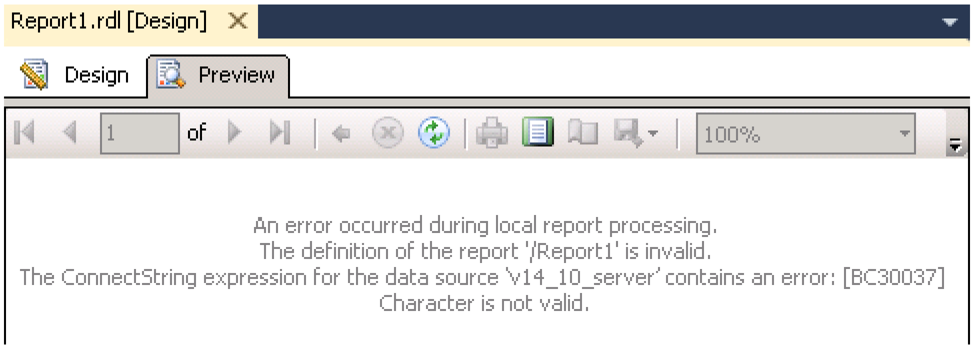
11. Next, modify expression syntax for correctness as shown in Figure 22, pay particular attention to:
    - Single quotes: Replaces initial double quotes around Query Band statement
    - Double quotes: Used between strings and 'static' values
    - '&' placement: Used between Built-in Fields



```
="Data Source=12.105.999.9;
Query Band='APPLICATIONNAME="& Globals!ReportName &";CLIENTUSER="& User!UserID &";GROUP=marketing;'"
```

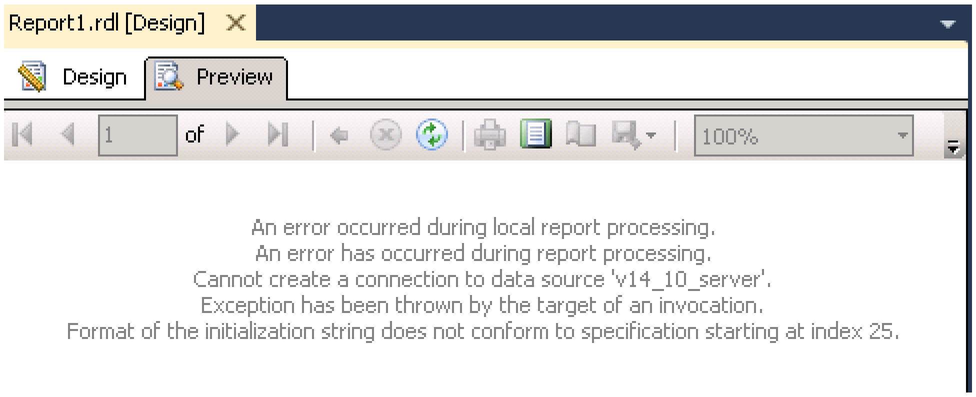
Figure 22: Expression based Connection String using built-in fields for Query Band

12. Click **OK** twice and test report with new expression based connection string using **Preview** tab. The following as shown in Figure 23 and 24 are typical errors due to misplacement of single and/or double quotes in your expression.



An error occurred during local report processing.
The definition of the report '/Report1' is invalid.
The ConnectString expression for the data source 'V14_10_server' contains an error: [BC30037] Character is not valid.

Figure 23: Error due to misplaced double quote



```
An error occurred during local report processing.
An error has occurred during report processing.
Cannot create a connection to data source 'v14_10_server'.
Exception has been thrown by the target of an invocation.
Format of the initialization string does not conform to specification starting at index 25.
```

Figure 24: Error due to misplaced single quote

13. Successful run of your report will generate the Query Band statement with the report query to Teradata Database as captured in Teradata DBQL logs.

    *SET QUERY_BAND = 'APPLICATIONNAME=**Report1**;CLIENTUSER=**TD\rs122158**;GROUP=**marketing**;QueryIssueTime=2013-06-11T15:08:27.206075Z;' FOR SESSION*

    select * from realtd.vtbl_dim_store

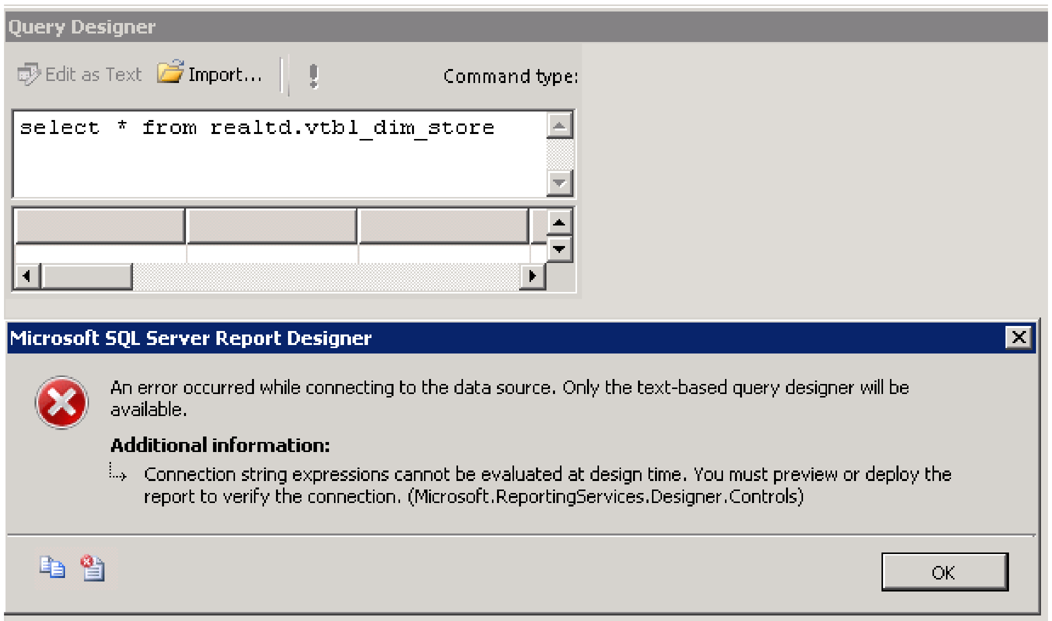
    SET QUERY_BAND = NONE FOR SESSION;

    Hint: For setting up Query Bands with Teradata Trusted Sessions, use the Built-in field UserID with the following code to pass the current SSRS user logged on (minus the domain) to Query Band attribute PROXYUSER. To remove the domain in the UserID field use the following code:

    "& MID(User!UserID,InStr(User!UserID,"\")+1, Len(User!UserID)) &";

14. Updates to a report after deployment and/or during report design after establishing an Expression based Connection String may require converting between ECS and ‘static’ (i.e. Data Source Properties Edit button) connection during testing.

    Note, Query Designer does not support reports with an Expression based Connection Strings as shown in Figure 25. You must convert back to a ‘static’ connection string to use Query Designer.



```
select * from realtd.vtbl_dim_store
```

An error occurred while connecting to the data source. Only the text-based query designer will be available.

Additional information:

Connection string expressions cannot be evaluated at design time. You must preview or deploy the report to verify the connection. (Microsoft.ReportingServices.Designer.Controls)

Figure 25: Query Designer error message with Expression based Connection String

As the above examples have demonstrated, Reporting Services has some 'limited' out-of-the-box capabilities with Teradata Query Band feature to provide better governance and workload management of applications running against Teradata database.

Some of the benefits are,

- Works for Reporting Services and Report Builder
- Allows dynamic and static parameter provision to Teradata
- Can configure Teradata Trusted Sessions using Query Band *key/value* pair Proxyuser and Proxyrole.
- Teradata workload management software can pick up Query Band statements for meeting agreed upon SLA requirements.

Some implementation considerations to be aware of are,

- No global parameter setting for pre and post based processing.
  - Requires creating an Expression for each report with an embedded connection string.
- Prevents metadata retrieval of columns in data set until run time.
  - Users must first create static connection and replace with expression
- Report Builder Data Source Properties Connection String Edit/Build button is greyed out.
  - Users will need to manually create or cut and paste 'static' connection string definition for Teradata.
- Updates to reports will require converting between Expression based and 'static connections string definition.
- Query Designer does not work with Expressions based Connections String

Working with Teradata Temporal Feature

Though, no ‘official’ integration or support exists between Microsoft Reporting Services and Teradata Temporal Table feature. There is some ‘rudimentary’ capabilities ‘out-of-the-box’ to enable Reporting Services to design reports against Teradata Temporal tables where applications need to design and build databases where information changes over time.

Teradata Temporal Table feature was introduced in Teradata Database v13.10 to support temporal ‘time-based’ analytics. Teradata Database provides the built-in capabilities that are required in a temporal database management system. Temporal data types and temporal statements facilitate creating applications that need to represent time and the information that changes over time. The ‘major’ components for this feature are:

- **Temporal Data Types** – The PERIOD data type represents an anchored duration of time.

- **Kinds of time** – Teradata temporal table support adds the capability to add VALIDTIME and TRANSACTIONTIME column attributes to time dimension tables.

- **Temporal statements** – Temporal SQL modifiers for existing statements that let you create and alter temporal tables, query and modify data that changes over time.

Period Data Type
Teradata provides temporal table support at the data type level with period data types. A period is an anchored duration that represents a set of contiguous time granules within the duration. It has a beginning bound (defined by the value of a beginning element) and an ending bound (defined by the value of an ending element). Beginning and ending elements can be DATE, TIME, or TIMESTAMP types, but both must be the same type.

Temporal Table
Temporal tables store and maintain information with respect to time. Temporal tables include one or two special columns, which store time information:

- **TRANSACTIONTIME** column records and maintains the time period for which Teradata Database was aware of the information in the row. Teradata Database automatically enters and maintains the transaction-time column data, and consequently automatically tracks the history of such information.

- **VALIDTIME** column models the real world, and stores information such as the time an insurance policy or product warranty is valid, the length of employment of an employee, or other information that is important to track and manipulate in a time-aware fashion. When you add a new row to this type of table, you use the valid-time column to specify the time period for which the row information is valid. This is the period of validity (PV) of the information in the row.

As rows are changed in temporal tables, the database automatically creates new rows as necessary to maintain the time dimensions. UNITL_CLOSED and UNTIL_CHANGED are used to represent the end bound values of the newly created rows for TRANSACTIONTIME and VALIDTIME period values.

A Teradata table is considered non-temporal if VALIDTIME and TRANSACTIONTIME columns attributes are not defined with PERIOD data type column. Besides, being able to mix and match temporal and non-temporal columns, you can also define both VALIDTIME and TRANSACTIONTIME on a table, which is then called a BI-Temporal table.

Temporal Statements
Queries and modifications can include temporal qualifiers that reference a time dimension and act as criteria or selectors on the data. They affect only the data that meets the time criterion. Temporal DML statements can be generally qualified as:

- CURRENT, affecting only data that is currently in effect
- SEQUENCED, affecting only data that is in effect for a specified time period
- AS OF, affecting only data that is in effect at a specified point in time
- NONSEQUENCED, where the time dimension is ignored, the table is treated as a non-temporal table, and DML is applied to all data in the table

For more information, please reference *Teradata Temporal Support* documentation here.

Interoperability
At a 'high-level', Reporting Services 'out-of-the-box' does recognize PERIOD data types in queries with

- **Non-Temporal –** Tables which does not use TRANSACTION and VALIDTIME column attributes with the PERIOD data type column
- **Temporal** – Tables which use the TRANSACTION and VALIDTIME column attributes with the PERIOD data type column

However, workarounds are required when writing Temporal SQL statements with parameterized queries using unnamed parameters (i.e. ?) for DATE values with Reporting Services Report Wizard. Initial reports need to be designed with a 'hard-coded' value and then modified to a parameterized query using Query Designer with a Report Parameter. In addition, Query Expression Editor ***fx*** will be required if DATE value from Calendar control UI is not an option and requires a 'manual' DATE to be enter by user.

There is no integration to help design temporal queries with Temporal SQL Qualifiers to the extent of what you can enter in the Reporting Services Report Wizard and Query Designer **'Text'** *only* editor for Teradata.

Note, it is assume readers of this section are familiar with Teradata Temporal Table feature and Reporting Services features.

Teradata Database
The examples below will reference a company's payroll where if you know what Event (i.e. when someone got Hired, Fired or moved to another organization) as shown in Figure 26 and the DATE you're interested, you will be able to send a request to the database such as

"Show me the payroll report after some Event has happened in our company"

Tables and data used in this section can be found in *Appendix B – Teradata Database.* For simplicity, examples below use only VALIDTIME, but a large part of temporal databases require BI-Temporal implementation.

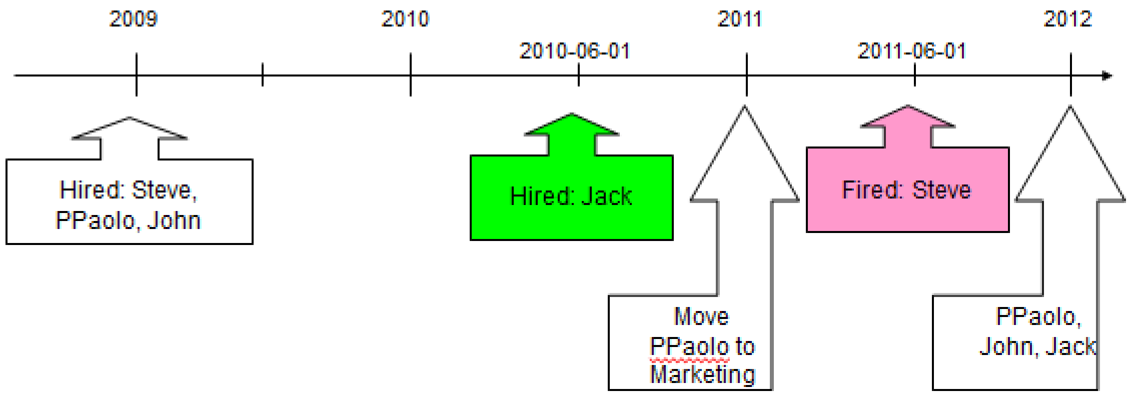


Figure 26: Temporal Events

Create a Teradata Temporal query

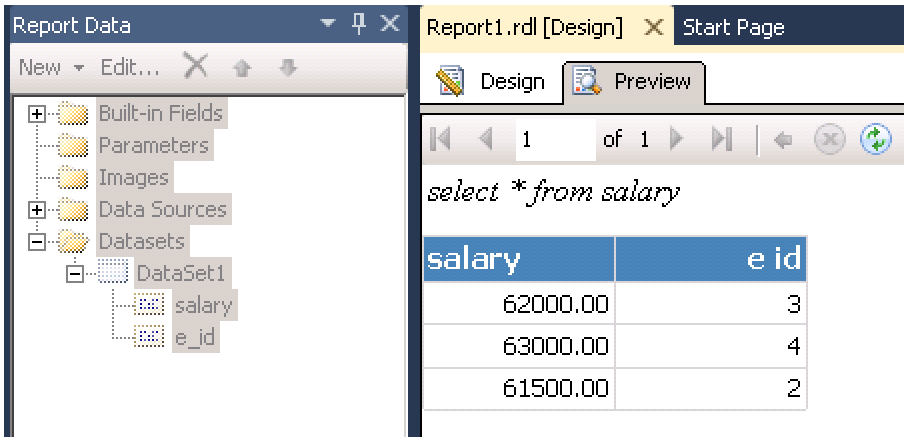
The following are a few 'simple' examples of creating a Teradata Temporal query.

1. In Solution Explorer, right-click on the **Reports** node and then select **Add New Report** as shown in Figure 4 above. Chose or create a shared or New embedded Data Source and click **Next**

2. On the **Design the Query** page (Figure 6), enter the Temporal query example below. To save report click **Next** and **Finish**. Click **Preview** tab to run the report. To repeat for each example below, double click **DataSet1** object to display **Dataset Properties** to enter/modify a new query.

   Example 1
   A normal select will equate to current validtime and will display the 'current' salary values for an employee as show in Figure 27. This is similar to *"current validtime select * from salary"* and would give the same result.

   *select * from salary*



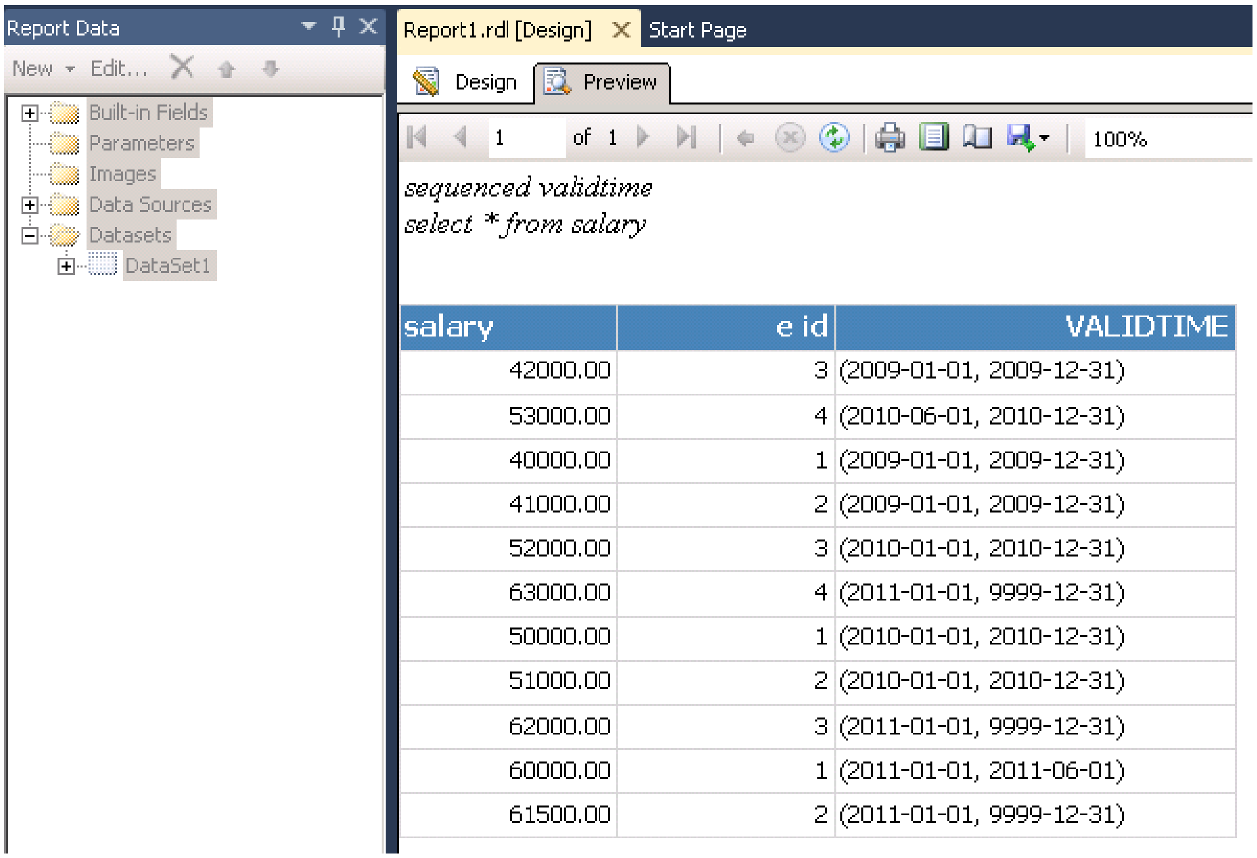
*select * from salary*

salary	e id
62000.00	3
63000.00	4
61500.00	2

Figure 27: Current employee salary

Example 2
To show the validtime column and show all salary data over time for an employee as shown in Figure 28, add the temporal SQL qualifier sequenced validtime and add the column VAILDTIME to the report.

SEQUENCED VALIDTIME
*select * from salary*



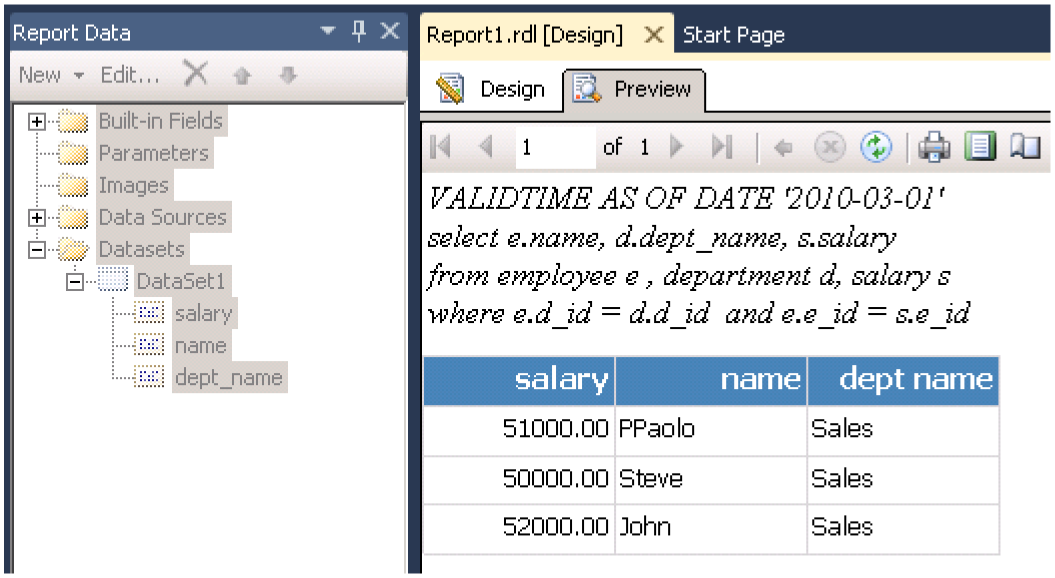
sequenced validtime
*select * from salary*

salary	e id	VALIDTIME
42000.00	3	(2009-01-01, 2009-12-31)
53000.00	4	(2010-06-01, 2010-12-31)
40000.00	1	(2009-01-01, 2009-12-31)
41000.00	2	(2009-01-01, 2009-12-31)
52000.00	3	(2010-01-01, 2010-12-31)
63000.00	4	(2011-01-01, 9999-12-31)
50000.00	1	(2010-01-01, 2010-12-31)
51000.00	2	(2010-01-01, 2010-12-31)
62000.00	3	(2011-01-01, 9999-12-31)
60000.00	1	(2011-01-01, 2011-06-01)
61500.00	2	(2011-01-01, 9999-12-31)

Figure 28: Shows validtime along with all salaries over time

Example 3
Finally as shown in Figure 29, to view all current employee's salary and departments they work for prior to Jack being hired '2010-06-01' (Figure 26) enter following SQL and modify report columns accordingly.

VALIDTIME AS OF DATE '2010-03-01'
select e.name, d.dept_name, s.salary
from employee e , department d, salary s
where e.d_id = d.d_id and e.e_id = s.e_id

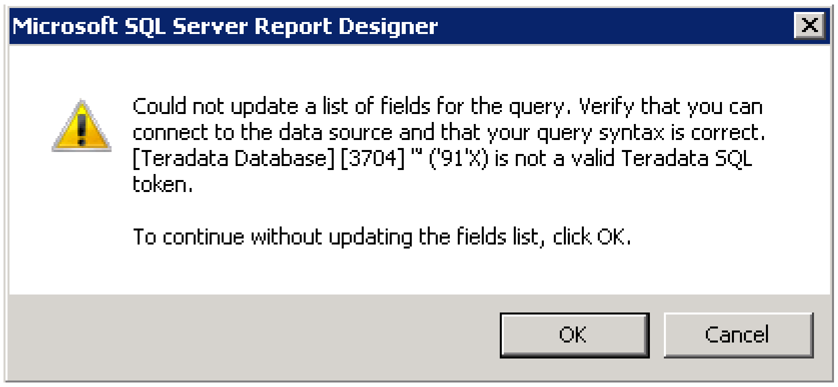


```
VALIDTIME AS OF DATE '2010-03-01'
select e.name, d.dept_name, s.salary
from employee e , department d, salary s
where e.d_id = d.d_id  and e.e_id = s.e_id
```

salary	name	dept name
51000.00	PPaolo	Sales
50000.00	Steve	Sales
52000.00	John	Sales

Figure 29: Employee salary and departments before Jack was hired

Tip: Check for appropriate single quote usage for literal values after any cut and paste task. Otherwise, you may experience following warning messages as shown in Figure 30 and 31.

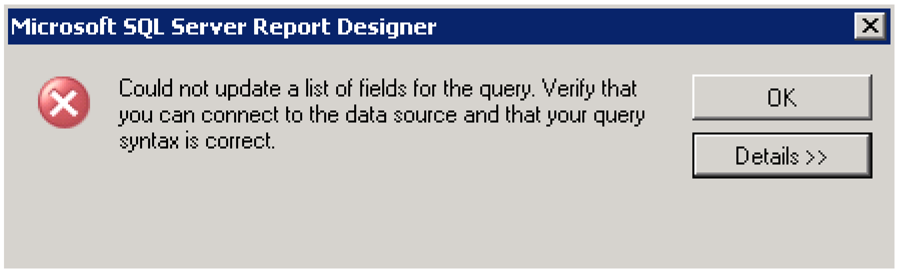


Microsoft SQL Server Report Designer

Could not update a list of fields for the query. Verify that you can connect to the data source and that your query syntax is correct.
[Teradata Database] [3704] '' ('91'X) is not a valid Teradata SQL token.

To continue without updating the fields list, click OK.

Figure 30: After enter SQL and clicking OK

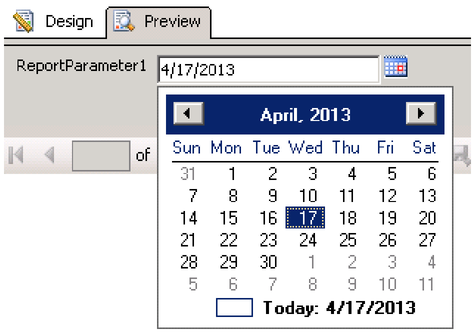

Microsoft SQL Server Report Designer

Could not update a list of fields for the query. Verify that you can connect to the data source and that your query syntax is correct.

Figure 31: After clicking on Refresh Fields

Create a Teradata Temporal 'parameterized' query

The following steps are what a typical user would need to perform for Temporal parameterized statements using Reporting Services Report Wizard and Calendar control UI to prompt a user for a DATE value. High-level steps include first creating Temporal parameterized statement with a hard-coded value, similar to Example 3 above. Then replacing hard-coded value with unnamed parameter (i.e. ?) and then create a Report Parameter to prompt user for a DATE value using the Calendar control UI as shown in Figure 32.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Figure 32: Calendar control UI for DATE value

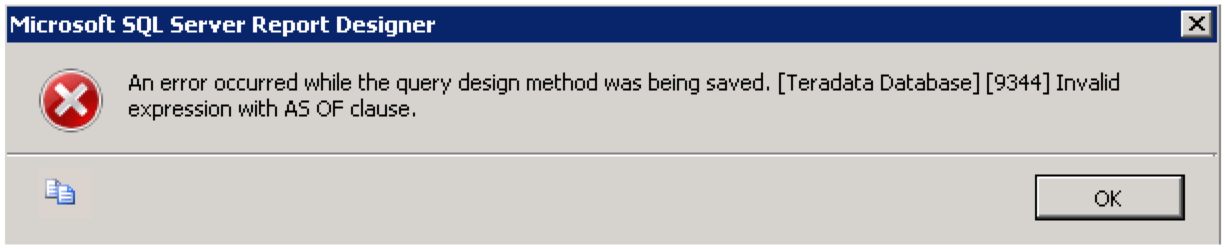
1. Entering the following Temporal parameterized statement during Report Wizard (Figure 6) will display an error message as shown in Figure 33 and Report Wizard will not continue.

   VALIDTIME AS OF ?

   select e.name, d.dept_name, s.salary

   from employee e , department d, salary s

   where e.d_id = d.d_id and e.e_id = s.e_id

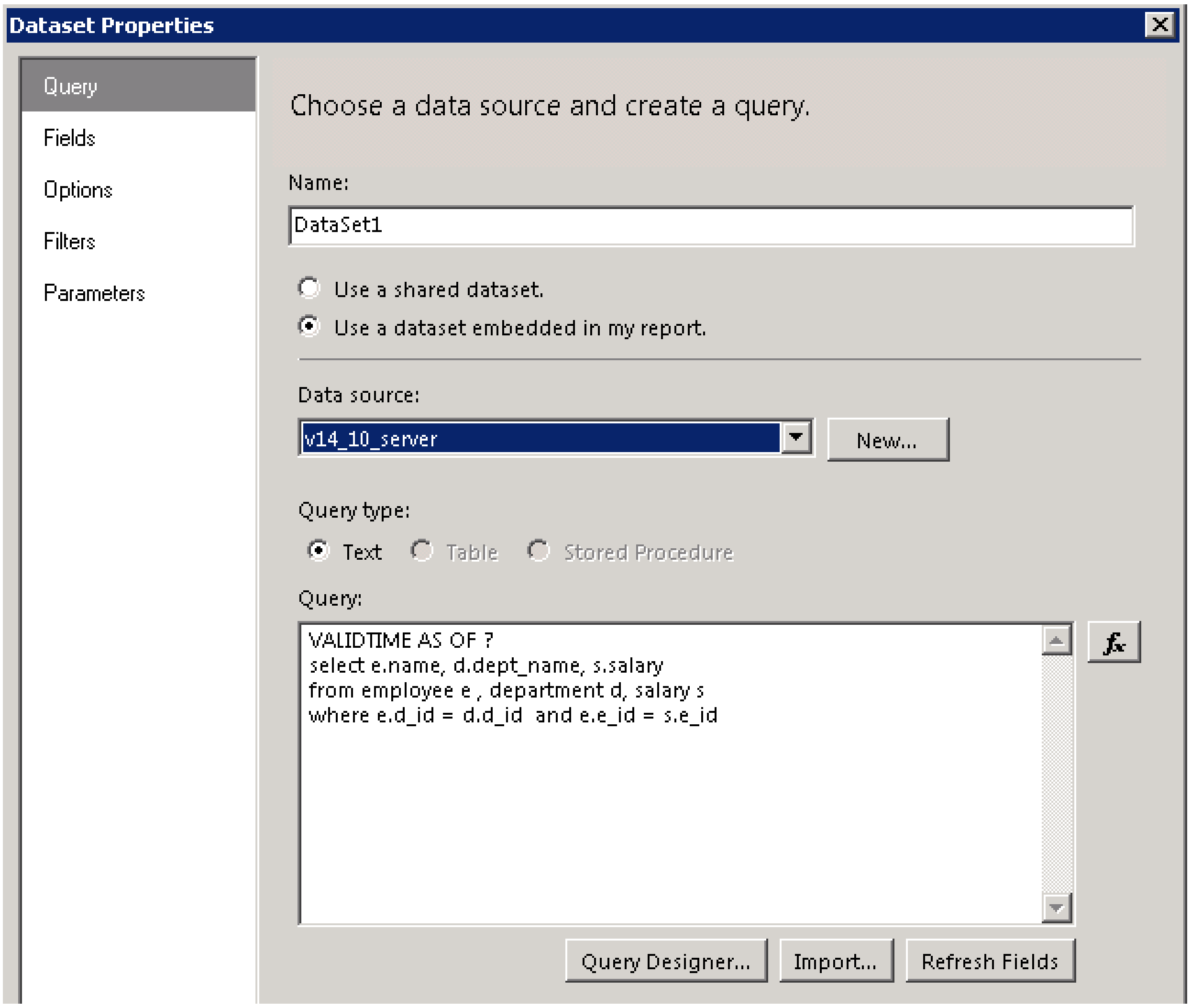

Microsoft SQL Server Report Designer

An error occurred while the query design method was being saved. [Teradata Database] [9344] Invalid expression with AS OF clause.

Figure 33: Warning message for Temporal parameterized statement

2. Instead, 'hard-code' the temporal statement with the DATE value as shown in Example 3 above and save report.

3. Then double click **DataSet1** object to display **Dataset Properties** to modify query for unnamed parameter '?' as shown in Figure 34.


Dataset Properties

Choose a data source and create a query.

Name: DataSet1

○ Use a shared dataset.
● Use a dataset embedded in my report.

Data source: v14_10_server

Query type: ● Text ○ Table ○ Stored Procedure

Query:

```
VALIDTIME AS OF ?
select e.name, d.dept_name, s.salary
from employee e , department d, salary s
where e.d_id = d.d_id  and e.e_id = s.e_id
```

Figure 34: Dataset Properties Query dialog

Note, Temporal statement does not need DATE syntax when using Reporting Services Calendar control UI (Figure 32)

4. Click **OK** and ignore **Define Query Parameters** message as shown in Figure 35 and click **OK twice do not** CANCEL. Otherwise, **DataSet** columns in your report will not appear under **Report Data** and make report usable and received error message as shown in Figure 35-1 when you try to run the report.

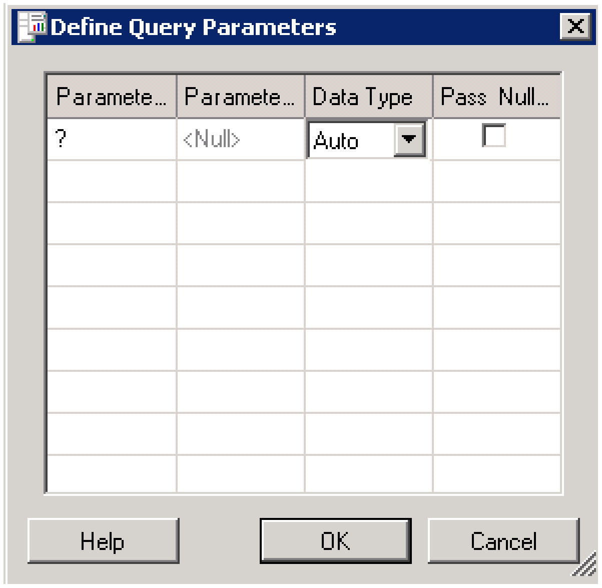
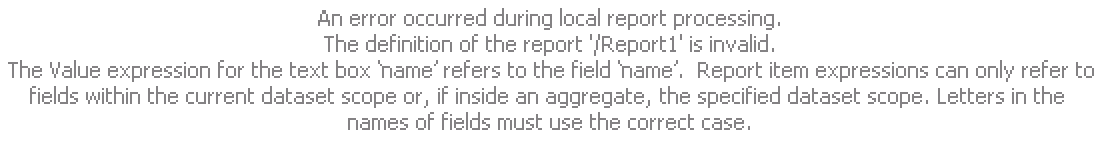


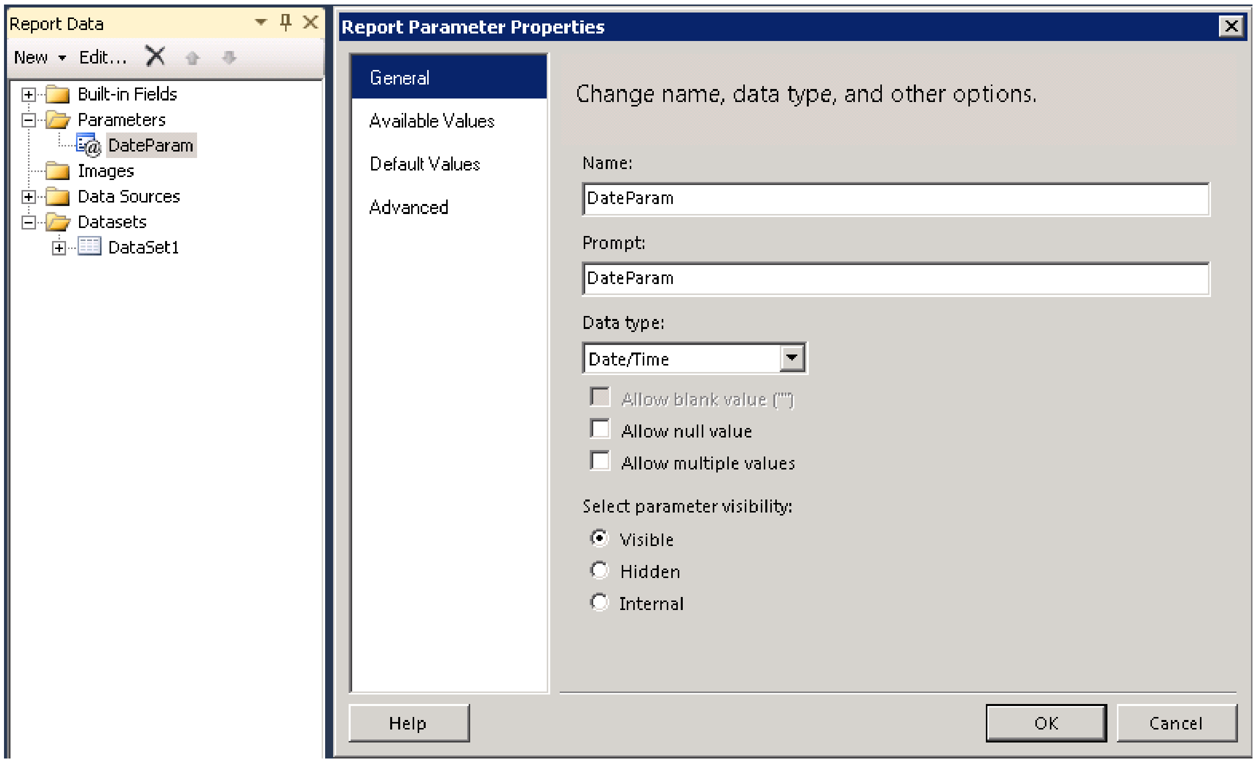
Figure 35: Define Query Parameter dialog



An error occurred during local report processing.
The definition of the report '/Report1' is invalid.
The Value expression for the text box 'name' refers to the field 'name'. Report item expressions can only refer to fields within the current dataset scope or, if inside an aggregate, the specified dataset scope. Letters in the names of fields must use the correct case.

Figure 35-1: Error message if Cancel is chosen

5. Next create the Report Parameter, click on **Design** tab, and right click **Parameters** in **Report Data** pane and **Add Parameter**. Change ReportParameter1 **Name** and **Prompt** to something more meaningful like *DateParam* and set **Data type** to ‘*Datetime’* as shown in Figure 36 and click **OK**.



Change name, data type, and other options.

Name: DateParam

Prompt: DateParam

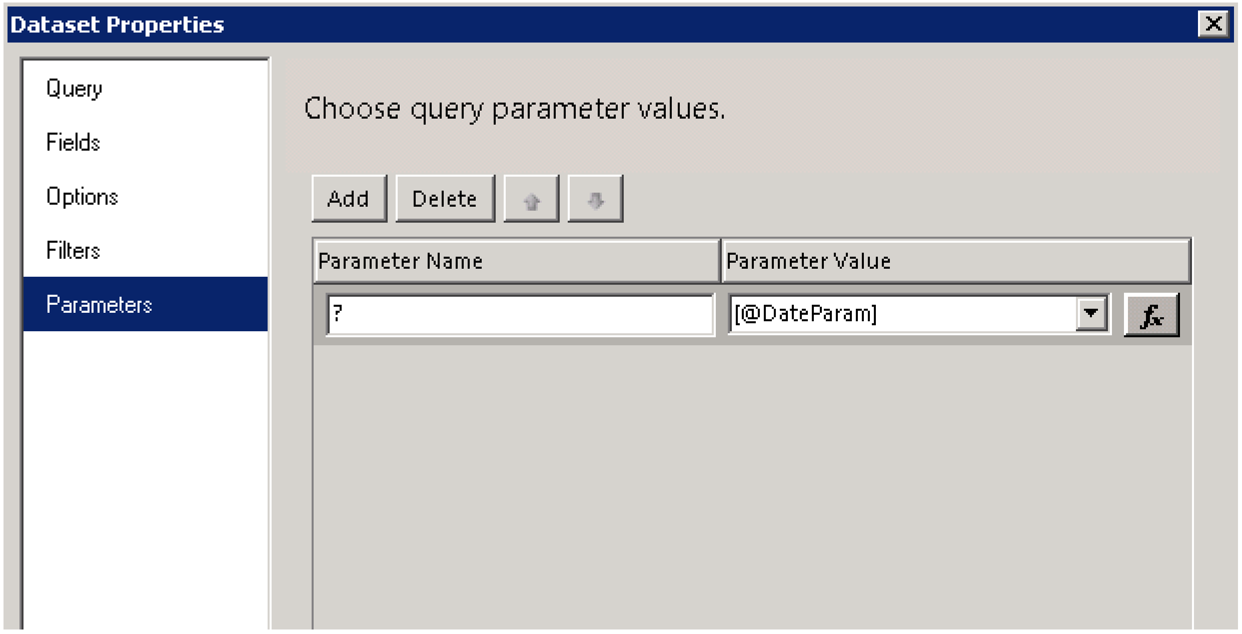
Data type: Date/Time

[ ] Allow blank value ("")
[ ] Allow null value
[ ] Allow multiple values

Select parameter visibility:
[✓ Visible]
Hidden
Internal

Figure 36: Report Parameter Properties

6. **Add** the Report Parameter (i.e. @DateParam) to the **Dataset Properties** page for **Parameters** as shown in Figure 37. Parameter Name is the unnamed parameter ? and Parameter Value is @DateParam you just created in the previous step. Click **OK**



Choose query parameter values.

Parameter Name	Parameter Value
?	[@DateParam]

Figure 37: Adding a Report Parameter in Dataset Properties page

7. Click **Preview** tab to run report. Pick Date from Calendar prompt or type Date value (i.e. 2010-09-01 or 9/1/2010) as shown in Figure 38.

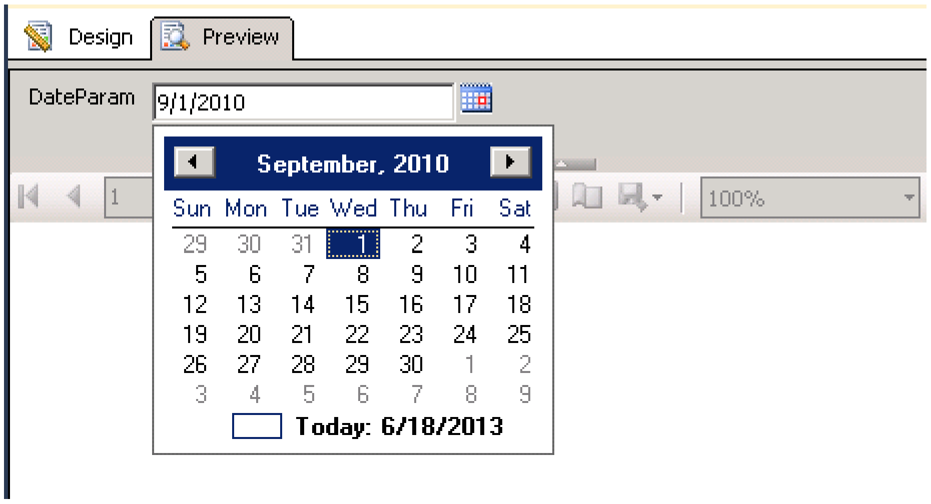
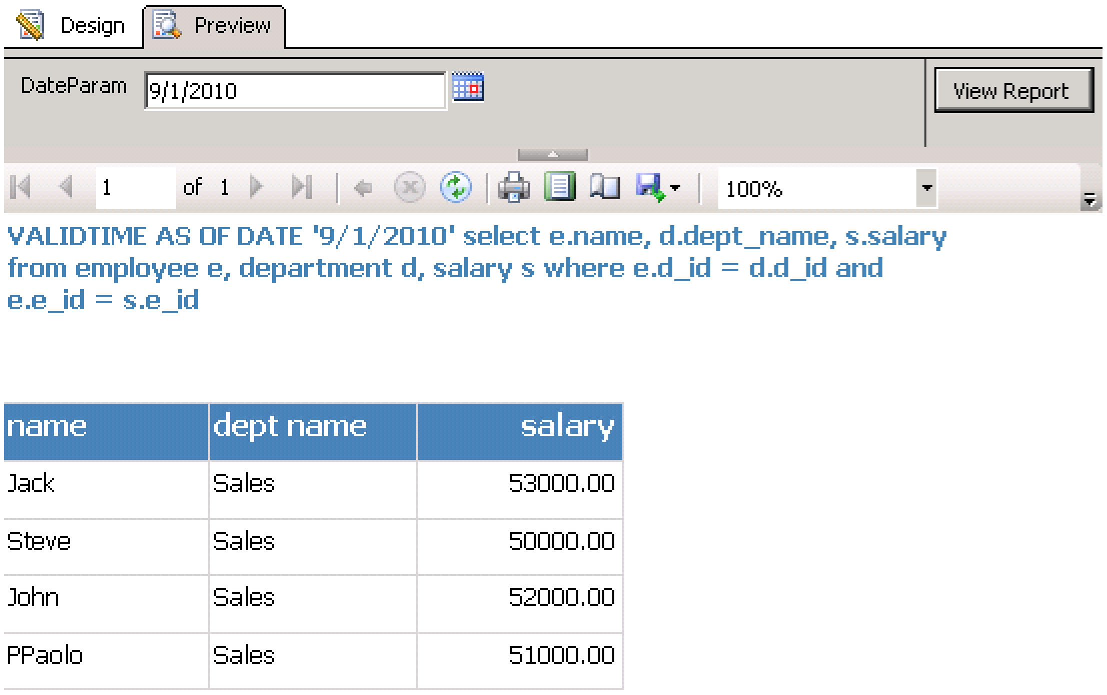


Figure 38: Preview tab and Calendar control UI

8. Click **View Report** to run report as shown in Figure 39.



VALIDTIME AS OF DATE '9/1/2010' select e.name, d.dept_name, s.salary from employee e, department d, salary s where e.d_id = d.d_id and e.e_id = s.e_id

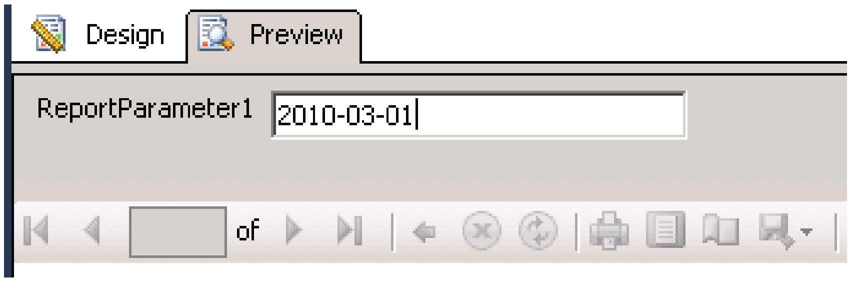
name	dept name	salary
Jack	Sales	53000.00
Steve	Sales	50000.00
John	Sales	52000.00
PPaolo	Sales	51000.00

Figure 39: After Jack was hired

Use Expression editor if Calendar Control UI is NOT an option

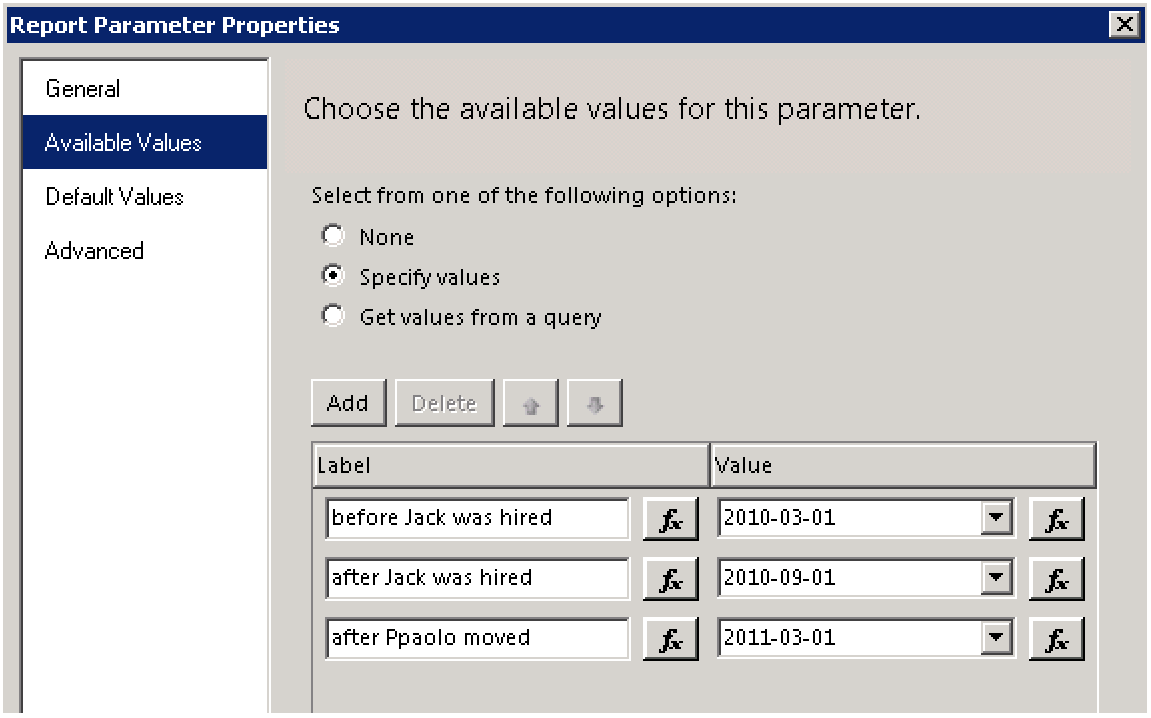
If Calendar control UI is not an option for your reporting requirements. For example, end users will be required to enter a DATE value as shown in Figure 40 which is also possible with Calendar control UI or be given specific drop down list of pre-defined or queried DATE values as shown in Figure 41. Then the following Teradata temporal statement would apply with the DATE syntax.

VALIDTIME AS OF ***DATE*** *?*
select e.name, d.dept_name, s.salary
from employee e , department d, salary s
where e.d_id = d.d_id and e.e_id = s.e_id.



ReportParameter1 2010-03-01

Figure 40: Manually enter DATE value



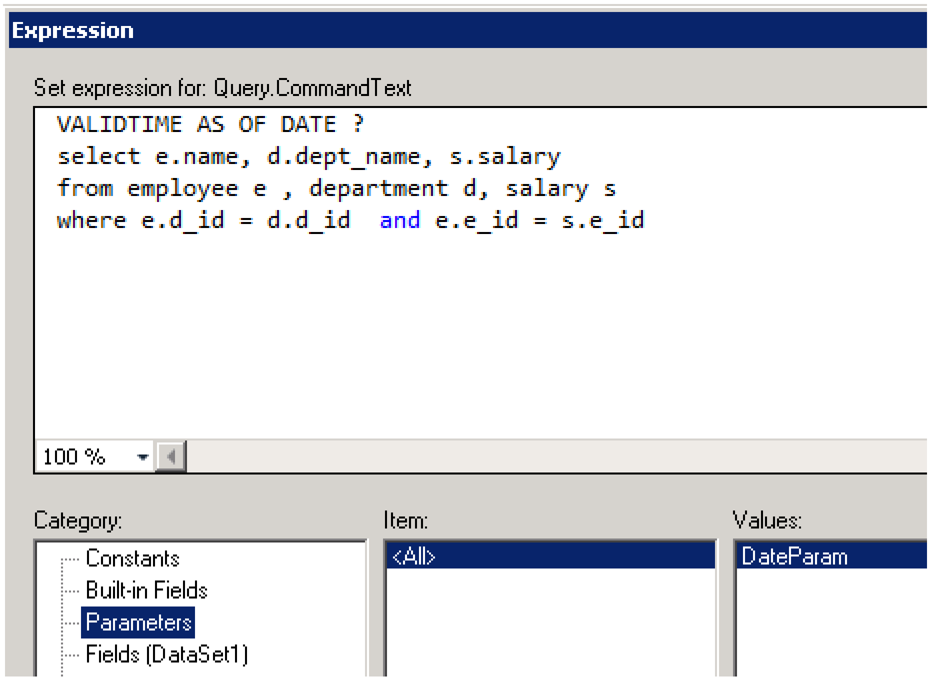
Choose the available values for this parameter.

Select from one of the following options:
- None
- [✓ Specify values]
- Get values from a query

Label	Value
before Jack was hired	2010-03-01
after Jack was hired	2010-09-01
after Ppaolo moved	2011-03-01

Figure 41: Report Parameter Properties predefine specific DATE values

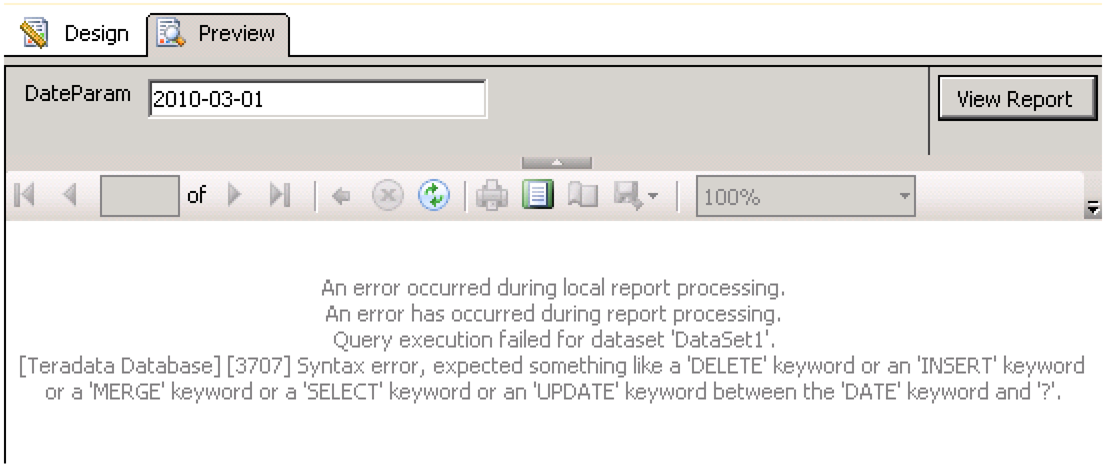
You would follow the same steps above, create the report with a hard-coded DATE value, edit the report with the unnamed parameter '?'. Create the Report Parameter with Date type **'Text'** instead of 'Date/Time', but then use Expression editor for Query as shown in Figure 42 by clicking on ***fx*** button (Figure 34) to replace ? parameter with a Report Parameter expression.



```
VALIDTIME AS OF DATE ?
select e.name, d.dept_name, s.salary
from employee e , department d, salary s
where e.d_id = d.d_id  and e.e_id = s.e_id
```

Figure 42: Expression editor for Query

Otherwise, you will receive the following error message.



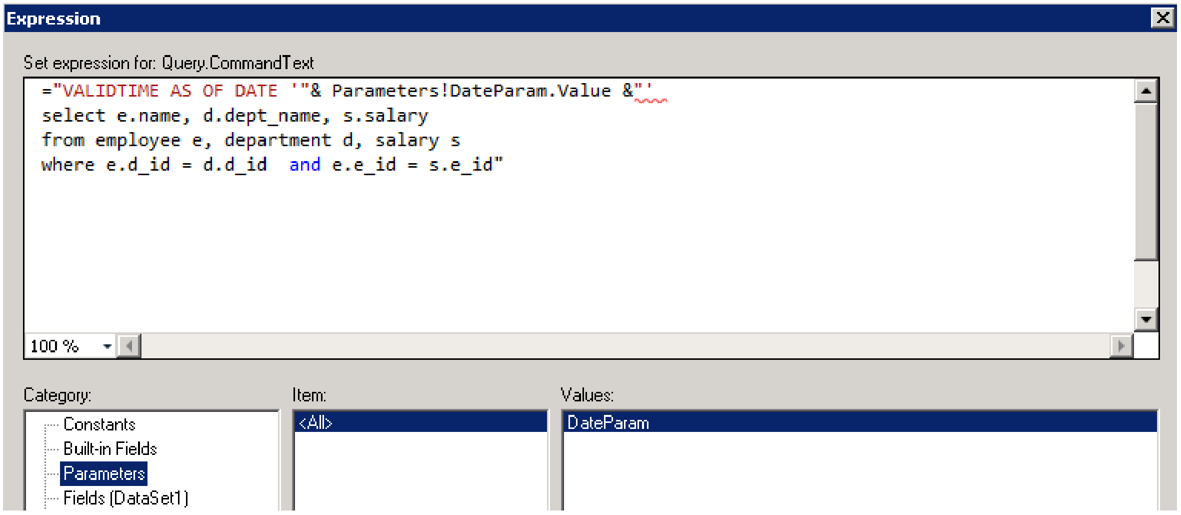
DateParam: 2010-03-01

```
An error occurred during local report processing.
An error has occurred during report processing.
Query execution failed for dataset 'DataSet1'.
[Teradata Database] [3707] Syntax error, expected something like a 'DELETE' keyword or an 'INSERT' keyword
or a 'MERGE' keyword or a 'SELECT' keyword or an 'UPDATE' keyword between the 'DATE' keyword and '?'.
```

Figure 43: Error message when running report with ?

The same rules apply when replacing and modifying your report with Report Parameters within the Expression editor see *Creating a 'dynamic' Query Band statement* steps 10 and 11 above.

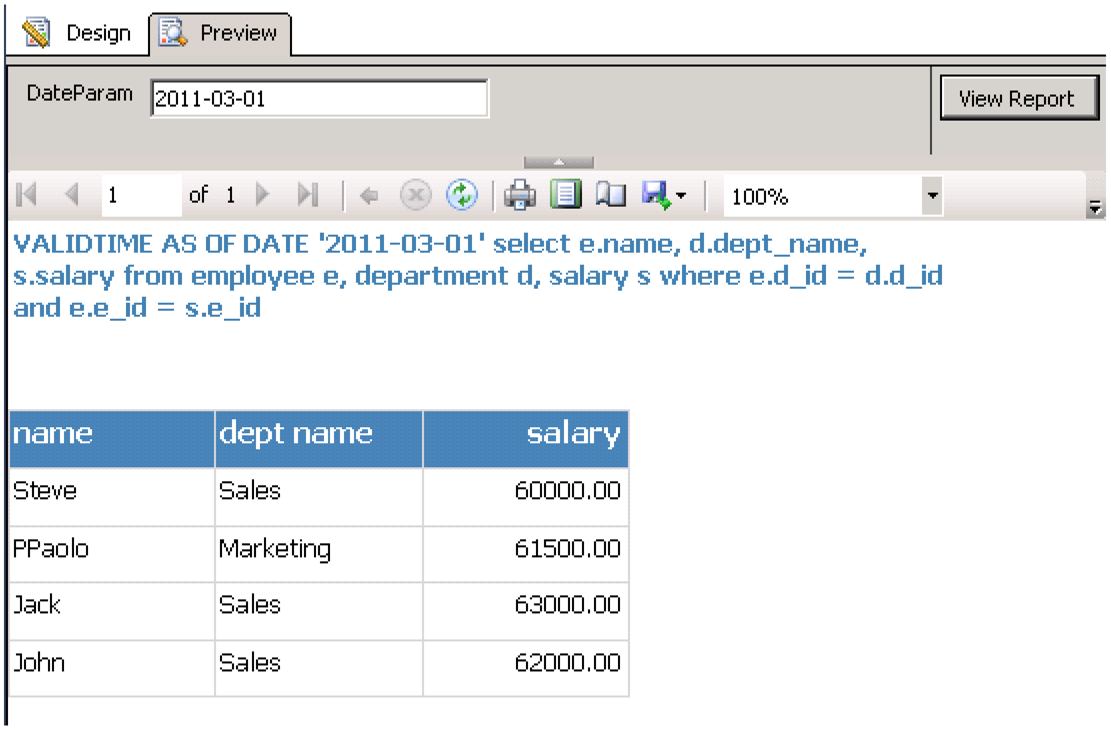
In this example, we will be replacing ? with DateParam report parameter as shown in Figure 44. Check your single and double quote placements.



```
="VALIDTIME AS OF DATE '"& Parameters!DateParam.Value &"'
select e.name, d.dept_name, s.salary
from employee e, department d, salary s
where e.d_id = d.d_id  and e.e_id = s.e_id"
```

Figure 44: Expression editor for Query replacing ? with report parameter

Finally, **Preview** the report as shown in Figure 45. If you encounter any errors, especially when converting Calendar control UI to an Expression based report parameter review steps 10 and 11 above and delete any Report Parameter in Dataset Properties page (Figure 37) you may have added for Calendar control UI designed reports which is not required when a report query is converted to an Expression.



VALIDTIME AS OF DATE '2011-03-01' select e.name, d.dept_name, s.salary from employee e, department d, salary s where e.d_id = d.d_id and e.e_id = s.e_id

name	dept name	salary
Steve	Sales	60000.00
PPaolo	Marketing	61500.00
Jack	Sales	63000.00
John	Sales	62000.00

Figure 45: After Ppaolo moved to Marketing

Hint: Ensure you test your reports in SSDT Designer using 'green' refresh arrows (Figure 46). Otherwise, you might experience errors with reports going against cached results versus against the database. Refresh ensures current report definitions are accurate based on source.

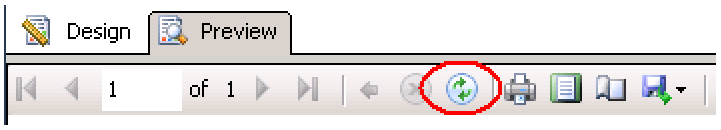


Figure 46: Refresh/rerun report button

Note, Reporting Services does not recognize 'set session' syntax. The following will not work.

set session *validtime as of date '2010-06-01'*
select b.projectname, a.date_id, sum(a.hours)
from fact_hours a, lu_employee_temporal b
where a.employee_id = b.employee_id
group by b.projectname, a.date_id

As the above examples have demonstrated, Reporting Services has some rudimentary 'out-of-the-box' capabilities with Teradata Temporal table feature for applications that might need to design report where information changes over time in a temporal based database.

Working with Teradata BLOB Data Type column

Reporting Services supports Teradata BLOB data type. The following is a summary of the steps that a typical user goes through to create a report to display an image stored in a Teradata BLOB data type column. Tables and data used in this section can be found in *Appendix B – Teradata Database*.

1. In Solution Explorer, right-click on the **Reports** node and then select **Add New Report** as shown in (Figure 4) above. Chose or create a Shared or New 'Embedded' Data Source and click Next

2. On the **Design the Query** page (Figure 6), enter the SQL query example below and click **Next** and **Finish** to save report.

   SELECT TestID, BLOBName, BLOBData FROM BLOBTest

3. Before you **Preview** the report, set the appropriate image properties to the BLOB data type column in your report.

4. In **Design** view, right click our BLOBData column as shown in Figure B-1 and click *Insert→Image*

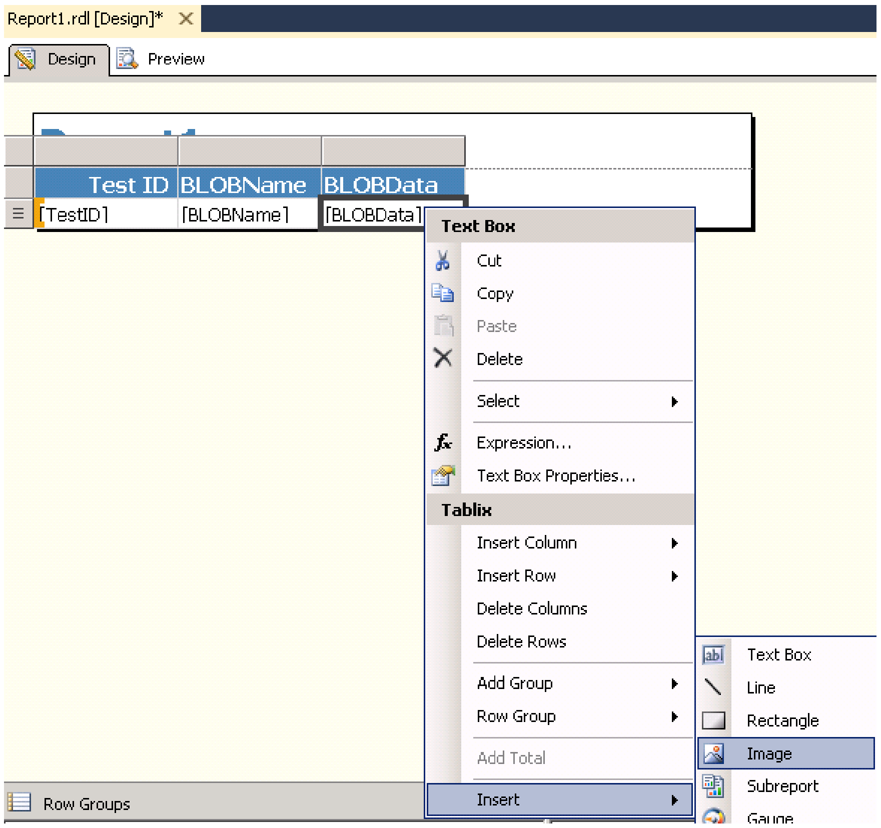


Figure B-1: Open Image Properties

5. In **General** page of the **Image Properties** as shown in Figure B-2, set **Select the image source** to *'Database'*, **Use this field** to *'BLOBData'* and **Use this MIME type** to *'image/jpeg'* and Click OK.

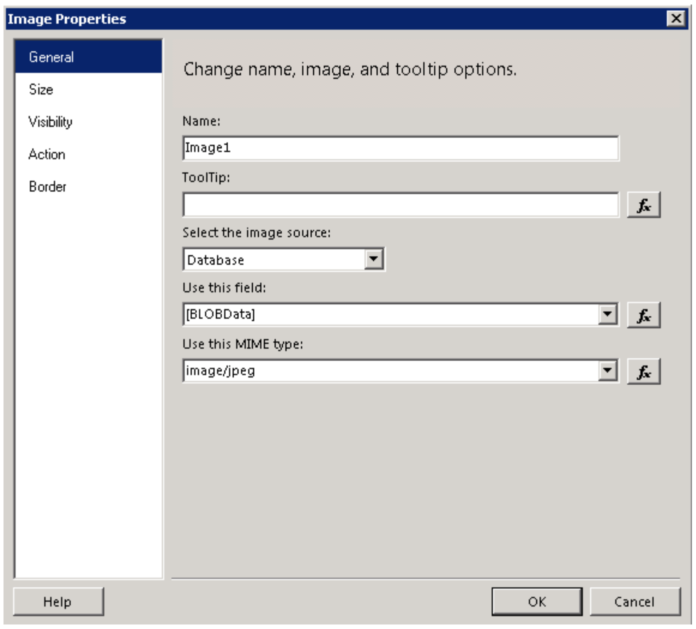


Image Properties

Change name, image, and tooltip options.

Name: Image1

ToolTip:

Select the image source: Database

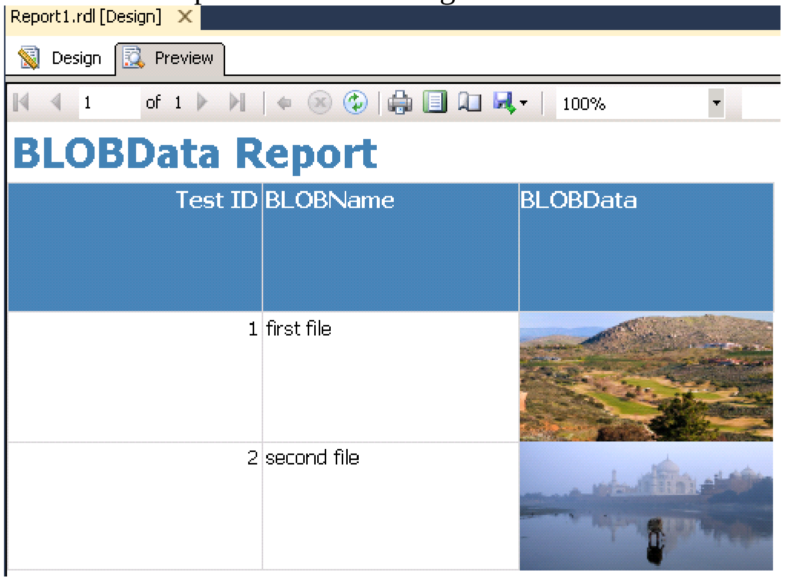
Use this field: [BLOBData]

Use this MIME type: image/jpeg

Figure B-2: Image Properties General page to set BLOB column properties

6. (Optional) Click Size, Visibility, Action, or Border to set additional properties for the image report item

7. **Preview** the report as shown in Figure B-3.



Report1.rdl [Design]

BLOBData Report

Test ID	BLOBName	BLOBData
1	first file	
2	second file	

Figure B-3: Preview report

For more details, see http://technet.microsoft.com/en-us/library/dd239394.aspx

Working with Teradata Spatial Data

Since the release of SQL Server 2008 Reporting Services (SSRS), Teradata users can take advantage of some of the visualization capabilities the tool has to offer. In particular, using Map Report feature to create, display and visualize/analyze your data. Using the Map Wizard will allow you to create maps layers to let you visualize data against a geographic background.

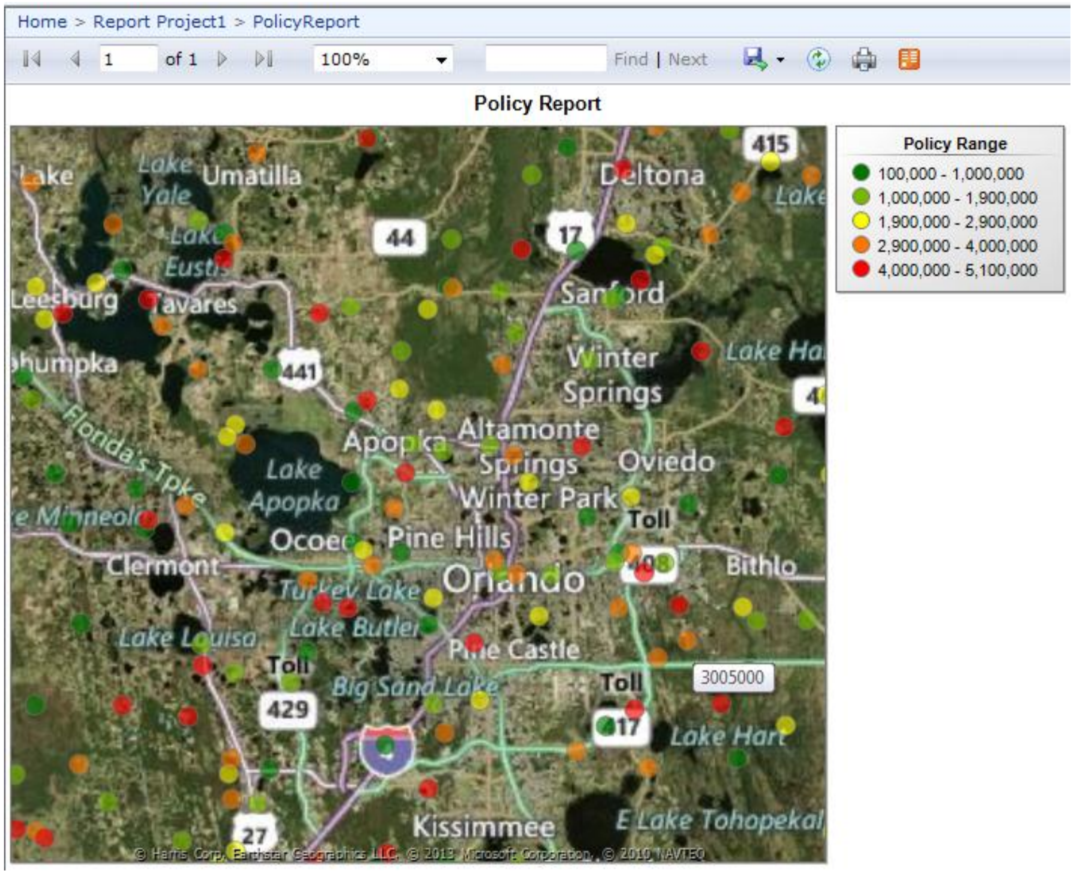


Figure S-1: Map Report called Policy Report

In this section, we will examine the following:

- Map Report Overview
- Map Report Sources for Spatial Data
- Creating a Map Report with Teradata
- Creating an ESRI Shapefile for Teradata
- Creating a Map Report with an ESRI Shapefile
- Adding an analytic data set to a Map Layer
- Adding multiple ESRI Shapefile to a Map Layer
- Creating a Map Report using SSAS cube

Map Report Overview

Adding a map to your report you can visualize analytical data against a geographical background. The four important concepts that need to be understood before you begin your report are:

Spatial Data

Spatial data represents the geographical background in the report and it can be:

- Points: represent specific locations, such as cities and business locations.
- Lines: represent routes and/or paths.
- Polygons: represent areas, such as, states, countries and regions in general.
- Bing map tiles: represent Bing Map aerial or road views in the background.

Typical sources for spatial data are:

- ESRI (Environmental Systems Research Institute, Inc.) shapefiles, more information on ESRI shapefiles for SSRS can be found here.
- Maps from the Map Gallery, provided out-of-the-box maps, but limited to USA.
- Spatial data from SQL Server spatial data sources

Analytical Data

Analytical data represents your business data, such as sales by state or crime rate by city.

Matching Field(s)

Matching field(s) are fundamental to map reports because they are used to tie your Spatial data with your Analytical data. For example to display sales by city: your spatial data set should have at least the name of the city (or code) and its latitude/longitude; your analytical data set should have city name (or code) and total sales; and the matching field in this case would be city name (or code).

Map Viewport

SSRS uses the Map viewport to control what is displayed and how. For example, you can use the map viewport to determine the main/max coordinates of the map to display, center the map, and control the zoom level.

Map Report Sources for Spatial Data

SSRS only supports spatial data stored in SQL Server. Hence, for Teradata users, there is no support to view or run a Teradata spatial query request within Map Report feature. Instead, Teradata spatial data stored in a ST_GEOMETRY data type column must be exported to an ESRI shapefile for Map Report to consume. In this section, we will demonstrate how you can create map report using an exported ESRI shapefile from Teradata to analyze your data against a geographical background. We will use Teradata TDGeoImportExport tool to export our Teradata spatial data to an ESRI shapefile.

It is assumed reader is familiar SSRS Map Reports and with TDGeoImportExport[3] tool which can be download here.

Creating a Map Report for Teradata

The Map Report shown above in Figure S-1 is a Policy Report for a particular state. Basically, shows the location of our policy holders/customers and associated coverage amount. The report uses a Teradata POLICIES table (*see Appendix B – Teradata Database*) which has both spatial 'point' data in the GEOMETRY column and analytic or policy 'coverage' value in the POLICY_VALUE column. Ideally, for performance spatial data and analytic data should exist in separate tables or views. I created 2 views to represent my spatial and analytic data for my report. The first view represents my policy holder locations and the second view represents their policy amount.

replace view POLICIES_GEOM_V as select policy_id, ***geometry*** *from policies;*

replace view POLICIES_DATA_V as select policy_id, ***policy_value*** *from policies;*

Besides for demonstration purposes to show how to add an analytic data set onto a Map Layer. The benefits to creating these views will ensure smaller shapefiles which translates to faster rendering for my report and faster query response for my analytic data set query without returning spatial data which map report does not support anyway. In addition, using views as a source for a shapefile is a good way to leverage Teradata Database and Teradata spatial feature and functions to define your spatial reporting needs. Definition of the table and sample of the data used in this section can be found in *Appendix B – Teradata Database*.

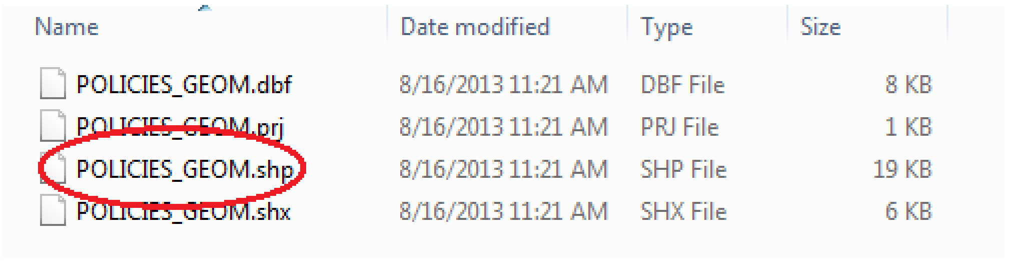
Creating an ESRI Shapefile for Teradata

The first step is to create a shapefile for your Teradata spatial reporting needs. At a high-level, after download and unzipping TDGeoImportExport tool into C:\Temp\TDGeoImportExport\bin directory. The following command was ran to create a shapefile for POLICIES_GEOM_V.

[3] Reference Teradata Orange Book – Teradata 13.10 Spatial Features User's Guide Version B02

```
C:\Temp\TdGeoImportExport\bin java -Xms256m -Xmx1024m -classpath
.;tdgssconfig.jar;terajdbc4.jar;tdgeospatial.jar; com.teradata.geo.TDGeoExport -l
12.105.999.99/dbc,dbc -s dbc -t policies_geom_v -f "ESRI Shapefile" point -o
c:\temp\geodata\policies_geom -n policies_geom
```

The output ESRI shape files will show up in c:\temp\geodata\policies directory as shown in Figure S-2.



Name	Date modified	Type	Size
POLICIES_GEOM.dbf	8/16/2013 11:21 AM	DBF File	8 KB
POLICIES_GEOM.prj	8/16/2013 11:21 AM	PRJ File	1 KB
POLICIES_GEOM.shp	8/16/2013 11:21 AM	SHP File	19 KB
POLICIES_GEOM.shx	8/16/2013 11:21 AM	SHX File	6 KB

Figure S-2: ESRI Shapefile output for POLICIES spatial data

A few helpful notes (as of the writing of this document):

- Include the tdgeospatial.jar in CLASSPATH
- Users should install java jdk (1.6) on the machine you plan to run TDGeoImportExport tool. You can check java version using c:\temp\TDGeoImportExport\bin>**java –version**
  java version "1.6.0_41"
- Delete C:\Temp\geodata\* when creating new shapefiles or create separate directories
- Use the appropriate spatial data option (i.e. point, linestring or polygon) after –f "ESRI Shapefile" based on stored spatial data
- Mixed shapefile types (i.e. point, linestring and polygon) are not supported
- -t option can be table or view
- Create Teradata tables with last column as spatial column (i.e. ST_Geometry data type) and one Unique Primary Index column for better results

Creating a Map Report using ESRI Shapefile

After the shapefile has been created you are ready to create your Map Report using ESRI shapefile.

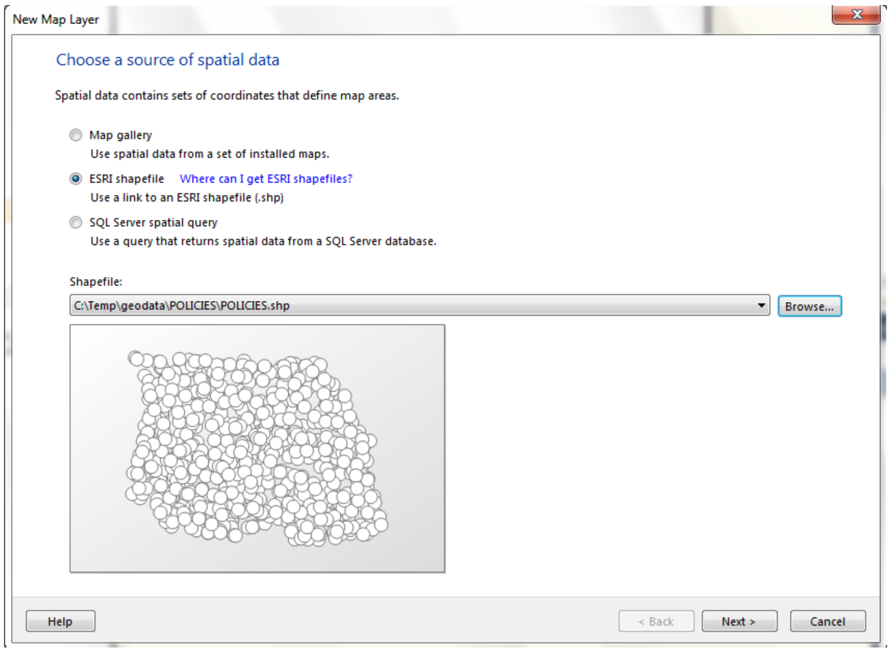
1. Open **SQL Server Data Tools (SSDT) or Microsoft Business Intelligence Development Studio (BIDS)** and create a new **Report Server** Project. Note that this report can also be created with **Report Builder 3.0**.

2. Add a new report. Do not use **Report Wizard** because it does not support creating maps, instead use **Report**. Name the report **PolicyReport.rdl**.

3. Open the toolbar and double click the **Map** report item. This adds a **Map Report** Item to the report and opens the **New Map Layer** wizard. The first page of the wizard is for setting the

source for spatial data. There are three options: Map gallery, ESRI shapefiles, SQL Server spatial query.

Since Map Layer does not support Teradata spatial data type/queries. We need to use an ESRI shapefile for drawing our spatial data. Select the second radio button and browse to the directory where our shapefile POLICIES.shp is located as shown in Figure S-3.


New Map Layer

Choose a source of spatial data

Spatial data contains sets of coordinates that define map areas.

( ) Map gallery
Use spatial data from a set of installed maps.

[✓ ESRI shapefile] Where can I get ESRI shapefiles?
Use a link to an ESRI shapefile (.shp)

( ) SQL Server spatial query
Use a query that returns spatial data from a SQL Server database.

Shapefile: C:\Temp\geodata\POLICIES\POLICIES.shp

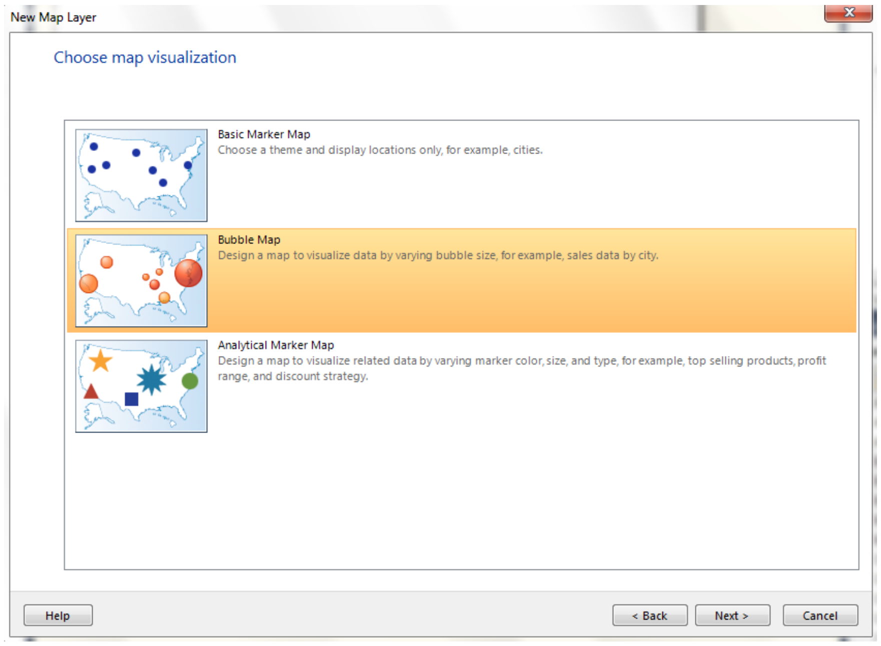
Figure S-3: Choose a source of spatial data

Click **Next** to continue.

4. Review the settings in **Choose spatial data and map view options**. In this example, we will checkmark **Add a Bing Maps layer** tile type Hybrid for our geographical background. Alternatively, you could have used and started with Map Gallery (built-in map reports) and then add your shapefile to the report. Click **Next**.

Adding an analytic data set to a Map Layer

5. There are three map visualizations under **Choose Map Visualization**. If you select either **Color Analytical Map** or **Bubble Map**, you will need to provide a data set that contains data to be analyzed. If our exported ESRI shapefile contained analytic data, we could select **Basic Map** and continue without providing a data set. However, since we did create separate views to access spatial and analytic data and created a shapefile without analytic data. We will use **Color Analytical Map or Bubble Map** and click **Next** to show how you can add an analytic data set to your Map Report.


New Map Layer

Choose map visualization

Basic Marker Map
Choose a theme and display locations only, for example, cities.

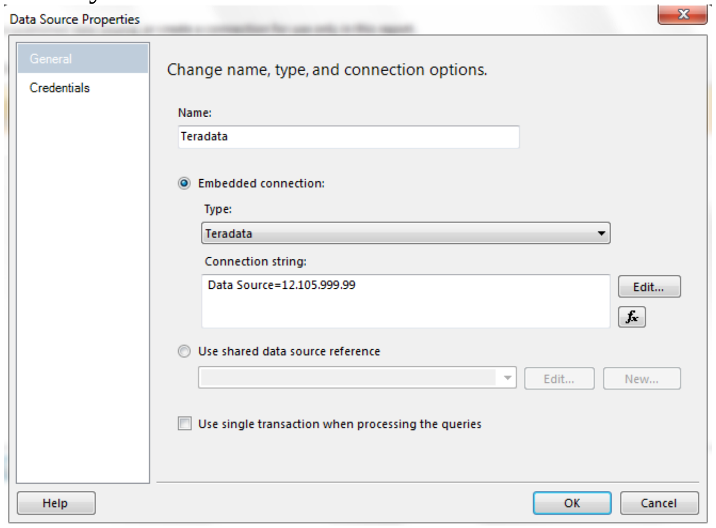
Bubble Map
Design a map to visualize data by varying bubble size, for example, sales data by city.

Analytical Marker Map
Design a map to visualize related data by varying marker color, size, and type, for example, top selling products, profit range, and discount strategy.

Figure S-4: Choose map visualization

6. On **Choose a connection to a data source,** click **New** create a data source (embedded or shared) for analytical data. Choose Teradata for type and ensure Data source references .NET Data Provider for Teradata. Enter Server name, User name and Password and Test Connection see *Appendix A – Teradata Connectivity* for additional connectivity guidance.

7. Name your data source and click **OK** and then **Next** to continue


Data Source Properties

Change name, type, and connection options.

Name: Teradata

Embedded connection: ●

Type: Teradata

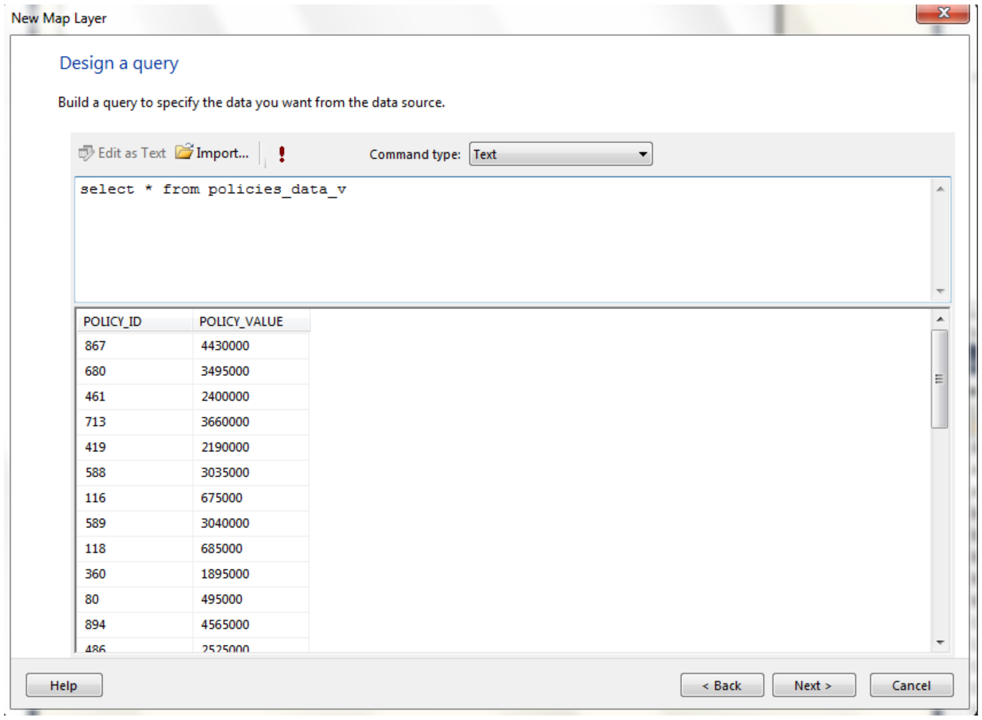
Connection string: Data Source=12.105.999.99

Use shared data source reference ○

Use single transaction when processing the queries ☐

Figure S-5: Data Source Properties

8. In **Design a query** window (Figure S-6), type in our analytic (i.e. POLICIES_DATA_V) data query. The query returns the POLICY_ID and POLICY_VALUE for each policy holder. Click **Next**

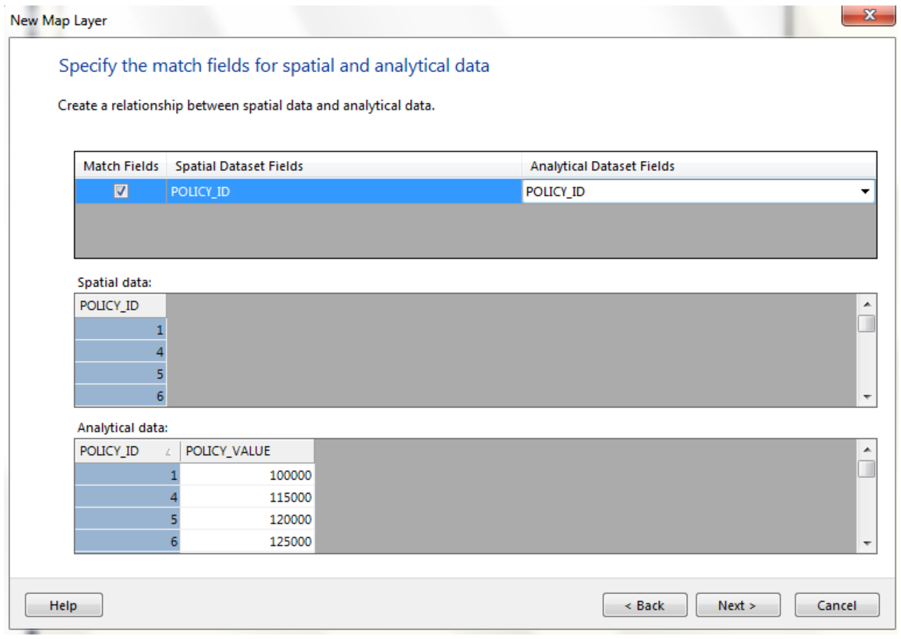


```
select * from policies_data_v
```

POLICY_ID	POLICY_VALUE
867	4430000
680	3495000
461	2400000
713	3660000
419	2190000
588	3035000
116	675000
589	3040000
118	685000
360	1895000
80	495000
894	4565000
486	2525000

Figure S-6: Design a query

9. On the **Specify the match fields for spatial and analytical data** screen. This allows us to specify the relationship between the spatial data loaded from POLICIES.shp file and the data loaded from the query in the previous step. The first grid on the screen allows you to set links between the two data sources. The second grid shows the columns in the spatial data set, highlighting the column we use to match. The third grid contains data from the query, again highlighting the column we use to match. We make a link with POLICY_ID between data sets.



New Map Layer

Specify the match fields for spatial and analytical data

Create a relationship between spatial data and analytical data.

Match Fields	Spatial Dataset Fields	Analytical Dataset Fields
✓	POLICY_ID	POLICY_ID

Spatial data:

POLICY_ID
1
4
5
6

Analytical data:

POLICY_ID	POLICY_VALUE
1	100000
4	115000
5	120000
6	125000

Figure S-7: Specify the match fields for spatial and analytical data

10. On the **Choose color theme and data visualization** screen, select [Policy_Value] for Field to use bubble colors to visualize data. Click **Finish** to close the wizard and **Preview** the report.

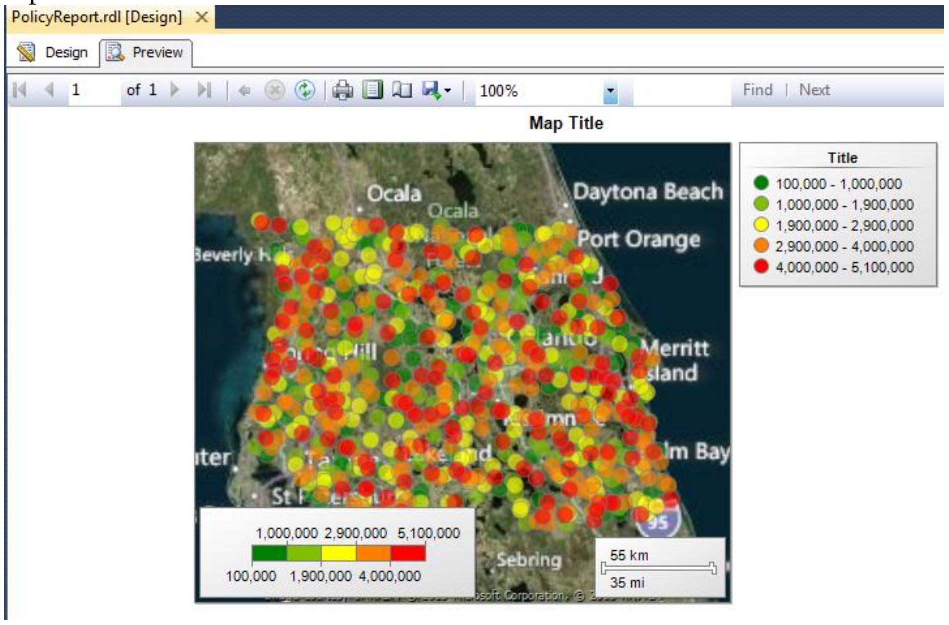


Figure S-8: Preview Policy Report

Note, Legend does not update in Design mode only in Preview mode with appropriate field

11. Next, on the report layout, we do not require **Distance Scale** and **Color Scale**. Right click the Map area and deselect these from the menu as shown in Figure S-9.

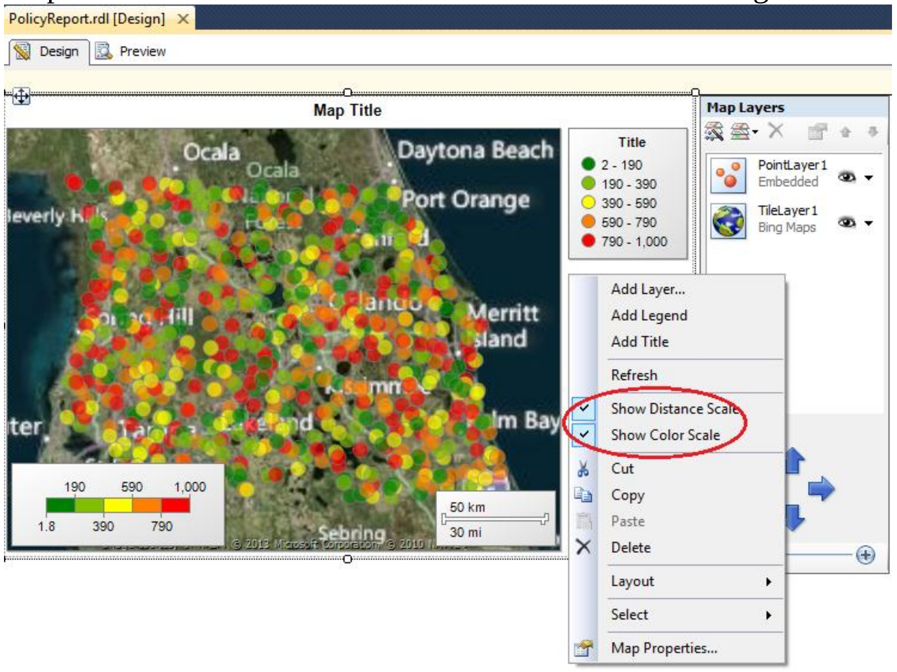


Figure S-9: Map area properties Distance and Color Scale options

12. Next, we want to see Policy Values when we hover over a geographical point or policy holder location. Right click your **PointLayer1** under the **Map Layers** and select **Point Properties** as shown in Figure S-10.

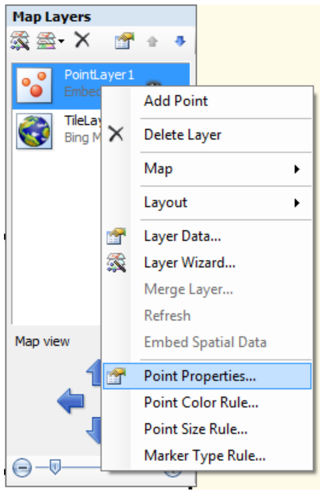
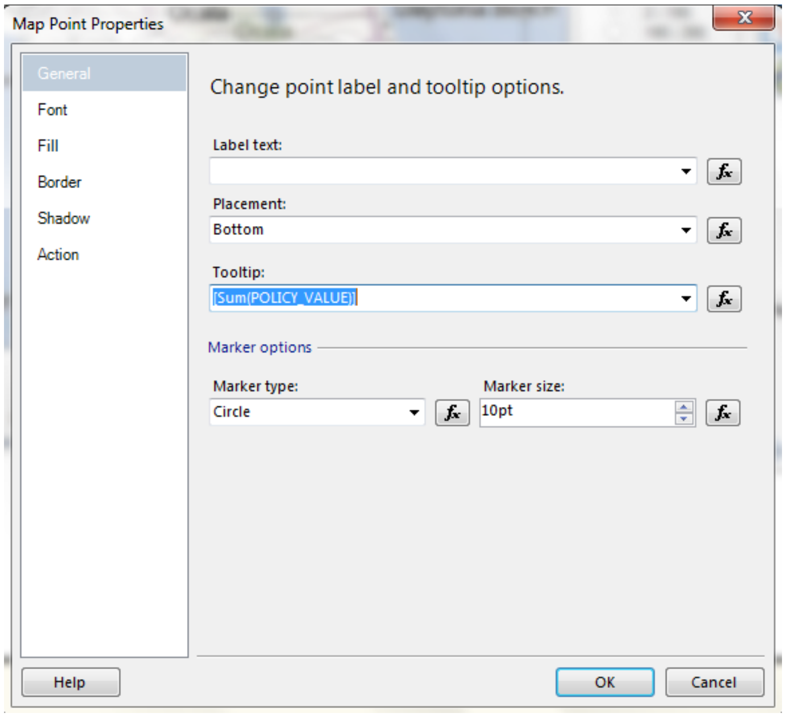


Figure S-10: Map Layer and Point Properties

13. In **Map Point Properties** page, we select **Tooltip** and choose POLICY_VALUE column and click **OK**


Map Point Properties

Change point label and tooltip options.

Label text:

Placement: Bottom

Tooltip: [Sum(POLICY_VALUE)]

Marker options

Marker type: Circle

Marker size: 10pt

Figure S-11: Map Point Properties

14. Finally, we can edit the report and legend names and **Preview** and **Deploy** the report as shown in Figure S-1 above.

Adding multiple ESRI shapefile type to a Map Layer

To continue with our Policy Report example, we are told there is a hurricane reported in this area and would like to know which policy holders this event might affect. We have spatial data that not only identifies the path of the hurricane, but also the affect area once this hurricane hits landfall. Our HURRICANE_PATH table contains spatial 'line' data for the hurricane path and HURRICANE_AREA table with spatial 'polygon' data for the affected area. Definition of tables and samples of the data used in this section can be found in *Appendix B – Teradata Database.*

The first step again, is to create the appropriate ESRI shapefiles for our hurricane path and area spatial data information so we can add to our Map Report. Using TDGeoImportExport export command earlier and the appropriate table/view, directory and ESRI Shapefile format (i.e. linestring and polygon) we can create our shapefiles HURRICANE_PATH.shp and HURRICANE_AREA.shp.

15. To add our new shapefiles, we open our Policy Report in **Design** mode, right click on Map area and click on **Add Layer**

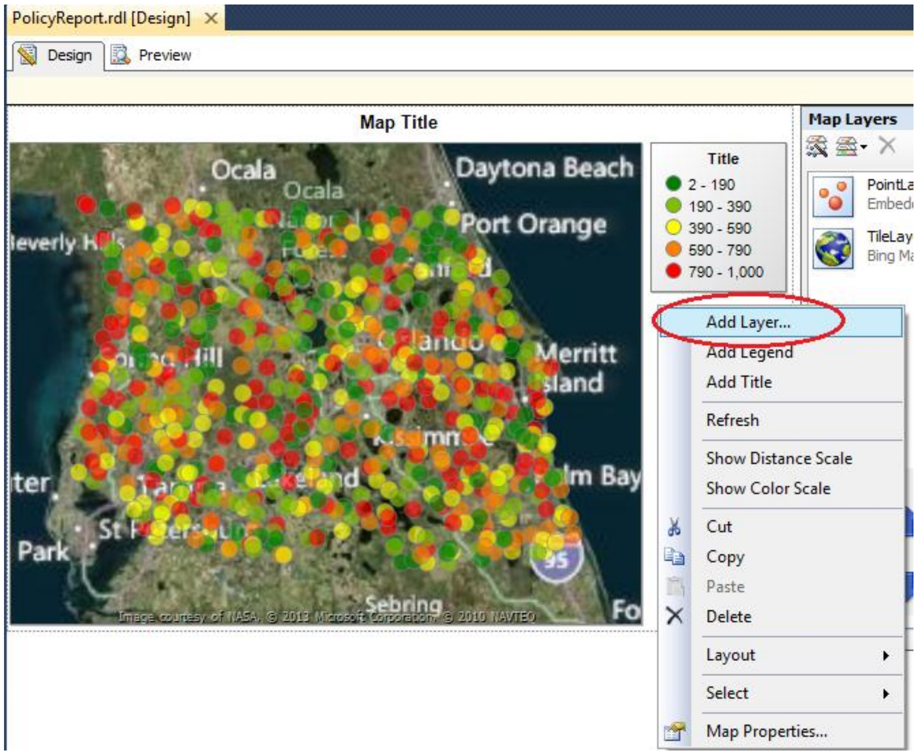


Figure S-11: Map area properties Add Layer option

16. Next, in **New Map Layer** as shown in Figure S-3 above, we choose ESRI Shapefile option and select the HURRICANE_PATH.shp we just created and add that to our report as shown in Figure S-12 and click **Next**.

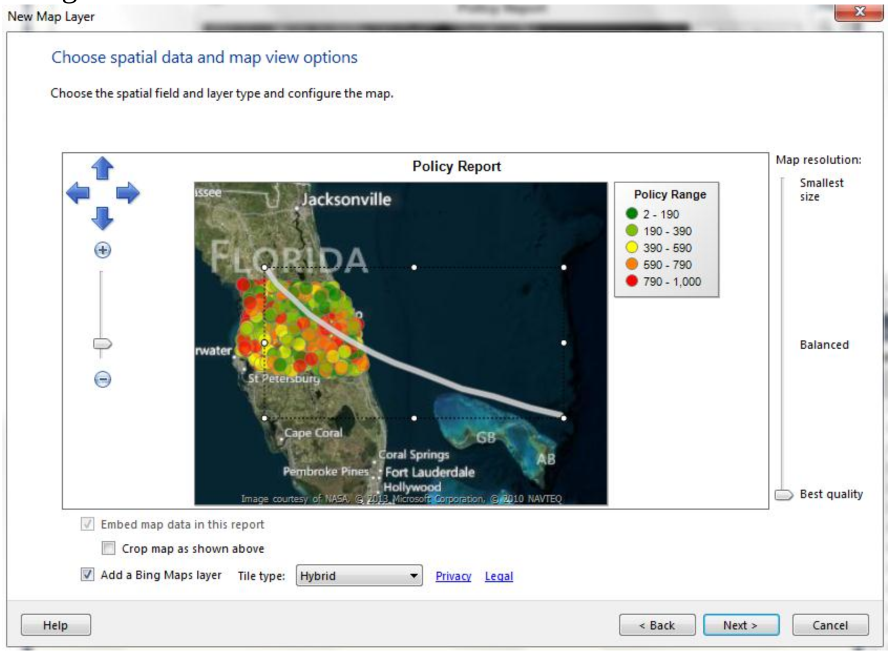


Figure S-12: Choose spatial data and map view options

17. Review **Choose Map Visualization**, our Line Map does not contain any analytic data nor do we care about color and theme for our Line Map, so we pick **Basic Line Map** and click **Next** twice and click **Finish**.

18. Similarly, we can add our HURRICANE_AREA.shp which contains spatial 'polygon' data and again choose the defaults to add the shapefile to our report and **Preview** the report as shown in Figure S-13.

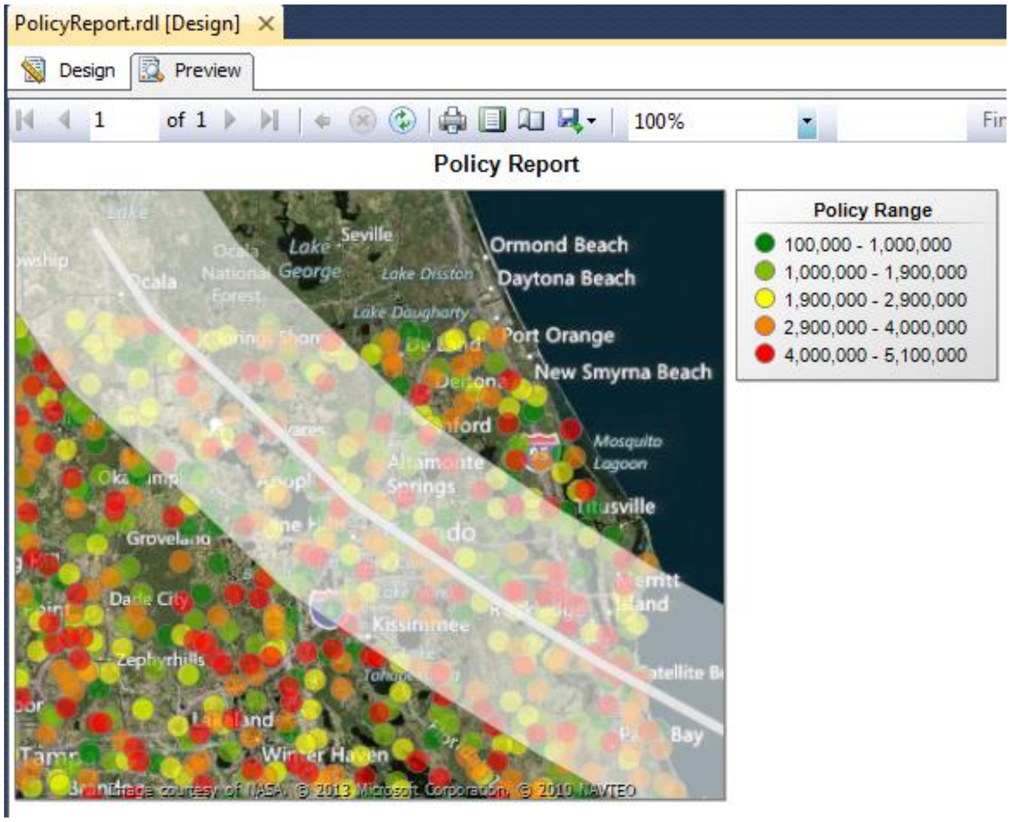


Figure S-13: Policy Report with polygon data layer

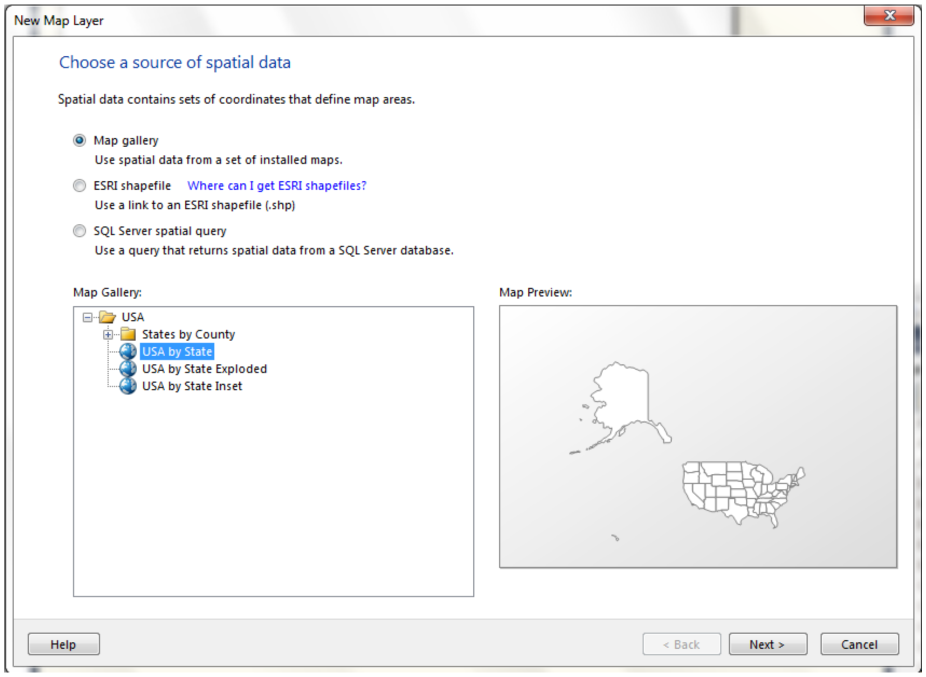
As above has demonstrated, you can use SSRS Map Report feature with Teradata though not ideal versus 'direct' query support like SQL Server. We did show Map Report can consume Teradata spatial data once exported to an ESRI shapefile using TDGeoImportExport tool.

Creating a Map Report using SSAS cube

Geographical visualization of a cube is very similar to adding an analytic data set to your Map Layer. The approach is to map the cube analytic data to a Map Gallery (built-in map reports) or an ESRI shapefile. This requires the SSAS cube (MOLAP or ROLAP) to have some type of geography hierarchy which we can then map to a Map Gallery image or ESRI shapefile based on analytic 'geography' data. Note, SSAS does not support 'true' spatial data types from SQL Server or Teradata. Meaning, if cube contains world data Map Gallery reports will not work because it contains only US maps. Hence, you would need to download an appropriate shapefile to draw the world map. Free shapefiles can be found here or codeplex site. In addition, any analytic data set introduced to a Map Layer must match to map appropriately between spatial and analytic columns. Note SSAS does not support spatial data types from SQL Server or Teradata.

The SSAS cube in this example is called Adventure Works which contains USA retail sales data and a variety of dimensions including a geography hierarchy. We will follow most of the same steps mentioned above with the modification of using Map Gallery report for the USA and adding the cube as an analytic data set to the report.

1. Create a new **Report Server** project, add a new **Report** and double click the **Map** report item. This time we will choose **Map Gallery** (built-in map report) choose **USA by State** and click **Next**.



New Map Layer

Choose a source of spatial data

Spatial data contains sets of coordinates that define map areas.

[✓ Map gallery]
Use spatial data from a set of installed maps.

ESRI shapefile Where can I get ESRI shapefiles?
Use a link to an ESRI shapefile (.shp)

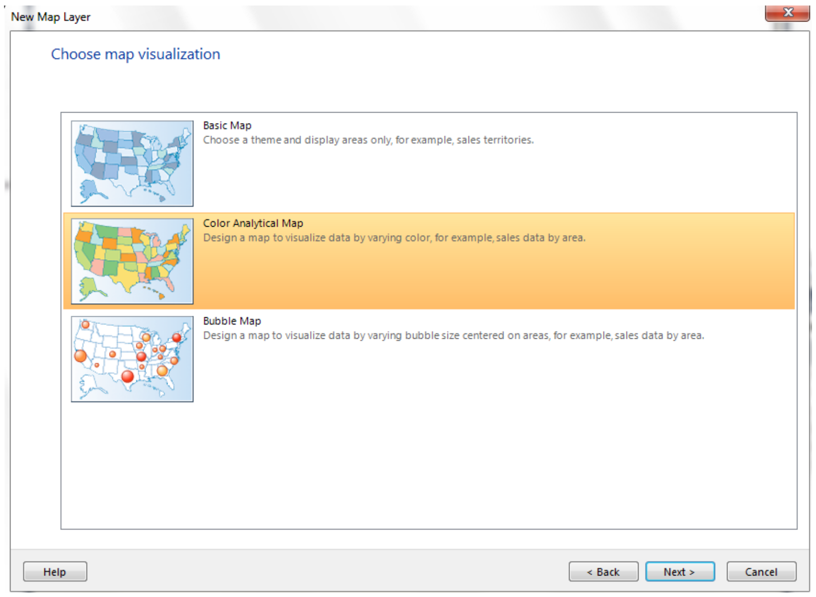
SQL Server spatial query
Use a query that returns spatial data from a SQL Server database.

Map Gallery:
- USA
  - States by County
  - [✓ USA by State]
  - USA by State Exploded
  - USA by State Inset

Map Preview:

Figure S-14: Choose a source of spatial data

2. Review the settings in **Choose spatial data and map view options** and click **Next**
   .
3. Under **Choose Map Visualization**, choose **Color Analytical Map** for our cube and click **Next.**


New Map Layer

Choose map visualization

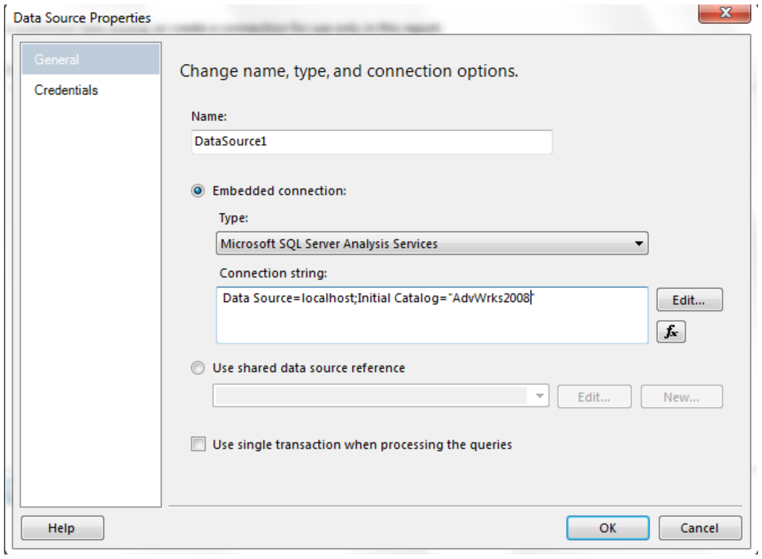
Basic Map
Choose a theme and display areas only, for example, sales territories.

Color Analytical Map
Design a map to visualize data by varying color, for example, sales data by area.

Bubble Map
Design a map to visualize data by varying bubble size centered on areas, for example, sales data by area.

Figure S-15: Choose map visualization

4. On **Choose a connection to a data source,** click **New** create a data source for your cube. Choose Microsoft SQL Server Analysis Services and enter server and cube name and click **OK** twice and click **Next**.


Data Source Properties

Change name, type, and connection options.

Name:
DataSource1

Embedded connection:

Type:
Microsoft SQL Server Analysis Services

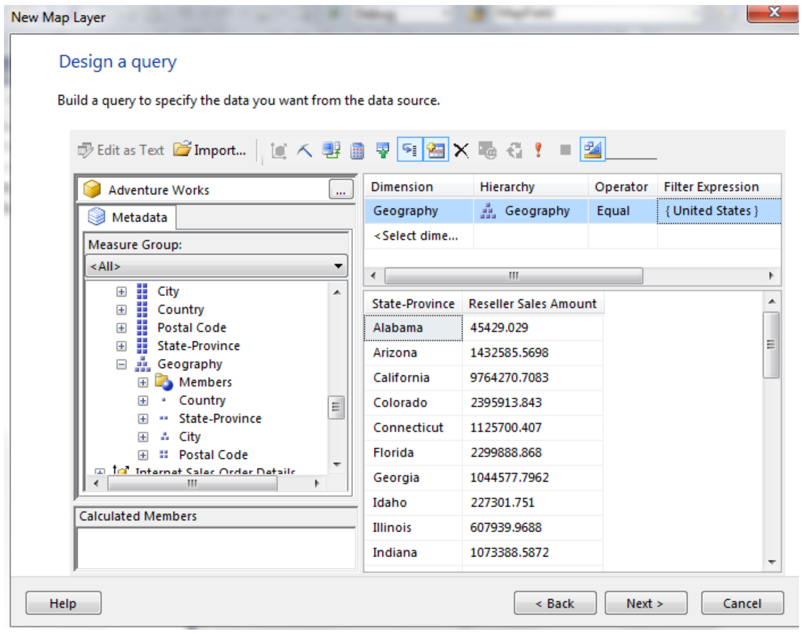
Connection string:
Data Source=localhost;Initial Catalog="AdvWrks2008"

Use shared data source reference

Use single transaction when processing the queries

Figure S-16: Data Source Properties

5. In **Design a query** window, Map Wizard recognizes the data source is a cube and brings up the cube browser to design our analytic query we want to map to our geographical map report. We will drag ‘State-Province’ under ‘Geography’ hierarchy and drag ‘Reseller Extended Amount’ measure. We will also filter on ‘Country’ for USA since our **Map Gallery** report only contains map elements of USA as shown in Figure S-17.



New Map Layer

Design a query

Build a query to specify the data you want from the data source.

Edit as Text Import...

Adventure Works

Metadata

Measure Group:
<All>

- City
- Country
- Postal Code
- State-Province
- Geography
  - Members
  - Country
  - State-Province
  - City
  - Postal Code
- Internet Sales Order Details

Calculated Members

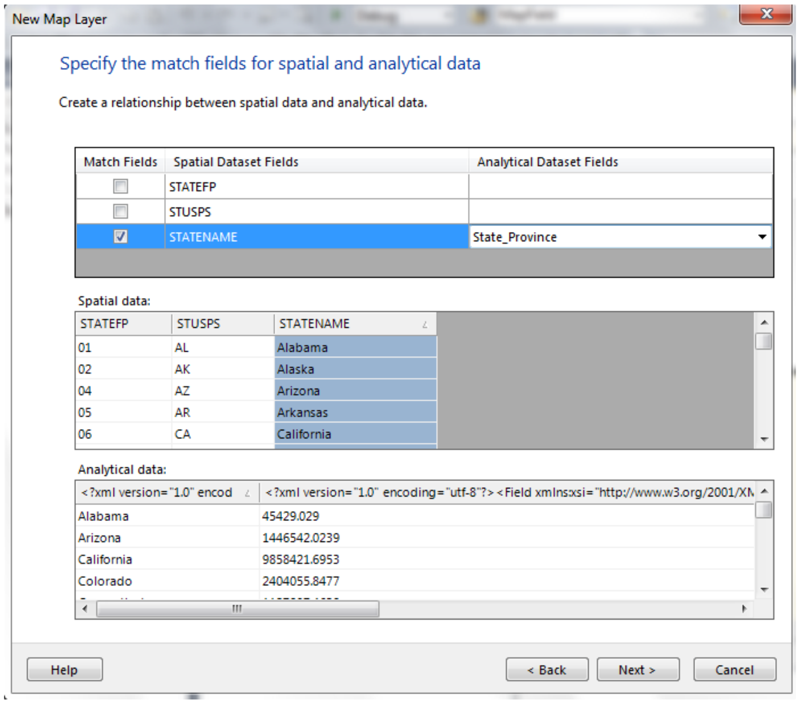
Dimension	Hierarchy	Operator	Filter Expression
Geography	Geography	Equal	{ United States }
<Select dime...			

State-Province	Reseller Sales Amount
Alabama	45429.029
Arizona	1432585.5698
California	9764270.7083
Colorado	2395913.843
Connecticut	1125700.407
Florida	2299888.868
Georgia	1044577.7962
Idaho	227301.751
Illinois	607939.9688
Indiana	1073388.5872

Help < Back Next > Cancel

Figure S-17: Design a query

6. Similar to step 9 above, on the **Specify the match fields for spatial and analytical data** screen. We will specify a relationship between the spatial data from **Map Gallery** report **USA by State** 'STATENAME' and our analytic (cube) data column 'State-Province' and click **Next**.



New Map Layer

Specify the match fields for spatial and analytical data

Create a relationship between spatial data and analytical data.

Match Fields	Spatial Dataset Fields	Analytical Dataset Fields
☐	STATEFP	
☐	STUSPS	
☑	STATENAME	State_Province

Spatial data:

STATEFP	STUSPS	STATENAME
01	AL	Alabama
02	AK	Alaska
04	AZ	Arizona
05	AR	Arkansas
06	CA	California

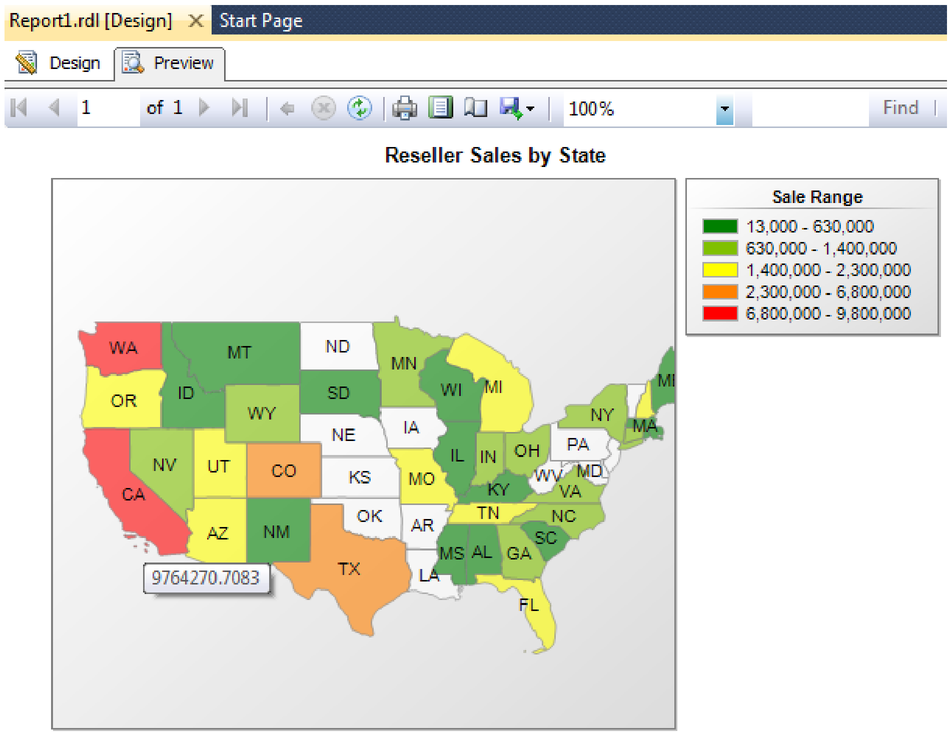
Analytical data:

<?xml version="1.0" encod	<?xml version="1.0" encoding="utf-8"?><Field xmlns:xsi="http://www.w3.org/2001/XM
Alabama	45429.029
Arizona	1446542.0239
California	9858421.6953
Colorado	2404055.8477

Help < Back Next > Cancel

Figure S-18: Specify the match fields for spatial and analytical data

7. On the **Choose color theme and data visualization** screen, select [Reseller_Sales_Amount] for Field to visualize data. Click **Finish** to close the wizard and **Preview** the report.



Reseller Sales by State

Sale Range
🟩 13,000 - 630,000
🟩 630,000 - 1,400,000
🟨 1,400,000 - 2,300,000
🟧 2,300,000 - 6,800,000
🟥 6,800,000 - 9,800,000

Figure S-19: Preview Reseller Sales by States report

As above has demonstrated, you can use SSRS Map Report feature with a SSAS cube. In this case, we showed how you can use a spatial map report from Map Gallery (or an ESRI shapefile) and combine it with a SSAS cube data.

Additional Considerations on SSRS Maps

The legend, color scale and rules can sometimes be trick configure or understand, in this article you will have a detailed explanation of how they work and how they are configured.

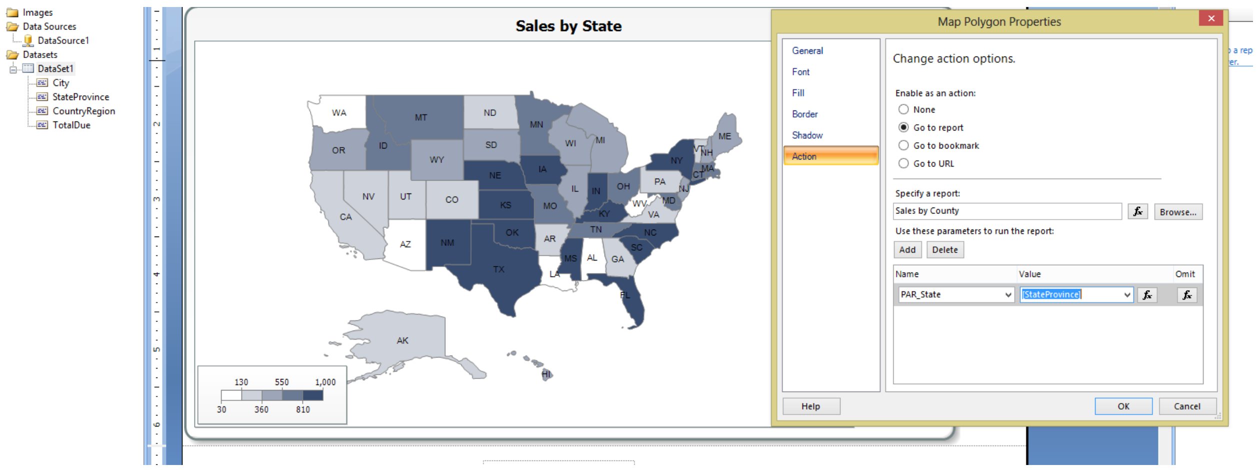
The Map Viewport does not support zoom and change map center (move the map) in run-time. A possible work around to this scenario is to set map viewport properties, such as, Zoom level (%), View Center (X)%, and View Center Y(%), through report parameters. This article gives the step by step.

If you need to enable drill down functionality in your maps, for example from country to state, you could use a combination of report actions and sub reports.

Let’s suppose you are creating a report with sales by state and you want to enable your users to click on a state and then open a report on sales by county for that state. In this case you will need:

1. Create a sales by state report.

2. Create a second report where you have sales by county having the state as parameter (in my example it is called PAR_State. You will need to load the appropriate shapefiles on the map viewport.
3. Now you can go the Polygon properties of the sales by state report, and configure an action to go to the sales by county report, as shown in the figure S-20 below.



Change action options.

Enable as an action:
None [✓ Go to report] Go to bookmark Go to URL

Specify a report: Sales by County

Use these parameters to run the report:

Name	Value	Omit
PAR_State	[StateProvince]	

Figure S-20: Map Polygon Properties

There are several good articles that you can use to learn more about the map viewport and also to troubleshoot report maps.

Report Model on the Reporting Services Server

Report Models is a feature available to SSRS report developers in SQL Server prior to SQL Server 2012. The recommended approach to modeling report data sets in SSRS with SSRS 2012 is using BI Semantic Models via Power Pivot or SSAS 2012 Tabular Models. You can find the details of building Semantic Models for Teradata reports in SSRS 2008 at this link.

Although you can you continue to use existing report models as data sources in SQL Server 2012 Reporting Services reports you should consider updating your reports to remove their dependency on report models.

SQL Server 2012 Reporting Services does not include tools for creating or updating report models. For more information, see Breaking Changes in SQL Server Reporting Services in SQL Server 2012.

Working with Teradata Data Types and Semantic Query Functions

Most of the Teradata native data types are supported in Reporting Services. The following table shows the mapping from Teradata database data types to the .NET types.

Teradata database type	System.Data.DbType mapping
BIGINT	Int64
BINARY LARGE OBJECT, BLOB	Binary
BYTE	Byte
BYTEINT	Sbyte
CHAR, CHARACTER	StringFixedLength
CHAR VARYING, CHARACTER VARYING (same as VARCHAR)	String
CHARACTER LARGE OBJECT, CLOB	String
DATE	Date
DEC, DECIMAL	Decimal
DOUBLE PRECISION	Double
FLOAT	Double
GRAPHIC	StringFixedLength
INT, INTEGER	Int32
INTERVAL DAY	StringFixedLength
INTERVAL DAY TO HOUR	StringFixedLength
INTERVAL DAY TO MINUTE	StringFixedLength
INTERVAL DAY TO SECOND	StringFixedLength
INTERVAL HOUR	StringFixedLength
INTERVAL HOUR TO MINUTE	StringFixedLength
INTERVAL HOUR TO SECOND	StringFixedLength
INTERVAL MINUTE	StringFixedLength
INTERVAL MINUTE TO SECOND	StringFixedLength
INTERVAL MONTH	StringFixedLength
INTERVAL SECOND	StringFixedLength
INTERVAL YEAR	StringFixedLength
INTERVAL YEAR TO MONTH	StringFixedLength
LONG VARCHAR	String
LONG VARGRAPHIC	String
NUMERIC	Decimal
REAL	Double

SMALLINT	Int16
TIME	Time
*TIME WITH TIMEZONE	String (not supported)
TIMESTAMP	DateTime
*TIMESTAMP WITH TIMEZONE	String (not supported)
**user-defined type* (UDT)	Partially supported. See Note following.**
VARBYTE	Binary
VARCHAR	String
VARGRAPHIC	String

* These types are not supported because their type mapping to System.Data.DbType types were not compatible with types that semantic query engine functions expected.

** Note: There are two types of UDTs: distinct, based on a single predefined type, and structured, that is a collection of one or more fields (similar to a C language struct). For structured UDTs, the Teradata database transforms the structured UDT into a primitive data type when it is selected. Distinct UDTs are converted to their underlying primitive type. In practice, the database column type is exposed using the Columns schema as a UDT. However, when the field data is read, the underlying primitive type is returned. For example, if a UDT is defined as an Integer, then the data reader will report the column type as *Int32* and the *GetInt32* can be used to retrieve column values.

The INTERVAL data types appear as character strings using the .NET Data Provider for Teradata. Teradata supports using ANSI interval expressions and arithmetic operators on interval data types as well as certain aggregation functions. However, be aware of certain string functions, such as RTrim and LTrim, which do not yield meaningful results when used over database fields of type INTERVAL, or which may result in errors at the query execution time.

Teradata Host Naming Convention

Teradata .NET Data Provider performs the following actions to resolve the host name that is entered into the Connection Properties dialog box to an IP address:

1. It will attempt to resolve the hostname using the Teradata host naming convention (TDPID or DBC-Name).

2. If it cannot resolve the host name using the preceding method, then it will attempt to resolve the server name as it is entered by the user.

Teradata users are most familiar with the Teradata system naming conventions. However, for the Reporting Services users, the host naming convention is explained as follows:

The hostname that is entered into the **Server name** box in the Connection Properties dialog box is appended with a COP*n* suffix, where *n* is a sequential number starting from 1, before resolving the hostname to a network address. Hence the number *n* corresponds to the number of Gateways (COP) supported by the system. For example, to connect to a system called MYDATA, the following entry is required in the hosts file, which is located under *%SystemRoot%\System32\drivers\etc\hosts* (assuming the MYDATA machine IP address is 10.0.0.1):

```
10.0.0.1    MYDATAcop1
```

Then, the .NET Data Provider for Teradata resolves the hostname properly. The system name is also restricted to eight characters or less (the system name refers to the part without COP*n* suffix; MYDATA in this example).

Alternatively, you can use an IP address instead of the hostname to establish a connection.

Known Issues

Working with Large Rows

The maximum row size for Teradata is approximately 64,000 bytes (roughly 64KB). If a SQL query requires row sizes larger than 64KB, then the database will generate an error similar to the following:

```
[Teradata Database] [3577] Row size or Sort Key size overflow
```

If you ever encounter this error, then you may need to restructure your query to accommodate this Teradata 64KB row limit size.

Teradata INTERVAL Data Type Range

Teradata databases have a limit of 9999 units on the INTERVAL data type. Therefore, if a SQL or semantic query exceeds this limit, it generates an error. For more information about the INTERVAL data type and its usage, see Teradata reference manuals. Generally, the INTERVAL data type is used for performing arithmetic operations on the TIMESTAMP type.

For example, assume that you have a table called *promotions* and this table has a columns called *start_time* and *duration*; with *duration* in hours. To obtain the end date of all promotion campaigns, write the following query:

```
sel

   start_time as “StartTime”,

   StartTime + cast(duration as interval hour(4)) as “EndTime”

 from promotions;
```

If you have a promotion with duration greater than 9999 hours, the query will generate an error, because you are using an interval type of greater than 9999 units.

You can run into similar issues when using semantic queries. For example, assume your are using *DateDiff* and *DateAdd* functions in Report Builder to subtract the time difference in hours of two columns and then add that difference to a third column. If the number of hours obtained from the first operation is greater than 9999 hours, then you will exceed the INTERVAL type limit.

Installing oReplace User-Defined Functions

By default, Teradata databases do not have a string REPLACE function. A REPLACE function is used for replacing all instances of one string with another. For example, if you replace all instances of *aa* in *Faa* with *ee*, then you will end up with *Fee*.

However, REPLACE functionality is required by Reporting Services, specifically if Replace functionality is used in Report Builder with your report model. Luckily, Teradata offers a group of User-Defined Functions (UDF) called *Oracle UDFs* for compatibility purposes. The REPLACE function is one of these UDFs. For more information and to download the UDF bundle, go to Teradata Download and download *Oracle UDF*.

Note: The REPLACE Oracle UDF is called *oReplace*, perhaps to distinguish that it is an Oracle UDF. The following sequence of commands must be used to install the oReplace UDF:

```
REPLACE FUNCTION oreplace(

     Str VARCHAR(4000),

    aFrom VARCHAR(512),

     aTo VARCHAR(512)

)RETURNS   VARCHAR(4000)

LANGUAGE C

NO SQL

SPECIFIC oreplace2

EXTERNAL NAME 'SC!oreplace2!<fullpath>/oreplace2.c'

PARAMETER STYLE SQL;
```

The oReplace is written in C, and upon issuing the preceding commands. The code is compiled and installed on the server. However, before you install this UDF, make sure that values in bold are adjusted based on your requirements. Specifically, you must adjust the value pertaining to the sizes of input and output strings, which is 4000 in the preceding code sample.

Setting the return output to 64KB will trigger the error that was described in the preceding Working with Large Rows section, and setting it too low may truncate your strings without notification. The best solution is to adjust these values based on your reporting requirements before installing the UDF.

Troubleshooting and Additional Information

Teradata Authentication Mechanisms

The following figure shows the authentication mechanisms available in the **Teradata Connection** dialog box and which a report author can use while creating a data source in Reporting Services.

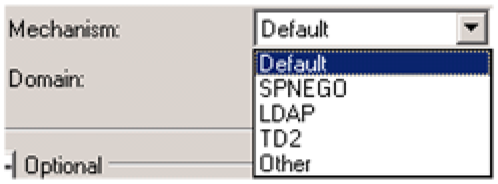


Figure 25: Authentication mechanism

The authentication mechanisms are described as follows:

· SPNEGO (The Simple and Protected GSS-API Negotiation): SPNEGO protocol negotiates different security mechanisms. It is used to negotiate with the Teradata Gateway to use Microsoft Kerberos on Windows.

· TD2 (Teradata Method 2): Uses a user name and password to authenticate.

· LDAP (Lightweight Directory Access Protocol): LDAP is a standard directory protocol that can be used for authentication as well. For more information about LDAP, see LDAP.

For more information about the authentication mechanisms, session security, and Default authentication mechanism, see the help file that accompanies the .NET Data Provider for Teradata.

Exception Caught Instantiating TERADATA Report Server Extension

After you install Reporting Services, you might see the following error message in the Reporting Services log file and the system event log:

"Exception caught instantiating TERADATA report server extension."

This error is logged under the following circumstances:

· A new installation of SQL Server Reporting Services.

· Each time the Report Server service restarts.

This error occurs because the Teradata extension is registered in the Reporting Services configuration file by default, but the Teradata provider is not shipped with SQL Server or as part of the .NET Framework. You can ignore this error if you are not planning to need Teradata connectivity. However, if you want to work around this issue, do one of the following:

· Open the Reporting Services configuration file (Reportserver.config), and remove or comment out the Teradata extension. Do this only if you do not require functionality that the Teradata extension provides.

· Install the .NET Data Provider for Teradata. Do this only if you require functionality that the Teradata extension provides. You can obtain the provider from the Teradata Web site. Reporting Services.

You may see a similar exception when trying to deploy a report server, or report model project, to a report server which does not have the provider installed, and when the project contains references to the .NET Data Provider for Teradata:

An attempt has been made to use a data extension 'TERADATA' that is either not registered for this report server or is not supported in this edition of Reporting Services.

If you encounter this error, then you will need to install the.NET Data Provider Teradata on the report server. For more information, see the Prerequisites section.

Conclusion

This paper explained usage of Teradata as a Reporting Services data source. It also explored tips and tricks of working with Teradata and Reporting Services.

For more information:

SQL Server Reporting Services: http://www.microsoft.com/en-us/sqlserver/solutions-technologies/business-intelligence/reporting.aspx

SQL Server Reporting Services Forum: http://social.msdn.microsoft.com/Forums/sqlserver/en-US/home?forum=sqlreportingservices

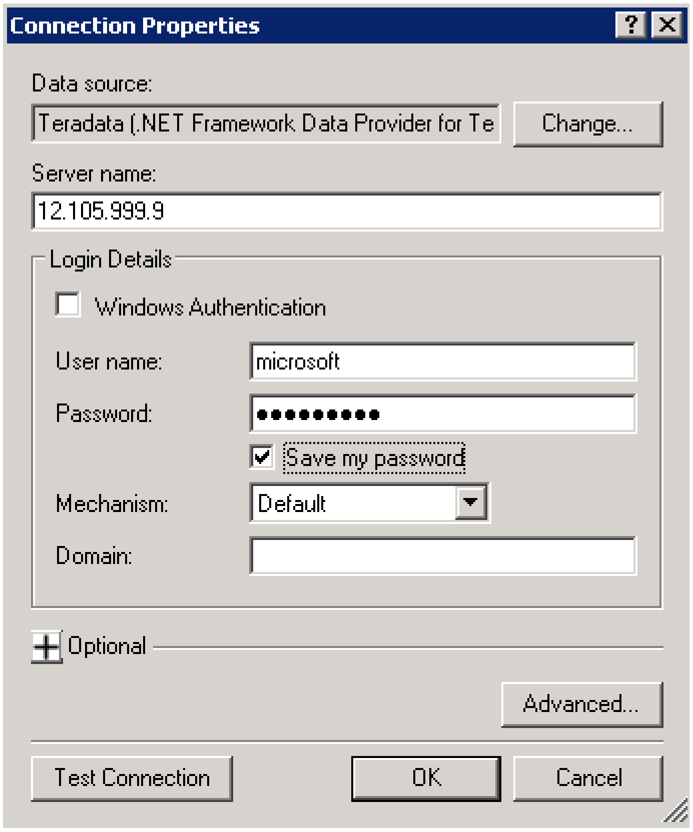
SQL Server TechCenter: http://technet.microsoft.com/en-us/library/ms159106.aspx

SQL Server DevCenter: http://msdn.microsoft.com/en-us/library/ms159106.aspx

Appendix A – Teradata Connectivity

The Connection Properties is native to the .NET Data Provider for Teradata as shown in Figure A-1. Readers are encouraged to reference the .Net Data Provider for Teradata Help for more information on some of the following settings mentioned below, available under *Program File→.Net Data Provider for Teradata→Help.*

In the **Connection Properties**, at a minimum requires the server name or IP address, user name and password credentials.



Data source: Teradata (.NET Framework Data Provider for Te

Server name: 12.105.999.9

☐ Windows Authentication

User name: microsoft

Password: ●●●●●●●●●●

☑ Save my password

Mechanism: Default

Domain:

Figure A-1: Connection Properties for Teradata

Before proceeding, be aware that some additional connection string parameters can provide a better DSV experience. Especially, when interrogating large databases and user credentials with access to large number of schemas.

The **Connection Properties** allows you to build a connection string rather than editing one directly. If you click **Advanced**, you will see all of the connection string properties that are available for the .NET Data Provider for Teradata (Figure A-2)

The following list includes properties relevant to ensure optimal behavior experience between Reporting Services and Teradata Database using the .NET Data Provider for Teradata.

Response Buffer Size (8K to 64K)

To maximize the .Net Data Provider for Teradata response buffers used for SQL requests and data retrieval administrators are encouraged to perform their own due diligence on benefits when setting from 8K to 64K with the read ahead option below. With the understanding this allocates additional resources. This will help in large answer set retrieval. The difference in performance may be considerable if the result set is large. If the Row Size of the result set is large and the Number of Rows in the result set is also large, then better performance can be expected by increasing the 'Maximum Response Buffer'. With small result sets the difference in performance is negligible.

Read Ahead = True

On default, Read Ahead is set to True. True enables additional buffering for results, while the current buffer is consumed by an application. This is useful when executing queries that return large result sets and should be considered when building MOLAP cubes.

Use X Views

On default, Use X Views is set to True. True limits the schema data to objects associated with the requesting user, such as objects the user owns, is associated with, has been granted privileges on, or is assigned a role which has privileges.

To improve DSV experience, set 'Use X Views' to false and ensure User Name/credentials have been scoped appropriately for your reporting project. Hence, appropriate Semantic Layer in place is critical.

Restrict to Default Database

On default, Restrict to Default Database is set to False.

To improve DSV experience, set this parameter to True to alleviate the number of calls generated by DSV component and restrict all schema collections to ‘just’ the Default Database (see next bullet).

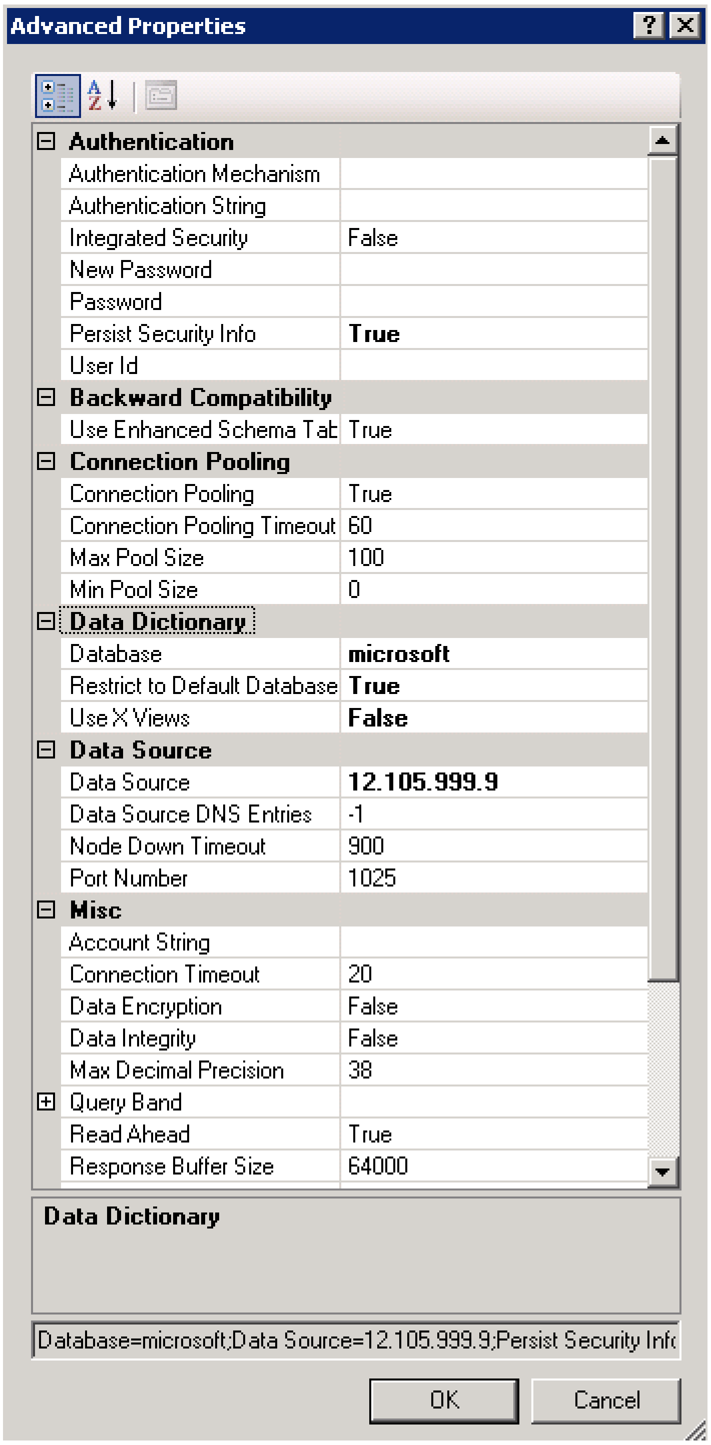
Database

On default, Database is blank. This can be set via the connection string/Connection Manager screen (see below) or it will default to user’s profile.

To improve DSV experience, set this parameter to the (default) database/schema in question.

You can also edit the data source directly and update the connection string. The following example is a rich connection string including the settings for restricting the default database as well as disabling the use of X views:

Database = db_name; Use X Views=False; Restrict to Default Database=True; Read Ahead = True



Advanced Properties

Authentication	
Authentication Mechanism	
Authentication String	
Integrated Security	False
New Password	
Password	
Persist Security Info	**True**
User Id	
Backward Compatibility	
Use Enhanced Schema Tab	True
Connection Pooling	
Connection Pooling	True
Connection Pooling Timeout	60
Max Pool Size	100
Min Pool Size	0
Data Dictionary	
Database	**microsoft**
Restrict to Default Database	**True**
Use X Views	**False**
Data Source	
Data Source	**12.105.999.9**
Data Source DNS Entries	-1
Node Down Timeout	900
Port Number	1025
Misc	
Account String	
Connection Timeout	20
Data Encryption	False
Data Integrity	False
Max Decimal Precision	38
Query Band	
Read Ahead	True
Response Buffer Size	64000

Data Dictionary

Database=microsoft;Data Source=12.105.999.9;Persist Security Infc

Figure A-2: Advanced Properties for .NET Data Provider for Teradata

Appendix B – Teradata Database

Temporal Tables and Data

```
CREATE MULTISET TABLE department (
    d_id INTEGER,
    dept_name VARCHAR(100) CHARACTER SET LATIN CASESPECIFIC,
    eff_date PERIOD(DATE) AS VALIDTIME)
PRIMARY INDEX ( d_id );

CREATE MULTISET TABLE employee (
    e_id INTEGER,
    name VARCHAR(100) CHARACTER SET LATIN CASESPECIFIC,
    d_id INTEGER,
    depassigned PERIOD(DATE) AS VALIDTIME)
PRIMARY INDEX ( e_id );

CREATE MULTISET TABLE salary (
    e_id INTEGER,
    salary DECIMAL(8,2),
    sal_effective PERIOD(DATE) AS VALIDTIME)
PRIMARY INDEX ( e_id );
```

Department

```
SEQUENCED VALIDTIME insert into department values (1, 'Sales', PERIOD (DATE '2009-01-01', UNTIL_CHANGED) );
SEQUENCED VALIDTIME insert into department values (2, 'Marketing', PERIOD (DATE '2009-01-01',
    UNTIL_CHANGED));
```

Employee

```
SEQUENCED VALIDTIME insert into employee values (1, 'Steve', 1, PERIOD (DATE '2009-01-01', DATE '2011-06-01') );
SEQUENCED VALIDTIME insert into employee values (2, 'PPaolo', 1, PERIOD (DATE '2009-01-01', DATE '2010-12-31') );
SEQUENCED VALIDTIME insert into employee values (2, 'PPaolo', 2, PERIOD (DATE '2011-01-01', UNTIL_CHANGED) );
SEQUENCED VALIDTIME insert into employee values (3, 'John', 1, PERIOD (DATE '2009-01-01', UNTIL_CHANGED) );
SEQUENCED VALIDTIME insert into employee values (4, 'Jack', 1, PERIOD (DATE '2010-06-01', UNTIL_CHANGED) );
```

Salary

```
SEQUENCED VALIDTIME insert into salary values (1, 40000, PERIOD (DATE '2009-01-01', DATE '2009-12-31') );
SEQUENCED VALIDTIME insert into salary values (1, 50000, PERIOD (DATE '2010-01-01', DATE '2010-12-31') );
SEQUENCED VALIDTIME insert into salary values (1, 60000, PERIOD (DATE '2011-01-01', DATE '2011-06-01') );
SEQUENCED VALIDTIME insert into salary values (2, 41000, PERIOD (DATE '2009-01-01', DATE '2009-12-31') );
SEQUENCED VALIDTIME insert into salary values (2, 51000, PERIOD (DATE '2010-01-01', DATE '2010-12-31') );
SEQUENCED VALIDTIME insert into salary values (2, 61500, PERIOD (DATE '2011-01-01', UNTIL_CHANGED) );
SEQUENCED VALIDTIME insert into salary values (3, 42000, PERIOD (DATE '2009-01-01', DATE '2009-12-31') );
SEQUENCED VALIDTIME insert into salary values (3, 52000, PERIOD (DATE '2010-01-01', DATE '2010-12-31') );
SEQUENCED VALIDTIME insert into salary values (3, 62000, PERIOD (DATE '2011-01-01', UNTIL_CHANGED) );
SEQUENCED VALIDTIME insert into salary values (4, 53000, PERIOD (DATE '2010-06-01', DATE '2010-12-31') );
SEQUENCED VALIDTIME insert into salary values (4, 63000, PERIOD (DATE '2011-01-01', UNTIL_CHANGED) );
```

BLOB Table and Data

You can import Large Objects into Teradata Database using Teradata SQL Assistant tool by setting up an import job. Or if you have images on SQL Server database you can move BLOB data using SQL Server Integration Services (SSIS) with Teradata ODBC Driver.

Load BLOB data from Flat file to Teradata

1. Extract the 2 images and place them in C:\Temp\Blobdata

   Images.zip

2. Create Teradata table BLOBTest table

```
CREATE SET TABLE BLOBTest ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      TestID INTEGER,
      BLOBName VARCHAR(50) CHARACTER SET LATIN NOT CASESPECIFIC,
      BLOBData BLOB(2097088000))
PRIMARY INDEX ( TestID );
```

3. You need a parameterized query:

```
insert into BLOBTest (testid, blobname, blobdata) values (?, ?, ?B);
```

   The question mark fields in the “values” clause correspond to the columns in your control file. The “question mark B” column tells SQL Assistant that the third column is a BLOB and that the control file lists the file name of the blob. The control file (i.e. notepad) looks like this:

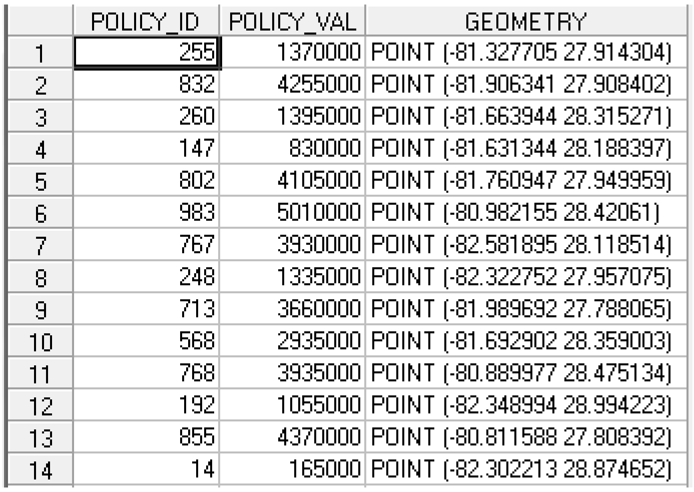
```
1,first file,BLOBData001.jpeg
2,second file,BLOBData002.jpeg
```

   Note, ensure you set the appropriate delimiter in SQL Assistant under Tool→Options

4. Finally, you set SQL Assistant into “Import mode” (i.e. File→Import Data) and run the query. When you do, SQL Assistant will prompt you for the location of the control file. Notice that the file names (the 3rd column) do not have paths. SQL Assistant assumes that the files representing the LOB data are in the same directory as the control file. If you attempt to use path names to locate files in other directories SQL Assistant will abort the import and display an error in the status bar.

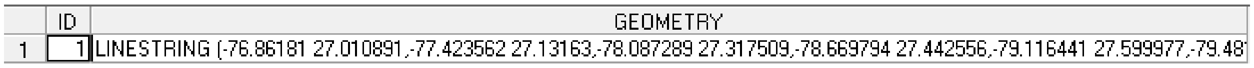
Spatial Table and Data

```
CREATE SET TABLE POLICIES ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      POLICY_ID INTEGER,
      POLICY_VALUE INTEGER,
      GEOMETRY SYSUDTLIB.ST_Geometry)
UNIQUE PRIMARY INDEX ( POLICY_ID );
```



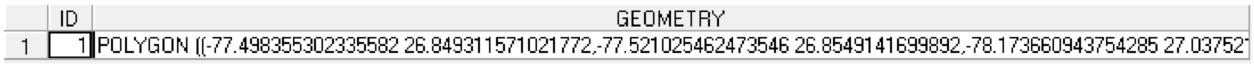
	POLICY_ID	POLICY_VAL	GEOMETRY
1	255	1370000	POINT (-81.327705 27.914304)
2	832	4255000	POINT (-81.906341 27.908402)
3	260	1395000	POINT (-81.663944 28.315271)
4	147	830000	POINT (-81.631344 28.188397)
5	802	4105000	POINT (-81.760947 27.949959)
6	983	5010000	POINT (-80.982155 28.42061)
7	767	3930000	POINT (-82.581895 28.118514)
8	248	1335000	POINT (-82.322752 27.957075)
9	713	3660000	POINT (-81.989692 27.788065)
10	568	2935000	POINT (-81.692902 28.359003)
11	768	3935000	POINT (-80.889977 28.475134)
12	192	1055000	POINT (-82.348994 28.994223)
13	855	4370000	POINT (-80.811588 27.808392)
14	14	165000	POINT (-82.302213 28.874652)

```
CREATE SET TABLE HURRICANE_PATH ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      ID INTEGER,
      GEOMETRY SYSUDTLIB.ST_Geometry)
UNIQUE PRIMARY INDEX ( ID );
```



	ID	GEOMETRY
1	1	LINESTRING (-76.86181 27.010891,-77.423562 27.13163,-78.087289 27.317509,-78.669794 27.442556,-79.116441 27.599977,-79.48

```
CREATE SET TABLE HURRICANE_AREA ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      ID INTEGER,
      GEOMETRY SYSUDTLIB.ST_Geometry)
UNIQUE PRIMARY INDEX ( ID );
```



	ID	GEOMETRY
1	1	POLYGON ((-77.498355302335582 26.849311571021772,-77.521025462473546 26.8549141699892,-78.173660943754285 27.03752