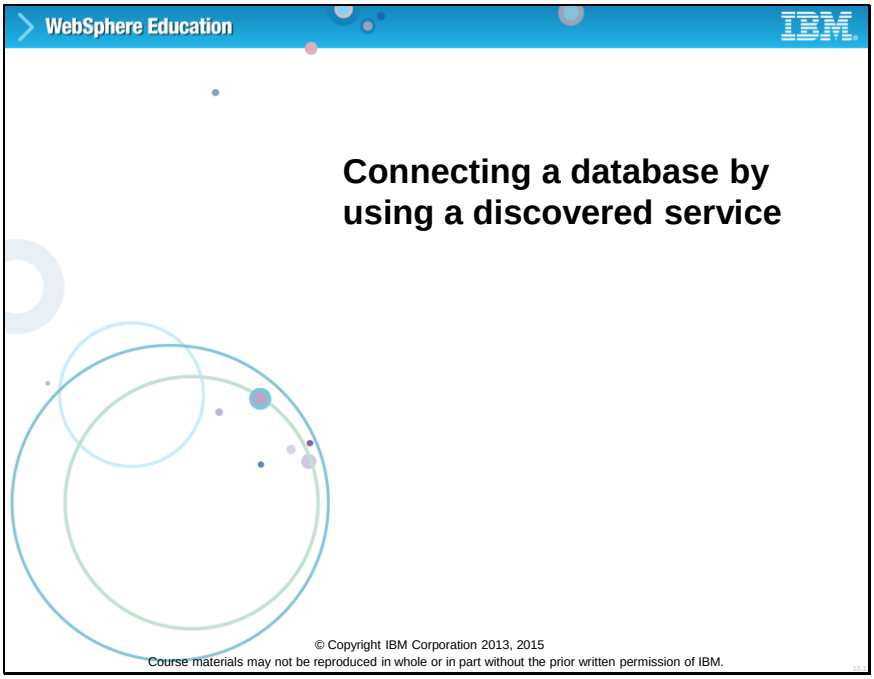
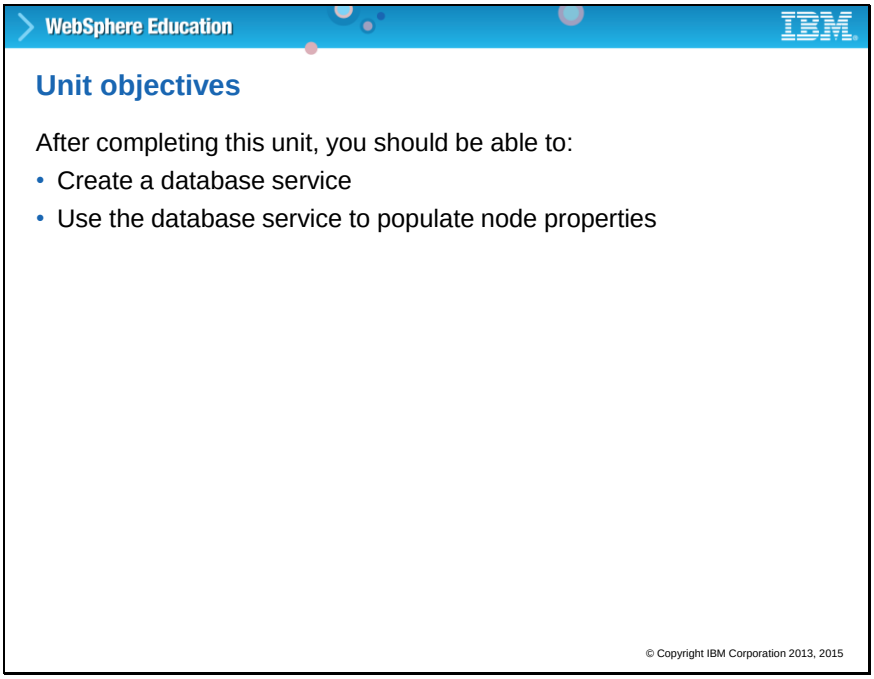
Slide 1


WebSphere Education
IBM
Connecting a database by using a discovered service
© Copyright IBM Corporation 2013, 2015
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Slide 2

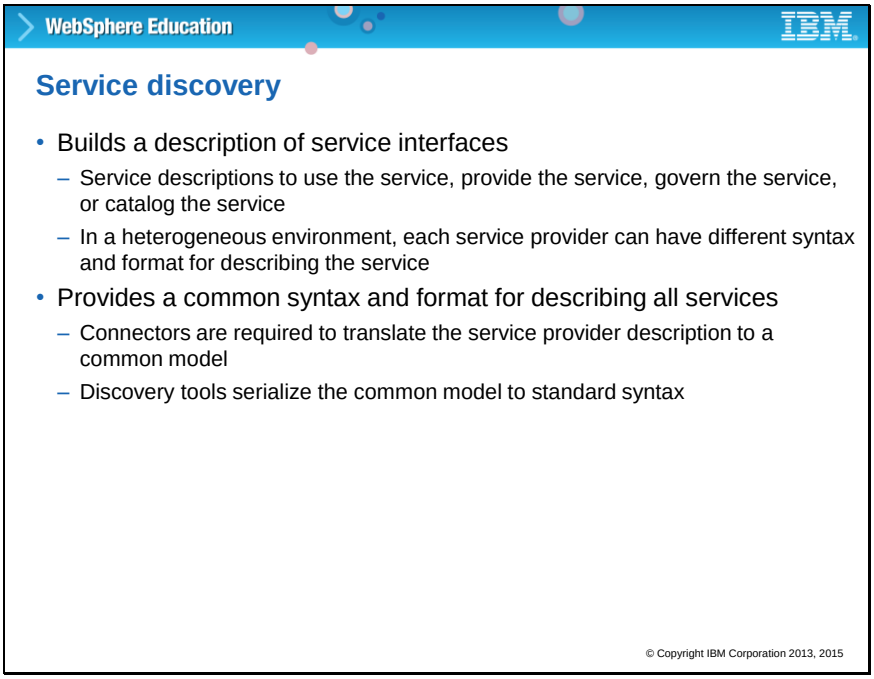


Unit objectives

An IBM Integration Bus database service defines the interaction between Integration Bus and a database, and can be used with a message flow node to automatically configure connectivity to that database. In this unit, you learn how to define and implement a database service.

After completing this unit, you should be able to:
- Create a database service
- Use the database service to populate node properties

Slide 3



Service discovery

In a heterogeneous environment, each external application has a different syntax and format for describing their service. Integration Bus needs a description of these interfaces to use, provide, govern, or catalog the service. Service discovery in Integration Bus provides you with a common experience for describing service interfaces.

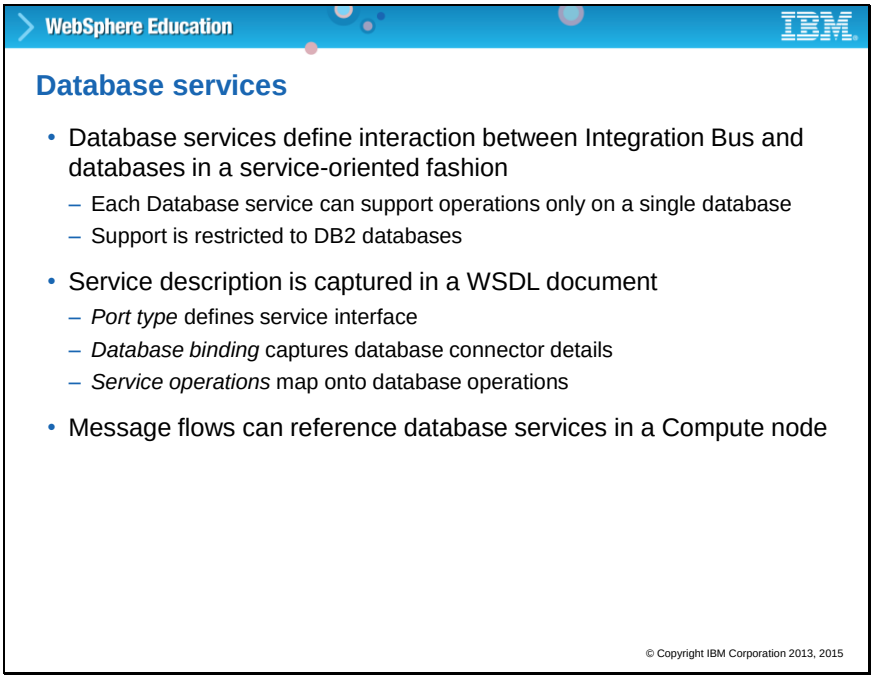
In Integration Bus, a discovered service is a representation of the external application with which your message flow interacts. Currently, Integration Bus supports service discovery for DB2 databases and MQ queues.

You can use a database service to make database operations accessible both within a message flow, and to external applications that call a message flow.

You can use an MQ service to discover artifacts from MQ queue managers. You can then use these artifacts in a message flow to configure the properties of an MQ node. You can reuse an MQ service across multiple nodes, which simplifies the MQ connectivity within your message flows.

This unit concentrates on database services. MQ services are described in Unit 10.

Slide 4



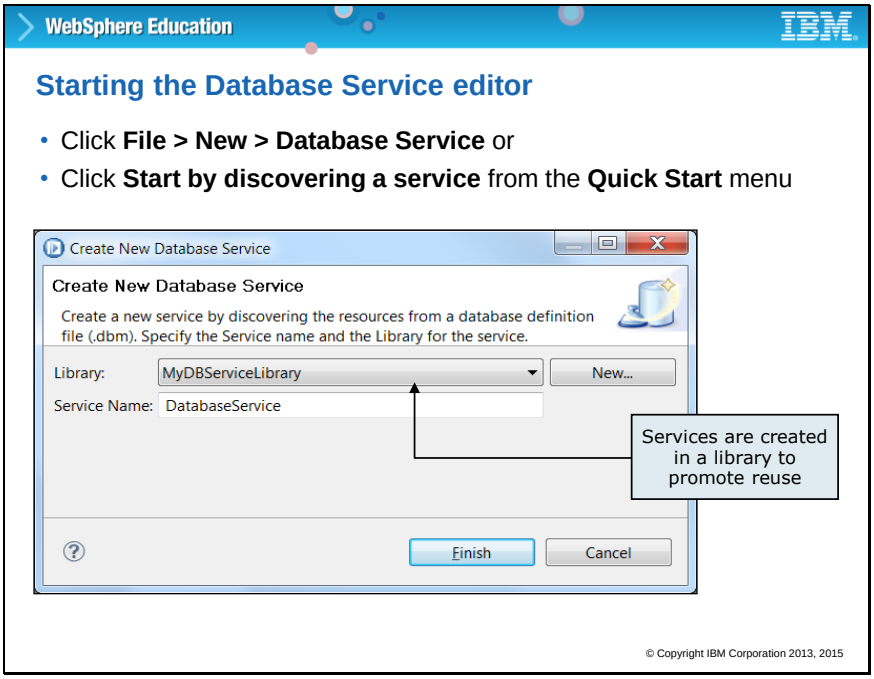
Database services

You can create a database service in the Integration Toolkit Database Service editor by combining metadata that is discovered from the database, with information supplied by the flow developer about the required database interaction.

The service description generates a WSDL file that contains:
- Operation-oriented interface definitions about the database interaction
- A database connection binding that captures the semantics of the operations on the database
- XML schema files that describe the inputs and outputs of the database operations

Currently, a database service has the following limitations:
- It can interact only with a DB2 database.
- It can be used only with a Compute node in a message flow.
- Each database service can support operations only on a single table in your database.

Slide 5



Starting the Database Service editor

The service discovery workflow consists of the following steps:

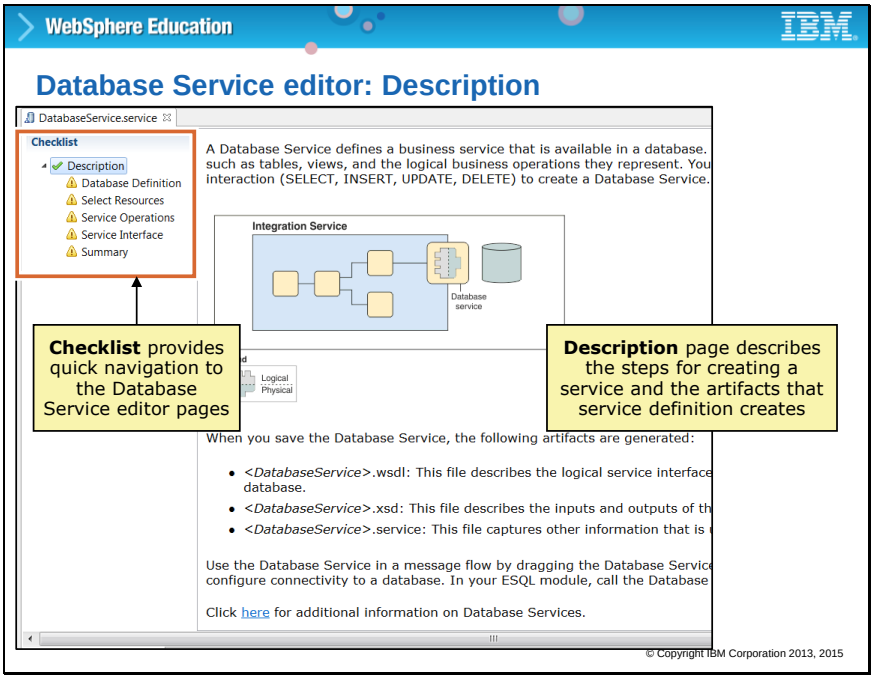
1. The developer enters connection details in the Database Service editor.
2. The Database Service editor discovers the technical assets based on the connection details.
3. The developer selects the required technical assets for elaboration.
4. The editor creates a service definition, and stores it in a catalog for later use.

You start the Database Service editor by using one of the following methods:

- Click **File > New > Database Service**.
- In the **Application Development** view, click **New > Database Service**.
- Right-click the white space of the **Application Development** view, and then click **New > Database Service**.
- Click **Start by discovering a service** from the **Quick Start** menu.

In the workspace, services are stored in a library for reuse. Be sure to provide a descriptive name for your service.

Slide 6



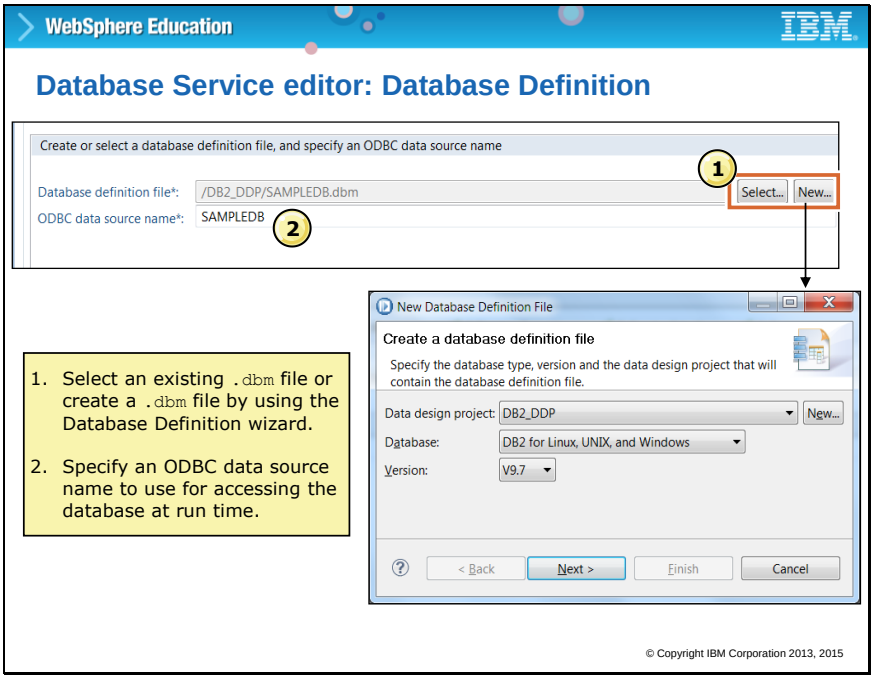
Database Service editor: Description

The Database Service editor in the Integration Toolkit has five main panels that guide you through the service definition. You click the option in the **Checklist** in the Database Service editor to go directly to a specific panel.

The Database Service editor opens on the **Description** panel. The **Description** panel contains the steps for defining a database service and a description of the artifacts that the Database Service editor generates.

To start defining the database service, click **Database Definition** in the **Checklist**.

Slide 7



Database Service editor: Database Definition

The Database Service editor uses a database definition file (.dbm file) for the database connection information. In addition to the database connection information, the database connection file also contains information about the database contents such as the database schemas, tables, and stored procedures.

When you create a database definition file, you must also specify the ODBC data source name. The database definition step assumes that an ODBC data source is already created for the database.

You learned about database definition files and ODBC in the prerequisite course *IBM Integration Bus V10 Application Development I*.

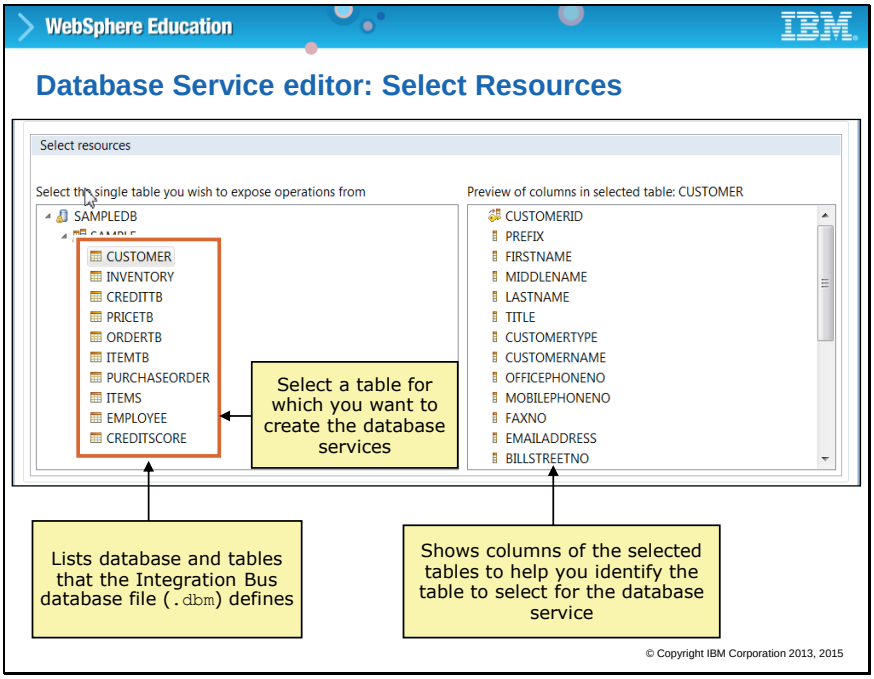
On **the Database Definition** panel, you discover the database definition by using one of the following methods:

- Specify the name of an existing database definition file (.dbm file).
- Click **New** to create a database definition file for the database.

Be sure to test the connection in the database definition file to ensure that the Integration Toolkit can connect to the database.

The next step is to select the database resources on the **Select Resources** panel.

Slide 8



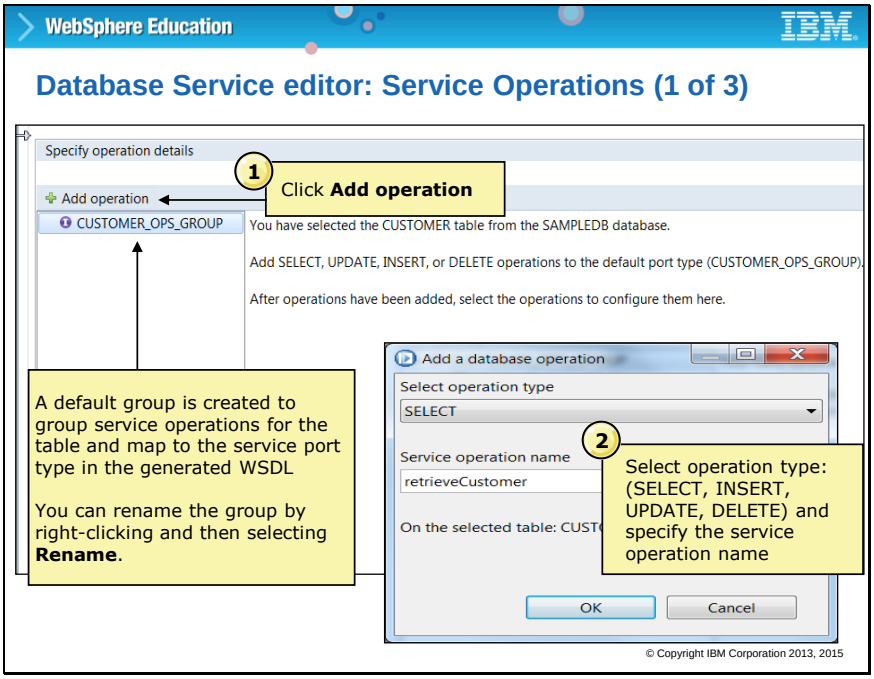
Database Service editor: Select Resources

On the **Select Resources** panel, you select the table for which you want to provide operations. Currently, each database service can support operations on a single table only.

When you select a table, the table columns are shown in the preview pane to help you to determine the table to select.

The next step is to define the database service operations on the **Service Operations** panel.

Slide 9



Database Service editor: Service Operations (1 of 3)

On the **Service Operations** panel, add the required SELECT, UPDATE, INSERT, or DELETE service operations. You can define several service operations, but they can relate to the one database table only.

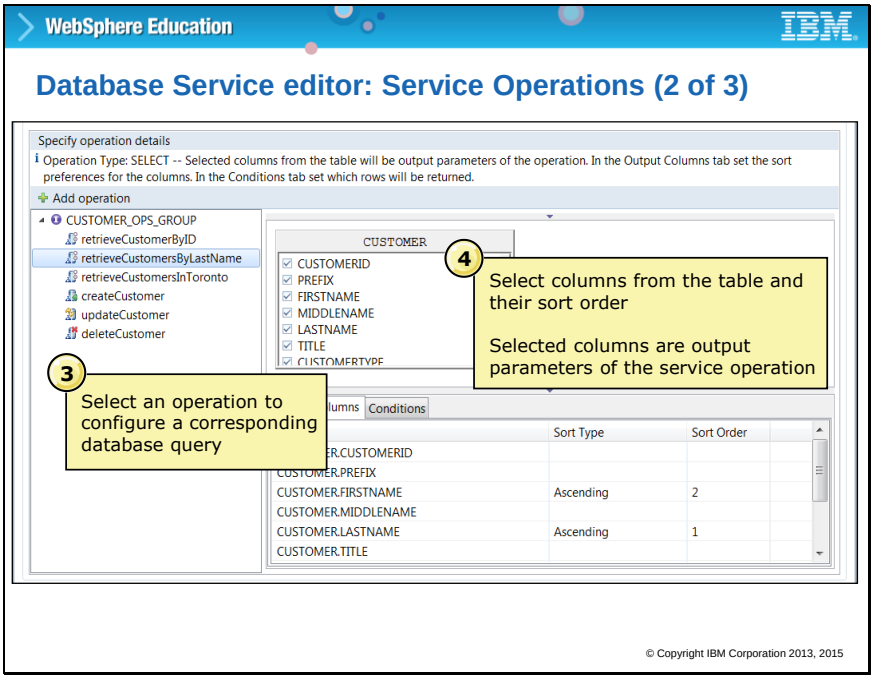
A default group is created to group the service operations for the table. By default, the group is named <table_name>_OPS_GROUP. In the example, the group is named CUSTOMER_OPS_GROUP because the CUSTOMER table was selected on the **Select Resources** panel. This group maps to the service port type in the WSDL that the Database Service editor generates. You can rename the group by right-clicking the group name and then clicking **Rename**.

To add an operation:
1. Click **Add operation**.
2. Select the operation type and then enter a descriptive service operation name.

You create a SELECT operation in the lab exercise for this unit.

The next step is to identify the table columns for the service operation.

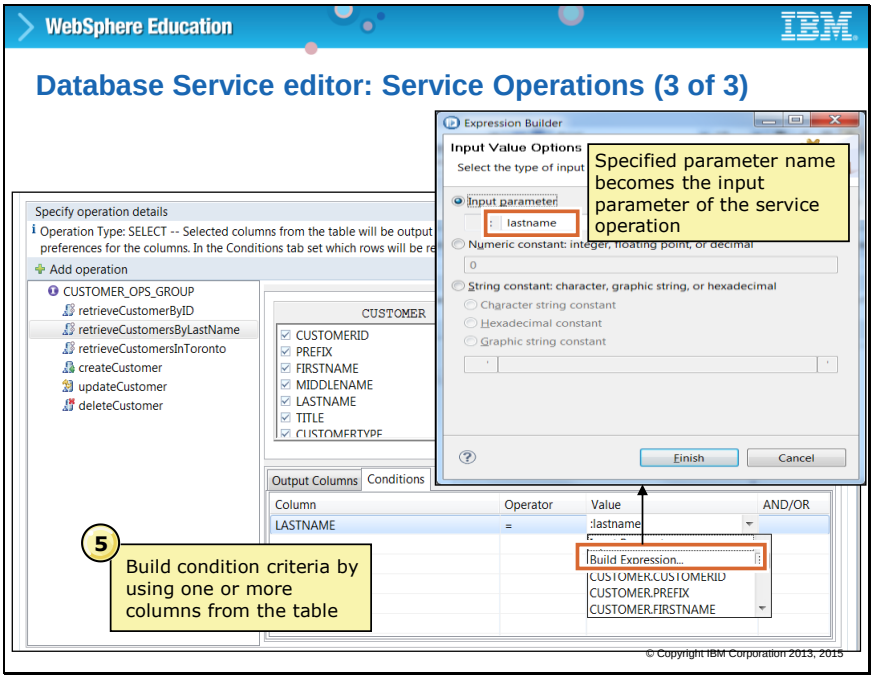
Slide 10



Database Service editor: Service Operations (2 of 3)

The next steps on the Service Operations panel are:
3. Select the operation.
4. Select the table columns that the service operation uses.

Slide 11



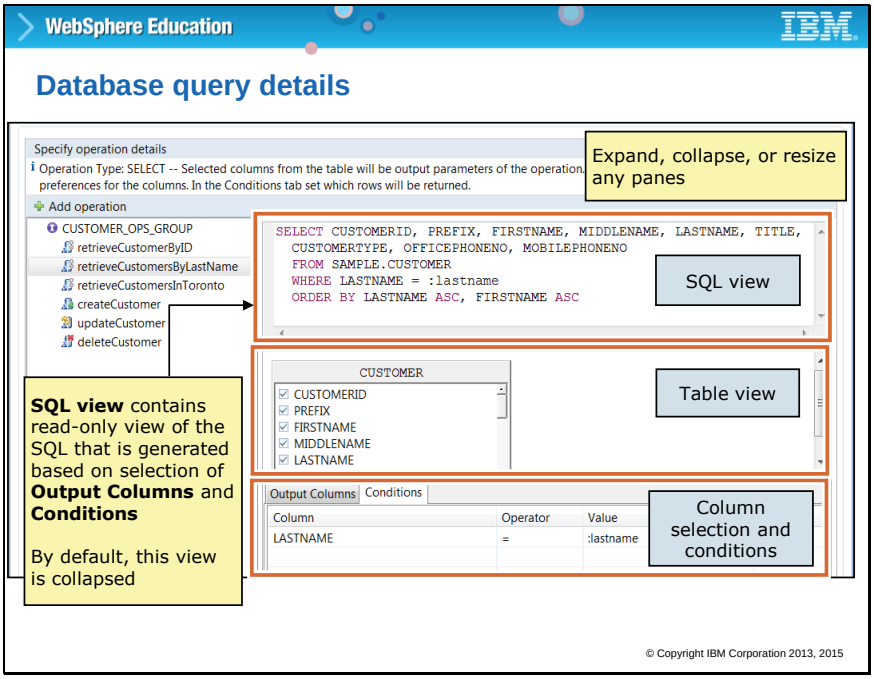
Database Service editor: Service Operations (3 of 3)

The next step is to define the operation conditions on the **Conditions** tab. For example, if the operation type is SELECT, specify the query condition to create a complete SELECT statement.

The value that you specify in the **Value** column becomes the input parameter of the service operation. When you click in the **Value** column, you have the option of selecting a column or starting the Expression Builder to help you build an expression.

You get practice defining a condition in lab exercise 5.

Slide 12



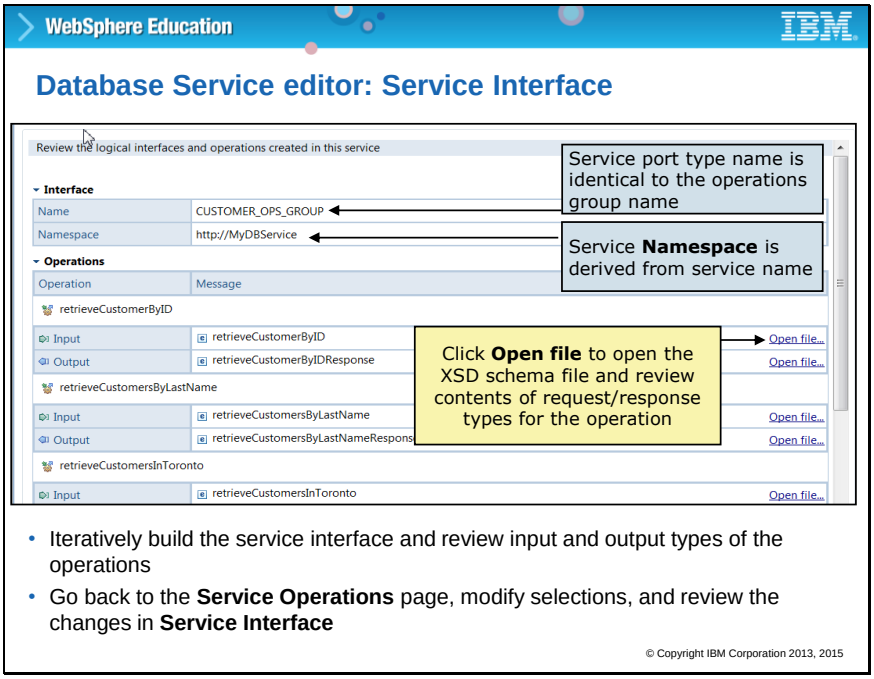
Database query details

When you define an operation, you can open an SQL view pane that shows the SQL statement that this service operation sends to the database.

You cannot change the statement in the SQL view. If the statement is not correct, verify the columns that you selected and the conditions you included.

The next step for defining a database service is to define the service interface on the **Service Interface** panel.

Slide 13



Database Service editor: Service Interface

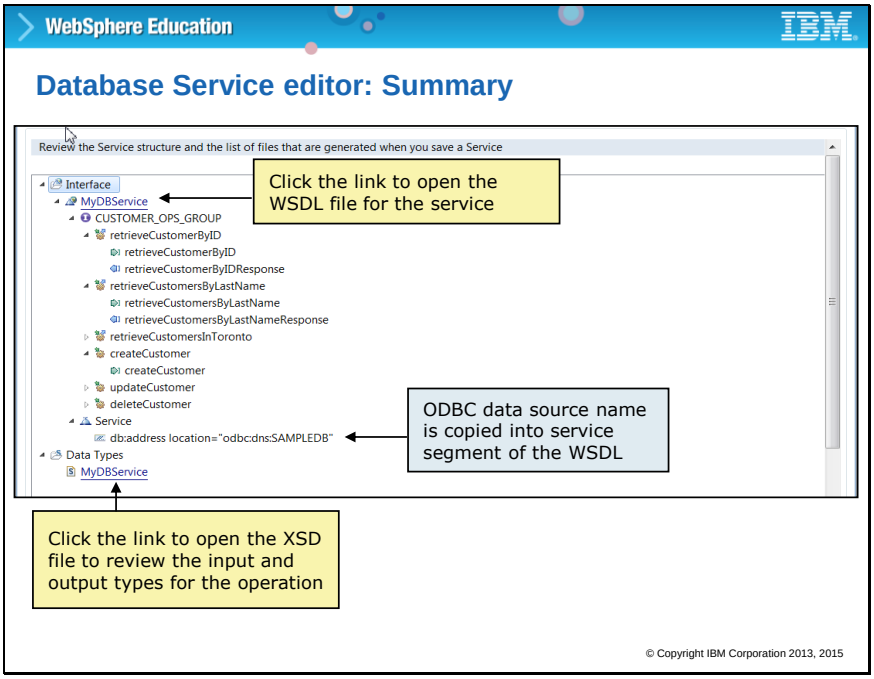
The **Service Interface** panel contains two sections: **Interface** and **Operations**.

The **Interface** section contains the group name and the namespace. The service port type name in the WSDL is the same as the interface name. The namespace is derived from the service name.

The **Operations** section contains an entry for each operation that you create on the **Service Operations** panel. For any interface, you can click the **Open file** link to open the XML schema for the operation. If the database service is not saved, clicking **Open file** prompts you to save the service file, which generates the WSDL and XSD files for service.

The final panel in the Database Service editor contains a summary of the database service.

Slide 14



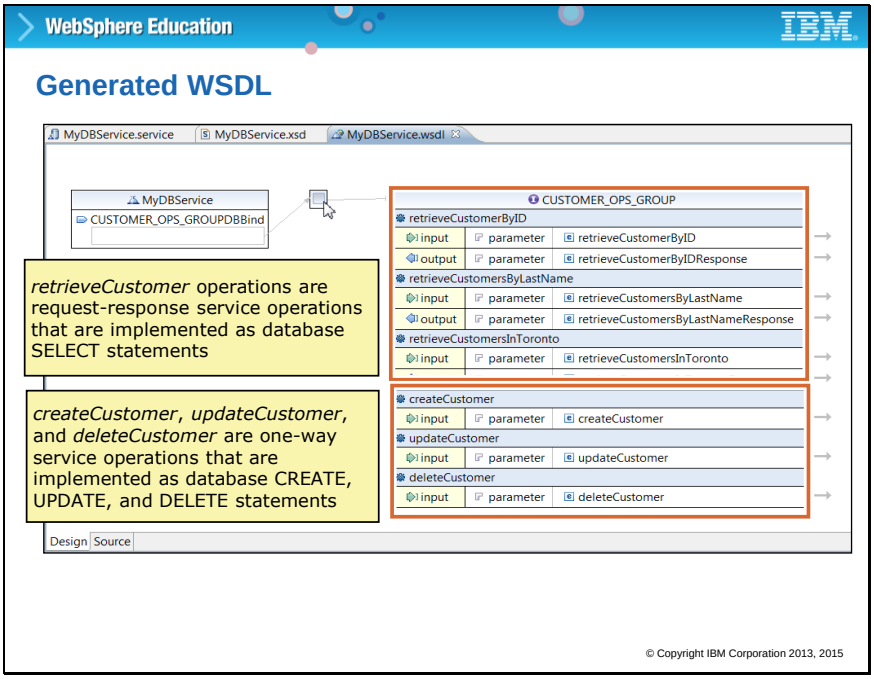
Database Service editor: Summary

On the Database Service editor **Summary** page, you can review the service interface, which includes the operations and input and output message types.

You can click the service link to open the WSDL for the service. You can also click the link under the **Data Types** folder to open the XML schema file to review the input and output types for the operation.

The next slide contains an example of a database service WSDL file.

Slide 15



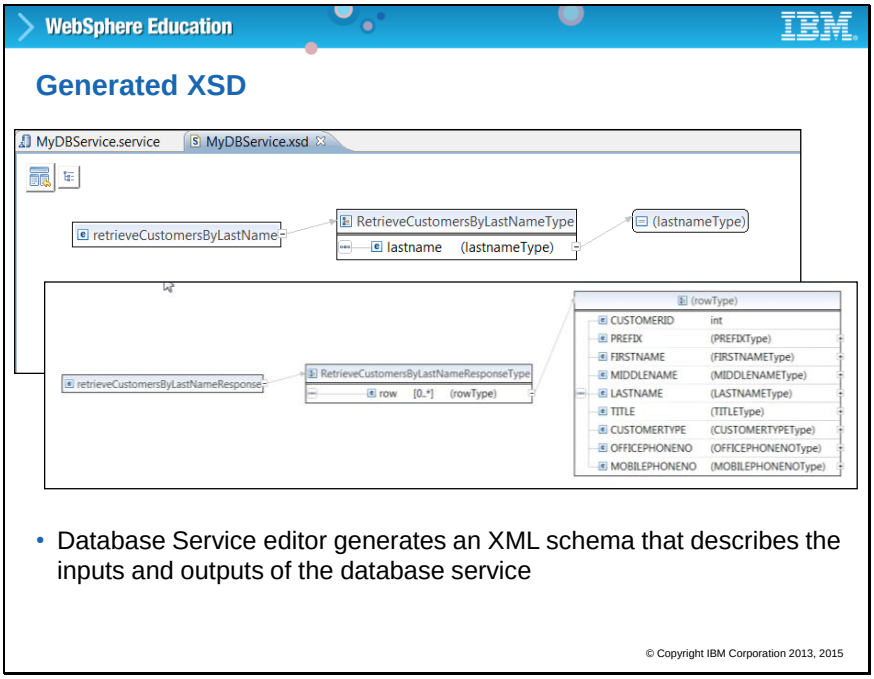
Generated WSDL

You can view the WSDL in the Integration Toolkit.

The database service WSDL in this example contains six operations:

- **retrieveCustomerByID, retrieveCustomersByLastName,** and **retrieveCustomersInToronto** are request-response service operations that are implemented as database SELECTs.
- **createCustomer** is a one-way service operation that is implemented as a database CREATE.
- **updateCustomer** is a one-way service operation that is implemented as a database UPDATE.
- **deleteCustomer** is a one-way service operation that is implemented as a database DELETE.

Slide 16

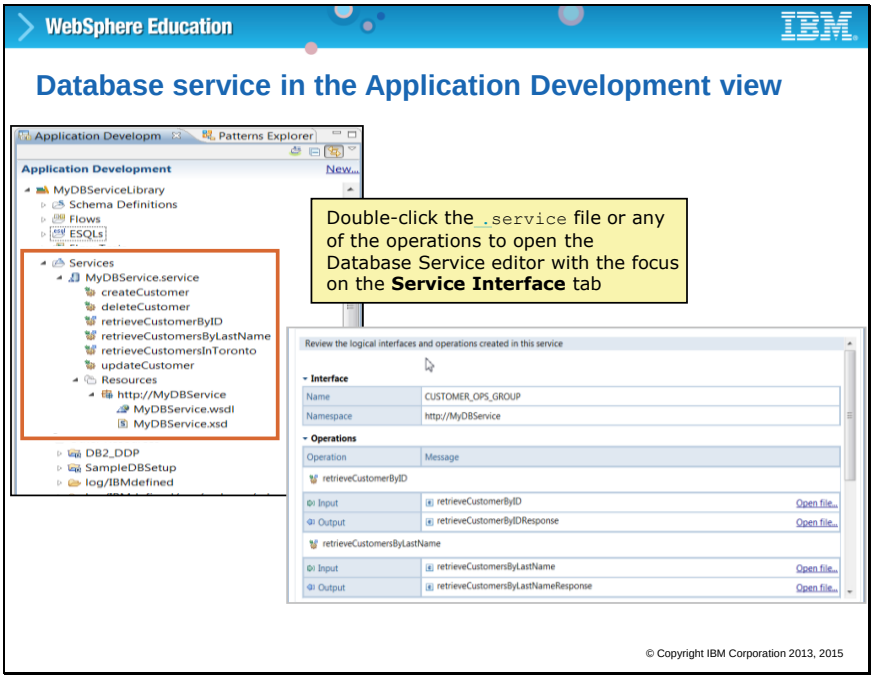


Generated XSD

The Database Service editor also generates an XML schema that describes the inputs and outputs of the database service. You can open the schema file in the Toolkit by double-clicking the file in the **Application Development** view to review the input and output types for the operation. You can also click the link under the **Data Types** folder on the Database Service editor **Summary** panel to open the XML schema file.

This example shows the inputs and outputs for the **retrieveCustomerByLastName** and the **retrieveCustomersByLastNameResponse** operations.

Slide 17



Database service in the Application Development view

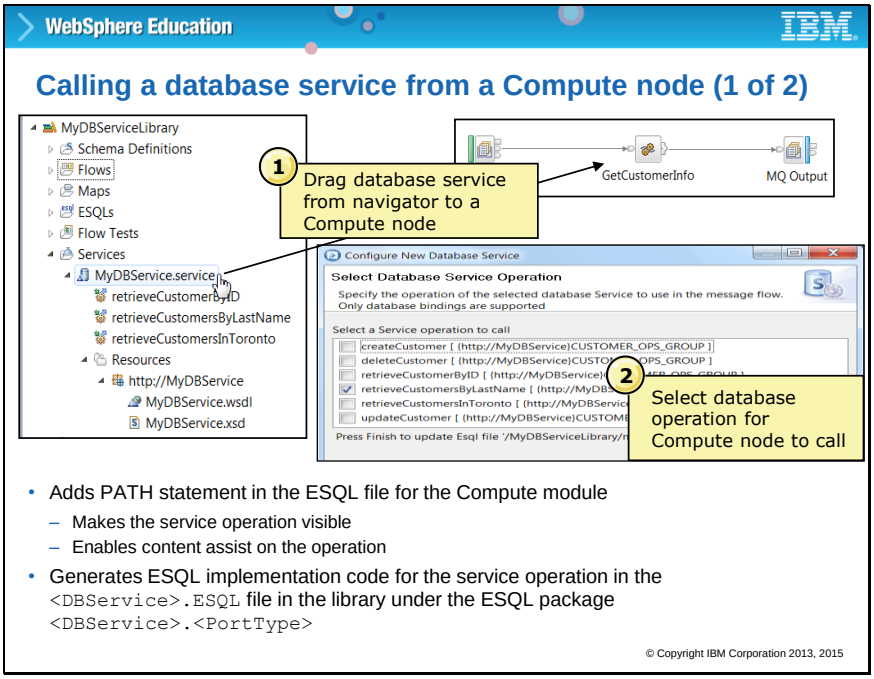
When you save the database service, the database service artifacts are added to the **Application Development** view.

A service file with a .service extension captures metadata that is used during the discovery process. The Integration Toolkit uses this file to store state information from the Database Service editor, and iteratively discovers previously discovered artifacts.

The service artifacts are added to the Application Development view under the .service file. The service artifacts include the operations, the WSDL that describes the service, and the XML schema definition file that describes the inputs and outputs of the database service.

You can open the Database Service editor and modify the service by double-clicking the .service file in the **Application Development** view.

Slide 18



Calling a database service from a Compute node (1 of 2)

After you create the database service, you can use it to create the ESQL for a Compute node in a message flow.

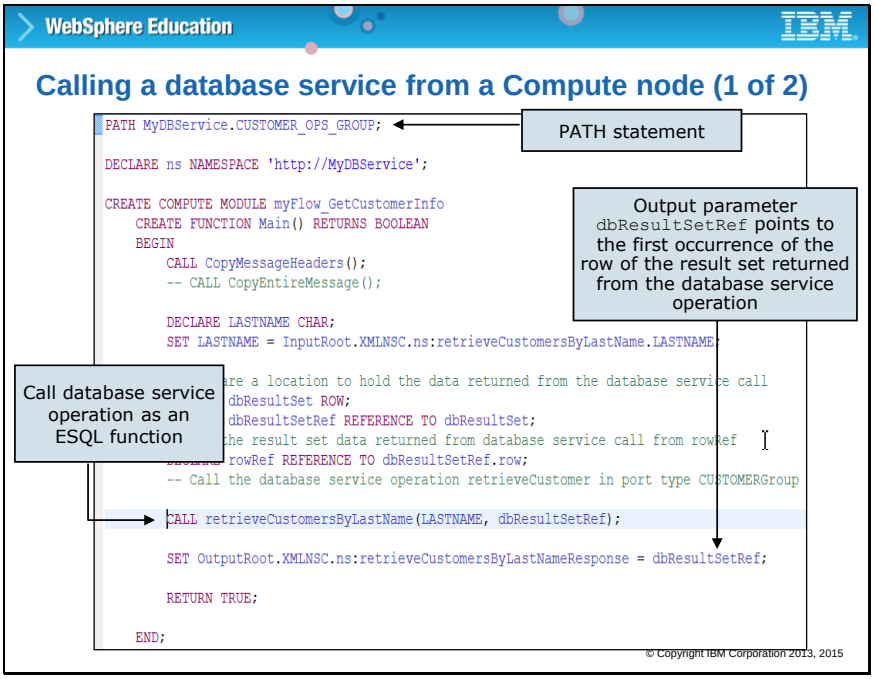
1. Drag the database service (.service file) from the **Application Development** view onto the Compute node in the Message Flow editor. The Message Flow editor also supports a menu on the Compute node for selecting a database service.
2. Select the operation from the list of operations that are defined in the database service.

A compute module that is named <*DatabaseServiceName*>.esql is generated in the broker schema that is derived from the namespace of the service and port type of the database binding. This file is in the library that contains the .service file, and contains an ESQL procedure definition for each database operation selected.

The generated Compute module contains a PATH statement so that the database service can be called.

Next, is an example of the ESQL that the database service generates.

Slide 19



Calling a database service from a Compute node (2 of 2)

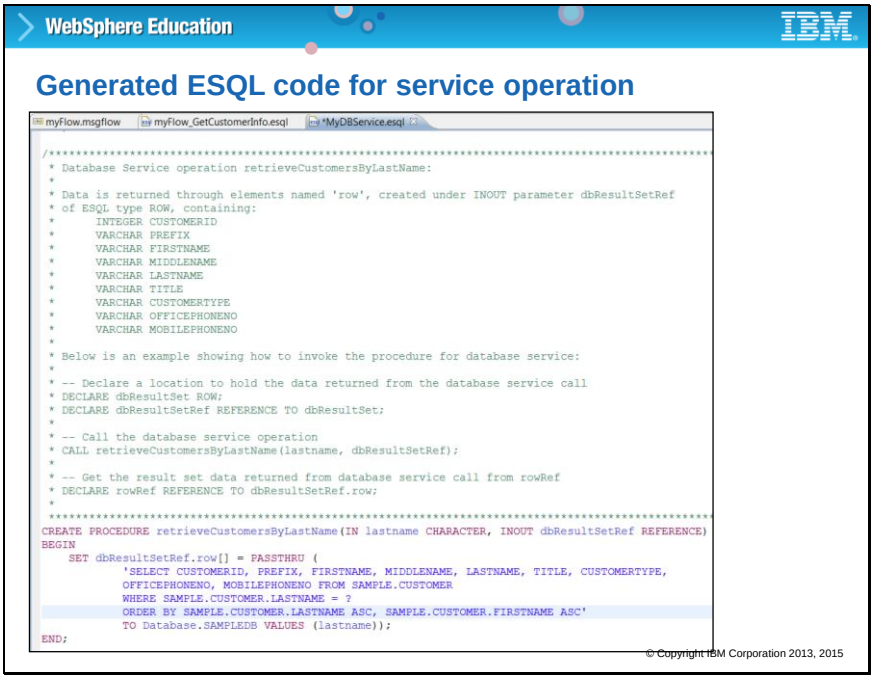
The ESQL in this example calls the database service.

The PATH statement at the top of the ESQL module identifies the database service.

The CALL statement calls the database service operation. In this example, the operation is retrieveCustomersByLastName. The CALL statement includes an input parameter (LASTNAME), and an output parameter (dbResultSetRef).

The output parameter, dbResultSetRef, points to the first occurrence of the row of the result set that is returned from the database service operation.

Slide 20



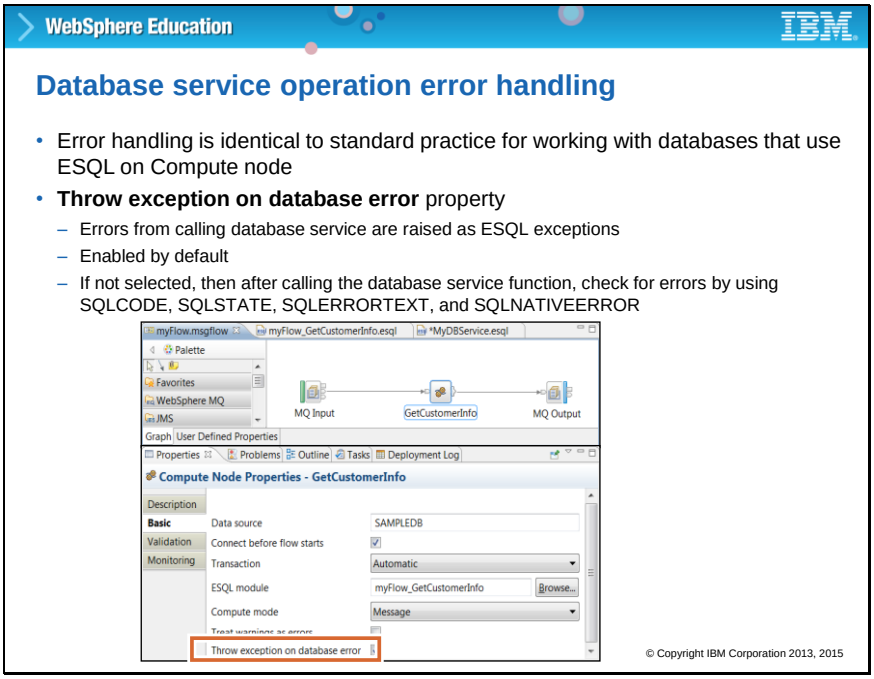
Generated ESQL code for service operation

The ESQL code also contains an ESQL procedure that is based on the operation that you select when you drop the service onto the Compute node in the message flow.

The procedure that is shown in the example is the procedure that the ESQL in the previous slide calls. In the example, an ESQL procedure is created for a service operation that is named retrieveCustomersByLastName. The service operation determines the input and output parameters for the operation.

In this example, the generated ESQL uses the ESQL PASSTHRU for the SELECT statement because it contains ORDER BY clause, which ESQL does not support.

Slide 21



Database service operation error handling

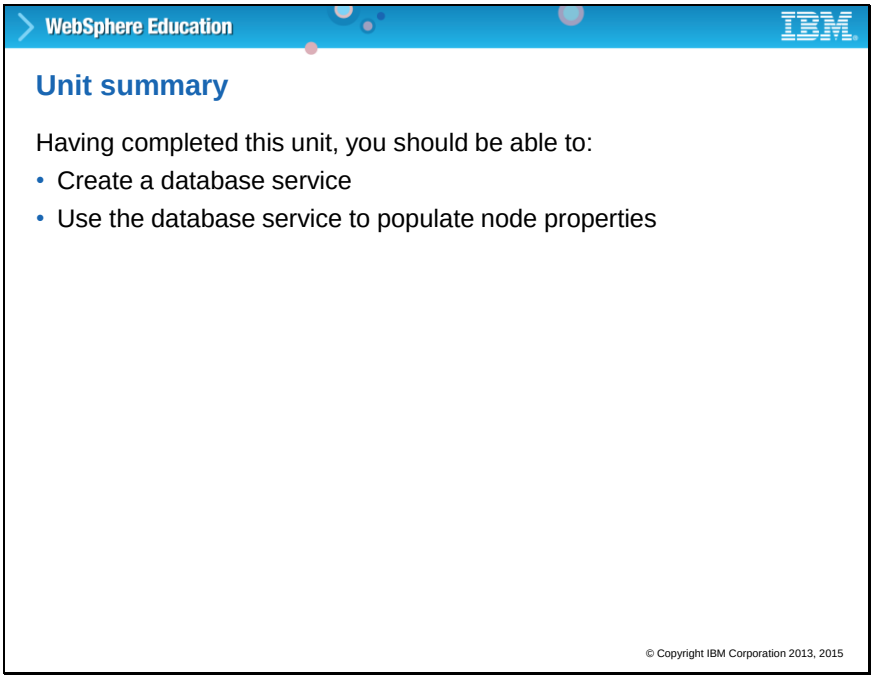
Error handling for the database service is the same as when calling a database from any other node in a message flow.

One option is to add ESQL code to handle database errors that might occur when you call the generated procedure.

Another option is to use the option to **Throw exception on database error** on the Compute node properties, which is enabled by default.

Database error handling is described in detail in the prerequisite course *IBM Integration Bus V10 Application Development I*.

Slide 22



Unit summary

An IBM Integration Bus database service defines the interaction between Integration Bus and a database, and can be used with a message flow node to automatically configure connectivity to that database. In this unit, you learned how to define and implement a database service.

Having completed this unit, you should be able to:
- Create a database service
- Use the database service to populate node properties