

**Module 6:  
Building DataSets from  
Existing Data Sources**

# Overview

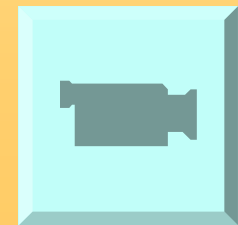
- **Configuring a DataAdapter to Retrieve Information**
- **Populating a DataSet Using a DataAdapter**
- **Configuring a DataAdapter to Update the Underlying Data Source**
- **Persisting Changes to a Data Source**
- **How to Handle Conflicts**

# Lesson: Configuring a DataAdapter to Retrieve Information

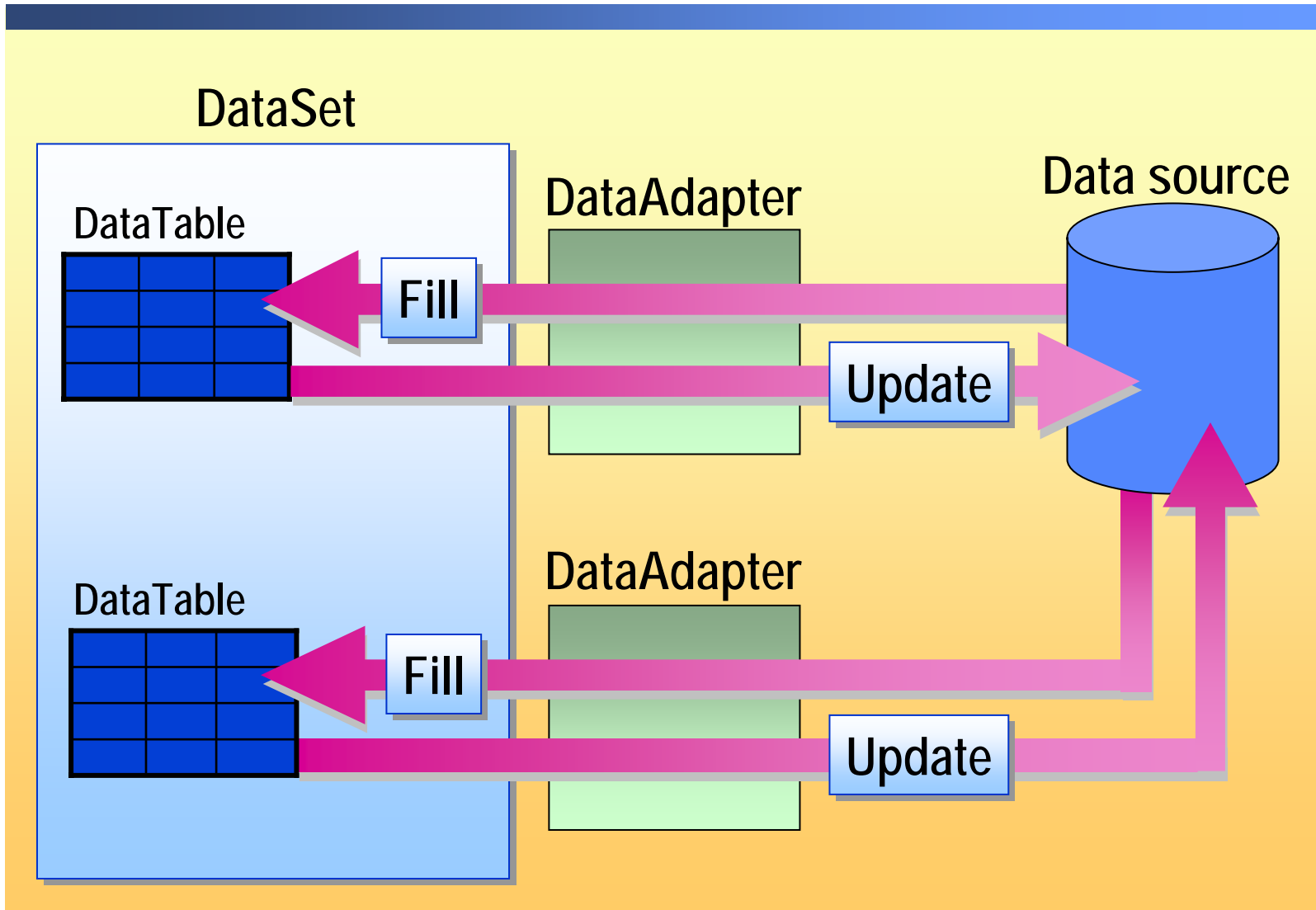
- Multimedia: Overview of Creating and Populating a DataSet
- What is a DataAdapter?
- The XxxDataAdapter Object Model
- DataAdapter Properties and Methods
- How to Create a DataAdapter That Uses a New SELECT Statement
- How to Create a DataAdapter That Uses an Existing Stored Procedure

# Multimedia: Overview of Creating and Populating a DataSet

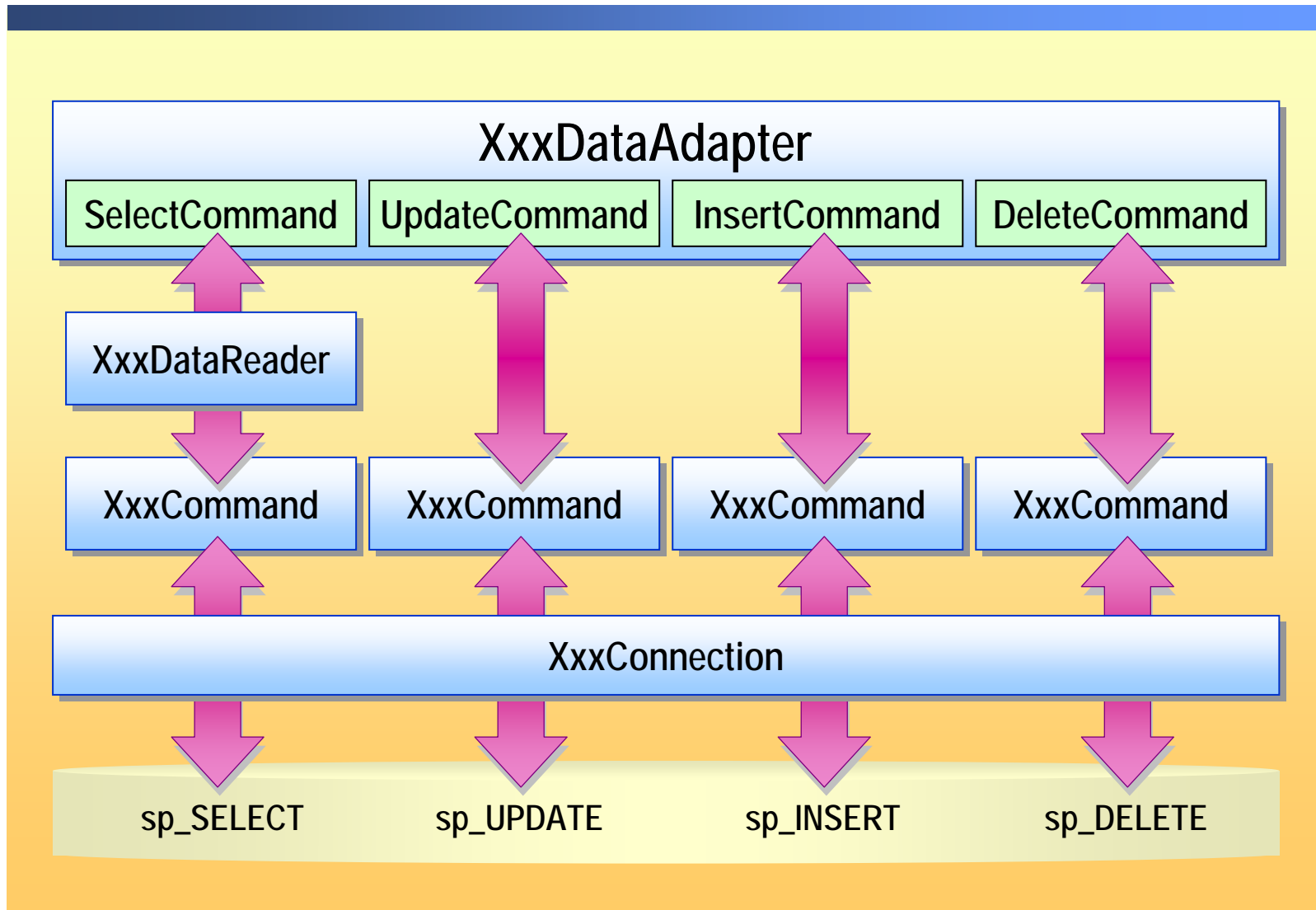
- Multimedia: Overview of Creating and Populating a DataSet



# What Is a DataAdapter?



# The XxxDataAdapter Object Model



# DataAdapter Properties and Methods

## ■ DataAdapter properties

- SelectCommand
- InsertCommand
- UpdateCommand
- DeleteCommand

## ■ Methods used by DataAdapters

- Fill
- Update

# How to Create a DataAdapter That Uses a New SELECT Statement

- You can create a DataAdapter to execute a new SELECT statement
  - Read-only data access for disconnected applications
- Two ways to create the DataAdapter
  - Use the Data Adapter Configuration Wizard
  - Write the code yourself
- You must specify
  - A new or existing connection
  - The SELECT statement for the query

# How to Create a Data Adapter That Uses an Existing Stored Procedure

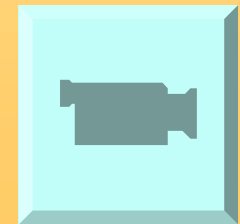
- You can create a DataAdapter programmatically to execute an existing stored procedure by:
  - Specifying a stored procedure for SelectCommand
  - Specifying stored procedures for InsertCommand, UpdateCommand, and DeleteCommand if required
- Create the DataAdapter by using the wizard, or in code
- You must specify:
  - A new or existing connection
  - The stored procedure(s)
- Practice

# Lesson: Populating a DataSet Using a DataAdapter

- **Multimedia: How the Fill Method of a DataAdapter Creates and Populates a DataTable in a DataSet**
- **How to Fill a DataSet Table by Using a DataAdapter**
- **How to Infer Additional Constraints for a DataSet**
- **How to Fill a Dataset Efficiently**
- **How to Fill a DataSet from Multiple DataAdapters**

# Multimedia: How the Fill Method of a DataAdapter Creates and Populates a DataTable in a DataSet

- How the Fill Method of a DataAdapter Creates and Populates a DataTable in a DataSet



# How to Fill a DataSet Table Using a DataAdapter

- You can fill a DataSet table by using a DataAdapter
  - Call the Fill method on the DataAdapter
- The Fill method executes the SelectCommand
  - Fills the DataSet table with the structure and content of the query result
- To optimize performance
  - `aDataSet.EnforceConstraints=False`
  - Call the BeginLoadData method on the DataTable

# How to Infer Additional Constraints for a DataSet

- You can fill a DataSet even if the schema is not known at design time
  - The DataSet schema is created at run time
- Set the MissingSchemaAction property to control how the schema is created
  - Add, AddWithKey, Error, or Ignore
- Call FillSchema to build a new DataSet schema
  - FillSchema executes SelectCommand on the DataAdapter, to determine the structure of the data

# How to Fill a Dataset Efficiently

- Define an explicit schema before you fill the DataSet
  - DataTables, DataColumnns, and DataRelations are known before the data is loaded
  - Enables the data to be loaded more efficiently
- To define an explicit DataSet schema
  - Create a typed DataSet class:

```
dsCustomers.Customers.BeginLoadData()  
daCustomers.Fill(dsCustomers.Customers)  
dsCustomers.Customers.EndLoadData()  
DataGrid1.DataSource =  
dsCustomers.Customers.DefaultView
```
  - Or, create the DataTables, DataColumnns, and DataRelations programmatically

# How to Fill a DataSet from Multiple DataAdapters

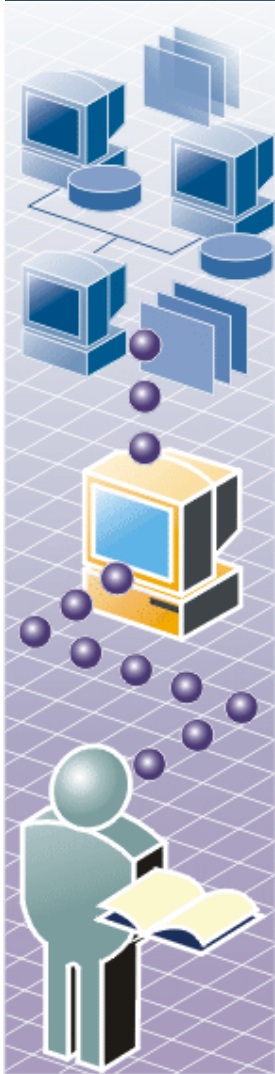
- You can use multiple DataAdapters to fill a DataSet
  - Each DataAdapter fills a separate table in the DataSet
- Call the Fill method on each DataAdapter
  - Specify the table to fill in the DataSet

## ■ Visual Basic example

```
daCustomers.Fill(dsCustomerOrders.Customers)
daOrders.Fill(dsCustomerOrders.Orders)
DataGrid1.DataSource = dsCustomerOrders.Customers
```

## ■ Practice

# Lab 6.1: Retrieving Data into a Disconnected Application



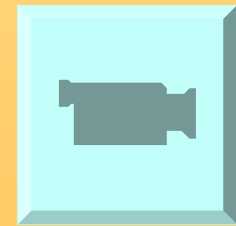
- Exercise 1: Reviewing the Application
- Exercise 2: Building a DataSet to Hold Employees and Application Settings
- Exercise 3: Loading and Displaying Employee Information
- Exercise 4: Specifying and Using a Different Server Name

# Lesson: Configuring a DataAdapter to Update the Underlying Data Source

- Multimedia: How a DataSet Tracks Changes
- How Does a DataSet Track Changes?
- What Are the Data Modification Commands?
- How to Set the Data Modification Commands Using the Data Adapter Configuration Wizard

# Multimedia: How a DataSet Tracks Changes

- How the DataSet Tracks Changes



# How Does a DataSet Track Changes?

- Each DataRow has a RowState property
  - Indicates the status of each row
  - Added, Deleted, Detached, Modified, Unchanged
- The DataSet maintains two copies of data for each row
  - Original version
  - Current version

# What Are the Data Modification Commands?

- A SqlDataAdapter or OleDbDataAdapter object has command properties that are themselves command objects that you can use to modify data at the data source
  - InsertCommand
  - UpdateCommand
  - DeleteCommand
- Syntax – essentially the same for both Sql and OleDb DataAdapters and for the series of command objects

```
public SqlCommand InsertCommand {get; set;}
```

# How to Set the Data Modification Commands Using the Data Adapter Configuration Wizard

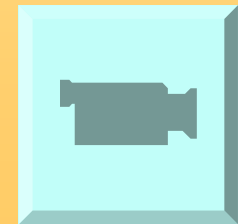
- You can create data modification commands by using the Data Adapter Configuration Wizard
- The wizard can generate the commands in three different ways:
  - By using SQL statements
  - By creating new stored procedures
  - By using existing stored procedures
- Practice

# Lesson: Persisting Changes to a Data Source

- **Multimedia: How the DataAdapter's Update Method Modifies the Underlying Data Source**
- **When to Use the GetChanges Method of a DataSet Object**
- **How to Merge Changes into the DataSet**
- **How to Update a Data Source by Using a DataSet**
- **How to Accept Changes into the DataSet**

# Multimedia: How the DataAdapter's Update Method Modifies the Underlying Data Source

- How the DataAdapter's Update Method Modifies the Underlying Data Source



# When to Use the GetChanges Method of a DataSet Object

- Use the GetChanges method when you need to give the changes to another class for use by another object
- Code example

```
If dsCustomers.HasChanges(DataRowState.Modified) Then
    Dim dsTemp As DataSet
    dsTemp =
        dsCustomers.GetChanges(DataRowState.Modified)
    DataGrid1.DataSource = dsTemp.Tables(0).DefaultView
End If
```

- Use the GetChanges method to get a copy of a DataSet that contains all changes made to the DataSet
  - Since it was loaded, or
  - Since the last time the AcceptChanges method was called.

# How to Merge Changes into the DataSet

- Use the Merge method to merge two DataSets – an original, and one containing only the changes to the original

- Code example

```
aDataSet.Merge(anotherDataSet)
```

- The two merged DataSets should have similar schemas

# How to Update a Data Source by Using a DataSet

- The Update method of a DataAdapter object calls the appropriate statement for each changed row in a specific DataTable:
  - INSERT
  - UPDATE
  - DELETE
- Code example

```
aDataAdapter.Update(aDataSet, aDataTable)
```

# How to Accept Changes into the DataSet

- The `AcceptChanges` method of the `DataSet` commits all changes made to a specific `DataSet` since it was last loaded, or since `AcceptChanges` was called

- Code example

```
aDataSet.AcceptChanges()
```

- You can invoke `AcceptChanges` for an entire `DataSet` or for a each `DataRow` in each `DataTable`
- Practice

# Lesson: How to Handle Conflicts

- What Conflicts Can Occur?
- How to Detect Conflicts
- How to Resolve Conflicts

# What Conflicts Can Occur?

- **Disconnected applications use optimistic concurrency**
  - Releases database locks between data operations
- **Data conflicts can occur when you update the database**
  - Another application or service might have already changed the data
- **Examples**
  - Deleting a previously deleted row
  - Changing a previously changed column
- **Practice**

# How to Detect Conflicts

- The Data Adapter Configuration Wizard can generate SQL statements to detect conflicts
- When you update the database:
  - Data modification commands compare the current data in the database against your original values
  - Any discrepancies cause a conflict error

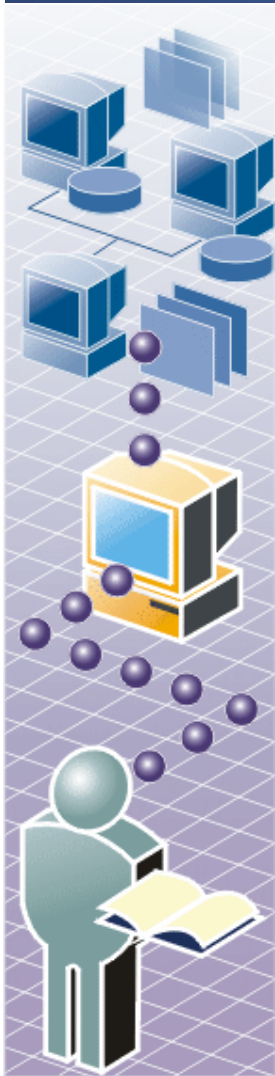
# How to Resolve Conflicts

- Use the HasErrors property to test for errors
  - Test a DataSet, DataTable, or DataRow
- Choose one of these strategies to resolve conflicts:
  - "Last in wins"
  - Retain conflicting rows in your DataSet so you can update the database again later
  - Reject conflicting rows and revert to the original values in your DataSet
  - Reject conflicting rows and reload the latest data from the database
- Practice

# Review

- **Configuring a DataAdapter to Retrieve Information**
- **Populating a DataSet Using a DataAdapter**
- **Configuring a DataAdapter to Update the Underlying Data Source**
- **Persisting Changes to a Data Source**
- **How to Handle Conflicts**

# Lab 6.2: Retrieving and Updating Customers and Orders Data



- Exercise 1: Preparing to Load and Update Multiple Tables in the Database
- Exercise 2: Filling a DataSet by Using Multiple Data Adapters
- Exercise 3: Updating the Central Database