



COMPSCI 280 S1 2011 Enterprise Software Development

DataSet Updates



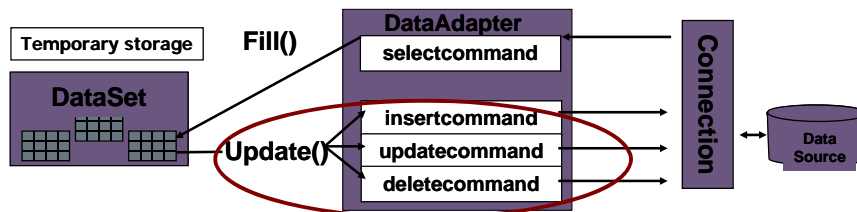
Agenda & Reading

- ▶ **Agenda:**
 - ▶ Introduction to DataSet Updates
 - ▶ Updating, Inserting and Deleting Records
 - ▶ Identifying and Returning Changed Rows
- ▶ **Recommended Reading:**
 - ▶ Microsoft ADO.NET 2.0 step by step, Rebecca M. Riordan
 - ▶ Chapter 10: Editing and Updating Data
 - ▶ MSDN Library
 - ▶ <http://msdn.microsoft.com/library/en-us/vbcon/html/vbconDataUpdatesInVisualStudioNET.asp>
 - ▶ <http://msdn.microsoft.com/library/en-us/vbcon/html/vbconADONETDataAdapters.asp>
 - ▶ [http://msdn2.microsoft.com/en-us/library/7zt3ycf2\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/7zt3ycf2(VS.80).aspx)
- ▶ **Hands-on Lab:**
 - ▶ Lecture24Lab: DataSet Updates



DataSet Updates

- ▶ **Two-Stage Updates**
 - ▶ Changes have been made to the **in-memory** copy of the data **ONLY**
 - ▶ Need to transfer all of the changes made to it back to the original data source



◆ The process of updating the DataSet does not also write the changes through to an underlying data source; you must explicitly perform this by calling the Update method of the same DataAdapter that you used to populate the DataSet, although you can also use different adapters.



Save Changes

- ▶ **Call the update method of the TableAdapter**
 - ▶ Information about the changes in a dataset is maintained in two ways
 - ▶ by **flagging** the row that indicates whether it has been changed (**RowState**), and
 - RowState: unchanged, Detached, Added, Modified and Deleted
 - ▶ by keeping multiple copies of a record (**DataRowVersion**)
 - Current, Original, Proposed and Default
 - ▶ Automatically loops through the records in a table to determine what type of update is needed and then performs the required SQL
 - ▶ If modified, calls configured UPDATE command
 - ▶ If added, calls configured INSERT command
 - ▶ If deleted, calls configured DELETE command
 - ▶ Return Values
 - ▶ Number of records affected by Update
 - ▶ Zero, if operation fails
- ▶ Call AcceptChanges method to update the row state values

◆ As you make changes to records in a dataset by updating, inserting, and deleting records, the dataset maintains original and current versions of the records. In addition, each row's RowState property is set to indicate whether the records are in their original state or if they have been updated, inserted or deleted.

◆ When an update is sent to the data source, the new information for the database is in the current version of a record, and information from the original version is used to locate the record to update.



Editing

Example: UpdateDemo

- AcceptChanges:
 - Original values -> change to the current values
 - RowState: Modified -> Unchanged
- RejectChanges
 - Return all columns to their original values
 - RowState: Modified -> Unchanged

```

DataRow dr = northwindDataSet.Customers.Rows[0];
dr.BeginEdit();
dr["ContactName"] = "Michael";
dr.EndEdit();
accept
...
northwindDataSet.AcceptChanges(); //update rowversion & rowstates

```

Unchanged, c:Ana Trujillo, o:Ana Trujillo, d:Ana Trujillo

Modified, c:Michael, o:Ana Trujillo, d:Michael

Unchanged, c:Michael, o:Michael, d:Michael

northwindDataSet.RejectChanges();

reject

- The value of the DataRow.RowState property for that row changes from Unchanged to Modified. The current version changes from "Ana Trujillo" to "Michael".
- When your application calls the Update method, it inspects each row in turn.
- For that row, the Update method automatically invokes the **UPDATE** statement because the value of the RowState property is Modified.
- The SQL statement includes a **WHERE** clause indicating that the target of the UPDATE statement is the row whose CustomerID = ANATR (CustomerID is the primary key). The information for the WHERE clause is derived from the original version of the record, in case values required to identify the row have been changed.
- The SQL statement includes the **SET** clause, to set the new values of the modified columns. The value is from the current version: Michael.



Inserting

- AcceptChanges:
 - Original values -> change to the current values
 - RowState: Added -> Unchanged
- RejectChanges
 - RowState: Added -> Detached

```

DataRow drNew = northwindDataSet.Customers.NewRow();
drNew["ContactName"] = "Dick Smith";
...
dsNorthwind1.Customers.Rows.Add(drNew);
accept
...
northwindDataSet.AcceptChanges(); //update rowversion & rowstates

```

Added, c:Dick Smith, d: Dick Smith

Detached, p: Dick Smith, d: Dick Smith

No Original version

northwindDataSet.RejectChanges();

reject

Detached

Unchanged, c: Dick Smith, o: Dick Smith, d: Dick Smith

- The value of the DataRow.RowState property for that row changes from Detached to Added.
- When you insert a new row into a table, there is no original version, only a current version.
- For that row, the Update method automatically invokes the **INSERT** statement because the value of the RowState property is Added.
- Value is read from the current version for each column if it exists.
- Default: The row the default version for the current DataRowState. For a DataRowState value of Added, Modified or Current, the default version is Current. For a DataRowState of Deleted, the version is Original. For a DataRowState value of Detached, the version is Proposed.



Deleting

- AcceptChanges:
 - RowState: Deleted -> Detached
- RejectChanges
 - Return all columns to their original values
 - RowState: Deleted -> Unchanged

```

DataRow dr = northwindDataSet.Customers.Rows[0];
dr.Delete();
accept
...
northwindDataSet.AcceptChanges(); //update rowversion & rowstates

```

Deleted, o:ALFKI

No Current version

Unchanged, c:ALFKI, o:ALFKI, d:ALFKI

northwindDataSet.RejectChanges();

reject

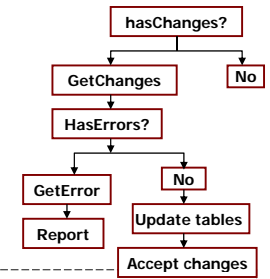
Detached

- The value of the DataRow.RowState property for that row changes from Unchanged to Deleted.
- When you delete a row, there is an original version, but no current version.
- For that row, the Update method automatically invokes the **DELETE** statement because the value of the RowState property is Deleted.
- The SQL statement includes a **WHERE** clause indicating that the target of the DELETE statement is the row whose CustomerID = ANATR (CustomerID is the primary key). The information for the WHERE clause is derived from the original version of the record, in case values required to identify the row have been deleted.



Identifying and Returning Changed Rows

- HasChanges
 - Returns a value indicating whether the DataSet has changes, including new, deleted or modified rows
- GetChanges
 - Gets a copy of a DataSet that contains all changes made to the DataSet
 - An empty argument to retrieve all changed rows, OR
 - specify type of changes using enumeration values
 - DataRowState.Deleted, Added, Modified ...
- HasErrors
 - Checks for errors in the newly created DataSet
- GetError
 - Gets an array of DataRow objects that contain errors.



- Before performing an update to the data source, you may want to examine specific rows.
- The GetChanges method of a dataset or data table returns a new dataset or data table that contains only records that have been changed. If you want to get only specific records – for example, only new records or only changed records – you can pass as a parameter a row state value.
- If you find the dataset contains errors, iterates through the collections of tables, then iterates the rows to print out the error information.



Example

- Iterate through the collections of tables, get an array of DataRow objects that contain error, and loop through the array to print out the row error

```

if (northwindDataSet.HasChanges()){
    NorthwindDataSet dsChanged=(NorthwindDataSet) northwindDataSet.GetChanges();
    if (dsChanged.HasErrors) { //error
        foreach (DataTable dt in northwindDataSet.Tables)
        {
            DataRow[] errorRows = dt.GetErrors();
            foreach (DataRow dr in errorRows)
            {
                foreach (DataColumn dc in dr.GetColumnsInError())
                {
                    txtLog.AppendText(dr.GetColumnError(dc) + Environment.NewLine);
                }
            }
        }

        txtLog.AppendText("Save failed." + Environment.NewLine);
    }
    else {
        customersTableAdapter.Update(northwindDataSet);
    }
}

```

Return an array of DataRow

Print the column error

Call AcceptChanges automatically

Adapter.AcceptChangesDuringUpdate = true

9

COMPSCI280

Handout24



TableAdapters - Methods

Example: TableAdaptersDemo

- Fill
 - Populates the TableAdapter's associated data table with the results of the TableAdapter's SELECT command.

```
ProductsTableAdapter.Fill(northwindDataSet.Products);
```

- Update

```

Validate();
customersBindingSource.EndEdit();
customersTableAdapter.Update(northwindDataSet.Customers);

```

- Sends changes back to the database.
 - The Validate method Verifies the value of the control losing focus by causing the Validating and Validated events to occur, in that order.
 - The EndEdit method of the BindingSource object ends all edit operations.

- GetData

- Returns a new DataTable filled with data.

```

NorthwindDataSet.CustomersDataTable newCustomersTable =
customersTableAdapter.GetData();

```

10

COMPSCI280

Handout24