



COMPSCI 280 S1 2011 Enterprise Software Development

ADO.NET
Advanced DataSet Updates



Agenda & Reading

- ▶ **Agenda:**
 - ▶ Updating Related Tables
 - ▶ Referential Integrity
 - ▶ Creating Foreign Key Constraint
 - ▶ Updating Problems
 - ▶ Order of Updating Related Tables
 - ▶ Form Closing Problem
 - ▶ Saving Flowchart
- ▶ **Recommended Reading:**
 - ▶ MSDN Library
 - ▶ <http://msdn2.microsoft.com/en-us/library/4esb49b4.aspx>
 - ▶ <http://msdn.microsoft.com/library/en-us/vbcon/html/vbtskperformingupdates.asp>
 - ▶ <http://www.it-faq.pl/mskb/313/483.HTM>
- ▶ **Hands-on Lab:**
 - ▶ Lecture26Lab: Updating data



Updating Related Tables

- ▶ The **Update** method issues the proper **INSERT, DELETE, and UPDATE** SQL commands for a single table


```
CustomersTableAdapter.Update(northwindDataSet.Customers);
```
- ▶ When updating multiple tables (with relation between tables)
 - ▶ Update related tables individually separately
 - ▶ You may encounter some problems if the data source itself enforcing relational integrity rules
 - ▶ Problem 1:A record being created in a child (detail) table without a corresponding record in the parent (master) table
 - ▶ Problem 2:A record being deleted in a parent (master) table when corresponding records exist in a child (detail) table
 - ▶ Referential integrity is a system of rules that ensure relationships between rows in related tables are valid and that you do not accidentally delete or change related data.
 - ▶ When referential integrity is enforced, you must observe the following rules:
 - ▶ You cannot **enter** a value in the foreign-key column of the related table if that value does not exist in the primary key of the related table.
 - ▶ You cannot **delete** a row from a primary key table if rows matching it exist in a related table.
 - ▶ You cannot **change** a primary key value in the primary key table if that row has related rows.



Referential Integrity

- ▶ To enforce referential integrity
 - ▶ Conditions:
 - ▶ The matching column from the primary table is a primary key or has a unique constraint.
 - ▶ The related columns in the foreign table have the same data type and size.
 - ▶ Create the Foreign Key Constraint
 - ▶ The constraint maintains referential integrity between a parent and child table
 - ▶ When you delete a value that is used in one or more related tables, a ForeignKeyConstraint determines whether the values in the related tables are also deleted, or set to null values, or set to default values, or whether no action occurs.

Action	Description
Cascade	Deletes or updates related rows.
SetNull	Sets values in related rows to DBNull.
SetDefault	Sets values in related rows to the DefaultValue.
None	No action is taken, but an exception is raised.

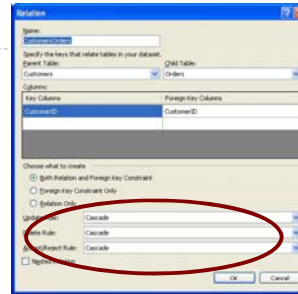
- ◆ You can specify referential integrity rules that are applied at three points: 1) when a parent record is updated (Update Rule), 2) when a parent record is deleted (Delete Rule), and 3) when a change is accepted or rejected (Accept/Reject Rule).
- ◆ Cascade: The change (delete or update) made to the parent record is made in related records in the child table as well
- ◆ SetNull: Child records are not deleted, but the foreign key in the child records is set to DBNull, that is no relationship to parent records.
- ◆ SetDefault: The foreign key in the related child records is set to its default value.



Creating constraint

- ▶ To create a foreign key constraint at design time
 - ▶ Edit the xsd file
 - ▶ Open the EditRelation Dialog box
 - ▶ Set up the relationship between tables
 - ▶ Set Update, Delete, Accept/Reject Rule to Cascade
- ▶ To create a foreign key constraint at run time
 - ▶ Create a new ForeignKeyConstraint object
 - ▶ Add the object to the Constraints Collection of the child DataTable
 - ▶ Set EnforceConstraints property to TRUE

Example: UpdateTablesDemo



```

ForeignKeyConstraint custOrderFK = new ForeignKeyConstraint("CustOrderFK",
DsNorthwind1.Customers.Columns["CustomerID"],
DsNorthwind1.Orders.Columns["CustomerID"]);
custOrderFK.DeleteRule = Rule.Cascade;
custOrderFK.UpdateRule = Rule.Cascade;
custOrderFK.AcceptRejectRule = Rule.Cascade;

//Add the constraint and set EnforceConstraints to true
DsNorthwind1.Orders.Constraints.Add(custOrderFK);
DsNorthwind1.EnforceConstraints = true;

```

Add it to the Child table



Problems

- ▶ Problem 1:
 - ▶ To add a new order for a new customer
 - ▶ Solution: Order of updating:
 - Add a new Customer first, (CustomerID, names....)
 - Then add a new order (OrderID, CustomerID, OrderedDate...)
- ▶ Problem 2:
 - ▶ Deleting an existing customer while related orders still exist in the Orders table
 - ▶ Solution: Order of updating:
 - ▶ Delete all orders from the Orders table
 - ▶ Delete the existing customer

CustomerID	FirstName	LastName
ALFKI	Maria Anders	Alfreds Futterkiste
ANA	Ana Trujillo	Ana Trujillo

OrderID	CustomerID	OrderDate
10248	ALFKI	04-Jul-1996
10249	TRADH	05-Jul-1996

What to do for this record? This CustomerID doesn't exist in the customers table anymore.

- ▶ A common scenario is that you have added both parent and related child records to a dataset – for example, a new customer record and one or more related order records. If the data source itself is enforcing relational integrity rules, it will raise errors if you send the new child record to the data source before the parent record has been created.
- ▶ Conversely, if you delete related records in the dataset, the data source is likely to raise an error because referential integrity rules will prevent you from deleting a parent record while related child records still exist.



Order of Updating Related Tables

Example: AdvancedUpdateDemo

- ▶ Updating related tables
 - ▶ MUST update in the following order:
 - ▶ Child Deletes
 - ▶ All Parent Updates
 - (Inserts, Updates, Deletes)
 - ▶ Child Inserts & Updates
 - ▶ To update related Tables
 - ▶ End Edits
 - ▶ Get Changes

```

OrdersBindingSource.EndEdit();
CustomersBindingSource.EndEdit();

NorthwindDataSet.OrdersDataTable deletedOrders= null, newOrders=null, modifiedOrders=null;

deletedOrders = (NorthwindDataSet.OrdersDataTable)
northwindDataSet.Orders.GetChanges(DataRowState.Deleted);
newOrders = (NorthwindDataSet.OrdersDataTable)
northwindDataSet.Orders.GetChanges(DataRowState.Added);
modifiedOrders = (NorthwindDataSet.OrdersDataTable)
northwindDataSet.Orders.GetChanges(DataRowState.Modified);

```

▶ If your dataset contains multiple tables, you have to update them individually by calling the Update method of each DataAdapter separately. If the tables have a parent-child relationship, it is likely that you will have to send updates to the data source in a particular order in order to reduce the chance of violating referential integrity constraints.



Order of Updating Related Tables

- Child Deletes


```
if (deletedOrders != null)
ordersTableAdapter.Update(deletedOrders);
```
- All Parent Updates :(Inserts, Updates, Deletes)

```
customersTableAdapter.Update(northwindDataSet.Customers);
```
- Child Inserts & Updates

```
if (newOrders != null)
ordersTableAdapter.Update(newOrders);
if (modifiedOrders != null)
ordersTableAdapter.Update(modifiedOrders);
```
- Dispose tables

```
if (deletedOrders != null)
deletedOrders.Dispose();
if (newOrders != null)
newOrders.Dispose();
if (modifiedOrders != null)
modifiedOrders.Dispose();
```

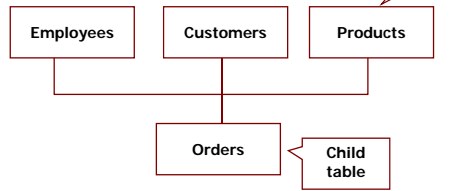
▶ The above example shows how to update a data source with a dataset that contains related tables. The Update method will be called for each subset of rows. Note: You should implement the above code within a Try...Catch block. If update errors occur, the suggested course of action is to branch off and resolve them. Then the dataset commits the changes by using the AcceptChanges method. Finally dispose of the temporary DataTables to release the resources.



More Examples in updating DataSets

▶ Example 1:

- ▶ Customers, Employees, Products & Orders



- ▶ Order Deletes
- ▶ Customers All
- ▶ Employees All
- ▶ Products All
- ▶ Orders Inserts/Updates

◆ Example 2:

- A, B & C

- ◆ C Deletes
- ◆ B Deletes
- ◆ A Deletes/Inserts/Updates
- ◆ B Inserts/Updates
- ◆ C Inserts/Updates

