Disconnected Application
Data Access Architecture

Traditional Data Access Architecture

1. makes a connection
2. passes some query and read result from DB
3. passes another query and read result from DB
4. disconnects from the DB system

ADO.NET Disconnected Data Access Architecture

1. makes a connection
2. passes some query and get result back
3. disconnects from the DB system
4. makes connection again
5. passes another query and get result back
6. disconnects from the DB system
Disconnected Data Access Architecture of ADO.NET

- ADO.NET introduces the concept of disconnected data architecture
- ADO.NET solves this problem (traditional data access) by managing a local buffer of persistent data called dataset
- Your application automatically connects to the database server when it needs to pass some query and then disconnects immediately after getting the result back and storing it in dataset
- This design of ADO.NET is called disconnected data architecture and is very much similar to the connectionless services of http over the internet
ADO.NET Architecture

.NET Framework Data Provider
- Connection
- Transaction
- Command
- Parameters
- DataReader

DataAdapter
- SelectCommand
- InsertCommand
- UpdateCommand
- DeleteCommand

DataSet
- DataTable
- DataRowCollection
- DataColumnCollection
- ConstraintCollection
- DataRelationCollection

Database

XML
The ADO.NET DataSet contains DataTableCollection and their DataRelationCollection. It represents a collection of data retrieved from the Data Source. The Dataset can work with the data it contain, without knowing the source of the data coming from. That is, the Dataset can work with a disconnected mode from its Data Source.
SqlDataAdapter

- An SqlDataAdapter object works as a mediator between a Dataset and a Database.
- The main functions performed by DataAdapter are:
  1. Populate the dataset by fetching data from database, using the 'Fill' method
  2. Update changes made to the dataset back to the database, using the 'Update' method
SqlCommandBuilder

- To generate INSERT, UPDATE, or DELETE statements, the SqlCommandBuilder uses the SelectCommand property to retrieve a required set of metadata automatically.
- If you change the SelectCommand after the metadata has been retrieved, such as after the first update, you should call the RefreshSchema method to update the metadata.
Managing Concurrency

What happens when two users try to update the same row ??

- When two or more users retrieve the data in the same row of a database table at the same time, it is called concurrency. Because ADO.NET uses a disconnected data architecture, the database management system can’t prevent this from happening.
- By default, ADO.NET uses optimistic concurrency (the program checks to see whether the database row that’s going to be updated or deleted has been changed since it was retrieved.)
Dataset Changes

Data Source

Fill DataSet

Original Database Record
FirstName="James"
LastName="Wilson"
DataRowVersion=Original/Current

DataRowState Unchanged

Stage 1
Original Version Persists
Updates the DataSet
User Edits Data

FirstName="Jim"
LastName="Wilson"
DataRowVersion=Current

DataRowState Modified

WHERE Statement in SQL
uses original column values to locate the correct record

Stage 2
Update Data Source with Modified DataSet

SET Statement in SQL
uses current column values to update the record

Data Source (same as above)
In ADO.NET, you must add your own code for submitting database updates to the DataAdapter object. There are three ways of doing this:

1. You can supply your own updating logic.
2. You can use the Data Adapter Configuration Wizard to generate the updating logic.
3. You can use the CommandBuilder object to generate the updating logic.
SqlDataAdapter Syntax

' The DataSet that holds the data
Dim ds As New DataSet(DATASET_NAME)

' Create the SqlDataAdapter
Dim da As SqlDataAdapter
da = New SqlDataAdapter(SELECT_STRING, CONNECT_OBJECT)

' Create the SqlCommandBuilder
Dim cb As SqlCommandBuilder = New SqlCommandBuilder(da)

' Fill the DataSet
da.Fill(ds)
```
Dim dt As DataTable = ds.Tables("Authors")

' First process deletes.
da.Update(dt.Select(Nothing, Nothing, _
        DataViewRowState. Deleted))

' Next process updates.
da.Update(dt.Select(Nothing, Nothing, _
        DataViewRowState. ModifiedCurrent))

' Finally process inserts.
da.Update(dt.Select(Nothing, Nothing, _
        DataViewRowState.Added))
```
After your dataset is populated with data, you will typically perform some kind of manipulation of the data before sending it back to the data source or to another process or application.
Updating Existing Records in a Dataset

Dim rowUbah As DataRow = dtPenjualan.Rows(0)

rowUbah.BeginEdit()
rowUbah("Tanggal") = _
   dtpPenjualan.Value.ToShortDateString
rowUbah("Kode_Pelanggaran") = tbKodePelanggaran.Text
rowUbah.EndEdit()
Dim rowBaru As DataRow = dtPenjualan.NewRow

rowBaru("Tanggal") = _
    dtpPenjualan.Value.ToShortDateString
rowBaru("Kode_Pelanggan")=tbKodePelanggan.Text
dtPenjualan.Rows.Add(rowBaru)
Deleting Records in a Dataset

Dim rowHapus As DataRow = dtPenjualan.Rows(0)
dtPenjualan.Rows.Remove(rowHapus)

Or

dtPenjualan.Rows(0).Delete()
Committing Changes in the Dataset

- You can commit the pending changes to the dataset by calling the `AcceptChanges` method.
- Typically, `AcceptChanges` would be called at the following times in your application.

```vbnet
'commit
dtPenjualan.AcceptChanges()

'rollback
dtPenjualan.RejectChanges()
```
Daftar Pustaka

http://www.programmersheaven.com/2/FAQ-ADONET-Disconnected-Data-Access
http://vb.net-informations.com/dataset/ado.net-dataset.htm
http://www.akadia.com/services/dotnet_rowstate.html