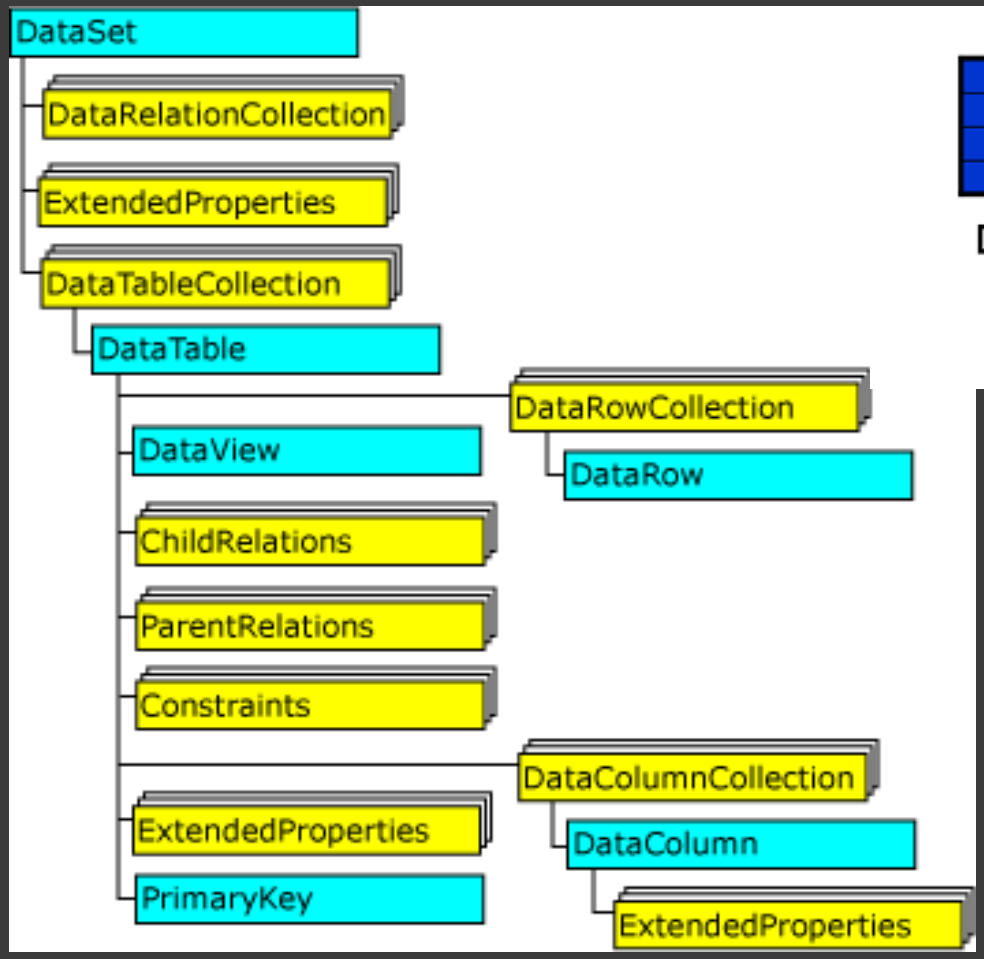
ADO.NET – Objekt DataSet (offline scenář)

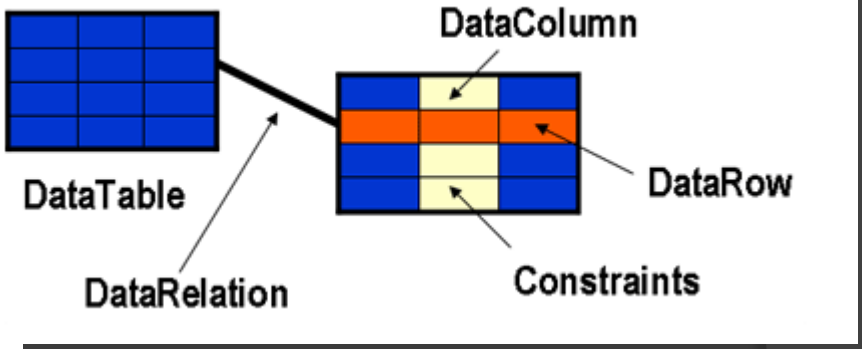
Centrum pro virtuální a moderní metody a formy vzdělávání na Obchodní akademii T.G. Masaryka, Kostelec nad Orlicí

DataSet

- Jednoduchá relační DB v paměti počítače (je kopie dat vytažených a nakešovaných na klienta)
- Není připojen k DB – umožňuje práci s daty nezávisle na stálém připojení
- Obsahuje tabulky a vztahy mezi nimi
- Dvě kolekce:
  - ***DataTableCollection*** (kolekce objektu DataTable, Columns, Rows, Constrains)
  - ***DataRelationCollection*** (kolekce objektu DataRelation)
- //Vytvoření DataSet
  DataSet dset = new DataSet();

Objekt DataSet





- Existují 2 typy **DataSetů**:
  1. ***Typed***
  2. ***Untyped***

Typed DataSet

- **1. Typed dataset** je potomkem třídy **DataSet** a používá pro svůj popis XML Schema file (.xsd soubor) který slouží k vygenerování nové třídy.

- **Typed dataset** je jednoduché číst.Podporuje IntelliSense ve Visual Studio Code Editor.
  V čase kompilace, obsahuje datovou kontrolu na přiřazované hodnoty členům DataSet objektu.
  Má několik výhod
  Příklad:
  Následující kód zpřístupní CustomerID sloupec v prvním řádku tabulky Customers

  string str;
  str=dset.Customers[0].CustomerID;

Untyped DataSet

- 2. **Untyped** dataset není definován schématem, místo toho, musíte přidávat tabulky, sloupce, a ostatní elementy ručně,
  a to buď nastavováním vlastností vytváření aplikace nebo jejich přidáváním při jejím běhu.
  například: ve scénáři, kdy dopředu nevíte, která struktura Vašeho programu je v interakci s komponentou, která vrací DataSet

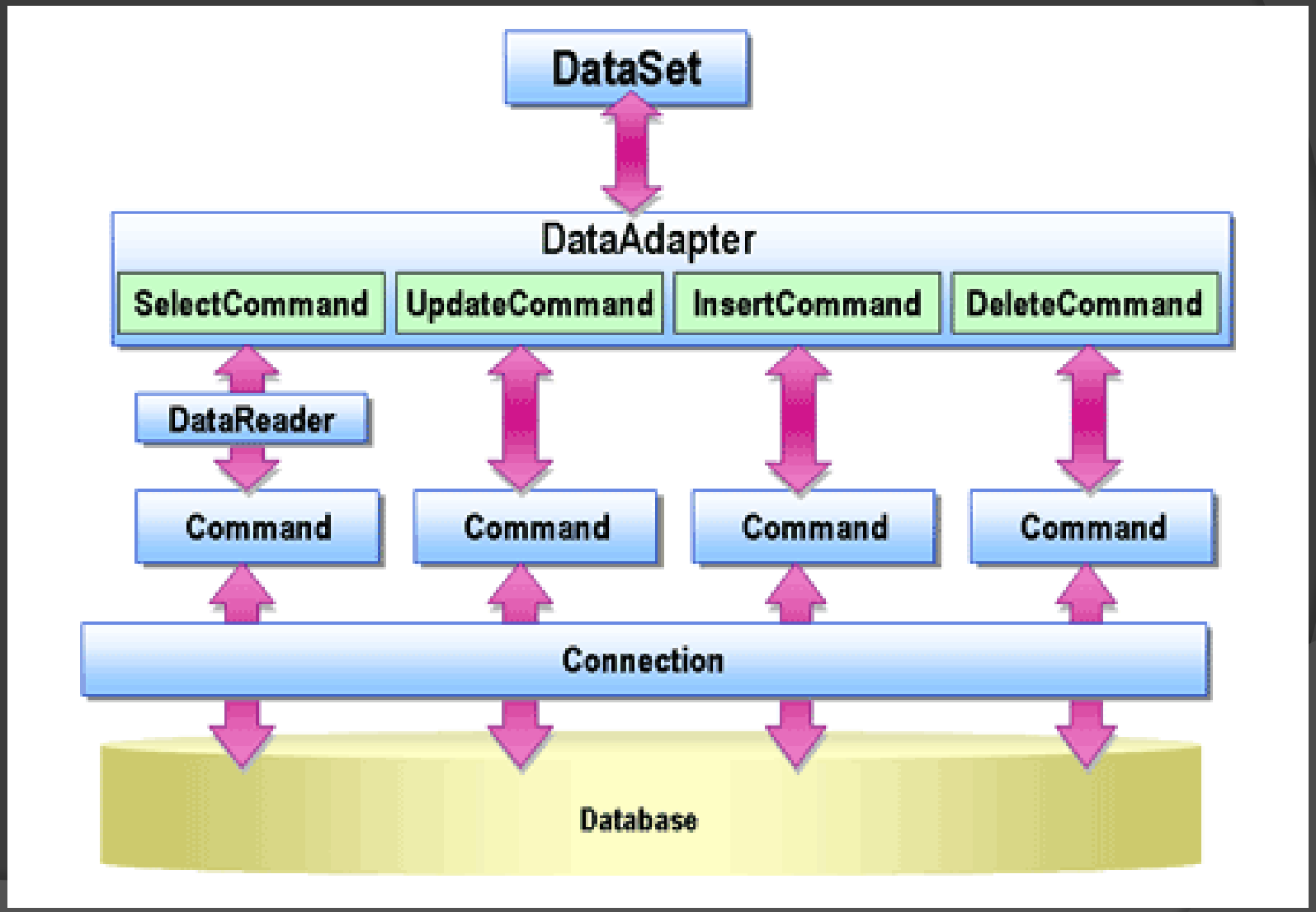
  ekvivalent výše uvedeného kódu pro Untyped dataset je:

```
string str;
str=(string)dset.Tables["Customers"].Row[0].["CustomerID"];
```

Plnění Datasetu

- **Dataset je container; proto musí být naplněn daty.**
  K naplnění je možno použít několik možností:
  - **použít metodu *Fill***
  - **vytvořit DataRow objekty a přidat je do kolekce řádků tabulky(pouze za běhu programu).**
  - **čtením XML dokumentu nebo streamu do datasetu**
  - **Spojit (zkopírovat) obsah jiného datasetu**

Funkce datového adaptéru



Objekt DataAdapter

- je jako most spojující DB úložiště a objekt Connection s ADO.NET DataSet objektem .
- Specifikuje, která data přesunout do a z objektu DataAdapter, obvykle formou SQL dotazu, nebo formou uložených procedur.
- poskytuje čtyři vlastnosti, které umožňují kontrolovat jak budou updaty aplikovány na server:
  SelectCommand, *UpdateCommand*, *InsertCommand*, a *DeleteCommand*.
  Tyto vlastnosti jsou navázány na Command objekty , které jsou použity v případě nutnosti manipulace s daty.
- obsahuje 3 hlavní metody:
  - *Fill* (plní DataSet daty)
  - *FillSchema* (queries the database for schema information that is necessary to update)
  - *Update* (změna databáze DataAdapter volá *DeleteCommand*, *InsertCommand* a *UpdateCommand* vlastnost)
  Například, když voláme *Fill* metodu DataAdapter objektu k získání dat z DB a naplnění DataSetu , je použita vlastnost *SelectCommand* objektu Command.
- DataAdapter „strážce brány“ , který stojí mezi naším DataSetem a zdrojem dat.

Příklad naplnění DataSetu

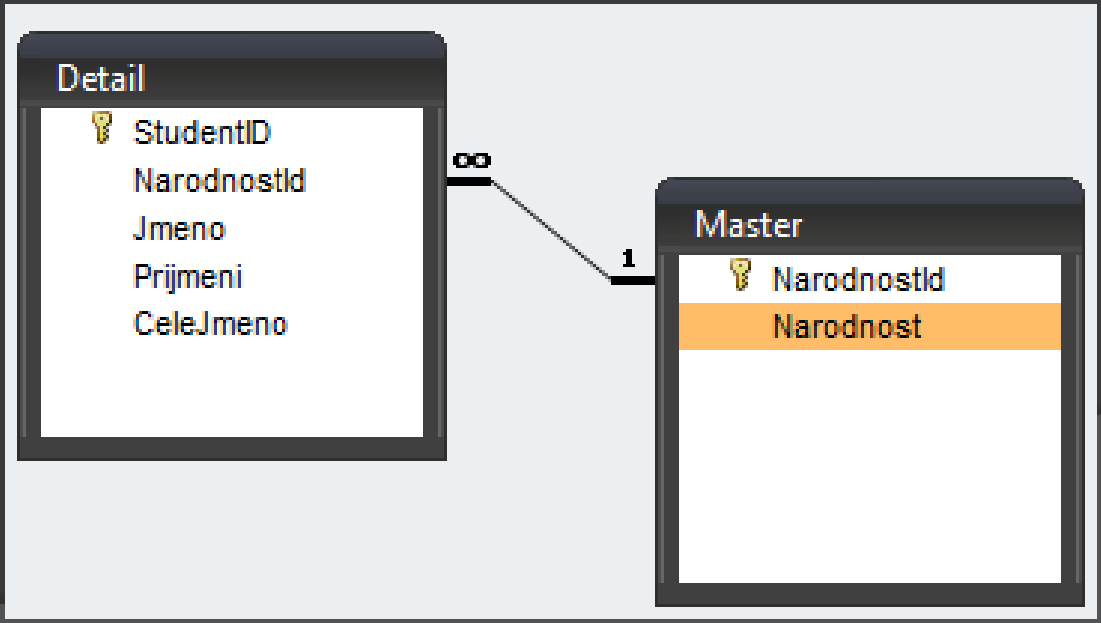
```
//Vytvořím instanci OleDbDataAdapter parametry OleDbConnection a select.. dotaz
OleDbDataAdapter dAdapter = new OleDbDataAdapter ("select * from PersonTable", con );
//naplnění DataSetu záznamy z tabulky "PersonTable"
dAdapter.Fill(dSet,"PersonTable");


//Příklad použití:
public bool fnGetDataConnection()
{
     try {
         con = new OleDbConnection(conString);
         dAdapter=new OleDbDataAdapter("select * from PersonTable", con);
         dSet=new DataSet();
         //obnoví řádky v DataSetu
         dAdapter.Fill(dSet, "PersonTable" );
      } catch(Exception ex) {
            MessageBox.Show("Error : " + ex.Message );
            //pokud připojení selhalo
            return false;
      } //try-catch
        //connection ok!
      return true;
}
```

Step by Step

- Vytvořte aplikaci, která vytvoří DataSet odpovídající tabulkám Detail a Master (viz. Obrázek). Naplňte Dataset daty a otestujte jeho funkčnost.

- Vytvořte nový projekt ConsoleApp s názvem ***DataSet_Demo_01***



Vytvoření tabulky

```
static DataTable MasterTable()
{

    Console.WriteLine();
    Console.WriteLine("Creating DataTable...");
    DataTable tb = new DataTable("MasterTable");
    Console.WriteLine(tb.TableName);
    //přidání dvou sloupců do vytvářené tabulky
    {
        tb.Columns.Add("NarodnostID", System.Type.GetType("System.Int32"));
        tb.Columns.Add("Narodnost", System.Type.GetType("System.String"));

    }
    //sloupec NarodnostID je automaticky inkrementované číslo od 0
    tb.Columns["NarodnostID"].AutoIncrement = true;
    tb.Columns["NarodnostID"].AutoIncrementSeed = 0;
    //primárním klíčem je sloupec NarodnostID
    DataColumn[] pkey = { tb.Columns["NarodnostID"] };
    tb.PrimaryKey = pkey;

    DataRow row = default(DataRow);
    //vytvoření dvou řádků tabulky MasterTable
    row = tb.NewRow();
    row["Narodnost"] = "Česká";
    tb.Rows.Add(row);

    row = tb.NewRow();
    row["Narodnost"] = "Slovenská";
    tb.Rows.Add(row);

    return tb;
}
```

Obdobně vytvořte tabulku Detail tak že:

- Sloupec StudentID je autoinkrement, s počáteční hodnotou 100 a krokem 10
- Sloupec StudentID je primární klíč
- Sloupec Prijmeni musí být unikátní hodnota a nesmí obsahovat nulovou hodnotu

```
tb.Columns["StudentID"].AutoIncrement = true;
tb.Columns["StudentID"].AutoIncrementSeed = 100;
tb.Columns["StudentID"].AutoIncrementStep = 10;


DataColumn[] pkey = { tb.Columns["StudentID"] };
tb.PrimaryKey = pkey;

tb.Columns["Prijmeni"].Unique = true;
tb.Columns["Prijmeni"].AllowDBNull = false;
```

Metoda pro vypsání obsahu tabulky

```
static void ShowTable(DataTable t)
{

    Console.WriteLine();
    foreach ( DataRow row in t.Rows) {
        foreach ( DataColumn col in t.Columns) {
            if (!row.IsNull(col)) {
                //row[col].ToString().PadRight(10)
                Console.Write(row[col]);
            }

        }
        Console.WriteLine();
    }
}
```

Main

- Vytvořte pomocí připravených metod tabulku Master a Detail, vytvořte DataSet a tyto tabulky do něj přidejte.

```
DataTable tbDetail = DetailTable();
ShowTable(tbDetail);

DataTable tbMaster = MasterTable();
ShowTable(tbMaster);



Console.WriteLine("Creating DataSet...");
System.Data.DataSet dsMyFirstDataSet = new System.Data.DataSet("MyFirstDataSet");

dsMyFirstDataSet.Tables.Add(tbDetail);
dsMyFirstDataSet.Tables.Add(tbMaster);
```

Vytvoření Constrainu (cizího klíče)

```
ForeignKeyConstraint fk = new ForeignKeyConstraint("FK_master_detail",
    dsMyFirstDataSet.Tables["MasterTable"].Columns["NarodnostID"],
    dsMyFirstDataSet.Tables["DetailTable"].Columns["NarodnostID"]);

fk.DeleteRule = Rule.Cascade;
dsMyFirstDataSet.Tables["DetailTable"].Constraints.Add(fk);

dsMyFirstDataSet.EnforceConstraints = false;
Console.WriteLine(dsMyFirstDataSet.Tables["DetailTable"].Rows.Count);
dsMyFirstDataSet.Tables["DetailTable"].Rows[0]["NarodnostID"] = 0;
dsMyFirstDataSet.Tables["DetailTable"].Rows[1]["NarodnostID"] = 0;
//dsMyFirstDataSet.Tables("MasterTable").Rows(0).Delete()
//Console.WriteLine(dsMyFirstDataSet.Tables("DetailTable").Rows.Count)
```

Vytvoření relace child/parent

- Použití vytvořené relace

```
dsMyFirstDataSet.Relations.Add("REL_master_detail",
    dsMyFirstDataSet.Tables["MasterTable"].Columns["NarodnostID"],
    dsMyFirstDataSet.Tables["DetailTable"].Columns["NarodnostID"],false);

dsMyFirstDataSet.Tables["MasterTable"].Columns.Add("PocetStudentu",
    System.Type.GetType("System.Int32"),
    "Count(Child(REL_master_detail).StudentID)");

ShowTable(tbMaster);


foreach (DataRow r in tbMaster.Rows[0].GetChildRows("REL_master_detail"))
{
    Console.WriteLine(r[2]);
}


Console.WriteLine(dsMyFirstDataSet.DataSetName);
Console.WriteLine();
```

Úkol

- Další ukázku práce s *Datasetem*, využití metody Fill adapteru najdete v šabloně CODE_TEMPLATE_DATASET_DEMO.TXT
- Zkopírujte kód do nového formulářového projektu, na formulář vložte tlačítko - nahraďte popisy konkrétními přístupovými údaji k DB a konkrétním SQL dotazem a otestujte aplikaci.

Co jsme neprobrali a neprobereme

- Technologie práce s DB se stále vyvíjí
- Ukázali jsme si pouze základy a principy programování pomocí knihovny ADO.NET
- LINQ (nutná znalost problematika delegátů, lambda výrazů)

linq on wikipedia