Fakultativne folije

2014/15.05.dodatak

Ostali članovi *IDbCommand*

- **Svojstva**
  - `CommandTimeout`: broj sekundi čekanja na izvršenje (standardno 30s)
  - `Parameters` – kolekcija parametara (argumenata) naredbe
    - `Parameter` - parametar parametrizirane SQL naredbe ili pohranjene procedure, sa svojstvima: `DbType, IsNullable, OleDbType, ParameterName, Precision, Scale, Size, SourceColumn`
  - `Transaction` – transakcija koje je naredba dio (o transakcijama kasnije)
  - `UpdatedRowSource` – određuje način ažuriranja izvora podataka, kad se naredba koristi sa skupom podataka i prilagodnikom podataka
- **Postupci**
  - `Cancel` – pokušaj prekida naredbe koja se izvršava
    - da bi prekid bio moguć, naredba mora biti pokrenuta na drugoj niti
    - u protivnom će kod biti blokiran, jer se naredbe izvršavaju sinkrono
  - `CreateParameter` – kreira novi `Parameter` objekt, koji se dodaje u kolekciju `Command.Parameters`
    - primjer: `public DbParameter CreateParameter();`
  - `Prepare` – postupak se koristi za pripremu (prekompilaciju) naredbe na izvoru podataka, s namjerom poboljšanja brzine njenog izvođenja

Transakcije

Transakcije

- **Transakcija - slijed naredbi koji se mora obaviti u cjelini ili se ne obavlja.**
  - "Ručne" transakcije
    - Transakcije definirane eksplicitnim naredbama za početak i kraj transakcije
  - Automatske transakcije
    - Usluga COM+ tehnologije koja omogućuje dizajn razreda koji će u pogonu sudjelovati u transakcijama (`System.EnterpriseServices.ServicedComponent`)
    - Jedina opcija za transakcije nad podacima iz raznorodnih izvora.
  - Transakcije SUBP
    - Logika ovih transakcija sadržana je u pohranjenim procedurama
    - Pružaju najbolje performanse, ali ih je nešto teže ugraditi
    - Ograničenje - Kad se iz neke transakcije poziva dvije ili više procedura, koristi se samo jedan model transakcija (ručne ili automatske).
- **Ručne transakcije započinju postupkom konekcije**
  - `BeginTransaction` — programsko započimanje transakcije
  - npr: `conn.BeginTransaction();` ili `conn.BeginTransaction(isolationLevel, transName);`

OleDbTransaction i *SQLTransaction*

- **Svojstva:**
  - `Connection` – OleDbConnection, odnosno SQLConnection objekt pridružen transakciji ili `null` referenca ako transakcija nije više validna
  - `IsolationLevel` – razina izolacije zaključavanja za transakciju, vrijednost odgovarajućeg enumeratora izvedenog iz `IDbTransaction.IsolationLevel`
    - `Chaos` – slično `ReadUncommitted`, s tim da nepotvrđene promjene načinjene na višoj razini izolacije ne mogu biti nadvladane ("pregažene")
    - `ReadCommitted` (standardno) – čitaju se samo potvrđeno pohranjeni podaci (izbjegava prljavo čitanje) uz dijeljeno zaključavanje, koji ipak mogu biti promijenjeni prije kraja transakcije → non-repeatable reads or phantom data
    - `ReadUncommitted` – prljavo čitanje, bez zaključavanja
    - `RepeatableRead` – pročitani zapisi zaključavaju se do kraja transakcije, onemogućujući promjenu drugim korisnicima. Eliminira neponovljiva čitanja, ali ne i fantomske retke.
    - `Serializable` – zaključava se čitav skup pročitanih podataka, koji ostaju zaključani do završetka transakcije
    - `Unspecified` – koristi se neko drugo zaključavanje, čija razina se trenutno ne može odrediti

OleDbTransaction i *SQLTransaction*

- **Postupci**
  - `BeginTransaction` — programsko započimanje transakcije
  - `Begin` — programsko započimanje ugniježdene transakcije
    - osnovna transakcija ne može biti potvrđena tako dugo dok se ne potvrde sve ugniježdene transakcije)
    - samo `OleDbTransaction` ima postupak `Begin`
  - `Commit` — spremanje/potvrđivanje promjena načinjenih tijekom transakcije te završetak transakcije
  - `RollbackTransaction` — odbacivanje promjena uz završetak transakcije
  - `CommitTransaction` i `RollbackTransaction` mogu započeti novu transakciju

- **Nakon što transakcija započne na konekciji, sve naredbe obavljene na toj konekciji moraju sudjelovati u transakciji**
  - Da bi se to stvarno i dogodilo, naredbe moraju referencirati konekciju, što se ne događa automatski.
  - Ipak, za naredbe koje su sudjelovale u transakcijama koje su završile, referenca na konekciju se automatski postavlja na `null`

Naredbe s transakcijom

- **Otvaranje konekcije i kreiranje transakcije**

```
SqlConnection conn = new SqlConnection(...);
conn.Open();
SqlTransaction trans =
  conn.BeginTransaction(IsolationLevel.ReadCommitted);
```

- **Kreiranje naredbe i njeno priključivanje transakciji**

```
SqlCommand cmd = new SqlCommand("DELETE FROM ...", conn);
cmd.Transaction = trans;
```

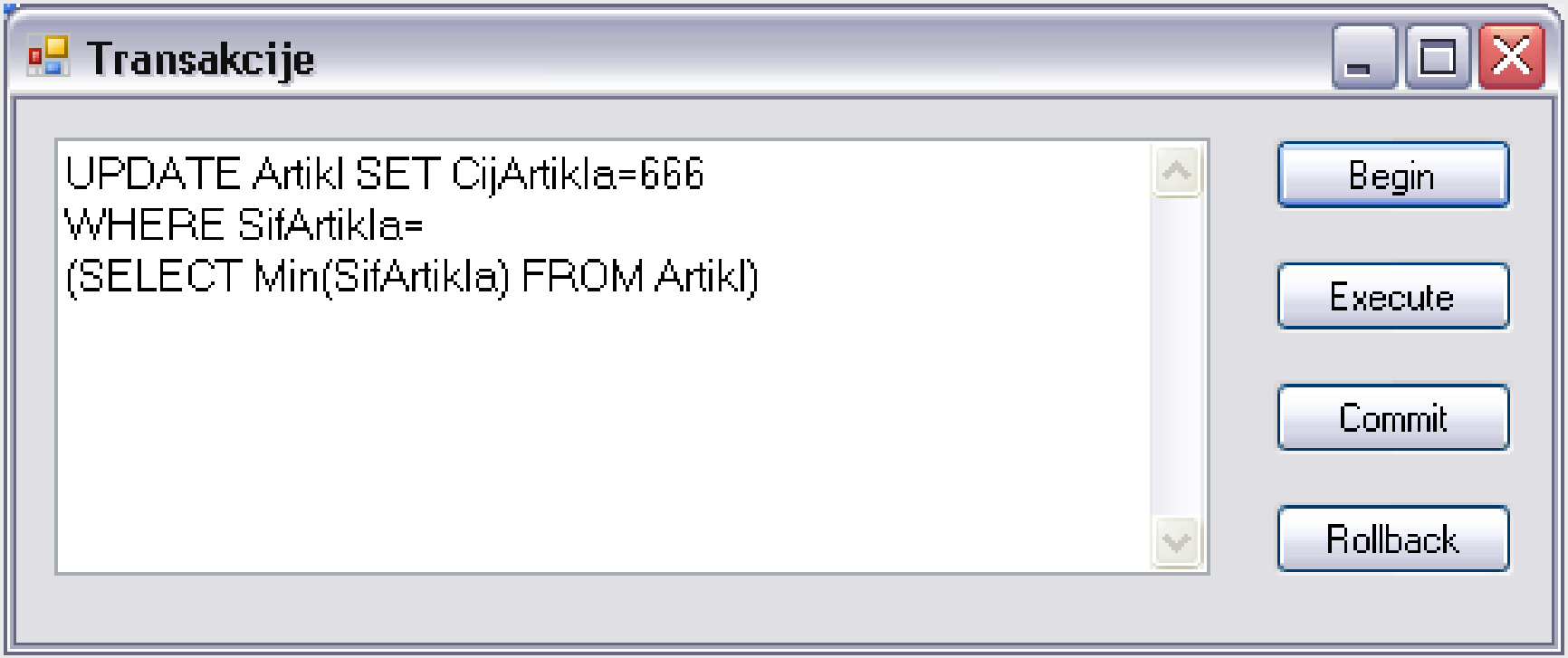
- **Izvođenje naredbe i završetak transakcije**

```
try {
  int cnt = cmd.ExecuteNonQuery();
  trans.Commit();
} catch {
  trans.Rollback();
} finally {
  conn.Close();
}
```

- **Primjer: 🗁ADO\Transakcije**

Primjer s transakcijom

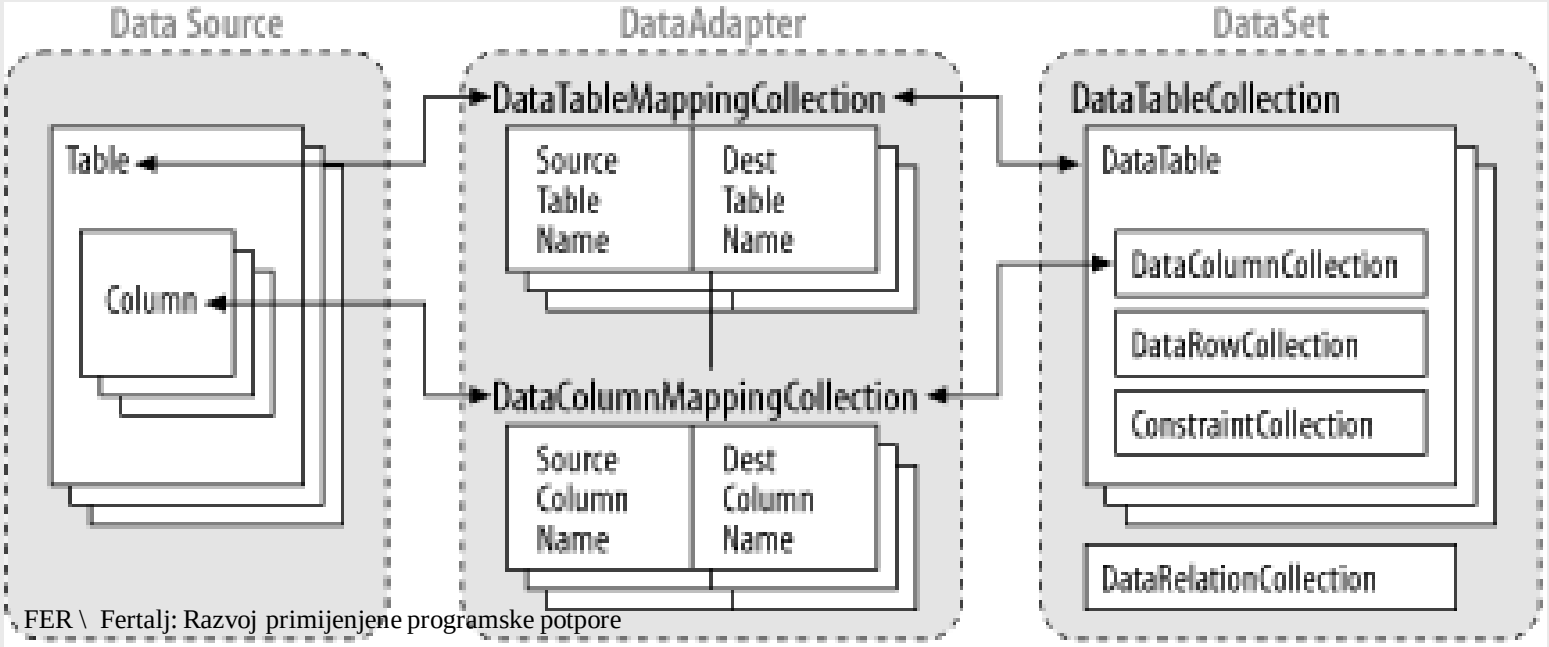
- **Primjer: 🗁ADO\Transakcije**



Dinamičko stvaranje skupova podataka

Mapiranje objekata

- **Izvorne tablice i stupci mogu se različito zvati u skupu podataka**
- **`TableMappings` svojstvo DataAdaptera**
  - `DataTableMappingCollection` kolekcija pridruživanja izvornih i DataSet tablica, instanca razreda
  - Pojedini `DataTableMapping` objekt ima svojstvo `DataColumnMapping` – kolekciju pridruživanja stupaca
- **Primjer: 🗁ADODataSet\Artikl ili ADODataSet\Drzava ... \<idataset>.Designer.cs**
  - DataAdapter / Properties / TableMapping (collection)



Razred *DataTable*

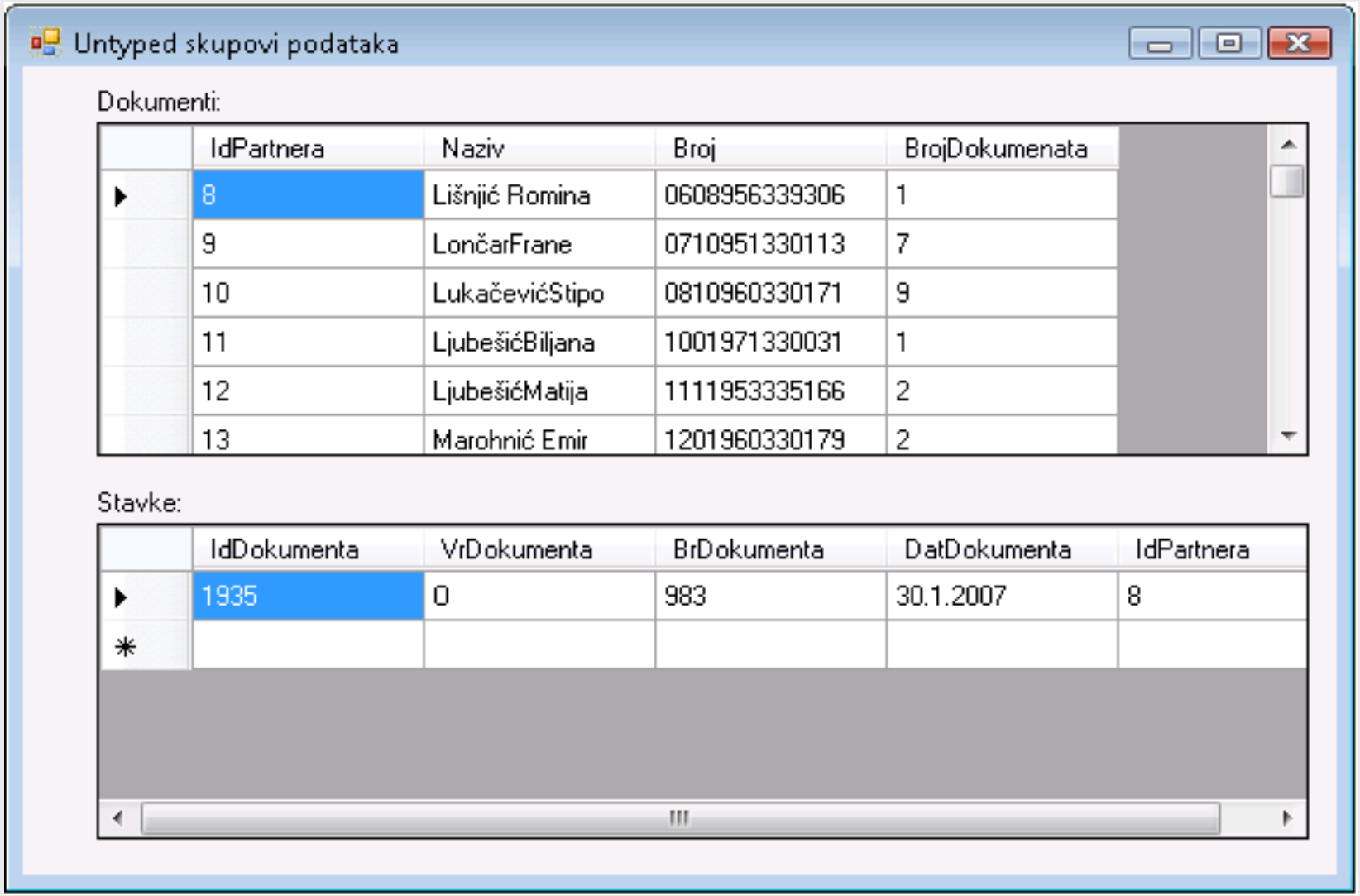
- **Reprezentacija tablice s podacima u memoriji – definira strukturu podataka**
  - `DataTable();`
  - `DataTable(string);`
- **Svojstva**
  - `Columns` – kolekcija atributa
  - `Rows` - kolekcija `DataRow` objekata
  - `ChildRelations` - kolekcija veza u kojima tablica referencira druge tablice
  - `ParentRelations` – kolekcija veza kojima je tablica referencirana iz drugih
  - `Constraints` – kolekcija ograničenja
  - `PrimaryKey` – kolekcija `DataColumn` objekata koji čine primarni ključ
  - `TableName` – naziv tablice
- **Neki postupci**
  - `Clear` – čisti podatke
  - `Clone` – kopira kompletnu strukturu uključujući ograničenja, ali ne i podatke
  - `Copy` – kopira strukturu i podatke objekta koji izvodi postupak
  - `NewRow` – kreira novi `DataRow`
  - `Select` – vraća kolekciju `DataRow` objekata

Razred *DataColumn*

- **`DataColumn` – reprezentacija sheme `DataTable` stupca**
  - `DataColumn()`
  - `DataColumn(columnName, dataType)`
- **Svojstva**
  - `AllowDbNull` – neobvezne vrijednosti
  - `AutoIncrement, AutoIncrementSeed, AutoIncrementStep` – polje sa samopovećavajućim vrijednostima, početna vrijednost, korak
  - `Caption` - labela
  - `ColumnName` - naziv
  - `DataType` – tip podatka (Boolean, Byte, Char, DateTime, Decimal, Double, Int16, Int32, Int64, SByte, Single, String, TimeSpan, UInt16, UInt32, UInt64)
  - `DefaultValue` – standardna, predviđena vrijednost
  - `Expression` – izraz za izračunatu vrijednost (npr. formula, agregatna f-ja)
  - `MaxLength` – najveća duljina tekstovnog podatka
  - `ReadOnly` – oznaka da se vrijednost ne mijenja nakon što je redak dodan
  - `Unique` – bool oznaka jedinstvene vrijednost u tablici

Dinamičko kreiranje tablica

- **Primjer: 🗁ADO\DataSetUnTyped**



Dinamičko kreiranje tablica

- **Primjer: 🗁ADO\DataSetUnTyped**
  - System.Type.GetType(string); # dobavlja tip za zadani naziv

```
DataTable dt = new DataTable("Dokument");

dt.Columns.Add("IdDokumenta", typeof(int));
dt.Columns.Add("VrDokumenta", typeof(string));
dt.Columns.Add("BrDokumenta", typeof(int));
dt.Columns.Add("DatDokumenta", typeof(System.DateTime));
dt.Columns.Add("IdPartnera", typeof(int));
dt.Columns.Add("IznosDokumenta", typeof(decimal));

dsUntyped = new DataSet("DataSetUntyped");
dsUntyped.Tables.Add(dt);
```

Dinamičko dodavanje stupaca tablice

- **Primjer: 🗁ADO\DokumentStavka – izračunata polja**

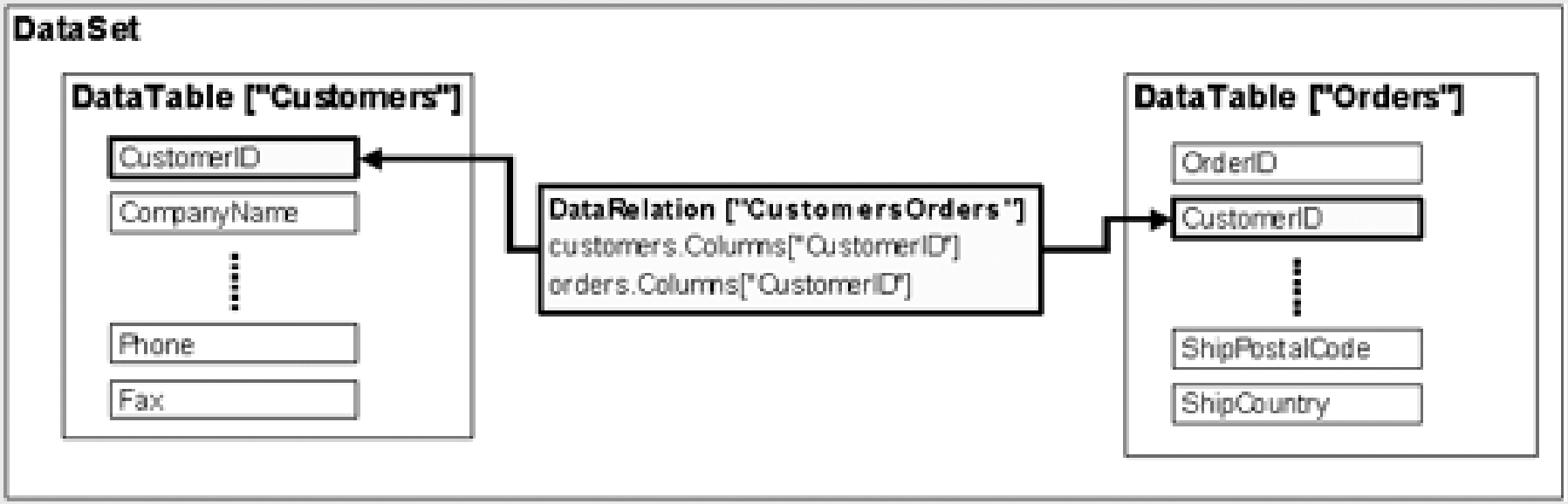
```
dataSetDokumentStavka.Stavka.Columns.Add( // 1.NAČIN
    "UkCijArtikla", typeof(float),
    "JedCijArtikla * KolArtikla * (1 - PostoRabat)");

dataSetDokumentStavka.Dokument.Columns.Add( // 2.NAČIN
    new DataColumn(
    "UkIznosDokumenta", typeof(float),
    "Sum(Child(FK_Dokument_Stavka).UkCijArtikla)
                * (1 + PostoPorez)")
    );
dataSetDokumentStavka.Dokument.Columns.Add(
    new DataColumn(
    "BrStavki", typeof(int),
    "Count(Child(FK_Dokument_Stavka).IdDokumenta)")
    );
```

Razred *DataRelation*

- **Logička veza između dva `DataTable` objekta**
  - uspostavlja se parovima `DataColumn` objekata iz pojedinih `DataTable`
    - `DataType` odgovarajućih `DataColumn` mora biti jednak
    - npr. nije dozvoljeno povezivanje `Int32` i `String` stupaca
- **Konstruktori**
  - `public DataRelation(string, DataColumn, DataColumn);`
  - `public DataRelation(string, DataColumn[], DataColumn[]);`
- **Svojstva**
  - `ChildTable, ParentTable` – referencirajuća/zavisna tablica, odnosno referencirana/nadređena tablica
  - `ChildColumns, ParentColumns` – kolekcije atributa povezanih tablica
  - `ChildKeyConstraint, ParentKeyConstraint` – definiraju `ForeignKeyConstraint` ograničenje na dijete, odnosno roditelja
  - `DataSet` – skup podataka kojem veza pripada
  - `RelationName` – naziv veze

Dinamičko kreiranje veza




```
DataRelation relOrdDet;
DataColumn colMaster;
DataColumn colDetail;

colMaster = dsUntyped.Tables["Partner"].Columns["IdPartnera"];
colDetail = dsUntyped.Tables["Dokument"].Columns["IdPartnera"];
relOrdDet = new DataRelation("FK_Partner_Dokument",
                             colMaster, colDetail);
dsUntyped.Relations.Add(relOrdDet);
```

Ograničenje veze – stranog ključa

- **`ForeignKeyConstraint` razred sa svojstvima**
  - `AcceptRejectRule` – akcija koja se obavlja prigodom `AcceptChanges`
  - `Table, Columns` – podređena tablica i njeni atributi ograničenja
  - `RelatedTable, RelatedColumns` – nadređena tablica i njeni atributi ograničenja
  - `DeleteRule, UpdateRule` - ograničenje dodavanja, odnosno izmjene
    - `enum Rule = { Cascade, None, SetDefault, SetNull }`

- **Primjer: 🗁ADO\DatasetUntyped**

```
ForeignKeyConstraint fk;
fk = new System.Data.ForeignKeyConstraint
       ("FK_Partner_Dokument",
         dsUntyped.Tables["Partner"].Columns["IdPartnera"],
         dsUntyped.Tables["Dokument"].Columns["Idpartnera"]
       );
dsUntyped.Tables["Dokument"].Constraints.Add(fk);
```

Ograničenje jedinstvene vrijednosti

- **`UniqueConstraint` razred sa svojstvima**
  - `Columns` – polje atributa na koje se odnosi ograničenje
  - `ConstraintName` – naziv ograničenja
  - `IsPrimaryKey` – bool oznaka da je ograničenje primarni ključ

- **Primjer: 🗁ADO\DataSetUntyped**

```
UniqueConstraint uc;
uc = new System.Data.UniqueConstraint(
       "NewUnique",
       new DataColumn[] {
         dt.Columns["VrDokumenta"],
         dt.Columns["BrDokumenta"] }
       false );
//     dt.Columns["IdDokumenta"]); // bez new

dt.Constraints.Add(uc);
```

Dohvaćanje zavisnih podataka

- **`DataRow` postupci**
  - `GetChildRows` – dohvaća sve zavisne retke
  - `GetParentRows` – dohvaća nadređeni redak
- **`DataSet` postupak `Merge` – udružuje druge objekte u `DataSet`**
  - `Merge(DataRow[]); // udružuje polje DataRow objekata`
  - `Merge(DataSet);  // udružuje skup i njegovu shemu`
  - `Merge(DataTable); // udružuje tablicu i njezinu shemu`
- **`DataSet` događaj**
  - `MergeFaled` – narušeni integriret, uz postavljeno `EnforceConstraints`
- **Primjer:**

```
DataRowView drvCurr;
DataSet dsPartnerDoc;
dsPartnerDoc = new System.Data.DataSet();
// kreirati vezu PartnerDokument po potrebi
...
dsPartnerDoc.Merge(
  drvCurr.Row.GetChildRows("PartnerDokument"));
```

Zadaci za vježbu

- Primjer *DataSetUntyped* prilagoditi tako da se iznos dokumenta računa kao agregatna funkcija u upitu izvora podataka.

- U *DataSetUntyped* dodati tablicu *VrstaDokumenta* te ju povezati s tablicom Dokument stranim ključem. Ograničenje na podatke podržati prethodnim dodavanjem svih do sada korištenih vrsta dokumenata iz tablice *Dokument* u tablicu *VrstaDokumenta*.