Objektově relační mapování

Ondřej Motl, 2012

1. Základní přehled

Objektově relační mapování (ORM) je programovací technika, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.

Objekty reálného světa jsou v aplikaci reprezentovány jako entity, které jsou popsány svými vlastnostmi. V relační databázi je entita reprezentována jako řádek v databázové tabulce, v objektově orientovaném jazyce je entita reprezentována jako instance nějaké třídy.

Třída v objektově orientovaném jazyce

Osoba
+Id : int +Jmeno : string +Prijmeni : string

Tabulka v relační databázi

Osoba		
PK	**Id**	**int**
	Jmeno **Prijmeni**	**varchar(35)** **varchar(35)**

Instance třídy Osoba

Jan Becher : Osoba
Id : int = 1 Jmeno : string = Jan Prijmeni : string = Becher

jack Daniels : Osoba
Id : int = 2 Jmeno : string = Jack Prijmeni : string = Daniels

Řádky v tabulce Osoba

Id	Jmeno	Prijmeni
1	Jan	Becher
2	Jack	Daniels

ORM se stará o konverzi mezi relační databází a objekty, se kterými se pracuje v objektově orientovaném jazyce.

Vývojář je při práci s daty, které jsou v aplikaci reprezentovány pomocí objektů, odstíněn od nutnosti pracovat s SQL dotazy konkrétní relační databáze.

ORM usnadňuje provádění běžných databázových operací jako je čtení, zápis, úprava a mazání dat.

ORM se stará o automatickou konverzi rozdílných datových typů mezi databázovým systémem a programovacím jazykem.

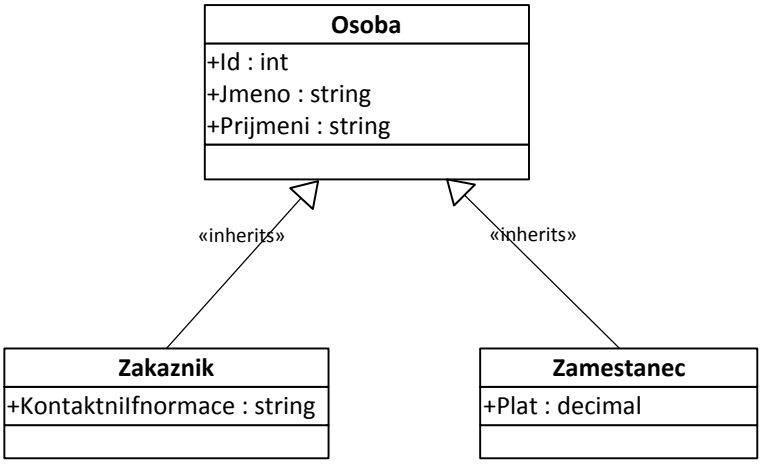
Pokročilé techniky ORM řeší možnost využití dědičnosti, kterou relační databáze nepodporují.

Hlavním cílem ORM je synchronizace mezi objekty používanými v aplikaci a jejich reprezentací v databázovém systému tak, aby byla zajištěna persistence dat. Vývojář potřebuje persistentně uchovávat objekty, ale nepotřebuje se starat, jak se tato persistence provede.

2. Dědičnost

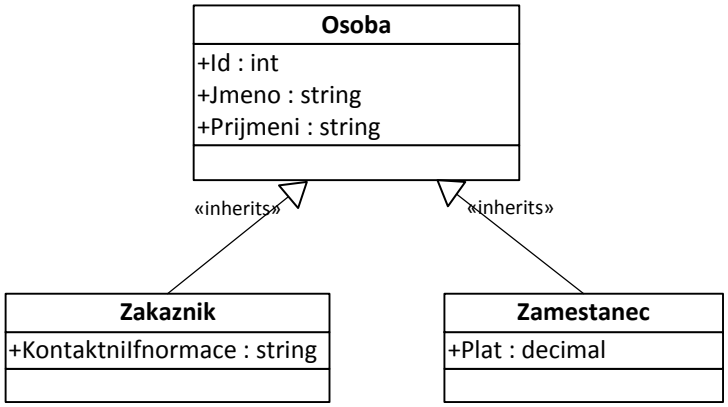
Relační databáze nepodporují dědičnost a proto je nutné hierarchii tříd rozložit do databázových tabulek.

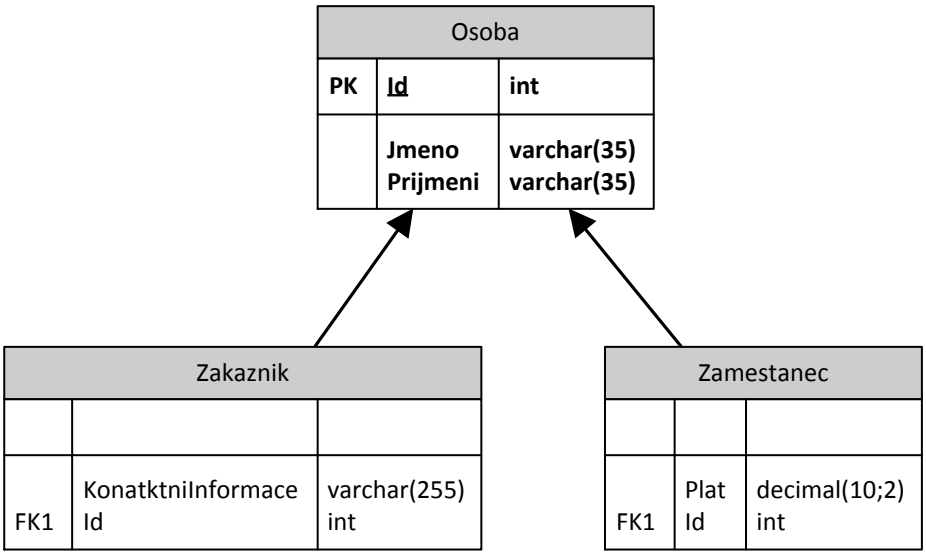
Single Table Inheritance



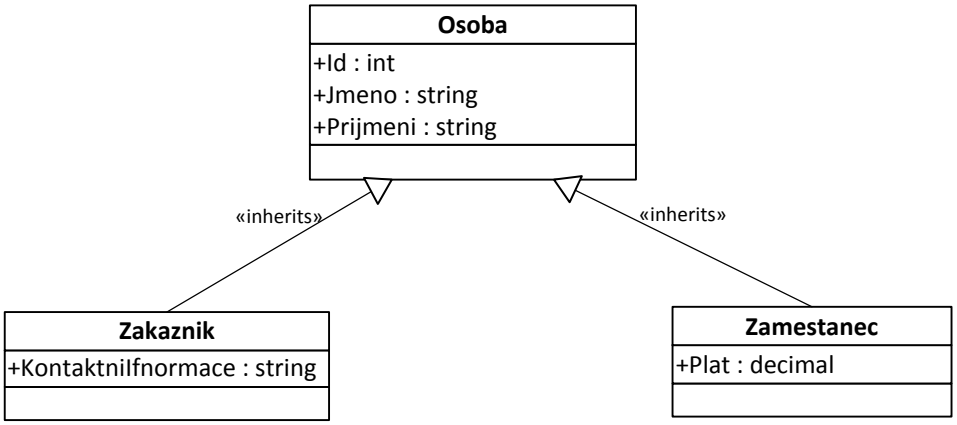
Osoba		
PK	**Id**	**int**
	Jmeno	**varchar(35)**
	Prijmeni	**varchar(35)**
	Plat	decimal(10;2)
	KonatktniInformace	varchar(255)

Class Table Inheritance





Concrete Table Inheritance



Zakaznik		
PK	**Id**	**int**
	Jmeno **Prijmeni** KonatktniInformace	**varchar(35)** **varchar(35)** varchar(255)

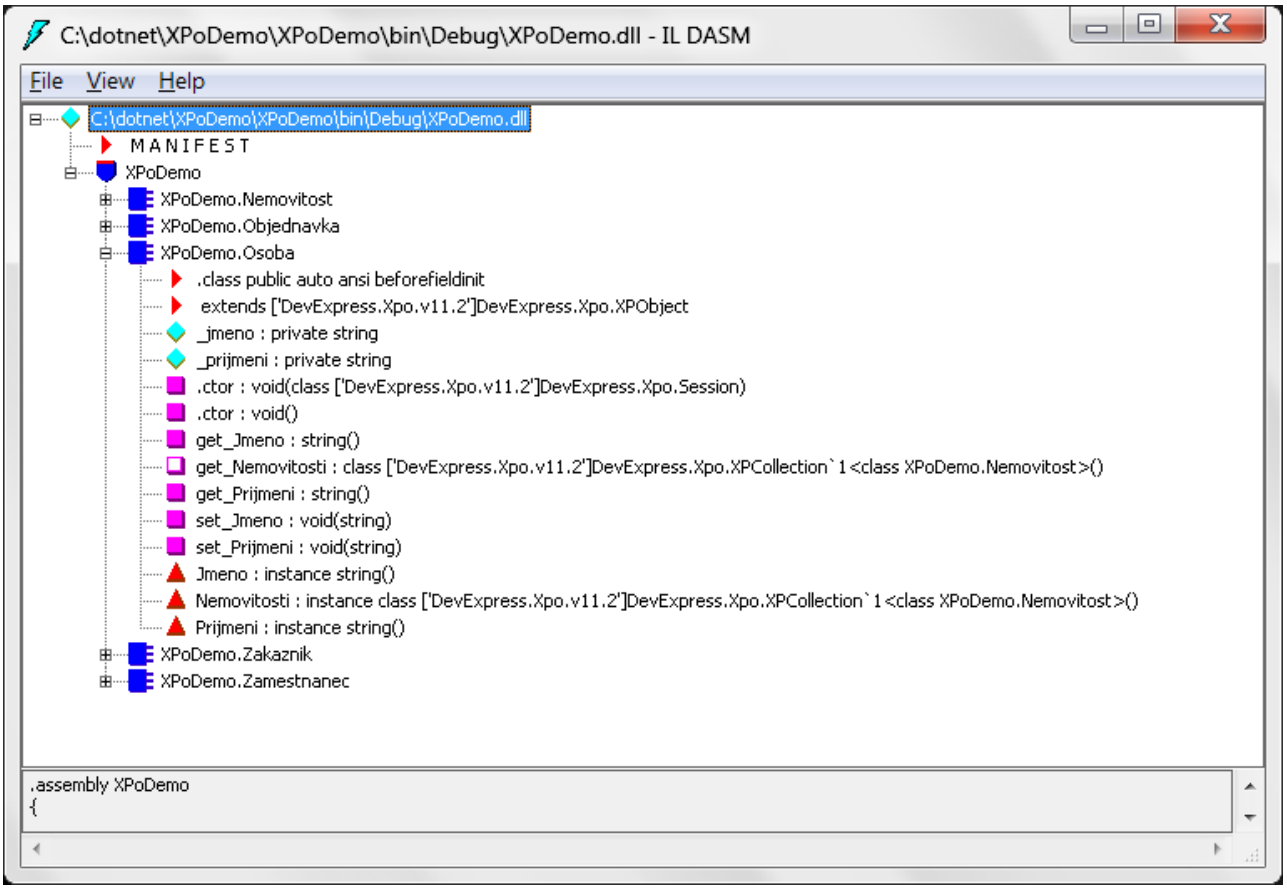
Zamestnanec		
PK	**Id**	**int**
	Jmeno **Prijmeni** Plat	**varchar(35)** **varchar(35)** decimal(10;2)

3. eXpress Persistent Objects

Komerční produkt na platformě .NET

- OR mapování s využitím .NET reflexe a uživatelských atributů
- Automatické generování databázových tabulek
- Podpora OR mapování pro existující databáze
- Mapování vztahů (relationships) mezi objekty
- Čtení dat podle výběrových kritérií a podpora LINQ
- Podpora transakcí
- Podpora až 14 relačních databází – Oracle, MS SQL Server, Postgres, SQLLite, MySql etc.

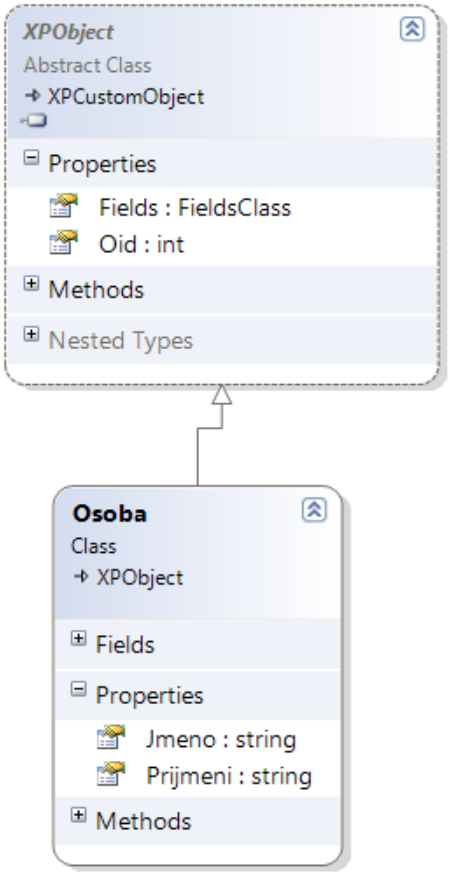
.NET reflexe



3.1 Persitentní třídy

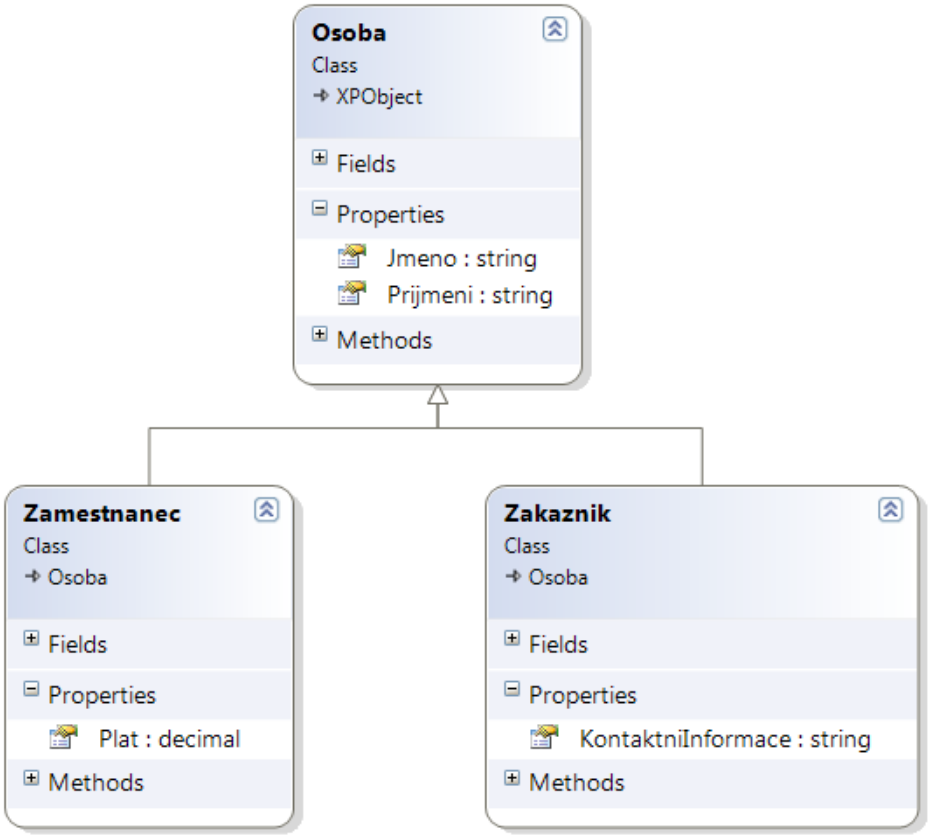
Konkrétní implemtentace persistentích tříd jsou odvozeny od abstraktní třídy XPObject.

```
public class Osoba : XPObject
{
    private string _jmeno;
    private string _prijmeni;

    [Size(35)]
    public string Jmeno
    {
        get { return _jmeno; }
        set { _jmeno = value; }
    }

    [Size(35)]
    public string Prijmeni
    {
        get { return _prijmeni; }
        set { _prijmeni = value; }
    }
}
```



Osoba		
PK	**OID**	**int identity**
	Jmeno Prijmeni OptimisticLockField GCRecord	nvarchar(35) nvarchar(35) int int

3.2 Implementace dědičnosti



Single Table Inheritance

```
[MapInheritance(MapInheritanceType.ParentTable)]
public class Zamestnanec : Osoba
{
    private decimal _plat;

    public decimal Plat
    {
        get { return _plat; }
        set { _plat = value; }
    }
}

[MapInheritance(MapInheritanceType.ParentTable)]
public class Zakaznik : Osoba
{
    private string _kontaktniInformace;

    [Size(255)]
    public string KontaktniInformace
    {
        get { return _kontaktniInformace; }
        set { _kontaktniInformace = value; }
    }
}
```

Osoba		
PK	**OID**	**int identity**
	Jmeno	nvarchar(35)
	Prijmeni	nvarchar(35)
	OptimisticLockField	int
	GCRecord	int
FK1	ObjectType	int
	KontaktniInformace	nvarchar(255)
	Plat	money

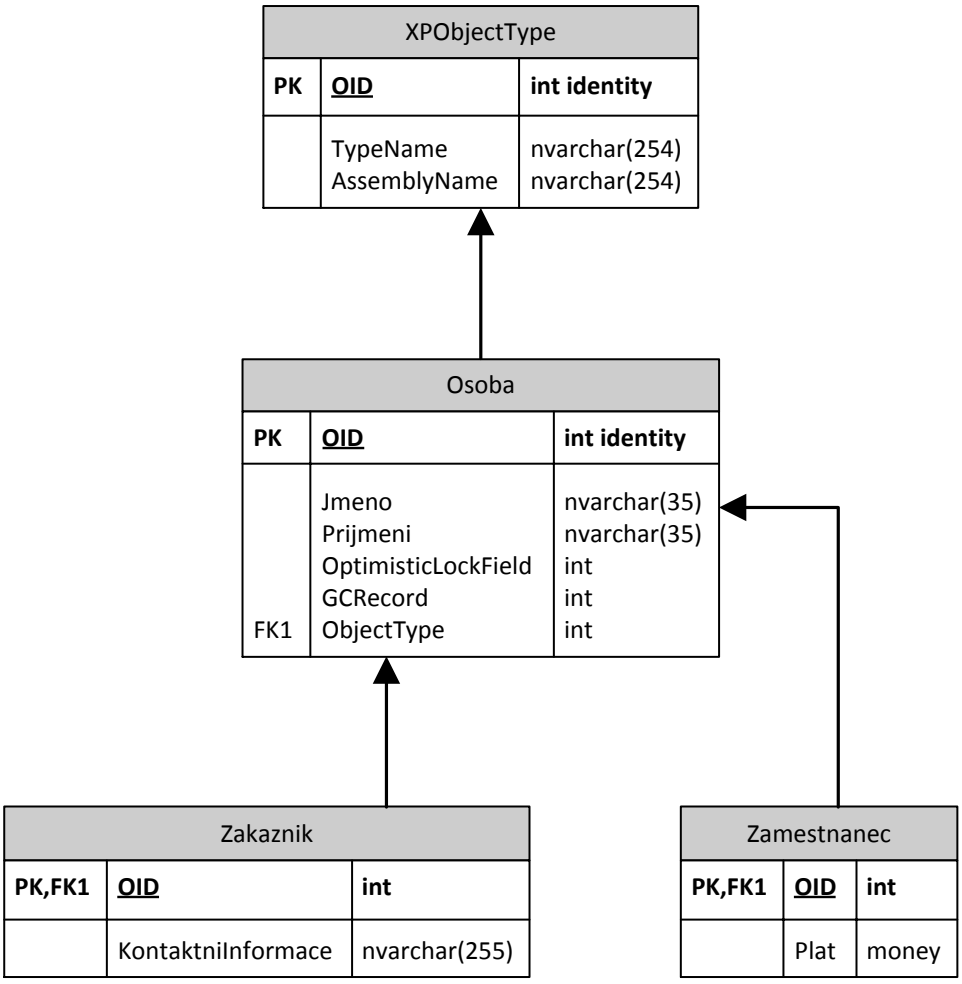
XPObjectType		
PK	**OID**	**int identity**
	TypeName	nvarchar(254)
	AssemblyName	nvarchar(254)

Data v tabulce XPObjectType

OID	TypeName	AssemblyName
1	XPoDemo.Zamestnanec	XPoDemo
2	XPoDemo.Zakaznik	XPoDemo

Class Table Inheritance

```
[MapInheritance(MapInheritanceType.OwnTable)]
public class Zamestnanec : Osoba
{
    private decimal _plat;

    public decimal Plat
    {
        get { return _plat; }
        set { _plat = value; }
    }
}

[MapInheritance(MapInheritanceType.OwnTable)]
public class Zakaznik : Osoba
{
    private string _kontaktniInformace;

    [Size(255)]
    public string KontaktniInformace
    {
        get { return _kontaktniInformace; }
        set { _kontaktniInformace = value; }
    }
}
```



3.3 Mapování do existující databáze

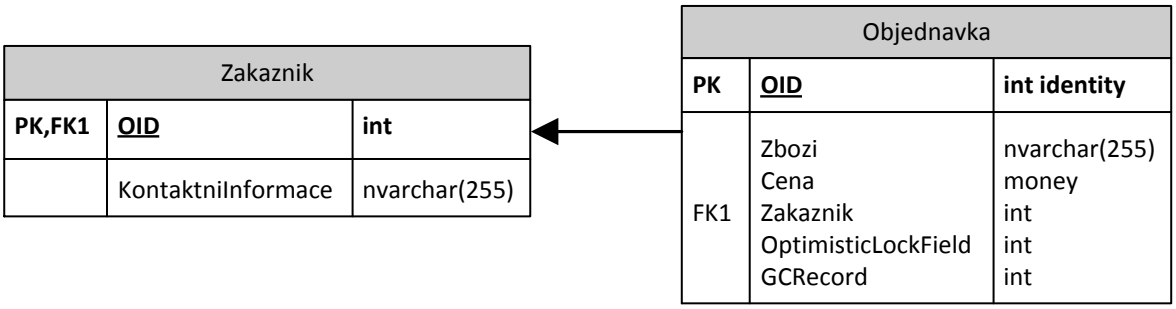
PREFIX_OSOBA		
PK	**ID**	**int identity**
	PRIJMENI **JMENO**	**varchar(35)** **varchar(35)**

```
[Persistent("PREFIX_OSOBA")]
public class Osoba : XPLiteObject
{
    private int _id;
    private string _jmeno;
    private string _prijmeni;

    [Persistent("ID"), Key(true)]
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }

    [Persistent("JMENO")]
    public string Jmeno
    {
        get { return _jmeno; }
        set
        {
            _jmeno = value;
        }
    }

    [Persistent("PRIJMENI")]
    public string Prijmeni
    {
        get { return _prijmeni; }
        set { _prijmeni = value; }
    }
}
```

3.4 Implementace vztahů mezi objekty

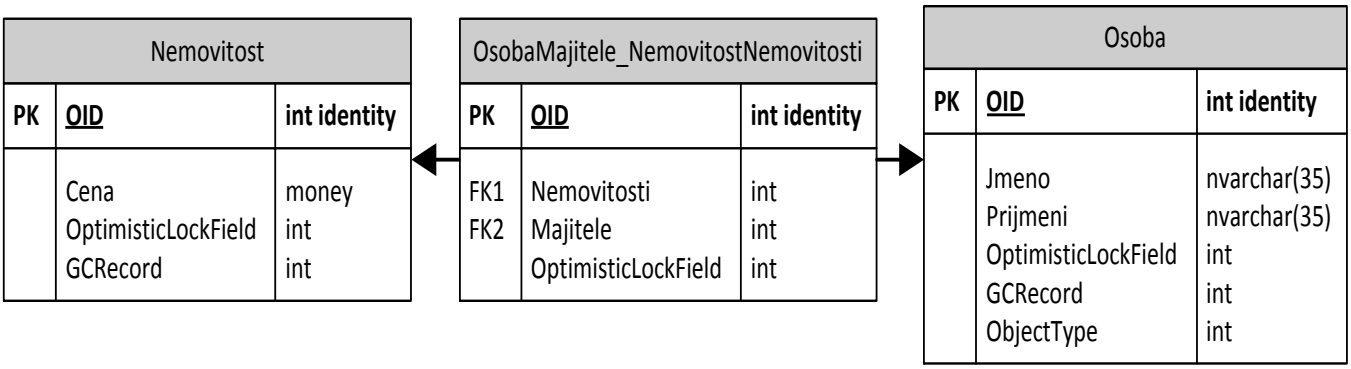
Asociace 1 : N

```
public class Objednavka : XPObject
{
    private string _zbozi;
    private decimal _cena;
    private Zakaznik _zakaznik;

    [Size(255)]
    public string Zbozi
    {
        get { return _zbozi; } set { _zbozi = value; }
    }

    public decimal Cena
    {
        get { return _cena; } set { _cena = value; }
    }

    [Association("Zakaznik-Objednavka")]
    public Zakaznik Zakaznik
    {
        get { return _zakaznik; } set { _zakaznik = value; }
    }
}

public class Zakaznik : Osoba
{
    private string _kontaktniInformace;

    [Size(255)]
    public string KontaktniInformace
    {
        get { return _kontaktniInformace; }
        set { _kontaktniInformace = value; }
    }

    [Association("Zakaznik-Objednavka")]
    public XPCollection<Objednavka> Objednavky
    {
        get { return GetCollection<Objednavka>("Objednavky"); }
    }

}
```



Asociace M:N

```
public class Nemovitost : XPObject
{
    private decimal _cena;

    public decimal Cena
    {
        get { return _cena; } set { _cena = value; }
    }

    [Association("Nemovitost-Majitel")]
    public XPCollection<Osoba> Majitele
    {
        get { return GetCollection<Osoba>("Majitele"); }
    }
}

public class Osoba : XPObject
{
    private string _jmeno;
    private string _prijmeni;

    public string Jmeno
    {
        get { return _jmeno; } set { _jmeno = value; }
    }

    public string Prijmeni
    {
        get { return _prijmeni; } set { _prijmeni = value; }
    }

    [Association("Nemovitost-Majitel")]
    public XPCollection<Nemovitost> Nemovitosti
    {
        get { return GetCollection<Nemovitost>("Nemovitosti"); }
    }

}
```



3.5 Čtení dat z databáze

XpoDefault – statická třída, poskytuje globální nastavení

XPCollection<T> - kolekce persistentích objektů, implementuje *delayed loading*, kolekce ja naplněna daty, až když je poprvé přistupováno k jejím položkám

Session – objekty třídy Session representují cache persisteních objektů

```
XpoDefault.ConnectionString = "XpoProvider=MSSqlServer; Data source=SW41; Initial
catalog=XAF_DEMO01; Integrated Security=SSPI";

XPCollection<Osoba> osoby = new XPCollection<Osoba>(XpoDefault.Session);
foreach (Osoba osoba in osoby)
{
      Console.WriteLine(osoba.Jmeno + " " + osoba.Prijmeni);
}
```

```
Výstup:
Jan Becher
Jack Daniels
```

Vygenerovaný SQL dotaz:

```
select N0."OID",N0."Jmeno",N0."Prijmeni",N0."OptimisticLockField",
N0."GCRecord" from "dbo"."Osoba" N0
```

Třídění

```
XPCollection<Osoba> osoby = new XPCollection<Osoba>(XpoDefault.Session);
osoby.Sorting.Add(new SortProperty("Prijmeni",SortingDirection.Ascending));
osoby.Sorting.Add(new SortProperty("Jmeno", SortingDirection.Ascending));
```

3.6 Kriteria pro výběr dat

```
CriteriaOperator criteriaOperator = new BinaryOperator("Cena", 1000000,
                                                BinaryOperatorType.GreaterOrEqual);

XPCollection<Nemovitost> nemovitosti = new XPCollection<Nemovitost>(
                                        XpoDefault.Session, criteriaOperator);
```

Kriterium může být napsáno jako string:

```
CriteriaOperator criteriaOperator = CriteriaOperator.Parse("Cena > 1000000");
```

Logické operátory

```
CriteriaOperator criteriaOperator = CriteriaOperator.Parse(
                                        "Jmeno = 'Jan' And Prijmeni = 'Becher'");

XPCollection<Osoba> osoby = new XPCollection<Osoba>(
                                        XpoDefault.Session, criteriaOperator);
```

```
CriteriaOperator criteriaOperator = CriteriaOperator.And(
                                    new BinaryOperator("Jmeno", "Jan"),
                                    new BinaryOperator("Prijmeni", "Becher"));

XPCollection<Osoba> osoby = new XPCollection<Osoba>(XpoDefault.Session,
                                                    criteriaOperator);
```

```
CriteriaOperator criteriaOperator = CriteriaOperator.Parse(
                                "Prijmeni = 'Becher' || Prijmeni = 'Daniels'");

XPCollection<Osoba> osoby = new XPCollection<Osoba>(XpoDefault.Session,
                                                    criteriaOperator);
```

Is Null operátor

```
XPCollection<Objednavka> objednavky =
        new XPCollection<Objednavka>(XpoDefault.Session,
                                     CriteriaOperator.Parse("Zakaznik is null"));
```

Agregační funkce

```
XPCollection<Zakaznik> zakaznici =
new XPCollection<Zakaznik>(XpoDefault.Session,
                           CriteriaOperator.Parse("Objednavky.Sum(Cena) > 1000000"));
```

3.7 LINQ to XPO

LINQ (Language Integrated Query) je integrovaný jazyk pro dotazování. Dotazy se píší přímo v programovacím jazyce (C# 3.0, Visual Basic 9.0 a vyšší).

```
XPQuery<Osoba> query = new XPQuery<Osoba>(XpoDefault.Session);

IEnumerable<Osoba> osoby = from osoba in query
                           orderby osoba.Prijmeni , osoba.Jmeno
                           select osoba;

foreach (Osoba osoba in osoby)
{
      Console.WriteLine(osoba.Jmeno + " " + osoba.Prijmeni);
}
```

```
XPQuery<Osoba> query = new XPQuery<Osoba>(XpoDefault.Session);
var osoby = from osoba in query
               where osoba.Prijmeni == "Daniels" || osoba.Prijmeni == "Becher"
               orderby osoba.Prijmeni , osoba.Jmeno
               select new {osoba.Prijmeni, osoba.Jmeno};

foreach (var osoba in osoby)
{
    Console.WriteLine(osoba.Jmeno + " " + osoba.Prijmeni);
}
```

```
XPQuery<Objednavka> query = new XPQuery<Objednavka>(XpoDefault.Session);
var sumaZaMesic = from obj in query
            where obj.Datum.Year == 2011
            group obj by obj.Datum.Month
            into gr
            select new {Mesic = gr.Key, CenaCelkem = gr.Sum(ob => ob.Cena)};

foreach (var suma in sumaZaMesic)
{
    Console.WriteLine(suma.Mesic + " " +suma.CenaCelkem);
}
```

```
XPQuery<Objednavka> query = new XPQuery<Objednavka>(XpoDefault.Session);
decimal sumaZaRok = (from obj in query
                         where obj.Datum.Year == 2011
                         select obj.Cena).Sum();
```

3.8 Ukládání dat a transakce

Vytvoření nového objektu

```
Osoba osoba = new Osoba(XpoDefault.Session);
osoba.Jmeno = "Jan";
osoba.Prijmeni = "Becher";
osoba.Save();
```

```
declare @P1 int
set @P1=9
declare @P2 int
set @P2=1
exec sp_executesql N'insert into "dbo"."Osoba"
("Jmeno","Prijmeni","OptimisticLockField","GCRecord","ObjectType")
values(@p1,@p2,@p3,null,@p4) set
@p0=SCOPE_IDENTITY() set @r=1', N'@p0 int output,@p1 nvarchar(4000),
@p2 nvarchar(4000),@p3 int,@p4 int,@r int output', @p0 = @P1 output, @p1 =
N'Jan', @p2 = N'Becher', @p3 = 0, @p4 = 1, @r = @P2 output
select @P1, @P2
```

Změna a uložení stávajících dat

```
XPQuery<Osoba> qry = new XPQuery<Osoba>(XpoDefault.Session);
Osoba becher = (from osoba in qry
                    where osoba.Prijmeni == "Becher"
                    && osoba.Jmeno == "Jan" select osoba).First();
becher.Jmeno = "Johan";
becher.Save();
```

Smazání objektu

```
XPQuery<Osoba> qry = new XPQuery<Osoba>(XpoDefault.Session);
Osoba becher = (from osoba in qry
                    where osoba.Prijmeni == "Becher"
                    && osoba.Jmeno == "Jan" select osoba).First();

becher.Delete();
```

U objektů odvozených ze třídy XPObject je implemtována tzv. odložený delete. Odpovídající řádek v tabulce není fyzicky smazán, ale označen jako smazaný.

OID	Jmeno	Prijmeni	GCRecord
1	Jan	Becher	462662090

Řádky označené jako smazané, lze fyzicky odstranit voláním metody PurgeDeletedObjects objektu třídy Session:

```
XpoDefault.Session.PurgeDeletedObjects();
```

Transakce

```
XPQuery<Objednavka> qry = new XPQuery<Objednavka>(XpoDefault.Session);
IEnumerable<Objednavka> objednavky =
    (from obj in qry where obj.Cena < 1000 select obj);

XpoDefault.Session.BeginTransaction();

try
{
    foreach (Objednavka objednavka in objednavky)
    {
        objednavka.Cena += 500;
        objednavka.Save();
    }

    XpoDefault.Session.CommitTransaction();
}
catch (Exception)
{
    XpoDefault.Session.RollbackTransaction();
    throw;
}
```