Эта статья продолжение <http://infostart.ru/public/393228/>

Там же лежат и обработка для генерации C# файлов.

Суть та же, что и для 7.7 , но есть особенности.

Начнем с класса предка.

public class СсылочныйТип

{

[Key]

[Column("\_IDRRef")]

[MaxLength(16)]

public byte[] ID { get; set; }

[NotMapped]

public virtual int НомерТаблицы { get { return 0; } }

}

Номер таблицы нам понадобится для неопределенных типов.

Кроме того мы сможем сделать дженерик метод

public TEntity ПолучитьСсылочныйЭлемент<TEntity>(byte[] ID) where TEntity : СсылочныйТип

{

var query = from спр in this.Set<TEntity>()

where спр.ID == ID

select спр;

return query.SingleOrDefault<TEntity>();

}

Пока реализованы справочники, табличные части и перечисления

Это описание Справочника предка

public abstract class СправочникПредок : СсылочныйТип

{

[Column("\_Version", TypeName = "timestamp")]

[MaxLength(8)]

[Timestamp]

public byte[] Версия { get; set; }

[Column("\_Marked")]

[Required]

[MaxLength(1)]

public byte[] ПометкаУдаленияId { get; set; }

[Column("\_PREDEFINEDID")]

[Required]

[MaxLength(16)]

public byte[] ПредопределенныйId { get; set; }

[NotMapped]

public bool ПометкаУдаления

{

get { return ПометкаУдаленияId.ВБулево(); }

set

{

ПометкаУдаленияId = value.БулевоВМассив();

}

}

public abstract string Наименование { get; set; }

public abstract byte[] ЭтоГруппаId { get; set; }

public abstract object ПолучитьКод();

public abstract string Вид();

public bool ЭтоГруппа() { return ЭтоГруппаId.ВБулево(); }

}

А Это табличной части

public partial class ТабличнаяЧастьПредок

{

virtual public byte[] СсылкаID { get; set; }

[Key]

[Column("\_KeyField", Order = 1)]

[MaxLength(4)]

public byte[] KeyField { get; set; }

virtual public int НомерСтроки { get; set; }

}

На основании этих предков можно уже создавать потомков

Я сделал несколько тетовых справочников со всевозможными типами.

Например

[Table("\_Reference10")]

public partial class Тестовый:СправочникПредок

{

public const int НомТаблицы=10;

public const string ВидСпр="Тестовый";

[NotMapped]

override public byte[] ЭтоГруппаId { get{return КонстантыБД.Ложь;} set{} }

[Column("\_CODE",TypeName = "nvarchar")]

[Required]

[StringLength(9)]

public string Код { get; set; }

[Column("\_DESCRIPTION",TypeName = "nvarchar")]

[Required]

[StringLength(25)]

override public string Наименование { get; set; }

[Column("\_Fld12",TypeName = "nvarchar")]

[StringLength(20)]

public string Строка { get; set; }

[Column("\_Fld46",TypeName = "nchar")]

[StringLength(10)]

public string СтрокаФиксированная { get; set; }

[Column("\_Fld47",TypeName = "ntext")]

public string СтрокаДлинная { get; set; }

[Column("\_Fld13",TypeName = "numeric")]

[Required]

public System.Decimal Число { get; set; }

[Column("\_Fld14",TypeName = "numeric")]

[Required]

public System.Int64 Целое { get; set; }

[Column("\_Fld15")]

[Required]

public DateTime Дата { get; set; }

[Column("\_Fld16")]

[Required]

[MaxLength(1)]

public byte[] БулевоId { get; set; }

[Column("\_Fld17",TypeName = "image")]

[Required]

public byte[] ХранилищеЗначения { get; set; }

[Column("\_Fld18")]

[Required]

[MaxLength(16)]

public byte[] УникальныйИдентификатор { get; set; }

public НеопределенныеТипы.ЛюбаяСсылка\_Fld19 ЛюбаяСсылкаId { get; set; }

[Column("\_Fld20RRef")]

[Required]

[MaxLength(16)]

public byte[] ПеречислениеДляТестовId { get; set; }

public НеопределенныеТипы.РевизитВсеТипы\_Fld37 РевизитВсеТипыId { get; set; }

public НеопределенныеТипы.НеопределенныйНеССылочный\_Fld40 НеопределенныйНеССылочныйId { get; set; }

[Column("\_Fld48RRef")]

[Required]

[MaxLength(16)]

public byte[] РеквизитСправочникId { get; set; }

[Column("\_Fld49RRef")]

[Required]

[MaxLength(16)]

public byte[] РеквизитДокументId { get; set; }

override public string ToString(){return Код.ToString();}

[NotMapped]

public bool Булево {

get { return БулевоId.ВБулево(); }

set

{

БулевоId = value.БулевоВМассив();

}

}

[NotMapped]

public object ЛюбаяСсылка { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(ЛюбаяСсылкаId);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,ЛюбаяСсылкаId);} }

virtual public Перечисление.ДляТестовСсылка ПеречислениеДляТестов { get; set; }

[NotMapped]

public object РевизитВсеТипы { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(РевизитВсеТипыId);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,РевизитВсеТипыId);} }

[NotMapped]

public object НеопределенныйНеССылочный { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(НеопределенныйНеССылочныйId);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,НеопределенныйНеССылочныйId);} }

virtual public Справочник.Номенклатура РеквизитСправочник { get; set; }

[InverseProperty("Владелец")]

public ICollection<Справочник.ПодчиненныйОдинВладелец> ПодчиненныеПодчиненныйОдинВладелец { get; set; }

[InverseProperty("Ссылка")]

public ICollection<ТЧ.ТабЧасть> ПодчиненныеТабЧасть { get; set; }

public override int НомерТаблицы { get { return НомТаблицы; } }

public override string Вид(){return ВидСпр;}

public override object ПолучитьКод(){return Код;}

}

В отличие от 7.7 здесь введены новые типы для неопределенных типов.

Например

[ComplexType]

public class РевизитВсеТипы\_Fld37:НеопределенныйТип,НеопределенныйТипБулево,НеопределенныйТипЧисло,НеопределенныйТипДата,НеопределенныйТипСтрока,НеопределенныйТипСсылочный

{

[Column("\_Fld37\_TYPE")]

[Required]

[MaxLength(1)]

override public byte[] Тип { get; set; }

[Column("\_Fld37\_L")]

[Required]

[MaxLength(1)]

public byte[] БулевоId { get; set; }

[Column("\_Fld37\_N",TypeName = "numeric")]

[Required]

public decimal Число { get; set; }

[Column("\_Fld37\_T")]

[Required]

public DateTime Дата { get; set; }

[Column("\_Fld37\_S",TypeName = "nvarchar")]

[StringLength(42)]

public string Строка { get; set; }

[Column("\_Fld37\_RTREF")]

[Required]

[MaxLength(4)]

public byte[] НомерТаблицы { get; set; }

[Column("\_Fld37\_RRREF")]

[Required]

[MaxLength(16)]

public byte[] Ссылка { get; set; }

[NotMapped]

public bool Булево {

get { return БулевоId.ВБулево(); }

set

{

БулевоId = value.БулевоВМассив();

}

}

}

Такие типы помечаются атрибутом [ComplexType]

Что позволяет использовать их так

public НеопределенныеТипы.РевизитВсеТипы\_Fld37 РевизитВсеТипыId { get; set; }

и получение и установка объекта

[NotMapped]

public object РевизитВсеТипы { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(РевизитВсеТипыId);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,РевизитВсеТипыId);} }

Так как состав типа может быть различным для получения и установки полей я ввел интерфейсы

public class НеопределенныйТип

{

virtual public byte[] Тип { get; set; }

}

public interface НеопределенныйТипЧисло

{

Decimal Число { get; set; }

}

public interface НеопределенныйТипСтрока

{

string Строка { get; set; }

}

public interface НеопределенныйТипБулево

{

bool Булево { get; set; }

}

public interface НеопределенныйТипДата

{

DateTime Дата { get; set; }

}

public interface НеопределенныйТипСсылочный

{

byte[] НомерТаблицы { get; set; }

byte[] Ссылка { get; set; }

}

Теперь можно написать методы чтения и установки объектов

public static object ПолучитьЗначениеНеопределенногоСсылочного(НеопределенныйТип ВладелецId)

{

var НТС=ВладелецId as НеопределенныйТипСсылочный;

Func<byte[], object> значение;

if (ОбъектыПоТипу.TryGetValue(НТС.НомерТаблицы.byteArrayToInt(), out значение))

return значение(НТС.Ссылка);

return null;

}

public static object ПолучитьЗначениеНеопределенногоТипа(НеопределенныйТип ВладелецId)

{

var тип = ВладелецId.Тип.ВБайт();

switch(тип)

{

case 1: return null;

case 2: return (ВладелецId as НеопределенныйТипБулево).Булево;

case 3: return (ВладелецId as НеопределенныйТипЧисло).Число;

case 4: return (ВладелецId as НеопределенныйТипДата).Дата;

case 5: return (ВладелецId as НеопределенныйТипСтрока).Строка;

case 8: return ПолучитьЗначениеНеопределенногоСсылочного(ВладелецId);

}

return null;

}

ОбъектыПоТипу это хэш таблица которя заполняется так

public partial class КонстантыБД

{

static КонстантыБД()

{

ОбъектыПоТипу = new Dictionary<int, Func<byte[], object>>();

ОбъектыПоТипу.Add(Справочник.Тестовый.НомТаблицы,

(ключ) => { return БД.ПолучитьСсылочныйЭлемент<Справочник.Тестовый>(ключ); }

);

ОбъектыПоТипу.Add(Справочник.Номенклатура.НомТаблицы,

(ключ) => { return БД.ПолучитьСсылочныйЭлемент<Справочник.Номенклатура>(ключ); }

);

ОбъектыПоТипу.Add(Справочник.ПодчиненныйОдинВладелец.НомТаблицы,

(ключ) => { return БД.ПолучитьСсылочныйЭлемент<Справочник.ПодчиненныйОдинВладелец>(ключ); }

);

ОбъектыПоТипу.Add(Справочник.ПодчиненныйДваВладельца.НомТаблицы,

(ключ) => { return БД.ПолучитьСсылочныйЭлемент<Справочник.ПодчиненныйДваВладельца>(ключ); }

);

ОбъектыПоТипу.Add(Справочник.Справочник1.НомТаблицы,

(ключ) => { return БД.ПолучитьСсылочныйЭлемент<Справочник.Справочник1>(ключ); }

);

ОбъектыПоТипу.Add(Перечисление.ДляТестовСсылка.НомТаблицы,

(ключ) => { var значение = БД.ПолучитьСсылочныйЭлемент<Перечисление.ДляТестовСсылка>(ключ);

if (значение == null)

return null;

return значение.Значение;

}

);

}

}

Ну и реализация установки значений

public static void УстановитьЗначениеНеопределенногоТипа(object value,НеопределенныйТип ВладелецId)

{

ОчиститьСтароеЗначениеНеопределенного(ВладелецId);

if (value==null)

{

ВладелецId.Тип[0] =1;

return;

}

var тип=value.GetType();

if (тип==typeof(bool))

{

ВладелецId.Тип[0] =2;

(ВладелецId as НеопределенныйТипБулево).Булево = (bool)value;

}

else if (тип == typeof(decimal))

{

ВладелецId.Тип[0]=3;

(ВладелецId as НеопределенныйТипЧисло).Число = (decimal)value;

}

else if (тип == typeof(DateTime))

{

ВладелецId.Тип[0] = 4;

(ВладелецId as НеопределенныйТипДата).Дата = (DateTime)value;

}

else if (тип == typeof(string))

{

ВладелецId.Тип[0] =5;

(ВладелецId as НеопределенныйТипСтрока).Строка = (string)value;

}

else if (value is СсылочныйТип)

{

ВладелецId.Тип[0] = 8;

var значение = value as СсылочныйТип;

var Владелец = ВладелецId as НеопределенныйТипСсылочный;

Владелец.НомерТаблицы = значение.НомерТаблицы.intToByteArray();

Владелец.Ссылка = значение.ID;

}

}

Еще одна особенность это описание перечислений

namespace Linq21c8.Перечисление

{

[Table("\_Enum11")]

public partial class ДляТестовСсылка : СсылочныйТип

{

[Column("\_ENUMORDER", TypeName = "numeric")]

[Required]

public ДляТестов Значение { get; set; }

public const int НомТаблицы = 11;

public override int НомерТаблицы { get { return НомТаблицы; } }

}

public enum ДляТестов

{

Первый // Первый

, Второй // Второй

}

public static class РасширениеПеречисления

{

public static ДляТестовСсылка ПолучитьСсылку(this ДляТестов ID)

{

var БД = КонстантыБД.БД;

var query = from спр in БД.ДляТестов\_Перечисл

where спр.Значение == ID

select спр;

return query.SingleOrDefault();

}

}

}

Так в Code Firs числовому значению можно задать тип перечисления

public ДляТестов Значение { get; set; }

Кроме того реализован класс для расширения перечисления

Пример использования

var перечисление\_Первый = Перечисление.ДляТестов.Первый.ПолучитьСсылку();

var перечисление\_Второй = Перечисление.ДляТестов.Второй.ПолучитьСсылку();

foreach (var спр in qr2)

{

спр.СтрокаФиксированная = "";

спр.Целое = ++целое;

if (целое % 2 == 1)

спр.ПеречислениеДляТестов = перечисление\_Первый;

else

спр.ПеречислениеДляТестов = перечисление\_Второй;

бд.Entry(спр).State = System.Data.Entity.EntityState.Modified;

}

Так же можно задавать условие

var qr2 = from Номенклатура in бд.Тестовый\_Спр

where Номенклатура.ПеречислениеДляТестов.Значение==Перечисление.ДляТестов.Второй

select Номенклатура;

Есть особенность для ComplexType

public partial class Тестовый : СправочникПредок

{

public partial class НеопределенныеТипы

{

[ComplexType]

public class ЛюбаяСсылка\_Fld19 : НеопределенныйТип, НеопределенныйТипСсылочный

{

[Column("\_Fld19\_TYPE")]

[Required]

[MaxLength(1)]

override public byte[] Тип { get; set; }

[Column("\_Fld19\_RTREF")]

[Required]

[MaxLength(4)]

public byte[] НомерТаблицы { get; set; }

[Column("\_Fld19\_RRREF")]

[Required]

[MaxLength(16)]

public byte[] Ссылка { get; set; }

}

}

}

public partial class Тестовый : СправочникПредок

{

public partial class ТЧ

{

public partial class НеопределенныеТипы

{

[ComplexType]

public class ЛюбаяСсылка\_Fld52 : НеопределенныйТип, НеопределенныйТипСсылочный

{

[Column("\_Fld52\_TYPE")]

[Required]

[MaxLength(1)]

override public byte[] Тип { get; set; }

[Column("\_Fld52\_RTREF")]

[Required]

[MaxLength(4)]

public byte[] НомерТаблицы { get; set; }

[Column("\_Fld52\_RRREF")]

[Required]

[MaxLength(16)]

public byte[] Ссылка { get; set; }

}

}

}

}

Есть поле в разных типах ЛюбаяСсылка, но в базе все названия [ComplexType] должны быть уникальными.  
Поэтому добавляю номер таблицы для неопределенных владельцев и номер поля для неопределенных полей

Теперь посмотрим на реализацию табличной части

public partial class ТабличнаяЧастьПредок

{

virtual public byte[] СсылкаID { get; set; }

[Key]

[Column("\_KeyField", Order = 1)]

[MaxLength(4)]

public byte[] KeyField { get; set; }

virtual public int НомерСтроки { get; set; }

}

[Table("\_Reference10\_VT31")]

public partial class ТабЧасть:ТабличнаяЧастьПредок

{

[Key]

[Column("\_Reference10\_IDRRef", Order = 0)]

[MaxLength(16)]

override public byte[] СсылкаID { get; set; }

[Key]

[Column("\_LineNo32", Order = 2, TypeName = "numeric")]

override public int НомерСтроки { get; set; }

[Column("\_Fld33RRef")]

[Required]

[MaxLength(16)]

public byte[] НоменклатураId { get; set; }

[Column("\_Fld34",TypeName = "numeric")]

[Required]

public System.Int64 Количество { get; set; }

[Column("\_Fld35",TypeName = "numeric")]

[Required]

public System.Decimal Сумма { get; set; }

public НеопределенныеТипы.ЛюбаяСсылка\_Fld52 ЛюбаяСсылкаId { get; set; }

virtual public Справочник.Тестовый Ссылка { get; set; }

virtual public Справочник.Номенклатура Номенклатура { get; set; }

[NotMapped]

public object ЛюбаяСсылка { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(ЛюбаяСсылкаId);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,ЛюбаяСсылкаId);} }

}

Так в ТЧ три ключевых поля СсылкаID, KeyField и НомерСтроки

Ссылка это владелец табличной части.

Основной метод создающий описание полей выглядит так

Процедура ПрописатьРеквизит(тип,ИмяСвойства,ПолеБД,Реквизит,ТекстПолей, ТекстВиртуальных,ТипВладельца)  
    ИмяРеквизита=ИмяСвойства;  
    СтрВиртуал ="";  
    Required="  
    |[Required]";  
    Стр="";  
    TypeName="";  
    Если Тип=Тип("Булево") Тогда  
        Стр=  "[MaxLength(1)]";  
        ИмяРеквизита=ИмяСвойства+"Id";  
        ТипРеквизита = "byte[]";  
        СтрВиртуал = СоздатьВиртуальноеБулево(ИмяСвойства);  
  
  
  
    ИначеЕсли Тип=Тип("Число") Тогда  
        TypeName="numeric";  
        ТипРеквизита =  ПолучитьТипЧисла(Реквизит.Тип);  
  
    ИначеЕсли Тип=Тип("Дата") Тогда  
  
        ТипРеквизита =  "DateTime";  
  
  
    ИначеЕсли Тип=Тип("Строка") Тогда  
  
        КвалификаторыСтроки=Реквизит.Тип.КвалификаторыСтроки;  
  
        ДлинаСтроки =  КвалификаторыСтроки.Длина;  
        Стр =  "[StringLength("+ДлинаСтроки+")]";  
        Если ДлинаСтроки=0 Тогда  
  
            TypeName="ntext";  
            Required="";  
            Стр = "";  
        ИначеЕсли КвалификаторыСтроки.ДопустимаяДлина=ДопустимаяДлина.Фиксированная Тогда  
            TypeName="nchar";  
            Required="";  
  
  
        Иначе  
            Required="";  
            TypeName="nvarchar";  
        КонецЕсли;  
  
  
  
        ТипРеквизита =  "string";  
  
    ИначеЕсли Тип=Тип("ХранилищеЗначения") Тогда  
        TypeName = "image";  
        ТипРеквизита = "byte[]";  
  
    ИначеЕсли Тип=Тип("УникальныйИдентификатор") Тогда  
        Стр=  "[MaxLength(16)]";  
        ТипРеквизита = "byte[]";  
    ИначеЕсли  Справочники.ТипВсеСсылки().СодержитТип(Тип) Тогда  
        ИмяРеквизита=ИмяСвойства+"Id";  
        стр="[MaxLength(16)]";  
        ТипРеквизита = "byte[]";  
        Вид=Метаданные.НайтиПоТипу(Тип).Имя;  
        ТипВид=Метаданные.НайтиПоТипу(Тип).ПолноеИмя();  
        СтрВиртуал="virtual public Справочник."+Вид+" "+ИмяСвойства+" { get; set; }";  
        // Когда вид реквизита видом самого справочника добавим коллекцию на подчиненные элементы  
        Если ТипВид=ТипВладельца Тогда  
            СтрВиртуал=СтрВиртуал+"  
            |[InverseProperty("""+ИмяСвойства+""")]  
            |public ICollection<Справочник."+Вид+"> ПодчиненныеДля"+ИмяСвойства+" { get; set; }";  
  
        КонецЕсли;  
    ИначеЕсли  Перечисления.ТипВсеСсылки().СодержитТип(Тип) Тогда  
        ИмяРеквизита=ИмяСвойства+"Id";  
        стр="[MaxLength(16)]";  
        ТипРеквизита = "byte[]";  
        Вид=Метаданные.НайтиПоТипу(Тип).Имя;  
        СтрВиртуал="virtual public Перечисление."+Вид+"Ссылка "+ИмяСвойства+" { get; set; }";  
  
    Иначе  
        // Ссылочный тип Который пока не реализован  
        ИмяРеквизита=ИмяСвойства+"Id";  
        ТипВид=Метаданные.НайтиПоТипу(Тип).ПолноеИмя();  
        стр="[MaxLength(16)]";  
        ТипРеквизита = "byte[]";  
        Вид=Метаданные.НайтиПоТипу(Тип).Имя;  
        //  СтрВиртуал="virtual public "+ТипВид+" "+ИмяСвойства+" { get; set; }";  
  
    КонецЕсли;  
  
    Если TypeName<>"" Тогда  
        Стр1=  "[Column("""+ПолеБД+""",TypeName = """+TypeName+""")]";  
  
    Иначе  
        Стр1=  "[Column("""+ПолеБД+""")]";  
  
  
  
    КонецЕсли;  
    Если Стр<>""  Тогда  
  
        Стр="  
        |"+Стр;  
  
    КонецЕсли;  
  
    Стр=Стр1+Required+стр+"  
    |public "+ТипРеквизита+" "+ИмяРеквизита+ " { get; set; }";  
  
  
    ТекстПолей.ДобавитьСтроку("");  
    ТекстПолей.ДобавитьСтроку(Стр);  
    Если СтрВиртуал <>"" Тогда  
  
        ТекстВиртуальных.ДобавитьСтроку(СтрВиртуал);  
  
    КонецЕсли;  
  
  
КонецПроцедуры  
  
  
  
Процедура ДобавитьВиртНеопред(ИмяСвойства,Текст)  
  
Стр="   [NotMapped]  
|public object %1 { get{return КонстантыБД.ПолучитьЗначениеНеопределенногоТипа(%1Id);} set{КонстантыБД.УстановитьЗначениеНеопределенногоТипа(value,%1Id);} }";  
  
Текст.ДобавитьСтроку(СтрШаблон(Стр,ИмяСвойства));  
  
КонецПроцедуры  
  
Функция ПрописьРеквизитовПоКоллекции(ТипВладельца,Коллекция,Поля,ТекстПолей,ТекстВиртуальных,ТекстНеопределенныхТипов)  
    Перем ИмяКлассаНТ;  
  
    Для каждого Реквизит Из Коллекция Цикл  
        ИмяРеквизита = Реквизит.Имя;  
        ПолеБД=ПолучитьПолеБД(Поля,ИмяРеквизита);  
        Сообщить("ИмяРеквизита="+ИмяРеквизита+"  ПолеБД="+ПолеБД);  
        Типы=Реквизит.Тип.Типы();  
        Если Типы.Количество()=1 Тогда  
            Тип=Типы[0];  
  
            ПрописатьРеквизит(тип,ИмяРеквизита,ПолеБД,Реквизит,ТекстПолей, ТекстВиртуальных,ТипВладельца)  
  
        Иначе  
            СоздатьНеопределенныйТип(ИмяРеквизита,ПолеБД,Реквизит.Тип,ТипВладельца,ТекстНеопределенныхТипов,ИмяКлассаНТ);  
            ТекстПолей.ДобавитьСтроку("public НеопределенныеТипы."+ИмяКлассаНТ+" "+ИмяРеквизита+"Id { get; set; }");  
             ДобавитьВиртНеопред(ИмяРеквизита,ТекстВиртуальных);  
  
  
        КонецЕсли;  
  
  
    КонецЦикла;  
  
  
КонецФункции