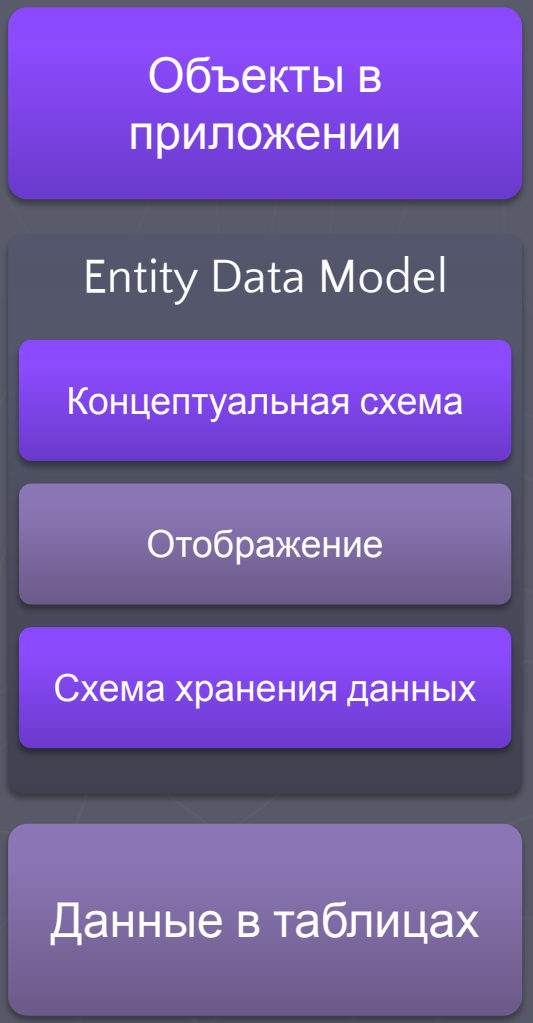
Доступ к данным на основе ADO.NET Entity Framework 4.0

Содержание

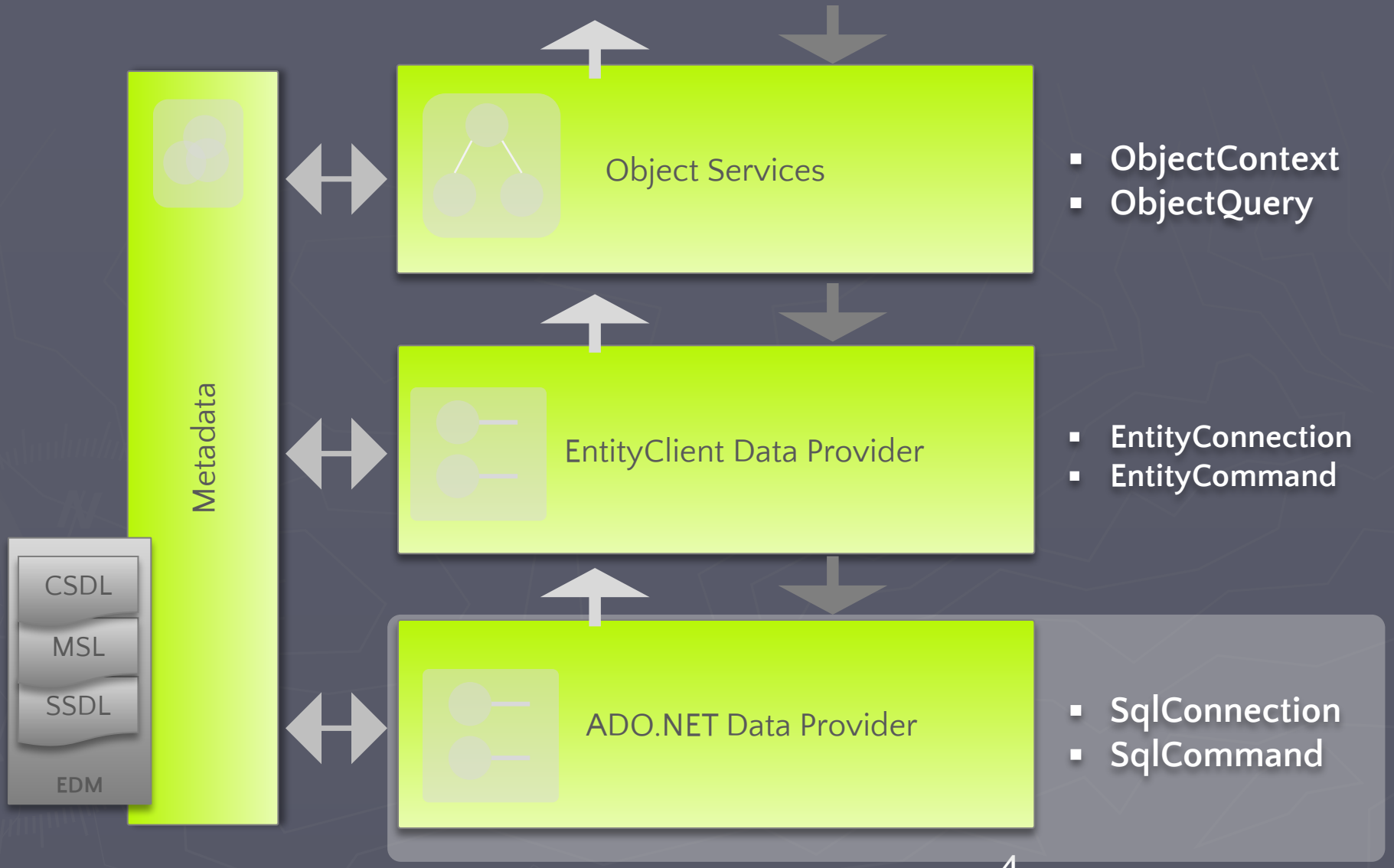
- Введение в Entity Framework
- Опыт первой версии
- Что необходимо разработчику?
- Практические сценарии

ADO.NET Entity Framework

- Работа с данными как с объектами
- Объектные модели могут быть сложные
- Схемы хранения данных могут быть сложные
- Модели и схемы хранения данных могут изменяться



Архитектура Entity Framework

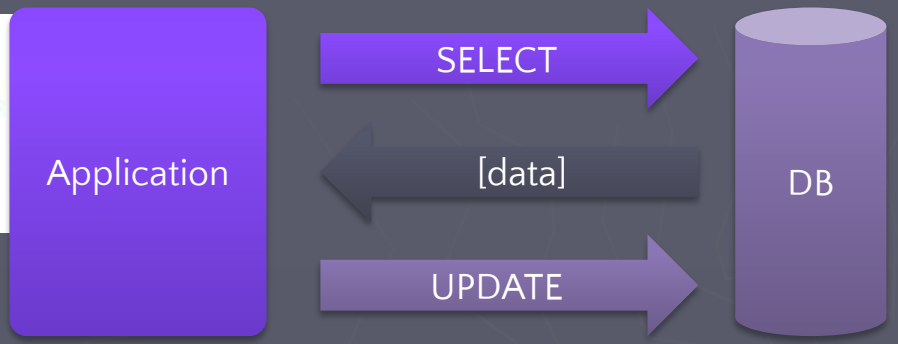


Опыт первой версии

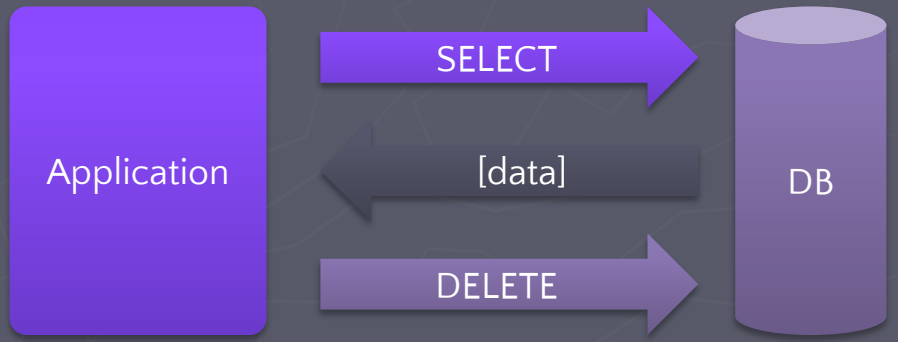
- Дизайнер
- Борьба за чистоту кода
  - Model First
  - POCO
  - Code Only
  - N-Tier?
- Работа с СУБД
  - Поддержка хранимых процедур
  - Поддержка хранимых функций
  - Поддержка LINQ
- Быстродействие
  - Лишние обращения к СУБД
  - Неэффективный SQL

Избыточные запросы

```
void ChangePerson()
{
    var person = context.Peoples.Where( p => p.Id == id).Single();
    person.Country = "RU";
    context.SaveChanges();
}
```

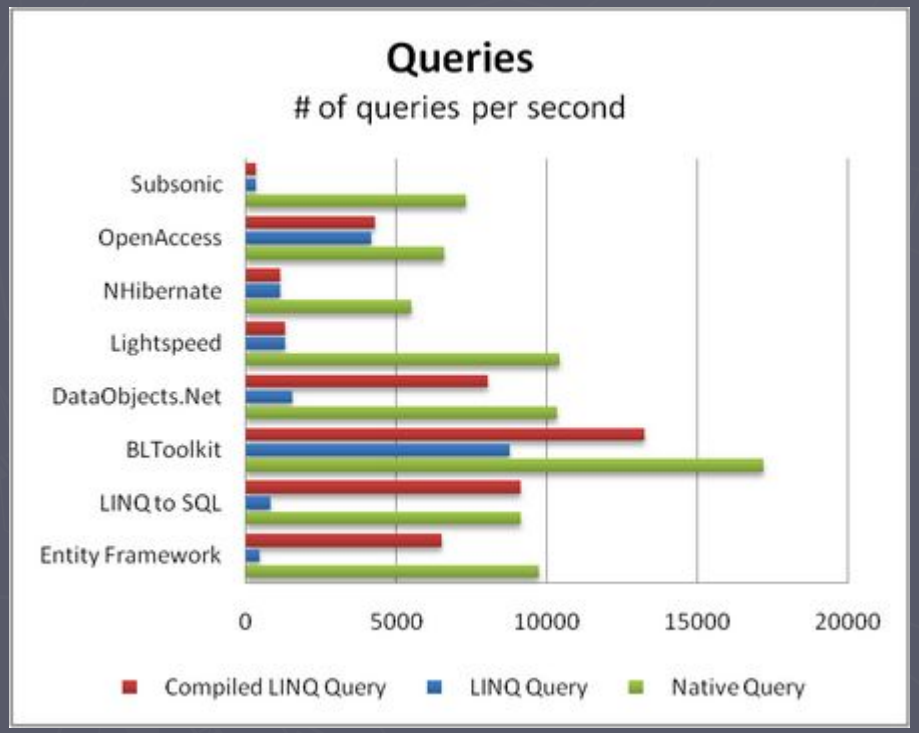


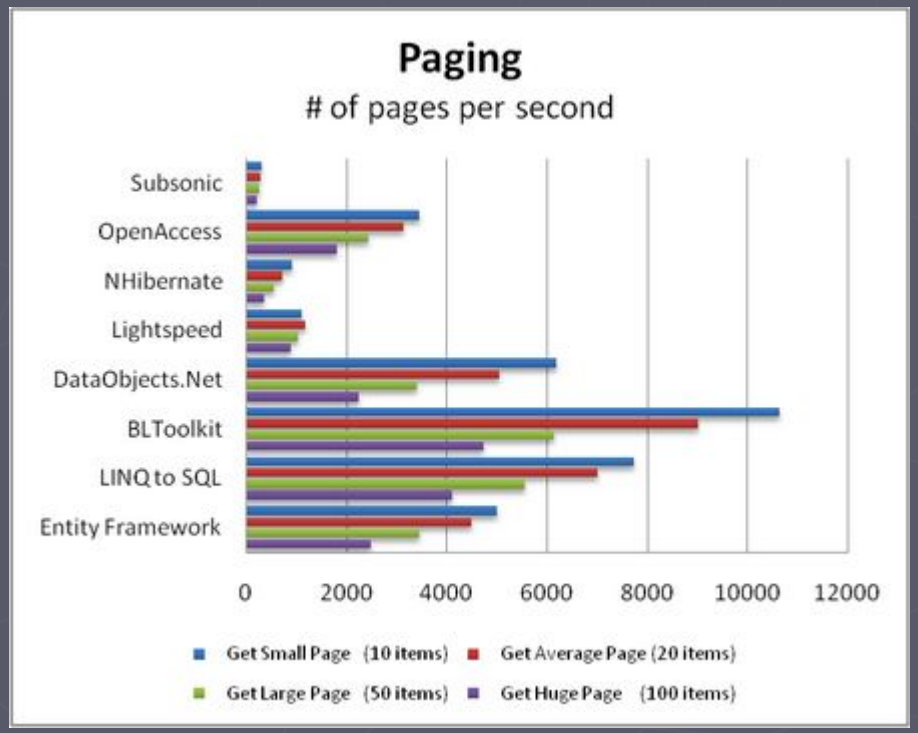

```
void DeletePerson()
{
    var person = context.Peoples.Where( p => p.Id == id).Single();
    context.DeleteObject(person);
    context.SaveChanges();
}
```



Быстродействие

http://www.ormbattle.net/

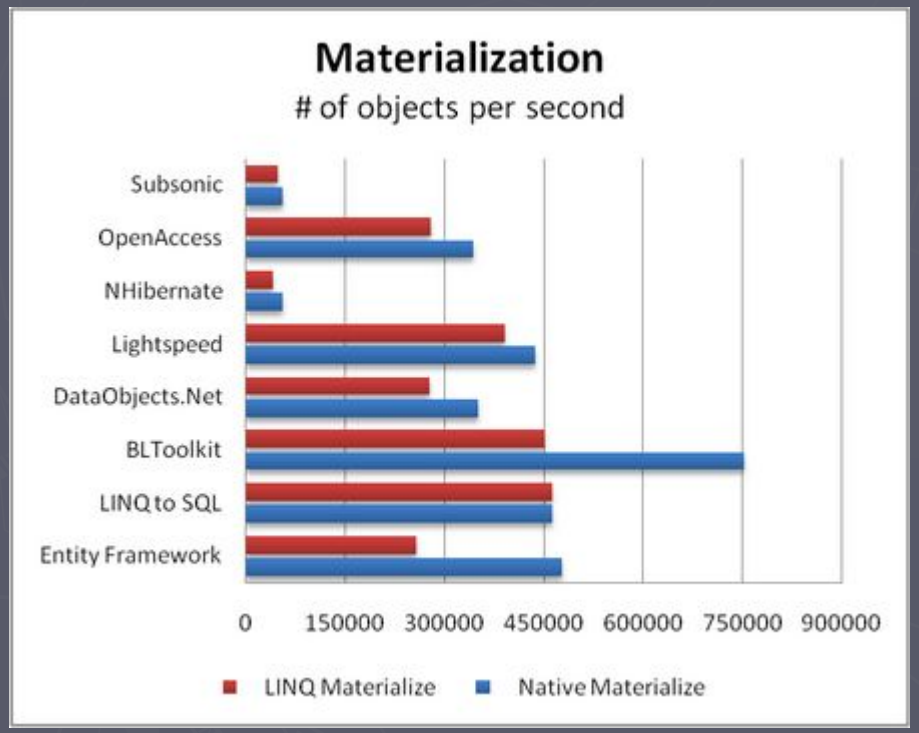


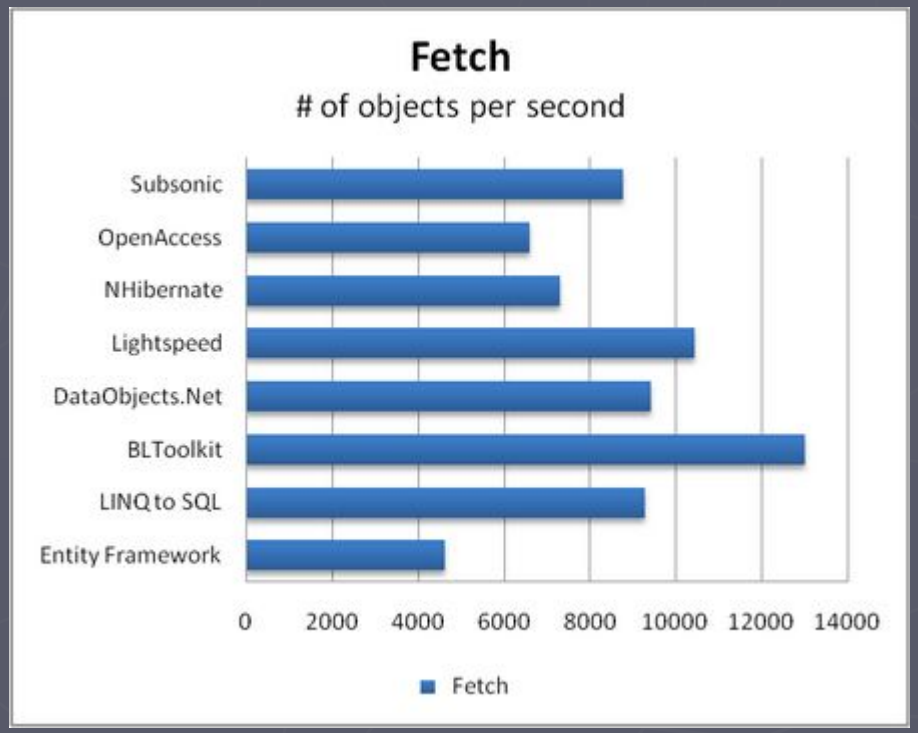


* ADO.NET Entity Framework v.1.0

Быстродействие

http://www.ormbattle.net/

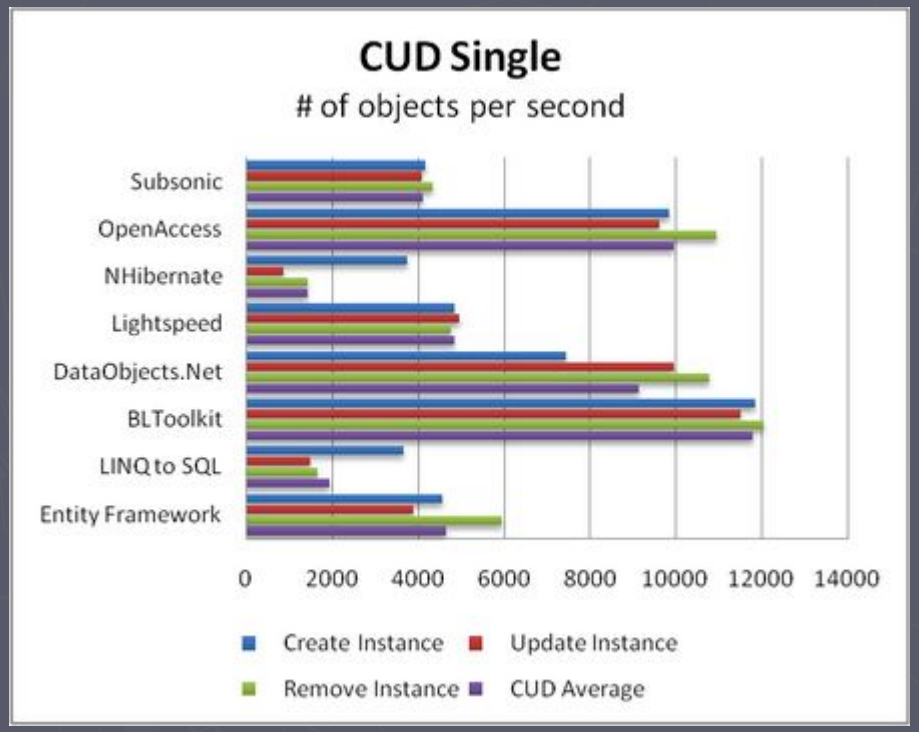


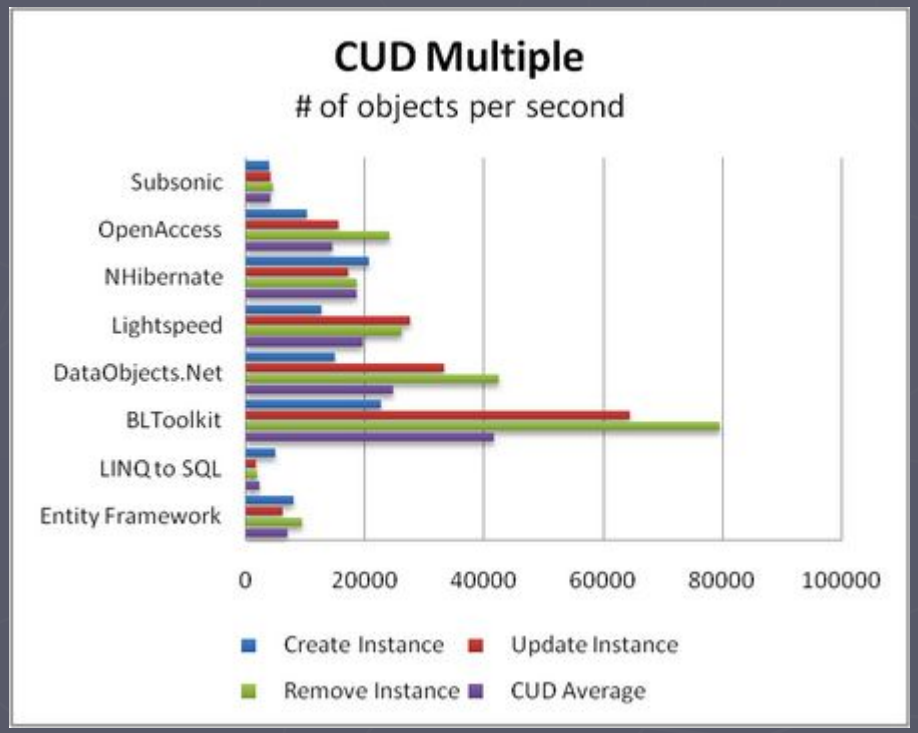


* ADO.NET Entity Framework v.1.0

Быстродействие

http://www.ormbattle.net/





* ADO.NET Entity Framework v.1.0

Entity Framework v.4

- Улучшения в дизайнере
- Разработка
  - Подход «Model First»
  - Подход «Code Only»
  - Поддержка POCO
  - Lazy loading
  - Поддержка многоуровневых приложений
- СУБД
  - Поддержка хранимых процедур и функций
  - Улучшенная поддержка LINQ
  - Собственные правила генерации скрипта по созданию схемы данных
- Быстродействие
  - Исключение лишних обращений к БД при обновлении
  - Более эффективные SQL-запросы

Хранимые функции

```
class Program
{
    static void Main(string[] args)
    {
        using (var context = new TheMyModelContainer())
        {
            foreach (var item in context.Teachers.Select(t => Functions.MyFunc1(t.FullName)))
            {
                Console.WriteLine(item);
            }
        }

        Console.Read();
    }
}

static class Functions
{
    [EdmFunction("DD03Model.Store", "Func1")]
    public static string MyFunc1(string s)
    {
        throw new InvalidOperationException();
    }
}
```

Хранимые процедуры

Сгенерированная сущность

```
[EdmComplexTypeAttribute(NamespaceName="DD03Model", Name="SP1_Result")]
[DataContractAttribute(IsReference=true)]
[Serializable()]
public partial class SP1_Result : ComplexObject
{
    #region Factory Method

    /// <summary>
    /// Create a new SP1_Result object.
    /// </summary>
    /// <param name="id">Initial value of the Id property.</param>
    public static SP1_Result CreateSP1_Result(global::System.Int32 id)
    {
        SP1_Result sP1_Result = new SP1_Result();
        sP1_Result.Id = id;
        return sP1_Result;
    }
```

Сгенерированная процедура

```
public ObjectResult<SP1_Result> SP1()
{
    return base.ExecuteFunction<SP1_Result>("SP1");
}
```

Использование

```
using (var context = new TheMyModelContainer())
{
    foreach (var item in context.SP1())
    {
        Console.WriteLine("{0}\t{1}", item.Id, item.Name);
    }
}

Console.Read();
```

Генерация SQL в первой версии

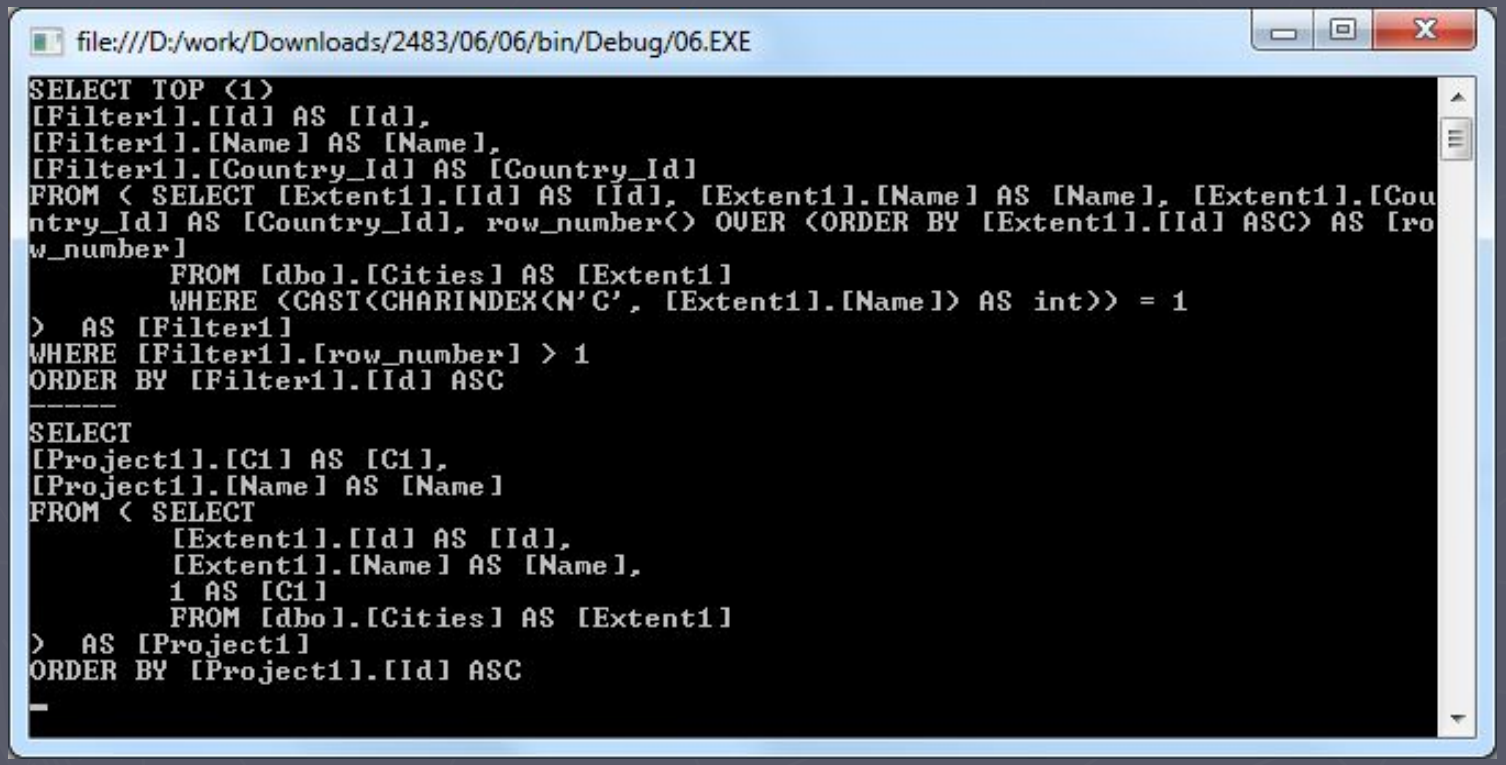
Исходный код

```
using (var context = new TheMyModelContainer())
{
    var q1 = (from c in context.Cities
              where c.Name.StartsWith("C")
              orderby c.Id
              select c).Skip(1).Take(1);
    var q2 = (from c in context.Cities
               orderby c.Id
               select new { c.Name });

    Console.WriteLine(((ObjectQuery)q1).ToTraceString());
    Console.WriteLine("-----");
    Console.WriteLine(((ObjectQuery)q2).ToTraceString());
}
```

Генерация SQL в первой версии

Результат



Генерация SQL в версии v4

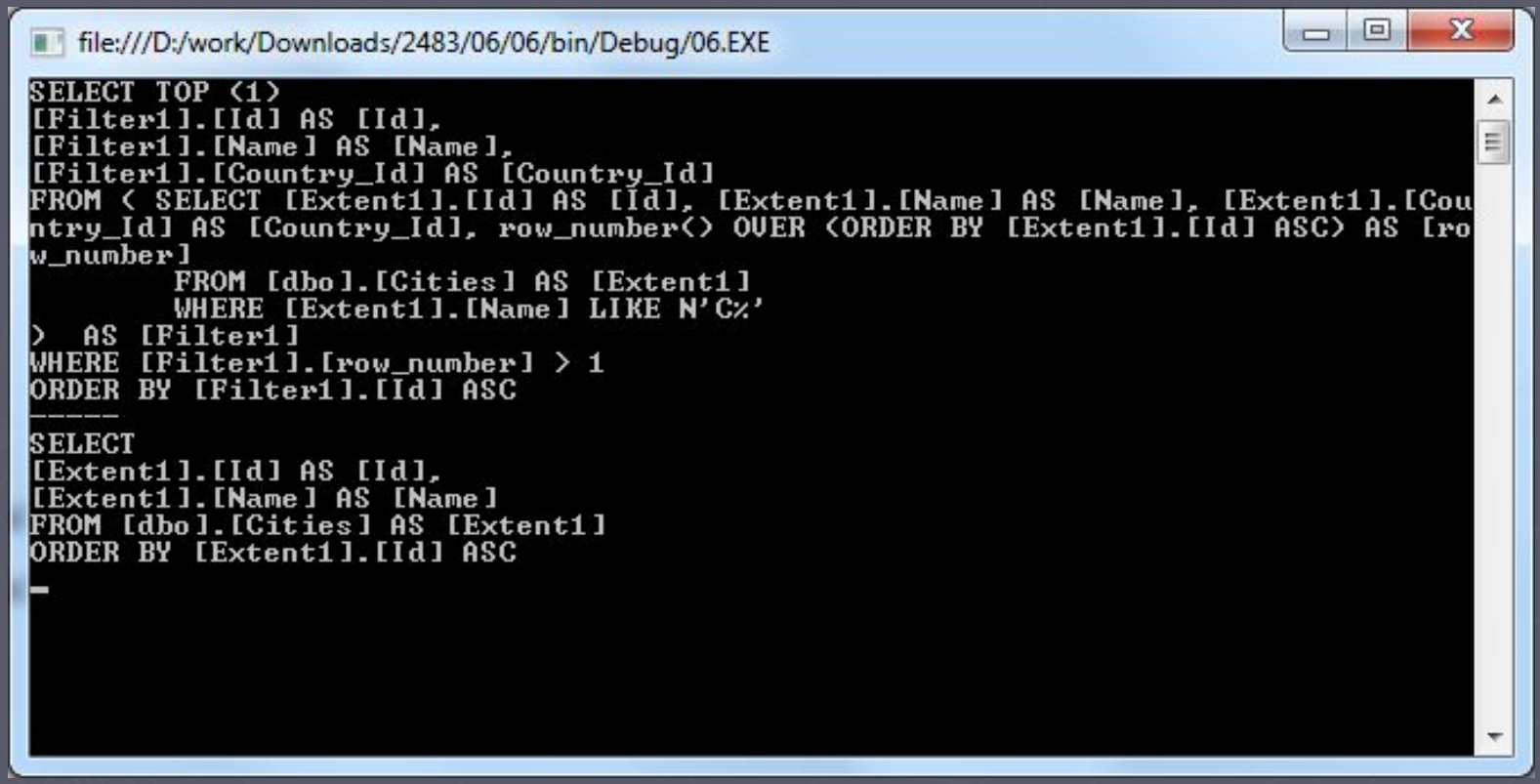
Исходный код

```
using (var context = new TheMyModelContainer())
{
    var q1 = (from c in context.Cities
              where c.Name.StartsWith("C")
              orderby c.Id
              select c).Skip(1).Take(1);
    var q2 = (from c in context.Cities
               orderby c.Id
               select new { c.Name });

    Console.WriteLine(((ObjectQuery)q1).ToTraceString());
    Console.WriteLine("-----");
    Console.WriteLine(((ObjectQuery)q2).ToTraceString());
}
```

Генерация SQL в версии v4

Результат



Применение Lazy Load

```
public class TheMyModelContainer : ObjectContext
{
    public TheMyModelContainer()
        : base("name=TheMyModelContainer")
    {
        ContextOptions.LazyLoadingEnabled = true;
    }

    public TheMyModelContainer(EntityConnection connection)
        : base(connection)
    {
        ContextOptions.LazyLoadingEnabled = true;
    }

    public ObjectSet<City> Cities { get { return CreateObjectSet<City>(); } }

    public ObjectSet<Country> Countries { get { return CreateObjectSet<Country>(); } }
}
```

Создание сущностей во время выполнения

```
var builder = new ContextBuilder<TheMyModelContainer>();

builder.Entity<Country>().Property(e => e.Id)
    .IsIdentity();
builder.Entity<Country>().Property(e => e.Name)
    .IsRequired()
    .HasMaxLength(255);

builder.Entity<Country>()
    .Relationship<City>(e => e.Cities);

builder.Entity<City>().Property(e => e.Id)
    .IsIdentity();
builder.Entity<City>().Property(e => e.Name)
    .IsRequired()
    .HasMaxLength(255);
```

Ресурсы

- Entity Framework Team Blog
  http://blogs.msdn.com/adonet/
- Entity Framework Design Blog
  http://blogs.msdn.com/efdesign/