The ADO.NET Entity Framework

Part I

Beyond Object Relational Mapping

Overview

- Background
- Models, Mapping, and Metadata
- Entity SQL and LINQ to Entities
- Object Services
- Compare LINQ to Entities with LINQ to SQL

Impedance Mismatch Redux

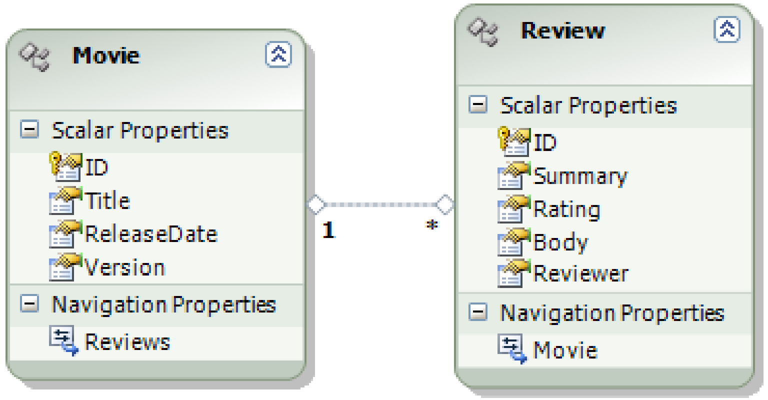
Objects	Databases
Built using OOP principles	Built using relational algebra
Use inheritance and aggregation	Requires data normalization
Link with references	Link with foreign keys
Identified by memory location	Identified by primary key
Use data types defined by runtime	Use datatypes defined by database
Can hold data in lists and trees	Can hold data in tuples
Not transactional (today)	Heavily transactional

Entity Framework

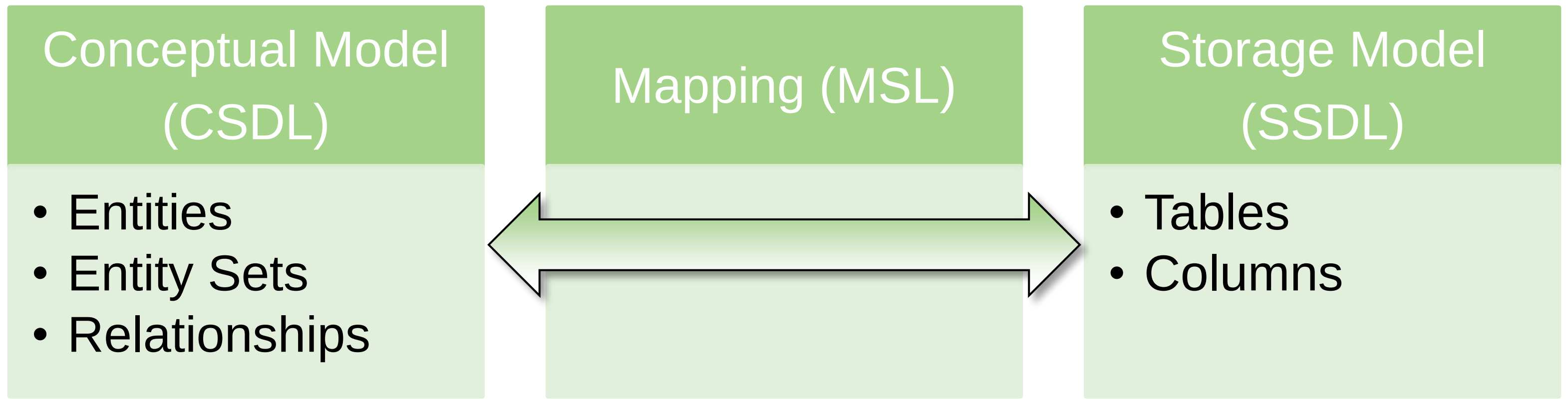
- **The new ADO.NET**
  - Higher level of abstraction than ADO.NET
  - Introduces the concept of an Entity Data Model
  - Vision goes beyond traditional ORM tools to provide “data services”
- **Features**
  - LINQ Provider
  - Visual Studio designer support
  - Flexible mapping
  - Data provider model (to support Oracle, DB2, etc)

Entity Focus

- **An Entity is**
  - An object we can persist
- **An Entity has**
  - An entity key that makes the entity uniquely identifiable
  - One or more scalar or complex properties
  - One or more relationships to other entities



The Entity Data Model

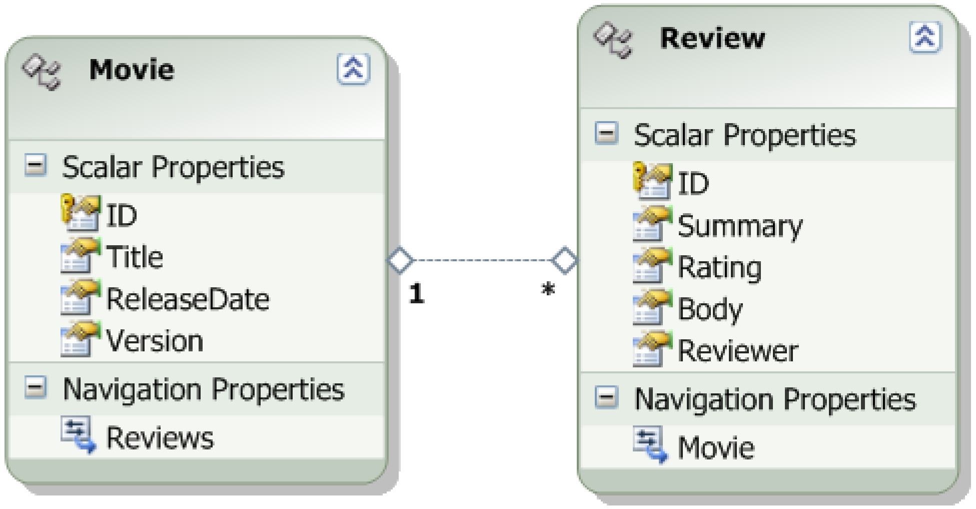


- Entities
- Entity Sets
- Relationships

- Tables
- Columns

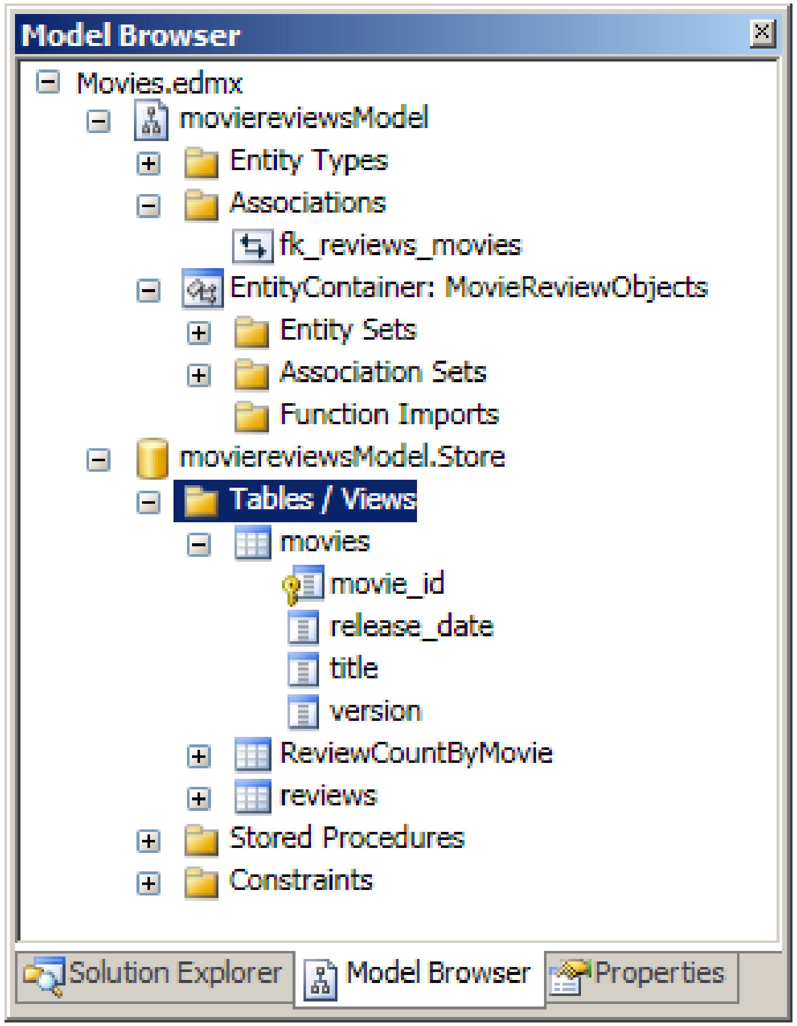
Entity Designer

- Create entities and relationships
- Define keys, types, nullability



Model Browser

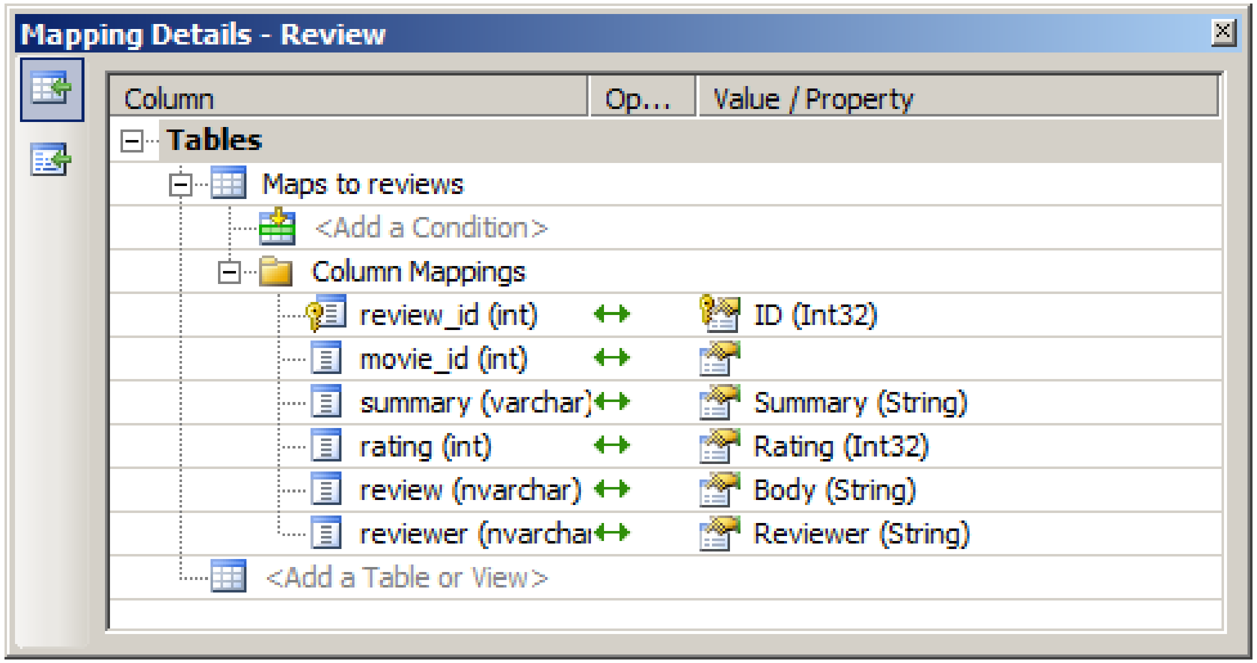
- **Browse model**
  - GUI can be difficult to navigate
- **Validate model**
- **Update model from database**



```
Movies.edmx
  moviereviewsModel
    Entity Types
    Associations
      fk_reviews_movies
    EntityContainer: MovieReviewObjects
      Entity Sets
      Association Sets
      Function Imports
  moviereviewsModel.Store
    Tables / Views
      movies
        movie_id
        release_date
        title
        version
      ReviewCountByMovie
      reviews
    Stored Procedures
    Constraints
```

Mapping Details

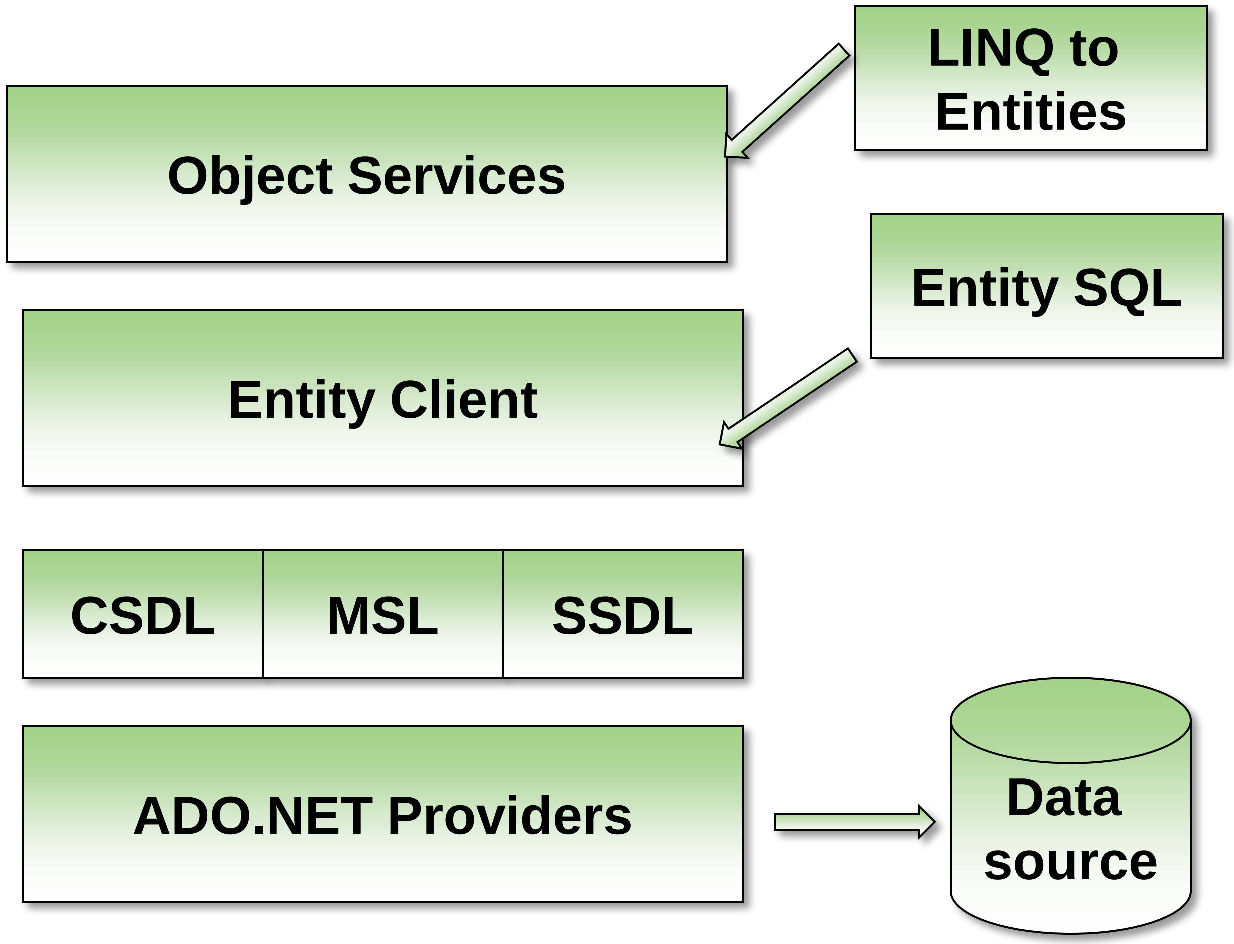
- Map entities across one or more tables
- Right-click on entity in design or browser and select “table mapping”



Mapping Details - Review

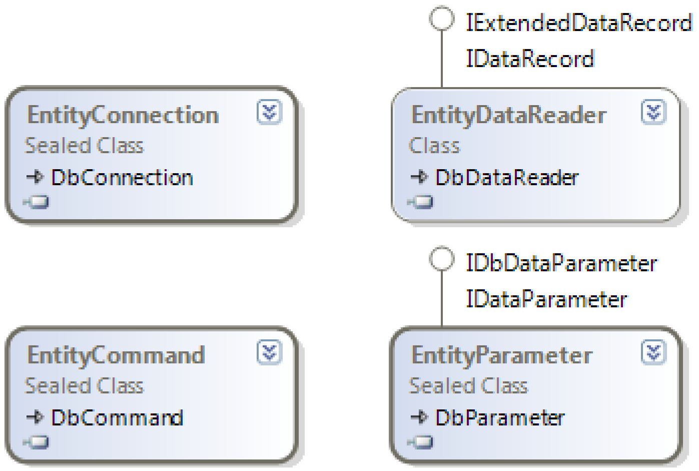
Column	Op...	Value / Property
Tables		
Maps to reviews		
<Add a Condition>		
Column Mappings		
review_id (int)	↔	ID (Int32)
movie_id (int)	↔	
summary (varchar)	↔	Summary (String)
rating (int)	↔	Rating (Int32)
review (nvarchar)	↔	Body (String)
reviewer (nvarchai	↔	Reviewer (String)
<Add a Table or View>		

Entity Framework Services



Entity Client

- **No more database specific constructs in ADO.NET code**
  - Queries sent to client as eSQL (Entity SQL)
- **Queries run against entity model, not the underlying storage model**
  - Entity client communicates with a database specific provider
- **Results can be consumed through a DbDataReader**



Entity SQL

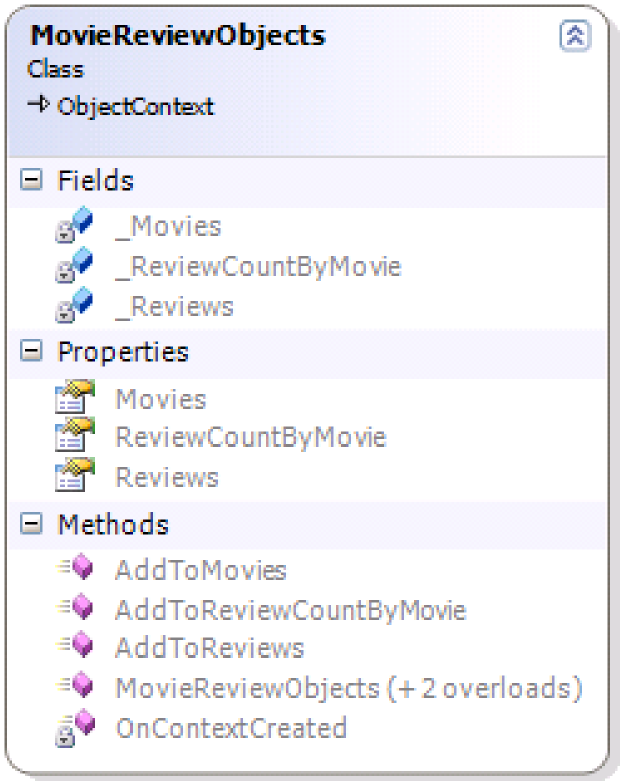
- Structured Query Language for the entity data model
- Provider neutral

```
using(MovieReviewObjects ctx =
         new MovieReviewObjects(connectionString))
{
    string command = "SELECT VALUE m FROM Movies as m";
    var movies = new ObjectQuery<Movie>(command, ctx);

    foreach (Movie m in movies)
    {
        Console.WriteLine(m.Title);
    }
}
```

ObjectContext

- **Gateway to all entities**
  - Relies on mapping and object metadata
- **Entities live inside Entity Sets**
  - Exposed as ObjectQuery<T> properties on the ObjectContext
  - ObjectQuery<T> implements IQueryable<T>
- **Materializes objects instead of returning a data reader**



```
MovieReviewObjects
Class
→ ObjectContext

Fields
  _Movies
  _ReviewCountByMovie
  _Reviews
Properties
  Movies
  ReviewCountByMovie
  Reviews
Methods
  AddToMovies
  AddToReviewCountByMovie
  AddToReviews
  MovieReviewObjects (+ 2 overloads)
  OnContextCreated
```

LINQ to Entities

- **Same standard operators and query syntax**

```
using (MovieReviewObjects context =
           new MovieReviewObjects(connectionString)) {
    var movies = from m in context.Movies
                 where m.Reviews.Count > 1
                 select m;

    foreach (var m in movies) {
        Console.WriteLine(m.Title);
        m.Reviews.Load();
        foreach (var r in m.Reviews) {
            Console.WriteLine("\t" + r.Summary);
        }
    }
}
```

Deferred Loading

- Entity Framework does use “lazy loading” for relationships
- Related entities must be explicitly loading using Load
- Can also eager load using an Include method on the ObjectContext

```
foreach (var m in movies) {
    Console.WriteLine(m.Title);
    m.Reviews.Load();
    foreach (var r in m.Reviews) {
        Console.WriteLine("\t" + r.Summary);
    }
}
```

Inserting Data

- Use AddObject to add any type of entity
- Strongly typed ObjectContext includes Add methods for each entity.

```csharp
using (MovieReviewObjects ctx =
        new MovieReviewObjects(connectionString))
{
    Movie movie = new Movie()
    {
        ReleaseDate = new DateTime(2008, 1, 1),
        Title = "Revenge of Riverdance"
    };
    ctx.AddToMovies(movie);
    ctx.SaveChanges();
}
```

Updates

- Change tracking service will record any changes to materialized entities
- SaveChanges will atomically update all changed entities

```
using (MovieReviewObjects ctx =
      new MovieReviewObjects(connectionString))
{
    var movie = (from m in ctx.Movies
                 where m.Title == "Revenge of Riverdance"
                 select m).First();

    movie.ReleaseDate = movie.ReleaseDate.AddDays(1);
    ctx.SaveChanges();
}
```

Deletes

- **Use DeleteObject on the object context.**
- **SaveChanges will create one DELETE for each object**

```
using (MovieReviewObjects ctx =
           new MovieReviewObjects(connectionString))
{
    var movies = from m in ctx.Movies
                 where m.Title == "Revenge of Riverdance"
                 select m;

    foreach (var m in movies)
    {
        ctx.DeleteObject(m);
    }
    ctx.SaveChanges();
}
```

Compare and Contrast

	LINQ to SQL	Entity Framework
Advanced Mapping	No	Yes
POCO support	Yes	No
Lazy loading	Implicit	Explicit
Other database support	SQL only	Planned
Full query language	No	eSQL
Enhanced in .NET 4.0	?	Yes

Summary

- Entity Data Model is the centerpiece
  - Broken into three layers
- Object Services
  - Change tracking
  - LINQ to Entities