****

Zentity (Version 1.0)

Data Model Extensibility User Guide

May 2009

Contents

[Introduction 4](#_Toc229295849)

[Zentity Core EDM 4](#_Toc229295850)

[Zentity Data Model (ZDM) 5](#_Toc229295851)

[Zentity Data Model Elements 6](#_Toc229295852)

[Data Model Module 6](#_Toc229295853)

[Resource Type 7](#_Toc229295854)

[Scalar Property 7](#_Toc229295855)

[Association 7](#_Toc229295856)

[Navigation Property 7](#_Toc229295857)

[Processing the Zentity Data Model 7](#_Toc229295858)

[Modifying the backend schema 7](#_Toc229295859)

[Generation of source code 8](#_Toc229295860)

[Generation of mapping files 8](#_Toc229295861)

[System Structure 8](#_Toc229295862)

[Metadata Tables 9](#_Toc229295863)

[Core.DataModelModule 10](#_Toc229295864)

[Core.ResourceType 10](#_Toc229295865)

[Core.ScalarProperty 10](#_Toc229295866)

[Core.NavigationProperty 11](#_Toc229295867)

[Core.Association 11](#_Toc229295868)

[Core.Resource Table 12](#_Toc229295869)

[Association Views 12](#_Toc229295870)

[Stored Procedures 12](#_Toc229295871)

[Resource Management API 12](#_Toc229295872)

[Extensibility API 13](#_Toc229295873)

[SQL Script Generation 14](#_Toc229295874)

[Handling Data Model Module Changes 14](#_Toc229295875)

[Handling Resource Type Changes 15](#_Toc229295876)

[Handling Scalar Property Changes 15](#_Toc229295877)

[Handling Navigation Property Changes 16](#_Toc229295878)

[Handling Association Changes 16](#_Toc229295879)

[Creating Resource Type CUD Procedures 19](#_Toc229295880)

[Entity Framework Artifacts Generation 20](#_Toc229295881)

[Simple Resource Type with Scalar Properties 20](#_Toc229295882)

[ZDM 20](#_Toc229295883)

[Consolidated SSDL 21](#_Toc229295884)

[Extended Core CSDL 22](#_Toc229295885)

[Module CSDL 22](#_Toc229295886)

[Consolidated MSL 22](#_Toc229295887)

[Non One-To-XXX Association 23](#_Toc229295888)

[ZDM 23](#_Toc229295889)

[Consolidated SSDL 23](#_Toc229295890)

[Extended Core CSDL 25](#_Toc229295891)

[Module CSDL 26](#_Toc229295892)

[Consolidated MSL 26](#_Toc229295893)

[One-To-XXX Association 28](#_Toc229295894)

[ZDM 28](#_Toc229295895)

[Consolidated SSDL 29](#_Toc229295896)

[Extended Core CSDL 30](#_Toc229295897)

[Module CSDL 31](#_Toc229295898)

[Consolidated MSL 31](#_Toc229295899)

[Performance Considerations 33](#_Toc229295900)

[Creating Indexes for Scalar Properties 33](#_Toc229295901)

[Creating Indexes for Association Views 33](#_Toc229295902)

[Creating Full-Text Indexes on String Properties 33](#_Toc229295903)

[Entity Framework Best Practices 34](#_Toc229295904)

[Walkthroughs 35](#_Toc229295905)

[Re-creating Zentity Database 35](#_Toc229295906)

[Working with data model modules 35](#_Toc229295907)

[Creating Module, Assembly and EF Artifacts 35](#_Toc229295908)

[Using the Generated Assembly and Artifacts 36](#_Toc229295909)

[Generating Assembly with Embedded Entity Framework Artifacts 37](#_Toc229295910)

[Modules with One-To-XXX Associations 38](#_Toc229295911)

[Creating Module, Assembly and EF Artifacts 38](#_Toc229295912)

[Using the Generated Assembly and Artifacts 40](#_Toc229295913)

[Modules with Non One-To-XXX Associations 41](#_Toc229295914)

[Creating Module, Assembly and EF Artifacts 41](#_Toc229295915)

[Using the Generated Assembly and Artifacts 42](#_Toc229295916)

[Generating Pre-Compiled Views for Custom Modules 43](#_Toc229295917)

# Introduction

Zentity allows developers to introduce new entities and associations in the Core entity data model that comes with the installation. The process of extending the Core entity data model involves defining new entities and associations between them, modifying the database schema to accommodate new entities, generation of source code and generation of Entity Framework artifacts for the new entities. Zentity provides an API to do most of the required tasks. This document discusses the details of this API, structure of the generated database objects and Entity Framework artifacts and some performance improvement techniques for applications that use the extensibility feature. Also, this document contains simple walkthroughs for some common scenarios.

This document assumes that the readers are well-versed with Entity Framework concepts and the Entity Data Model (EDM, <http://msdn.microsoft.com/en-us/library/bb387122.aspx>).

## Zentity Core EDM

The Zentity Core EDM has just the basic entities required to create a tuple store. ‘Resource’, ‘Relationship’, ‘Predicate’, and ‘RelationshipProperty’ are the main entities that can be used to represent the tuple information. This model can be extended to include new entities that inherit from ‘Resource’ using the extensibility API. The API provides constructs to define new entities, their inheritance hierarchy and associations between themselves. It then generates appropriate Entity Framework artifacts that can be used in conjunction with the Core EDM to create domain-specific applications.

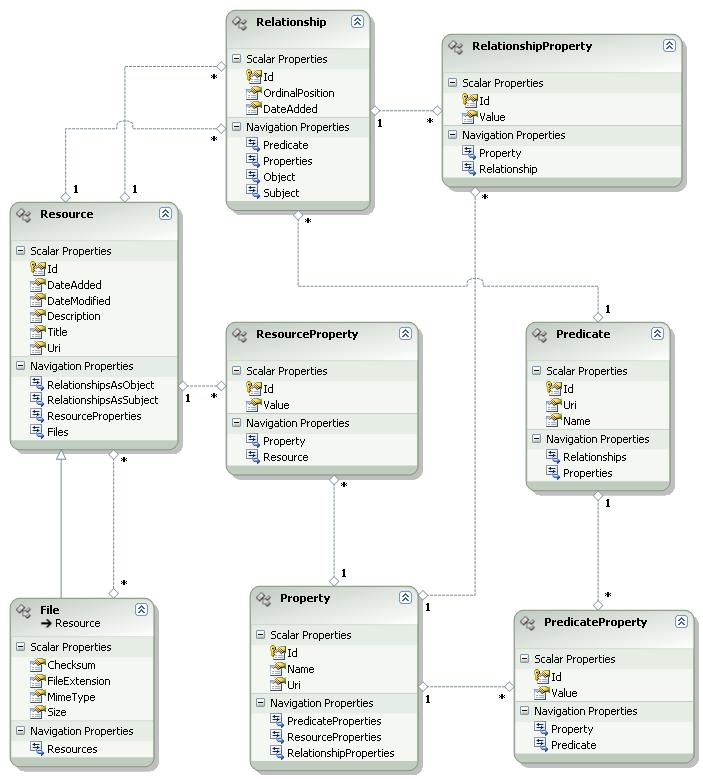


Figure 1. Zentity Core EDM

# Zentity Data Model (ZDM)

The Zentity Data Model can be used to define the domain-specific resource types and associations in an application. Since Zentity is built on top of the Entity Framework, we tried to stay as close as possible to the Entity Data Model (EDM). Most of the concepts here are very similar in nature to the Entity Data Model. The table below shows the correspondence between the two models.

|  |  |
| --- | --- |
| Zentity Data Model (ZDM) Construct | Entity Data Model (EDM) Construct |
| Resource Type | Entity |
| Scalar Property | Scalar Property |
| Navigation Property | Navigation Property |
| Association | Association |

The rest of the document uses the terms ZDM and ‘data model’ interchangeably. The extensibility API works on ZDM constructs and generates the EDM constructs in addition to other items. The generated EDM has all its entities deriving directly or indirectly from the ‘Resource’ entity in the Zentity Core EDM. There is a single ZDM associated with the entire store. Multiple data model modules can be present in a ZDM.

Figure 2. Zentity Data Model Elements

## Zentity Data Model Elements

### Data Model Module

A data model module is a container for resource types belonging to a particular domain. For example, the module ‘ScholarlyWorks’ hosts resource types for the research domain. Each data model module is identified by a Namespace. The following figure shows the Core data model module.

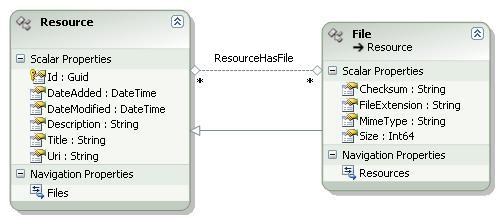


Figure 3. Zentity.Core Module

### Resource Type

A resource type represents a concept in a particular domain. For example, ‘Publication’, ‘Journal’ etc. are concepts in the ‘ScholarlyWorks’ domain.

### Scalar Property

A scalar property is an attribute of a resource type that can be represented as a string value. For example, ‘Publication’ resource type has scalar properties such as ‘DatePublished’, ‘DOI’ etc.

### Association

An association is a relationship between two resource types. Each Association has a logical direction from ‘Subject’ resource type to the ‘Object’ resource type. Subject and object multiplicities can be one of {‘Many’, ‘One’, ‘ZeroOrOne’}. As an example, ‘Publication’ and ‘Person’ can have a Many-To-Many association ‘IsAuthoredBy’ from ‘Publication’ to ‘Person’. Here, ‘Publication’ and ‘Person’ are instances of ‘ResourceType’ and ‘IsAuthoredBy’ is an instance of ‘Association’. When this data model is converted into C# source code, classes will be generated for ‘Publication’ and ‘Person’ resource types. ‘IsAuthoredBy’ association will not appear directly as a class but will result into creation of properties (Navigation Properties) within ‘Publication’ and ‘Person’ classes. These properties will hold the references of other related type. Following statements explain this further.

|  |
| --- |
| ‘Publication’ is an instance of ‘ResourceType’.  ‘Person’ is an instance of ‘ResourceType’.  ‘IsAuthoredBy’ is an instance of ‘Association’.  myPub is an instance of ‘Publication’.  bob is an instance of ‘Person’.  myPub.Persons.First() returns bob.  bob.Publications.First() returns myPub. |

### Navigation Property

A navigation property represents a named end point of an Association and is hosted by a resource type. For example, ‘Publication’ and ‘Person’ may participate in an association ‘IsAuthoredBy’. The end point of the association that is attached to ‘Publication’ resource type is named ‘Persons’ and the other end point of the association that is attached to the ‘Person’ resource type is named ‘Publications’. While generating C# code from the data model, two classes ‘Publication’ and ‘Person’ are created. ‘Publication’ has a property ‘Persons’, which is a collection of references to ‘Person’ objects. Likewise, ‘Person’ class has a property, ‘Publications’, which is a collection of references to ‘Publication’ objects.

## Processing the Zentity Data Model

The Zentity data model is interpreted by the extensibility API to generate the database objects, source code and Entity Framework artifacts for the domain-specific entities. The extensibility API is designed in such a way that these tasks can be performed independent of each other. Following sections summarize these tasks.

### Modifying the backend schema

This involves adding/removing/updating of database objects in response to the changes in the ZDM. To modify the backend schema, we can use DataModel.Synchronize method. This method generates the required TSQL scripts and executes them in a transaction. We can also generate the TSQL scripts using the DataModel.GetSynchronizationScripts method. Generated scripts can be used by an administrator to understand the database modifications done by Zentity.

### Generation of source code

This is the process of translating ZDM items into .NET classes. To generate the source code, we can use DataModel.GenerateSourceCode or DataModel.GenerateExtensionsAssembly methods. These methods take as an input the namespaces of data model modules for which to generate the .NET types. While generating the assembly, we can also specify a list of data model modules for which we want to generate the entity framework artifacts. These artifacts can be optionally embedded in the generated assembly.

### Generation of mapping files

To work with the generated classes, we require a set of entity framework specific mapping files. Readers are encouraged to refer the Entity Data Model Schemas and Mapping Specifications (<http://msdn.microsoft.com/en-us/library/bb399604.aspx>) for the details of these files. To generate the Entity Framework artifacts, we can use the DataModel.GenerateEFArtifacts method. The method takes as a parameter the list of data model modules for which to generate the mapping files. Consider that we have three modules inside our data model, Core, ScholarlyWorks and Museum. If we pass {Core, ScholarlyWorks} as an input to this method the generated artifacts will have information about the entities and associations of only these two modules. If we pass NULL or {Core, ScholarlyWorks, Museum} to this method, the generated artifacts will have the information about all the modules. Generating artifacts for {Core, ScholarlyWorks} and then for {Core, Museum} WILL NOT generate the same artifacts as those generated for {Core, ScholarlyWorks, Museum}.

NOTE: Even though it is possible to generate artifacts for various combinations of modules present in the model, it is recommended to use only one of these combinations to avoid application errors. For example, consider a custom module M1 with resource type RT1 that has discriminator value 100. App1 creates resources using artifacts generated for M1 module and App2 uses artifacts embedded in Zentity.Core.dll to enumerate resources on the same store. Now rows corresponding to RT1 in Core.Resource table will have discriminator value as 100, but the artifacts used by App2 do not have any entity type defined that maps to this discriminator. So, resources retrieved by App2 will not have any RT1 instances which may result into application errors.

# System Structure

Figures below show the components involved in the generation of a custom data model.

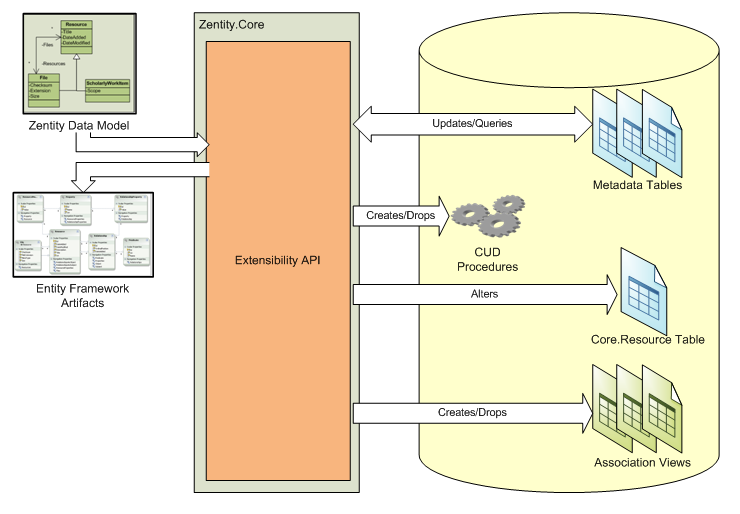


Figure 4. Generating Custom Data Model.

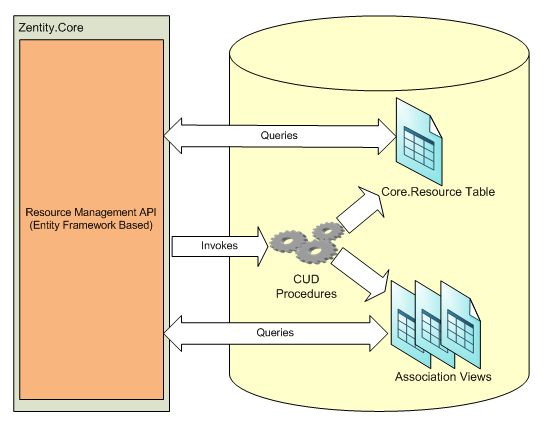


Figure 5. Using Custom Data Model.

## Metadata Tables

These tables store the information about data model modules, resource types, scalar properties, navigation properties and associations. These tables can be updated either by invoking DataModel.Synchronize() or by executing the scripts generated by DataModel.GetSynchronizationScripts(). Following sections describe the structure of metadata tables.

### Core.DataModelModule

This table stores the information about the modules of data model. Changing the default model items of IsMsShipped modules may be error prone. For example, DataModel.Synchronize() will raise errors if resource type ‘Zentity.Core.Resource’ is removed from the module ‘Zentity.Core’. Sample entries in this table are shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Namespace | Uri | Description | IsMsShipped |
| 6B9CDECB-3152-46D0-BECB-A23ECC71109F | Zentity.Core | urn:zentity/module/zentity-core | The core data model. | 1 |

### Core.ResourceType

This table stores information about all the resource types in the model. Zentity uses Table-Per-Hierarchy type inheritance (<http://msdn.microsoft.com/en-us/library/bb738443.aspx>) to create database objects for its resource types. All resource types share the same table, Core.Resource, to store rows for their instances. The instance rows are differentiated based on the value of a ‘Discriminator’ column. Core.ResourceType table contains the discriminator value for each of the resource type. Sample entries in this table are shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | DataModelModuleId | BaseTypeId | Name | Uri | Description | Discriminator |
| 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | 6B9CDECB-3152-46D0-BECB-A23ECC71109F | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | File | urn:zentity/module/zentity-core/resource-type/file | Represents a binary file. | 2 |
| D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | 6B9CDECB-3152-46D0-BECB-A23ECC71109F | NULL | Resource | urn:zentity/module/zentity-core/resource-type/resource | The ultimate base resource type for all resource types. | 1 |

### Core.ScalarProperty

This table hosts the information about all the scalar properties in the model. ‘TableName’ and ‘ColumnName’ columns specify the mapping information of the scalar property. Each scalar property is mapped to a column of Core.Resource table. Sample entries in this table are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Id | ResourceTypeId | Name | Uri |
| A8076A7F-DC0F-4B28-AB17-492742813A04 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | DateAdded | urn:zentity/module/zentity-core/resource-type/resource/property/date-added |
| 4996CB49-15B3-44ED-B46D-0CE2132DA900 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | DateModified | urn:zentity/module/zentity-core/resource-type/resource/property/date-modified |
| 0E671214-CFD7-43C9-8267-8F87D12B6687 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | Description | urn:zentity/module/zentity-core/resource-type/resource/property/description |
| 9664A465-080D-4CFE-B5D2-450AC24CDE03 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | Id | urn:zentity/module/zentity-core/resource-type/resource/property/id |
| 2F2D690A-7FE1-4A16-BA78-0AD6E49ABF9F | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | Title | urn:zentity/module/zentity-core/resource-type/resource/property/title |
| EEE5DD55-79AF-4480-B51A-5A7FC360C6C3 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | Uri | urn:zentity/module/zentity-core/resource-type/resource/property/uri |
| 03E621E3-4602-4D29-8F7E-6C00178D95FB | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | Checksum | urn:zentity/module/zentity-core/resource-type/file/property/checksum |
| 720D7515-4FE4-4EFB-AA8A-89558B7FD9D6 | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | FileExtension | urn:zentity/module/zentity-core/resource-type/file/property/file-extension |
| 9FFD4730-1007-46FC-B538-EBE27FBAA992 | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | MimeType | urn:zentity/module/zentity-core/resource-type/file/property/mime-type |
| ABFBEEC4-2A46-4A3F-8B48-6F50EEC10A9F | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | Size | urn:zentity/module/zentity-core/resource-type/file/property/size |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Description | DataType | Nullable | MaxLength | Scale | Precision | TableName | ColumnName |
| Gets the date on which the Resource was added in the repository. | DateTime | 1 | NULL | NULL | NULL | Resource | DateAdded |
| Gets the date on which the Resource was last modified. | DateTime | 1 | NULL | NULL | NULL | Resource | DateModified |
| Gets or sets the description of the Resource. | String | 1 | 4000 | NULL | NULL | Resource | Description |
| Gets the Id value that uniquely identifies the Resource. | Guid | 0 | NULL | NULL | NULL | Resource | Id |
| Gets or sets the title of the Resource. | String | 1 | 425 | NULL | NULL | Resource | Title |
| Gets or sets the URI of the Resource. | String | 1 | 1024 | NULL | NULL | Resource | Uri |
| Gets or sets the hash value of file data. | String | 1 | 256 | NULL | NULL | Resource | Checksum |
| Gets or sets the extension of uploaded content file. | String | 1 | 128 | NULL | NULL | Resource | FileExtension |
| Gets or sets the MIME type of data. | String | 1 | 128 | NULL | NULL | Resource | MimeType |
| Gets or sets the size of data file. | Int64 | 1 | NULL | NULL | NULL | Resource | Size |

### Core.NavigationProperty

This table hosts the information about all the navigation properties in the data model. ‘TableName’ and ‘ColumnName’ specify the mapping of this navigation property to some column. Not all navigation properties require mapping. It is only the XXX side navigation properties of a One-To-XXX or XXX-To-One association that need this mapping. The mapped column is used as a foreign key to the entity on ‘One’ side of the association. Sample entries in the table are shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | ResourceTypeId | Name | Uri | Description | TableName | ColumnName |
| 054F2A48-2FCE-420F-8763-EAB97A01BC56 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | Files | urn:zentity/module/zentity-core/resource-type/resource/navigation-property/files | Gets the associated files with this resource. | NULL | NULL |
| 8493213F-1C82-40CB-99D0-D9A0C092C416 | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | Resources | urn:zentity/module/zentity-core/resource-type/file/navigation-property/resources | Gets the associated resources for this file. | NULL | NULL |

### Core.Association

This table hosts the information about all the associations in the model. Each association is associated with a predicate in the repository. Subject and object multiplicities can be one of {‘Many’, ‘One’, ‘ZeroOrOne’}. Sample entries in the table are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Uri | SubjectNavigationPropertyId |
| 10538DC2-1CAC-4755-9A34-2938201EB06E | ResourceHasFile | urn:zentity/module/zentity-core/association/resource-has-file | 054F2A48-2FCE-420F-8763-EAB97A01BC56 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ObjectNavigationPropertyId | PredicateId | SubjectMultiplicity | ObjectMultiplicity | ViewName |
| 8493213F-1C82-40CB-99D0-D9A0C092C416 | 818A93F5-25A9-4149-A8D2-19104A352DA0 | Many | Many | ResourceHasFile |

One-To-One associations are not supported by the Extensibility API. Enforcing a one to one relationship in SQL Server is not possible using simple Primary Key, Foreign Key or Unique Key constraints. For example, let’s assume that a one to one relationship exists between table A and B. Now if we insert a row in table A, SQL Server should error out because there is no corresponding row in table B. Likewise, we should not be able to insert values in table B. In effect, we will not be able to insert values in these tables ever which is not consistent with the semantics of the cardinality.

A workaround is to always interact with a stored procedure to insert values in the backend tables and the tables themselves do not have any constraints. Such an implementation is not trivial, so we leave out this cardinality in this version of Zentity.

## Core.Resource Table

This table hosts the actual data for the resources. Each resource is an instance of a resource type. New columns are added to resource table with the introduction of new scalar properties in the data model. This table is subjected to a limit of 1024 columns, which in turn limits the maximum number of scalar and navigation properties in the data model. In addition, we put a restriction of 250 properties (scalar plus navigation) on each resource type. This includes all the properties inherited by the resource type from its ancestors. Sample entries in Core.Resource table is shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | ResourceTypeId | Discriminator | DateAdded | DateModified | Description |
| 628662D5-F67E-4F6B-8A07-722A4531DD95 | D2BD64DF-6609-4EA4-AE99-9669DA69BF7A | 1 | 22:37.1 | 22:37.1 | A resource |
| CC1E658F-AC45-4960-91DB-F2D539D8818B | 94C567F8-B3EB-4CCB-9BF1-FC88970F78D7 | 2 | 22:37.1 | 22:37.1 | A file |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Title | Uri | Checksum | FileExtension | MimeType | Size |
| First resource | NULL | NULL | NULL | NULL | NULL |
| First file | NULL | NULL | NULL | txt | 1000 |

## Association Views

These views are created on Core.Relationship table for each of the defined association in the data model. A predicate is used as a filter for the view definition. For example, the view definition for association ‘ResourceHasFile’ looks like the following.

|  |
| --- |
| CREATE VIEW [Core].[ResourceHasFile]  WITH SCHEMABINDING  AS  SELECT [SubjectResourceId], [ObjectResourceId]  FROM [Core].[Relationship] T  WHERE [T].[PredicateId] = '818A93F5-25A9-4149-A8D2-19104A352DA0'; |

These views are mapped to associations in the data model. Next sections explain in detail the structure of mapping files.

## Stored Procedures

Each resource type in the data model has one procedure each for creating, updating and deleting the resources of that type from the repository. Likewise, Many-To-Many, Many-To-ZeroOrOne, ZeroOrOne-To-Many and ZeroOrOne-To-ZeroOrOne associations also have these procedures to manipulate the relationships in the repository.

## Resource Management API

The resource management API is a typical Entity Framework object layer generated for Zentity entities. Major classes in this API are as follows.

|  |  |
| --- | --- |
| Class | Description |
| ZentityContext | Represents the repository context. |
| Resource | Represents a resource, usually the research output of an organization. |
| Relationship | Represents a relationship between two resources. |
| Predicate | Represents the nature of relationships between two resources. |
| File | Represents a binary file. |
| Property | Represents a characteristic of a resource in the repository. |
| ResourceProperty | The ResourceProperty object provides a value for a Property on a Resource. |
| RelationshipProperty | The RelationshipProperty object provides a value for a Property on a Relationship. |
| PredicateProperty | The PredicateProperty object provides a value for a Property on a Predicate. |

## Extensibility API

As mentioned in an earlier section, the extensibility API provides constructs to

* Define the data model,
* Generate TSQL scripts or update the database schema directly,
* Generate Entity Framework artifacts,
* Generate source code or the extensions assembly.

The figure below shows the major classes of this API.



Figure 6. Extensibility API Class Diagram.

# SQL Script Generation

Following sections describe how database objects are created/dropped/updated in response to the changes in the data model.

## Handling Data Model Module Changes

No database objects are created or dropped. Just the metadata information in Core.DataModelModule table gets updated.

## Handling Resource Type Changes

For each added resource type, a new row is created in the Core.ResourceType table with a discriminator value that is greater than all other discriminators present in the table. Recall that this value is used by Entity Framework to distinguish between resources of different types in a TPH based hierarchy implementation.

Likewise, for deleted and updated resource types, only the metadata gets updated. No database objects are created or dropped.

## Handling Scalar Property Changes

For added scalar properties, a new column is created in Core.Resource table. The SqlScriptGenerator generates statements like the following to create the new column.

|  |
| --- |
| ALTER TABLE [Core].[Resource] ADD [c7a2b2e1-49b3-4ca0-a1a1-6e7b114007e4] nvarchar(256) NULL |

To avoid naming conflicts, the name of the column is the ID value of ScalarProperty. This also helps keeping the change history logging module simple, since the column name is not changed across scalar property updates. Otherwise, sp\_rename invocations were needed with each scalar property name update. Following table shows the mapping between scalar property DataType value and SQL Server data types.

|  |  |
| --- | --- |
| Scalar Property Data Type | SQL Server Data Type |
| Int32 | Int |
| Binary | Varbinary |
| Boolean | Bit |
| Byte | Int |
| DateTime | Datetime |
| Decimal | Decimal |
| Double | Float |
| Guid | Uniqueidentifier |
| Single | Real |
| Int16 | Smallint |
| Int64 | Bigint |
| String | Nvarchar |

For string and binary types, a value of -1 for the MaxLength will generate nvarchar(max) and varbinary(max) respectively in the script. All generated column definitions are null able. This is required to adhere with the TPH implementation. Another section on CUD procedures explains how we enforce the not-null check in the generated CUD procedures.

For deleted scalar properties, a statement similar to the following is generated by the script generator.

|  |
| --- |
| ALTER TABLE [Core].[Resource] DROP COLUMN [c7a2b2e1-49b3-4ca0-a1a1-6e7b114007e4] |

Since the column is dropped from the database, all data is lost for the scalar property.

For updated scalar properties, ALTER TABLE ALTER COLUMN statements are generated if the datatypes of previous and next scalar properties are compatible. Otherwise, the column is dropped from the table and is recreated with new definition. So, data losses might occur during scalar property updates. Name of the column remains the same during this process of reconstruction. Following table lists compatible data type conversions.

|  |  |
| --- | --- |
| Previous Data Type | Next Data Type |
| Binary | Binary with MaxLength >= previous MaxLength or -1 |
| Boolean | Binary, Boolean, Int16, Int32, Int64, Single, String |
| Byte | Byte |
| DateTime | DateTime |
| Decimal | Decimal, String with MaxLength >= 40 or -1 |
| Double | Double, Single, String with MaxLength >= 40 or -1 |
| Guid | Guid, String with MaxLength >= 40 or -1 |
| Int16 | Int16, Int32, Int64, Double, Single, String with MaxLength >= 40 or -1 |
| Int32 | Int32, Int64, Double, Single, String with MaxLength >= 40 or -1 |
| Int64 | Int64, Double, Single, String with MaxLength >= 40 or -1 |
| Single | Double, Single, String with MaxLength >= 40 or -1 |
| String | String with MaxLength >= previous MaxLength or -1 |

Also, the column is re-created if the scalar property undergoes a parent change.

Finally, the Core.ScalarProperty table is updated with appropriate data. All table and column mapping information for the scalar property is also persisted in the same table.

## Handling Navigation Property Changes

We willrefer Many-To-One, One-To-Many, One-To-ZeroOrOne, ZeroOrOne-To-One associations as OneToXXX in rest of the document.

For added navigation properties, if the navigation property belongs to the XXX side of a OneToXXX association, a foreign key column is created in Core.Resource table. The definition looks like the following.

|  |
| --- |
| ALTER TABLE [Core].[Resource] ADD [d340f653-28fc-4e00-b497-c156b9d7733b] uniqueidentifier CONSTRAINT [FK\_d340f653-28fc-4e00-b497-c156b9d7733b] FOREIGN KEY REFERENCES [Core].[Resource] ([Id]) |

Names of the column and the foreign key constraint are derived from the navigation property Id.

For deleted navigation properties, if the navigation property belongs to the XXX side of a OneToXXX association, the column generated above is dropped.

For updated navigation properties, if there is a change in parent resource type, the effect is the same as removing the navigation property from the data model and then adding it again. Appropriate database objects are dropped and re-created as mentioned above. The metadata however is only updated, we do not remove and add rows in Core.NavigationProperty table. In addition, this re-create logic is propagated to the association that was attached to the navigation property. So, the views for the association are also dropped and recreated. No change in metadata of the association happens because of this.

Finally, Core.NavigationProperty is updated to reflect the latest changes. The table and column mappings are not null if the navigation property belongs to the XXX side of a OneToXXX association. In this case, the mapping information points to the foreign key column.

## Handling Association Changes

A view is created for each association added in the data model. These views are created on the relationship table with schema binding and expose just the subject and object resource IDs for a given predicate. The view definition is similar to as shown below.

|  |
| --- |
| CREATE VIEW [Core].[0a62762a195841baaa3c8abd8a4a5aee]  WITH SCHEMABINDING  AS  SELECT [SubjectResourceId], [ObjectResourceId]  FROM [Core].[Relationship] T  WHERE [T].[PredicateId] = 'a5781cc2-dd1b-4355-9a9b-4675103a2e7f'; |

Name of the view is derived from the association Id. A new predicate is always created for added associations. The Id of new predicate is used as a filter in the view definition.

Depending on the association end multiplicities, unique indexes are created on the view to enforce cardinality constraints. For example, for a Many-To-Many association, the index will contain both the subject and object resource ids as shown below.

|  |
| --- |
| CREATE UNIQUE CLUSTERED INDEX [PK\_0a62762a195841baaa3c8abd8a4a5aee] ON [Core].[0a62762a195841baaa3c8abd8a4a5aee]  (  [SubjectResourceId], [ObjectResourceId]  ) |

Following table lists the columns included in the clustered and non-clustered index definitions.

|  |  |  |
| --- | --- | --- |
| Association Cardinality | Unique clustered index columns | Unique non-clustered index columns |
| Many-To-Many | SubjectResourceId, ObjectResourceId | NIL |
| Many-To-One | SubjectResourceId | NIL |
| Many-To-ZeroOrOne | SubjectResourceId | NIL |
| One -To-Many | ObjectResourceId | NIL |
| One -To-ZeroOrOne | SubjectResourceId | NIL |
| ZeroOrOne -To-Many | ObjectResourceId | NIL |
| ZeroOrOne -To-One | ObjectResourceId | NIL |
| ZeroOrOne -To-ZeroOrOne | SubjectResourceId | ObjectResourceId |

For non OneToXXX associations, CREATE and DELETE procedures are created to insert and remove values from the view. The definition of these procedure is similar to as shown below.

|  |
| --- |
| CREATE PROCEDURE [Core].[Insert0a62762a195841baaa3c8abd8a4a5aee]  @SubjectResourceId [uniqueidentifier],  @ObjectResourceId [uniqueidentifier]  AS  BEGIN  SET NOCOUNT ON  INSERT INTO [Core].[Relationship]  (  [Id],  [SubjectResourceId],  [ObjectResourceId],  [PredicateId]  )  VALUES  (  NEWID(),  @SubjectResourceId,  @ObjectResourceId,  'a5781cc2-dd1b-4355-9a9b-4675103a2e7f'  )  END  GO  CREATE PROCEDURE [Core].[Delete0a62762a195841baaa3c8abd8a4a5aee]  @SubjectResourceId [uniqueidentifier],  @ObjectResourceId [uniqueidentifier]  AS  BEGIN  SET NOCOUNT ON  DELETE FROM [Core].[Relationship]  WHERE [SubjectResourceId] = @SubjectResourceId  AND [ObjectResourceId] = @ObjectResourceId  AND [PredicateId] = 'a5781cc2-dd1b-4355-9a9b-4675103a2e7f';  END  GO |

While processing deleted associations, if the association is OneToXXX and the Core.Relationship table contains instances of this association, the delete is rejected. It is required to delete the XXX side entity to delete OneToXXX relationships. A statement similar to the following is generated for OneToXXX association deletions.

|  |
| --- |
| IF EXISTS( SELECT 1 FROM [Core].[Association] [A]  INNER JOIN [Core].[Relationship] [R]  ON [A].[PredicateId] = [R].[PredicateId]  WHERE [A].[Id] = '005dbf5a-39ee-434e-bfec-b81ba62e4e4d')  BEGIN  RAISERROR('Association with Id = [005dbf5a-39ee-434e-bfec-b81ba62e4e4d] cannot be deleted since Relationships are present in the store with this association. It is a One-To-Many association. Delete the resource on [Many] side to remove the relationship and then delete the association.', 16, 1);  END |

Next, a statement to delete the association instances from the Relationship table is generated.

|  |
| --- |
| DELETE FROM [Core].[Relationship] WHERE [PredicateId] = 'a5781cc2-dd1b-4355-9a9b-4675103a2e7f'; |

Then the association view is dropped.

|  |
| --- |
| DROP VIEW [Core].[0a62762a195841baaa3c8abd8a4a5aee] |

Then the CREATE and DELETE procedures are dropped for OneToXXX associations.

|  |
| --- |
| DROP PROCEDURE [Core].[Insert0a62762a195841baaa3c8abd8a4a5aee]  GO  DROP PROCEDURE [Core].[Delete0a62762a195841baaa3c8abd8a4a5aee]  GO |

Processing of updated associations happens as follows:

1. If there is a change in subject or object navigation properties or the previous and new cardinalities are not compatible, the effect is same as dropping the association from the model and re-creating it. The metadata however is only updated and not re-created. The re-create logic is also propagated to the navigation properties. So, the foreign key columns etc are dropped and recreated for OneToXXX associations. The table below shows the data loss behavior on cardinality updates. The update is considered to be compatible if the cardinalities are ‘completely convertible without information loss’.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | M2M | M2O | M2Z | O2M | O2O | O2Z | Z2M | Z2O | Z2Z |
| M2M |  |  |  |  |  |  |  |  |  |
| M2O |  |  |  |  |  |  |  |  |  |
| M2Z |  |  |  |  |  |  |  |  |  |
| O2M |  |  |  |  |  |  |  |  |  |
| O2Z |  |  |  |  |  |  |  |  |  |
| Z2M |  |  |  |  |  |  |  |  |  |
| Z2O |  |  |  |  |  |  |  |  |  |
| Z2Z |  |  |  |  |  |  |  |  |  |
|  | Completely convertible without information loss. | | | | | | | | |
|  | Convertible with data loss. | | | | | | | | |
|  | Convertible only if no Relationships are present in the repository. | | | | | | | | |
|  | Not convertible at all. | | | | | | | | |

1. If the change is only in cardinalities and the update is compatible,
   1. if the cardinality is changed from OneToXXX to a non OneToXXX the foreign key constraint and the column is dropped from the resource table.
   2. Drop the association view and the CREATE/DELETE procedures if they exist.
   3. Create new views and CREATE/DELETE procedures as per the new cardinality definition.

Finally, entries are added/removed/updated in Core.Association table based on the change type.

## Creating Resource Type CUD Procedures

For all the resource types that have undergone some change in terms of parent, scalar properties or navigation properties, stored procedures are created to CREATE, UPDATE and DELETE instances of this resource type. The generated script is similar to the following.

|  |
| --- |
| CREATE PROCEDURE [Core].[Insert0e3efa4a7bfa471cb7bd17685cdef487]  @LastName nvarchar(256)  , @FirstName nvarchar(256)  , @MiddleName nvarchar(256)  , @Email nvarchar(2048)  , @Scope nvarchar(128)  , @Title nvarchar(425)  , @DateModified datetime  , @Id uniqueidentifier  , @DateAdded datetime  , @Uri nvarchar(1024)  , @Description nvarchar(4000)  AS  BEGIN  SET NOCOUNT ON;  -- Not NULL value check  IF @Id IS NULL  BEGIN  RAISERROR (N'Cannot insert the value NULL into column [Id].', 16, 1);  RETURN -1;  END  INSERT INTO [Core].[Resource]  (  [c7a2b2e1-49b3-4ca0-a1a1-6e7b114007e4],  [3d82f5e2-b84a-4422-a0be-82317ad5059f],  [95d6858e-6294-4572-8686-c2c2c8dddd17],  [e7fa1d49-3936-4ec3-9414-058909d19c83],  [165e8189-f6a3-42c6-ba9f-bddaa81364df],  [Title],  [DateModified],  [Id],  [DateAdded],  [Uri],  [Description],  [ResourceTypeId],  [Discriminator]  )  VALUES  (  @LastName,  @FirstName,  @MiddleName,  @Email,  @Scope,  @Title,  @DateModified,  @Id,  @DateAdded,  @Uri,  @Description,  '0e3efa4a-7bfa-471c-b7bd-17685cdef487',  7  );  END  GO  CREATE PROCEDURE [Core].[Update0e3efa4a7bfa471cb7bd17685cdef487]  @LastName nvarchar(256)  , @FirstName nvarchar(256)  , @MiddleName nvarchar(256)  , @Email nvarchar(2048)  , @Scope nvarchar(128)  , @Title nvarchar(425)  , @DateModified datetime  , @Id uniqueidentifier  , @DateAdded datetime  , @Uri nvarchar(1024)  , @Description nvarchar(4000)  AS  BEGIN  SET NOCOUNT ON;  -- Not NULL value check  IF @Id IS NULL  BEGIN  RAISERROR (N'Cannot insert the value NULL into column [Id].', 16, 1);  RETURN -1;  END  UPDATE [Core].[Resource]  SET  [c7a2b2e1-49b3-4ca0-a1a1-6e7b114007e4] = @LastName  , [3d82f5e2-b84a-4422-a0be-82317ad5059f] = @FirstName  , [95d6858e-6294-4572-8686-c2c2c8dddd17] = @MiddleName  , [e7fa1d49-3936-4ec3-9414-058909d19c83] = @Email  , [165e8189-f6a3-42c6-ba9f-bddaa81364df] = @Scope  , [Title] = @Title  , [DateModified] = @DateModified  , [DateAdded] = @DateAdded  , [Uri] = @Uri  , [Description] = @Description  WHERE  [Id] = @Id;  END  GO  CREATE PROCEDURE [Core].[Delete0e3efa4a7bfa471cb7bd17685cdef487]  @Id [uniqueidentifier]  AS  BEGIN  SET NOCOUNT ON;  DELETE FROM [Core].[Resource]  WHERE [Id] = @Id;  END  GO |

CREATE and UPDATE procedures take as parameter all the scalar properties defined in the resource type and its ancestors up until Zentity.Core.Resource.

# Entity Framework Artifacts Generation

This section describes the structure of CSDL, SSDL and MSL generated by the extensibility API after interpreting the Zentity Data Model.

## Simple Resource Type with Scalar Properties

### ZDM

Figure 7. ZDM for Simple Resource Type.

Refer to the walkthrough ‘Working with data model modules’ in this document for the detailed steps to define and use this data model.

### Consolidated SSDL

As the snippet below shows, Core.Resource table is altered to include a new column and new CUD stored procedures are created for the resource type.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core.Store" Alias="Self" Provider="System.Data.SqlClient" ProviderManifestToken="2005" xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator" xmlns="http://schemas.microsoft.com/ado/2006/04/edm/ssdl">  <EntityContainer Name="Core">  ...  </EntityContainer>  ...  <EntityType Name="Resource">  <Key>  <PropertyRef Name="Id" />  </Key>  <Property Name="Id" Type="uniqueidentifier" Nullable="false" />  ...  <Property Name="ff0ca96d-a3b1-4980-b658-64b516825808" Type="nvarchar" Nullable="true" MaxLength="4000" />  </EntityType>  ...  <Function Name="Insertc1b2f0514a5940afb09920f0b0f3ad99" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Copyright" Type="nvarchar" Mode="In" />  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Updatec1b2f0514a5940afb09920f0b0f3ad99" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Copyright" Type="nvarchar" Mode="In" />  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Deletec1b2f0514a5940afb09920f0b0f3ad99" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  </Function>  </Schema> |

### Extended Core CSDL

The EntityContainer element is updated to include the FunctionImport definitions for CUD procedures.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityContainer Name="ZentityContext">  ...  <FunctionImport Name="Insertc1b2f0514a5940afb09920f0b0f3ad99">  <Parameter Name="Copyright" Type="String" Mode="In" />  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Updatec1b2f0514a5940afb09920f0b0f3ad99">  <Parameter Name="Copyright" Type="String" Mode="In" />  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Deletec1b2f0514a5940afb09920f0b0f3ad99">  <Parameter Name="Id" Type="Guid" Mode="In" />  </FunctionImport>  </EntityContainer>  </Schema> |

### Module CSDL

This CSDL has all the EntityType details for the custom module.

|  |
| --- |
| <Schema Namespace="Zentity.Samples" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityType Name="ScholarlyWork" BaseType="Zentity.Core.Resource">  <Property Name="Copyright" Type="String" Nullable="true" Unicode="true" MaxLength="4000" FixedLength="false" />  </EntityType>  </Schema> |

### Consolidated MSL

New EntitySetMapping element is created for the resource type. FunctionImportMapping elements are created for the CUD procedures.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Mapping Space="C-S" xmlns="urn:schemas-microsoft-com:windows:storage:mapping:CS">  <EntityContainerMapping StorageEntityContainer="Core" CdmEntityContainer="ZentityContext">  ...  <EntitySetMapping Name="Resources">  ...  <EntityTypeMapping TypeName="Zentity.Samples.ScholarlyWork">  <MappingFragment StoreEntitySet="Resource">  <ScalarProperty Name="Id" ColumnName="Id" />  <ScalarProperty Name="DateAdded" ColumnName="DateAdded" />  <ScalarProperty Name="DateModified" ColumnName="DateModified" />  <ScalarProperty Name="Description" ColumnName="Description" />  <ScalarProperty Name="Title" ColumnName="Title" />  <ScalarProperty Name="Uri" ColumnName="Uri" />  <Condition ColumnName="Discriminator" Value="3" />  <ScalarProperty Name="Copyright" ColumnName="ff0ca96d-a3b1-4980-b658-64b516825808" />  </MappingFragment>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Insertc1b2f0514a5940afb09920f0b0f3ad99">  <ScalarProperty Name="Copyright" ParameterName="Copyright" />  <ScalarProperty Name="Title" ParameterName="Title" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" />  <ScalarProperty Name="Id" ParameterName="Id" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" />  <ScalarProperty Name="Uri" ParameterName="Uri" />  <ScalarProperty Name="Description" ParameterName="Description" />  </InsertFunction>  <UpdateFunction FunctionName="Zentity.Core.Store.Updatec1b2f0514a5940afb09920f0b0f3ad99">  <ScalarProperty Name="Copyright" ParameterName="Copyright" Version="Current" />  <ScalarProperty Name="Title" ParameterName="Title" Version="Current" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" Version="Current" />  <ScalarProperty Name="Id" ParameterName="Id" Version="Current" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" Version="Current" />  <ScalarProperty Name="Uri" ParameterName="Uri" Version="Current" />  <ScalarProperty Name="Description" ParameterName="Description" Version="Current" />  </UpdateFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Deletec1b2f0514a5940afb09920f0b0f3ad99">  <ScalarProperty Name="Id" ParameterName="Id" />  </DeleteFunction>  </ModificationFunctionMapping>  </EntityTypeMapping>  </EntitySetMapping>  ...  <FunctionImportMapping FunctionImportName="Insertc1b2f0514a5940afb09920f0b0f3ad99" FunctionName="Zentity.Core.Store.Insertc1b2f0514a5940afb09920f0b0f3ad99" />  <FunctionImportMapping FunctionImportName="Updatec1b2f0514a5940afb09920f0b0f3ad99" FunctionName="Zentity.Core.Store.Updatec1b2f0514a5940afb09920f0b0f3ad99" />  <FunctionImportMapping FunctionImportName="Deletec1b2f0514a5940afb09920f0b0f3ad99" FunctionName="Zentity.Core.Store.Deletec1b2f0514a5940afb09920f0b0f3ad99" />  </EntityContainerMapping>  </Mapping> |

## Non One-To-XXX Association

### ZDM

Figure 8. ZDM for Non One-To-XXX Association.

Refer to the walkthrough ‘Modules with Non One-To-XXX Associations’ in this document for the detailed steps to define and use this data model.

### Consolidated SSDL

A view is created in the database for the association and corresponding EntitySet and EntityType elements are created in the SSDL. AssociationSet and Association elements are created to represent foreign keys from the view to Core.Resource table. Finally, CUD procedures are created for custom resource types and the association.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core.Store" Alias="Self" Provider="System.Data.SqlClient" ProviderManifestToken="2005" xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator" xmlns="http://schemas.microsoft.com/ado/2006/04/edm/ssdl">  <EntityContainer Name="Core">  ...  <EntitySet Name="3f48fbf585f848b9b6837353935ee7ce" EntityType="Zentity.Core.Store.3f48fbf585f848b9b6837353935ee7ce" store:Type="Views" store:Schema="Core" store:Name="3f48fbf585f848b9b6837353935ee7ce" />  <AssociationSet Name="FK\_d6fdd3a4-a4a0-48ea-903c-501b66f70ce3" Association="Zentity.Core.Store.FK\_d6fdd3a4-a4a0-48ea-903c-501b66f70ce3">  <End Role="3f48fbf585f848b9b6837353935ee7ce" EntitySet="3f48fbf585f848b9b6837353935ee7ce" />  <End Role="Resource" EntitySet="Resource" />  </AssociationSet>  <AssociationSet Name="FK\_fa41a3d9-57a9-4fed-9660-2814407d897d" Association="Zentity.Core.Store.FK\_fa41a3d9-57a9-4fed-9660-2814407d897d">  <End Role="3f48fbf585f848b9b6837353935ee7ce" EntitySet="3f48fbf585f848b9b6837353935ee7ce" />  <End Role="Resource" EntitySet="Resource" />  </AssociationSet>  </EntityContainer>  ...  <Function Name="Insertb94dbd8259074ec6b08f828d5ae157da" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Updateb94dbd8259074ec6b08f828d5ae157da" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Deleteb94dbd8259074ec6b08f828d5ae157da" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  </Function>  <Function Name="Inserte79b9aeb365e4f3dbe420f43efedd695" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Updatee79b9aeb365e4f3dbe420f43efedd695" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Deletee79b9aeb365e4f3dbe420f43efedd695" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  </Function>  <EntityType Name="3f48fbf585f848b9b6837353935ee7ce">  <Key>  <PropertyRef Name="SubjectResourceId" />  <PropertyRef Name="ObjectResourceId" />  </Key>  <Property Name="SubjectResourceId" Type="uniqueidentifier" Nullable="false" />  <Property Name="ObjectResourceId" Type="uniqueidentifier" Nullable="false" />  </EntityType>  <Association Name="FK\_d6fdd3a4-a4a0-48ea-903c-501b66f70ce3">  <End Role="3f48fbf585f848b9b6837353935ee7ce" Type="Zentity.Core.Store.3f48fbf585f848b9b6837353935ee7ce" Multiplicity="\*" />  <End Role="Resource" Type="Zentity.Core.Store.Resource" Multiplicity="1" />  <ReferentialConstraint>  <Principal Role="Resource">  <PropertyRef Name="Id" />  </Principal>  <Dependent Role="3f48fbf585f848b9b6837353935ee7ce">  <PropertyRef Name="SubjectResourceId" />  </Dependent>  </ReferentialConstraint>  </Association>  <Association Name="FK\_fa41a3d9-57a9-4fed-9660-2814407d897d">  <End Role="3f48fbf585f848b9b6837353935ee7ce" Type="Zentity.Core.Store.3f48fbf585f848b9b6837353935ee7ce" Multiplicity="\*" />  <End Role="Resource" Type="Zentity.Core.Store.Resource" Multiplicity="1" />  <ReferentialConstraint>  <Principal Role="Resource">  <PropertyRef Name="Id" />  </Principal>  <Dependent Role="3f48fbf585f848b9b6837353935ee7ce">  <PropertyRef Name="ObjectResourceId" />  </Dependent>  </ReferentialConstraint>  </Association>  <Function Name="Insert3f48fbf585f848b9b6837353935ee7ce" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="SubjectResourceId" Type="uniqueidentifier" Mode="In" />  <Parameter Name="ObjectResourceId" Type="uniqueidentifier" Mode="In" />  </Function>  <Function Name="Delete3f48fbf585f848b9b6837353935ee7ce" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="SubjectResourceId" Type="uniqueidentifier" Mode="In" />  <Parameter Name="ObjectResourceId" Type="uniqueidentifier" Mode="In" />  </Function>  </Schema> |

### Extended Core CSDL

The EntityContainer element now has FunctionImport elements for CUD procedures and also the AssociationSet element for the view.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityContainer Name="ZentityContext">  ...  <FunctionImport Name="Insertb94dbd8259074ec6b08f828d5ae157da">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Updateb94dbd8259074ec6b08f828d5ae157da">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Deleteb94dbd8259074ec6b08f828d5ae157da">  <Parameter Name="Id" Type="Guid" Mode="In" />  </FunctionImport>  <FunctionImport Name="Inserte79b9aeb365e4f3dbe420f43efedd695">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Updatee79b9aeb365e4f3dbe420f43efedd695">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Deletee79b9aeb365e4f3dbe420f43efedd695">  <Parameter Name="Id" Type="Guid" Mode="In" />  </FunctionImport>  <AssociationSet Name="PublicationHasAuthors" Association="Zentity.Samples.PublicationHasAuthors">  <End Role="Publication" EntitySet="Resources" />  <End Role="Person" EntitySet="Resources" />  </AssociationSet>  <FunctionImport Name="Insert3f48fbf585f848b9b6837353935ee7ce">  <Parameter Name="SubjectResourceId" Type="Guid" Mode="In" />  <Parameter Name="ObjectResourceId" Type="Guid" Mode="In" />  </FunctionImport>  <FunctionImport Name="Delete3f48fbf585f848b9b6837353935ee7ce">  <Parameter Name="SubjectResourceId" Type="Guid" Mode="In" />  <Parameter Name="ObjectResourceId" Type="Guid" Mode="In" />  </FunctionImport>  </EntityContainer>  ...  </Schema> |

### Module CSDL

This contains the EntityType and Association definitions.

|  |
| --- |
| <Schema Namespace="Zentity.Samples" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityType Name="Publication" BaseType="Zentity.Core.Resource">  <NavigationProperty Name="Authors" Relationship="Zentity.Samples.PublicationHasAuthors" FromRole="Publication" ToRole="Person" />  </EntityType>  <EntityType Name="Person" BaseType="Zentity.Core.Resource">  <NavigationProperty Name="AuthoredPublications" Relationship="Zentity.Samples.PublicationHasAuthors" FromRole="Person" ToRole="Publication" />  </EntityType>  <Association Name="PublicationHasAuthors">  <End Role="Publication" Type="Zentity.Samples.Publication" Multiplicity="\*" />  <End Role="Person" Type="Zentity.Samples.Person" Multiplicity="\*" />  </Association>  </Schema> |

### Consolidated MSL

Mapping elements are created for entity types, associations and CUD procedures.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Mapping Space="C-S" xmlns="urn:schemas-microsoft-com:windows:storage:mapping:CS">  <EntityContainerMapping StorageEntityContainer="Core" CdmEntityContainer="ZentityContext">  ...  <EntitySetMapping Name="Resources">  ...  <EntityTypeMapping TypeName="Zentity.Samples.Publication">  <MappingFragment StoreEntitySet="Resource">  <ScalarProperty Name="Id" ColumnName="Id" />  <ScalarProperty Name="DateAdded" ColumnName="DateAdded" />  <ScalarProperty Name="DateModified" ColumnName="DateModified" />  <ScalarProperty Name="Description" ColumnName="Description" />  <ScalarProperty Name="Title" ColumnName="Title" />  <ScalarProperty Name="Uri" ColumnName="Uri" />  <Condition ColumnName="Discriminator" Value="3" />  </MappingFragment>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Insertb94dbd8259074ec6b08f828d5ae157da">  <ScalarProperty Name="Title" ParameterName="Title" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" />  <ScalarProperty Name="Id" ParameterName="Id" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" />  <ScalarProperty Name="Uri" ParameterName="Uri" />  <ScalarProperty Name="Description" ParameterName="Description" />  </InsertFunction>  <UpdateFunction FunctionName="Zentity.Core.Store.Updateb94dbd8259074ec6b08f828d5ae157da">  <ScalarProperty Name="Title" ParameterName="Title" Version="Current" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" Version="Current" />  <ScalarProperty Name="Id" ParameterName="Id" Version="Current" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" Version="Current" />  <ScalarProperty Name="Uri" ParameterName="Uri" Version="Current" />  <ScalarProperty Name="Description" ParameterName="Description" Version="Current" />  </UpdateFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Deleteb94dbd8259074ec6b08f828d5ae157da">  <ScalarProperty Name="Id" ParameterName="Id" />  </DeleteFunction>  </ModificationFunctionMapping>  </EntityTypeMapping>  <EntityTypeMapping TypeName="Zentity.Samples.Person">  <MappingFragment StoreEntitySet="Resource">  <ScalarProperty Name="Id" ColumnName="Id" />  <ScalarProperty Name="DateAdded" ColumnName="DateAdded" />  <ScalarProperty Name="DateModified" ColumnName="DateModified" />  <ScalarProperty Name="Description" ColumnName="Description" />  <ScalarProperty Name="Title" ColumnName="Title" />  <ScalarProperty Name="Uri" ColumnName="Uri" />  <Condition ColumnName="Discriminator" Value="4" />  </MappingFragment>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Inserte79b9aeb365e4f3dbe420f43efedd695">  <ScalarProperty Name="Title" ParameterName="Title" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" />  <ScalarProperty Name="Id" ParameterName="Id" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" />  <ScalarProperty Name="Uri" ParameterName="Uri" />  <ScalarProperty Name="Description" ParameterName="Description" />  </InsertFunction>  <UpdateFunction FunctionName="Zentity.Core.Store.Updatee79b9aeb365e4f3dbe420f43efedd695">  <ScalarProperty Name="Title" ParameterName="Title" Version="Current" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" Version="Current" />  <ScalarProperty Name="Id" ParameterName="Id" Version="Current" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" Version="Current" />  <ScalarProperty Name="Uri" ParameterName="Uri" Version="Current" />  <ScalarProperty Name="Description" ParameterName="Description" Version="Current" />  </UpdateFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Deletee79b9aeb365e4f3dbe420f43efedd695">  <ScalarProperty Name="Id" ParameterName="Id" />  </DeleteFunction>  </ModificationFunctionMapping>  </EntityTypeMapping>  </EntitySetMapping>  ...  <FunctionImportMapping FunctionImportName="Insertb94dbd8259074ec6b08f828d5ae157da" FunctionName="Zentity.Core.Store.Insertb94dbd8259074ec6b08f828d5ae157da" />  <FunctionImportMapping FunctionImportName="Updateb94dbd8259074ec6b08f828d5ae157da" FunctionName="Zentity.Core.Store.Updateb94dbd8259074ec6b08f828d5ae157da" />  <FunctionImportMapping FunctionImportName="Deleteb94dbd8259074ec6b08f828d5ae157da" FunctionName="Zentity.Core.Store.Deleteb94dbd8259074ec6b08f828d5ae157da" />  <FunctionImportMapping FunctionImportName="Inserte79b9aeb365e4f3dbe420f43efedd695" FunctionName="Zentity.Core.Store.Inserte79b9aeb365e4f3dbe420f43efedd695" />  <FunctionImportMapping FunctionImportName="Updatee79b9aeb365e4f3dbe420f43efedd695" FunctionName="Zentity.Core.Store.Updatee79b9aeb365e4f3dbe420f43efedd695" />  <FunctionImportMapping FunctionImportName="Deletee79b9aeb365e4f3dbe420f43efedd695" FunctionName="Zentity.Core.Store.Deletee79b9aeb365e4f3dbe420f43efedd695" />  <FunctionImportMapping FunctionImportName="Insert3f48fbf585f848b9b6837353935ee7ce" FunctionName="Zentity.Core.Store.Insert3f48fbf585f848b9b6837353935ee7ce" />  <FunctionImportMapping FunctionImportName="Delete3f48fbf585f848b9b6837353935ee7ce" FunctionName="Zentity.Core.Store.Delete3f48fbf585f848b9b6837353935ee7ce" />  <AssociationSetMapping Name="PublicationHasAuthors" TypeName="Zentity.Samples.PublicationHasAuthors" StoreEntitySet="3f48fbf585f848b9b6837353935ee7ce">  <EndProperty Name="Publication">  <ScalarProperty Name="Id" ColumnName="SubjectResourceId" />  </EndProperty>  <EndProperty Name="Person">  <ScalarProperty Name="Id" ColumnName="ObjectResourceId" />  </EndProperty>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Insert3f48fbf585f848b9b6837353935ee7ce">  <EndProperty Name="Publication">  <ScalarProperty Name="Id" ParameterName="SubjectResourceId" />  </EndProperty>  <EndProperty Name="Person">  <ScalarProperty Name="Id" ParameterName="ObjectResourceId" />  </EndProperty>  </InsertFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Delete3f48fbf585f848b9b6837353935ee7ce">  <EndProperty Name="Publication">  <ScalarProperty Name="Id" ParameterName="SubjectResourceId" />  </EndProperty>  <EndProperty Name="Person">  <ScalarProperty Name="Id" ParameterName="ObjectResourceId" />  </EndProperty>  </DeleteFunction>  </ModificationFunctionMapping>  </AssociationSetMapping>  </EntityContainerMapping>  </Mapping> |

## One-To-XXX Association

### ZDM

Figure 9. ZDM for One-To-XXX Association.

Refer to the walkthrough ‘Modules with One-To-XXX Associations’ in this document for the detailed steps to define and use this data model.

### Consolidated SSDL

A foreign key column is created in Core.Resource table to represent the association. No CUD procedure is created for the association.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core.Store" Alias="Self" Provider="System.Data.SqlClient" ProviderManifestToken="2005" xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator" xmlns="http://schemas.microsoft.com/ado/2006/04/edm/ssdl">  <EntityContainer Name="Core">  ...  <AssociationSet Name="FK\_0cc74a63-9303-49ba-b316-6f398c31a2e5" Association="Zentity.Core.Store.FK\_0cc74a63-9303-49ba-b316-6f398c31a2e5">  <End Role="Resource" EntitySet="Resource" />  <End Role="0cc74a63-9303-49ba-b316-6f398c31a2e5" EntitySet="Resource" />  </AssociationSet>  </EntityContainer>  ...  <EntityType Name="Resource">  <Key>  <PropertyRef Name="Id" />  </Key>  <Property Name="Id" Type="uniqueidentifier" Nullable="false" />  ...  <Property Name="0cc74a63-9303-49ba-b316-6f398c31a2e5" Type="uniqueidentifier" Nullable="true" />  </EntityType>  ...  <Function Name="Inserte40550f097974b6b9837015298ce6ed9" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Updatee40550f097974b6b9837015298ce6ed9" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Deletee40550f097974b6b9837015298ce6ed9" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  </Function>  <Function Name="Insertd64011cdc0034e8d98bd404337842004" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Order" Type="uniqueidentifier" Mode="In" />  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Updated64011cdc0034e8d98bd404337842004" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Order" Type="uniqueidentifier" Mode="In" />  <Parameter Name="Title" Type="nvarchar" Mode="In" />  <Parameter Name="DateModified" Type="datetime" Mode="In" />  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="DateAdded" Type="datetime" Mode="In" />  <Parameter Name="Uri" Type="nvarchar" Mode="In" />  <Parameter Name="Description" Type="nvarchar(max)" Mode="In" />  </Function>  <Function Name="Deleted64011cdc0034e8d98bd404337842004" Aggregate="false" BuiltIn="false" NiladicFunction="false" IsComposable="false" ParameterTypeSemantics="AllowImplicitConversion" Schema="Core">  <Parameter Name="Id" Type="uniqueidentifier" Mode="In" />  <Parameter Name="Order" Type="uniqueidentifier" Mode="In" />  </Function>  <Association Name="FK\_0cc74a63-9303-49ba-b316-6f398c31a2e5">  <End Role="Resource" Type="Zentity.Core.Store.Resource" Multiplicity="0..1" />  <End Role="0cc74a63-9303-49ba-b316-6f398c31a2e5" Type="Zentity.Core.Store.Resource" Multiplicity="\*" />  <ReferentialConstraint>  <Principal Role="Resource">  <PropertyRef Name="Id" />  </Principal>  <Dependent Role="0cc74a63-9303-49ba-b316-6f398c31a2e5">  <PropertyRef Name="0cc74a63-9303-49ba-b316-6f398c31a2e5" />  </Dependent>  </ReferentialConstraint>  </Association>  </Schema> |

### Extended Core CSDL

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Schema Namespace="Zentity.Core" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityContainer Name="ZentityContext">  ...  <FunctionImport Name="Inserte40550f097974b6b9837015298ce6ed9">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Updatee40550f097974b6b9837015298ce6ed9">  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Deletee40550f097974b6b9837015298ce6ed9">  <Parameter Name="Id" Type="Guid" Mode="In" />  </FunctionImport>  <FunctionImport Name="Insertd64011cdc0034e8d98bd404337842004">  <Parameter Name="Order" Type="Guid" Mode="In" />  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Updated64011cdc0034e8d98bd404337842004">  <Parameter Name="Order" Type="Guid" Mode="In" />  <Parameter Name="Title" Type="String" Mode="In" />  <Parameter Name="DateModified" Type="DateTime" Mode="In" />  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="DateAdded" Type="DateTime" Mode="In" />  <Parameter Name="Uri" Type="String" Mode="In" />  <Parameter Name="Description" Type="String" Mode="In" />  </FunctionImport>  <FunctionImport Name="Deleted64011cdc0034e8d98bd404337842004">  <Parameter Name="Id" Type="Guid" Mode="In" />  <Parameter Name="Order" Type="Guid" Mode="In" />  </FunctionImport>  <AssociationSet Name="OrderHasLineItems" Association="Zentity.Samples.OrderHasLineItems">  <End Role="Order" EntitySet="Resources" />  <End Role="LineItem" EntitySet="Resources" />  </AssociationSet>  </EntityContainer>  ...  </Schema> |

### Module CSDL

|  |
| --- |
| <Schema Namespace="Zentity.Samples" Alias="Self" xmlns="http://schemas.microsoft.com/ado/2006/04/edm">  <EntityType Name="Order" BaseType="Zentity.Core.Resource">  <NavigationProperty Name="LineItems" Relationship="Zentity.Samples.OrderHasLineItems" FromRole="Order" ToRole="LineItem" />  </EntityType>  <EntityType Name="LineItem" BaseType="Zentity.Core.Resource">  <NavigationProperty Name="Order" Relationship="Zentity.Samples.OrderHasLineItems" FromRole="LineItem" ToRole="Order" />  </EntityType>  <Association Name="OrderHasLineItems">  <End Role="Order" Type="Zentity.Samples.Order" Multiplicity="1" />  <End Role="LineItem" Type="Zentity.Samples.LineItem" Multiplicity="\*" />  </Association>  </Schema> |

### Consolidated MSL

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Mapping Space="C-S" xmlns="urn:schemas-microsoft-com:windows:storage:mapping:CS">  <EntityContainerMapping StorageEntityContainer="Core" CdmEntityContainer="ZentityContext">  ...  <EntitySetMapping Name="Resources">  ...  <EntityTypeMapping TypeName="Zentity.Samples.Order">  <MappingFragment StoreEntitySet="Resource">  <ScalarProperty Name="Id" ColumnName="Id" />  <ScalarProperty Name="DateAdded" ColumnName="DateAdded" />  <ScalarProperty Name="DateModified" ColumnName="DateModified" />  <ScalarProperty Name="Description" ColumnName="Description" />  <ScalarProperty Name="Title" ColumnName="Title" />  <ScalarProperty Name="Uri" ColumnName="Uri" />  <Condition ColumnName="Discriminator" Value="3" />  </MappingFragment>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Inserte40550f097974b6b9837015298ce6ed9">  <ScalarProperty Name="Title" ParameterName="Title" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" />  <ScalarProperty Name="Id" ParameterName="Id" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" />  <ScalarProperty Name="Uri" ParameterName="Uri" />  <ScalarProperty Name="Description" ParameterName="Description" />  </InsertFunction>  <UpdateFunction FunctionName="Zentity.Core.Store.Updatee40550f097974b6b9837015298ce6ed9">  <ScalarProperty Name="Title" ParameterName="Title" Version="Current" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" Version="Current" />  <ScalarProperty Name="Id" ParameterName="Id" Version="Current" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" Version="Current" />  <ScalarProperty Name="Uri" ParameterName="Uri" Version="Current" />  <ScalarProperty Name="Description" ParameterName="Description" Version="Current" />  </UpdateFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Deletee40550f097974b6b9837015298ce6ed9">  <ScalarProperty Name="Id" ParameterName="Id" />  </DeleteFunction>  </ModificationFunctionMapping>  </EntityTypeMapping>  <EntityTypeMapping TypeName="Zentity.Samples.LineItem">  <MappingFragment StoreEntitySet="Resource">  <ScalarProperty Name="Id" ColumnName="Id" />  <ScalarProperty Name="DateAdded" ColumnName="DateAdded" />  <ScalarProperty Name="DateModified" ColumnName="DateModified" />  <ScalarProperty Name="Description" ColumnName="Description" />  <ScalarProperty Name="Title" ColumnName="Title" />  <ScalarProperty Name="Uri" ColumnName="Uri" />  <Condition ColumnName="Discriminator" Value="4" />  </MappingFragment>  <ModificationFunctionMapping>  <InsertFunction FunctionName="Zentity.Core.Store.Insertd64011cdc0034e8d98bd404337842004">  <AssociationEnd AssociationSet="OrderHasLineItems" From="LineItem" To="Order">  <ScalarProperty Name="Id" ParameterName="Order" />  </AssociationEnd>  <ScalarProperty Name="Title" ParameterName="Title" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" />  <ScalarProperty Name="Id" ParameterName="Id" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" />  <ScalarProperty Name="Uri" ParameterName="Uri" />  <ScalarProperty Name="Description" ParameterName="Description" />  </InsertFunction>  <UpdateFunction FunctionName="Zentity.Core.Store.Updated64011cdc0034e8d98bd404337842004">  <AssociationEnd AssociationSet="OrderHasLineItems" From="LineItem" To="Order">  <ScalarProperty Name="Id" ParameterName="Order" Version="Current" />  </AssociationEnd>  <ScalarProperty Name="Title" ParameterName="Title" Version="Current" />  <ScalarProperty Name="DateModified" ParameterName="DateModified" Version="Current" />  <ScalarProperty Name="Id" ParameterName="Id" Version="Current" />  <ScalarProperty Name="DateAdded" ParameterName="DateAdded" Version="Current" />  <ScalarProperty Name="Uri" ParameterName="Uri" Version="Current" />  <ScalarProperty Name="Description" ParameterName="Description" Version="Current" />  </UpdateFunction>  <DeleteFunction FunctionName="Zentity.Core.Store.Deleted64011cdc0034e8d98bd404337842004">  <ScalarProperty Name="Id" ParameterName="Id" />  <AssociationEnd AssociationSet="OrderHasLineItems" From="LineItem" To="Order">  <ScalarProperty Name="Id" ParameterName="Order" />  </AssociationEnd>  </DeleteFunction>  </ModificationFunctionMapping>  </EntityTypeMapping>  </EntitySetMapping>  ...  <FunctionImportMapping FunctionImportName="Inserte40550f097974b6b9837015298ce6ed9" FunctionName="Zentity.Core.Store.Inserte40550f097974b6b9837015298ce6ed9" />  <FunctionImportMapping FunctionImportName="Updatee40550f097974b6b9837015298ce6ed9" FunctionName="Zentity.Core.Store.Updatee40550f097974b6b9837015298ce6ed9" />  <FunctionImportMapping FunctionImportName="Deletee40550f097974b6b9837015298ce6ed9" FunctionName="Zentity.Core.Store.Deletee40550f097974b6b9837015298ce6ed9" />  <FunctionImportMapping FunctionImportName="Insertd64011cdc0034e8d98bd404337842004" FunctionName="Zentity.Core.Store.Insertd64011cdc0034e8d98bd404337842004" />  <FunctionImportMapping FunctionImportName="Updated64011cdc0034e8d98bd404337842004" FunctionName="Zentity.Core.Store.Updated64011cdc0034e8d98bd404337842004" />  <FunctionImportMapping FunctionImportName="Deleted64011cdc0034e8d98bd404337842004" FunctionName="Zentity.Core.Store.Deleted64011cdc0034e8d98bd404337842004" />  <AssociationSetMapping Name="OrderHasLineItems" TypeName="Zentity.Samples.OrderHasLineItems" StoreEntitySet="Resource">  <EndProperty Name="Order">  <ScalarProperty Name="Id" ColumnName="0cc74a63-9303-49ba-b316-6f398c31a2e5" />  </EndProperty>  <EndProperty Name="LineItem">  <ScalarProperty Name="Id" ColumnName="Id" />  </EndProperty>  <Condition ColumnName="0cc74a63-9303-49ba-b316-6f398c31a2e5" IsNull="false" />  </AssociationSetMapping>  </EntityContainerMapping>  </Mapping> |

# Performance Considerations

## Creating Indexes for Scalar Properties

The Extensibility API creates only basic database objects required for the data model. The API does not provide index management features for database columns. Creation of these indexes is, however, a fairly simple task. To figure out the column mapped to a scalar property run a query similar to the following. All columns are created in Core.Resource table in this version of Zentity.

|  |
| --- |
| SELECT [RT].[Name] [ResourceType], [SP].[Name] [ScalarProperty],  [SP].[TableName], [SP].[ColumnName]  FROM [Core].[ResourceType] [RT]  INNER JOIN [Core].[ScalarProperty] [SP]  ON [RT].[Id] = [SP].ResourceTypeId  ORDER BY [RT].[Name], [SP].[Name]; |

Thereafter an index can be created using the query as shown below.

|  |
| --- |
| CREATE NONCLUSTERED INDEX [IX\_Book\_ISBN] ON [Core].[Resource]  (  [bc13df5d-58dc-4fc7-8010-ec7eb04ac0c0] ASC,  [Discriminator] ASC,  [Id] ASC  )  INCLUDE  (  [Title],  [Uri]  ) |

## Creating Indexes for Association Views

Likewise, you can also create indexes for association views. The mapping information of views can be retrieved by executing a query similar to the following. All views are created in the Core schema.

|  |
| --- |
| SELECT [ViewName]  FROM [Core].[Association] [A]; |

## Creating Full-Text Indexes on String Properties

The limitation of the above method is the maximum allowed index key size. SQL Server allows index keys to be a maximum of 900 bytes. Thus, we cannot create indexes on columns of type, for example, nvarchar(4000).

Zentity provides a limited API to create basic full text index infrastructure. Execute the following code to create the full text catalog.

|  |
| --- |
| using System;  using Zentity.Administration;  namespace ZentitySamples  {  class Program  {  const string connectionString = @"provider=System.Data.SqlClient;  metadata=res://Zentity.Core; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (AdministrationContext context = new AdministrationContext(connectionString))  {  context.EnableFullTextSearch(@"C:\FullTextCatalog.ndf");  }  }  }  } |

The code above creates a new full text catalog, ZentityCatalog, and a full text index on the Resource table. Execute the following script to create full text index for ‘CopyRight’ scalar property.

|  |
| --- |
| USE [Zentity]  GO  ALTER FULLTEXT INDEX ON [Core].[Resource] ADD ([bec19f51-7fe7-45a8-9a7a-88b008ade630])  GO |

Execute the following code to get the full text information of scalar properties.

|  |
| --- |
| using Zentity.Core;  using System;  using System.Linq;  namespace ZentitySamples  {  class Program  {  const string connectionString = @"provider=System.Data.SqlClient;  metadata=res://Zentity.Core; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionString))  {  Console.WriteLine("Full text indexed properties...");  foreach (ScalarProperty sp in context.DataModel.Modules.  SelectMany(module => module.ResourceTypes).  SelectMany(resourceType => resourceType.ScalarProperties).  Where(scalarProperty => scalarProperty.IsFullTextIndexed))  {  Console.WriteLine(" {0}", sp.Parent.FullName + "." + sp.Name);  }  }  }  }  } |

These full text indexes will be used internally by the Search API. Entity Framework does not provide a mechanism to leverage full text indexes.

## Entity Framework Best Practices

Following articles provide information about the performance considerations for Entity Framework applications.

Performance Considerations for Entity Framework Applications - <http://msdn.microsoft.com/en-us/library/cc853327.aspx>

Exploring the Performance of the ADO.NET Entity Framework - Part 1 - <http://blogs.msdn.com/adonet/archive/2008/02/04/exploring-the-performance-of-the-ado-net-entity-framework-part-1.aspx>

Exploring the Performance of the ADO.NET Entity Framework – Part 2 - <http://blogs.msdn.com/adonet/archive/2008/02/11/exploring-the-performance-of-the-ado-net-entity-framework-part-2.aspx>

# Walkthroughs

Each of the following walkthroughs require the Zentity database to be recreated. Creation of Zentity database is a two step process. First we create a blank database with a FILESTREAM filegroup. Second, we create all the database objects within the context of the database created in the first step.

## Re-creating Zentity Database

Execute a script similar to the following to create a blank database with a FILESTREAM filegroup.

|  |
| --- |
| USE [master]  GO  IF DB\_ID('Zentity') IS NOT NULL  BEGIN  EXEC msdb.dbo.sp\_delete\_database\_backuphistory @database\_name = N'Zentity'  ALTER DATABASE [Zentity] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE  DROP DATABASE [Zentity]  END  GO  CREATE DATABASE [Zentity];  GO  ALTER DATABASE [Zentity]  ADD FILEGROUP FilestreamGroup CONTAINS FILESTREAM;  GO  ALTER DATABASE [Zentity]  ADD FILE (NAME = FilestreamFile , FILENAME = 'C:\Filestream.ndf')  TO FILEGROUP FilestreamGroup  GO  ALTER DATABASE [Zentity] COLLATE SQL\_Latin1\_General\_CP1\_CI\_AI  GO |

Execute <Scripts folder path>\Database.sql within the context of the database created above. This script will create all Zentity database objects in the current database.

## Working with data model modules

Re-create the Zentity database before this walkthrough. Detailed instructions for creating the database are provided in the walkthrough ‘Re-creating Zentity Database’.

### Creating Module, Assembly and EF Artifacts

Use the code snippet below to create a simple data model module and generate the extensions assembly for it.

|  |  |
| --- | --- |
| using Zentity.Core;  using System.IO;  using System.Linq;  namespace ZentitySamples  {  class Program  {  const string connectionString = @"provider=System.Data.SqlClient;  metadata=res://Zentity.Core; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionString))  {  // Create a new module.  DataModelModule module = new DataModelModule { NameSpace = "Zentity.Samples" };  context.DataModel.Modules.Add(module);  // Create the ScholarlyWork type.  ResourceType resourceTypeResource = context.DataModel.Modules["Zentity.Core"].ResourceTypes["Resource"];  ResourceType resourceTypeScholarlyWork = new ResourceType { Name = "ScholarlyWork", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypeScholarlyWork);  // Create some Scalar Properties.  ScalarProperty Copyright = new ScalarProperty { Name = "Copyright", DataType = DataTypes.String, MaxLength = 4000 };  resourceTypeScholarlyWork.ScalarProperties.Add(Copyright);  // Synchronize to alter the database schema.  context.DataModel.Synchronize();  // Generate Extensions Assembly.  using (FileStream fout = new FileStream(@"C:\Zentity.Samples.dll", FileMode.Create, FileAccess.Write))  {  byte[] rawAssembly = context.DataModel.GenerateExtensionsAssembly(  "Zentity.Samples", false, null, new string[] { "Zentity.Samples" }, null);  fout.Write(rawAssembly, 0, rawAssembly.Length);  }  // Generate Entity Framework artifacts.  EFArtifactGenerationResults results = context.DataModel.GenerateEFArtifacts("Zentity.Samples");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Core").First().Value.Save(@"C:\ExtendedCore.csdl");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Samples").First().Value.Save(@"C:\Zentity.Samples.csdl");  results.Msl.Save(@"C:\Consolidated.msl");  results.Ssdl.Save(@"C:\Consolidated.ssdl");  }  }  }  } |  |

### Using the Generated Assembly and Artifacts

Create a console application and add references to the extensions assembly generated in previous step. Update Program.cs as shown below. Take note of the metadata path in the connection string. This points to the Entity Framework artifacts generated in the previous step and may be different for you.

|  |
| --- |
| using Zentity.Core;  using Zentity.Samples;  using System;  namespace ZentitySamples  {  class Program  {  static void Main(string[] args)  {  const string connectionString = @"provider=System.Data.SqlClient;  metadata=C:\ExtendedCore.csdl|C:\Zentity.Samples.csdl|C:\Consolidated.msl|C:\Consolidated.ssdl;  provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  using (ZentityContext context = new ZentityContext(connectionString))  {  // Create resources.  ScholarlyWork book = new ScholarlyWork { Title = "A Book", Copyright = "Some Copyright" };  context.AddToResources(book);  context.SaveChanges();  // Retrieve resources.  DisplayScholarlyWorks(context);  // Update resources.  book.Copyright = "A new copyright";  context.SaveChanges();  DisplayScholarlyWorks(context);  // Delete resources.  context.DeleteObject(book);  context.SaveChanges();  DisplayScholarlyWorks(context);  }  }  private static void DisplayScholarlyWorks(ZentityContext context)  {  foreach (ScholarlyWork sw in context.Resources.OfType<ScholarlyWork>())  Console.WriteLine("Id: [{0}], Copyright: [{1}]", sw.Id, sw.Copyright);  Console.WriteLine("----------------------------");  }  }  } |

## Generating Assembly with Embedded Entity Framework Artifacts

Re-create the Zentity database before this walkthrough. Detailed instructions for creating the database is provided in the walkthrough ‘Re-creating Zentity Database’.

This example shows how to create an extensions assembly with embedded Entity Framework artifacts and uses the generated assembly via .NET reflection. This way, we can use the generated assembly from a running application rather than compiling the application for the extensions assembly.

Notice how we use the embedded metadata from Zentity.Core for creating the extensions assembly and then use the embedded metadata from the extensions assembly for creating and retrieving resources.

|  |
| --- |
| using Zentity.Core;  using System.Xml;  using System.Reflection;  using System;  using System.Collections;  using System.Linq;  namespace ZentitySamples  {  class Program  {  const string extensionsNamespace = "Zentity.Extensions.ScholarlyWorks";  const string connectionStringFormat = @"provider=System.Data.SqlClient;  metadata=res://{0}; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  const string extensionsAssemblyName = "Zentity.Extensions.ScholarlyWorks";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(  string.Format(connectionStringFormat, "Zentity.Core")))  {  // Create a new module.  DataModelModule module = new DataModelModule { NameSpace = extensionsNamespace };  context.DataModel.Modules.Add(module);  // Create the resource type.  ResourceType resourceTypeResource = context.DataModel.Modules["Zentity.Core"].ResourceTypes["Resource"];  ResourceType resourceTypeScholarlyWork = new ResourceType { Name = "ScholarlyWork", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypeScholarlyWork);  // Create some Scalar Properties.  ScalarProperty copyright = new ScalarProperty { Name = "Copyright", DataType = DataTypes.String, MaxLength = 4000 };  resourceTypeScholarlyWork.ScalarProperties.Add(copyright);  // Synchronize to alter the database schema.  context.DataModel.Synchronize();  // Generate Extensions Assembly.  // NOTE: We have embedded the generated Entity Framework artifacts in the assembly itself.  byte[] rawAssembly = context.DataModel.GenerateExtensionsAssembly(  extensionsAssemblyName, true, new string[] { extensionsNamespace },  new string[] { extensionsNamespace }, null);  Assembly extensions = Assembly.Load(rawAssembly);  // Create some repository items using the generated assembly.  CreateRepositoryItems(extensions);  // Retrieve the created repository items.  FetchRepositoryItems(extensions);  }  }  private static void FetchRepositoryItems(Assembly extensionsAssembly)  {  using (ZentityContext context = new ZentityContext(  string.Format(connectionStringFormat, extensionsAssemblyName)))  {  Console.WriteLine("Getting ScholarlyWorks...");  Type resourceTypeScholarlyWork = extensionsAssembly.GetType(extensionsNamespace + ".ScholarlyWork");  PropertyInfo pi = resourceTypeScholarlyWork.GetProperty("Copyright");  MethodInfo ofTypeMethod = context.Resources.GetType().GetMethod("OfType").  MakeGenericMethod(resourceTypeScholarlyWork);  var customTypeInstances = ofTypeMethod.Invoke(context.Resources, null);  foreach (Resource scholarlyWork in (IEnumerable)customTypeInstances)  {  Console.WriteLine("Id:[{0}], Copyright:[{1}]", scholarlyWork.Id,  pi.GetValue(scholarlyWork, null));  }  }  }  private static void CreateRepositoryItems(Assembly extensionsAssembly)  {  using (ZentityContext context = new ZentityContext(  string.Format(connectionStringFormat, extensionsAssemblyName)))  {  Type resourceTypeScholarlyWork = extensionsAssembly.GetType(extensionsNamespace + ".ScholarlyWork");  var aScholarlyWork = Activator.CreateInstance(resourceTypeScholarlyWork);  PropertyInfo pi = resourceTypeScholarlyWork.GetProperty("Copyright");  pi.SetValue(aScholarlyWork, "A copyright value.", null);  // Save the items to repository.  context.AddToResources((Resource)aScholarlyWork);  context.SaveChanges();  }  }  }  } |

## Modules with One-To-XXX Associations

Re-create the Zentity database before this walkthrough. Detailed instructions for creating the database is provided in the walkthrough ‘Re-creating Zentity Database’.

### Creating Module, Assembly and EF Artifacts

|  |
| --- |
| using Zentity.Core;  using System.Linq;  using System.IO;  namespace ZentitySamples  {  class Program  {  const string connectionStringFormat = @"provider=System.Data.SqlClient;  metadata=res://Zentity.Core; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionStringFormat))  {  ResourceType resourceTypeResource = context.DataModel.Modules["Zentity.Core"].ResourceTypes["Resource"];  // Create module.  DataModelModule module = new DataModelModule { NameSpace = "Zentity.Samples" };  context.DataModel.Modules.Add(module);  // Create parent resource type.  ResourceType resourceTypeOrder = new ResourceType { Name = "Order", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypeOrder);  NavigationProperty navLineItems = new NavigationProperty { Name = "LineItems" };  resourceTypeOrder.NavigationProperties.Add(navLineItems);  // Create dependent resource type.  ResourceType resourceTypeLineItem = new ResourceType { Name = "LineItem", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypeLineItem);  NavigationProperty navOrder = new NavigationProperty { Name = "Order" };  resourceTypeLineItem.NavigationProperties.Add(navOrder);  // Create association.  Association association = new Association  {  Name = "OrderHasLineItems",  SubjectNavigationProperty = navLineItems,  ObjectNavigationProperty = navOrder,  SubjectMultiplicity = AssociationEndMultiplicity.One,  ObjectMultiplicity = AssociationEndMultiplicity.Many  };  // Synchronize the in-memory models with the backend.  context.DataModel.Synchronize();  // Generate Extensions Assembly.  using (FileStream fout = new FileStream(@"C:\Zentity.Samples.dll", FileMode.Create, FileAccess.Write))  {  byte[] rawAssembly = context.DataModel.GenerateExtensionsAssembly(  "Zentity.Samples", false, null, new string[] { "Zentity.Samples" }, null);  fout.Write(rawAssembly, 0, rawAssembly.Length);  }  // Generate Entity Framework artifacts.  EFArtifactGenerationResults results = context.DataModel.GenerateEFArtifacts("Zentity.Samples");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Core").First().Value.Save(@"C:\ExtendedCore.csdl");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Samples").First().Value.Save(@"C:\Zentity.Samples.csdl");  results.Msl.Save(@"C:\Consolidated.msl");  results.Ssdl.Save(@"C:\Consolidated.ssdl");  }  }  }  } |

### Using the Generated Assembly and Artifacts

|  |
| --- |
| using Zentity.Core;  using Zentity.Samples;  using System;  namespace ZentitySamples  {  class Program  {  static string connectionString = @"provider=System.Data.SqlClient;  metadata=C:\ExtendedCore.csdl|C:\Zentity.Samples.csdl|C:\Consolidated.msl|C:\Consolidated.ssdl;  provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionString))  {  // Create resources.  Order o = new Order();  LineItem li1 = new LineItem();  LineItem li2 = new LineItem();  // Create relationships.  o.LineItems.Add(li1);  li2.Order = o;  // Save changes.  context.AddToResources(o);  context.SaveChanges();  // Retrieve resources.  DisplayRepositoryItems();  // Update associations.  Order o2 = new Order();  li1.Order = o2;  context.SaveChanges();  DisplayRepositoryItems();  // Delete items.  context.DeleteObject(li1);  context.SaveChanges();  DisplayRepositoryItems();  }  }  private static void DisplayRepositoryItems()  {  using (ZentityContext context = new ZentityContext(connectionString))  {  foreach (var order in context.Resources.OfType<Order>())  {  Console.WriteLine("Order Id: [{0}]", order.Id);  order.LineItems.Load();  foreach (var lineItem in order.LineItems)  Console.WriteLine("\tLineItem Id: [{0}]", lineItem.Id);  }  Console.WriteLine("----------------------------");  }  }  }  } |

## Modules with Non One-To-XXX Associations

Re-create the Zentity database before this walkthrough. Detailed instructions for creating the database are provided in the walkthrough ‘Re-creating Zentity Database’.

### Creating Module, Assembly and EF Artifacts

|  |
| --- |
| using Zentity.Core;  using System.Linq;  using System.IO;  namespace ZentitySamples  {  class Program  {  const string connectionStringFormat = @"provider=System.Data.SqlClient;  metadata=res://Zentity.Core; provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionStringFormat))  {  ResourceType resourceTypeResource = context.DataModel.Modules["Zentity.Core"].ResourceTypes["Resource"];  // Create module.  DataModelModule module = new DataModelModule { NameSpace = "Zentity.Samples" };  context.DataModel.Modules.Add(module);  // Create first resource type.  ResourceType resourceTypePublication = new ResourceType { Name = "Publication", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypePublication);  NavigationProperty navAuthors = new NavigationProperty { Name = "Authors" };  resourceTypePublication.NavigationProperties.Add(navAuthors);  // Create second resource type.  ResourceType resourceTypePerson = new ResourceType { Name = "Person", BaseType = resourceTypeResource };  module.ResourceTypes.Add(resourceTypePerson);  NavigationProperty navAuthoredPublications = new NavigationProperty { Name = "AuthoredPublications" };  resourceTypePerson.NavigationProperties.Add(navAuthoredPublications);  // Create association.  Association association = new Association  {  Name = "PublicationHasAuthors",  SubjectNavigationProperty = navAuthors,  ObjectNavigationProperty = navAuthoredPublications,  SubjectMultiplicity = AssociationEndMultiplicity.Many,  ObjectMultiplicity = AssociationEndMultiplicity.Many  };  // Synchronize the in-memory models with the backend.  context.DataModel.Synchronize();  // Generate Extensions Assembly.  using (FileStream fout = new FileStream(@"C:\Zentity.Samples.dll", FileMode.Create, FileAccess.Write))  {  byte[] rawAssembly = context.DataModel.GenerateExtensionsAssembly(  "Zentity.Samples", false, null, new string[] { "Zentity.Samples" }, null);  fout.Write(rawAssembly, 0, rawAssembly.Length);  }  // Generate Entity Framework artifacts.  EFArtifactGenerationResults results = context.DataModel.GenerateEFArtifacts("Zentity.Samples");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Core").First().Value.Save(@"C:\ExtendedCore.csdl");  results.Csdls.Where(tuple => tuple.Key == "Zentity.Samples").First().Value.Save(@"C:\Zentity.Samples.csdl");  results.Msl.Save(@"C:\Consolidated.msl");  results.Ssdl.Save(@"C:\Consolidated.ssdl");  }  }  }  } |

### Using the Generated Assembly and Artifacts

|  |
| --- |
| using Zentity.Core;  using Zentity.Samples;  using System;  using System.Collections.Generic;  namespace ZentitySamples  {  class Program  {  static string connectionString = @"provider=System.Data.SqlClient;  metadata=C:\ExtendedCore.csdl|C:\Zentity.Samples.csdl|C:\Consolidated.msl|C:\Consolidated.ssdl;  provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionString))  {  // Create resources.  Publication pub1 = new Publication();  Publication pub2 = new Publication();  Person per1 = new Person();  Person per2 = new Person();  // Create relationships.  pub1.Authors.Add(per1);  per2.AuthoredPublications.Add(pub2);  // Save changes.  context.AddToResources(pub1);  context.AddToResources(pub2);  context.SaveChanges();  // Retrieve resources.  DisplayRepositoryItems();  // Update relationships.  pub1.Authors.Clear();  context.SaveChanges();  DisplayRepositoryItems();  }  }  private static void DisplayRepositoryItems()  {  using (ZentityContext context = new ZentityContext(connectionString))  {  Console.WriteLine("People");  foreach (var person in context.Resources.OfType<Person>())  Console.WriteLine("\t[{0}]", person.Id);  Console.WriteLine("Publications");  foreach (var publication in context.Resources.OfType<Publication>())  Console.WriteLine("\t[{0}]", publication.Id);  Console.WriteLine("PublicationHasAuthors");  foreach (var publication in context.Resources.OfType<Publication>().Include("Authors"))  foreach (var author in publication.Authors)  Console.WriteLine("\t[{0}]<-->[{1}]", publication.Id, author.Id);  Console.WriteLine("----------------------------");  }  }  }  } |

## Generating Pre-Compiled Views for Custom Modules

In Entity Framework applications, the first query usually takes more time than the subsequent ones. The primary reason for this is Entity Framework generates views for the entity sets defined in the metadata artifacts. This time can be reduced by compiling the view definitions in advance for custom modules.

* Re-create the Zentity database before this walkthrough. Detailed instructions for creating the database are provided in the walkthrough ‘Re-creating Zentity Database’.
* Using the steps provided in walkthrough ‘Modules with Non One-To-XXX Associations’, alter the database schema, generate assembly and Entity Framework artifacts.
* Generate the precompiled views source file by executing a command similar to the following on visual studio command prompt.

|  |
| --- |
| edmgen /mode:ViewGeneration /incsdl:"C:\Zentity.Samples.csdl" /incsdl:"C:\ExtendedCore.csdl" /inssdl:"C:\Consolidated.ssdl" /inmsl:"C:\Consolidated.msl" /outviews:"C:\Views.cs" |

* Create a new class library project, include Views.cs into it and compile it to generate C:\PreCompiledViews.dll. You may have to add a reference to System.Data.Entity.
* Create a console application with references to the extensions assembly. It is not required to add a reference to the pre-compiled views assembly. Replace the body of Program.cs with the following.

|  |
| --- |
| using Zentity.Core;  using Zentity.Samples;  using System;  using System.Linq;  using System.Reflection;  namespace ZentitySamples  {  class Program  {  static string connectionString = @"provider=System.Data.SqlClient;  metadata=C:\Zentity.Samples.csdl|C:\ExtendedCore.csdl|C:\Consolidated.msl|C:\Consolidated.ssdl;  provider connection string='Data Source=.;  Initial Catalog=Zentity;Integrated Security=True;MultipleActiveResultSets=True'";  static void Main(string[] args)  {  using (ZentityContext context = new ZentityContext(connectionString))  {  //context.MetadataWorkspace.LoadFromAssembly(Assembly.LoadFile(@"C:\PrecompiledViews.dll"));  context.MetadataWorkspace.LoadFromAssembly(typeof(Publication).Assembly);  DateTime start = DateTime.Now;  var v = context.Resources.Count();  Console.WriteLine(DateTime.Now - start);  }  }  }  } |

* Execute the program and note the time.
* Now, uncomment the line that loads metadata from pre-compiled views assembly, execute the program and note the time again.

NOTE: If you are encountering errors like the one below, ensure that the order of CSDLs are the same while generating Views.cs (parameters to EdmGen) and while opening the context (context connection string).

|  |
| --- |
| The Mapping and Metadata information for EntityContainer 'ZentityContext' no longer matches the information used to create the pre generated views. |