

# Overview

The **ExpressEntityMapping Framework** is an object-relational mapping (ORM) tool that simplifies data access and management by converting relational data into objects, properties, and collections for use in code. With this tool, you can query data and implement business logic using [entity models](#) and an object-oriented approach instead of handling the low-level data access and executing SQL statements manually.

The main features include:

- | Support for the Code-First, Database-First, and Model-First [approaches](#) to application development;
- | A [standalone application](#) that helps you visually design entity models and generate classes based on them;
- | Transparent and controllable object-relational mapping via [built-in attributes](#) or [registration functions](#);
- | Support for Firebird, Microsoft SQL Server, MySQL, Oracle, SQLite, and Microsoft Access [database engines](#);
- | Comprehensive database generation, updates, and integrity validation based on definitions of your [entity classes](#);
- | Advanced support for existing databases with the capability to [prohibit](#) changing their schemas;
- | Data access using FireDAC and dbGo ADO connection components – no additional connections are required in your existing applications;
- | A [dataset component](#) allowing you to bind entity objects to data-aware controls;
- | Easy-to-build object queries allowing you to retrieve, sort, and filter data using [LINQ expressions](#), object-based criteria syntax, or plain text, including calculated conditions and a wide set of criteria operators and [functions](#);
- | An advanced [parser](#) supporting a database-independent criteria string syntax;
- | Support for ordinal, real, enumerated, and [nullable](#) types, string fields/properties of unlimited [size](#), and BLOB data serialization;
- | Custom object identifiers (keys) allowing you to mark a field/property of any supported data type as a key by a [corresponding](#) attribute;
- | IEnumerable and IEnumerable<T> object collections and LINQ expression results.

Use the following links to find more information on the **ExpressEntityMapping Framework**:

- | [Supported Database Engines](#);
- | [Entity Model Design and Customization](#);
- | [Entity Model Designer](#);
- | [Built-in Attributes](#);
- | [Entity Class Registration](#);
- | [Entity Relationships](#);
- | [Connecting to a Data Store](#);
- | [Entity Object Management](#);
- | [Querying Data](#).

In addition, we recommend that you refer to the tutorials shipped with the **ExpressEntityMapping Framework** for code examples on how to connect to a database and perform CRUD operations on its data using entity objects.

# Supported Database Engines

The **ExpressEntityMapping Framework** provides access to data stores using database engines (also called database systems or DBMS) and lists supported DBMS versions along with compatible provider/driver versions for dbGo ADO data access components.

Database Engine	Database Engine	Supported Versions	Usage Notes
Firebird	Firebird 1.5, Firebird 2.5.7 (with SQL dialect 3)	Firebird/InterBase ODBC driver v2.0.1	The <b>ExpressEntityMapping Framework</b> supports the Firebird database. To create a Firebird database, use the <a href="#">CreateSchema</a> procedure of a <a href="#">session</a> to create a FireDAC connection object (TFDConn) and set the <code>Params.OpenMode</code> property to <code>omOpen</code> before connecting to the database using a TADOConnection or the Firebird Interactive SQL utility (ISQL) after installation or other DB administration tool like FBExplorer, to create the database. Boolean type fields/properties are mapped to <code>0</code> or <code>1</code> corresponding to <code>False</code> and <code>True</code> . The following applies to FireDAC connection objects: names in tables are case-sensitive – names in <a href="#">entity classes</a> must match them. Keys in tables must be in upper case.
Microsoft Access	Microsoft Access 95-2003	Microsoft Jet ODBC driver version 3 or later for 32-bit applications	FireDAC connection objects do not support Microsoft Access databases.
	Microsoft Access 95-2010	Microsoft Jet ODBC driver version 12 or later for 32-bit and 64-bit applications	
Microsoft SQL Server	Microsoft SQL Server 7.0, SQL Server 2000, SQL Server 2000 Desktop Engine (MSDE 2000), SQL Server 2005, SQL Server 2005 Express Edition, SQL Server 2008, SQL Azure Database, SQL Server 2008 R2, SQL Server 2008 R2 Express, SQL Server 2012, SQL Server 2012 Express (including LocalDB), SQL Server 2014, SQL Server 2014 Express (including LocalDB), SQL Server 2016, SQL Server 2016 Express (including LocalDB)	Microsoft OLE DB provider for SQL Server	
MySQL	MySQL Server 4.1, MySQL Server 5.0, MySQL Server 5.1, MySQL Server 5.7	MySQL Connector/ODBC v5.1 driver	The <b>ExpressEntityMapping Framework</b> supports the MySQL database. To create it, use DB administration tool corresponding SQL statements on the database. Boolean type fields/properties are mapped to <code>0</code> or <code>1</code> corresponding to <code>False</code> and <code>True</code> . The following applies to ADO connections: use the Unicode version of the ODBC driver to support Unicode strings.
Oracle	Oracle 9i, Oracle 10g, Oracle 11g, Oracle 12c	OraOLEDB.Oracle.1 provider for OLE DB	Table/column names in tables are case-sensitive. Names in classes, fields, and properties must match them. String type columns whose size exceeds 255 characters cannot be manipulated using entity objects. Boolean type fields/properties are mapped to <code>0</code> or <code>1</code> corresponding to <code>False</code> and <code>True</code> . The following applies to FireDAC connection objects: column names must be in the upper case.
SQLite	SQLite 3	SQLite3 ODBC driver v0.99 <b>Note:</b> There's a known driver issue with reading BLOB data.	String manipulation functions are not supported for data store using FireDAC connection objects. To overcome this limitation by implementing equivalent functions, use them as custom functions (TFDSQLiteCustomFunctions).

# Entity Model Design and Customization

The **ExpressEntityMapping Framework** supports the Model-First, Database-First, and Code-First approaches to application development, allowing you to create and extend [entity models](#) at any development stage of your project. You can design an entity model and its classes with any of these approaches and later fine-tune them in the IDE's code editor. The sections below provide details on framework features that are intended for each approach.

## Model-First Approach

This approach means that you use the [Entity Model Designer](#) application shipped with the **ExpressEntityMapping Framework** to do the following:

- 1 Visually create and modify an entity model, including entity properties, relationships, inheritance, and mapping information (using context menus, built-in dialogs, and the **Object Inspector** window);
- 1 Store an entity model as a file to persist its data between application runs (via the File menu commands);
- 1 Generate unit files containing entity class declarations and interfaces that enable support for [LINQ expressions](#) based on an entity model (via the Model | Generate Code Files menu command). These files include the essential information about your entity model as detailed in the [Code-First Approach](#) section below.

<>

## Database-First Approach

As with the previous approach, you can use the [Entity Model Designer](#) application to design an entity model. However, instead of creating it from scratch, you can create it by importing an existing database. To accomplish this, specify the connection settings for the source database and select the Model | Create Model from Database menu command.

<>

The current [Entity Model Designer](#) version replicates the entire database schema by importing tables, columns, and keys as entities, their properties, and relationships (associations) between them. Once created, the entity model is ready for further customization and use in the same manner as with the Model-First approach.

## Code-First Approach

This approach provides more granular control over entity models than the other two and means that you create an entity model by reflecting a target data store's schema in code as follows:

- 1 Define entities (normally, they correspond to database tables) as classes and table columns as fields and properties in these classes. Class inheritance is fully supported, so you can create base classes and inherit other classes from them;
- 1 Register classes in the entity model and specify how they and their members map to tables and other data store objects using [built-in attributes](#) or an entity manager. Refer to the [Entity Class Registration](#) topic to learn about these options;
- 1 [Specify](#) where each entity class descendant stores its data. Storing data in the base class's table (corresponds to the [TdxMapInheritanceType.ParentTable](#) option) requires that you introduce a [discriminator column](#) in the base class and provide this class and each entity descendant class with a discriminator column [value](#) that uniquely identifies them in the inheritance tree;
- 1 Identify one or more [columns](#) as a [primary key](#) for each entity class that maps to a database table. Entity class descendants inherit their primary keys from their ancestors – so you need to identify it only once per inheritance tree;
- 1 Establish relationships via [associations](#) and implement code that initializes fields and properties they use.

After that, you can add the entity model's classes to your project to start using them and their members in it.

Consider a simple entity model including a Person entity with two attributes: name and age. The following code example shows how to declare a class for this entity using built-in attributes.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
```

```

type
  [Entity]
  [Table('Person')]
  TPerson = class
  strict private
    FId: Integer;
    FAge: Integer;
    FName: string;
  public
    [Column, Key, Generator(TdxGeneratorType.Identity)]
    property Id: Integer read FId write FId;
    [Column]
    property Age: Integer read FAge write FAge;
    [Column]
    property Name: string read FName write FName;
  end;

```

The class contains three properties, one of which is an auto-incremented primary key (whose values are [generated](#) automatically), and the other two store personal information. Call the [CreateSchema](#) procedure of a session component [connected](#) to an SQLite database and pass the class type as a parameter to create the database whose schema matches this class.

```

[Delphi]
type
  TForm1 = class(TForm)
  // ...
    dxEMFSession: TdxEMFSession;
  // ...
  end;
var
  Form1: TForm1;
implementation
  // ...
begin
  // Creating a database and updating its schema to match the TPerson class declaration
  dxEMFSession.DataProvider.Options.AutoCreate := TdxAutoCreateOption.DatabaseAndSchema;
  dxEMFSession.CreateSchema(TPerson);
end;

```

This creates the following database schema:

Name	Type	Schema
Tables (2)		
Person		
Id	INTEGER	'Id' INTEGER PRIMARY KEY AUTOINCREMENT
Age	int	'Age' int NOT NULL
Name	nvarchar ( 100 )	'Name' nvarchar ( 100 ) NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

# Entity Model Designer

The **ExpressEntityMapping Framework** ships with the Entity Model Designer – a standalone application that helps you visually [design and customize](#) entity models and generate classes based on them. The application provides context menus, built-in dialogs, and dockable windows for in-depth customization of entities, their properties, relationships, inheritance, and various mapping options.

<>

With the Entity Model Designer, you can:

- | Create an entity model from scratch by selecting File | New in the main menu;
- | Create an entity model using an existing database by selecting Model | Create Model from Database in the main menu. This requires that you specify connection settings for this database in the Connection Editor dialog, which you can invoke by selecting Edit Connection... in the Database Browser window's context menu;

<>

- | Manage entities using the Entity Editor dialog and context menus provided by the Model Browser and Diagram windows (via the Add Entity... and Delete menu commands and clipboard operations);

<>

- | Manage entity properties using the Property Editor dialog and context menus provided by the Model Browser and Diagram windows for any selected entity (via the Add Property... and Delete menu commands and clipboard operations);

<>

- | Specify how entities and their properties map to data store objects (tables, column names and data types, keys, indexes, etc.) using the Entity Editor and Property Editor dialogs. You can invoke the corresponding dialog by double-clicking an entity or entity property in the Model Browser or Diagram window. Alternatively, you can select an entity or entity property in one of these windows, invoke the context menu, and select Edit...;
- | Add or customize entity [inheritance](#) using the Inheritance Editor dialog. To add an entity inheritance, select Add Inheritance... in the context menus provided by the Model Browser and Diagram windows. To customize an entity inheritance, either double-click it or select it in one of these windows, invoke the context menu, and select Edit...;

<>

- | Add or customize entity [associations](#) (relationships) using the Association Editor dialog. To add an association, select Add Association... in the context menus provided by the Model Browser and Diagram windows. To customize an entity association, either double-click it or select it in one of these windows, invoke the context menu, and select Edit...;

<>

- | Rearrange diagram objects using drag and drop, collapse/expand, horizontally size, or remove them from the diagram (if necessary). In addition, you can scroll the mouse wheel while holding down the Ctrl key to zoom in/out of the diagram. To re-create it and switch to the default layout, select Model | Create Diagram from Model in the main menu;
- | Save your entity model, its diagram, and database connection settings to a DMF file and load it later using the File menu commands;
- | Generate unit files containing entity class declarations and interfaces that enable support for [LINQ expressions](#) based on your entity model (via the Model | Generate Code Files menu command). These files include the essential information about the entity model, complete with attributes and initialization code for fields and properties used by associations, as detailed in [this section](#). You can add the generated unit files to your project and start using their classes and interfaces to [manage](#) and [query](#) data stores via entity objects. Use the entity model's CodeGenerationEntityType property to select the method used to [register](#) entity classes in these unit files.

# Built-in Attributes

The **ExpressEntityMapping Framework** uses built-in attributes and runtime type information (RTTI) to map entities defined in an [entity model](#) to data store objects. These attributes allow you to provide mapping details, establish relationships between entity classes, define entity inheritance, etc. Add the [dxEMF.Attributes](#) unit to the 'uses' clause to enable these attributes in your project.

The following table lists all built-in attributes by mapping category.

Mapping Category	Attribute	Description
<b>Entity</b> (applies to classes)	<a href="#">Entity</a>	
<b>Table and Column (Simplified Mapping)</b> (applies to classes and fields/properties)	<a href="#">Automapping</a>	
	<a href="#">NonPersistent</a>	
<b>Table</b> (applies to classes)	<a href="#">Table</a>	
	<a href="#">SchemaName</a>	
<b>Inheritance</b> (applies to classes)	<a href="#">Inheritance</a>	
	<a href="#">DiscriminatorColumn</a>	
	<a href="#">Discriminator</a>	
<b>Primary Key Column</b> (applies to fields/properties)	<a href="#">Key</a>	
	<a href="#">Generator</a>	
<b>Column</b> (applies to fields/properties)	<a href="#">Column</a>	
	<a href="#">DBType</a>	
	<a href="#">Nullable</a>	
	<a href="#">Default</a>	Reserved for future use.
	<a href="#">Size</a>	
	<a href="#">Blob</a>	
	<a href="#">ReadOnly</a>	
<b>Column Index</b> (applies to classes)	<a href="#">Indexes</a>	
<b>Column Index</b> (applies to fields/properties)	<a href="#">Indexed</a>	
	<a href="#">Unique</a>	
<b>Relationship</b> (applies to fields/properties)	<a href="#">Association</a>	
	<a href="#">Aggregated</a>	
	<a href="#">NoForeignKey</a>	

Refer to the descriptions of the attribute classes listed above for details and code examples on how to use them.

# Entity Class Registration

The **ExpressEntityMapping Framework** provides the following options to register [entity model](#) classes for use in code and specify mapping information for them and their members:

- 1 [Built-in attributes](#) (Delphi only). The attributes allow you to provide mapping details, establish relationships between entity classes, define entity inheritance, etc. in class declarations. A class marked with the [EntityAttribute](#) is automatically registered in the entity model at runtime. To enable the built-in attributes in your project, add the [dxEMF.Attributes](#) unit to the 'uses' clause. Note that this option applies to the Delphi code only, because Delphi attributes are not supported in C++Builder.
- 1 **The entity manager** (Delphi and C++Builder). This option means that you declare each entity class without attributes and provide its mapping information in the initialization section using the entity manager's RegisterEntity function and a sequence of calls to functions made from another call's result. The functions include: RegisterField, RegisterProperty, and equivalents of the built-in attributes (they bear the same names and accept the same parameters). Call the global EntityManager function declared in the dxEMF.Metadata unit to access the entity manager and its API.

Refer to [this section](#) for a code example on how to declare a simple TPerson entity class mapped to the Person table using the [Entity](#), [Table](#), [Column](#), [Key](#), and [Generator](#) built-in attributes. The following code example shows how to register this entity class using the entity manager.

```
[Delphi]
uses
  ..., dxEMF.Core, dxEMF.Types, dxEMF.Metadata;
type
  TPerson = class
  strict private
    FId: Integer;
    FAge: Integer;
    FName: string;
  public
    property Id: Integer read FId;
    property Age: Integer read FAge write FAge;
    property Name: string read FName write FName;
  end;
implementation
initialization
  EntityManager.RegisterEntity(TPerson).Table('Person').
    RegisterField('FId').Generator(TdxGeneratorType.Identity).Key.
    RegisterProperty('Age').
    RegisterProperty('Name');
```

In the [Entity Model Designer](#), you can select the method used to register entity classes in generated unit files using the entity model's CodeGenerationEntityRegistrationType property.

# Entity Relationships

In relational databases, relationships (also called associations) created between tables are defined using foreign keys. A foreign key enforces a link between the data in two tables participating in a relationship. The **ExpressEntityMapping Framework** supports all three available types of table relationships (one-to-one, one-to-many, and many-to-many) and allows you to map them to **entity relationships**. To learn about a particular relationship type and how to declare it in [entity models](#), jump to the corresponding section by clicking a link below.

- | [One-to-Many Relationships](#)
- | [One-to-One Relationships](#).
- | [Many-to-Many Relationships](#).

## One-to-Many Relationships

A one-to-many relationship is the most common type of relationship. In this relationship, an entity object of type A can have many associated objects of type B, but an entity object of type B can have only one associated object of type A. For example, team members can be associated with a specific team by creating a relationship (association) between a Members collection in the Team entity object (the primary key) and a field that references a Team entity object in the Member entity object (the foreign key). As a result, each team can have multiple members.

The following code shows how to implement this example. The TTeam.FMembers and TMember.FTeam fields marked with the [AssociationAttribute](#) reference the "many" and "one" ends of the association, respectively. The FMembers field is a collection of TMember entity objects associated with a particular TTeam. This collection implements the [IdxEMFCollection](#) or [IdxEMFCollection<T>](#) interface. Note that the [AggregatedAttribute](#) applied to this field enables the cascade option for TTeam entity objects. With the option, update and delete operations applied to a TTeam entity object reflect in its TMember objects.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types, dxEMF.Core.Collections;
type
    TTeam = class;
    TMember = class;
    [Entity]
    [Automapping]
    TTeam = class
    strict private
        [Generator(TdxGeneratorType.Identity)]
        FId: Integer;
        FName: string;
        [Association, Aggregated]
        FMembers: IdxEMFCollection<TMember>;
    public
        constructor Create;
        property Id: Integer read FId;
        property Name: string read FName write FName;
        property Members: IdxEMFCollection<TMember> read FMembers;
    end;
    [Entity]
    [Automapping]
    TMember = class
    strict private
        FId: Integer;
        FName: string;
        [Association, Nullable]
```

```

    FTeam: TTeam;
    procedure SetTeam(AValue: TTeam);
public
    [Generator(TdxGeneratorType.Identity)]
    property Id: Integer read FId;
    property Name: string read FName write FName;
    property Team: TTeam read FTeam write SetTeam;
end;
// ...
implementation
// ...
constructor TTeam.Create;
begin
    inherited;
    FMembers := TdxEMFCollections.Create<TMember>(Self, 'FMembers');
end;
procedure TMember.SetTeam(AValue: TTeam);
begin
    FTeam := AValue;
    if FTeam <> nil then
        FTeam.Members.Add(Self);
end;

```

## One-to-One Relationships

In a one-to-one relationship, an entity object of type A has one associated object of type B. Their primary keys act also as foreign keys and no separate foreign key column is required. Each entity object stores a reference (also called entity reference) to the associated object.

The following code example shows how to establish a one-to-one relationship between a User entity and its Profile (a UserProfile entity). The TUser and TUserProfile entity classes refer to each other via the TUser.Profile and TUserProfile.User properties, which store entity references. You don't need to mark them with the [AssociationAttribute](#), because the **ExpressEntityMapping Framework** automatically establishes a relationship based on the entity references.

```

[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
type
    TUserProfile = class;
    TUser = class;
    [Entity]
    [Automapping]
    TUserProfile = class
    strict private
        [Generator(TdxGeneratorType.Identity)]
        FId: Integer;
        FName: string;
        FUser: TUser;
    public
        constructor Create;
        property Id: Integer read FId;
        property Name: string read FName write FName;
        property User: TUser read FUser write FUser;
    end;

```

```
[Entity]
[Automapping]
TUser = class
strict private
  FId: Integer;
  FName: string;
  FProfile: TUserProfile;
public
  [Generator(TdxGeneratorType.Identity)]
  property Id: Integer read FId;
  property Name: string read FName write FName;
  property Profile: TUserProfile read FProfile write FProfile;
end;
```

## Many-to-Many Relationships

In a many-to many-to-many relationship, many entity objects of type A can have one or more associated objects of type B, and vice versa. This type of relationship involves defining a third entity object (mapped to a junction table), whose primary key is composed of the foreign keys from the related objects. The current version of the **ExpressEntityMapping Framework** allows you to create this relationship using two [one-to-many relationships](#) and a junction entity class representing the "many" end of them.

# Connecting to a Data Store

The **ExpressEntityMapping Framework** interacts with [supported](#) data stores using [session components](#). Each component requires either a FireDAC or ADO connection object configured to access a specific data store, and a corresponding data provider component used as a session/connection mediator. Data provider components shipped with the **ExpressEntityMapping Framework** include:

- 1 A FireDAC-based data provider (the [TdxEMFFireDACDataProvider](#) component declared in the [dxEMF.DataProvider.FireDAC](#) unit). This data provider is designed to link to a TADConnection or TFDCConnection object.
- 1 An ADO-based data provider (the [TdxEMFADODDataProvider](#) component declared in the [dxEMF.DataProvider.ADO](#) unit). This data provider is designed to link to a TADODConnection object.

Follow the steps below to connect a session component ([TdxEMFSession](#)) to a data store using a FireDAC or ADO connection object:

- 1 Select an existing connection object (TADConnection, TFDCConnection, or TADODConnection) or add a new connection object and specify its connection settings for the data store;
- 1 Depending on the connection object, add either the [TdxEMFFireDACDataProvider](#) or [TdxEMFADODDataProvider](#) component;
- 1 Link this data provider component to the connection object using the component's Connection property ([TdxEMFFireDACDataProvider.Connection](#) or [TdxEMFADODDataProvider.Connection](#));
- 1 Link the session and data provider components using the session component's [DataProvider](#) property;
- 1 Select a target database engine using the data provider component's [Options.DBEngine](#) property.

After that, the session component is ready to [manage](#) and [query](#) data store objects at runtime. You don't need to open the connection object – the session component does this automatically.

# Entity Object Management

An entity object (an instance of an [entity class](#) defined in an [entity model](#)) represents a fact about its entity. In terms of relational databases, an entity object corresponds to a record in one or more (joined) tables and contains values for one or more columns. Managing entity objects and storing/updating their data via a [session component](#) that is [connected](#) to a database allows you to perform corresponding operations with its table records.

## Entity Object Lifetime

You can create, modify, and destroy entity objects in the same manner as you do with normal objects. However, no changes made to entity objects are reflected in a data store until you connected it to a session component and associated this component with the entity objects. To associate them, do one of the following via a session component:

- 1 Perform any CRUD operation (described below);
- 1 Attach an entity object using the [Attach](#) procedure.

**Important:** Once associated with an entity object, the session component is responsible for destroying it and deallocating memory. You don't need to accomplish this to let the session component properly finalize all object-related operations.

## CRUD Operations

The acronym stands for Create, Read, Update, and Delete operations, which you can perform with entity objects via a session component and reflect them in a data store connected to this component. The operations and examples on how to perform them are listed below.

### Creating Records

You can create an entity object (an instance of an entity class) by doing one of the following:

- 1 Call the entity class's constructor;
- 1 Call a session component's [CreateObject](#) function and pass the entity class's type as a parameter.

Then, call a session component's [Save](#) method and pass the created entity object as a parameter to create a corresponding record and initialize it with the object's property/field values.

The following code creates a record in a table to which the [TPerson](#) entity class described in the [Entity Model Design and Customization](#) topic maps its data.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
var
    APerson: TPerson;
    // ...
begin
    // ...
    // Creating a database and updating its schema to match
    // the TPerson entity class declaration (this creates the Person table)
    dxEMFSession.DataProvider.Options.AutoCreate := TdxAutoCreateOption.DatabaseAndSchema;
    dxEMFSession.CreateSchema(TPerson);
    // Creating an entity object and specifying its properties
    APerson := TPerson.Create;
    APerson.Age := 25;
    APerson.Name := 'John Doe';
    // Saving the entity object, creating a corresponding record
```

```
// in the Person table and populating this record with the object's data
dxEMFSession.Save(APerson);
end;
```

## Reading Records

This operation retrieves data store records as entity objects using queries built with LINQ expressions, object-based criteria, or criteria strings. Query results are returned as [object collections](#) and [LINQ expression](#) results, which you can enumerate and access elements (entity objects) and their data. Refer to the [Querying Data](#) topic to learn more.

## Updating Records

This operation includes:

- | Reading data from data store records to entity objects (see the section above);
- | Modifying properties and fields in these objects;
- | Apply the changes to the data store using a session component's [Save](#) method.

The following code example modifies the Age and Name property values of the TPerson entity object created in the previous section and updates the record.

```
[Delphi]
APerson.Age := 28;
APerson.Name := 'John Smith';
// Saving the entity object and updating the corresponding record
// with new values
dxEMFSession.Save(APerson);
```

Note that each call to the [Save](#) method accesses a data store and updates the corresponding record (and [aggregated](#) records), which may take time and significantly hinder the performance when updating multiple records. For batch updates, we recommend that you use the Track Changes mode (see below) to accumulate all the changes and apply them in a batch update, similar to a database transaction.

## Deleting Records

You can delete an entity object using any of the following methods:

- | Pass this entity object to a session component's [Delete](#) method;
- | [Remove](#) this entity object from an object collection whose [DeleteObjectOnRemove](#) property is set to **True**.

As with the [Save](#) method, we recommend that you delete multiple entity objects in Track Changes mode to improve performance.

## Track Changes Mode

The Track Changes mode works in a similar way to a database transaction, which groups atomic operations and either applies or discards all of them, based on a condition. In this mode, a session component tracks all entity object changes to a data store and accumulates them in memory until either of the following takes place:

- | The session component's [FlushChanges](#) method is called to apply the changes to the data store;
- | The session component's [DropChanges](#) method is called to discard pending changes to the data store. This, however, retains changes in entity objects.

To enable the mode and start tracking object changes, call the session component's [BeginTrackingChanges](#) method.

The following code example shows how to handle CRUD operations using the Track Changes mode.

```
[Delphi]
dxEMFSession.BeginTrackingChanges;
```

```
try
  <Your CRUD operations here>
  // ...
  // Applying the changes
  dxEMFSession.FlushChanges;
except
  // Discarding the changes
  dxEMFSession.DropChanges;
  <Handle an error here (log it, display a message, etc.)>
end;
```

# Querying Data

Querying (or reading) data is one of the [CRUD operations](#) available in all data stores. By querying data with the **ExpressEntityMapping Framework**, you retrieve data store records as entity objects only if they match specific criteria. You can build queries using LINQ expressions, object-based criteria, and strings that follow a database-independent criteria syntax. The sections below provide details of these options.

## Object-Based Criteria

This method allows you to retrieve entity objects that match specific criteria by calling a session component's [GetObjects](#) or [Find](#) function and passing the criteria expression and entity class type as parameters. You can build criteria expressions using operator and operand classes declared in the [dxEMF.DB.Criteria](#) unit. Each operator accepts one or more operands, which can be constants, values, entity properties, or other operators. With this flexible expression model, you can build complex criteria by combining simple criteria expressions.

The following table lists all available operator and operand classes by category.

Category	Class	Description
Operator	<a href="#">TdxAggregateOperand</a>	An aggregate operator that calculates aggregate expressions (SUM, MIN, MAX, etc.)
	<a href="#">TdxBetweenOperator</a>	An operator that checks if an operand value falls within a specified range.
	<a href="#">TdxBinaryOperator</a>	An operator that evaluates a <a href="#">TdxBinaryOperatorType</a> operation with two operands.
	<a href="#">TdxContainsOperator</a>	An operator that checks if a collection contains at least one object matching specific criteria.
	<a href="#">TdxFunctionOperator</a>	An operator based on a <a href="#">TdxFunctionOperatorType</a> function.
	<a href="#">TdxGroupOperator</a>	An operator that evaluates a logical AND or OR operation with two or more operands.
	<a href="#">TdxInOperator</a>	An operator that checks if an operand value matches any element in a specified list.
	<a href="#">TdxJoinOperand</a>	An operator that joins entity objects on a specified condition, and calculates aggregate functions against matching objects.
	<a href="#">TdxUnaryOperator</a>	An operator that evaluates an unary operation (with only one operand).
Operand	<a href="#">TdxConstantValue</a>	A constant value operand.
	<a href="#">TdxOperandProperty</a>	An object property operand.
	<a href="#">TdxOperandValue</a>	A value operand.

The code example below shows how to retrieve all persons under the age of 30 using a [TdxBinaryOperator](#) object and a generic [GetObjects](#) function.

```
[Delphi]
uses
  ..., dxEMF.Core, dxEMF.Types, dxEMF.DB.Criteria;
implementation
// ...
var
  ACollection: IdxEMFCollection<TPerson>;
  ACriteriaOperator: IdxCriteriaOperator;
//
```

```

ACriteriaOperator := TdxBinaryOperator.Create('Age', 30, TdxBinaryOperatorType.Less);
ACollection := dxEMFSession.GetObjects<TPerson>(ACriteriaOperator);
end;

```

Query results are returned as [object collections](#), which you can enumerate and access elements (entity objects) and their data. The following code shows how to enumerate the resulting object collection and obtain names of persons contained in it as a semicolon-delimited list.

```

[Delphi]
var
  APerson: TPerson;
  APersonNames: string;
//...
begin
//...
  for APerson in ACollection do
    APersonNames := APersonNames + APerson.Name + ';';
end;

```

### Database-Independent Criteria Strings

Unlike the previous method, you define a criteria expression as a string using a database-independent language and parse this string into criteria operands and operators by calling the [TdxCriteriaOperator.Parse](#) or [TdxCriteriaOperator.ParseList](#) class function. Then, you can pass the resulting criteria expression as a parameter to a session component's [GetObjects](#) or [Find](#) function, as described above.

The following code example calls the [TdxCriteriaOperator.Parse](#) function to create a criteria expression that is identical to the one described in the previous topic section.

```

[Delphi]
ACriteriaOperator := TdxCriteriaOperator.Parse('Age < 30');

```

You can also pass operands as parameters and use question marks in a string as operand placeholders. The following criteria expressions are identical.

```

[Delphi]
ACriteriaOperator1 := TdxCriteriaOperator.Parse('Age < 30 and YearOf(HireDate) >= 2010');
ACriteriaOperator2 := TdxCriteriaOperator.Parse('Age < ? and YearOf(HireDate) >= ?', [30, 2010]);

```

### LINQ Expressions

Language Integrated Query (LINQ) introduces query capabilities to programming languages by adding a set of methods that allow writing SQL-like query expressions based on types and their members. In addition to improving code readability, LINQ expressions help validate column names used in queries at design time. Refer to the "Language Integrated Query" online article at [https://en.wikipedia.org/wiki/Language\\_Integrated\\_Query](https://en.wikipedia.org/wiki/Language_Integrated_Query) to learn more.

Our LINQ implementation (currently supported in Delphi only) extends query capabilities with support for entities. It is based on overloaded operators and interfaces whose function declarations replicate an [entity model](#), its classes, and their properties. You do not need to implement these interfaces – they are building blocks for LINQ expressions and contracts for parameters used in expression clauses. In addition, storing queries and query expressions in variables declared as interface types frees you from manually destroying their reference objects.

In the **ExpressEntityMapping Framework**, a LINQ query is a sequence of function calls made from another call's result. Functions are equivalent to LINQ clauses and bear the same names.

Each LINQ query includes:

- 1 A Linq function call, which starts a query expression;
- 1 A From function call specifying a [Data Context](#) and a queried entity. A Data Context is the source of all queryable entities in your entity model. A Data Context references a [session component](#) to load data for entity objects returned as the query result;
- 1 Optional Where and/or OrderBy function calls specifying the filter criteria and how entity objects are sorted in the query result;

The following code shows an example of a LINQ query expression based on the [TPerson](#) entity class described in the [Entity Model Design and Customization](#) topic.

```
[Delphi]
AQuery := Linq.
  From<TPerson>(APersonExpression).
  Where(APersonExpression.Age < 30).
  OrderBy(APersonExpression.Name).
  Select.
  Take(5);
```

To enable the use of LINQ expressions, add the `dxEMF.Linq` and `dxEMF.Linq.Expressions` units shipped with the **ExpressEntityMapping Framework** to your project. Then, add declarations of interfaces that correspond to your entity model and register them for entity-based LINQ expressions by adding `TdxLinqExpressionFactory.Register` method calls to the 'initialization' section. We recommend that you use the [Entity Model Designer](#) to generate a unit file that contains all the interfaces and initialization code sufficient for creating LINQ expressions based on your project's entity model.

The interfaces and the initialization code required for the [TPerson](#) entity class can be declared as follows:

```
[Delphi]
uses
  ..., dxEMF.Linq, dxEMF.Linq.Expressions;
//...
type
  IPersonExpression = interface(IdxEntityInfo)
  ['{DF35C49D-1E2C-4207-BF54-AEE64F713120}']
    function Id: TdxLinqExpression;
    function Age: TdxLinqExpression;
    function Name: TdxLinqExpression;
  end;
  IDataModelContext = interface(IdxDataContext)
  ['{3D1FE13A-33C7-411A-8EE2-BC9CF55CD619}']
    function Person: IPersonExpression;
  end;
//...
implementation
//...
initialization
  TdxLinqExpressionFactory.Register<TPerson, IPersonExpression>;
```

The following code example creates a LINQ query that is identical to the criteria described in the previous topic sections.

```
[Delphi]
uses
  ..., dxEMF.Core, dxEMF.Types, dxEMF.DB.Criteria, dxEMF.Linq, dxEMF.Linq.Expressions;
//...
var
  ADataContext: IDataModelContext;
  APersonExpression: IPersonExpression;
  AQuery: IdxQueryable<TPerson>;
//...
begin
//...
  // Obtaining a LINQ Data Context (it's the source of all queryable entities in your entity model)
```

```
APersonExpression := ADataContext.Person;
```

The following line is equivalent to the two above.

```
APersonExpression := dxEMFSession.GetEntityInfo<IPersonExpression>;
```

A LINQ query can be created using an expression tree, as shown below.

```
AQuery := Linq.  
    From<TPerson>(APersonExpression).  
    Where(APersonExpression.Age < 30).  
    Select;
```

Note that any LINQ query inherits either from `IEnumerable` or `IEnumerable<T>`. A LINQ query retrieves entity objects only when the enumeration executes it. In addition, since variables are declared as interface types, you don't need to manually destroy them.

The following code shows how to enumerate the query and obtain names of persons retrieved by the query's enumeration as a semicolon delimited list.

```
[Delphi]  
    for APerson in AQuery do  
        APersonNames := APersonNames + APerson.Name + ';';  
end;
```

# dxEMF.Core Unit

## Classes

[TdxEMFCollectionOptions](#)

[TdxEMFCollections](#)

[TdxEMFCustomCollection](#)

[TdxEMFCustomDataProvider](#)

[TdxEMFCustomSession](#)

[TdxEMFDataProviderOptions](#)

[TdxEMFSession](#)

[TdxEMFSessionOptions](#)



# TdxEMFADODataProvider Object

[Hierarchy](#) [Properties](#)

---

An ADO-based data provider for a [session component](#).

## Unit

[dxEMF.DataProvider.ADO](#)

## Syntax

```
TdxEMFADODataProvider = class(TdxEMFCustomDataProvider)
```

## Description

This data provider is designed to link a session component to an ADO connection object (TADOConnection) specified via the [Connection](#) property. Refer to the [Connecting to a Data Store](#) topic to learn how to connect a session component to a data store using data providers.

## Related Information

- | [TdxEMFFireDACDataProvider Object](#)



# TdxEMFDataSet Object

[Hierarchy](#) [Properties](#) [Methods](#) [Events](#)

---

A dataset used to obtain data from a data store via a [session component](#).

## Unit

[dxEMF.DataSet](#)

## Syntax

```
TdxEMFDataSet = class(TdxEMFCustomDataSet)
```



# TdxEMFFireDACDataProvider Object

[Hierarchy](#) [Properties](#)

---

A FireDAC-based data provider for a [session component](#).

## Unit

[dxEMF.DataProvider.FireDAC](#)

## Syntax

```
TdxEMFFireDACDataProvider = class(TdxEMFCustomDataProvider)
```

## Description

This data provider is designed to link a session component to a FireDAC connection object (TADConnection or TFDCConnection) specified via the [Connection](#) property. Refer to the [Connecting to a Data Store](#) topic to learn how to connect a session component to a data store using data providers.

## Related Information

- | [TdxEMFADODataProvider Object](#)



# TdxEMFSession Object

[Hierarchy](#) [Properties](#) [Methods](#)

A component that allows you to [connect](#) to any [supported](#) data store, and [query/manage](#) its records using [entity](#) objects.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFSession = class(TdxEMFCustomSession)
```

## Description

This component is the primary method for the **ExpressEntityMapping Framework** to interact with data stores. The component does the following:

- | Obtains mapping information from an [entity model](#) and validates it against a data store's schema;
- | Keeps track of objects created using the entity model's classes and manages their lifetime;
- | Provides methods for object manipulation;
- | Converts object manipulation actions to SQL statements and executes them.

You can use any number of session components simultaneously, each having their own entity model and object cache.

The **TdxEMFSession** component introduces the [CreateSchema](#) procedure and makes the inherited [DataProvider](#) and [Options](#) properties published.

# Hierarchy

TCustomAttribute

|

[AggregatedAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[AssociationAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[AutomappingAttribute](#)

# Hierarchy

---

TCustomAttribute

|

[BlobAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[ColumnAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[DBTypeAttribute](#)

# Hierarchy

{\$IFDEF DELPHIXE3}

Classes.DefaultAttribute

{\$ELSE}

TCustomAttributes

{\$ENDIF}

|

[DefaultAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[DiscriminatorAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[DiscriminatorColumnAttribute](#)

---

# Hierarchy

Exception

|

[EdxODBCError](#)

# Hierarchy

---

TCustomAttribute

|

[EntityAttribute](#)

# Hierarchy

---

TCustomAttribute

|

[GeneratorAttribute](#)

# AggregatedAttribute Object

## [Hierarchy](#)

---

Indicates if an [entity class](#)'s collection field/property referencing an [association](#)'s "many" end aggregates its referenced entity objects.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
AggregatedAttribute = class(TCustomAttribute)
```

## Description

Mark an entity class's collection field/property with the **AggregatedAttribute** to set the cascade option for this class. With the option, the entity class's instance aggregates all entity objects that refer to it. The aggregated objects cannot exist without the referenced entity object, thus updating/deleting this object also updates/deletes the aggregated objects. If the cascade option is not set, update/delete operations with the referenced object do the following:

- | An update operation doesn't affect the referenced objects;
- | A delete operation removes the object and assigns NULL to the corresponding field/property in the referenced objects.

The **AggregatedAttribute** applies only to fields/properties marked with the [AssociationAttribute](#). Refer to the [Entity Relationships](#) topic for examples of how to enable the cascade option in various entity relationships.

# AssociationAttribute Object

## [Hierarchy](#)

---

Identifies the end of an association that establishes a [one-to-many](#) entity relationship.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
AssociationAttribute = class(TCustomAttribute)
```

## Description

Apply this attribute to both fields/properties at the ends of the association to establish a one-to-many relationship between their entity classes. If the field/property is a collection that implements the [IdxEMFCollection](#) or [IdxEMFCollection<T>](#) interface, we recommend that you apply the **AssociationAttribute** only to the field rather than the property associated with it.

You can optionally pass an association name as a parameter to both the **AssociationAttributes** at the ends to identify them. This name must be unique among all the associations defined in your entity model. If no association name is provided, the **ExpressEntityMapping Framework** automatically resolves association ends by the types being linked – so you don't have to provide association names in most cases. If a type is used in more than one association and cannot be singled out for an end, an association name should be provided.

Combine the **AssociationAttribute** with the [AggregatedAttribute](#) to make an aggregate relationship. Refer to the [Entity Relationships](#) topic for examples of how to establish relationships using these attributes.

# AutomappingAttribute Object

## [Hierarchy](#)

Enables simplified table and column mapping for an [entity class](#).

## Unit

[dxEMF.Attributes](#)

## Syntax

```
AutomappingAttribute = class(TCustomAttribute)
```

## Description

Simplified table and column mapping causes the **ExpressEntityMapping Framework** to infer mapping information from the class's name and fields in the following manner:

- The class name is considered the table name. The 'T' prefix is automatically removed;
- A primary key column called Id is created automatically if the class includes an Integer or Int64 field called 'FId'. The column is an [auto-incremented](#) identity if this field is associated with a read-only Id property. Once the Id primary key column is created, all other [KeyAttributes](#) in this class are ignored;
- A column is created for each ordinal, [nullable](#), [collection](#), or [generic collection](#) field, or a field that references an entity class. Fields marked with the [NonPersistentAttribute](#) are ignored. The field's name is considered the column name. The 'F' prefix is automatically removed. The column type and size are determined based on the field's data type and are specific to the [database](#) within which a [session component](#) creates the entity model's [schema](#). The string column size limit is **100** characters. Use the `TdxSQLConnectionProvider.DefaultStringSize` class variable to adjust it.

You can use the [TableAttribute](#), [ColumnAttribute](#), [DBTypeAttribute](#), [SizeAttribute](#), and other attributes to override the default mapping. Attributes applied to properties are ignored.

The following code example uses the **AutomappingAttribute** to create the schema that is identical to the schema created by the class declaration described in [this section](#). Note that the [NonPersistentAttribute](#) excludes the FQuickNote field (and a corresponding column) from the schema.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [Automapping]
    TPerson = class
    strict private
        FId: Integer;
        FAge: Integer;
        FName: string;
        [NonPersistent]
        FQuickNote: string;
    public
        property Id: Integer read FId;
    end;
```

# BlobAttribute Object

## [Hierarchy](#)

Indicates if a column to which a string or BLOB data field/property of an [entity class](#) is mapped stores Memo or BLOB data.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
BlobAttribute = class(TCustomAttribute)
```

## Description

With this attribute, the mapped column's type and size are sufficient for storing Memo or BLOB data. They are specific to the [database](#) within which a [session component](#) creates the entity model's [schema](#).

The following code shows how to configure an entity class to store [TdxSmartImage](#) object with the Photo property using the **BlobAttribute**:

```
[Delphi]
uses
  ..., dxEMF.Core, dxEMF.Attributes, dxGDIPlusClasses;
type
  [Entity]
  [Automapping]
  TPerson = class
  strict private
    FId: Integer;
    [Blob]
    FPhoto: TdxSmartImage;
  public
    constructor Create;
    destructor Destroy; override;
    property Id: Integer read FId;
    property Photo: TdxSmartImage read FPhoto write FPhoto;
  end;
// ...
implementation
constructor TPerson.Create;
begin
  inherited Create;
  FPhoto := TdxSmartImage.Create;
end;
destructor TPerson.Destroy;
begin
  FreeAndNil(FPhoto);
  inherited Destroy;
end;
```

## Related Information

- | [SizeAttribute Object](#)

# ColumnAttribute Object

## [Hierarchy](#)

Specifies a table column to which an [entity class](#) maps its field/property.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
ColumnAttribute = class(TCustomAttribute)
```

## Description

Mark a field or a property with this attribute and pass the column's name as a parameter. If no name is specified, the name of the field or property is used instead. The field's 'F' prefix is automatically removed.

The following code example shows how to map the FId and FName fields to the Id and FullName columns. Note that applying the **ColumnAttribute** to the Name property is equivalent to applying it to the FName field.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    TPerson = class
    strict private
        [Column, Key]
        FId: Integer;
        FName: string;
        // ...
    public
        property Id: Integer read FId;
        [Column('FullName')]
        property Name: string read FName write FName;
        // ...
    end;
```

**Note:** A field or property that makes up a [primary key](#) requires adding the **ColumnAttribute**.

## Related Information

- | [AutomappingAttribute Object](#)
- | [DBTypeAttribute Object](#)

# DbTypeAttribute Object

## [Hierarchy](#)

---

Specifies the data store type of a table column to which an [entity class](#)'s field/property maps its data.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
DbTypeAttribute = class(TCustomAttributes)
```

## Description

The **ExpressEntityMapping Framework** automatically chooses a [database](#)-specific column type based on the mapped field's/property's data type. You can use the **DbTypeAttribute** to override the default behavior.

The following code example shows how to customize the default column type for a TGUID field.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    TPerson = class
    strict private
        [Column, Key, DbType('char(40)')]
        FId: TGUID;
        // ...
    end;
```

## Related Information

- | [AutomappingAttribute Object](#)
- | [ColumnAttribute Object](#)

# DefaultAttribute Object

[Hierarchy](#)

---

Reserved for future use.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
{ $IFDEF DELPHIXE3 }
  DefaultAttribute = Classes.DefaultAttribute;
{ $ELSE }
  DefaultAttribute = class(TCustomAttribute)
{ $ENDIF }
```

# DiscriminatorAttribute Object

## [Hierarchy](#)

---

Specifies a [discriminator column](#) value that uniquely identifies rows that correspond to an [entity class](#) in the ancestor's table.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
DiscriminatorAttribute = class(TCustomAttribute)
```

## Description

Pass a value that corresponds to the column's value type as the **DiscriminatorAttribute**'s parameter. If not specified, the following value is used:

- | The class's fully qualified name applies to a [string](#) type column;
- | The hash value calculated for the class's fully qualified name applies to an [Integer](#) type column.

Refer to the [InheritanceAttribute](#) class description for an example on how to provide discriminator column values using the **DiscriminatorAttribute**.

# DiscriminatorColumnAttribute Object

## [Hierarchy](#)

Specifies a discriminator column whose [values](#) identify rows that correspond to an [entity class](#) and its [descendants](#) in the class's mapped table.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
DiscriminatorColumnAttribute = class(TCustomAttribute)
```

## Description

Configuring a descendant class to store its data in the ancestor's mapped table (by passing the [TdxMapInheritanceType.ParentTable](#) option to the [InheritanceAttribute](#)) requires that you create a discriminator column in this table. Starting with the ancestor, each entity class is associated with a value (called a discriminator column value) that uniquely identifies it in the inheritance tree. This value is stored with the class's data in a table row, allowing you to discern rows that correspond to the ancestor and its descendants.

The **DiscriminatorColumnAttribute** allows you to pass a discriminator column's name, [value type](#), and optionally, size for the string value type (corresponds to the [TdxDiscriminatorType.String](#) option) as parameters. Depending on the parameters specified, the attribute creates the following discriminator column:

- | If only the column name is specified, an integer discriminator column (corresponds to the [TdxDiscriminatorType.Integer](#) option) is created.
- | If no parameters are specified, an integer discriminator column with the name provided by the `TdxDiscriminator.ObjectTypePropertyName` constant is created.
- | If a string discriminator column's size is not specified, it equals the `TdxDiscriminator.ObjectTypePropertySize` constant value.

Refer to the [InheritanceAttribute](#) class description for an example on how to create a discriminator column using the **DiscriminatorColumnAttribute**.

# EdxODBCError Object

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.DataProvider.ADO](#)

## Syntax

```
EdxODBCError = class(Exception)
```

## Description

Help is not yet available.

# EntityAttribute Object

## [Hierarchy](#)

---

Marks a class as an entity class.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
EntityAttribute = class(TCustomAttribute)
```

## Description

This attribute allows you to include or exclude specific classes to/from an [entity model](#). A [session component](#) can only [manipulate](#) data stores and [query](#) their data using instances of entity classes.

You can specify a table to which an entity class is mapped using the [TableAttribute](#). Otherwise, the class name is considered the table name. The 'T' prefix is automatically removed.

The following code marks the TPerson class as an entity class. [Creating](#) a data store schema for this class maps it to the Person table.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    TPerson = class
    // ...
end;
```

# GeneratorAttribute Object

## [Hierarchy](#)

Specifies how an auto-incremented [primary key](#) generates its values.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
GeneratorAttribute = class(TCustomAttribute)
```

## Description

Automatic key value generation is supported for Integer, Int64, and TGUID type fields/properties. TGUID key values are always generated on the client side, while Integer and Int64 key values are generated by a data store, on its side. Depending on the data store type, automatic key generation is implemented using identity/autonumber/auto-increment columns or sequences. If a data store doesn't support any of these, the **ExpressEntityMapping Framework** generates key values.

Pass the generation type (a [TdxGeneratorType](#) enumeration value) as a parameter to the **GeneratorAttribute** parameter to specify how key values are generated. Refer to the [TdxGeneratorType](#) type description to learn about the available options.

The following code is the [KeyAttribute](#) example extended with automatic key value generation for the Id primary key column.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
type
    [Entity]
    TPerson = class
    strict private
        [Column, Key, Generator(TdxGeneratorType.Identity)]
        FId: Integer;
        [Column]
        FName: string;
    public
        // ...
    end;
```

# IndexedAttribute Object

## [Hierarchy](#)

Specifies if a field's/property's mapped column makes up a non-unique index on a data store table to which an [entity class](#) is mapped.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
IndexedAttribute = class(TCustomAttribute)
```

## Description

Unlike the [IndexesAttribute](#), the **IndexedAttribute** applies to a field/property and allows you to create a non-unique index on its mapped column. To make this index unique, add the [UniqueAttribute](#).

The following code shows how to use the **IndexedAttribute** and the [UniqueAttribute](#) to create a simple unique index on the ISDN column mapped to the FISDN field.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [Automapping]
    TGadget = class
    strict private
        FId: Integer;
        [Indexed, Unique]
        FISDN: string;
    public
        // ...
    end;
```

# IndexesAttribute Object

## [Hierarchy](#)

Specifies one or more columns that make up a non-unique index on a data store table to which an [entity class](#) is mapped.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
IndexesAttribute = class(TCustomAttribute)
```

## Description

Unlike the [IndexedAttribute](#), the **IndexesAttribute** applies to an entity class and allows you to create simple and multi-column indexes. To specify a multi-column index, pass two or more column names delimited by a comma or a semicolon. You can apply multiple **IndexesAttributes** to the same entity class – each of them creates a new index.

The following code shows how to use the **IndexesAttribute** to create two indexes: a simple index on the Age column and a multi-column index on the FirstName and LastName columns mapped to corresponding fields.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [Automapping]
    [Indexes('Age')]
    [Indexes('FirstName,LastName')]
    TPerson = class
    strict private
        FId: Integer;
        FAge: Integer;
        FFirstName: string;
        FLastName: string;
    public
        // ...
    end;
```

## Related Information

| [UniqueAttribute Object](#)

# InheritanceAttribute Object

## [Hierarchy](#)

Specifies where an [entity](#) descendant class stores its data.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
InheritanceAttribute = class(TCustomAttributes)
```

## Description

Defining class inheritance with the **InheritanceAttribute** requires that you specify where each entity class descendant stores its data. You can accomplish this by passing a [TdxMapInheritanceType](#) option as the attribute's parameter. The attribute uses the [TdxMapInheritanceType.ParentTable](#) option if nothing is passed. Storing data in the ancestor's table (corresponds to the [TdxMapInheritanceType.ParentTable](#) option) expects that you:

- Introduce a discriminator column in the ancestor using the [DiscriminatorColumnAttribute](#);
- Provide this class and each entity descendant class with a discriminator column value that uniquely identifies their rows in the table using the [DiscriminatorAttribute](#).

Entity class descendants inherit their [primary keys](#) from their ancestors – so you need to identify it only once per inheritance tree (in the ancestor).

The following code examples show how to define class inheritance for a simple [entity model](#) using the attributes described above. Note how the [TdxMapInheritanceType](#) options affect the resulting data store schemas.

Consider an entity model including two entities:

- A Person entity containing generic personal information;
- An Employee entity (a Person descendant), extending its ancestor with employment information.

The following entity model uses the [TdxMapInheritanceType.ParentTable](#) option to store the Employee entity's information in the Person entity's table. A discriminator column called 'Discriminator' is added to this table, providing identity values for Person and Employee entity rows (**0** and **1**, respectively).

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
type
    [Entity]
    [Automapping]
    [DiscriminatorColumn('Discriminator')]
    [Discriminator(0)]
    TPerson = class
    strict private
        FId: Integer;
        FAge: Integer;
        FName: string;
    public
        [Generator(TdxGeneratorType.Identity)]
```

```

property Id: Integer read FId;
property Age: Integer read FAge write FAge;
property Name: string read FName write FName;
end;
[Entity]
[Automapping]
[Inheritance(TdxMapInheritanceType.ParentTable)]
[Discriminator(1)]
TEmployee = class(TPerson)
strict private
  FHireDate: TDate;
  [Size(20)]
  FJob: string;
public
  property HireDate: TDate read FHireDate write FHireDate;
  property Job: string read FJob write FJob;
end;

```

Calling the [CreateSchema](#) procedure of a session component [connected](#) to an empty SQLite database and passing 'TEmployee' as a parameter creates the following schema:

Name	Type	Schema
▲ Tables (2)		
▲ Person		
HireDate	datetime	`HireDate` datetime
Job	nvarchar ( 20 )	`Job` nvarchar ( 20 )
Id	INTEGER	`Id` INTEGER PRIMARY KEY AUTOINCREMENT
Age	int	`Age` int NOT NULL
Name	nvarchar ( 100 )	`Name` nvarchar ( 100 ) NOT NULL
Discriminator	int	`Discriminator` int
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

The following entity model uses the [TdxMapInheritanceType.OwnTable](#) option to store the Employee entity's information in a separate table. With this option, the entity classes no longer require the [DiscriminatorColumnAttribute](#) and [DiscriminatorAttribute](#).

```

[Delphi]
uses
  ..., dxEMF.Core, dxEMF.Attributes, dxEMF.Types;
type
  [Entity]
  [Automapping]
  TPerson = class
  strict private
    FId: Integer;
    FAge: Integer;
    FName: string;
  public
    [Generator(TdxGeneratorType.Identity)]
    property Id: Integer read FId;
    property Age: Integer read FAge write FAge;

```

```

    property Name: string read FName write FName;
end;
[Entity]
[Automapping]
[Inheritance(TdxMapInheritanceType.OwnTable)]
TEmployee = class(TPerson)
strict private
    FHireDate: TDate;
    [Size(20)]
    FJob: string;
public
    property HireDate: TDate read FHireDate write FHireDate;
    property Job: string read FJob write FJob;
end;

```

Calling the [CreateSchema](#) procedure and passing 'TEmployee' as a parameter now produces the following schema in an empty SQLite database:

Name	Type	Schema
▲ Tables (3)		
▲ Employee		
HireDate	datetime	`HireDate` datetime NOT NULL
Job	nvarchar ( 20 )	`Job` nvarchar ( 20 ) NOT NULL
Id	int	`Id` int NOT NULL
▲ Person		
Id	INTEGER	`Id` INTEGER PRIMARY KEY AUTOINCREMENT
Age	int	`Age` int NOT NULL
Name	nvarchar ( 100 )	`Name` nvarchar ( 100 ) NOT NULL
Discriminator	int	`Discriminator` int
▶ sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

# KeyAttribute Object

## [Hierarchy](#)

Specifies one or more table columns that make up a primary key for an [entity class](#).

## Unit

[dxEMF.Attributes](#)

## Syntax

```
KeyAttribute = class(TCustomAttribute)
```

## Description

Primary key values uniquely identify entity objects (instances of entity classes) and records in a [mapped](#) data store table – no more than one entity object can refer to the same table record.

Options to apply the **KeyAttribute** include:

- 1 A field or a read-write property. Map this field/property to a primary key column using the [ColumnAttribute](#) or [AutomappingAttribute](#). Applying the **KeyAttribute** to a field allows you to expose this field via a read-only property, thus prohibiting any changes to its key value using the property. The following code shows how to accomplish this.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    TPerson = class
    strict private
        [Column, Key]
        FId: Integer;
        // ...
    public
        property Id: Integer read FId;
        // ...
    end;
```

- 1 An entity class. Pass the name of a column as the **KeyAttribute**'s parameter to designate this column as a primary key. To specify a composite primary key, pass two or more column names delimited by a comma or a semicolon as shown below. Alternatively, you can apply multiple **KeyAttributes** specifying each column as their parameter.

```
[Delphi]
type
    [Entity]
    [Key('ID,CourseID')]
    TPerson = class
    strict private
        FId: Integer;
        FCourseID: Integer;
        // ...
    public
        [Column]
```

```
property Id: Integer read FId;
[Column]
property CourseID: Integer read FCourseID write FCourseID;
// ...
end;
```

Do one of the following to provide a unique key value:

- 1 Manually assign a unique value to the key field in the class's constructor. [Saving](#) a newly created entity object stores this value to the mapped key column.
- 1 For a simple primary key, mark the key/property field with the [GeneratorAttribute](#) to allow the data store to automatically generate values for the key column mapped to this field/property when inserting a new table record.

# NoForeignKeyAttribute Object

[Hierarchy](#)

---

Disables automatic creation of foreign key constraints for an [association](#).

## Unit

[dxEMF.Attributes](#)

## Syntax

```
NoForeignKeyAttribute = class(TCustomAttribute)
```

## Description

Foreign keys preserve referential integrity in a relational database. A lack of referential integrity in a database can lead to errors or returning incomplete data without any indication of an error. To enforce the data integrity at the database level, we recommend that you leave automatic creation of the foreign key constraints enabled (or always add them manually).

# NonPersistentAttribute Object

[Hierarchy](#)

---

Excludes a field from a schema created for an [entity class](#) marked with the [AutomappingAttribute](#).

## Unit

[dxEMF.Attributes](#)

## Syntax

```
NonPersistentAttribute = class(TCustomAttribute)
```

## Description

Refer to the [AutomappingAttribute](#) description for more information and an example of how to use the **NonPersistentAttribute** attribute.

# NullableAttribute Object

## [Hierarchy](#)

---

Allows a table column mapped to an [entity class](#)'s field/property storing NULL values.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
NullableAttribute = class(TCustomAttribute)
```

## Description

All columns are set to contain only non-NULL values, except for the following:

- | Columns mapped to [nullable](#) fields or properties;
- | Columns storing BLOB values (mapped to TBytes or TStrings fields/properties, or fields/properties marked with the [BlobAttribute](#));
- | Columns that correspond to foreign keys created by any of the following: entity references used in [relationships](#), inherited primary keys in tables created for [TdxMapInheritanceType.OwnTable](#) descendant classes, etc.;
- | Columns that correspond to fields/properties mapped from [TdxMapInheritanceType.ParentTable](#) descendant classes to their ancestor's table.

Applying the **NullableAttribute** to a specific column overrides the default behavior and allows it to store NULL values. This is useful, for instance, when binding entity objects to data-aware controls using the [TdxEMFDataSet](#) component, because these controls normally populate a newly inserted table record with NULL values.

# ReadOnlyAttribute Object

## [Hierarchy](#)

---

Makes a [TdxEMFDataSet](#) component's field bound to an entity class's field/property read-only.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
ReadOnlyAttribute = class(TCustomAttribute)
```

## Description

[Assigning](#) a collection of entity objects to a [TdxEMFDataSet](#) component automatically creates a dataset field for each field/property contained in these objects. The **ReadOnlyAttribute** allows you to initialize a dataset field's ReadOnly property to **True** by marking the corresponding field/property with the **ReadOnlyAttribute**.

# SchemaNameAttribute Object

## [Hierarchy](#)

---

Specifies the name of a database schema in which a table mapped to an [entity class](#) is [created](#).

## Unit

[dxEMF.Attributes](#)

## Syntax

```
SchemaNameAttribute = class(TCustomAttribute)
```

## Description

The specified schema name is used only if the [connected](#) database supports schemas. The table is created in the default schema (e.g., 'dbo' for Microsoft SQL Server databases) if the schema name is not specified. In addition to the **SchemaNameAttribute**, you can specify a schema name via the [TableAttribute](#). If both the attributes specify schema names, the most recently specified one is in effect for the table.

The following code shows how to designate a schema called 'MySchema' for the TPerson entity class.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [SchemaName( 'MySchema' )]
    TPerson = class
    // ...
end;
```

# SizeAttribute Object

## [Hierarchy](#)

Specifies the size of a column to which a string or BLOB data field/property of an [entity class](#) is mapped.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
SizeAttribute = class(TCustomAttribute)
```

## Description

The default size is determined based on the field's/property's data type and is specific to the [database](#) within which a [session component](#) creates the entity model's [schema](#). String columns are limited to **100** characters (as specified by the `TdxSQLConnectionProvider.DefaultStringSize` class variable, which you can adjust as required). You can customize the default column size by passing a new size as the **SizeAttribute**'s parameter.

Parameter values are treated as follows:

- | **0** or no value. The default size is used (as described above);
- | A negative value or a value exceeding the maximum size allowed for variable character column types (such as `varchar` or `nvarchar`) in the connected database. A column size sufficient for storing Memo or BLOB data is used. This is equivalent to applying the [BlobAttribute](#) this field/property;
- | The specified size is used.

The following code shows how to use the **SizeAttribute** to limit the Name column mapped to the FName field to **40** characters.

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [Automapping]
    TPerson = class
    strict private
        FId: Integer;
        [Size(40)]
        FName: string;
    public
        // ...
    end;
```

## Related Information

- | [BlobAttribute Object](#)

# TableAttribute Object

## [Hierarchy](#)

---

Specifies a data store table to which an [entity class](#) maps its data.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
TableAttribute = class(TCustomAttributes)
```

## Description

Use this attribute to do the following:

- 1 Map an entity class to a specific data store table by passing its name as a parameter. Otherwise, the class name is considered the table name. The 'T' prefix is automatically removed;
- 1 Optionally specify a database schema within which this table is [created](#). The specified schema name is used only if the [connected](#) database supports schemas. The table is created in the default schema (e.g., 'dbo' for Microsoft SQL Server databases) if the schema name is not specified. In addition to the **TableAttribute**, you can specify a schema name via the [SchemaNameAttribute](#). If both the attributes specify schema names, the most recently specified one is in effect for the table.

The following code shows how to map the TPerson entity class to the 'MyPerson' table created in the 'MySchema' database schema (e.g., in a Microsoft SQL Server database, this corresponds to a 'MySchema.MyPerson' table).

```
[Delphi]
uses
    ..., dxEMF.Core, dxEMF.Attributes;
type
    [Entity]
    [Table('MySchema', 'MyPerson')]
    TPerson = class
    // ...
end;
```

# TdxAggregateOperand Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxAggregateOperand = class(TdxCriteriaOperator, IdxAggregateOperand)
```

## Description

Help is not yet available.

# TdxBetweenOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxBetweenOperator = class(TdxCriteriaOperator, IdxBetweenOperator)
```

## Description

Help is not yet available.

# TdxBinaryOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxBinaryOperator = class(TdxCriteriaOperator, IdxBinaryOperator)
```

## Description

Help is not yet available.

# TdxConnectionTypes Object

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
TdxConnectionTypes = class(TObject)
public const
    Unknown = 'Unknown';
    ADO = 'ADO';
    FireDAC = 'FireDAC';
end;
```

## Description

Help is not yet available.

# TdxConstantValue Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxConstantValue = class(TdxOperandValue, IdxConstantValue)
```

## Description

Help is not yet available.

# TdxContainsOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxContainsOperator = class(TdxAggregateOperand)
```

## Description

Help is not yet available.

# TdxCriteriaOperator Object

[Hierarchy](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxCriteriaOperator = class(TInterfacedObject, IdxCriteriaOperator, IdxExpression)
```

## Description

Help is not yet available.

# TdxCriteriaOperatorCollection Object

[Hierarchy](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxCriteriaOperatorCollection = class(TList<TdxCriteriaOperator>)
```

## Description

Help is not yet available.

# TdxDBEngines Object

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
TdxDBEngines = class(TObject)
public const
    Empty = '';
    Unknown = 'Unknown';
    Firebird = 'Firebird';
    MySQL = 'MySQL';
    MSSQL = 'MSSQLServer';
    Oracle = 'Oracle';
    SQLite = 'SQLite';
    MSAccess = 'MSAccess';
end;
```

## Description

Help is not yet available.

# TdxEMFCollectionOptions Object

[Hierarchy](#) [Properties](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFCollectionOptions = class(TPersistent)
```

## Description

Help is not yet available.

# TdxEMFCollections Object

[Hierarchy](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFCollections = class sealed (TObject)
```

## Description

Help is not yet available.

# TdxEMFCustomCollection Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFCustomCollection = class(TInterfacedObject, IdxQueryable, IEnumerable)
```

## Description

Help is not yet available.

# TdxEMFCustomDataProvider Object

[Hierarchy](#) [Properties](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFCustomDataProvider = class(TComponent)
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet Object

[Hierarchy](#) [Properties](#) [Methods](#) [Events](#)

---

Help is not yet available.

## Unit

[dxEMF.DataSet](#)

## Syntax

```
TdxEMFCustomDataSet = class(TDataSet)
```

## Description

Help is not yet available.

# TdxEMFCustomSession Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFCustomSession = class(TComponent, IdxDataContext, IdxPersistentValueExtractor, IdxQue
```

## Description

Help is not yet available.

# TdxEMFDataProviderOptions Object

[Hierarchy](#) [Properties](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFDataProviderOptions = class(TPersistent)
```

## Description

Help is not yet available.

# TdxEMFDataSetBlobStream Object

[Hierarchy](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DataSet](#)

## Syntax

```
TdxEMFDataSetBlobStream = class(TMemoryStream)
```

## Description

Help is not yet available.

# TdxEMFSessionOptions Object

[Hierarchy](#) [Properties](#)

---

Help is not yet available.

## Unit

[dxEMF.Core](#)

## Syntax

```
TdxEMFSessionOptions = class(TPersistent)
```

## Description

Help is not yet available.

# TdxFunctionOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxFunctionOperator = class(TdxCriteriaOperator, IdxFunctionOperator)
```

## Description

Help is not yet available.

# TdxGroupOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxGroupOperator = class(TdxCriteriaOperator, IdxGroupOperator)
```

## Description

Help is not yet available.

# TdxInOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxInOperator = class(TdxCriteriaOperator, IdxInOperator)
```

## Description

Help is not yet available.

# TdxJoinOperand Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxJoinOperand = class(TdxCriteriaOperator, IdxJoinOperand)
```

## Description

Help is not yet available.

# TdxLikeCustomFunction Object

[Hierarchy](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxLikeCustomFunction = class(TObject)
```

## Description

Help is not yet available.

# TdxOperandParameter Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

For internal use only.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxOperandParameter = class(TdxOperandValue, IdxOperandParameter)
```

# TdxOperandProperty Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxOperandProperty = class(TdxCriteriaOperator, IdxOperandProperty)
```

## Description

Help is not yet available.

# TdxOperandValue Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxOperandValue = class(TdxCriteriaOperator, IdxOperandValue)
```

## Description

Help is not yet available.

# TdxQueryOperand Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxQueryOperand = class(TdxCriteriaOperator, IdxQueryOperand)
```

## Description

Help is not yet available.

# TdxSortByExpression Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxSortByExpression = class(TInterfacedObject, IdxSortByExpression)
```

## Description

Help is not yet available.

# TdxSortByExpressions Object

[Hierarchy](#) [Properties](#) [Methods](#) [Events](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxSortByExpressions = class(TInterfacedObject, IdxSortByExpressions)
```

## Description

Help is not yet available.

# TdxUnaryOperator Object

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
TdxUnaryOperator = class(TdxCriteriaOperator, IdxUnaryOperator)
```

## Description

Help is not yet available.

# UniqueAttribute Object

[Hierarchy](#)

---

Creates a unique index on a column that is mapped to an [entity class](#)'s field/property.

## Unit

[dxEMF.Attributes](#)

## Syntax

```
UniqueAttribute = class(TCustomAttribute)
```

## Description

You can apply this attribute exclusively or combine it with the [IndexedAttribute](#) to make the resulting column index unique. Refer to the [IndexedAttribute](#) topic for an example on how to use the **UniqueAttribute**.

# IdxAggregateOperand.AggregatedExpression

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregatedExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxAggregateOperand.AggregateFunctionType

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxAggregateOperand.CollectionProperty

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property CollectionProperty: IdxOperandProperty;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxAggregateOperand.Condition

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property Condition: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxAggregateOperand.GetAggregatedExpression

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetAggregatedExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxAggregateOperand.GetAggregateFunctionType

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetAggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

# IdxAggregateOperand.GetCondition

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetCondition: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxAggregateOperand.GetIsTopLevel

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetIsTopLevel: Boolean;
```

## Description

Help is not yet available.

# IdxAggregateOperand.GetProperty

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetProperty: IdxOperandProperty;
```

## Description

Help is not yet available.

# IdxAggregateOperand.IsTopLevel

[IdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property IsTopLevel: Boolean;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBetweenOperator.BeginExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property BeginExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBetweenOperator.EndExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property EndExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBetweenOperator.GetBeginExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
function GetBeginExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxBetweenOperator.GetEndExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
function GetEndExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxBetweenOperator.GetTestExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
function GetTestExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxBetweenOperator.TestExpression

[IdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property TestExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBinaryOperator.GetLeftOperand

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetLeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxBinaryOperator.GetOperatorType

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperatorType: TdxBinaryOperatorType;
```

## Description

Help is not yet available.

# IdxBinaryOperator.GetRightOperand

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetRightOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxBinaryOperator.LeftOperand

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property LeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBinaryOperator.OperatorType

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxBinaryOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxBinaryOperator.RightOperand

[IdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property RightOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxCollection<T>.Add

[IdxCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function Add(AValue: T): Integer;
```

## Description

Help is not yet available.

# IdxCollection<T>.Contains

[IdxCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function Contains(AValue: T): Boolean;
```

## Description

Help is not yet available.

# IdxCollection<T>.Count

[IdxCollection<T>](#)

---

Help is not yet available.

## Syntax

```
property Count: Integer;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxCollection<T>.GetCount

[IdxCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function GetCount: Integer;
```

## Description

Help is not yet available.

# IdxCollection<T>.Remove

[IdxCollection<T>](#)

---

Help is not yet available.

## Syntax

```
procedure Remove(AValue: T);
```

## Description

Help is not yet available.

# IdxCollection.Count

[IdxCollection](#)

---

Help is not yet available.

## Syntax

```
property Count: Integer;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxCollection.GetCount

[IdxCollection](#)

---

Help is not yet available.

## Syntax

```
function GetCount: Integer;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.DeleteObjectOnRemove

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
property DeleteObjectOnRemove: Boolean;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.Find

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
type
    TdxPredicate<T> = reference to function(AData: T): Boolean;
function Find(const AKey: TValue): T; overload;
function Find(APredicate: TdxPredicate<T>): T; overload;
```

### Description

Help is not yet available.

# IdxEMFCollection<T>.First

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function First: T;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.GetDeleteObjectOnRemove

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function GetDeleteObjectOnRemove: Boolean;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.GetObjects

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
type  
    TdxPredicate<T> = reference to function(AData: T): Boolean;  
function GetObjects(APredicate: TdxPredicate<T>): TArray<T>; overload;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.Last

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
function Last: T;
```

## Description

Help is not yet available.

# IdxEMFCollection<T>.SetDeleteObjectOnRemove

[IdxEMFCollection<T>](#)

---

Help is not yet available.

## Syntax

```
procedure SetDeleteObjectOnRemove(AValue: Boolean);
```

## Description

Help is not yet available.

# IdxEMFCollection.DeleteObjectOnRemove

[IdxEMFCollection](#)

---

Help is not yet available.

## Syntax

```
property DeleteObjectOnRemove: Boolean;
```

## Description

Help is not yet available.

# IdxEMFCollection.First

[IdxEMFCollection](#)

---

Help is not yet available.

## Syntax

```
function First: TObject;
```

## Description

Help is not yet available.

# IdxEMFCollection.GetDeleteObjectOnRemove

[IdxEMFCollection](#)

---

Help is not yet available.

## Syntax

```
function GetDeleteObjectOnRemove: Boolean;
```

## Description

Help is not yet available.

# IdxEMFCollection.Last

[IdxEMFCollection](#)

---

Help is not yet available.

## Syntax

```
function Last: TObject;
```

## Description

Help is not yet available.

# IdxEMFCollection.SetDeleteObjectOnRemove

[IdxEMFCollection](#)

---

Help is not yet available.

## Syntax

```
procedure SetDeleteObjectOnRemove(AValue: Boolean);
```

## Description

Help is not yet available.

# IdxFunctionOperator.GetOperands

[IdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

# IdxFunctionOperator.GetOperatorType

[IdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperatorType: TdxFunctionOperatorType;
```

## Description

Help is not yet available.

# IdxFunctionOperator.Operands

[IdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxFunctionOperator.OperatorType

[IdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxFunctionOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxGroupOperator.GetOperands

[IdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

# IdxGroupOperator.GetOperatorType

[IdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperatorType: TdxGroupOperatorType;
```

## Description

Help is not yet available.

# IdxGroupOperator.Operands

[IdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxGroupOperator.OperatorType

[IdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxGroupOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxInOperator.GetLeftOperand

[IdxInOperator](#)

---

Help is not yet available.

## Syntax

```
function GetLeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxInOperator.GetOperands

[IdxInOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

# IdxInOperator.LeftOperand

[IdxInOperator](#)

---

Help is not yet available.

## Syntax

```
property LeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxInOperator.Operands

[IdxInOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxJoinOperand.AggregatedExpression

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregatedExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxJoinOperand.AggregateFunctionType

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxJoinOperand.Condition

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property Condition: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxJoinOperand.GetAggregatedExpression

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function GetAggregatedExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxJoinOperand.GetAggregateFunctionType

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function GetAggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

# IdxJoinOperand.GetCondition

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function GetCondition: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxJoinOperand.GetJoinTypeName

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function GetJoinTypeName: string;
```

## Description

Help is not yet available.

# IdxJoinOperand.JoinTypeName

[IdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property JoinTypeName: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxList<T>.GetItems

[IdxList<T>](#)

---

Help is not yet available.

## Syntax

```
function GetItems(AIndex: Integer): T;
```

## Description

Help is not yet available.

# IdxList<T>.IndexOf

[IdxList<T>](#)

---

Help is not yet available.

## Syntax

```
function IndexOf(AItem: T): Integer;
```

## Description

Help is not yet available.

# IdxList<T>.Items

[IdxList<T>](#)

---

Help is not yet available.

## Syntax

```
property Items[AIndex: Integer]: T; default;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxList.Add

[IdxList](#)

---

Help is not yet available.

## Syntax

```
function Add(AValue: TObject): Integer;
```

## Description

Help is not yet available.

# IdxList.Contains

[IdxList](#)

---

Help is not yet available.

## Syntax

```
function Contains(AValue: TObject): Boolean;
```

## Description

Help is not yet available.

# IdxList.GetItems

[IdxList](#)

---

Help is not yet available.

## Syntax

```
function GetItems(AIndex: Integer): TObject;
```

## Description

Help is not yet available.

# IdxList.IndexOf

[IdxList](#)

---

Help is not yet available.

## Syntax

```
function IndexOf(AObject: TObject): Integer;
```

## Description

Help is not yet available.

# IdxList.Items

[IdxList](#)

---

Help is not yet available.

## Syntax

```
property Items[AIndex: Integer]: TObject; default;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxList.Remove

[IdxList](#)

---

Help is not yet available.

## Syntax

```
procedure Remove(AValue: TObject);
```

## Description

Help is not yet available.

# IdxOperandParameter.GetParameterName

[IdxOperandParameter](#)

---

Help is not yet available.

## Syntax

```
function GetParameterName: string;
```

## Description

Help is not yet available.

# IdxOperandParameter.ParameterName

[IdxOperandParameter](#)

---

Help is not yet available.

## Syntax

```
property ParameterName: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxOperandProperty.GetPropertyName

[IdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
function GetPropertyName: string;
```

## Description

Help is not yet available.

# IdxOperandProperty.PropertyName

[IdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
property PropertyName: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxOperandValue.GetValue

[IdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
function GetValue: TValue;
```

## Description

Help is not yet available.

# IdxOperandValue.SetValue

[IdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
procedure SetValue(const AValue: TValue);
```

## Description

Help is not yet available.

# IdxOperandValue.Value

[IdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
property Value: TValue;
```

## Description

Help is not yet available.

# IdxQueryable<T>.GetProvider

[IdxQueryable<T>](#)

---

Help is not yet available.

## Syntax

```
function GetProvider: IdxQueryProvider<T>;
```

## Description

Help is not yet available.

# IdxQueryable<T>.Provider

[IdxQueryable<T>](#)

---

Help is not yet available.

## Syntax

```
property Provider: IdxQueryProvider<T>;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxQueryable.GetProvider

[IdxQueryable](#)

---

Help is not yet available.

## Syntax

```
function GetProvider: IdxQueryProvider;
```

## Description

Help is not yet available.

# IdxQueryable.Provider

[IdxQueryable](#)

---

Help is not yet available.

## Syntax

```
property Provider: IdxQueryProvider;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxQueryOperand.ColumnName

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property ColumnName: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxQueryOperand.ColumnType

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property ColumnType: TdxDBCColumnType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxQueryOperand.GetColumnName

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function GetColumnName: string;
```

## Description

Help is not yet available.

# IdxQueryOperand.GetColumnType

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function GetColumnType: TdxDBCColumnType;
```

## Description

Help is not yet available.

# IdxQueryOperand.GetNodeAlias

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function GetNodeAlias: string;
```

## Description

Help is not yet available.

# IdxQueryOperand.NodeAlias

[IdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property NodeAlias: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxQueryProvider<T>.CreateQuery

[IdxQueryProvider<T>](#)

---

For internal use only.

## Syntax

```
function CreateQuery(const AExpression: IdxExpression): IdxQueryable<T>;
```

# IdxQueryProvider.CreateQuery

[IdxQueryProvider](#)

---

For internal use only.

## Syntax

```
function CreateQuery(const AExpression: IdxExpression): IdxQueryable;
```

# IdxSession.GetLoadingStrategy

[IdxSession](#)

---

For internal use only.

## Syntax

```
type
  TdxLoadingStrategy = (Lazy, Eager);
function GetLoadingStrategy: TdxLoadingStrategy;
```

# IdxSession.LoadingStrategy

[IdxSession](#)

---

For internal use only.

**ReadOnly Property**

**Syntax**

```
type
    TdxLoadingStrategy = (Lazy, Eager);
property LoadingStrategy: TdxLoadingStrategy;
```

# IdxSortByExpression.Expression

[IdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
property Expression: IdxExpression;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxSortByExpression.GetExpression

[IdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
function GetExpression: IdxExpression;
```

## Description

Help is not yet available.

# IdxSortByExpression.GetSortingOrder

[IdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
type
    TdxSortingOrder = (Ascending, Descending);
function GetSortingOrder: TdxSortingOrder;
```

## Description

Help is not yet available.

# IdxSortByExpression.SortingOrder

[IdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
type  
    TdxSortingOrder = (Ascending, Descending);  
property SortingOrder: TdxSortingOrder;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxUnaryOperator.GetOperand

[IdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# IdxUnaryOperator.GetOperatorType

[IdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetOperatorType: TdxUnaryOperatorType;
```

## Description

Help is not yet available.

# IdxUnaryOperator.Operand

[IdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
property Operand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxUnaryOperator.OperatorType

[IdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxUnaryOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# IdxAggregateOperand Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxAggregateOperand = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxBetweenOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxBetweenOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxBinaryOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxBinaryOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxCollection<T> Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxCollection<T> = interface(IEnumerable<T>)
```

## Description

Help is not yet available.

# IdxCollection Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxCollection = interface(IEnumerable)
```

## Description

Help is not yet available.

# IdxConstantValue Interface

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxConstantValue = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxCriteriaOperator Interface

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxCriteriaOperator = interface(IdxExpression)
```

## Description

Help is not yet available.

# IdxEMFCollection<T> Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxEMFCollection<T> = interface(IdxList<T>)
```

## Description

Help is not yet available.

# IdxEMFCollection Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxEMFCollection = interface(IdxList)
```

## Description

Help is not yet available.

# IdxExpression Interface

[Hierarchy](#)

---

For internal use only.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxExpression = interface(IUnknown)
```

# IdxFunctionOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxFunctionOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxGroupOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxGroupOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxInOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxInOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxJoinOperand Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxJoinOperand = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxLinq Interface

[Hierarchy](#)

---

For internal use only.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxLinq = interface(IUnknown)
```

# IdxList<T> Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxList<T> = interface (IdxCollection<T>)
```

## Description

Help is not yet available.

# IdxList Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxList = interface(IdxCollection)
```

## Description

Help is not yet available.

# IdxOperandParameter Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxOperandParameter = interface(IdxOperandValue)
```

## Description

Help is not yet available.

# IdxOperandProperty Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxOperandProperty = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxOperandValue Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxOperandValue = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxQueryable<T> Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxQueryable<T> = interface(IEnumerable<T>)
```

## Description

Help is not yet available.

# IdxQueryable Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxQueryable = interface(IEnumerable)
```

## Description

Help is not yet available.

# IdxQueryOperand Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxQueryOperand = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# IdxQueryProvider<T> Interface

[Hierarchy](#) [Methods](#)

---

For internal use only.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxQueryProvider<T> = interface(IdxQueryProvider)
```

# IdxQueryProvider Interface

[Hierarchy](#) [Methods](#)

---

For internal use only.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxQueryProvider = interface(IUnknown)
```

# IdxSession Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

For internal use only.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxSession = interface(IUnknown)
```

# IdxSortByExpression Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxSortByExpression = interface(IUnknown)
```

## Description

Help is not yet available.

# IdxSortByExpressions Interface

[Hierarchy](#)

---

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
IdxSortByExpressions = interface(IEnumerable<IdxSortByExpression>)
```

## Description

Help is not yet available.

# IdxUnaryOperator Interface

[Hierarchy](#) [Properties](#) [Methods](#)

---

Help is not yet available.

## Unit

[dxEMF.DB.Criteria](#)

## Syntax

```
IdxUnaryOperator = interface(IdxCriteriaOperator)
```

## Description

Help is not yet available.

# TdxAggregateOperand.Accept

[TdxAggregateOperand](#)

---

## Overloaded Variants

### Syntax

```
procedure Accept(const AVisitor: IdxCriteriaVisitor); override;  
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxAggregateOperand.AggregatedExpression

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregatedExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxAggregateOperand.AggregateFunctionType

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Avg

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Avg(const AAggregatedExpression: IdxCriteriaOperator): IdxAggregateOperand;
```

## Description

Help is not yet available.

# TdxAggregateOperand.CollectionProperty

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property CollectionProperty: IdxOperandProperty;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Condition

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property Condition: TdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Count

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Count(const AAggregatedExpression: IdxCriteriaOperator = nil): IdxAggregateOperand
```

## Description

Help is not yet available.

# TdxAggregateOperand.Create

[TdxAggregateOperand](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const ACollectionProperty: IdxOperandProperty; const AAggregatedExpression: AAggregatedExpression);  
constructor Create(const ACollectionProperty: string; const AAggregatedExpression: string; AAggregateFunctionType: TdxAggregateFunctionType);  
constructor Create(const ACollectionProperty: string; AAggregateFunctionType: TdxAggregateFunctionType);  
constructor Create(const ACollectionProperty: string; const AAggregatedExpression: string; AAggregateFunctionType: TdxAggregateFunctionType);  
constructor Create(const ACollectionProperty: string; AAggregateFunctionType: TdxAggregateFunctionType);
```

### Description

Help is not yet available.

# TdxAggregateOperand.Equals

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Exists

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Exists(const AAggregatedExpression: IdxCriteriaOperator = nil): IdxAggregateOperand
```

## Description

Help is not yet available.

# TdxAggregateOperand.GetHashCode

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxAggregateOperand.IsTopLevel

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
property IsTopLevel: Boolean;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxAggregateOperand.Max

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Max(const AAggregatedExpression: IdxCriteriaOperator): IdxAggregateOperand;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Min

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Min(const AAggregatedExpression: IdxCriteriaOperator): IdxAggregateOperand;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Single

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Single(const AAggregatedExpression: IdxCriteriaOperator): IdxAggregateOperand;
```

## Description

Help is not yet available.

# TdxAggregateOperand.Sum

[TdxAggregateOperand](#)

---

Help is not yet available.

## Syntax

```
function Sum(const AAggregatedExpression: IdxCriteriaOperator): IdxAggregateOperand;
```

## Description

Help is not yet available.

# TdxBetweenOperator.Accept

[TdxBetweenOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TListstring; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxBetweenOperator.BeginExpression

[TdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property BeginExpression: TdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxBetweenOperator.Create

[TdxBetweenOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const ATestPropertyName: string; const ABeginValue, AEndValue: TValue);  
constructor Create(const ATestExpression, ABeginExpression, AEndExpression: IdxCriteriaOper);  
constructor Create(const ATestPropertyName: string; const ABeginExpression, AEndExpression: IdxCriteriaOper);
```

### Description

Help is not yet available.

# TdxBetweenOperator.EndExpression

[TdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property EndExpression: TdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxBetweenOperator.Equals

[TdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxBetweenOperator.GetHashCode

[TdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxBetweenOperator.TestExpression

[TdxBetweenOperator](#)

---

Help is not yet available.

## Syntax

```
property TestExpression: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxBinaryOperator.Accept

[TdxBinaryOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxBinaryOperator.Create

[TdxBinaryOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create(const ALeftOperator, ARightOperator: IdxCriteriaOperator; AType: TdxBinaryOperatorType)
constructor Create(const APropertyName: string; const AValue: TValue; AType: TdxBinaryOperatorType)
constructor Create(const APropertyName: string; const AValue: string; AType: TdxBinaryOperatorType)
constructor Create(const APropertyName: string; AValue: Double; AType: TdxBinaryOperatorType)
constructor Create(const APropertyName: string; AValue: Integer; AType: TdxBinaryOperatorType)
```

### Description

Help is not yet available.

# TdxBinaryOperator.Equals

[TdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxBinaryOperator.GetHashCode

[TdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxBinaryOperator.LeftOperand

[TdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property LeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxBinaryOperator.OperatorType

[TdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxBinaryOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxBinaryOperator.RightOperand

[TdxBinaryOperator](#)

---

Help is not yet available.

## Syntax

```
property RightOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxContainsOperator.Create

[TdxContainsOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const ACollectionProperty: IdxOperandProperty; const ACondition: IdxCriteriaOperator);  
constructor Create(const ACollectionProperty: string; const ACondition: IdxCriteriaOperator);
```

### Description

Help is not yet available.

# TdxCriteriaOperatorCollection.Equals

[TdxCriteriaOperatorCollection](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxCriteriaOperatorCollection.GetHashCode

[TdxCriteriaOperatorCollection](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxCriteriaOperatorCollection.ToString

[TdxCriteriaOperatorCollection](#)

---

Help is not yet available.

## Syntax

```
function ToString: string; override;
```

## Description

Help is not yet available.

# TdxCriteriaOperator.Accept

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
procedure Accept(const AVisitor: IdxCriteriaVisitor); overload; virtual; abstract;
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; overload;
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; overload; virtual; abstract;
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; overload; virtual;
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxCriteriaOperator.And

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function &And(const ALeftOperator, ARightOperator: IdxCriteriaOperator): IdxCriteriaOperator
class function &And(const AOperands: TArray<IdxCriteriaOperator>): IdxCriteriaOperator; overload
class function &And(AOperands: TList<IdxCriteriaOperator>): IdxCriteriaOperator; overload;
```

### Description

Help is not yet available.

# TdxCriteriaOperator.CriteriaEquals

[TdxCriteriaOperator](#)

---

Help is not yet available.

## Syntax

```
class function CriterionEquals(ALeftOperator: TdxCriteriaOperator; ARightOperator: IdxCriteriaOperator)
```

## Description

Help is not yet available.

# TdxCriteriaOperator.Equals

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function Equals(AObjectA, AObjectB: TObject): Boolean; overload;  
class function Equals(const ACriteriaA, ACriteriaB: IdxCriteriaOperator): Boolean; overload
```

### Description

Help is not yet available.

# TdxCriteriaOperator.Not

[TdxCriteriaOperator](#)

---

Help is not yet available.

## Syntax

```
function &Not: IdxUnaryOperator;
```

## Description

Help is not yet available.

# TdxCriteriaOperator.Or

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function &Or(const ALeftOperator, ARightOperator: IdxCriteriaOperator): IdxCriteriaOperator;  
class function &Or(AOperands: TList<IdxCriteriaOperator>): IdxCriteriaOperator; overload;
```

### Description

Help is not yet available.

# TdxCriteriaOperator.Parse

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function Parse(const AStringCriteria: string; out ACriteriaParametersList: TArray<IdxCriteriaOperator>): IdxCriteriaOperator;  
class function Parse(const ACriteria: string; const AParameters: array of TValue): IdxCriteriaOperator;  
class function Parse(const ACriteria: string): IdxCriteriaOperator; overload;
```

### Description

Help is not yet available.

# TdxCriteriaOperator.ParseList

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function ParseList(const ACriteriaList: string; out ACriteriaParametersList: TArray<TdxCriteriaOperator>): TdxCriteriaOperator; overload;
class function ParseList(const ACriteriaList: string; const AParameters: array of TValue): TdxCriteriaOperator; overload;
class function ParseList(const ACriteriaList: string): TArray<TdxCriteriaOperator>; overload;
```

### Description

Help is not yet available.

# TdxCriteriaOperator.ToString

[TdxCriteriaOperator](#)

---

## Overloaded Variants

### Syntax

```
class function ToString(const ACriteria: TdxCriteriaOperator): string; overload;  
function ToString: string; overload; override;
```

### Description

Help is not yet available.

# TdxEMFADODataProvider.Connection

[TdxEMFADODataProvider](#)

---

Specifies an ADO connection object used to connect a [session component](#) to a data store.

## Syntax

```
property Connection: TADOConnection;
```

# TdxEMFCollectionOptions.LoadingStrategy

[TdxEMFCollectionOptions](#)

---

Help is not yet available.

## Syntax

```
type
    TdxLoadingStrategy = (Lazy, Eager);
property LoadingStrategy: TdxLoadingStrategy;
```

## Description

Help is not yet available.

The default value of the **LoadingStrategy** property is **TdxLoadingStrategy.Lazy**.

# TdxEMFCollections.Create

[TdxEMFCollections](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
function Create(AClass: TClass): IdxEMFCollection; overload; static;  
function Create<T: class>: IdxEMFCollection<T>; overload; static;  
function Create<T: class>(AObject: TObject; const APropertyName: string = ''): IdxEMFCollect
```

### Description

Help is not yet available.

# TdxEMFCollections.Initialize

[TdxEMFCollections](#)

---

Help is not yet available.

## Syntax

```
procedure Initialize(AObject: TObject); static;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.CollectionElementClass

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property CollectionElementClass: TClass;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxEMFCustomCollection.Criteria

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property Criteria: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.Init

[TdxEMFCustomCollection](#)

---

## Overloaded Variants

### Syntax

```
procedure Init(ASession: TdxEMFCustomSession); virtual;  
procedure Init(ASession: TdxEMFCustomSession; AOwner: TObject; AReferencedProperty: TdxMapp
```

### Description

Help is not yet available.

# TdxEMFCustomCollection.Insert

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
procedure Insert(AIndex: Integer; AObject: TObject); virtual; abstract;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.Load

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
procedure Load; virtual; abstract;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.Loader

[TdxEMFCustomCollection](#)

---

For internal use only.

## Syntax

```
property Loader: TdxDataLoader;
```

# TdxEMFCustomCollection.Reload

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
procedure Reload;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.SelectDeleted

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property SelectDeleted: Boolean;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.Session

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property Session: TdxEMFCustomSession;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxEMFCustomCollection.SkipReturnedObjects

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property SkipReturnedObjects: Integer;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.ToArray

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
function ToArray: TArray<TObject>;
```

## Description

Help is not yet available.

# TdxEMFCustomCollection.TopReturnedObjects

[TdxEMFCustomCollection](#)

---

Help is not yet available.

## Syntax

```
property TopReturnedObjects: Integer;
```

## Description

Help is not yet available.

# TdxEMFCustomDataProvider.ActualDBEngineType

[TdxEMFCustomDataProvider](#)

---

Help is not yet available.

## Syntax

```
type
  TdxDBEngine = type string;
property ActualDBEngineType: TdxDBEngine;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxEMFCustomDataProvider.CanCreateDatabase

[TdxEMFCustomDataProvider](#)

---

Help is not yet available.

## Syntax

```
property CanCreateDatabase: Boolean;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxEMFCustomDataProvider.CanCreateSchema

[TdxEMFCustomDataProvider](#)

---

Help is not yet available.

## Syntax

```
property CanCreateSchema: Boolean;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxEMFCustomDataProvider.Options

[TdxEMFCustomDataProvider](#)

---

Help is not yet available.

## Syntax

```
property Options: TdxEMFDataProviderOptions;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.BookmarkValid

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
function BookmarkValid(ABookmark: TBookmark): Boolean; override;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.CompareBookmarks

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
function CompareBookmarks(ABookmark1, ABookmark2: TBookmark): Integer; override;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.CreateBlobStream

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
function CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream; override;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.Current

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
function Current: TObject; overload;  
function Current<T: class>: T; overload;
```

### Description

Help is not yet available.

# TdxEMFCustomDataSet.FlushChanges

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
procedure FlushChanges ;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.GetFieldData

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
function GetFieldData(AField: TField; Buffer: TValueBuffer): Boolean; override;
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.Locate

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
function Locate(const KeyFields: string; const KeyValues: Variant; Options: TLocateOptions)
```

## Description

Help is not yet available.

# TdxEMFCustomDataSet.ReadOnly

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
property ReadOnly: Boolean;
```

## Description

Help is not yet available.

The default value of the **ReadOnly** property is **False**.

# TdxEMFCustomDataSet.Session

[TdxEMFCustomDataSet](#)

---

Help is not yet available.

## Syntax

```
property Session: TdxEMFCustomSession;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Attach

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure Attach(AObject: TObject);
```

## Description

Help is not yet available.

# TdxEMFCustomSession.BeginTrackingChanges

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure BeginTrackingChanges ;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.CreateObject

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function CreateObject(AClass: TClass): TObject; virtual;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.CustomCreate

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function CustomCreate<T: class>(AFunc: TFunc<T>): TdxEMFCustomSession;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.DataProvider

[TdxEMFCustomSession](#)

---

Specifies the data provider used to link the session component with a connection object.

## Syntax

```
property DataProvider: TdxEMFCustomDataProvider;
```

## Description

You can assign any data provider component (a [TdxEMFCustomDataProvider](#) descendant) to this property. Refer to the [TdxEMFCustomDataProvider](#) class description to learn about the data provider components shipped with the **ExpressEntityMapping Framework**.

# TdxEMFCustomSession.Delete

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure Delete(AObject: TObject); virtual;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Detach

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure Detach(AObject: TObject);
```

## Description

Help is not yet available.

# TdxEMFCustomSession.DropChanges

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure DropChanges ;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Find

[TdxEMFCustomSession](#)

---

## Overloaded Variants

### Syntax

```
type
  TdxPredicate<T> = reference to function(AData: T): Boolean;
function Find(AClass: TClass; const AKey: TValue): TObject; overload;
function Find(AClass: TClass; APredicate: TdxPredicate): TObject; overload;
function Find(AClass: TClass; const ACriteria: IdxCriteriaOperator): TObject; overload;
function Find<T: class>(AKey: TValue): T; overload;
function Find<T: class>(APredicate: TdxPredicate<T>): T; overload;
function Find<T: class>(ACriteria: IdxCriteriaOperator): T; overload;
```

### Description

Help is not yet available.

# TdxEMFCustomSession.FlushChanges

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure FlushChanges ;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.GetDataContext

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
function GetDataContext: IdxDataContext; overload;  
function GetDataContext<T: IdxDataContext>: T; overload;
```

### Description

Help is not yet available.

# TdxEMFCustomSession.GetEntityInfo

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
function GetEntityInfo(AClass: TClass): IdxEntityInfo; overload;  
function GetEntityInfo<T: IdxEntityInfo>: T; overload;
```

### Description

Help is not yet available.

# TdxEMFCustomSession.GetObjects

[TdxEMFCustomSession](#)

---

## Overloaded Variants

### Syntax

**type**

```
TdxPredicate<T> = reference to function(AData: T): Boolean;
```

```
function GetObjects(AClass: TClass): IdxEMFCollection;
```

```
function GetObjects(AClass: TClass; const ACriteria: IdxCriteriaOperator): IdxEMFCollection;
```

```
function GetObjects(AClass: TClass; const ACriteria: IdxCriteriaOperator; const ASortByExpressions: IdxSortByExpressions);
```

```
function GetObjects(const AQuery: IdxQueryable): IdxEMFCollection;
```

```
function GetObjects<T: class>: IdxEMFCollection<T>; overload;
```

```
function GetObjects<T: class>(APredicate: TdxPredicate<T>): IdxEMFCollection<T>; overload;
```

```
function GetObjects<T: class>(ACriteria: IdxCriteriaOperator): IdxEMFCollection<T>; overload;
```

```
function GetObjects<T: class>(ACriteria: IdxCriteriaOperator; ASortByExpressions: IdxSortByExpressions);
```

```
function GetObjects<T: class>(AQuery: IdxQueryable<T>): IdxEMFCollection<T>; overload;
```

### Description

Help is not yet available.

# TdxEMFCustomSession.GetObjectsToDelete

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function GetObjectsToDelete: TArray<TObject>; overload; inline;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.GetObjectsToSave

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function GetObjectsToSave: TArray<TObject>; overload; inline;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.GetQueryProvider<T>

[TdxEMFCustomSession](#)

---

For internal use only.

## Syntax

```
function GetQueryProvider<T: class>: IdxQueryProvider<T>;
```

# TdxEMFCustomSession.IsNewObject

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function IsNewObject(AObject: TObject): Boolean;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.IsObjectToDelete

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
function IsObjectToDelete(AObject: TObject): Boolean;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Options

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
property Options: TdxEMFSessionOptions;
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Reload

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure Reload(AObject: TObject);
```

## Description

Help is not yet available.

# TdxEMFCustomSession.Save

[TdxEMFCustomSession](#)

---

Help is not yet available.

## Syntax

```
procedure Save(AObject: TObject); virtual;
```

## Description

Help is not yet available.

# TdxEMFDataProviderOptions.AutoCreate

[TdxEMFDataProviderOptions](#)

---

Specifies the actions that the data provider performs on connecting to a data store.

## Syntax

```
type  
    TdxAutoCreateOption = (None, DatabaseAndSchema, SchemaOnly, SchemaAlreadyExists);  
property AutoCreate: TdxAutoCreateOption;
```

## Description

The default value of the **AutoCreate** property is **TdxAutoCreateOption.None**.

# TdxEMFDataProviderOptions.DBEngine

## [TdxEMFDataProviderOptions](#)

Specifies a target [database engine](#) by its name.

### Syntax

```
type
  TdxDBEngine = type string;
property DBEngine: TdxDBEngine;
```

### Description

You can assign any [TdxDBEngines](#) class's public constant to this property. Assigning the [TdxDBEngines.Unknown](#) or [TdxDBEngines.Empty](#) string causes the data provider to determine the engine type from its connection settings (see [TdxEMFFireDACDataProvider.Connection](#) or [TdxEMFADODDataProvider.Connection](#)).

At design time, the **DBEngine** property provides a drop-down list populated with supported database engines. Selecting a database engine from the drop-down list automatically adds a unit with a corresponding SQL connection provider used internally to build engine-specific queries. Specifying the **DBEngine** property in code requires that you manually add this unit to the 'uses' clause in your project. Use the following table for reference.

Database Engine Type	SQL Connection Provider Unit
TdxDBEngines.Firebird	dxEMF.DB.Firebird
TdxDBEngines.MSAccess	dxEMF.DB.MSAccess
TdxDBEngines.MSSQL	dxEMF.DB.MSSQL
TdxDBEngines.MySQL	dxEMF.DB.MySQL
TdxDBEngines.Oracle	dxEMF.DB.Oracle
TdxDBEngines.SQLite	dxEMF.DB.SQLite

The following code snippet uses the **DBEngine** property to select an SQLite database as a target database for a dataProvider. Note that to allow the **ExpressEntityMapping Framework** to access this database, you also need to add the dxEMF.DB.SQLite unit to the project's 'uses' clause.

```
[Delphi]
dataProvider.Options.DBEngine := TdxDBEngines.SQLite;
```

# TdxEMFDataSetBlobStream.Create

[TdxEMFDataSetBlobStream](#)

---

Help is not yet available.

## Syntax

```
constructor Create(AField: TBlobField; AMode: TBlobStreamMode);
```

## Description

Help is not yet available.

# TdxEMFDataSetBlobStream.Destroy

[TdxEMFDataSetBlobStream](#)

---

Help is not yet available.

## Syntax

```
destructor Destroy; override;
```

## Description

Help is not yet available.

# TdxEMFDataSetBlobStream.Write

[TdxEMFDataSetBlobStream](#)

---

Help is not yet available.

## Syntax

```
function Write(const Buffer; Count: Longint): Longint; override;
```

## Description

Help is not yet available.

# TdxEMFDataSet.AssignData

[TdxEMFDataSet](#)

---

Help is not yet available.

## Overloaded Variants

### Syntax

```
procedure AssignData(AData: TObject); overload;  
procedure AssignData(const AData: IdxEMFCollection); overload;  
procedure AssignData<T: class>(const AData: IdxEMFCollection<T>); overload;
```

### Description

Help is not yet available.

# TdxEMFFireDACDataProvider.Connection

[TdxEMFFireDACDataProvider](#)

---

Specifies a FireDAC connection object used to connect a [session component](#) to a data store.

## Syntax

```
{ $IFDEF DELPHIXE5 }
  TFDConnection = TADConnection;
{ $ENDIF }
property Connection: TFDConnection;
```

# TdxEMFSessionOptions.Collections

[TdxEMFSessionOptions](#)

---

Help is not yet available.

## Syntax

```
property Collections: TdxEMFCollectionOptions;
```

## Description

Help is not yet available.

# TdxEMFSessionOptions.LockingOption

[TdxEMFSessionOptions](#)

---

Reserved for future use.

## Syntax

```
type
    TdxLockingOption = (None, Optimistic);
property LockingOption: TdxLockingOption;
```

## Description

The default value of the **LockingOption** property is **TdxLockingOption.None**.

# TdxEMFSession.CreateSchema

[TdxEMFSession](#)

---

Creates a data store's schema, checks if it matches the [entity model](#), and adds missing data store tables and columns.

## Overloaded Variants

### Syntax

```
procedure CreateSchema(AClass: TClass);  
procedure CreateSchema(const AClasses: array of TClass);
```

### Description

This procedure accepts one or more [entity classes](#) whose declarations are checked against the [connected](#) data store's schema. When using [class inheritance](#), you can omit listing all entity classes in a hierarchy (starting from base classes) and list only final descendants, because they are sufficient for the **CreateSchema** procedure to build an entire hierarchy tree.

# TdxFunctionOperator.Accept

[TdxFunctionOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxFunctionOperator.Create

[TdxFunctionOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(AType: TdxFunctionOperatorType; const AOperands: TArray<IdxCriteriaOperat>;  
constructor Create(AType: TdxFunctionOperatorType; const AOperands: array of IdxCriteriaOper>;  
constructor Create(AType: TdxFunctionOperatorType; AOperands: TEnumerable<IdxCriteriaOperat>;  
constructor Create(const ACustomFunctionName: string; const AOperands: TArray<IdxCriteriaOper>;  
constructor Create(const ACustomFunctionName: string; const AOperands: array of IdxCriteria>;  
constructor Create(const ACustomFunctionName: string; const AOperands: TEnumerable<IdxCriter>;
```

### Description

Help is not yet available.

# TdxFunctionOperator.Destroy

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
destructor Destroy; override;
```

## Description

Help is not yet available.

# TdxFunctionOperator.Equals

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxFunctionOperator.GetHashCode

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxFunctionOperator.GuessIsLogicalCustomFunction

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
class function GuessIsLogicalCustomFunction(const AOperator: TdxFunctionOperator): Boolean;
```

## Description

Help is not yet available.

# TdxFunctionOperator.Operands

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxFunctionOperator.OperatorType

[TdxFunctionOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxFunctionOperatorType;
```

## Description

Help is not yet available.

# TdxGroupOperator.Accept

[TdxGroupOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxGroupOperator.Combine

[TdxGroupOperator](#)

---

## Overloaded Variants

### Syntax

```
class function Combine(AOperatorType: TdxGroupOperatorType; const ALeftOperator, ARightOperator: TdxGroupOperatorType): TdxGroupOperatorType;  
class function Combine(AOperatorType: TdxGroupOperatorType; const AOperands: array of IdxCriteriaOperator): TdxGroupOperatorType;  
class function Combine(AOperatorType: TdxGroupOperatorType; AOperands: TList<IdxCriteriaOperator>): TdxGroupOperatorType;
```

### Description

Help is not yet available.

# TdxGroupOperator.Create

[TdxGroupOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create(AType: TdxGroupOperatorType = TdxGroupOperatorType.TAnd); overload;  
constructor Create(AType: TdxGroupOperatorType; const AOperands: TArray<IdxCriteriaOperator>); overload;  
constructor Create(AType: TdxGroupOperatorType; const AOperands: array of IdxCriteriaOperator); overload;  
constructor Create(AType: TdxGroupOperatorType; const AOperands: TEnumerable<IdxCriteriaOperator>); overload;  
constructor Create(const AOperands: TArray<IdxCriteriaOperator>); overload;  
constructor Create(const AOperands: array of IdxCriteriaOperator); overload;
```

### Description

Help is not yet available.

# TdxGroupOperator.Destroy

[TdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
destructor Destroy; override;
```

## Description

Help is not yet available.

# TdxGroupOperator.Equals

[TdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxGroupOperator.GetHashCode

[TdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxGroupOperator.Operands

[TdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxGroupOperator.OperatorType

[TdxGroupOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxGroupOperatorType;
```

## Description

Help is not yet available.

# TdxInOperator.Accept

[TdxInOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxInOperator.Create

[TdxInOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create(const ALeftOperand: IdxCriteriaOperator); overload;  
constructor Create(const APropertyName: string; const AOperands: TArray<TObject>); overload;  
constructor Create(const APropertyName: string; const AOperands: array of const); overload;  
constructor Create(const ALeftOperand: IdxCriteriaOperator; const AOperands: TArray<IdxCriteriaOperator>); overload;  
constructor Create(const ALeftOperand: IdxCriteriaOperator; const AOperands: array of IdxCriteriaOperator); overload;  
constructor Create(const ALeftOperand: IdxCriteriaOperator; const AOperands: array of const IdxCriteriaOperator); overload;  
constructor Create(const ALeftOperand: IdxCriteriaOperator; AOperands: TEnumerable<IdxCriteriaOperator>); overload;  
class function Create<T>( const APropertyName: string; const AOperands: array of T): TdxInOperator
```

### Description

Help is not yet available.

# TdxInOperator.Destroy

[TdxInOperator](#)

---

Help is not yet available.

## Syntax

```
destructor Destroy; override;
```

## Description

Help is not yet available.

# TdxInOperator.Equals

[TdxInOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxInOperator.GetHashCode

[TdxInOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxInOperator.LeftOperand

[TdxInOperator](#)

---

Help is not yet available.

## Syntax

```
property LeftOperand: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxInOperator.Operands

[TdxInOperator](#)

---

Help is not yet available.

## Syntax

```
property Operands: TdxCriteriaOperatorCollection;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxJoinOperand.Accept

[TdxJoinOperand](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxJoinOperand.AggregatedExpression

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregatedExpression: TdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxJoinOperand.AggregateFunctionType

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property AggregateFunctionType: TdxAggregateFunctionType;
```

## Description

Help is not yet available.

# TdxJoinOperand.Avg

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Avg(const AAggregatedExpression: IdxCriteriaOperator): IdxJoinOperand;
```

## Description

Help is not yet available.

# TdxJoinOperand.Condition

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property Condition: IdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxJoinOperand.Count

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Count(const AAggregatedExpression: IdxCriteriaOperator = nil): IdxJoinOperand;
```

## Description

Help is not yet available.

# TdxJoinOperand.Create

[TdxJoinOperand](#)

---

## Overloaded Variants

### Syntax

```
constructor Create(const AJoinTypeName: string; const ACondition: IdxCriteriaOperator; AType: string);  
constructor Create(const AJoinTypeName: string; const ACondition: IdxCriteriaOperator); overloaded
```

### Description

Help is not yet available.

# TdxJoinOperand.Equals

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxJoinOperand.GetHashCode

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxJoinOperand.JoinTypeName

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
property JoinTypeName: string;
```

## Description

Help is not yet available.

# TdxJoinOperand.Max

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Max(const AAggregatedExpression: IdxCriteriaOperator): IdxJoinOperand;
```

## Description

Help is not yet available.

# TdxJoinOperand.Min

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Min(const AAggregatedExpression: IdxCriteriaOperator): IdxJoinOperand;
```

## Description

Help is not yet available.

# TdxJoinOperand.Sum

[TdxJoinOperand](#)

---

Help is not yet available.

## Syntax

```
function Sum(const AAggregatedExpression: IdxCriteriaOperator): IdxJoinOperand;
```

## Description

Help is not yet available.

# TdxLikeCustomFunction.Convert

[TdxLikeCustomFunction](#)

---

## Overloaded Variants

### Syntax

```
class function Convert(const ALike: IdxFunctionOperator): IdxBinaryOperator; overload;  
class function Convert(const ALike: IdxBinaryOperator): IdxFunctionOperator; overload;
```

### Description

Help is not yet available.

# TdxLikeCustomFunction.Create

[TdxLikeCustomFunction](#)

---

Help is not yet available.

## Syntax

```
class function Create(const AValue, APattern: IdxCriteriaOperator): TdxFunctionOperator;
```

## Description

Help is not yet available.

# TdxLikeCustomFunction.IsBinaryCompatibleLikeFunction

[TdxLikeCustomFunction](#)

---

Help is not yet available.

## Syntax

```
class function IsBinaryCompatibleLikeFunction(const AFunc: IdxFUNCTIONOperator): Boolean;
```

## Description

Help is not yet available.

# TdxLikeCustomFunction.IsName

[TdxLikeCustomFunction](#)

---

Help is not yet available.

## Syntax

```
class function IsName(const AName: string): Boolean;
```

## Description

Help is not yet available.

# TdxOperandParameter.Create

[TdxOperandParameter](#)

---

For internal use only.

## Overloaded Variants

### Syntax

```
constructor Create(const AParameterName: string; const AValue: TValue); overload;  
constructor Create(const AParameterName: string); overload;
```

# TdxOperandParameter.Equals

[TdxOperandParameter](#)

---

For internal use only.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

# TdxOperandParameter.GetHashCode

[TdxOperandParameter](#)

---

For internal use only.

## Syntax

```
function GetHashCode: Integer; override;
```

# TdxOperandParameter.ParameterName

[TdxOperandParameter](#)

---

For internal use only.

## Syntax

```
property ParameterName: string;
```

# TdxOperandProperty.Accept

[TdxOperandProperty](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxOperandProperty.Create

[TdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
constructor Create(const APropertyName: string);
```

## Description

Help is not yet available.

# TdxOperandProperty.Equals

[TdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxOperandProperty.GetHashCode

[TdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxOperandProperty.PropertyName

[TdxOperandProperty](#)

---

Help is not yet available.

## Syntax

```
property PropertyName: string;
```

## Description

Help is not yet available.

# TdxOperandValue.Accept

[TdxOperandValue](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TListstring; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxOperandValue.Create

[TdxOperandValue](#)

---

## Overloaded Variants

### Syntax

```
constructor Create(const AValue: TValue); overload;  
constructor Create(const AValue: TDateTime); overload;  
constructor Create(const AValue: string); overload;  
constructor Create(const AValue: Char); overload;  
class function Create<T>(AValue: T): TdxOperandValue; overload;
```

### Description

Help is not yet available.

# TdxOperandValue.Equals

[TdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxOperandValue.GetHashCode

[TdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxOperandValue.Value

[TdxOperandValue](#)

---

Help is not yet available.

## Syntax

```
property Value: TValue;
```

## Description

Help is not yet available.

# TdxQueryOperand.Accept

[TdxQueryOperand](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; override;
```

### Description

Help is not yet available.

# TdxQueryOperand.Clone

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function Clone: TdxQueryOperand;
```

## Description

Help is not yet available.

# TdxQueryOperand.ColumnName

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property ColumnName: string;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxQueryOperand.ColumnType

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property ColumnType: TdxDBCColumnType;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxQueryOperand.Create

[TdxQueryOperand](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const AColumnName: string; const ANodeAlias: string); overload;  
constructor Create(const AColumnName: string; const ANodeAlias: string; AColumnType: TdxDBC
```

### Description

Help is not yet available.

# TdxQueryOperand.Equals

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxQueryOperand.GetHashCode

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxQueryOperand.NodeAlias

[TdxQueryOperand](#)

---

Help is not yet available.

## Syntax

```
property NodeAlias: string;
```

## Description

Help is not yet available.

# TdxSortByExpressions.Add

[TdxSortByExpressions](#)

---

## Overloaded Variants

### Syntax

```
procedure Add(const ASortProperties: IdxSortByExpressions); overload;  
function Add(const AValue: IdxSortByExpression): Integer; overload;
```

### Description

Help is not yet available.

# TdxSortByExpressions.AddRange

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
procedure AddRange(const ASortProperties: TArray<TdxSortByExpression>);
```

## Description

Help is not yet available.

# TdxSortByExpressions.Clear

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
procedure Clear;
```

## Description

Help is not yet available.

# TdxSortByExpressions.Contains

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
function Contains(AValue: TdxSortByExpression): Boolean;
```

## Description

Help is not yet available.

# TdxSortByExpressions.Count

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
property Count: Integer;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxSortByExpressions.Create

[TdxSortByExpressions](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const ASortProperties: TArray<TdxSortByExpression>); overload;
```

### Description

Help is not yet available.

# TdxSortByExpressions.Destroy

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
destructor Destroy; override;
```

## Description

Help is not yet available.

# TdxSortByExpressions.IndexOf

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
function IndexOf(AValue: TdxSortByExpression): Integer;
```

## Description

Help is not yet available.

# TdxSortByExpressions.Insert

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
procedure Insert(AIndex: Integer; AValue: TdxSortByExpression);
```

## Description

Help is not yet available.

# TdxSortByExpressions.Items

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
property Items[AIndex: Integer]: IdxSortByExpression;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxSortByExpressions.OnChanged

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
property OnChanged: TNotifyEvent;
```

## Description

Help is not yet available.

# TdxSortByExpressions.Remove

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
procedure Remove(AValue: TdxSortByExpression);
```

## Description

Help is not yet available.

# TdxSortByExpressions.RemoveAt

[TdxSortByExpressions](#)

---

Help is not yet available.

## Syntax

```
procedure RemoveAt(AIndex: Integer);
```

## Description

Help is not yet available.

# TdxSortByExpression.Create

[TdxSortByExpression](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(const AProperty: IdxCriteriaOperator; ASortingOrder: TdxSortingOrder = TdxSortingOrderDefault);  
constructor Create(const APropertyName: string; ASortingOrder: TdxSortingOrder = TdxSortingOrderDefault);
```

### Description

Help is not yet available.

# TdxSortByExpression.Property

[TdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
property &Property: IdxCriteriaOperator;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxSortByExpression.PropertyName

[TdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
property PropertyName: string;
```

## Description

Help is not yet available.

# TdxSortByExpression.SortingOrder

[TdxSortByExpression](#)

---

Help is not yet available.

## Syntax

```
property SortingOrder: TdxSortingOrder;
```

## Description

Help is not yet available.

# TdxUnaryOperator.Accept

[TdxUnaryOperator](#)

---

## Overloaded Variants

### Syntax

```
function Accept(const AVisitor: IdxCriteriaOperatorVisitor): IdxCriteriaOperator; override;  
function Accept(const AVisitor: IdxSQLGeneratorVisitor): string; override;  
function Accept(const AVisitor: IdxClientCriteriaToStringVisitor): TdxCriteriaToStringVisitor;  
function Accept(const AVisitor: IdxCriteriaVisitorListString): TList<string>; override;  
function Accept(const AVisitor: IdxClientCriteriaVisitor<TdxBooleanCriteriaState>): TdxBooleanCriteriaState;
```

### Description

Help is not yet available.

# TdxUnaryOperator.Create

[TdxUnaryOperator](#)

---

## Overloaded Variants

### Syntax

```
constructor Create; overload;  
constructor Create(AOperatorType: TdxUnaryOperatorType; const AOperand: IdxCriteriaOperator
```

### Description

Help is not yet available.

# TdxUnaryOperator.Equals

[TdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
function Equals(AObject: TObject): Boolean; override;
```

## Description

Help is not yet available.

# TdxUnaryOperator.GetHashCode

[TdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
function GetHashCode: Integer; override;
```

## Description

Help is not yet available.

# TdxUnaryOperator.Operand

[TdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
property Operand: TdxCriteriaOperator;
```

## Description

Help is not yet available.

# TdxUnaryOperator.OperatorType

[TdxUnaryOperator](#)

---

Help is not yet available.

## Syntax

```
property OperatorType: TdxUnaryOperatorType;
```

## Description

Help is not yet available.

## ReadOnly Property

# TdxAggregateFunctionType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxAggregateFunctionType = (Exists, Count, Max, Min, Avg, Sum, Single);
```

## Description

Help is not yet available.

# TdxAttribute type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxAttribute = (  
  //Entity attributes  
  Entity, Automapping, Table, SchemaName, Inheritance,  
  //Column attributes  
  Column, NonPersistent, Association, Aggregated, Size, ReadOnly, Nullable, Default, Key,
```

## Description

Help is not yet available.

# TdxAttributes type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxAttribute = (  
  //Entity attributes  
  Entity, Automapping, Table, SchemaName, Inheritance,  
  //Column attributes  
  Column, NonPersistent, Association, Aggregated, Size, ReadOnly, Nullable, Default, Key,  
  TdxAttributes = set of TdxAttribute;
```

## Description

Help is not yet available.

# TdxAttributeType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxAttributeType = (&Class, &Property, Field, Index, Service);
```

## Description

Help is not yet available.

# TdxAutoCreateOption type

Enumerates options for the [Options.AutoCreate](#) property exposed by data providers ([TdxEMFFireDACDataProvider](#) and [TdxEMFADODDataProvider](#)).

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxAutoCreateOption = (None, DatabaseAndSchema, SchemaOnly, SchemaAlreadyExists);
```

## Description

Values include:

Value	Description
<b>DatabaseAndSchema</b>	The data provider attempts to create a database if it doesn't exist. It also creates or updates a database schema to match the entity model (i.e., its runtime type information, also called RTTI).
<b>SchemaOnly</b>	The data provider only creates or updates a database schema to match the entity model.
<b>None</b>	The data provider does not create a database or update its schema, but checks if the schema matches the entity model. Otherwise, the data provider raises the <code>EdxSchemaCorrectionNeededException</code> .
<b>SchemaAlreadyExists</b>	The data provider neither updates a database schema nor checks if it matches the entity model. This option suppresses the <code>EdxSchemaCorrectionNeededException</code> , but a database error may occur during data operations if the schema is not compatible with the entity model.

**Note:** `TdxAutoCreateOption` is a scoped enumeration type. To use its values in code, prefix them with the type name and a scope resolution operator (the `.` operation in Delphi). For example, use `TdxAutoCreateOption.DatabaseAndSchema` to refer to the `DatabaseAndSchema` value.

# TdxBinaryOperatorType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxBinaryOperatorType = (Equal, NotEqual, Greater, Less, LessOrEqual, GreaterOrEqual, Bitwise)
```

## Description

Help is not yet available.

# TdxBooleanCriteriaState type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxBooleanCriteriaState = (Logical, Value, Undefined);
```

## Description

Help is not yet available.

# TdxConnectionParameter type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxConnectionParameter = (Database);
```

## Description

Help is not yet available.

# TdxConnectionType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxConnectionType = type string;
```

## Description

Help is not yet available.

# TdxDBCColumnType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxDBCColumnType = (Unknown, Boolean, Byte, SByte, Char, Decimal, Double, Single, Int32, U
```

## Description

Help is not yet available.

# TdxDBEngine type

A string that matches any [TdxDBEngines](#) class's public constant.

## Unit

[dxEMF.Types](#)

## Syntax

```
type
  TdxDBEngine = type string;
```

# TdxDiscriminatorType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxDiscriminatorType = (&String, Integer);
```

## Description

Help is not yet available.

# TdxFunc<T, TResult> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxFunc<T, TResult> = reference to function (const AArg1: T): TResult;
```

## Description

Help is not yet available.

# TdxFunc<T1, T2, T3, T4, TResult> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxFunc<T1, T2, T3, T4, TResult> = reference to function (const AArg1: T1; const AArg2: T2; const AArg3: T3; const AArg4: T4) TResult
```

## Description

Help is not yet available.

# TdxFunc<T1, T2, T3, TResult> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxFunc<T1, T2, T3, TResult> = reference to function (const AArg1: T1; const AArg2: T2; c
```

## Description

Help is not yet available.

# TdxFunc<T1, T2, TResult> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxFunc<T1, T2, TResult> = reference to function (const AArg1: T1; const AArg2: T2): TResult
```

## Description

Help is not yet available.

# TdxFunc<TResult> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxFunc<TResult> = reference to function: TResult;
```

## Description

Help is not yet available.

# TdxFunctionCategory type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxFunctionCategory = (DateTime, Logical, Math, Text, All);
```

## Description

Help is not yet available.

# TdxFunctionOperatorType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

### type

```
TdxFunctionOperatorType = (None, Custom, CustomNonDeterministic, Iif, IsNull, IsNullOrEmpty,
    PadLeft, PadRight, StartsWith, EndsWith, Contains,
    ToInteger, ToInt64, ToSingle, ToDouble, ToDecimal,
    LocalDateTimeThisYear, LocalDateTimeThisMonth, LocalDateTimeLastWeek, LocalDateTimeThisYear,
    IsOutlookIntervalBeyondThisYear, // NextYear <= x
    IsOutlookIntervalLaterThisYear, // NextMonth <= x < NextYear
    IsOutlookIntervalLaterThisMonth, // TwoWeeksAway <= x < NextMonth
    IsOutlookIntervalNextWeek, // NextWeek <= x < TwoWeeksAway
    IsOutlookIntervalLaterThisWeek, // DayAfterTomorrow <= x < NextWeek
    IsOutlookIntervalTomorrow, // Tomorrow <= x < DayAfterTomorrow
    IsOutlookIntervalToday, // Today <= x < Tomorrow
    IsOutlookIntervalYesterday, // Yesterday <= x < Today
    IsOutlookIntervalEarlierThisWeek, // ThisWeek <= x < Yesterday
    IsOutlookIntervalLastWeek, // LastWeek <= x < ThisWeek
    IsOutlookIntervalEarlierThisMonth, // ThisMonth <= x < LastWeek
    IsOutlookIntervalEarlierThisYear, // ThisYear <= x < ThisMonth
    IsOutlookIntervalPriorThisYear, // x < ThisYear
    IsThisWeek, IsThisMonth, IsThisYear, IsNextMonth, IsNextYear, IsLastMonth, IsLastYear,
    DateDiffTicks, SecondsBetween, MillisecondsBetween, MinutesBetween, HoursBetween, DaysBetween,
    DateOf, MillisecondOf, SecondOf, MinuteOf, HourOf, DayOf, MonthOf, YearOf, DayOfTheWeek,
    AddTimeSpan, IncTick, IncMillisecond, IncSecond, IncMinute, IncHour, IncDay, IncMonth, IncYear)
```

## Description

Help is not yet available.

# TdxGenerator type

Reserved for future use.

## Unit

[dxEMF.Types](#)

## Syntax

```
type
  TdxGenerator = record
  private
    FGeneratorType: TdxGeneratorType;
  public
    constructor Create(AGeneratorType: TdxGeneratorType); overload;
    function NewGuid: TGUID;
    property GeneratorType: TdxGeneratorType read FGeneratorType;
  end;
```

# TdxGeneratorType type

Enumerates options for automatic generation of auto-incremented [primary key](#) values.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxGeneratorType = (None, Identity, GUID, SequentialGUID);
```

## Description

Options include:

Value	Description
<b>None</b>	No value generation. Manually assign a unique value to the key field before saving a newly created entity object.
<b>Identity</b>	Applies to Integer and Int64 type fields/properties. Depending on the data store type, automatic key generation is implemented using identity/autonumber/auto-increment columns or sequences.
<b>GUID</b>	Applies to TGUID type fields/properties. Generates a GUID.
<b>SequentialGUID</b>	Applies to TGUID type fields/properties. Generates a sequential GUID.

**Note:** **TdxGeneratorType** is a scoped enumeration type. To use its values in code, prefix them with the type name and a scope resolution operator (the `.` operation in Delphi). For example, use **TdxGeneratorType.Identity** to refer to the **Identity** value.

# TdxGroupOperatorType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxGroupOperatorType = (&And, &Or);
```

## Description

Help is not yet available.

# TdxJoinType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxJoinType = (Inner, LeftOuter);
```

## Description

Help is not yet available.

# TdxLoadingStrategy type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

```
type  
TdxLoadingStrategy = (Lazy, Eager);
```

## Description

Help is not yet available.

# TdxLockingOption type

Reserved for future use.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxLockingOption = (None, Optimistic);
```

# TdxMapInheritanceType type

Enumerates data storage options for entity class descendants marked by the [InheritanceAttribute](#).

## Unit

[dxEMF.Types](#)

## Syntax

```
type
  TdxMapInheritanceType = (ParentTable, OwnTable);
```

## Description

Values include:

Value	Description
<b>ParentTable</b>	A descendant class stores its data in the ancestor's table. This requires that you introduce a <a href="#">discriminator column</a> in the ancestor and provide discriminator column <a href="#">values</a> for each entity class in the inheritance tree.
<b>OwnTable</b>	A descendant class stores its data in a separate table and inherits its primary key from the ancestor. A foreign key constraint enforcing key integrity is created automatically in this table.

**Note:** **TdxMapInheritanceType** is a scoped enumeration type. To use its values in code, prefix them with the type name and a scope resolution operator (the `.` operation in Delphi). For example, use **TdxMapInheritanceType.ParentTable** to refer to the **ParentTable** value.

# TdxOptimisticLockingBehavior type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxOptimisticLockingBehavior = (NoLocking, ConsiderOptimisticLockingField, LockModified, I
```

## Description

Help is not yet available.

# TdxPredicate<T> type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxPredicate<T> = reference to function(AData: T): Boolean;
```

## Description

Help is not yet available.

# TdxPredicate type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxPredicate = reference to function(AObject: TObject): Boolean;
```

## Description

Help is not yet available.

# TdxSortingOrder type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxSortingOrder = (Ascending, Descending);
```

## Description

Help is not yet available.

# TdxUnaryOperatorType type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxUnaryOperatorType = (BitwiseNot, Plus, Minus, &Not, IsNull);
```

## Description

Help is not yet available.

# TdxUpdateSchemaResult type

Help is not yet available.

## Unit

[dxEMF.Types](#)

## Syntax

**type**

```
TdxUpdateSchemaResult = (SchemaExists, FirstTableNotExists);
```

## Description

Help is not yet available.

# dxEMF.Attributes Unit

## Classes

[AggregatedAttribute](#)

[AssociationAttribute](#)

[AutomappingAttribute](#)

[BlobAttribute](#)

[ColumnAttribute](#)

[DBTypeAttribute](#)

[DefaultAttribute](#)

[DiscriminatorAttribute](#)

[DiscriminatorColumnAttribute](#)

[EntityAttribute](#)

[GeneratorAttribute](#)

[IndexedAttribute](#)

[IndexesAttribute](#)

[InheritanceAttribute](#)

[KeyAttribute](#)

[NoForeignKeyAttribute](#)

[NonPersistentAttribute](#)

[NullableAttribute](#)

[ReadOnlyAttribute](#)

[SchemaNameAttribute](#)

[SizeAttribute](#)

[TableAttribute](#)

[UniqueAttribute](#)

# dxEMF.DataProvider.ADO Unit

## Classes

[EdxODBCError](#)

[TdxEMFADODataProvider](#)

# dxEMF.DataProvider.FireDAC Unit

## Classes

[TdxEMFFireDACDataProvider](#)

# dxEMF.DataSet Unit

## Classes

[TdxEMFCustomDataSet](#)

[TdxEMFDataSet](#)

[TdxEMFDataSetBlobStream](#)

# dxEMF.DB.Criteria Unit

## Classes

[TdxAggregateOperand](#)  
[TdxBetweenOperator](#)  
[TdxBinaryOperator](#)  
[TdxConstantValue](#)  
[TdxContainsOperator](#)  
[TdxCriteriaOperator](#)  
[TdxCriteriaOperatorCollection](#)  
[TdxFunctionOperator](#)  
[TdxGroupOperator](#)  
[TdxInOperator](#)  
[TdxJoinOperand](#)  
[TdxLikeCustomFunction](#)  
[TdxOperandParameter](#)  
[TdxOperandProperty](#)  
[TdxOperandValue](#)  
[TdxQueryOperand](#)  
[TdxSortByExpression](#)  
[TdxSortByExpressions](#)  
[TdxUnaryOperator](#)

## Interfaces

[IdxAggregateOperand](#)  
[IdxBetweenOperator](#)  
[IdxBinaryOperator](#)  
[IdxConstantValue](#)  
[IdxCriteriaOperator](#)  
[IdxFunctionOperator](#)  
[IdxGroupOperator](#)  
[IdxInOperator](#)  
[IdxJoinOperand](#)  
[IdxOperandParameter](#)  
[IdxOperandProperty](#)  
[IdxOperandValue](#)  
[IdxQueryOperand](#)  
[IdxUnaryOperator](#)

# dxEMF.Types Unit

## Classes

[TdxConnectionTypes](#)

[TdxDBEngines](#)

## Interfaces

[IdxCollection](#)

[IdxCollection<T>](#)

[IdxEMFCollection](#)

[IdxEMFCollection<T>](#)

[IdxExpression](#)

[IdxLinq](#)

[IdxList](#)

[IdxList<T>](#)

[IdxQueryable](#)

[IdxQueryable<T>](#)

[IdxQueryProvider](#)

[IdxQueryProvider<T>](#)

[IdxSession](#)

[IdxSortByExpression](#)

[IdxSortByExpressions](#)

## Other Types

[TdxAggregateFunctionType](#)

[TdxAttribute](#)

[TdxAttributes](#)

[TdxAttributeType](#)

[TdxAutoCreateOption](#)

[TdxBinaryOperatorType](#)

[TdxBooleanCriteriaState](#)

[TdxConnectionParameter](#)

[TdxConnectionType](#)

[TdxDBColumnType](#)

[TdxDBEngine](#)

[TdxDiscriminatorType](#)

[TdxFunctionCategory](#)

[TdxFunctionOperatorType](#)

[TdxFunc<TResult>](#)

[TdxGenerator](#)

[TdxGeneratorType](#)

[TdxGroupOperatorType](#)

[TdxJoinType](#)

[TdxLoadingStrategy](#)

[TdxLockingOption](#)

[TdxMapInheritanceType](#)

[TdxOptimisticLockingBehavior](#)

[TdxPredicate](#)

[TdxPredicate<T>](#)

[TdxSortingOrder](#)

[TdxUnaryOperatorType](#)

[TdxUpdateSchemaResult](#)

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IEnumerable<T>

|

IdxCollection<T>

---

# Hierarchy

IEnumerable

|

[IdxCollection](#)

# Hierarchy

---

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

# Hierarchy

---

IEnumerable<T>

|

[IdxCollection<T>](#)

|

[IdxList<T>](#)

|

[IdxEMFCollection<T>](#)

# Hierarchy

---

IEnumerable

|

[IdxCollection](#)

|

[IdxList](#)

|

[IdxEMFCollection](#)

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

[IdxLinq](#)

# Hierarchy

---

IEnumerable<T>

|

[IdxCollection<T>](#)

|

[IdxList<T>](#)

# Hierarchy

---

IEnumerable

|

[IdxCollection](#)

|

[IdxList](#)

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

[IdxOperandValue](#)

|

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

# Hierarchy

---

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IEnumerable<T>

|

[IdxQueryable<T>](#)

---

# Hierarchy

IEnumerable

|

[IdxQueryable](#)

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

IUnknown

|

IdxQueryProvider

|

IdxQueryProvider<T>

---

# Hierarchy

IUnknown

|

[IdxQueryProvider](#)

---

# Hierarchy

IUnknown

|

[IdxSession](#)

---

# Hierarchy

IEnumerable<[IdxSortByExpression](#)>

|

[IdxSortByExpressions](#)

---

# Hierarchy

IUnknown

|

[IdxSortByExpression](#)

---

# Hierarchy

IUnknown

|

[IdxExpression](#)

|

[IdxCriteriaOperator](#)

|

---

# Hierarchy

TCustomAttribute

|

[IndexedAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[IndexesAttribute](#)

# Hierarchy

TCustomAttribute

|

InheritanceAttribute

---

# Hierarchy

TCustomAttribute

|

[KeyAttribute](#)

---

# Legend

- Protected
- Published
- ▶ ReadOnly

---

# Hierarchy

TCustomAttribute

|

NoForeignKeyAttribute

---

# Hierarchy

TCustomAttribute

|

NonPersistentAttribute

---

# Hierarchy

TCustomAttribute

|

[NullableAttribute](#)

# TdxEMFCustomDataSet Events

[TdxEMFCustomDataSet](#) [Legend](#)

---

## Derived from TDataSet

- AfterCancel
- AfterClose
- AfterDelete
- AfterEdit
- AfterInsert
- AfterOpen
- AfterPost
- AfterRefresh
- AfterScroll
- BeforeCancel
- BeforeClose
- BeforeDelete
- BeforeEdit
- BeforeInsert
- BeforeOpen
- BeforePost
- BeforeRefresh
- BeforeScroll
- OnCalcFields
- OnDeleteError
- OnEditError
- OnFilterRecord
- OnNewRecord
- OnPostError

# TdxEMFDataSet Events

[TdxEMFDataSet](#) [Legend](#)

---

## In TdxEMFDataSet

- AfterCancel
- AfterClose
- AfterDelete
- AfterEdit
- AfterInsert
- AfterOpen
- AfterPost
- AfterRefresh
- AfterScroll
- BeforeCancel
- BeforeClose
- BeforeDelete
- BeforeEdit
- BeforeInsert
- BeforeOpen
- BeforePost
- BeforeRefresh
- BeforeScroll
- OnCalcFields
- OnDeleteError
- OnEditError
- OnFilterRecord
- OnNewRecord
- OnPostError

# TdxSortByExpressions Events

[TdxSortByExpressions](#) [Legend](#)

---

## In TdxSortByExpressions

[OnChanged](#)

# TdxAggregateOperand Methods

[TdxAggregateOperand](#) [Legend](#)

---

## In TdxAggregateOperand

[Accept](#)

[Avg](#)

[Count](#)

[Create](#)

[Equals](#)

[Exists](#)

[GetHashCode](#)

[Max](#)

[Min](#)

[Single](#)

[Sum](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxBetweenOperator Methods

[TdxBetweenOperator](#) [Legend](#)

---

## In TdxBetweenOperator

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxBinaryOperator Methods

[TdxBinaryOperator](#) [Legend](#)

---

## In TdxBinaryOperator

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxConstantValue Methods

[TdxConstantValue](#) [Legend](#)

---

Derived from [TdxOperandValue](#)

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxContainsOperator Methods

[TdxContainsOperator](#) [Legend](#)

---

## In TdxContainsOperator

[Create](#)

## Derived from [TdxAggregateOperand](#)

[Accept](#)

[Avg](#)

[Count](#)

[Create](#)

[Equals](#)

[Exists](#)

[GetHashCode](#)

[Max](#)

[Min](#)

[Single](#)

[Sum](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxCriteriaOperator Methods

[TdxCriteriaOperator](#) [Legend](#)

---

## In TdxCriteriaOperator

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxCriteriaOperatorCollection Methods

[TdxCriteriaOperatorCollection](#) [Legend](#)

---

## In TdxCriteriaOperatorCollection

[Equals](#)

[GetHashCode](#)

[ToString](#)

# TdxEMFCollections Methods

[TdxEMFCollections](#) [Legend](#)

---

## In TdxEMFCollections

[Create](#)

[Initialize](#)

# TdxEMFCustomCollection Methods

[TdxEMFCustomCollection](#) [Legend](#)

---

## In TdxEMFCustomCollection

[Init](#)

[Insert](#)

[Load](#)

[Reload](#)

[ToArray](#)

# TdxEMFCustomDataSet Methods

[TdxEMFCustomDataSet](#) [Legend](#)

---

## In TdxEMFCustomDataSet

[BookmarkValid](#)

[CompareBookmarks](#)

[CreateBlobStream](#)

[Current](#)

[FlushChanges](#)

[GetFieldData](#)

[Locate](#)

# TdxEMFCustomSession Methods

[TdxEMFCustomSession](#) [Legend](#)

---

## In TdxEMFCustomSession

[Attach](#)

[BeginTrackingChanges](#)

[CreateObject](#)

[CustomCreate](#)

[Delete](#)

[Detach](#)

[DropChanges](#)

[Find](#)

[FlushChanges](#)

[GetDataContext](#)

[GetEntityInfo](#)

[GetObjects](#)

[GetObjectsToDelete](#)

[GetObjectsToSave](#)

[GetQueryProvider](#)

[IsNewObject](#)

[IsObjectToDelete](#)

[Reload](#)

[Save](#)

# TdxEMFDataSet Methods

[TdxEMFDataSet](#) [Legend](#)

---

## In TdxEMFDataSet

[AssignData](#)

## Derived from [TdxEMFCustomDataSet](#)

[BookmarkValid](#)

[CompareBookmarks](#)

[CreateBlobStream](#)

[Current](#)

[FlushChanges](#)

[GetFieldData](#)

[Locate](#)

# TdxEMFDataSetBlobStream Methods

[TdxEMFDataSetBlobStream](#) [Legend](#)

---

## In TdxEMFDataSetBlobStream

[Create](#)

[Destroy](#)

[Write](#)

# TdxEMFSession Methods

[TdxEMFSession](#) [Legend](#)

---

## In TdxEMFSession

[CreateSchema](#)

## Derived from [TdxEMFCustomSession](#)

[Attach](#)

[BeginTrackingChanges](#)

[CreateObject](#)

[CustomCreate](#)

[Delete](#)

[Detach](#)

[DropChanges](#)

[Find](#)

[FlushChanges](#)

[GetDataContext](#)

[GetEntityInfo](#)

[GetObjects](#)

[GetObjectsToDelete](#)

[GetObjectsToSave](#)

[GetQueryProvider](#)

[IsNewObject](#)

[IsObjectToDelete](#)

[Reload](#)

[Save](#)

# TdxFunctionOperator Methods

[TdxFunctionOperator](#) [Legend](#)

---

## In TdxFunctionOperator

[Accept](#)

[Create](#)

[Destroy](#)

[Equals](#)

[GetHashCode](#)

[GuessIsLogicalCustomFunction](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxGroupOperator Methods

[TdxGroupOperator](#) [Legend](#)

---

## In TdxGroupOperator

[Accept](#)

[Combine](#)

[Create](#)

[Destroy](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxInOperator Methods

[TdxInOperator](#) [Legend](#)

---

## In TdxInOperator

[Accept](#)

[Create](#)

[Destroy](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxJoinOperand Methods

[TdxJoinOperand](#) [Legend](#)

---

## In TdxJoinOperand

[Accept](#)

[Avg](#)

[Count](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

[Max](#)

[Min](#)

[Sum](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxLikeCustomFunction Methods

[TdxLikeCustomFunction](#) [Legend](#)

---

## In TdxLikeCustomFunction

[Convert](#)

[Create](#)

[IsBinaryCompatibleLikeFunction](#)

[IsName](#)

# TdxOperandParameter Methods

[TdxOperandParameter](#) [Legend](#)

---

## In TdxOperandParameter

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxOperandValue](#)

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxOperandProperty Methods

[TdxOperandProperty](#) [Legend](#)

---

## In TdxOperandProperty

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxOperandValue Methods

[TdxOperandValue](#) [Legend](#)

---

## In TdxOperandValue

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxQueryOperand Methods

[TdxQueryOperand](#) [Legend](#)

---

## In TdxQueryOperand

[Accept](#)

[Clone](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxSortByExpression Methods

[TdxSortByExpression](#) [Legend](#)

---

In **TdxSortByExpression**

[Create](#)

# TdxSortByExpressions Methods

[TdxSortByExpressions](#) [Legend](#)

---

## In TdxSortByExpressions

[Add](#)

[AddRange](#)

[Clear](#)

[Contains](#)

[Create](#)

[Destroy](#)

[IndexOf](#)

[Insert](#)

[Remove](#)

[RemoveAt](#)

# TdxUnaryOperator Methods

[TdxUnaryOperator](#) [Legend](#)

---

## In TdxUnaryOperator

[Accept](#)

[Create](#)

[Equals](#)

[GetHashCode](#)

## Derived from [TdxCriteriaOperator](#)

[Accept](#)

[And](#)

[CriterionEquals](#)

[Equals](#)

[Not](#)

[Or](#)

[Parse](#)

[ParseList](#)

[ToString](#)

# TdxAggregateOperand Properties

[TdxAggregateOperand](#) [Legend](#)

---

## In TdxAggregateOperand

- [AggregatedExpression](#)
- [AggregateFunctionType](#)
- [CollectionProperty](#)
- [Condition](#)
- ▶ [IsTopLevel](#)

# TdxBetweenOperator Properties

[TdxBetweenOperator](#) [Legend](#)

---

## In TdxBetweenOperator

[BeginExpression](#)

[EndExpression](#)

[TestExpression](#)

# TdxBinaryOperator Properties

[TdxBinaryOperator](#) [Legend](#)

---

## In TdxBinaryOperator

[LeftOperand](#)

▸ [OperatorType](#)

[RightOperand](#)

# TdxConstantValue Properties

[TdxConstantValue](#) [Legend](#)

---

Derived from [TdxOperandValue](#)

[Value](#)

# TdxContainsOperator Properties

[TdxContainsOperator](#) [Legend](#)

---

**Derived from** [TdxAggregateOperand](#)

[AggregatedExpression](#)

[AggregateFunctionType](#)

[CollectionProperty](#)

[Condition](#)

▸ [IsTopLevel](#)

# TdxEMFADODataProvider Properties

[TdxEMFADODataProvider](#) [Legend](#)

---

## In TdxEMFADODataProvider

- [Connection](#)

## Derived from [TdxEMFCustomDataProvider](#)

- ▶ [ActualDBEngineType](#)
- ▶ [CanCreateDatabase](#)
- ▶ [CanCreateSchema](#)
- [Options](#)

# TdxEMFCollectionOptions Properties

[TdxEMFCollectionOptions](#) [Legend](#)

---

## In TdxEMFCollectionOptions

- [LoadingStrategy](#)

# TdxEMFCustomCollection Properties

[TdxEMFCustomCollection](#) [Legend](#)

---

## In TdxEMFCustomCollection

- ▶ [CollectionElementClass](#)
  - [Criteria](#)
  - [Loader](#)
  - [SelectDeleted](#)
- ▶ [Session](#)
  - [SkipReturnedObjects](#)
  - [TopReturnedObjects](#)

# TdxEMFCustomDataProvider Properties

[TdxEMFCustomDataProvider](#) [Legend](#)

---

## In TdxEMFCustomDataProvider

- ▶ [ActualDBEngineType](#)
- ▶ [CanCreateDatabase](#)
- ▶ [CanCreateSchema](#)
- [Options](#)

# TdxEMFCustomDataSet Properties

[TdxEMFCustomDataSet](#) [Legend](#)

---

## In TdxEMFCustomDataSet

[ReadOnly](#)

[Session](#)

## Derived from TDataSet

Active

Filter

Filtered

# TdxEMFCustomSession Properties

[TdxEMFCustomSession](#) [Legend](#)

---

## In TdxEMFCustomSession

- [DataProvider](#)

  - [Options](#)

# TdxEMFDataProviderOptions Properties

[TdxEMFDataProviderOptions](#) [Legend](#)

---

## In TdxEMFDataProviderOptions

- [AutoCreate](#)
- [DBEngine](#)

# TdxEMFDataSet Properties

[TdxEMFDataSet](#) [Legend](#)

---

**Derived from** [TdxEMFCustomDataSet](#)

- [ReadOnly](#)  
[Session](#)

**Derived from** TDataSet

- Active
- Filter
- Filtered

# TdxEMFFireDACDataProvider Properties

[TdxEMFFireDACDataProvider](#) [Legend](#)

---

## In TdxEMFFireDACDataProvider

- [Connection](#)

## Derived from [TdxEMFCustomDataProvider](#)

- ▶ [ActualDBEngineType](#)
- ▶ [CanCreateDatabase](#)
- ▶ [CanCreateSchema](#)
- [Options](#)

# TdxEMFSession Properties

[TdxEMFSession](#) [Legend](#)

---

Derived from [TdxEMFCustomSession](#)

- [DataProvider](#)
- [Options](#)

# TdxEMFSessionOptions Properties

[TdxEMFSessionOptions](#) [Legend](#)

---

## In TdxEMFSessionOptions

- [Collections](#)
- [LockingOption](#)

# TdxFunctionOperator Properties

[TdxFunctionOperator](#) [Legend](#)

---

## In TdxFunctionOperator

- ▶ [Operands](#)
- [OperatorType](#)

# TdxGroupOperator Properties

[TdxGroupOperator](#) [Legend](#)

---

## In TdxGroupOperator

- ▶ [Operands](#)
- [OperatorType](#)

# TdxInOperator Properties

[TdxInOperator](#) [Legend](#)

---

## In TdxInOperator

- ▶ [LeftOperand](#)
- ▶ [Operands](#)

# TdxJoinOperand Properties

[TdxJoinOperand](#) [Legend](#)

---

## In TdxJoinOperand

[AggregatedExpression](#)

[AggregateFunctionType](#)

[Condition](#)

[JoinTypeName](#)

# TdxOperandParameter Properties

[TdxOperandParameter](#) [Legend](#)

---

## In TdxOperandParameter

[ParameterName](#)

## Derived from [TdxOperandValue](#)

[Value](#)

# TdxOperandProperty Properties

[TdxOperandProperty](#) [Legend](#)

---

In **TdxOperandProperty**

[PropertyName](#)

# TdxOperandValue Properties

[TdxOperandValue](#) [Legend](#)

---

## In TdxOperandValue

[Value](#)

# TdxQueryOperand Properties

[TdxQueryOperand](#) [Legend](#)

---

## In TdxQueryOperand

- ▶ [ColumnName](#)
- ▶ [ColumnType](#)
- [NodeAlias](#)

# TdxSortByExpression Properties

[TdxSortByExpression](#) [Legend](#)

---

## In TdxSortByExpression

- ▶ [Property](#)
  - [PropertyName](#)
  - [SortingOrder](#)

# TdxSortByExpressions Properties

[TdxSortByExpressions](#) [Legend](#)

---

## In TdxSortByExpressions

- ▶ [Count](#)
- ▶ [Items](#)

# TdxUnaryOperator Properties

[TdxUnaryOperator](#) [Legend](#)

---

## In TdxUnaryOperator

[Operand](#)

- ▶ [OperatorType](#)

# IdxAggregateOperand Methods

[IdxAggregateOperand](#) [Legend](#)

---

## In IdxAggregateOperand

[GetAggregatedExpression](#)

[GetAggregateFunctionType](#)

[GetCondition](#)

[GetIsTopLevel](#)

[GetProperty](#)

# IdxBetweenOperator Methods

[IdxBetweenOperator](#) [Legend](#)

---

## In IdxBetweenOperator

[GetBeginExpression](#)

[GetEndExpression](#)

[GetTestExpression](#)

# IdxBinaryOperator Methods

[IdxBinaryOperator](#) [Legend](#)

---

## In IdxBinaryOperator

[GetLeftOperand](#)

[GetOperatorType](#)

[GetRightOperand](#)

# IdxCollection<T> Methods

[IdxCollection<T>](#) [Legend](#)

---

## In IdxCollection<T>

[Add](#)

[Contains](#)

[GetCount](#)

[Remove](#)

# IdxCollection Methods

[IdxCollection](#) [Legend](#)

---

## In IdxCollection

[GetCount](#)

# IdxEMFCollection<T> Methods

[IdxEMFCollection<T>](#) [Legend](#)

---

## In [IdxEMFCollection<T>](#)

[Find](#)

[First](#)

[GetDeleteObjectOnRemove](#)

[GetObjects](#)

[Last](#)

[SetDeleteObjectOnRemove](#)

## Derived from [IdxList<T>](#)

[GetItems](#)

[IndexOf](#)

## Derived from [IdxCollection<T>](#)

[Add](#)

[Contains](#)

[GetCount](#)

[Remove](#)

# IdxEMFCollection Methods

[IdxEMFCollection](#) [Legend](#)

---

## In [IdxEMFCollection](#)

[First](#)

[GetDeleteObjectOnRemove](#)

[Last](#)

[SetDeleteObjectOnRemove](#)

## Derived from [IdxList](#)

[Add](#)

[Contains](#)

[GetItems](#)

[IndexOf](#)

[Remove](#)

## Derived from [IdxCollection](#)

[GetCount](#)

# IdxFunctionOperator Methods

[IdxFunctionOperator](#) [Legend](#)

---

## In IdxFunctionOperator

[GetOperands](#)

[GetOperatorType](#)

# IdxGroupOperator Methods

[IdxGroupOperator](#) [Legend](#)

---

## In IdxGroupOperator

[GetOperands](#)

[GetOperatorType](#)

# IdxInOperator Methods

[IdxInOperator](#) [Legend](#)

---

## In IdxInOperator

[GetLeftOperand](#)

[GetOperands](#)

# IdxJoinOperand Methods

[IdxJoinOperand](#) [Legend](#)

---

## In IdxJoinOperand

[GetAggregatedExpression](#)

[GetAggregateFunctionType](#)

[GetCondition](#)

[GetJoinTypeName](#)

# IdxList<T> Methods

[IdxList<T>](#) [Legend](#)

---

## In `IdxList<T>`

[GetItems](#)

[IndexOf](#)

## Derived from [IdxCollection<T>](#)

[Add](#)

[Contains](#)

[GetCount](#)

[Remove](#)

# IdxList Methods

[IdxList](#) [Legend](#)

---

## In IdxList

[Add](#)

[Contains](#)

[GetItems](#)

[IndexOf](#)

[Remove](#)

## Derived from [IdxCollection](#)

[GetCount](#)

# IdxOperandParameter Methods

[IdxOperandParameter](#) [Legend](#)

---

## In IdxOperandParameter

[GetParameterName](#)

## Derived from [IdxOperandValue](#)

[GetValue](#)

[SetValue](#)

# IdxOperandProperty Methods

[IdxOperandProperty](#) [Legend](#)

---

In **IdxOperandProperty**

[GetPropertyName](#)

# IdxOperandValue Methods

[IdxOperandValue](#) [Legend](#)

---

## In IdxOperandValue

[GetValue](#)

[SetValue](#)

# IdxQueryable<T> Methods

[IdxQueryable<T>](#) [Legend](#)

---

In **IdxQueryable<T>**

[GetProvider](#)

# IdxQueryable Methods

[IdxQueryable](#) [Legend](#)

---

## In IdxQueryable

[GetProvider](#)

# IdxQueryOperand Methods

[IdxQueryOperand](#) [Legend](#)

---

## In IdxQueryOperand

[GetColumnName](#)

[GetColumnType](#)

[GetNodeAlias](#)

# IdxQueryProvider<T> Methods

[IdxQueryProvider<T>](#) [Legend](#)

---

## In IdxQueryProvider<T>

[CreateQuery](#)

## Derived from [IdxQueryProvider](#)

[CreateQuery](#)

# IdxQueryProvider Methods

[IdxQueryProvider](#) [Legend](#)

---

In **IdxQueryProvider**

[CreateQuery](#)

# IdxSession Methods

[IdxSession](#) [Legend](#)

---

## In IdxSession

[GetLoadingStrategy](#)

# IdxSortByExpression Methods

[IdxSortByExpression](#) [Legend](#)

---

## In IdxSortByExpression

[GetExpression](#)

[GetSortingOrder](#)

# IdxUnaryOperator Methods

[IdxUnaryOperator](#) [Legend](#)

---

## In IdxUnaryOperator

[GetOperand](#)

[GetOperatorType](#)

# IdxAggregateOperand Properties

[IdxAggregateOperand](#) [Legend](#)

---

## In IdxAggregateOperand

- ▶ [AggregatedExpression](#)
- ▶ [AggregateFunctionType](#)
- ▶ [CollectionProperty](#)
- ▶ [Condition](#)
- ▶ [IsTopLevel](#)

# IdxBetweenOperator Properties

[IdxBetweenOperator](#) [Legend](#)

---

## In IdxBetweenOperator

- ▶ [BeginExpression](#)
- ▶ [EndExpression](#)
- ▶ [TestExpression](#)

# IdxBinaryOperator Properties

[IdxBinaryOperator](#) [Legend](#)

---

## In IdxBinaryOperator

- ▶ [LeftOperand](#)
- ▶ [OperatorType](#)
- ▶ [RightOperand](#)

# IdxCollection<T> Properties

[IdxCollection<T>](#) [Legend](#)

---

In **IdxCollection<T>**

- ▶ [Count](#)

# IdxCollection Properties

[IdxCollection](#) [Legend](#)

---

## In IdxCollection

- ▶ [Count](#)

# IdxEMFCollection<T> Properties

[IdxEMFCollection<T>](#) [Legend](#)

---

In **IdxEMFCollection<T>**

[DeleteObjectOnRemove](#)

Derived from [IdxList<T>](#)

▸ [Items](#)

Derived from [IdxCollection<T>](#)

▸ [Count](#)

# IdxEMFCollection Properties

[IdxEMFCollection](#) [Legend](#)

---

## In IdxEMFCollection

[DeleteObjectOnRemove](#)

## Derived from [IdxList](#)

▸ [Items](#)

## Derived from [IdxCollection](#)

▸ [Count](#)

# IdxFunctionOperator Properties

[IdxFunctionOperator](#) [Legend](#)

---

## In IdxFunctionOperator

- ▶ [Operands](#)
- ▶ [OperatorType](#)

# IdxGroupOperator Properties

[IdxGroupOperator](#) [Legend](#)

---

## In IdxGroupOperator

- ▶ [Operands](#)
- ▶ [OperatorType](#)

# IdxInOperator Properties

[IdxInOperator](#) [Legend](#)

---

## In IdxInOperator

- ▶ [LeftOperand](#)
- ▶ [Operands](#)

# IdxJoinOperand Properties

[IdxJoinOperand](#) [Legend](#)

---

## In IdxJoinOperand

- ▶ [AggregatedExpression](#)
- ▶ [AggregateFunctionType](#)
- ▶ [Condition](#)
- ▶ [JoinTypeName](#)

# IdxList<T> Properties

[IdxList<T>](#) [Legend](#)

---

## In `IdxList<T>`

- ▶ [Items](#)

## Derived from [IdxCollection<T>](#)

- ▶ [Count](#)

# IdxList Properties

[IdxList](#) [Legend](#)

---

## In IdxList

- ▶ [Items](#)

## Derived from [IdxCollection](#)

- ▶ [Count](#)

# IdxOperandParameter Properties

[IdxOperandParameter](#) [Legend](#)

---

## In IdxOperandParameter

- ▶ [ParameterName](#)

## Derived from [IdxOperandValue](#)

[Value](#)

# IdxOperandProperty Properties

[IdxOperandProperty](#) [Legend](#)

---

## In IdxOperandProperty

- ▶ [PropertyName](#)

# IdxOperandValue Properties

[IdxOperandValue](#) [Legend](#)

---

In **IdxOperandValue**

[Value](#)

# IdxQueryable<T> Properties

[IdxQueryable<T>](#) [Legend](#)

---

## In IdxQueryable<T>

- ▶ [Provider](#)

# IdxQueryable Properties

[IdxQueryable](#) [Legend](#)

---

## In IdxQueryable

- ▶ [Provider](#)

# IdxQueryOperand Properties

[IdxQueryOperand](#) [Legend](#)

---

## In IdxQueryOperand

- ▶ [ColumnName](#)
- ▶ [ColumnType](#)
- ▶ [NodeAlias](#)

# IdxSession Properties

[IdxSession](#) [Legend](#)

---

## In IdxSession

- ▶ [LoadingStrategy](#)

# IdxSortByExpression Properties

[IdxSortByExpression](#) [Legend](#)

---

## In IdxSortByExpression

- ▶ [Expression](#)
- ▶ [SortingOrder](#)

# IdxUnaryOperator Properties

[IdxUnaryOperator](#) [Legend](#)

---

## In IdxUnaryOperator

- ▶ [Operand](#)
- ▶ [OperatorType](#)

---

# Hierarchy

TCustomAttribute

|

[ReadOnlyAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[SchemaNameAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[SizeAttribute](#)

---

# Hierarchy

TCustomAttribute

|

[TableAttribute](#)

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

[TdxAggregateOperand](#)

# Hierarchy

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxBetweenOperator

---

# Hierarchy

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxBinaryOperator

---

# Hierarchy

TObject

|

[TdxConnectionTypes](#)

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

[TdxOperandValue](#)

|

TdxConstantValue

# Hierarchy

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

[TdxAggregateOperand](#)

|

TdxContainsOperator

---

# Hierarchy

TList<[IdxCriteriaOperator](#)>

|

TdxCriteriaOperatorCollection

---

# Hierarchy

TInterfacedObject

|

[TdxCriteriaOperator](#)

---

# Hierarchy

TObject

|

[TdxDBEngines](#)

# Hierarchy

---

TComponent

|

[TdxEMFCustomDataProvider](#)

|

[TdxEMFADODDataProvider](#)

---

# Hierarchy

TPersistent

|

[TdxEMFCollectionOptions](#)

---

# Hierarchy

TObject

|

[TdxEMFCollections](#)

---

# Hierarchy

TInterfacedObject

|

[TdxEMFCustomCollection](#)

---

# Hierarchy

TComponent

|

[TdxEMFCustomDataProvider](#)

---

# Hierarchy

TDataSet

|

[TdxEMFCustomDataSet](#)

---

# Hierarchy

TComponent

|

[TdxEMFCustomSession](#)

---

# Hierarchy

TPersistent

|

[TdxEMFDataProviderOptions](#)

---

# Hierarchy

TMemoryStream

|

[TdxEMFDataSetBlobStream](#)

# Hierarchy

TDataSet

|

[TdxEMFCustomDataSet](#)

|

[TdxEMFDataSet](#)

# Hierarchy

---

TComponent

|

[TdxEMFCustomDataProvider](#)

|

[TdxEMFFireDACDataProvider](#)

---

# Hierarchy

TPersistent

|

[TdxEMFSessionOptions](#)

---

# Hierarchy

TComponent

|

[TdxEMFCustomSession](#)

|

[TdxEMFSession](#)

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxFunctionOperator

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxGroupOperator

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxInOperator

---

# Hierarchy

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxJoinOperand

---

# Hierarchy

TObject

|

TdxLikeCustomFunction

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

[TdxOperandValue](#)

|

TdxOperandParameter

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxOperandProperty

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxOperandValue

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxQueryOperand

---

# Hierarchy

TInterfacedObject

|

TdxSortByExpressions

---

# Hierarchy

TInterfacedObject

|

TdxSortByExpression

# Hierarchy

---

TInterfacedObject

|

[TdxCriteriaOperator](#)

|

TdxUnaryOperator

---

# Entity Model

An entity model defines a relational data store's schema in terms of entity classes and relationships between them. Entity classes are structured types that represent any real-world entities (persons, places, or things) in code. An entity model includes declarations of these classes and information on how they map to data store objects (also called mapping information).

An entity model allows you to:

- | Apply entity class declarations as conventional and validation rules when writing your entity-based code that [manages](#) or [queries](#) a data store's relational data at design time;
- | [Replicate](#) a data store's schema and use it with a [session component](#) to update the data store and validate its integrity at runtime.

Once added to a project, an entity model is considered a required part, as it acts as the contract between your entity-based code and data stores whose schema matches this entity model. Refer to this topic to learn how to [design](#) entity models using the **ExpressEntityMapping Framework**.

---

# Hierarchy

TCustomAttribute

|

[UniqueAttribute](#)