Microsoft Dynamics® AX 2012

Migrating Extended Data Type Relations in Microsoft Dynamics AX 2012

White Paper

This document describes how developers can migrate extended data type (EDT) relations to table relations in the Application Object Tree (AOT). The migration can be performed manually or with the EDT relation migration tool. This action is required in certain cases where a data model has been changed in Microsoft Dynamics AX 2012 or when the EDT relation must be modified.

Date: April 2011

Authors: Vasudev Burle, Software Development Engineer in Test II; Duc Luong, Software Development Engineer

Send suggestions and comments about this document to adocs@microsoft.com. Please include the title with your feedback.

Pre-release

Microsoft Dynamics

Table of Contents

Pre-release

Introduction

Changes to a data model in Microsoft Dynamics® AX 2012 sometimes require developers to migrate table relationships currently defined in the **Relations** node under extended data type (EDT) nodes to the **Relations** node under the relevant table nodes in the Application Object Tree (AOT).

In previous versions of Microsoft Dynamics AX, table relationships could be defined in the AOT under the **Relations** node of either a specific table node or an EDT node. Table relationships under an EDT have some disadvantages:

- They do not contain the rich relationship metadata, such as *cardinality* and *relation type*, that can be included in relations under a table node.
- They can only capture single field relationships, which might not represent the intended—and possibly more intricate—relationship between the tables.

A significant difficulty with having table relations defined under both an EDT and a table is that the order of relations matters when table relationships are defined in both locations. In such cases, the kernel will use different algorithms to decide which relationship to examine first, depending on the context.

Starting with Microsoft Dynamics AX 2012, all table relationships will be defined on tables only. Developers are not required to move existing EDT relations. However, if they wish to modify these relations, then the EDT relations must be migrated to the tables.

Note In Microsoft Dynamics AX 2012, developers are not permitted to create new EDT relations.

EDT relations in Microsoft Dynamics AX 2009 and previous versions were used primarily for two purposes:

- Some data access actions (delete actions, querying using implicit relations, renaming primary keys, and so on). In previous versions of Microsoft Dynamics AX, both the EDT **Relations** node and the table **Relations** node were always examined. In Microsoft Dynamics AX 2012, EDT relations are examined only if they have not been migrated.
- Dynamic-link and lookup actions for unbound controls.

Migrating table relations from an EDT **Relations** node to a table's **Relations** node in Microsoft Dynamics AX 2012 preserves current behavior.

To facilitate migration and preserve behavior, a new property, **Reference Table**, has been added to the EDT structure. Each EDT node in the AOT now contains a new **Table References** node that stores lookup information.

Note Changes in application behavior can be introduced when EDT relations are migrated and resulting multiple table relationships have been marked incorrectly. This possibility is related only to relations used for join conditions, such as a query that uses the implicit join condition. This behavior can be corrected by using the explicit join relation in the **Data Sources** node of the specific query node in the AOT.

Audience

This paper targets developers who need to update their table relations as a result of changes to a data model or the need to modify existing EDT relations.

Terminology

The following table lists Microsoft Dynamics AX 2012 terms relevant to migrating EDT relations.

Term	Definition
EDT relation	An entry under the **Relations** node of an EDT node in the AOT that defines the relationship between an extended data type and a table.

Pre-release

Hosting table	A table that contains the **Relations** node to which the EDT relation will be migrated.
Marker	A property on an object in the AOT that is used to track the status of an EDT relation.
Table relation	An entry under the **Relations** node of a table node in the AOT that describes a relationship between two tables.
Unique index	An index in which the key contains no duplicated values. A unique index may be a primary key or alternate key or is sometimes created only for enforcing uniqueness of the data.

EDT relation markers

The properties in the following table are used to track the EDT relation status.

Node	Property	Function
Table > Relations > *field*	EDTRelation	Indicates whether the relation has been directly migrated from an EDT.
Table > Relations > *field*	SourceEDT	Specifies the EDT relation migrated.
Table > Fields	IgnoreEDTRelation	Do not evaluate the EDT relation.
EDT	Reference Table	Stores lookup information.

Migration process

You can migrate EDT relations manually, or by using the new EDT relation migration tool. You must perform a manual migration when an EDT relation cannot be directly migrated with the tool because you need to correct the data model.

Manual migration process

To conduct a manual migration, follow these steps:

1. Examine your data model to verify that the referenced field on the EDT relation is either a primary key (PK) or an alternate key (AK), or part of a primary or alternate key. If it is not, you have two options:
   a. Revise the data model to make this value a PK or AK (or part of one) and continue to step 2.

      Note Choose this option if the field is part of a valid relation, but the key is not properly marked.

   b. Set the value of the new **IgnoreEDTRelation** property on the hosting tables to "Yes" and end the manual migration.

      Note Choose this option if the field is not part of a valid relation and will only be used for unbound lookup.
2. Check out the hosting tables in the AOT from source control.
3. Copy the relation from the EDT to all hosting tables. You can create a new relation or annotate existing relations.
   a. If possible, make the relation a foreign key.
   b. Otherwise, make the relation a normal relation or fixed relation.
4. Set the EDT migration properties (markers) to reflect migration status.

Pre-release

5. Add relation metadata such as *cardinality* or *relation type* to the relation in the hosting tables.
6. Move the EDT relation to the new **Table References** node under the EDT node.
7. Test the effectiveness of the relation after it has been migrated, to make sure that the migration will not cause any behavioral changes.
    a. If multiple relationships with another table are not present, perform a limited test, after determining the relevant test cases.
    b. If multiple relationships are present, perform a targeted test.
        i. Determine which relationships to test.
        ii. Based on the test results, explicitly set the join relations in the **Data Source** node of the affected object as needed.
8. Check in the changes to the tables that hold the migrated relations.

EDT relation migration tool

The EDT relation migration tool can be used to automate the migration process. The tool can perform the following actions:

- Copy an EDT relation to all hosting tables.
- Automatically set the EDT migration properties (markers) to reflect migration status.
- Automatically populate relation metadata.
- Derive *cardinality* from the index on the foreign key.
- Derive the *relationship type* from the delete action/key composition.
- Determine role names.
- Report AOT objects impacted by the migration, depending on the relation used. The objects that can be affected include:
    - Queries
    - Forms
    - Delete actions on tables
    - Data sets
    - X++ reports

Pre-release

To begin using the EDT relation migration tool, open the form for the tool by using the navigation path **Tools** > **Code upgrade** > **EDT relation migration tool**.

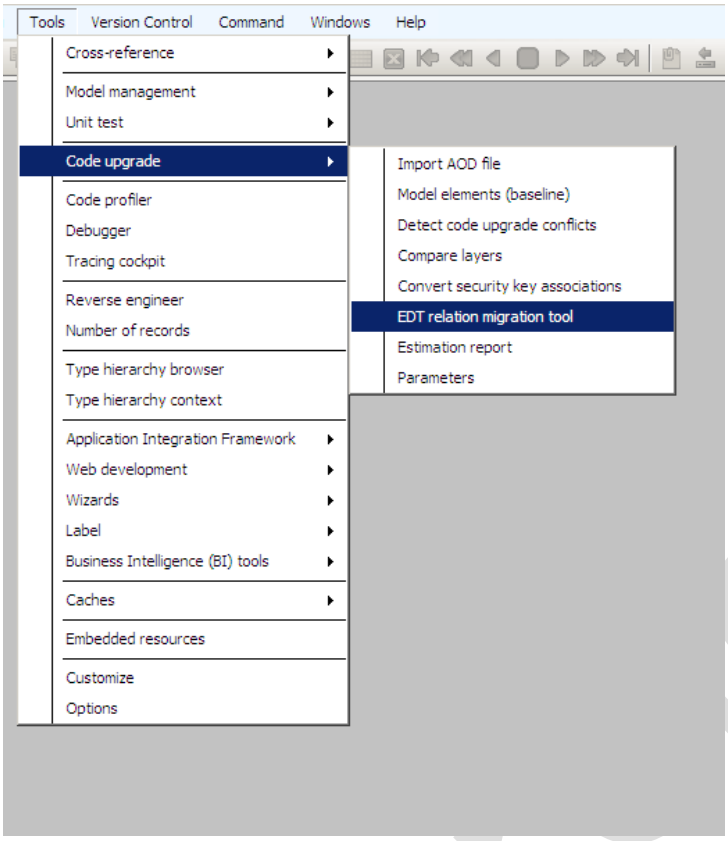

Figure 1: Path to EDT relation migration tool

The first time you open the tool, it will ask whether you want to refresh the EDT relations data. Select **Yes** to perform this action, which takes about 5–10 minutes to complete.

Pre-release

The EDT relation migration tool is shown in Figure 2.

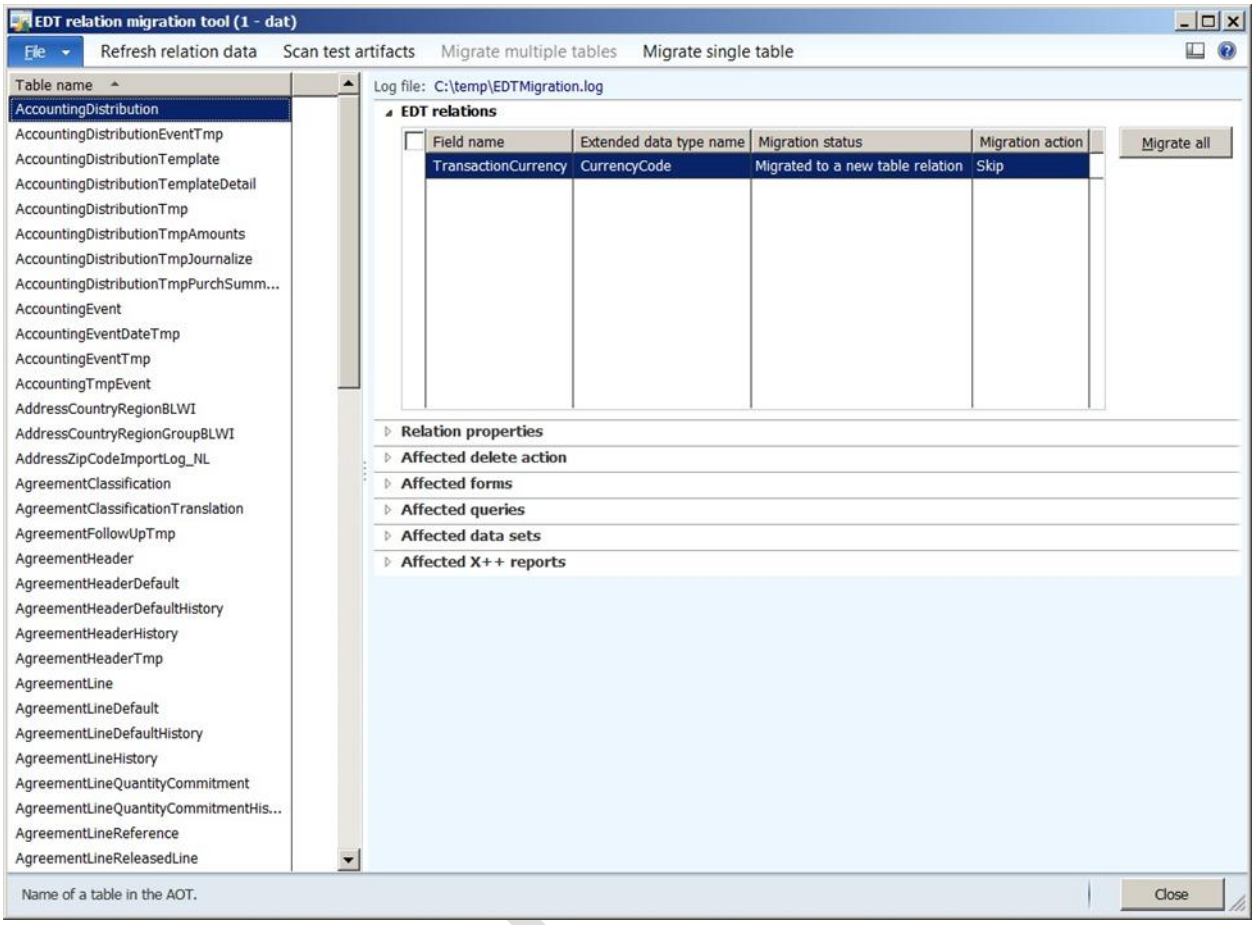

Figure 2: EDT relation migration tool

Key fields on the EDT relation migration tool

The **Table nam**e pane on the left side of the form displays all table names in the AOT.

The **EDT relations** FastTab displays all EDT relations for the selected tables in the left pane of the form.

Index type indicates whether the EDT field on the referenced table is part of any primary key (PK), alternate key (AK), unique index, or composite index.

Migration status indicates the current migration state. This is a read-only column that indicates the current status of the EDT relation. There are four options:

- Migrated
- NotMigrated
- MigratedToIgnoreEDTRelation

  Note An EDT relation is marked as "MigratedToIgnoreEDTRelation" when the relation in that EDT is invalid for the current table.
- MigratedToExistingTableRelation

Pre-

Note As part of the migration, the migration tool looks at existing table relations to determine whether the EDT relation matches an existing table relation. If it does, the tool marks the existing table relation instead of creating a new relation.

Migration action provides a drop-down menu for choosing one of the following actions:

- **Skip:** Skip the migration.
- **Migrate:** Migrate the relation to the table relation.
- **Mark as ignore EDT relation:** Set the value of the new **IgnoreEDTRelation** property on the table field to "Yes." The default value for this property is "No."

Process of migrating EDT relations on a single table

The process for migrating the EDT relations for a single table includes the following steps:

1. Select a table from the **Table name** pane.
2. Select each relation in the **EDT relations** table and choose an action from the **Migration action** drop-down menu for each of them.
3. After you have set an action on all the relations for that table, click the **Migrate single table** button on the ribbon at the top of the form.

Note The migration tool attempts to find a match for the EDT relation in the existing relations in the selected table. If a match is found, the SourceEDT property on the table relation is set to the name of the EDT.

If no match is found in the existing relations in the table, the tool will create a new table relation only if the index on the referenced table (shown by "IndexType") is "PK" or "AK." If the matching index for the EDT field on the referenced table is "NoIndex," "Unique," or "NonUnique," the tool will not create a new table relation.

Figure 3 shows migration actions being set on the EDT relations for a single table.

Pre-release

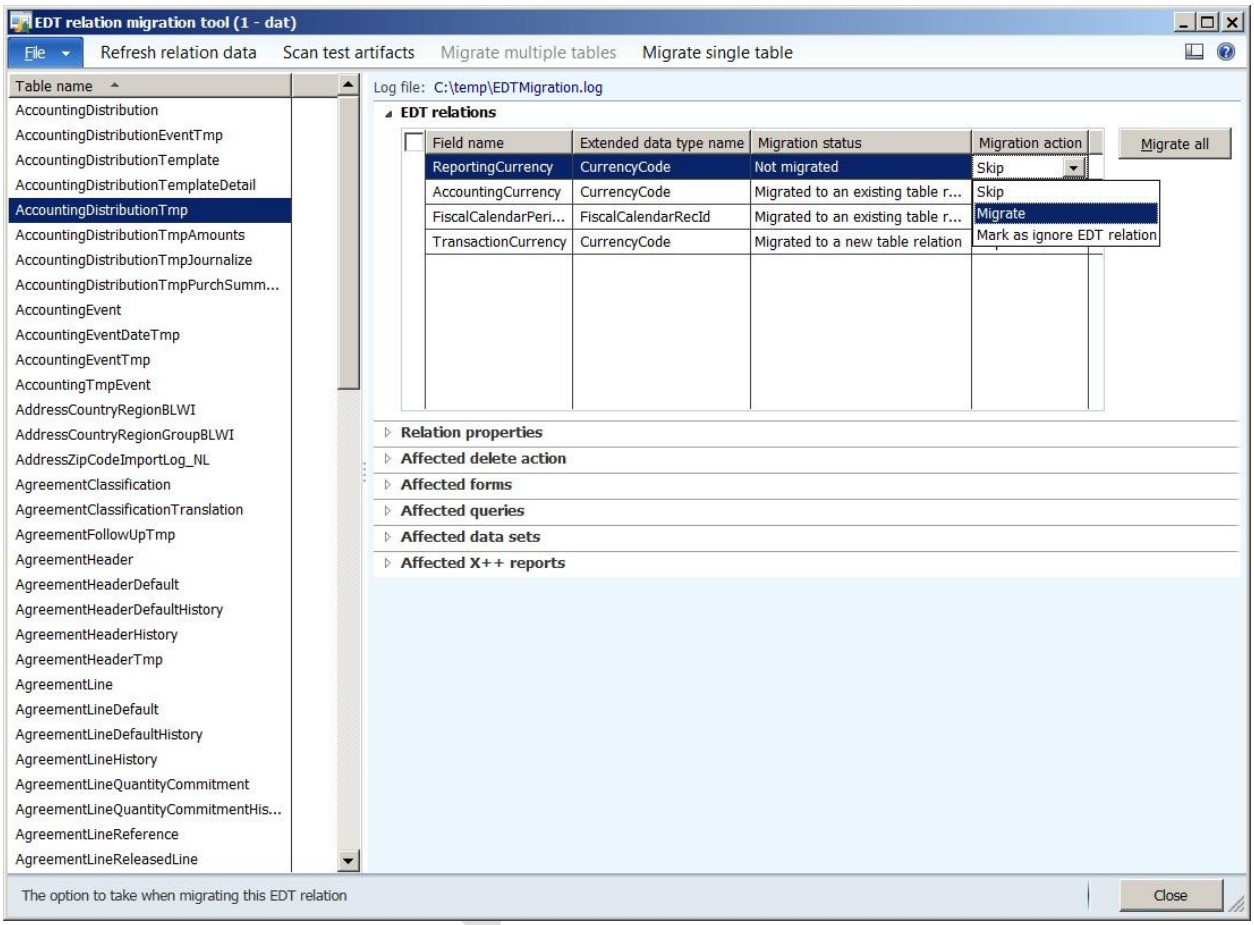

Figure 3: Setting migration actions for a single table

Process of migrating EDT relations on multiple tables

The process for migrating the EDT relations for multiple tables includes the following steps:

1. From the **Table name** pane, select all the tables to which you want to migrate the EDT relation.
2. Click the **Migrate multiple tables** button on the top ribbon.

   The migration action will always be **Migrate** for all tables selected.

Figure 4 shows the selection of multiple tables for the migration of EDT relations.

Pre-I

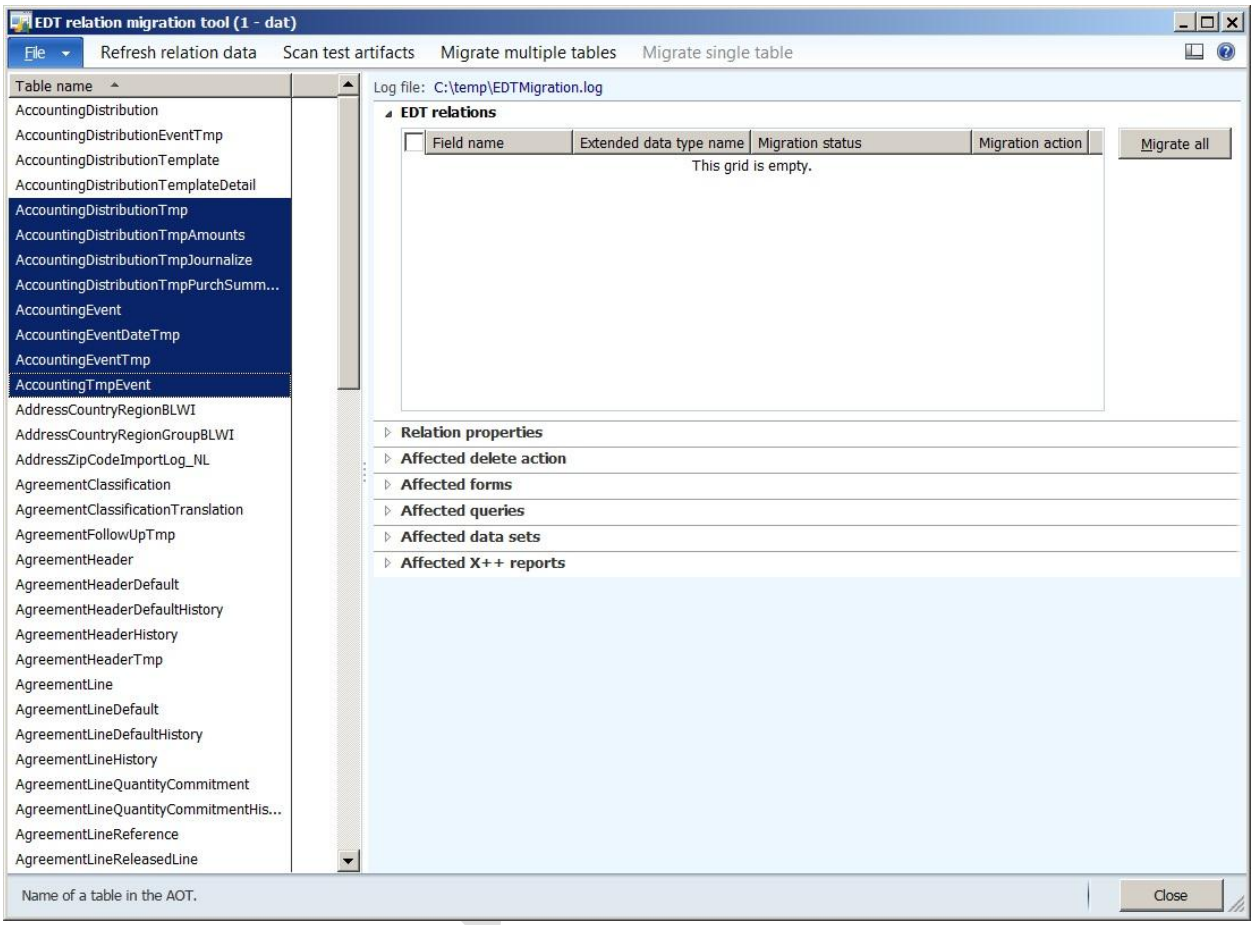

Figure 4: Migrating EDT relations on multiple tables

Pre-R

Migration scenarios

This section describes how to use the EDT migration tool in a number of migration scenarios.

The example scenarios all use the tables "PKTable" and "FKTable," with the PKTable holding the referenced key and the FKTable holding the referencing key that points to the PKTable.

The "Resulting runtime behavior" subsection in each scenario is provided to help developers understand how markers are used by the kernel to maintain backward compatibility. This is for your information only--how the kernel uses markers should not be of concern as long as the markers have been set correctly, which can be done by either the migration tool or a developer.

The following table describes the migration criteria for each scenario.

Scenario	EDT normal relation points to a unique key?	EDT fixed relation points to a unique key?	Existing table relation exactly matches the EDT relation?	Existing table relation is a superset of the EDT relation?
1. Migrating an EDT relation to a new table relation	Yes	No	No	No
2. Migrating an EDT relation to a matching table relation	Yes	No	Yes	No
3. Migrating an EDT relation to a table relation with a field link superset	Yes	No	No	Yes
4. Ignoring or manually migrating an EDT relation	No. EDT relation does not have fixed field links, but referenced field is not a unique key by itself.	No	No	No
5. Migrating an EDT relation using fixed field links to express the relation	Yes. EDT relation has fixed field links.	Yes	No	No
6. Migrating an EDT relation with fixed field links used as a filter condition	Yes. EDT relation has fixed field links.	EDT fixed relation points to a superset of a unique key.	No	No

Pre-release

Scenario 1: Migrating an EDT relation to a new table relation

This scenario illustrates the most straightforward case in which an EDT relation is migrated to a table where the relation was previously not defined. A new table relation is created.

Figure 5 shows the environment before migration. The EDT "PKTableField1" defines a relation to the PKTable.Field1 field, which is an alternate key AK1.

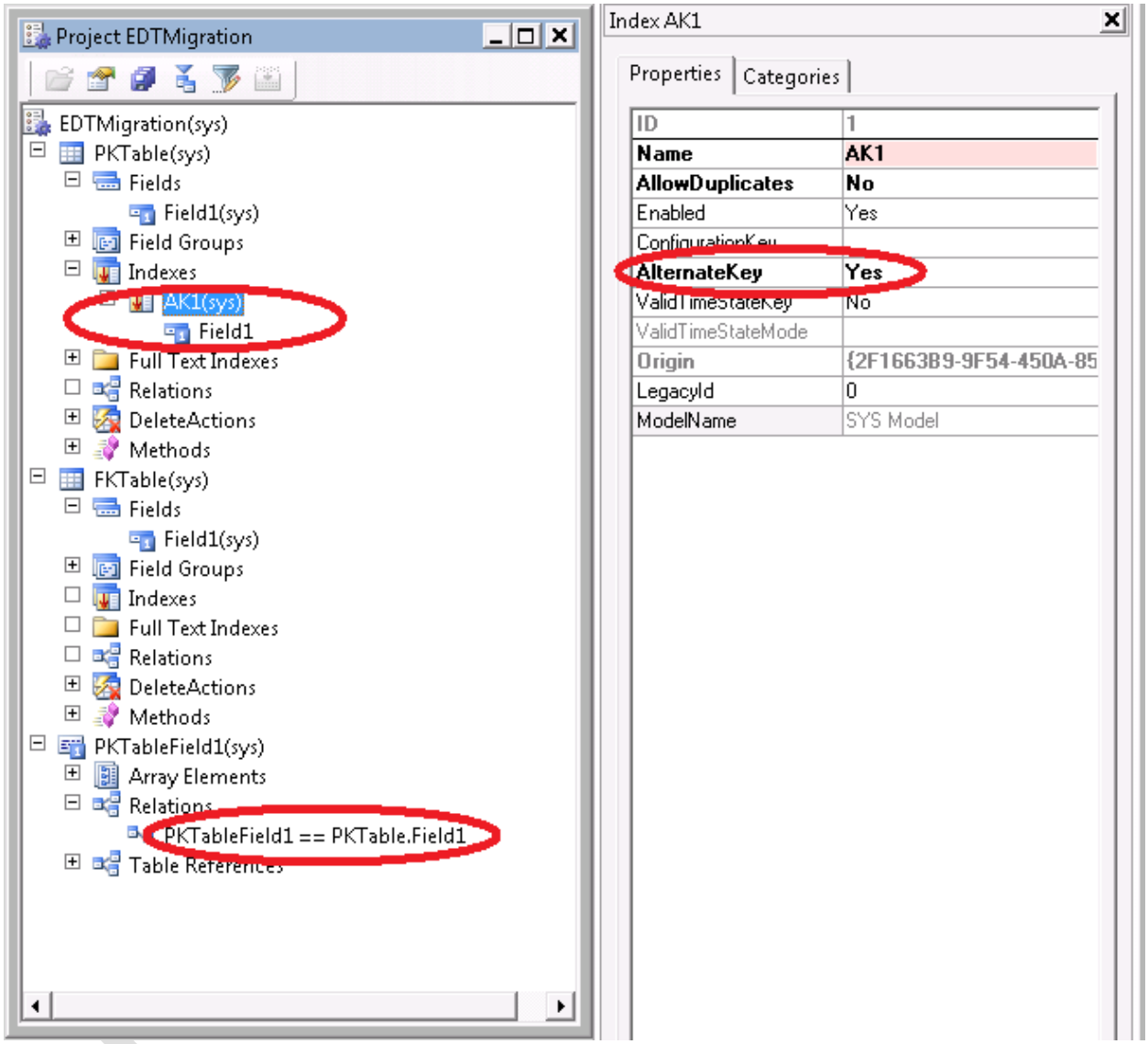

Figure 5: PKTableField1 EDT contains a relation to the PKTable.Field1

Also before migration, the FKTable.Field1 field uses the EDT PKTableField1, which makes it a foreign key into PKTable, but there is no table relation defined on the FKTable. Instead, the **ExtendedDataType** property on FKTable.Field1 is set to PKTableField1, as shown in Figure 6.

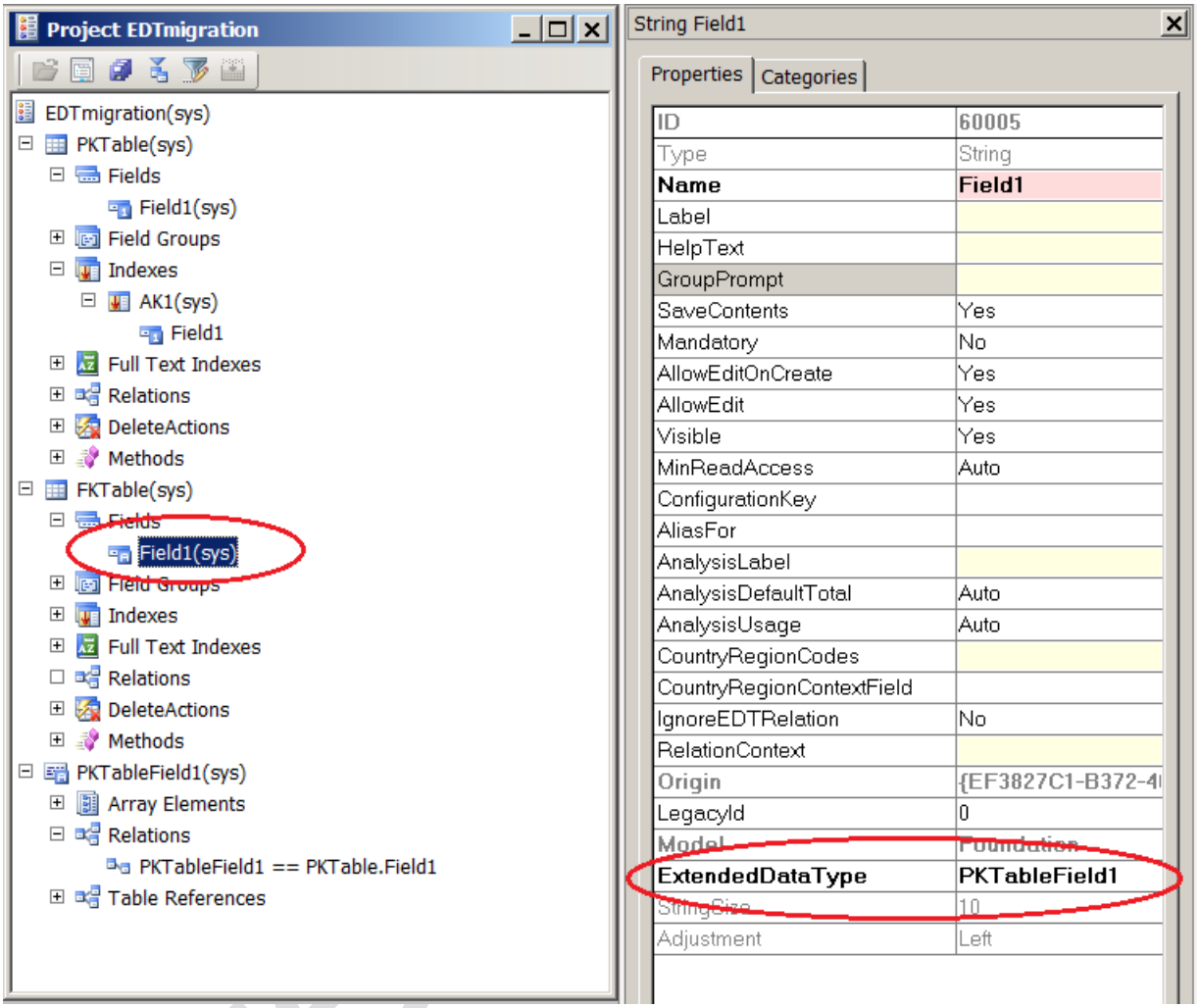

Figure 6: ExtendedDataType property of the FKTable > Fields > Field1 node set to PKTableField1

Pre

Figure 7 shows how the information from Figures 5 and 6 are displayed on the **EDT relation migration tool**.

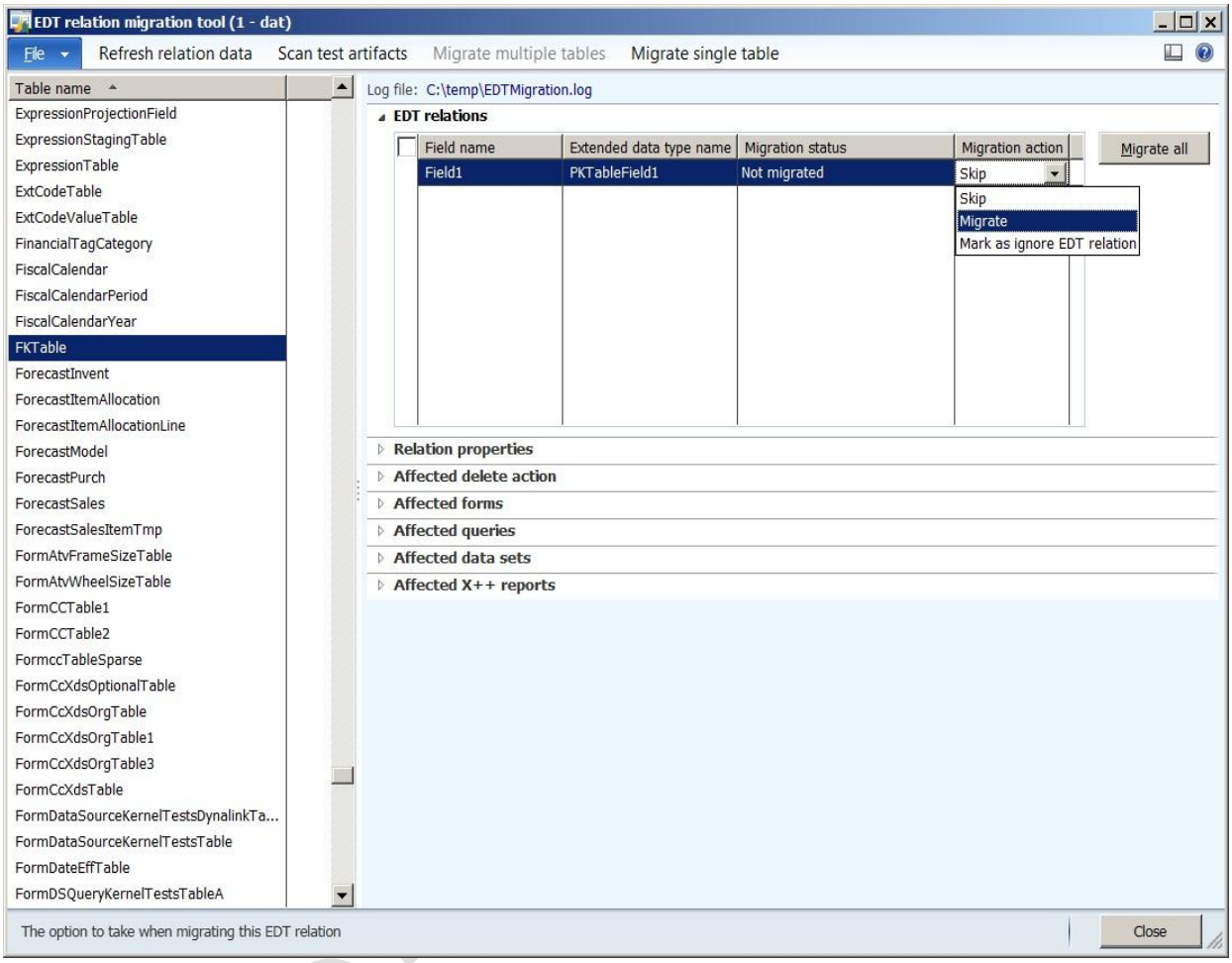

Figure 7: Setting the tool to migrate the EDT relation PKTable.Field1 to the FKTable

What the migration tool does

The EDT relation migration tool performs the following actions:

- Creates the new relation to the PKTable under the **Relations** node of the FKTable. This relation will be of type "Normal" because the key is not the primary key.
- Sets the **EDTRelation** property of the PKTable relation to "Yes" because the tool performs the direct migration of an EDT relation to the table relation.
- Creates one field link, FKTable.Field1 == PKTable.Field1, for the PKTable relation.
- Sets the **SourceEDT** property of the field link to PKTableField1.

Resulting runtime behavior

At runtime, after the changes have been made, the following behaviors will occur:

- APIs that used the EDT relation first on "FKTable.Field1" will now find the same relation with the same field link under the PKTable relation by examining its **SourceEDT** property.
- If a table relation that refers to PKTable already exists in FKTable, APIs that used table relations first will not pick up the PKTable relation because it is flagged as an **EDTRelation**.

Pre-release

After migration, Figure 8 shows the addition of the PKTable relation to the **Relations** node of the FKTable, with its **EDTRelation** property set to "Yes".

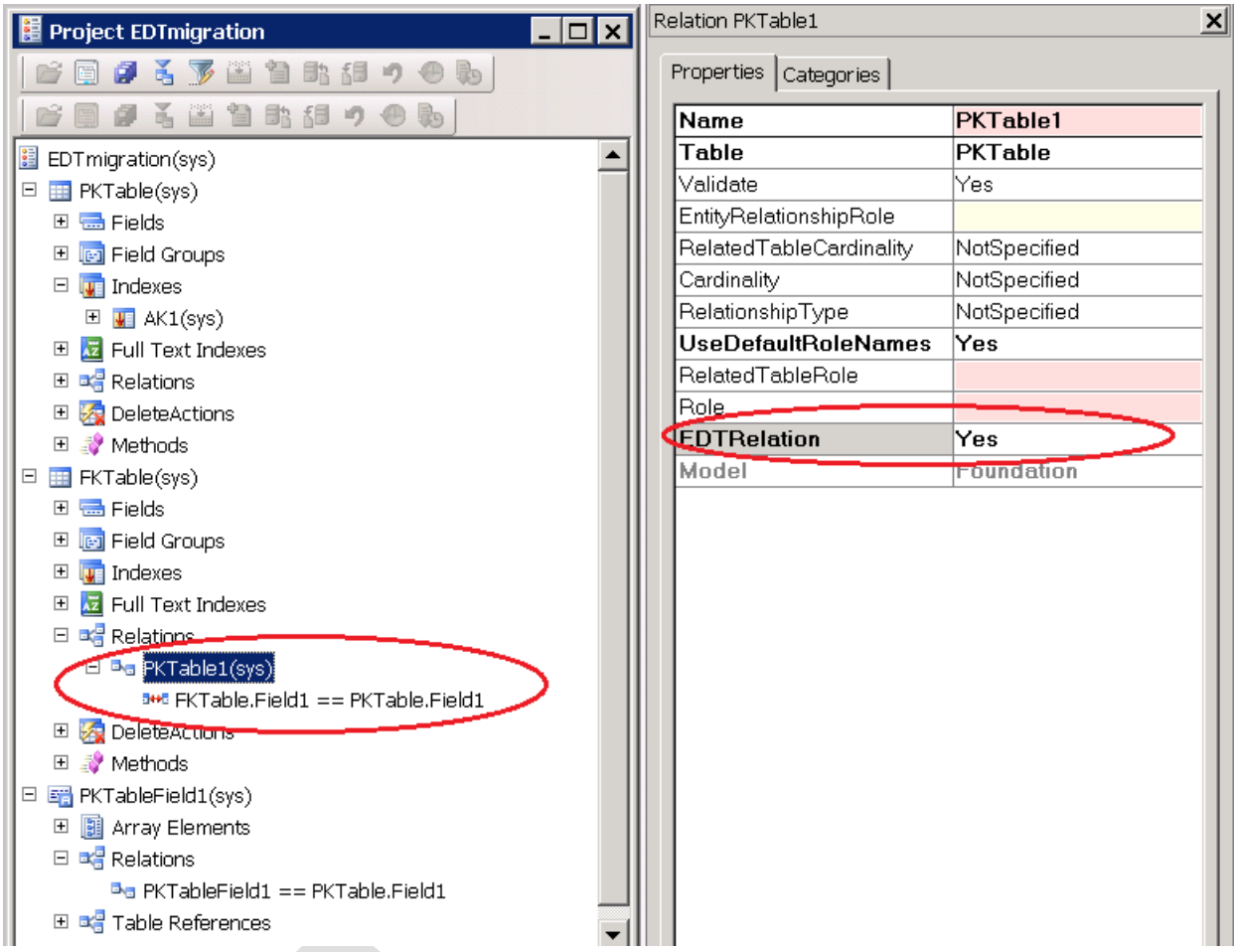

Figure 8: Result of the EDT migration

Pre

After migration, Figure 9 shows that the **SourceEDT** property of FKTable.Field1 is set to "PKTableField1" to maintain a relationship with the EDT.

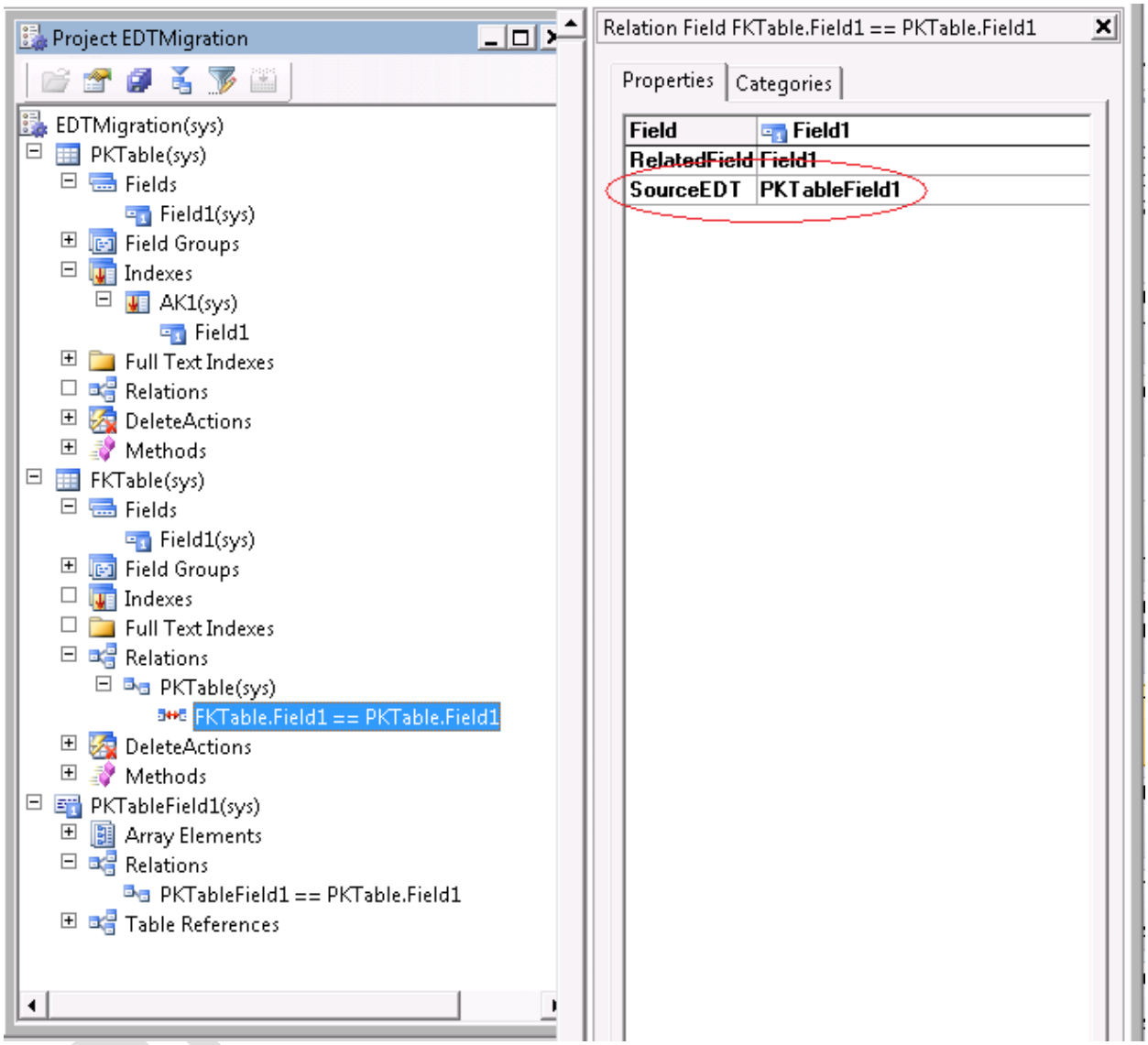

Figure 9: SourceEDT property set on the new table relation to indicate originating EDT

Scenario 2: Migrating an EDT relation to a matching table relation

This scenario is similar to the previous scenario. In this case, the FKTable.Field2 field holds the EDT PKTableField1 as a foreign key into the PKTable. However, there is already an existing relation to PKTable2 that is defined with exactly the same field link.

Because the table relation already exists, the EDT relation migration consists of setting the relevant properties of the table and the EDT to maintain correct behavior of the applications.

In Figure 10, the EDTRelation property on the PKTable2 relation is set to "No" because the EDT relation was not migrated to a new table relation.

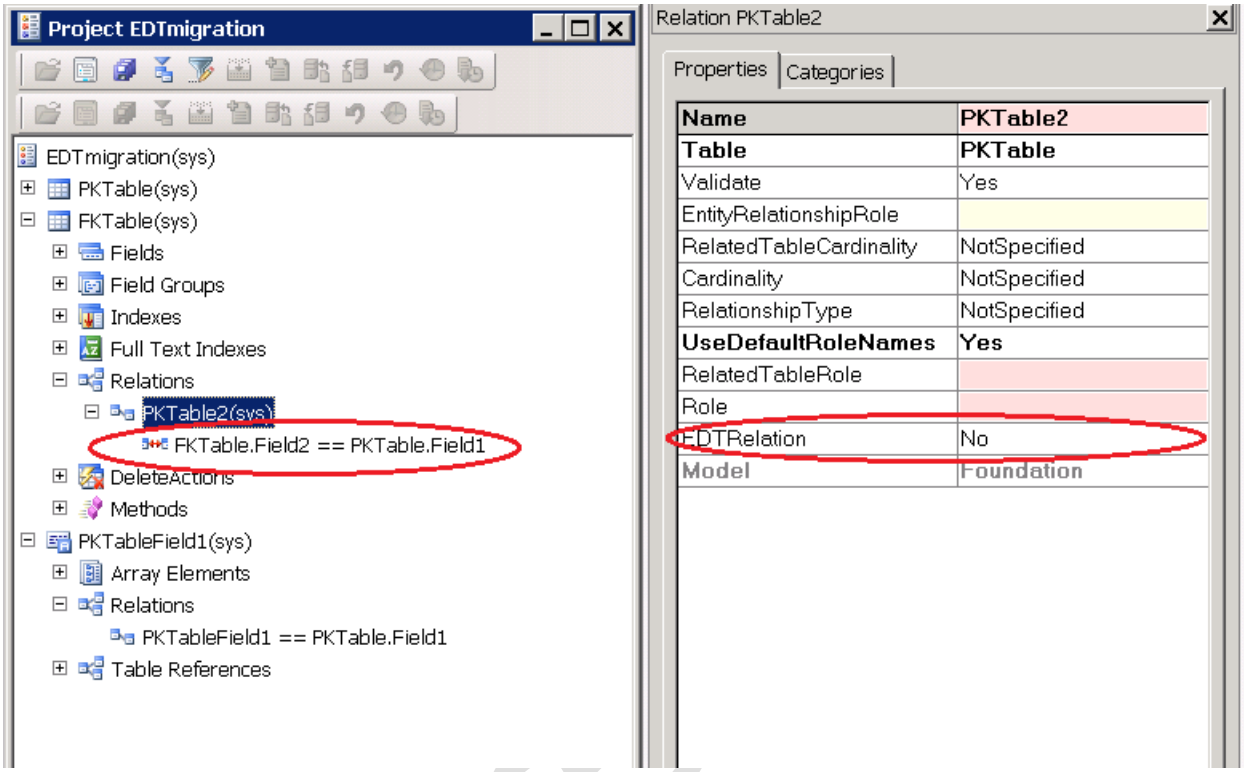

Figure 10: EDTRelation property indicates that the table relation already matched the EDT relation

What the migration tool does

The EDT relation migration tool performs the following actions:

- Does not create a new relation because there is already an existing table relation that matches the EDT relation.
- The **EDTRelation** property of PKTable2 remains set to "No" because this relation did not need to be added to the table **Relations** node.
- Sets the **SourceEDT** property on the field link to PKTableField1 to indicate that this field link matches the EDT relation defined on the EDT PKTableField1.

Resulting runtime behavior

At runtime, after the changes have been made, the following behaviors will occur:

- APIs that used the EDT relation first on the FKTable.Field2 will find the same relation with the same field link under the PKTable2 relation by examining the **SourceEDT** property.
- APIs that used table relations first will still pick up the PKTable2 relation first because it is not flagged as an **EDTRelation**.

Pre-re

Scenario 3: Migrating an EDT relation to a table relation with a field link superset

In scenario 3, The FKTable.Field3 field holds the EDT PKTableField1 as a foreign key into the PKTable. The relation table PKTable3 is already defined in the Relations node of the FKTable, but there are more field links on this table relation than are defined on the EDT Relations node.

In Figure 11, FKTable.Field4 has been duplicated from PKTable.Field2. The field link on PKTable.Field2 can be a normal or fixed field link.

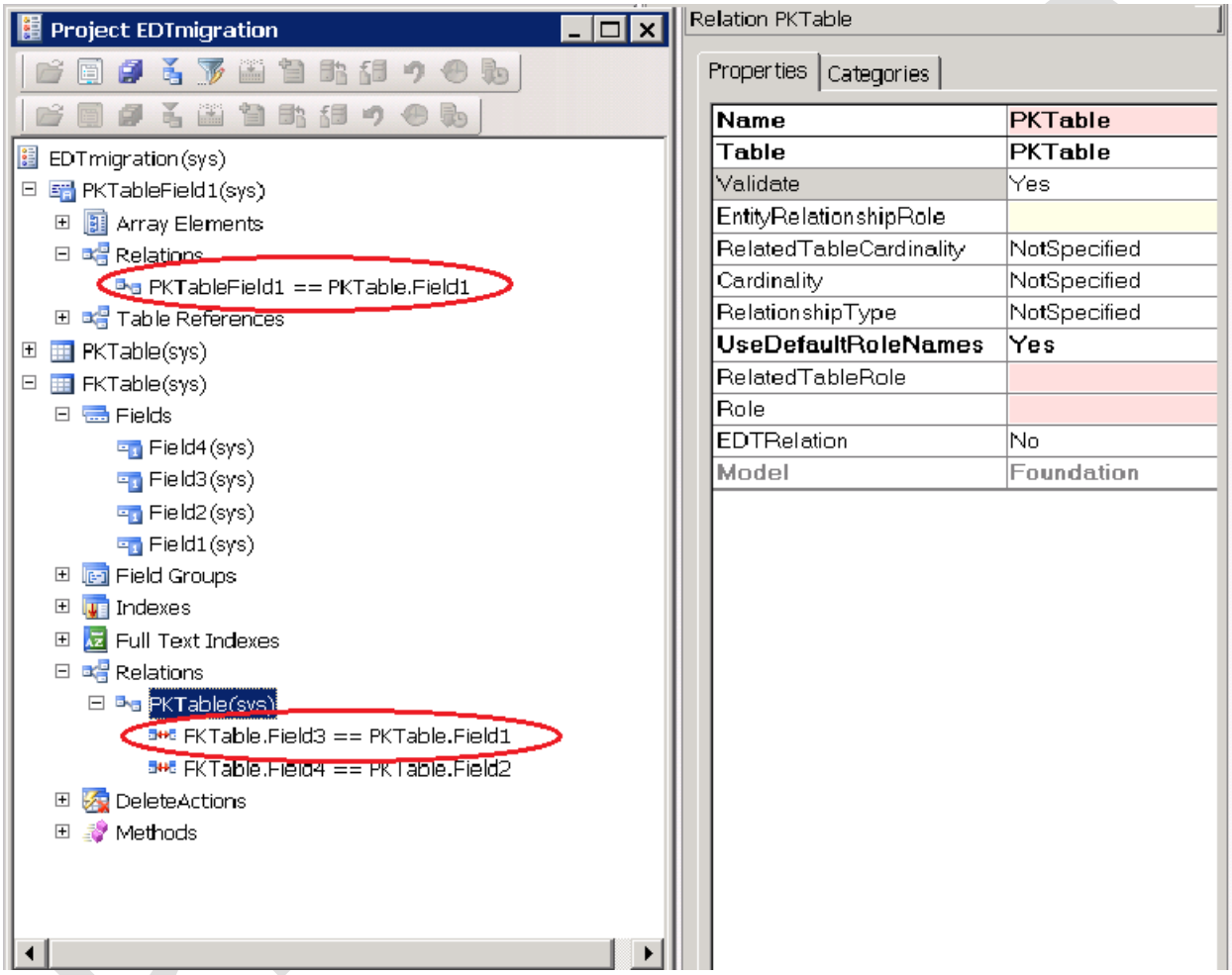

Figure 11: Table relation containing more field links (superset) than the EDT relation

What the migration tool does

The EDT relation migration tool performs the following actions:

- Does not create a new table relation because there is already an existing table relation that matches the EDT relation field link.
- The **EDTRelation** property remains set to "No" because a new table relation did not need to be added for the EDT relation.
- Sets the **SourceEDT** property to "PKTableField1" on the field link that matches the EDT relation field link. This is done to indicate that the two field links are the same.

Resulting runtime behavior:

At runtime, after the changes have been made, the following behaviors will occur:

- APIs that use the EDT relation first on FKTable.Field3 will find the same relation with the same field link under the PKTable3 table relation by examining the **SourceEDT** property on the field links and taking only the link that specifies the matching EDT.
- APIs that use table relations first will continue to pick up the PKTable3 relation first because the **EDTRelation** property does not flag it as an EDT relation.

Scenario 4: Ignoring or manually migrating an EDT relation

In this scenario, the relation in the EDT is invalid for the current table.

If the developer chooses the "Migrate" migration option on the form, the following error message will be displayed:

"The type of index covering the EDT in the PKTable table is neither a Primary Key nor an Alternate Key, or there is no covering index for the PKTableField2 EDT in the PKTable. Therefore the relation in the PKTableField2 EDT cannot be migrated."

The developer can proceed in one of two ways: ignore the EDT relation or manually migrate it.

Ignoring the EDT relation

If there is no semantic relationship between the two tables, the developer should choose the "Mark as ignore EDT relation" migration option. In this case, the tool will not create a new relation in the FKTable. Instead, it will set the IgnoreEDTRelation property on FKTable.Field2 to "Yes."

After migration, the resulting runtime behavior will be as follows: APIs that previously used the EDT relation on FKTable.Field2 and which require relational semantics (such as joins, delete actions, and so on) will not use the EDT relation because its IgnoreEDTRelation property is set to "Yes."

Manually migrating the EDT relation

If there is a semantic relationship between the two tables, the developer should manually migrate the relation from the EDT to the table. The developer must consider how to update the data model.

In the following example, the EDT PKTableField2 defines a relationship with the PKTable.Field2 field. By itself, this field is not a unique key, but is part of the alternate key AK2, which consists of PKTable.Field2 and PKTable.Field3.

Prerelease

Figure 12 shows the relationship between the EDT relation (on EDT PKTableField2), which is part of an alternate key, and the complete alternate key. It also shows that FKTable has no existing relation to the alternate key.

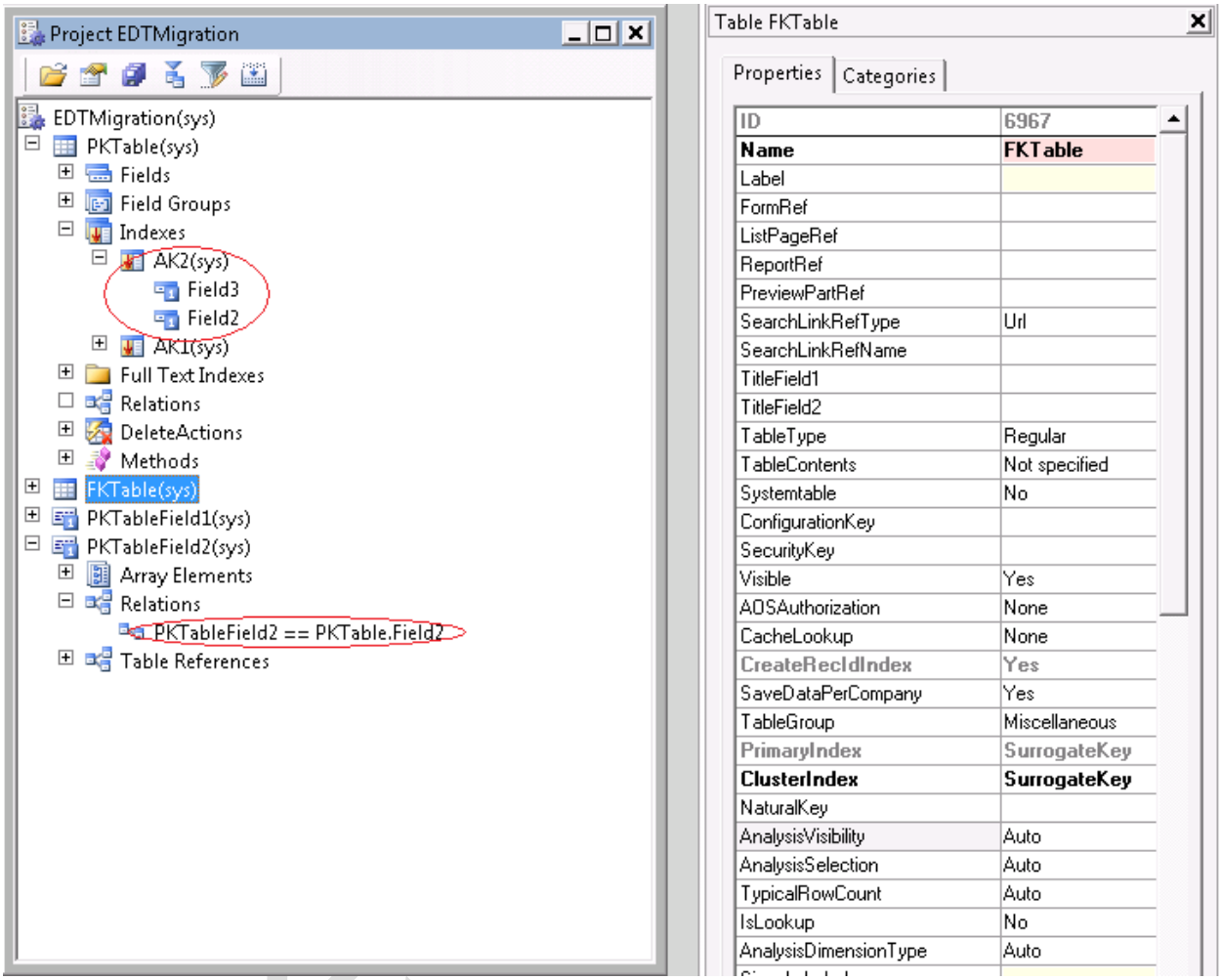

Figure 12: Partial alternate key defined on the EDT relation

FKTable.Field2 uses the EDT PKTableField2, but no existing table relation on the FKTable contains Field2.

The developer has two choices, depending on whether there is a relationship between FKTable and PKTable.

- If the corresponding foreign key on FKTable that points to PKTable.Field3 already exists, create a relation with that field together with FKTable.Field2.
- If the corresponding foreign key on FKTable does not exist, which indicates that the data model is not complete, create an additional field on FKTable and create a relation with that field together with FKTable.Field2. The developer might need to upgrade the script to populate the added fields.

Pre

Perform a manual migration:

1. Create a new table relation that includes two field links (FKTable.Field2 and FKTable.Field3) as described above.
2. Set the **EDTRelation** property of the table relation to "Yes." This relation is added because the EDT relation has been migrated to the table.
3. Set the **SourceEDT** property of the field link that matches the EDT relation to the extended data type PKTableField2. This is done to indicate that the field link matches the EDT relation field link defined on the PKTableField2 extended data type.
4. Do not set the **SourceEDT** property of the field link that does not match the EDT relation because it does not match the EDT relation field link.

Resulting runtime behavior

At runtime, after the changes have been made, the following behaviors will occur:

- APIs that use the EDT relation first on "FKTable.Field2" now will be able to find the same relation with the same field link under the PKTable relation by examining the **SourceEDT** property.
- APIs that use the table relation first will not pick up the PKTable relation first because it is flagged as an EDT relation (that is, the **EDTRelation** property is set to "Yes").

  Note EDT markers are used for maintaining the behavior that existed prior to EDT relation migration, and are therefore backward compatible. These markers can also be used to change the behavior of the application if desired.

Pre-release

Figure 13 shows the new table relation created by the manual migration.

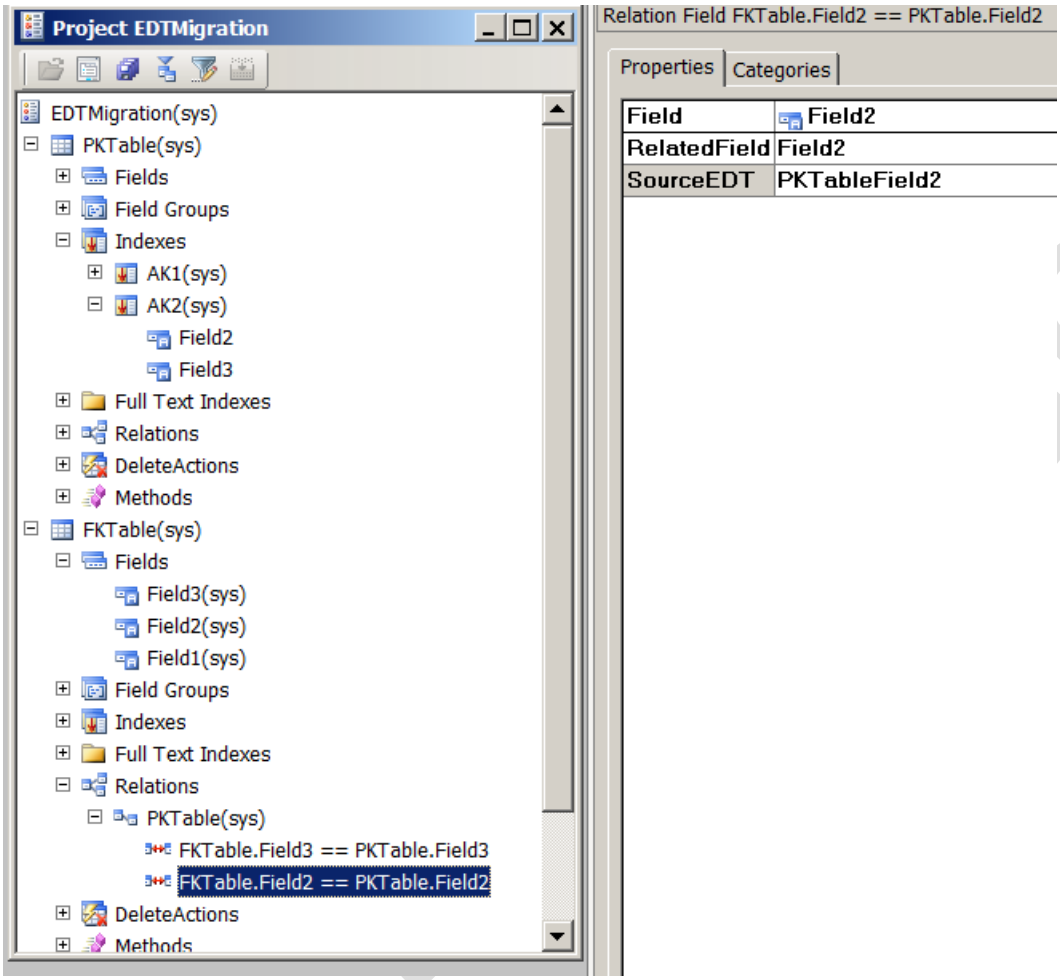

Figure 13: Result of the manual migration

Pre

Scenarios 5 and 6: Migrating an EDT relation with fixed field links

Scenarios 5 and 6 both describe the EDT relation migration process when fixed field links are involved.

- Scenario 5 describes the migration when the complete set of foreign field key attributes exists in the EDT relation, but not in the table relation.
- Scenario 6 describes the migration when the fixed field links in the EDT relation serve as a filter condition.

Scenario 5: Migrating an EDT relation using fixed field links to express the relation

In this scenario, the set of referenced fields in the EDT relation makes up a primary key or alternate key, but only one of the referenced fields exists in the referencing table (FKTable) field list. This foreign key relationship is not syntactically correct because the complete set of foreign key attributes is not available. However, because the rest of the foreign key attributes are constant—and therefore redundant—they are simply omitted from the referencing table and expressed in the fixed field of the migrated relation.

The migration tool creates new foreign key relationships, in which the **EDTRelation** property is set to "Yes" on the new relation and the **SourceEDT** property is set to the EDT on all of the field links, including the fixed field links.

Figure 14 show Scenario 5 before the migration. The set of referenced fields in the EDT relation consists of two fields, PKTable.pk3 and PKTable.pk2 (which is a constant).

Pre-release

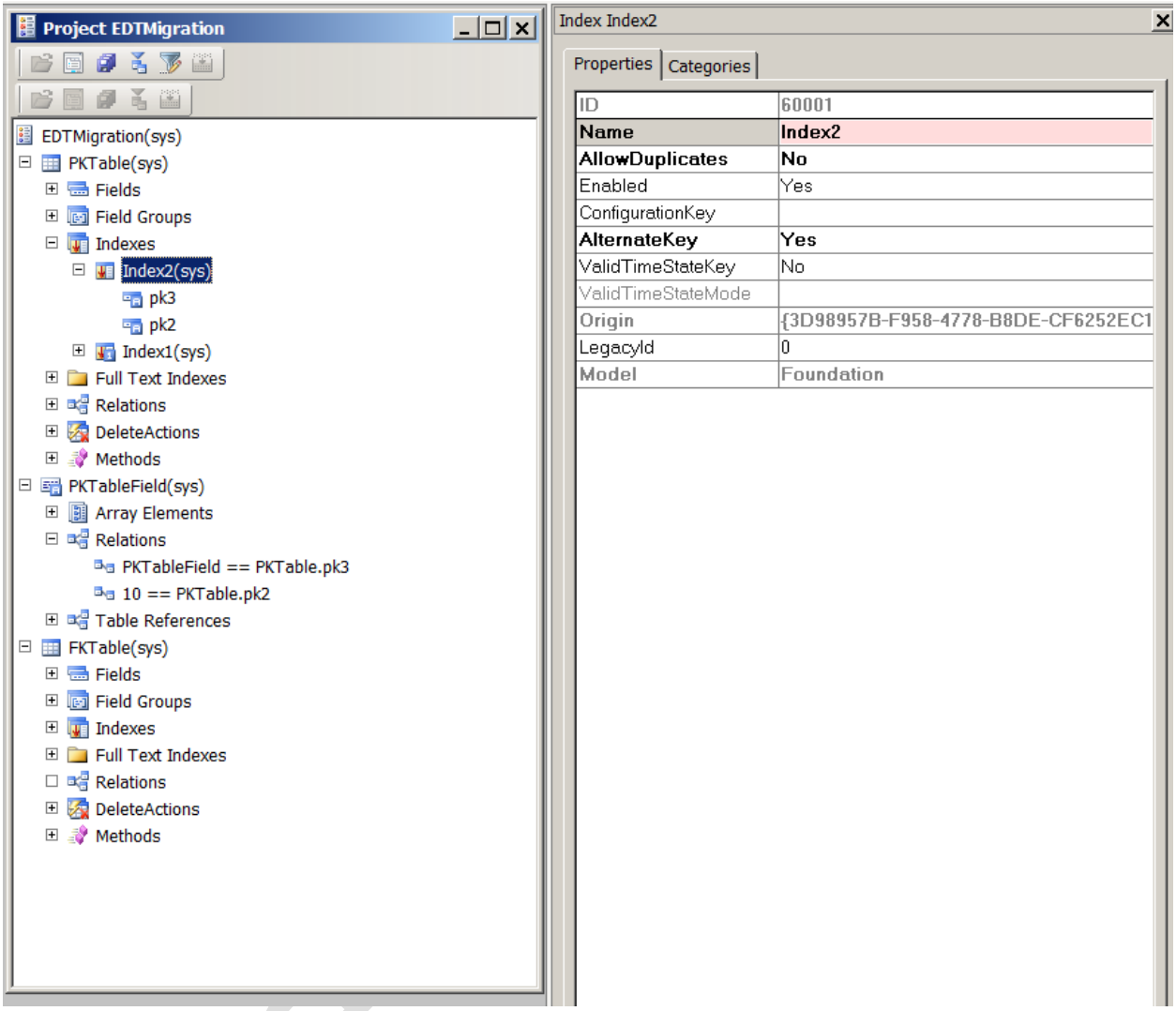

Figure 14: Scenario 5 before migration

Pre

Figure 15 shows scenario 5 after the migration.

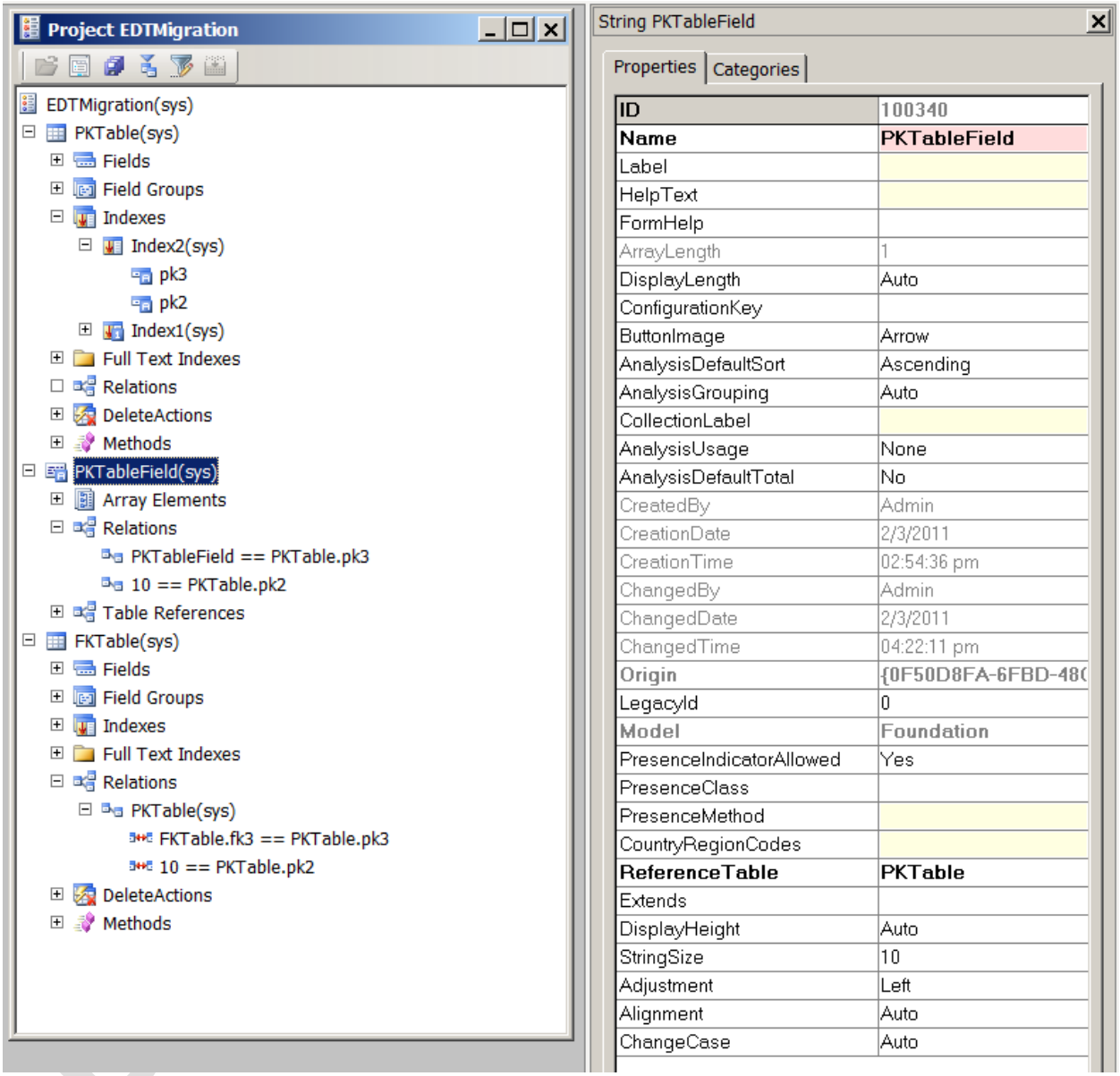

Figure 15: Scenario 5 after migration

Scenario 6: Migrating an EDT relation with fixed field links used as a filter condition

In this scenario, the fixed field links serve as a filter condition on top of the relation expressed by the normal field link.

The migration tool creates new foreign key relationships, in which the **EDTRelation** property is set to "Yes" on the new relation and the **SourceEDT** property is set to the EDT on all of the field links, including the fixed field links.

Figure 16 shows scenario 6 before the migration.

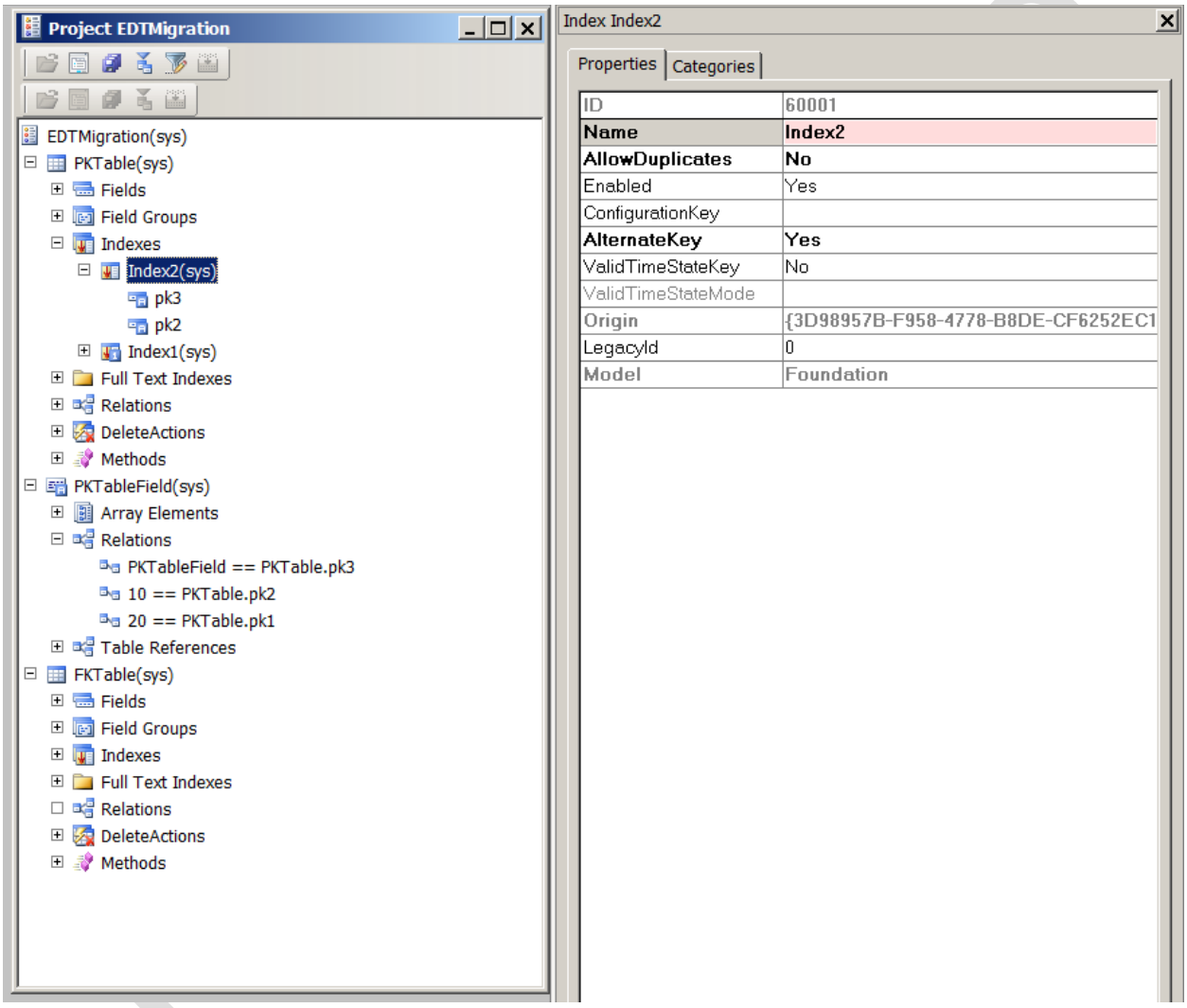

Figure 16: Scenario 6 before migration

Figure 17 shows scenario 6 after the migration.

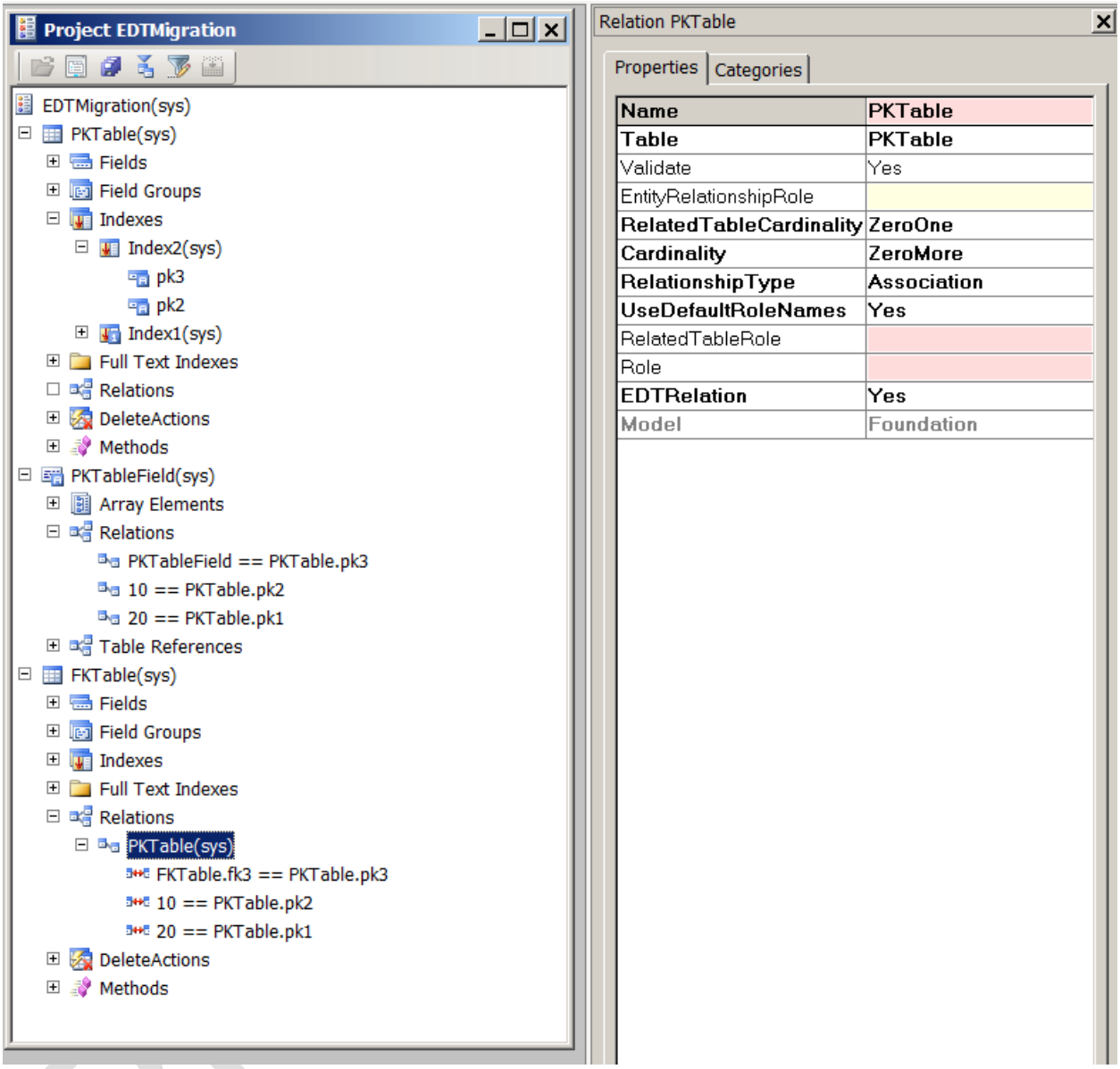

Figure 16: Scenario 6 after migration

Conclusion

Starting with Microsoft Dynamics AX 2012, all table relationships can be defined on tables only. Developers are not required to move existing EDT relations. However, if you wish to modify an EDT relation, that relation must be migrated to a table. You can migrate EDT relations manually, or by using the EDT relation migration tool.

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989
Worldwide +1-701-281-6500
www.microsoft.com/dynamics

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2011 Microsoft Corporation. All rights reserved.

Microsoft, Microsoft Dynamics, and the Microsoft Dynamics logo are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.