Installationsanleitung

All.Relation V4.0.1015.0002

Installation

1. Kopieren Sie die Datei AxRelation.dll in das PROFFIX-Programmverzeichnis.
2. Erstellen Sie im PROFFIX-Programmverzeichnis einen Ordner mit Namen „Schema“ und kopieren Sie die Datei „EntitySchema.xml“ in diesen Ordner.

Konfiguration Teil 1

Relationale Zusatztabelle

1. Erstellen Sie in PROFFIX auf der gewünschten Tabelle ein Zusatzfeld vom Typ „Schaltfäche“ und vergeben Sie eine beliebige Bezeichnung.
    a. Im Feld „Datei“ geben Sie den Pfad zu der Datei „AxRelation.dll“ an. Achten Sie bitte darauf, dass Sie den Pfad zwischen Anführungs- und Schlusszeichen setzen.
    b. Im selben Feld hinter dem Dateipfad setzen Sie nun ebenfalls zwischen Anführung- und Schlusszeichen einen eindeutigen Parameter. Dieser wird später wieder benötigt.
    Bsp. für den kompletten Pfad: "N:\Prog\AxRelation.dll" "ADR_Adressen001"
2. Erstellen Sie nun in PROFFIX Ihre gewünschte Zusatztabelle, nach Ihren Bedürfnissen
    a. Die Zusatztabelle muss ein Feld beinhalten, dass dem Fremdschlüssel der Relation entspricht. Erstellen Sie dieses Feld mit dem selben Datentyp und dem selben Namen wie in der Elterntabelle auf der Datenbank. Als Bsp. für die Adresstabelle würden Sie ein Feld mit Namen „AdressNrADR“ und Feldtyp „ganze Zahl“ erstellen.
    Bitte beachten Sie, dass dieses Feld den Parameter „Eingabe zwingend“ nicht gesetzt haben darf!
3. Öffnen Sie nun das File „EntitySchema.xml“ in einem Texteditor und ergänzen oder ändern Sie dieses wie folgt:
    a. Im Node „Entity“ wird im Attribut „name“ der Parameter hinterlegt, den Sie unter Punkt 1b erstellt haben.
    b. Im Node „PrimaryKey“ setzen Sie den Namen des Feldes ein, das dem Primärschlüssel auf der Elterntabelle entspricht, so wie Sie es unter Punkt 2a gemacht haben.
    c. Im Node „RelationEntity“ setzten Sie das Attribut „entity“ auf den Namen der Zusatztabelle, so wie er in der Datenbank gespeichert ist. Der Name entspricht dem Tabellennamen, den Sie eingegeben haben mit der Zeichfolge „ZUS_“ vorangestellt. Haben Sie also Ihre Zusatztabelle „ABC“ benannt, heisst diese auf der Datenbank „ZUS_ABC“.
    d. Im Node „Field“ setzen Sie das Attribut „typ“ auf „PrimaryKey“ und den Wert auf den Namen der Übergabevariablen von PROFFIX. Der Name der Übergabevariablen ist in der Regel der selbe wie der des PrimaryKey mit der Zeichfolge „dfn“ oder „dfs“ vorangestellt. dfn wird dabei bei nummerischen Werten verwendet, während „dfs“ bei Alphanummerischen Werten verwendet wird. Daher in unserem Beispiel „dfnAdressNrADR“. Um den Namen der Übergabevariable überprüfen zu können, müssen Sie eine Schaltfläche in PROFFIX definieren welche das Programm Notepad.exe aufruft. Nach anklicken der Schaltfläche öffnet sich Notepad mit allen Variablen darin.

Beispiel einer korrekt formatierten EntitySchema.xml Datei

```
<?xml version="1.0" encoding="utf-8" ?>
<Root>
  <Entities>
    <Entity name="ADR_Adressen001">
      <PrimaryKey>AdressNrADR</PrimaryKey>
      <RelationEntity entity="ZUS_AX_Adressrelation"></RelationEntity>
      <Paramfields>
        <Field typ="PrimaryKey">dfnAdressNrADR</Field>
      </Paramfields>
      <Indexdefinition active="false">
        <IndexContainerfield></IndexContainerfield>
        <Indexfield fieldname="" query=""></Indexfield>
      </Indexdefinition>
    </Entity>
    <Entity name="ADR_Adressen002">
      <PrimaryKey>AdressNrADR</PrimaryKey>
      <RelationEntity entity="ZUS_Grosshandel"></RelationEntity>
      <Paramfields>
        <Field typ="PrimaryKey">dfnAdressNrADR</Field>
      </Paramfields>
      <Indexdefinition active="true">
        <IndexContainerfield>Z_GH_Vorschau</IndexContainerfield>
        <Indexfield fieldname="Grossist" query="SELECT Name FROM ADR_Adressen WHERE AdressNrADR = {Grossist}"></Indexfield>
        <Indexfield fieldname="Typ" query=""></Indexfield>
      </Indexdefinition>
    </Entity>
  </Entities>
</Root>
```

Abbildung 1 „EntitySchema.xml“

Index

Achtung, wenn Sie keinen Index wünschen, lassen Sie einfach alle Nodes der Indexdefinition leer. Achten Sie aber darauf, dass diese nicht fehlen, da sonst ein Fehler beim Lesen der Datei auftreten würde.

a. Im Node „Indexdefinition“ setzen Sie das Attribut „active“ auf „true“ wenn Sie einen Index auf die Elterntabelle erstellen wollen.
b. Im Node „IndexContainerfield“ geben Sie den Namen des Feldes an, wie er auf der Datenbank gespeichert ist, in welches Sie den Index geschrieben bekommen möchten. Das IndexContainerfield sollte ein Feld vom Typ „Memo“ sein, damit der Index auf jeden Fall vollständig gespeichert werden kann.
c. Für jedes Feld der relationalen Tabelle, welches dem Index hinzugefügt werden soll erstellen Sie einen Node „Indexfield“. „Indexfield“ beinhaltet zwei Attribute. Erstens das Attribut „fieldname“. In diesem geben Sie den Namen des Feldes an, welches dem Index hinzugefügt werden soll. Bitte verwenden Sie den Namen wie er auf der Datenbank gespeichert ist. Das zweite Attribut lautet „query“ und wird im Normalfall leer gelassen. Query kann verwendet werden, um den Indexeintrag zu übersteuern. Dies ist dann sinnvoll, wenn der Eintrag einfach nur eine Zahl wie die Adressnummer einer Adresse wäre. Da die Nummer nichts aussagt kann mit query z.B. die dazugehörige Bezeichnung abgefragt werden.

Query ist eine normale SQL Abfrage bis auf den Teil der Where – Klausel. Die Where – Klausel soll einen eindeutigen Wert zurückbringen. Daher muss diese mit dem Primärschlüssel eingeschränkt werden. Um dies zu ermöglichen können Sie mittels geschweiften Klammern und Angabe des Felds einen Platzhalter definieren. Der Platzhalter wird von All.Relation durch den effektiven Wert ersetzt. Momentan wird im Platzhalter nur der selbe Feldname unterstützt, wie bereits im Attribut „fieldname“ definiert wurde. Setzten Sie den Platzhalter also immer auf den selben Wert, wie „fieldname“.

```
<Indexfield fieldname="Grossist" query="SELECT Name FROM ADR_Adressen WHERE
AdressNrADR = {Grossist}"></Indexfield>
```

Prüfungen (Checks)

Wenn Prüfungen benötigt werden kann unterhalb des Nodes „Indexdefinition“ der Node „Checks“ erstellt werden. Ein korrekt formatiertes Beispiel sehen Sie in Abbildung 2.

```
<?xml version="1.0" encoding="utf-8" ?>
<Root>
  <DebugMode>False</DebugMode>
  <Logfile>C:\Temp\Log.txt</Logfile>
  <Entities>
    <Entity name="Artikel001">
        <PrimaryKey>ArtikelNrLAG</PrimaryKey>
        <RelationEntity entity="ZUS_Bereich_Details"></RelationEntity>
        <Paramfields>
            <Field typ="PrimaryKey">dfsArtikelNrLAG</Field>
        </Paramfields>
        <Indexdefinition active="false">
            <IndexContainerfield></IndexContainerfield>
            <Indexfield fieldname="" query=""></Indexfield>
        </Indexdefinition>
    </Entity>
        <Entity name="Verbuchung">
        <PrimaryKey>FK_Artikel</PrimaryKey>
        <RelationEntity entity="ZUS_Verbuchungslisten"></RelationEntity>
        <Paramfields>
            <Field typ="PrimaryKey">dfsArtikelNrLAG</Field>
        </Paramfields>
        <Indexdefinition active="false">
            <IndexContainerfield></IndexContainerfield>
            <Indexfield fieldname="" query=""></Indexfield>
        </Indexdefinition>
        <Checks>
            <Check name="Verbuchung" type="Once" reaction="Abort" acceptEmptyTable="True"> <!-- types: Once, Each; reactior
                <Query1>SELECT Verkauf1 FROM LAG_Artikel WHERE ArtikelNrLAG = '{0}'</Query1>
                <Parameters>
                    <Parameter type="File">dfsArtikelNrLAG</Parameter> <!-- types: File = Px. Parameterfile, Column = Spalt
                </Parameters>
                <Query2>SELECT SUM(Preis) FROM ZUS_Verbuchungslisten WHERE FK_Artikel = '{0}'</Query2>
                <Parameters type ="File">
                    <Parameter>dfsArtikelNrLAG</Parameter>
                </Parameters>
                <ComparisonOperator>=</ComparisonOperator>
                <Message>Die Summe der Verbuchungslistenpreise muss gleich dem Verkaufspreis1 des Artikels sein.</Message>
            </Check>
        </Checks>
    </Entity>
  </Entities>
</Root>
```

Abbildung 2 „Prüfungen“

a. Im Node „Check“ definieren Sie folgende Attribute
   i. „name“ frei wählbar und ohne Funktion
   ii. „type“ Once oder Each -> Erklärung weiter unten
   iii. „reaction“ Warning oder Abort -> Erklärung weiter unten
   iv. „acceptEmtpyTable“ Dieser Schalter definiert, ob bei einer leeren Zusatztabelle die Prüfung ignoriert werden soll.

b. Im Node „Query1“ und „Query2“ wird eine SQL-Klausel erwartet, die einen skalaren Wert zurück gibt. Mittels geschweiften Klammern und Indexwert 0 bis n können Platzhalter für Ersetzungen angegeben werden. Selbstverständlich können hier für komplexe Aufgaben auch Stored Procedures oder Functions verwendet werden, die einen Outputparameter zurück geben.

c. Im Node Parameter definieren Sie folgende Eigenschaften

a. Das Attribut „type“ File oder Column -> Erklärung weiter unten

b. Der Name des abzufragenden Ersatzwertes. Im Fall von „File“ den Parametername im PROFFIX Parameterfile. Im Fall von „Column“ den Spaltenname auf der Zusatztabelle.

d. Im Node „ComparisonOperator“ definieren Sie einen gültigen Operator mit dem die beiden Werte verglichen werden können.

e. Im Node „Message“ definieren Sie die Meldung, welche der Anwender sehen soll, wenn die Prüfung negativ ausfällt.

Folgende Prüfungstypen stehen zur Verfügung:

- Once – Führt die Prüfung beim Schliessen der Zusatztabelle einmalig aus.
- Each – Führt die Prüfung beim Schliessen der Zusatztabelle für jeden Datensatz der Zusatztabelle aus.

Folgende Reaktionstypen stehen zur Verfügung:

- Warning – Gibt eine Warnung aus, diese kann ignoriert werden.
- Abort – Gibt eine Fehlermeldung zurück, um die Zusatztabelle zu schliessen muss die Regel eingehalten werden.

Folgende Parametertypen stehen zur Verfügung:

- File – Der angegebene Parameter wird durch den aktuellen Wert gem. PROFFIX Parameterfile ersetzt.
- Column – Der angegebene Parameter wird durch den aktuellen Wert der Spalte der Zusatztabelle ersetzt.

Spezialfall Tabellen ohne konstanten Primärschlüssel

Es gibt in PROFFIX Tabellen die keinen konstanten Primärschlüssel anbieten. Dies Beispielsweise «AUF_DokumentPos» oder «FIB_Buchungen». Auch einige Hilfstabellen wie z.B. «LAG_VerpackungenLink» verfügen über keine Primärschlüsselspalte welche für All.Relation verwendbar ist.

In einem solchen Fall kein auf der Muttertabelle selbst ein Primärschlüssel über ein Zusatzfeld definiert werden. Das Zusatzfeld muss vom Datentyp «Text» sein und eine Länge von mindestens 32 Zeichen zulassen. Bei der Erstellung des Zusatzfeldes bitte unbedingt die Option «Feld für Eingabe sperren» aktivieren.

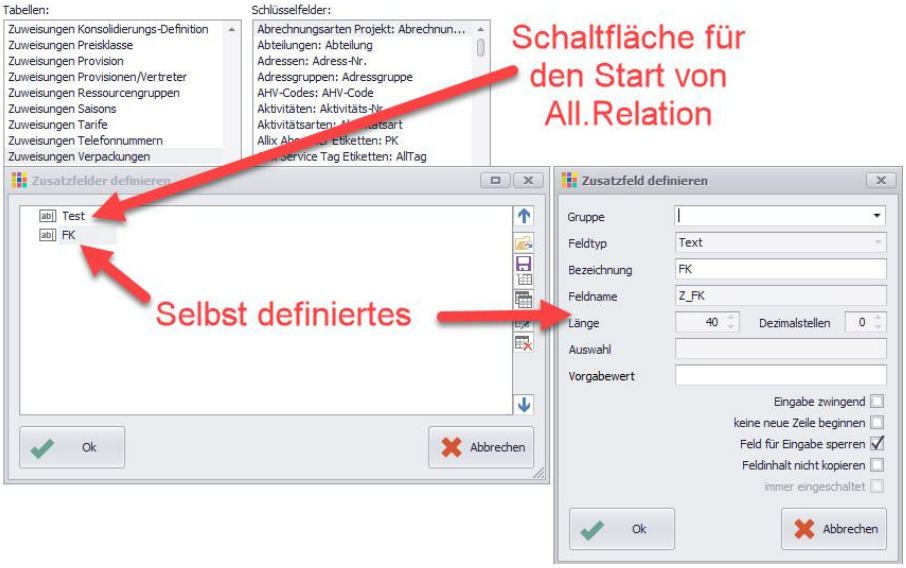

Abbildung 3 «Definition eines Primärschlüsselfeldes als Zusatzfeld»

Weiter muss in der Zusatztabelle ein entsprechendes Feld für den Fremdschlüssel auch mit Datentyp «Text» und wenigstens 32 Zeichen lang definiert werden.

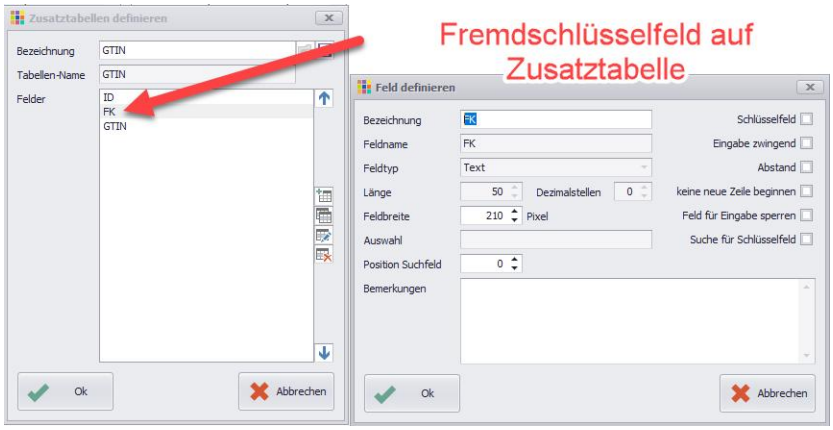

Abbildung 4 «Definition des Fremdschlüssels auf der Zusatztabelle»

In der Datei «EntitySchema.xml» wird nun noch die Konfiguration der Felder wie folgt vorgenommen:

```
<Entity name="Artikel003">
  <PrimaryKey>FK</PrimaryKey>
  <RelationEntity entity="ZUS_GTIN"></RelationEntity>
  <Paramfields>
    <Field typ="PrimaryKey">Z_FK</Field>
  </Paramfields>
  <Indexdefinition active="false">
    <IndexContainerfield></IndexContainerfield>
    <Indexfield fieldname="" query=""></Indexfield>
  </Indexdefinition>
</Entity>
```

Abbildung 5 «Ausschnitt aus EntitySchema.xml»

All.Relation generiert nun selbstständig eine GUID als Primärschlüssel und schreibt diese sowohl in das Zusatzfeld der Muttertabelle wie auch in jeden Datensatz der Detailtabelle.

Konfiguration Teil 2

Öffnen Sie nun über Ihren Button das Programm, wird es Sie auffordern eine Lizenzanfrage zu erstellen. Haken Sie alle Tabellen an, für die Sie eine Lizenz wünschen und ergänzen Sie die Lizenznehmerdaten mit Vor- und Nachname des Bestellers. Klicken Sie anschliessen auf den Button „Manuell“ um die Lizenzanfrage zu erstellen. Kopieren sie die erstelle Anfrage in ein Textfile und senden Sie uns dies per email an info@allix.ch.

Sie erhalten von uns umgehend eine 30 Tage gültige Demolizenz, die Lizenzvereinbarung und die entsprechende Lizenzrechnung. Nach Zahlungseingang und Erhalt der unterschriebenen Lizenzvereinbarung erhalten Sie von uns die unbeschränkt gültige Lizenz. Kopieren Sie das Lizenzfile „AxRelation.liz“ in das PROFFIX Programmverzeichnis um das Programm freizuschalten.

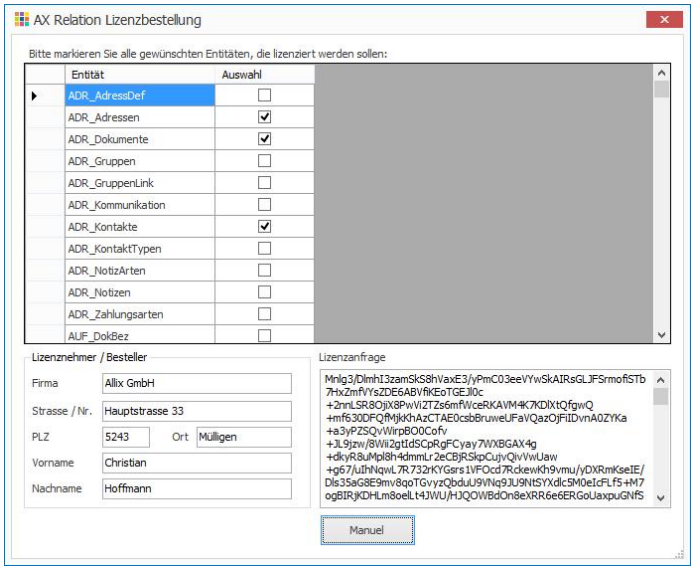

Abbildung 6 „Lizenzanfrage“

Abfragen von relationalen Zusatztabellen in PROFFIX

Wenn relationale Tabellen erstellt wurden, wird evtl. auch der Wunsch vorhanden sein, diese in Selektionen z.B. in einem Serienbrief abfragen zu können. PROFFIX bietet die Möglichkeit, Abfragen mittels SQL zu erstellen. Mittels SQL – Selektion ist es möglich, auf relationale Zusatztabellen zugreifen zu können, um genau die Datensätze abfragen zu können, die den gewünschten Kriterien entsprechen siehe Abbildung 7 „Beispiel einer SQL-Abfrage“.

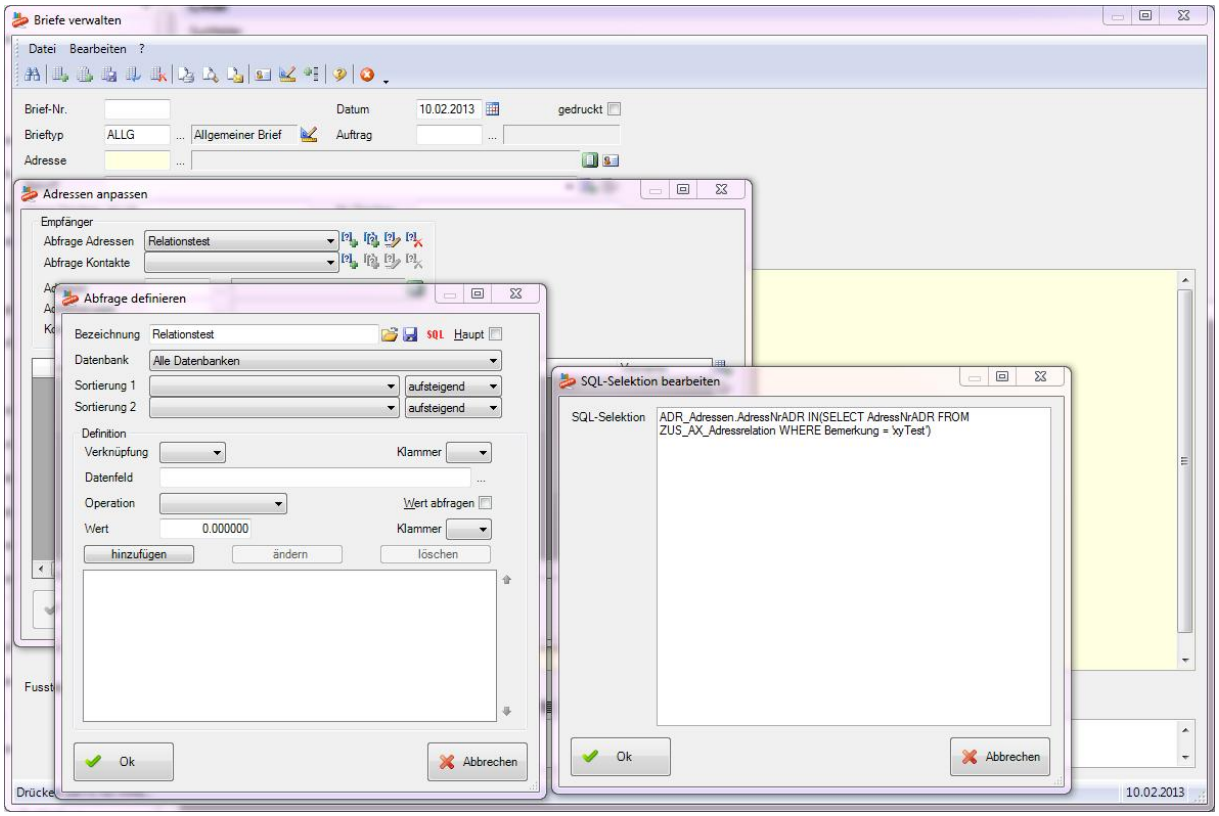

Abbildung 7 „Beispiel einer SQL-Abfrage“

In oben abgebildetem Beispiel sollen alle Adressen in den Serienbrief aufgenommen werden, für die in der relationalen Zusatztabelle „ZUS_AX_Adressrelation“ ein Datensatz besteht und in der Spalte „Bemerkungen“ der erwähnten Zusatztabelle der Wert „xyTest“ steht.

Auf diese Weise kann jede beliebige Selektion, in beliebig tief verschachtelten Relationen erfolgen.