Acumatica
Virtual DevCon
2021
March 10

Understanding & Using Acumatica Attributes

Evgeny Kralko
Integrations Team Lead
ekralko@acumatica.com | acumatica.com/developers

Agenda

- Attributes, what are they for?
- How to modify Attributes
- [PXDefault] vs [PXDBDefault]
- [PXDBCalced] vs [PXDBScalar]
- [PXFormula] vs [PXUnboundFormula]
- [PXProjection] Attribute
- How to create your own attributes

Glossary

- Graph (business logic controller) is a class that serves as the controller for a page and defines the data that is retrieved for the page and the business logic of the page.
- DAC (Data Access Class) is a type that primarily represent database table in the application. A data access class consists of data fields.
- Caches are objects that are created by the system to maintain data records retrieved from the database and the modifications to these data records.
- BQL (business query language) -in Acumatica Framework, you generally use BQL to query data from the database. BQL statements represent specific SQL queries and are translated into SQL.

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [PXDBScalar] [PXActionRestriction] [SubItem] [PXDBBaseCury] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Site] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [PXDBDate] [PXDBShort] [PXDefault] [PXRestrictor] [Branch] [PXProjection] [PXBool] [PXDBDecimal] [PXDimensionSelector] [PXCurrency] [MigratedRecord] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

Attributes, what are they for?

Attributes: What are they for?

C# Attributes

C# Attributes provide a way of associating information with code in a declarative way. They can also provide a reusable element that can be applied to a variety of targets.

Assembly **Module** **Class** **Struct** **Enum** **Constructor** **Method** **Property**
Field **Event** **Interface** **Parameter** **Delegate** **ReturnValue** **GenericParameter**

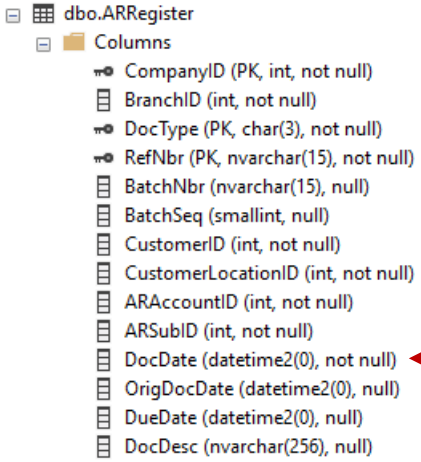
Acumatica Attributes

Basically, Attributes help Acumatica to link Database – Code – UI all together and, of course, to reuse existing logic and save time on development.

Type **Cache** **Item**

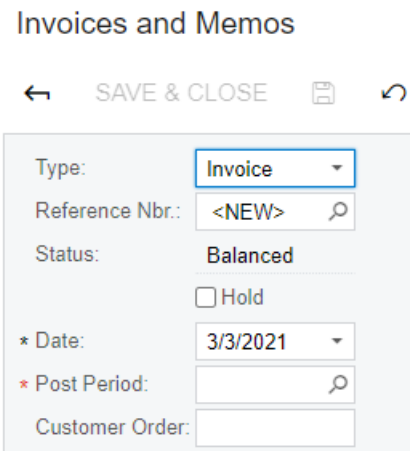
Attributes: What are they for?

Database table (ARRegister) ↔ Data Access Class (ARRegister) ↔ Acumatica Screen (Invoices and Memos)




```
public partial class ARRegister:
   IBqlTable
{
   public abstract class docDate :
      BqlDateTime.Field<docDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Date"]
   [PXDefault(typeof(
      AccessInfo.businessDate))]
   public virtual DateTime? DocDate
   ...
```



Attributes: What are they for?

Using PXDefault Attribute

```
public partial class ARRegister:
   IBqlTable
{
   public abstract class docDate :
      BqlDateTime.Field<docDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Date"]
   [PXDefault(typeof(
      AccessInfo.businessDate))]
   public virtual DateTime? DocDate
   ...
```

~1 line of code

~~Without PXDefault Attribute~~

- Subscribe to FieldSelecting event.
- Subscribe to FieldDefaulting event.
- Subscribe to RowPersisting event.
- and so on…

i You should do it for every Graph and Field where you need to default the value.

~500 lines of code

How to modify Attributes

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedDateTime] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBScalar] [PXActionRestriction] [SubItem] [PXDBBaseCury] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDBDecimal] [PXDimensionSelector] [PXCurrency] [MigratedRecord] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXDBScalar] [PXBaseCury] [PeriodID] [PXDBDefault] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

How to Modify Attributes

Original Attributes

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName =
       "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
    ...
```

CacheAttached event

```
[PXDBDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUIField(DisplayName = "Commission Amount")]
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void
   _(Events.CacheAttached<ARInvoice.commnAmt> e) { }
```

Redefines the whole set of the original Attributes.

How to Modify Attributes

Original Attributes

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName =
       "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
    ...
```

+ PXMergeAttributes

```
[PXMergeAttributes(Method = MergeMethod.Append)]
```

Replace | Append | Merge

```
[PXAdditionalAttribute(NecessaryProperty = true)]
protected virtual void
   _(Events.CacheAttached<ARInvoice.commnAmt> e) { }
```

```
[PXDBDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUIField(DisplayName = "Commission Amount")]
[PXAdditionalAttribute(NecessaryProperty = true)]
public virtual Decimal? CommnAmt
```

How to Modify Attributes

Original Attributes

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName =
       "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
    ...
```

+ PXCustomizeBaseAttribute

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXCustomizeBaseAttribute(typeof(PXUIField),
   nameof(PXUIFieldAttribute.DisplayName),
   "Base Currency Commission")]
protected virtual void
   _(Events.CacheAttached<ARInvoice.commnAmt> e) { }
```

```
[PXDBDecimal(4)]
[PXDefault(TypeCode.Decimal, "0.0")]
[PXUIField(DisplayName = "Base Currency Commission")]
public virtual Decimal? CommnAmt
```

How to Modify Attributes

Original Attributes

```
public class ARInvoice : IBqlTable
{
    [PXDBDecimal(4)]
    [PXDefault(TypeCode.Decimal, "0.0")]
    [PXUIField(DisplayName =
       "Commission Amount")]
    public virtual Decimal? CommnAmt
    {
        get;
        set;
    }
    ...
```

+ PXRemoveBaseAttribute

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXRemoveBaseAttribute(typeof(PXDefaultAttribute))]
protected virtual void
   _(Events.CacheAttached<ARInvoice.commnAmt> e) { }
```

```
[PXDBDecimal(4)]
[PXUIField(DisplayName = "Commission Amount")]
[PXDefault(TypeCode.Decimal, "0.0")]
public virtual Decimal? CommnAmt
```

How to Modify Attributes

Original Attributes

```
[PXDBInt]
[PXSelector(typeof(Search<FAClass.assetID,
   Where<FAClass.recordType,
      Equal<FARecordType.classType>>>))]
[PXUIField(DisplayName = "Asset Class")]
public virtual int? ClassID { get; set; }
```

+ PXRestrictor

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXRestrictor(typeof(Where<FAClass.active, Equal<True>>),
   Messages.InactiveFAClass,
   typeof(FAClass.assetCD))]
protected virtual void
   _(Events.CacheAttached<FAClass.classID> e) { }
```

```
[PXSelector(typeof(Search<FAClass.assetID,
   Where<FAClass.recordType,
      Equal<FARecordType.classType>,
      And<FAClass.active, Equal<True>>>>))]
```

PXRestrictorAttribute does not work alone, it should always be paired with a PXSelectorAttribute.

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [SubItem] [PXDBBaseCury] [PXDBScalar] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDimensionSelector] [PXCurrency] [MigratedRecord] [PXDBDecimal] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

PXDefault vs PXDBDefault

[PXDefault] vs [PXDBDefault]

PXDefault ≠ **PXDBDefault**

Is used to set a default value for a DAC field and check if the value was set (optional) before saving.

Is mainly used to maintain the master-child relationship between records and set a value from the parent DAC field (auto-generated key for example).

[PXParent(typeof(…))]

```
public partial class ARRegister: IBqlTable
{
   public abstract class docDate :
      BqlDateTime.Field<docDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Date"]
   [PXDefault(typeof(
      AccessInfo.businessDate))]
   public virtual DateTime? DocDate
   ...
```

```
public partial class ARTran: IBqlTable
{
   public abstract class tranDate :
      BqlDateTime.Field<tranDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Document Date"]
   [PXDBDefault(typeof(ARRegister.docDate))]
   public virtual DateTime? DocDate
   ...
```

[PXDefault] vs [PXDBDefault]

PXDefault ≠ **PXDBDefault**

- Set default value on FieldDefaulting event.
- Check if value was set on RowPersisting event.

- Set default value on FieldDefaulting event.
- Redefault value on RowPersisting event.
- Rollback changes if the record was not saved.

[PXParent(typeof(…))]

```
public partial class ARRegister: IBqlTable
{
   public abstract class docDate :
      BqlDateTime.Field<docDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Date"]
   [PXDefault(typeof(
      AccessInfo.businessDate))]
   public virtual DateTime? DocDate
   ...
```

```
public partial class ARTran: IBqlTable
{
   public abstract class tranDate :
      BqlDateTime.Field<tranDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Document Date"]
   [PXDBDefault(typeof(ARRegister.docDate))]
   public virtual DateTime? DocDate
   ...
```

[PXDefault] vs [PXDBDefault]

PXDefault

For the DB field use PXDefault Attribute.

```
public partial class ARRegister: IBqlTable
{
   public abstract class docDate :
      BqlDateTime.Field<docDate> { }

   [PXDBDate]
   [PXUIField(DisplayName = "Date"]
   [PXDefault(typeof(
      AccessInfo.businessDate))]
   public virtual DateTime? DocDate
   ...
```

PXUnboundDefault

For an unbound field use PXUnboundDefault Attribute.

```
public partial class Branch: IBqlTable
{
   public abstract class included :
      BqlBool.Field<included> { }

   [PXUnboundDefault(false,
      PersistingCheck = PXPersistingCheck.Nothing)]
   [PXBool]
   [PXUIField(DisplayName = "Included")]
   public virtual bool? Included
   ...
```

PXDBCalced vs PXDBScalar

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedDateTime] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBScalar] [SubItem] [PXDBBaseCury] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDimensionSelector] [PXCurrency] [MigratedRecord] [PXDBDecimal] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXInt] [PXDBLastModifiedByID] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

[PXDBCalced] vs [PXDBScalar]

PXDBCalced ≠ **PXDBScalar**

Is defines the SQL expression that calculates an unbound field from the fields of the same DAC whose values are taken from the database.

```
public partial class ARInvoice: IBqlTable
{
   public abstract class emailInvoice :
      BqlBool.Field<emailInvoice> { }

   [PXBool]
   [PXDBCalced(typeof(
      Switch<Case<Where<dontEmail, Equal<False>,
         And<emailed, Equal<False>>>,
            True>, False>), typeof(Boolean))]
   public virtual bool? EmailInvoice
   ...
```

Is defines a separate SQL subrequest! that will be used to retrieve the value for the unbound DAC field.

```
public partial class Branch: IBqlTable
{
   public abstract class baseCuryID :
      BqlString.Field<baseCuryID> { }

   [PXDBScalar(
      typeof(Search<Company.baseCuryID>))
   [PXString(5, IsUnicode = true)]
   [PXUIField(DisplayName = "Base Currency ID")]
   [PXSelector(typeof(Search<Currency.curyID>))]
   public virtual String BaseCuryID
   ...
```

[PXDBCalced] vs [PXDBScalar]

✓ **PXDBCalced** ≠ ! **PXDBScalar**

If, in contrast, you need to calculate the field on the server side at run time, use the PXFormula Attribute.

```
SELECT
[ARInvoice_ARRegister].[DocType] AS [DocType],
[ARInvoice_ARRegister].[RefNbr] AS [RefNbr],
...
(CASE WHEN ([ARInvoice_ARInvoice].[DontEmail] =
    CONVERT (BIT, 0)
AND [ARInvoice_ARInvoice].[Emailed] =
   CONVERT (BIT, 0))
THEN CONVERT (BIT, 1) ELSE CONVERT (BIT, 0) END)
AS [EmailInvoice],
...
FROM ([ARInvoice] AS [ARInvoice_ARInvoice]
INNER JOIN [ARRegister] AS [ARInvoice_ARRegister]
```

Use it very carefully, may cause performance degradation.

```
SELECT TOP (1)
[Branch].[OrganizationID],
[Branch].[BranchID],
...
(SELECT TOP (1) [Company].[BaseCuryID]
FROM [Company] AS [Company]
WHERE ([Company].[CompanyID] = 2)
ORDER BY [Company].[BaseCuryID]),
...
FROM [Branch] AS [Branch]
```

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedDateTime] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBScalar] [PXActionRestriction] [SubItem] [PXDBBaseCury] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDimensionSelector] [PXDBDecimal] [PXCurrency] [MigratedRecord] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXInt] [PXDBLastModifiedByID] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXQuantity] [PXCustomizeBaseAttribute] [PXDBCreatedByScreenID]

PXFormula vs PXUnboundFormula

[PXFormula] vs [PXUnboundFormula]

PXFormula Attribute

Automatically calculates a field from other fields of the same data* record and sets an aggregation formula to calculate a parent data record field from child data record fields.

- On field defaulting (inserting a new row; FieldDefaulting event of formula field)
- On updating of dependent fields (FieldUpdated event of each dependent field)
- On database selection (only for unbound fields; RowSelecting event)
- On database persisting (optional; RowPersisted event)

```
public partial class INTran: IBqlTable
{
    public abstract class tranCost :
        BqlDecimal.Field<tranCost> { }

    [PXFormula(
        typeof(Mult<INTran.qty, INTran.unitCost>))]
    [PXDBBaseCury(MinValue = 0)]
    [PXDefault(TypeCode.Decimal,"0.0")]
    [PXUIField(DisplayName = "Ext. Cost")]
    public virtual Decimal? TranCost
    ...
```

[PXFormula] vs [PXUnboundFormula]

PXFormula Attribute

*By default, anything written inside a PXFormulaAttribute is presumed to exist in the context of the current record. In order to leave the default context and access other data records, you need to use special context modifiers.

```
[PXMergeAttributes(Method = MergeMethod.Append)]
[PXRemoveBaseAttribute(typeof(PXDefaultAttribute))]
[PXFormula(typeof(
   IIf<Where<Current<APInvoice.docType>,
         NotEqual<APDocType.debitAdj>,
      And<Current2<APInvoice.docType>,
         NotEqual<APDocType.prepayment>>>,
      Selector<APInvoice.vendorID, Vendor.termsID>,
      Null>))]
protected virtual void
   APRecognizedInvoice_TermsID_CacheAttached(
      PXCache sender) { }
```

Current<TRecord.field>

Current2<TRecord.field>

Selector<KeyField, ForeignOperand>

IsTableEmpty<TRecord>

Parent<TParent.field>

…

[PXFormula] vs [PXUnboundFormula]

PXFormula Attribute

+ aggregate

! The order of fields is important. All dependent fields must be defined in the DAC before the formula field.

Developers can create their own aggregation

SumCalc | CountCalc | MinCalc | MaxCalc

[PXParent(typeof(…))]

```
public partial class INRegister: IBqlTable
{
   public abstract class totalCost :
      BqlDecimal.Field<totalCost> { }

   [PXDBBaseCury]
   [PXDefault(TypeCode.Decimal, "0.0")]
   [PXUIField(DisplayName = "Total Cost")]
   public virtual Decimal? TotalCost
   ...
```

```
public partial class INTran: IBqlTable
{
   public abstract class tranCost :
      BqlDecimal.Field<tranCost> { }

   [PXFormula(
      typeof(Mult<INTran.qty, INTran.unitCost>),
      typeof(SumCalc<INRegister.totalCost>))]
   [PXDBBaseCury(MinValue = 0)]
   [PXDefault(TypeCode.Decimal,"0.0")]
   [PXUIField(DisplayName = "Ext. Cost")]
   public virtual Decimal? TranCost
   ...
```

[PXFormula] vs [PXUnboundFormula]

PXFormula ≠ **PXUnboundFormula**

- Calculate the value (2 qty * 100).
- Set value.
- Calculate aggregate (3 records).
- Set value to the parent field.

Calculate the value (2 qty * 100).
~~Set value.~~
Calculate aggregate (3 records).
Set value to the parent field

```
public partial class INTran: IBqlTable
{
   public abstract class tranCost :
      BqlDecimal.Field<tranCost> { }

   [PXFormula(
      typeof(Mult<INTran.qty, INTran.unitCost>),
      typeof(SumCalc<INRegister.totalCost>))]
   [PXDBBaseCury(MinValue = 0)]
   [PXDefault(TypeCode.Decimal,"0.0")]
   [PXUIField(DisplayName = "Ext. Cost")]
   public virtual Decimal? TranCost
   ...
```

= 600

= 200

```
public partial class INTran: IBqlTable
{
   public abstract class tranCost :
      BqlDecimal.Field<tranCost> { }

   [PXUnboundFormula(
      typeof(Mult<INTran.qty, INTran.unitCost>),
      typeof(SumCalc<INRegister.totalCost>))]
   [PXDBBaseCury(MinValue = 0)]
   [PXDefault(TypeCode.Decimal,"0.0")]
   [PXUIField(DisplayName = "Ext. Cost")]
   public virtual Decimal? TranCost
   ...
```

= 600

= 0

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [SubItem] [PXDBBaseCury] [PXDBScalar] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDBDecimal] [PXDimensionSelector] [PXDBShort] [PXCurrency] [MigratedRecord] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXDBScalar] [PXBaseCury] [PeriodID] [PXDBDefault] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

PXProjection Attribute

[PXProjection] Attribute

PXProjection Attribute

The PXProjectionAttribute (also referred to as the projection) binds the DAC to an arbitrary data set defined by the BQL Select command. The Attribute defines a named view but implemented by the server side rather then the database. The Attribute is placed on the DAC declaration.

- Reducing column count.
- Filtering rows.
- Retrieving aggregates.
- Persisting data.
- and so on…

[PXProjection] Attribute

Reducing column count

You can create a new PXProjection with only needed fields from one or several tables, remove redundant attributes and so on.

```
[PXProjection(typeof(Select<APAdjust>))]
public partial class APAdjustX : IBqlTable
{
   public abstract class adjgDocType :
      BqlString.Field<adjgDocType> { }

   [PXDBString(3,
      IsKey = true, IsFixed = true,
      InputMask = "",
      BqlField = typeof(APAdjust.adjgDocType))]
   [PXUIField(DisplayName = "AdjgDocType")]
   public virtual string AdjgDocType
   ...
}
```

To map a field, you should set the BqlField property of the type attribute.

[PXProjection] Attribute

Filtering rows

You can create a new PXProjection to filter records by needed Bql condition and then reuse it in several places.

```
[PXProjection(typeof(Select<Vendor,
   Where<Vendor.payToVendorID,
      Equal<CurrentValue<Vendor.bAccountID>>>>))]
public class SuppliedByVendor : Vendor { }
```

! You can inherit projection from one of the DACs in the Select command.

! To get the current value stored in the cache – you should use CurrentValue modifier instead of ~~Current~~ or ~~Current2~~ for DAC.

[PXProjection] Attribute

Retrieving aggregates

The GLHistoryByPeriod projection is used to simplify selection and aggregation of proper GLHistory records on various inquiry and processing screens of the General Ledger module.

```
[PXProjection(typeof(Select5<GLHistory,
   InnerJoin<FinPeriod, On<FinPeriod.finPeriodID,
      GreaterEqual<GLHistory.finPeriodID>>>,
   Aggregate<
      GroupBy<GLHistory.branchID,
      GroupBy<GLHistory.ledgerID,
      GroupBy<GLHistory.accountID,
      GroupBy<GLHistory.subID,
      Max<GLHistory.finPeriodID, //LastActivityPeriod
      GroupBy<FinPeriod.finPeriodID>>>>>>>>))]
public partial class GLHistoryByPeriod : IBqlTable
```

! The name of the DAC with PXProjectionAttribute must be unique.

[PXProjection] Attribute

Persisting data

Additionally, you can use the constructor with two parameters to provide the list of updatable tables explicitly. This list should include the tables referenced in the Select command. This constructor also sets the Persistent property to true.

```
[PXProjection(typeof(Select2<GLTran,
   LeftJoin<FAAccrualTran, On<GLTran.tranID,
      Equal<FAAccrualTran.tranID>>>,
   Where<GLTran.module, NotEqual<BatchModule.moduleFA>,
      And<GLTran.released, Equal<True>>>>),
   new Type[]{typeof(FAAccrualTran)}, Persistent = true)]
public partial class FAAccrualTran : IBqlTable
{
   public abstract class adjgDocType :
      BqlString.Field<adjgDocType> { }
   [PXDBInt(BqlField = typeof(FAAccrualTran.tranID))]
   [PXDefault]
   [PXExtraKey]
   public virtual int? TranID
   ...
}
```

! By default, the projection is read-only, but you can make it updatable by setting the Persistent property to true.

! Only the first table is updated by default. If the data should be committed in joined tables, the fields that connect the tables must be marked with PXExtraKeyAttribute.

How to create your own attribute

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [SubItem] [PXDBBaseCury] [PXDBScalar] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDBDecimal] [PXDimensionSelector] [PXDBShort] [PXCurrency] [MigratedRecord] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXQuantity] [PXCustomizeBaseAttribute] [PXDBCreatedByScreenID]

How to Create your own Attributes

PXEventSubscriberAttribute

→ Derive a new Attribute class from PXEventSubscriberAttribute.
→ Implement the constructor.
→ override the CacheAttached method (if you read parameters or data from the database).
→ Implement interfaces that correspond to the events.

- IPXRowSelectingSubscriber
- IPXRowSelectedSubscriber
- IPXRowInsertingSubscriber
- IPXRowInsertedSubscriber
- IPXRowUpdatingSubscriber
- IPXRowUpdatedSubscriber
- IPXRowDeletingSubscriber
- IPXRowDeletedSubscriber
- IPXRowPersistingSubscriber
- IPXRowPersistedSubscriber

- IPXFieldSelectingSubscriber
- IPXFieldDefaultingSubscriber
- IPXFieldUpdatingSubscriber
- IPXFieldUpdatedSubscriber
- IPXFieldVerifyingSubscriber

- IPXCommandPreparingSubscriber

Your own attribute is in progress

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [SubItem] [PXDBBaseCury] [PXDBScalar] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDimensionSelector] [PXCurrency] [MigratedRecord] [PXDBDecimal] [PXDBShort] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXInt] [PXDBLastModifiedByID] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXQuantity] [PXCustomizeBaseAttribute] [PXDBCreatedByScreenID]

Summary

- Acumatica Attributes is a powerful tool that helps to reuse the same behavior or business logic in multiple places of the application and save time on development.
- There are hundreds of useful Attributes out of the box.
- Developers can easily create their own Attributes to cover any requirements.

Call to Action

- Get Involved in our Developer Community!
- Join us on **stackoverflow**: https://stackoverflow.com/questions/tagged/acumatica/
- Check https://www.acumatica.com/developers/ regularly!
- Read our developer blogs: https://www.acumatica.com/blog/category/developers/
- Contribute to Community Projects: https://github.com/acumatica/
- Attend Summit 2021 | Hackathon 2021: https://summit.acumatica.com/
- eMail (mfranks@acumatica.com) if you have any questions or have ideas to improve

[PXUnboundFormula] [PXDBCreatedByID] [PXDBLastModifiedByScreenID] [PXLineNbr] [PXDBLastModifiedDateTime] [SubItem] [PXDBBaseCury] [PXDBScalar] [PXActionRestriction] [PXString] [PXDBIdentity] [LocationID] [INUnit] [PXDBBool] [PXDecimal] [Thanks] [PXDBLong] [PXRemoveBaseAttribute] [Site] [PXDBDate] [PXRestrictor] [PXDBShort] [PXDefault] [PXBool] [Branch] [PXProjection] [PXDBDecimal] [PXDimensionSelector] [PXDBShort] [PXCurrency] [MigratedRecord] [PXForeignReference] [PXDBPriceCost] [PXLong] [PXCacheName] [PXBaseCury] [PXDBScalar] [PXDBDefault] [PeriodID] [ARDocType.List] [PXDBCalced] [PXDBLastModifiedByID] [PXInt] [Account] [PXDBQuantity] [PXDBCreatedDateTime] [PXDBCurrency] [PXDBString] [PXUIField] [CurrencyInfo] [PXDBTimestamp] [PXDBGuid] [PXShort] [PXNote] [PXSelector] [PXFormula] [Inventory] [PXParent] [SubAccount] [PXDate] [YourOwnAttribute] [PXDBInt] [PXDBCalced] [PXMergeAttributes] [PXCustomizeBaseAttribute] [PXQuantity] [PXDBCreatedByScreenID]

Evgeny Kralko

Integrations Team Lead

ekralko@acumatica.com | acumatica.com/developers