Aras Innovator 2025 Release

Query Builder Guide

Document #: *D-008124*

Last Modified: *5/27/2025*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Phone: 978-691-8900

E-mail: support@aras.com

Website: https://www.aras.com/

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
TechDocs@aras.com
Subject: Aras Product Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Technical Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support/

1 Overview

The Query Builder application enables administrators to build reusable AML queries using a graphical user interface (GUI). The intended use of the Query Builder application is to create queries that can be used in other applications, e.g., Tree Grid Views, Graph Navigation, Access Control, etc. Query Builder can be thought of as the common mechanism that allows users to define and execute queries against Aras Innovator data and data structures to provide useful results for any application needing it.

A Query can be defined to traverse a Relationship structure downward as in a sub-assembly, or upward in a “where-used” manner depending on business requirements. In addition, results can be refined using Where Conditions to produce highly selective output. User-defined Parameters allow input of search parameters to be deferred until runtime.

Specific property values to be returned from the Query can be defined, including xProperties. Output can be sorted according to user-specified order. (For more information about xProperties and xClasses, refer to the Aras Innovator Extended Classification document.)

The Query Builder User Interface includes an **Execute Query** function that displays query results to a subwindow to provide a preview of Query results. This allows end users to test their query, examine the results, and refine their search as needed.

A built-in **Exists()** function allows users to filter items based on the existence or absence of related items. Users can also duplicate Query Definition items and use them to create a Query Definition that is similar to the original.

A common usage of Query Builder results is to be formatted and displayed in a Tree Grid View. The Tree Grid View can generate a Relationship Tab for an ItemType where end users can access the structure display easily. (For more information on the Tree Grid View, refer to the *Aras Innovator 2025 Release – Tree Grid View Administrator Guide*.)

This guide walks through building the following sample query:

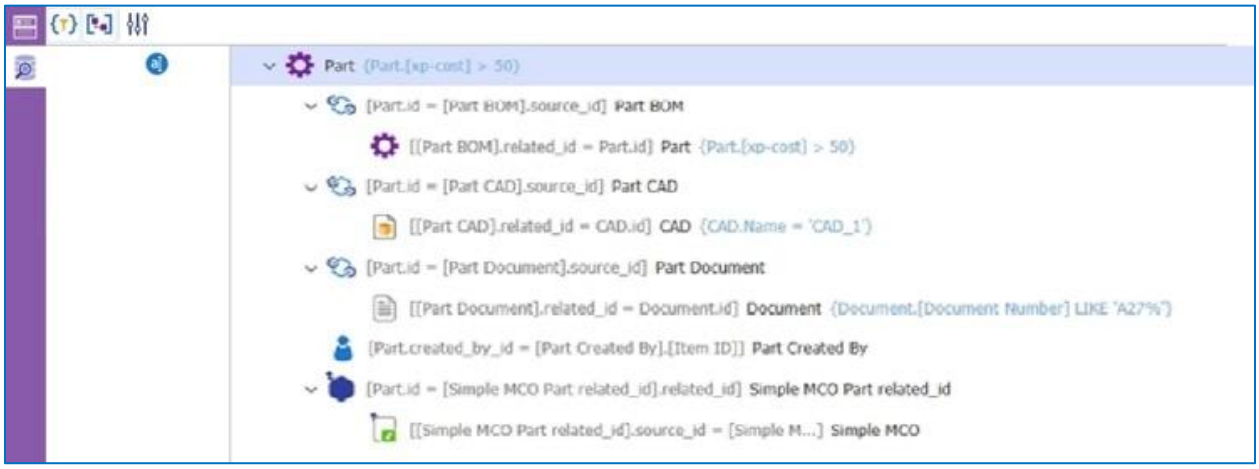

Figure 1.

2 Developing Query Definitions

Reusable queries are defined by the Query Definition item and found as follows: **Cocntnents** --> **Administration** --> **Configuration** --> **Query Definitions**.

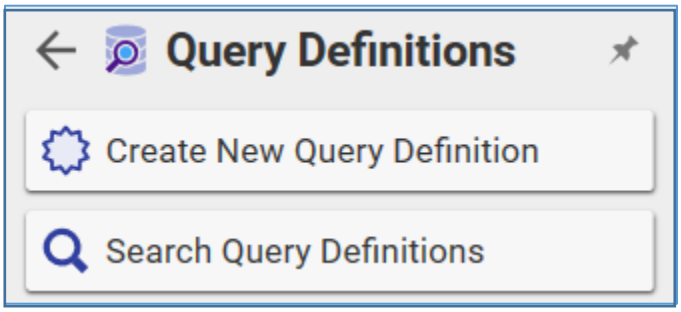

Figure 2.

When you click **Create New Query Definition**, the dialog shown in Figure 3 appears.

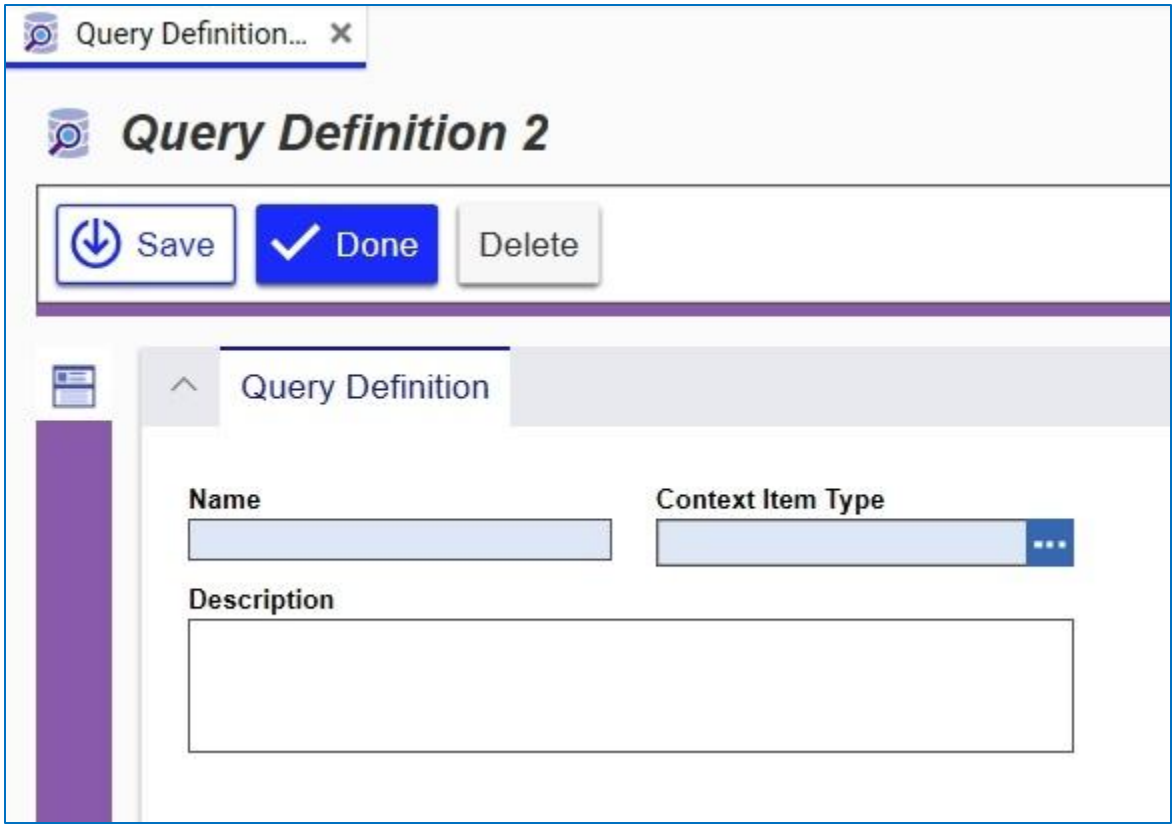

Figure 3.

Each Query Definition starts with an initial Context Item before building it out to a more extensive structure. This specifies the *root* of the query, from where query of structures and property values will begin.

The basic steps for creating a Query Definition are as follows:

1. Define the Context Item.
2. Build the Relationship Structure.
3. Apply Search Conditions.

2.1 Defining a Context Item

The following example creates a **Query Definition** Item for the **Part** ItemType as a Context Item. Create a new query definition and specify a unique **Name** and a **Context Item Type**.

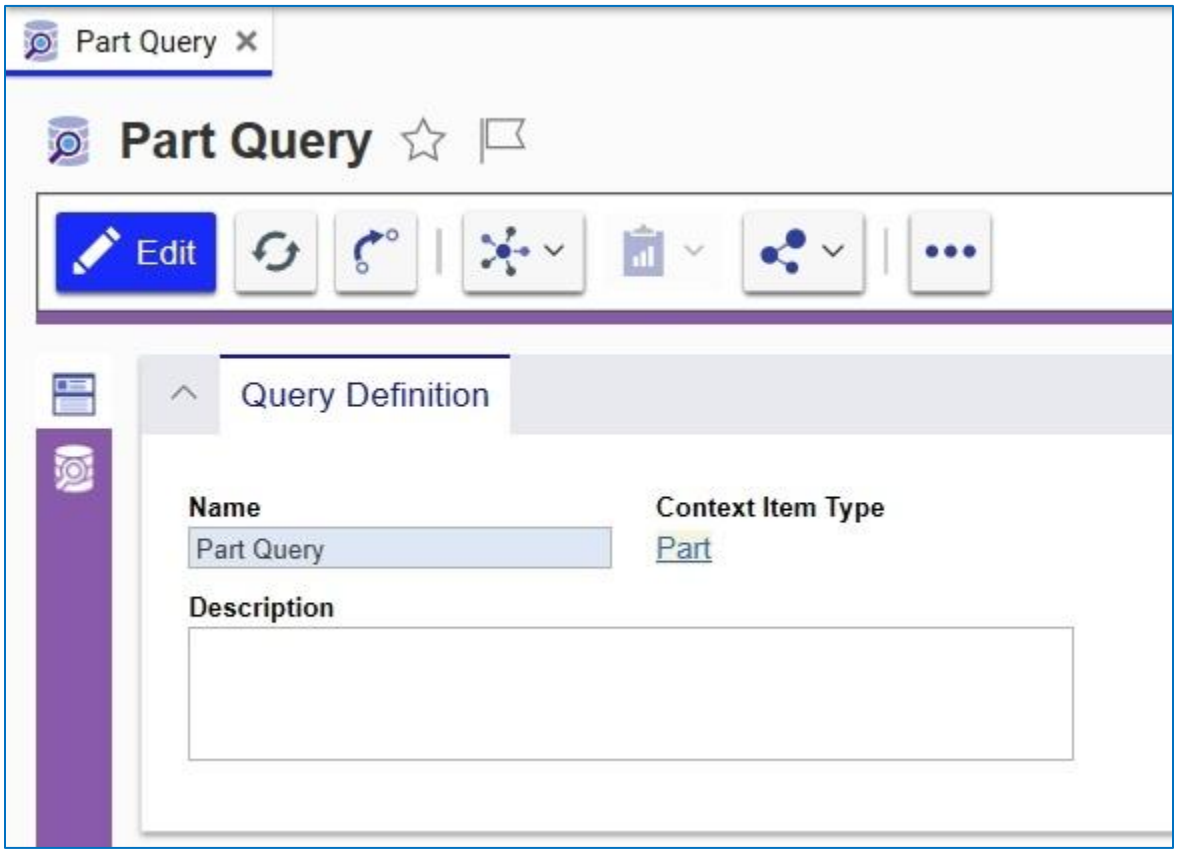

Figure 4.

1. Click 

. The Show Editor icon appears in the ribbon.
2. Click the **Show Editor** icon on the left sidebar to go into Query-Editing Mode.

3. Right-click the Context Item and select **Edit Query Element** --> **Selected Properties**.

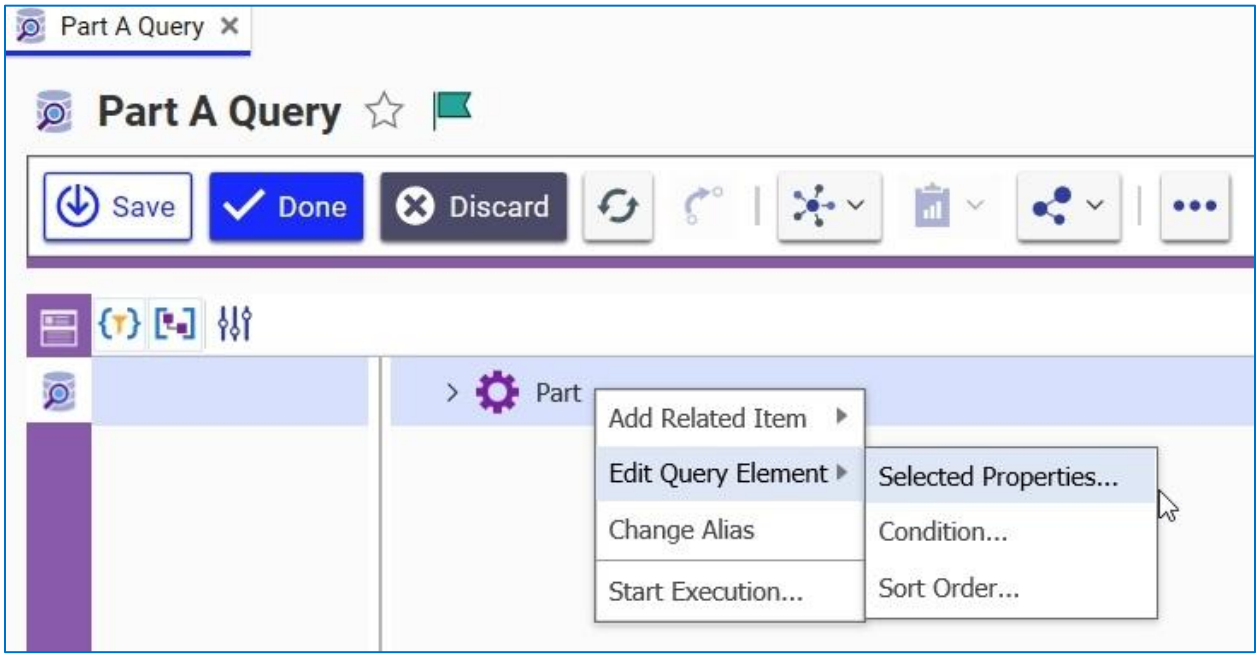

Figure 5.

Select the desired properties that the query should return. In this example they are: `id`, `Part Number`, `Name`, `State`, and `managed_by_id`.

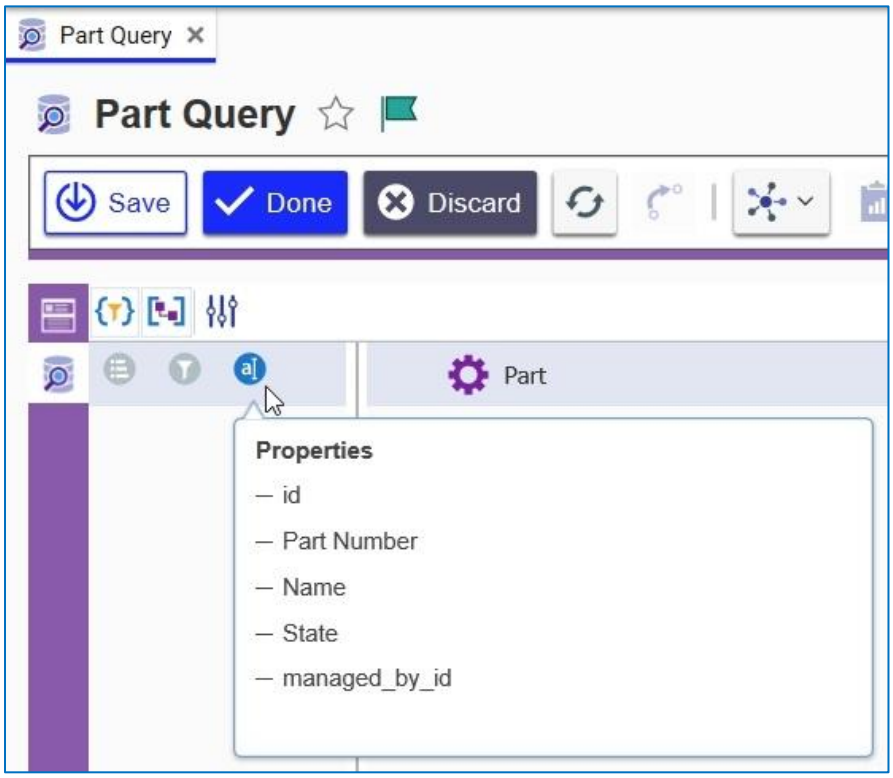

Figure 6.

Extended properties enable you to add values such as cost to an object in the database.

4. Right-click on the Context Item again and select **Edit Query Element --> Sort Order**.
5. Choose the properties that should be used for the Order By command and specify whether they should be in ascending or descending order.

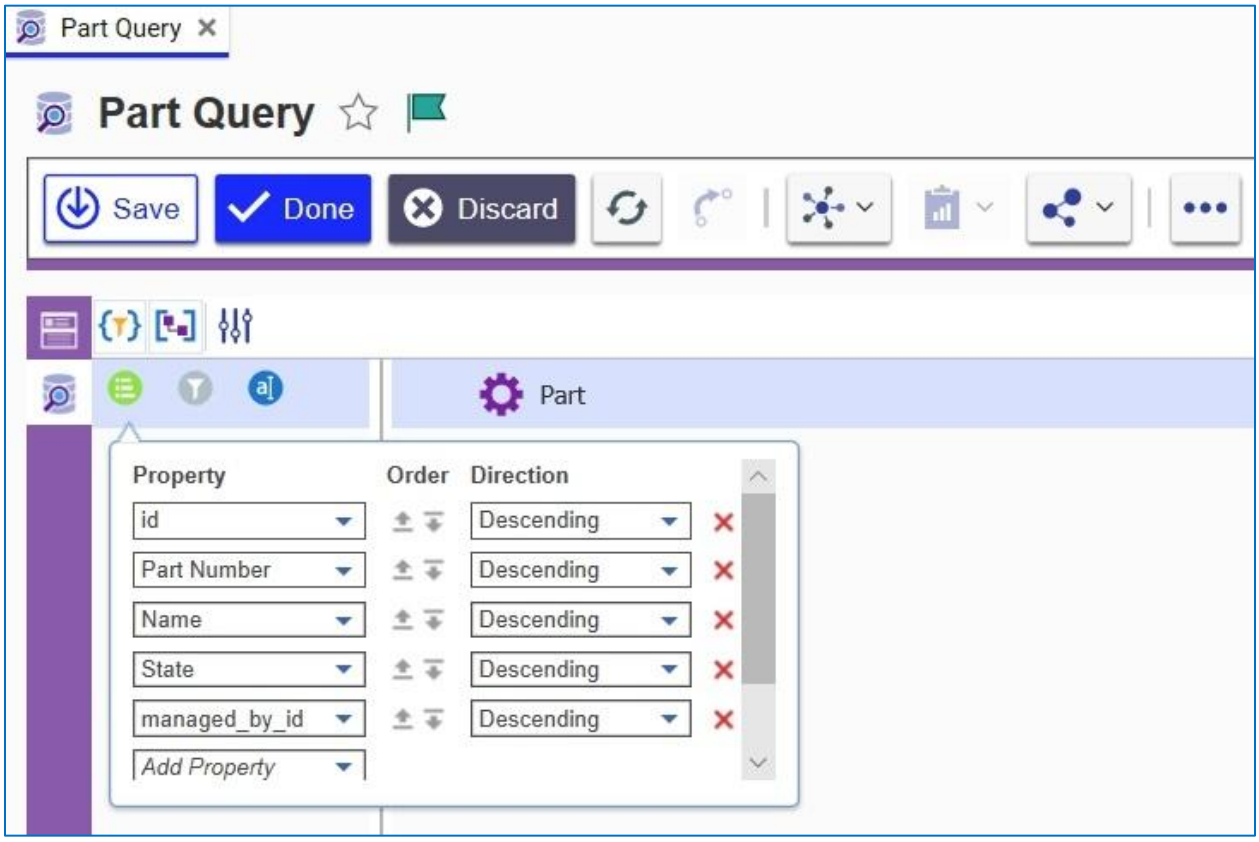

Figure 7.

6. Save the Query.

2.2 Building a Relationship Structure

It is possible to define a Relationship from the context item that can go either up or down the data structure. ‘Down’ refers to a relationship direction starting from the Item representing the *source* side of the relationship and following the related content to the *related* Item. ‘Up’ by contrast refers to a relationship direction starting from the Item representing the *related* side of the relationship and following the related content to the *source* Item. It is also possible to set up a recursive structure, such as the Part -> Part BOM --> Part, to reuse the same logic for multiple levels. The following procedures show how to expand the query both up and down for a Part Query and introduces recursion.

2.2.1 Building a Downward Relationship Structure

1. Right-click on the Part Context Item and select **Add Related Item** --> **Using Relationship**.

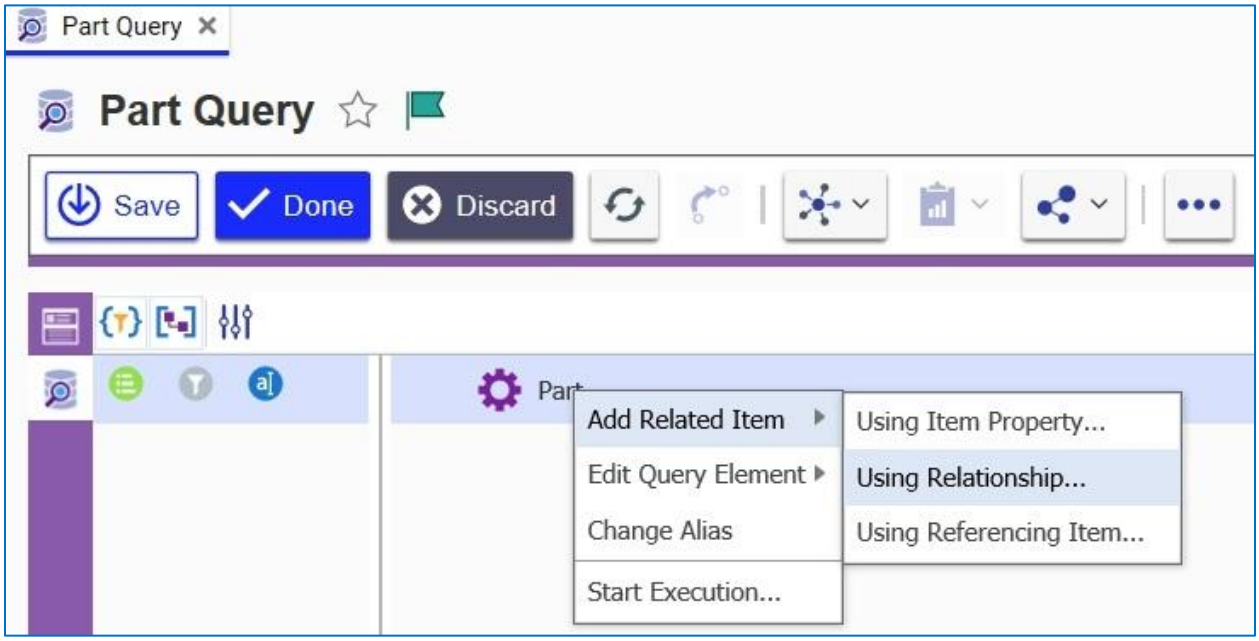

Figure 8.

2. Select the Relationship to include in the structure. In this case, select **BOM,** check **Include Related** and click **Add**.

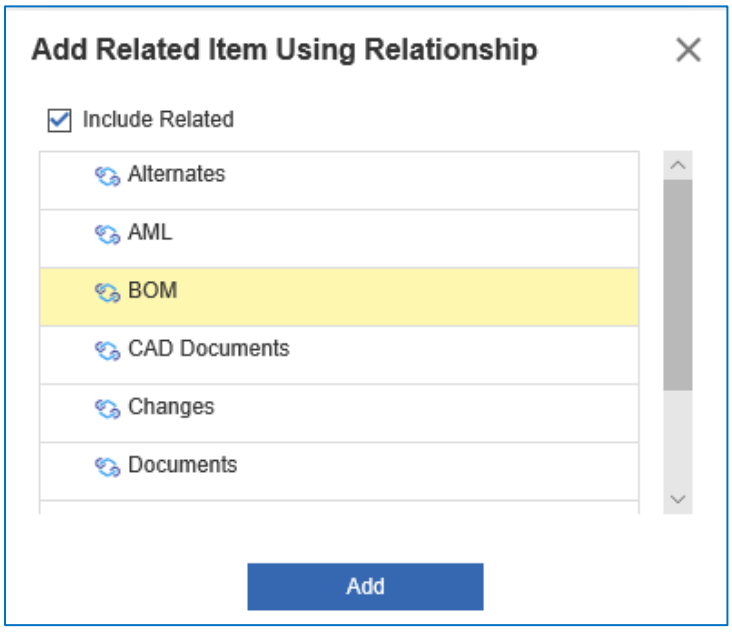

Figure 9.

The query now looks like shown in figure 10. Note that the **Part_1** alias is assigned to the related Part Query Element in the list.

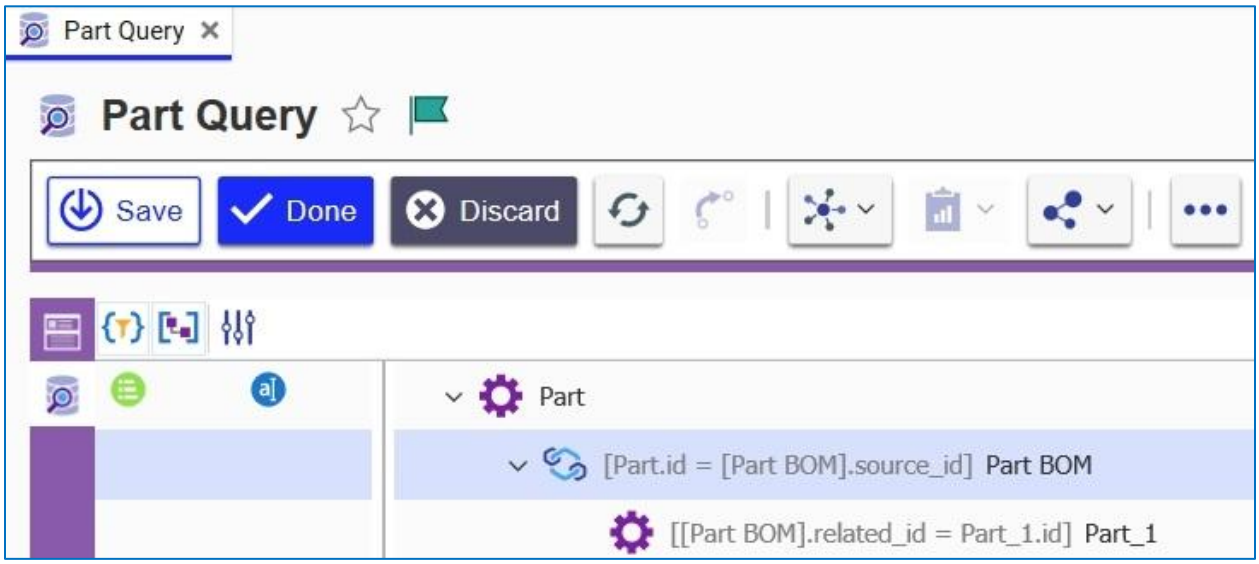

Figure 10.

3. Right-click the **Part** Context Item and click **Add Related Item** --> **Using Relationship**. In this case, click while pressing Ctrl to select **Documents** and **CAD Documents**. Select **Include Related**.

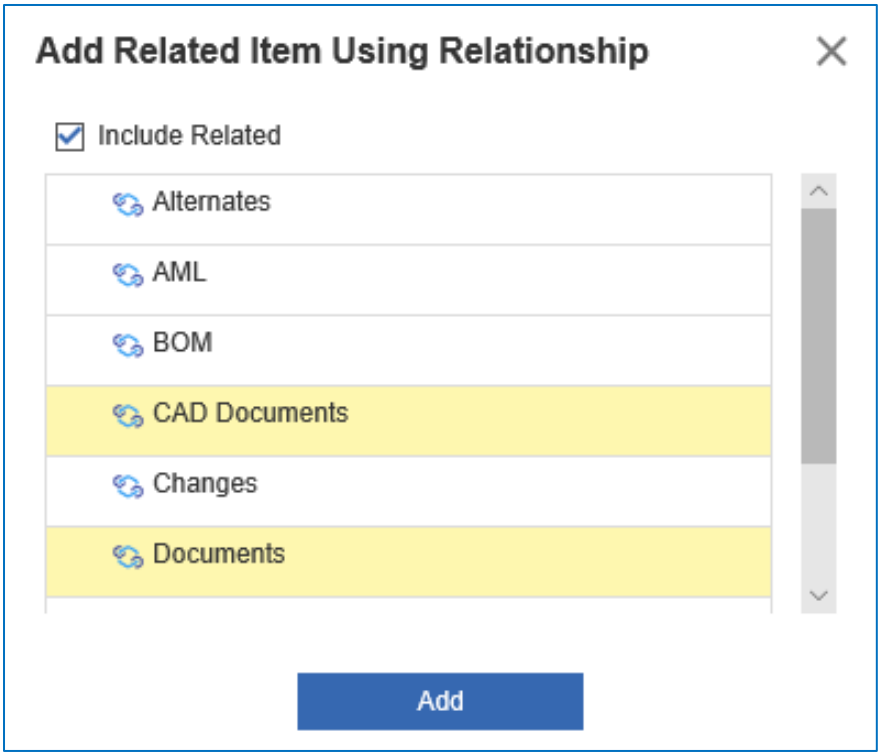

Figure 11.

4. Right-click on the top Context Item and select **Add Related Item** --> **Using Item Property**. Select an Item Property to add to the query structure. In this case, select **created_by_id**.

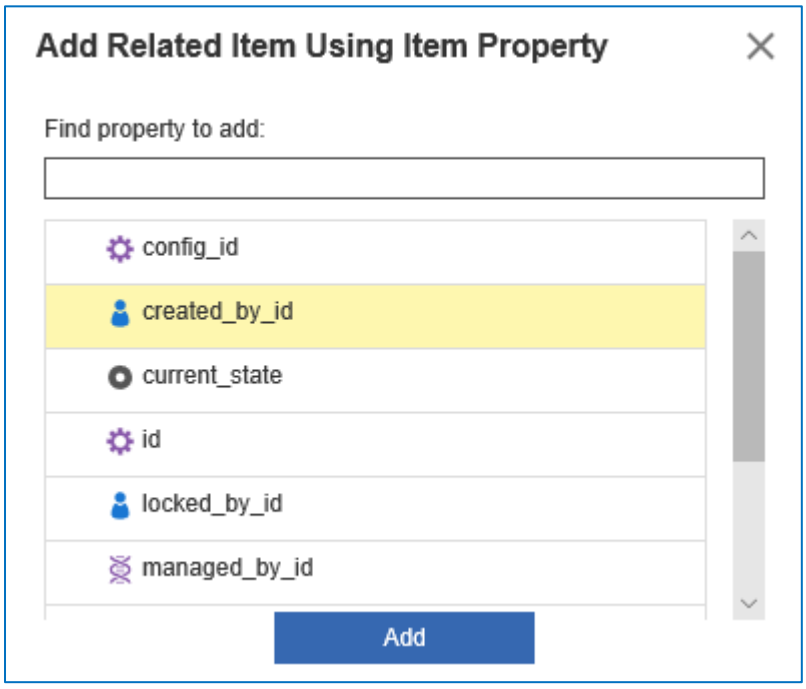

Figure 12.

5. Use the following procedure to specify the desired Properties for each new element in the Query Definition:

Note: Optionally, you can also set Order By settings as well.

a. For [created_by_id] User, specify the following properties:
   - `Item ID`
   - `First Name`
   - `Last Name`
   - `Login Name`

b. For [Relationship] Part CAD, specify the following property:
   - `Sequence`

c. For [Related] CAD, specify the following properties:
   - `id`
   - `Document Number`
   - `Name`
   - `State`

d. For [Relationship] Part Document, specify the following property:
   - `Sequence`

e. For [Related] Document, specify the following properties:

- id
- Document Number
- Name
- State

6. Change the Alias on the **CAD Documents** element to “Part CAD”:
   a. Right-click **CAD Documents** and click **Change Alias**.

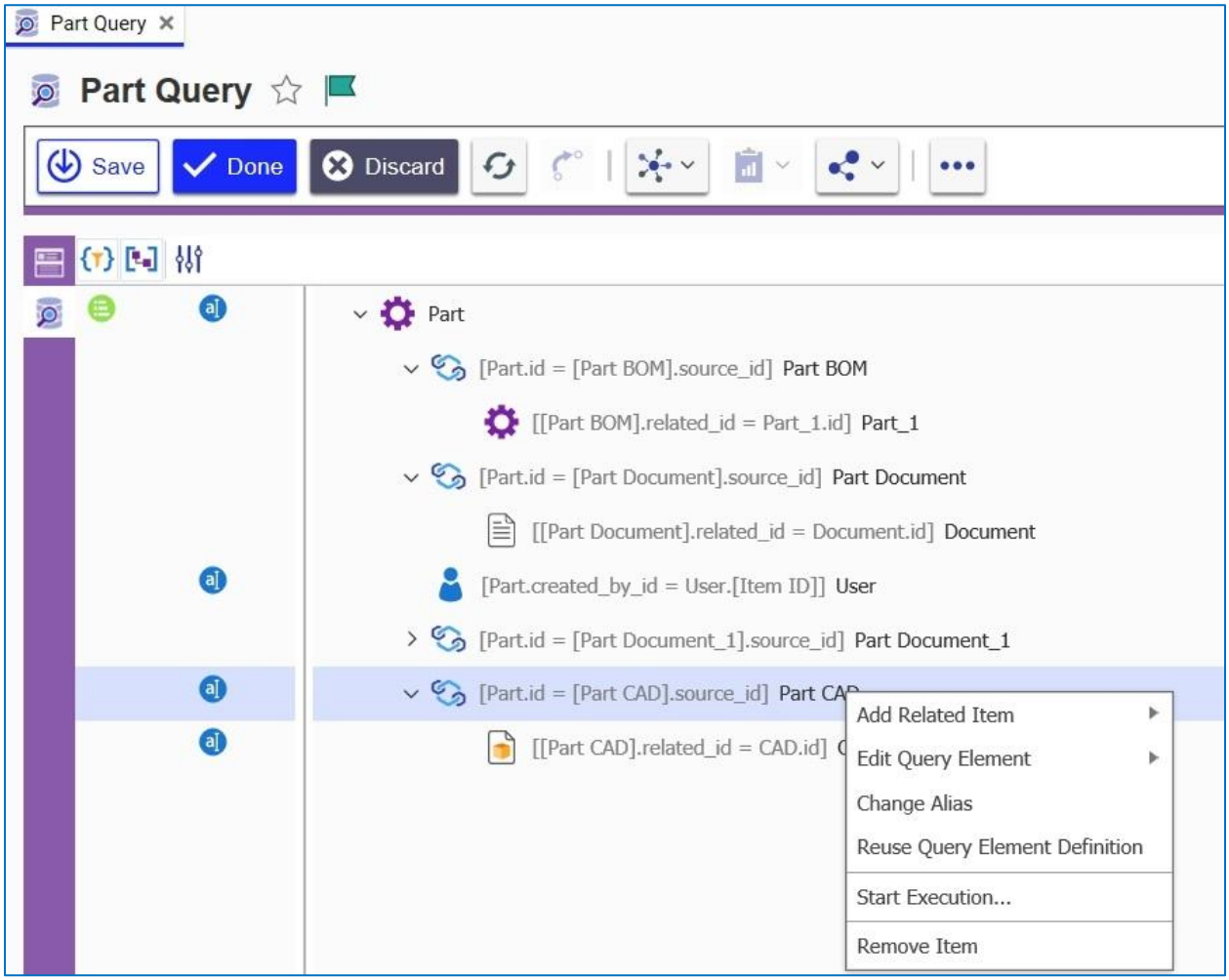

Figure 13.

b. Enter **Part CAD** as the name and click the check mark.

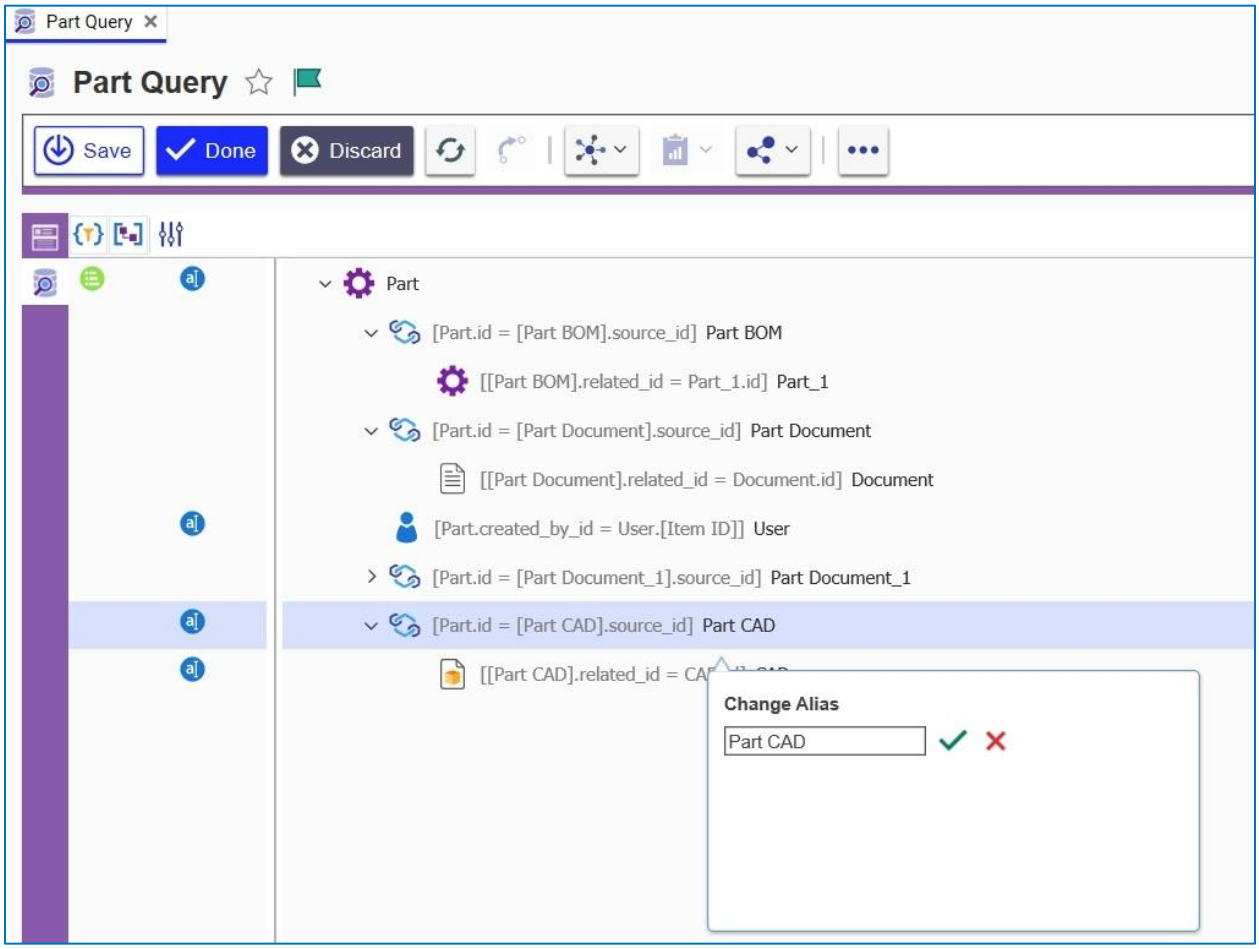

Figure 14.

7. Use the same procedure to change **Documents** Relationship to **Part Document**.
8. Use the same procedure to change **User** Relationship to **Part Created By**.

Note: Each element on a given level must have a unique alias.

9. Save the Query Definition.

2.2.2 Building an Upward Relationship Structure

1. Right-click the top **Part** element and click **Add Related Item** --> **Using Referencing Item**.

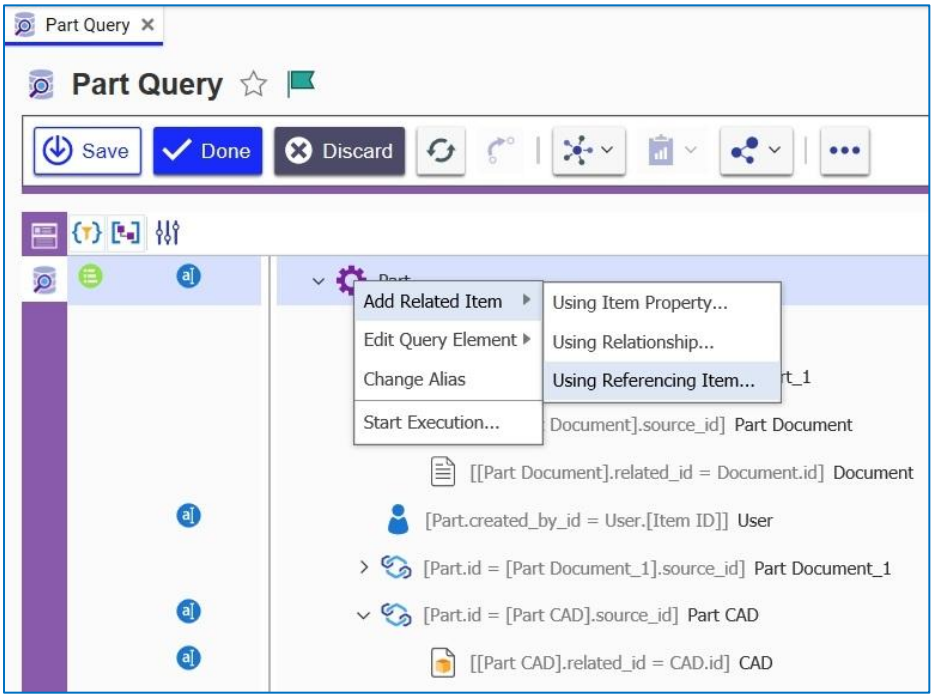

Figure 15.

2. Select an appropriate Referencing item from the structure. In this case, select **Simple MCO Part**.

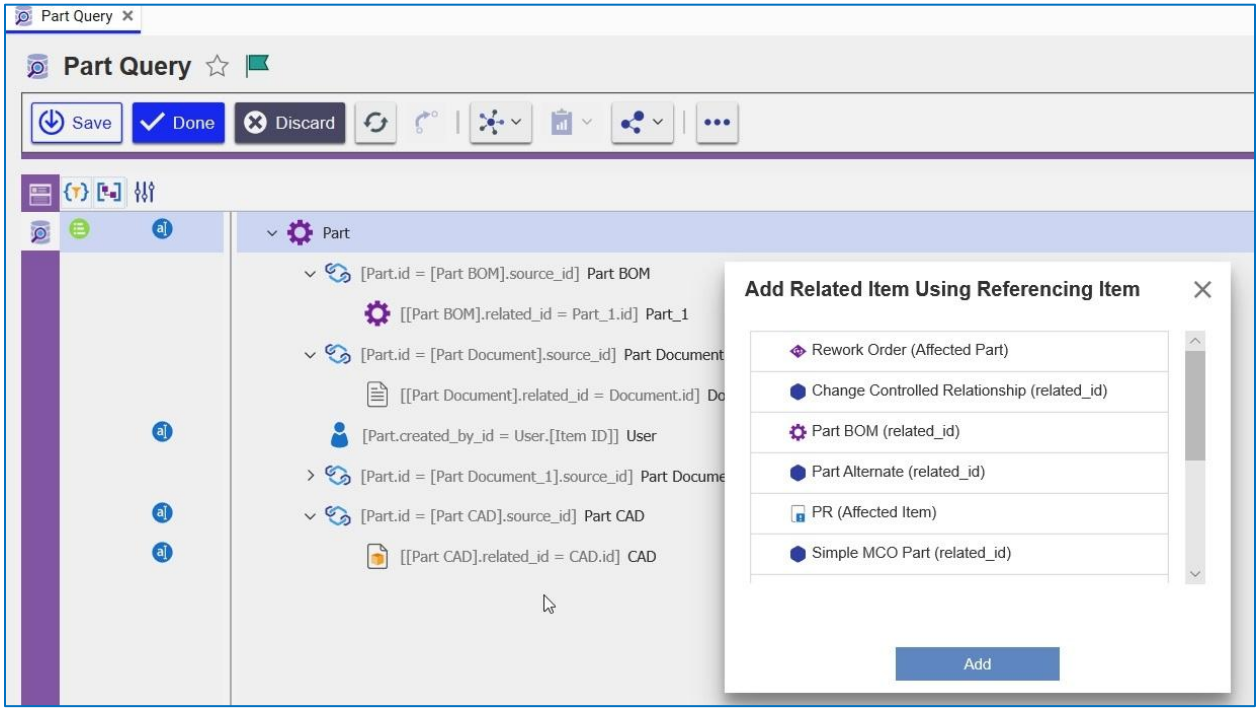

Figure 16.

3. Right-click the **Simple MCO Part** element and click **Add Related Item** --> **Using Item Property** to select **source_id**.

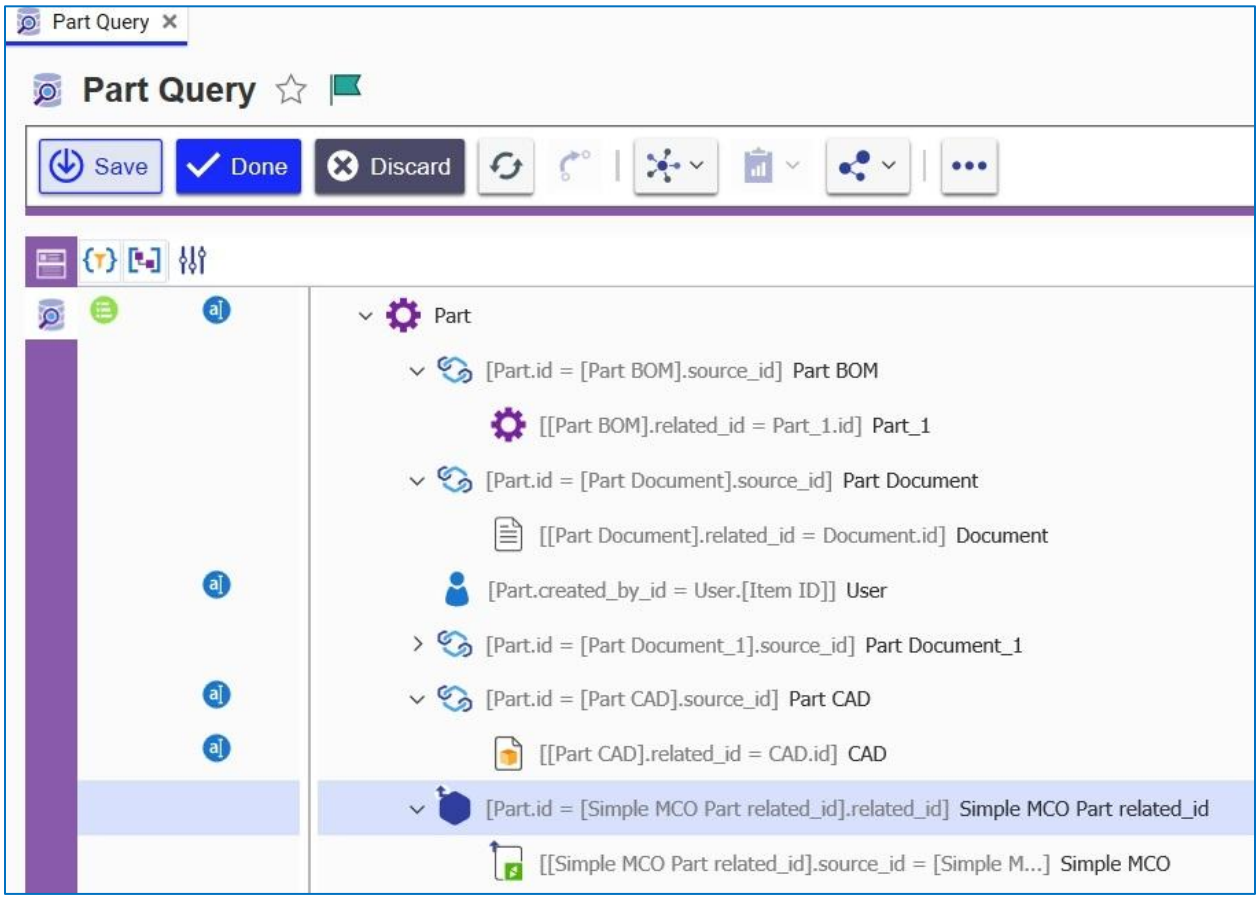

Figure 17.

4. Set Selected Properties on the Simple MCO element to:
   - `id`
   - `MCO Number`
   - `Status`
   - `Date Originated`
5. Save the Query Definition.

2.2.3 Adding a Recursive Call Structure

Recursive structures can reuse sections of the already defined queries. The following example shows how to set up a recursive call for the **Part** --> **Part BOM** --> **Part** relationship structure.

1. Right-click on the top-level **Part** element and add the **Part BOM** --> **Part** structure.

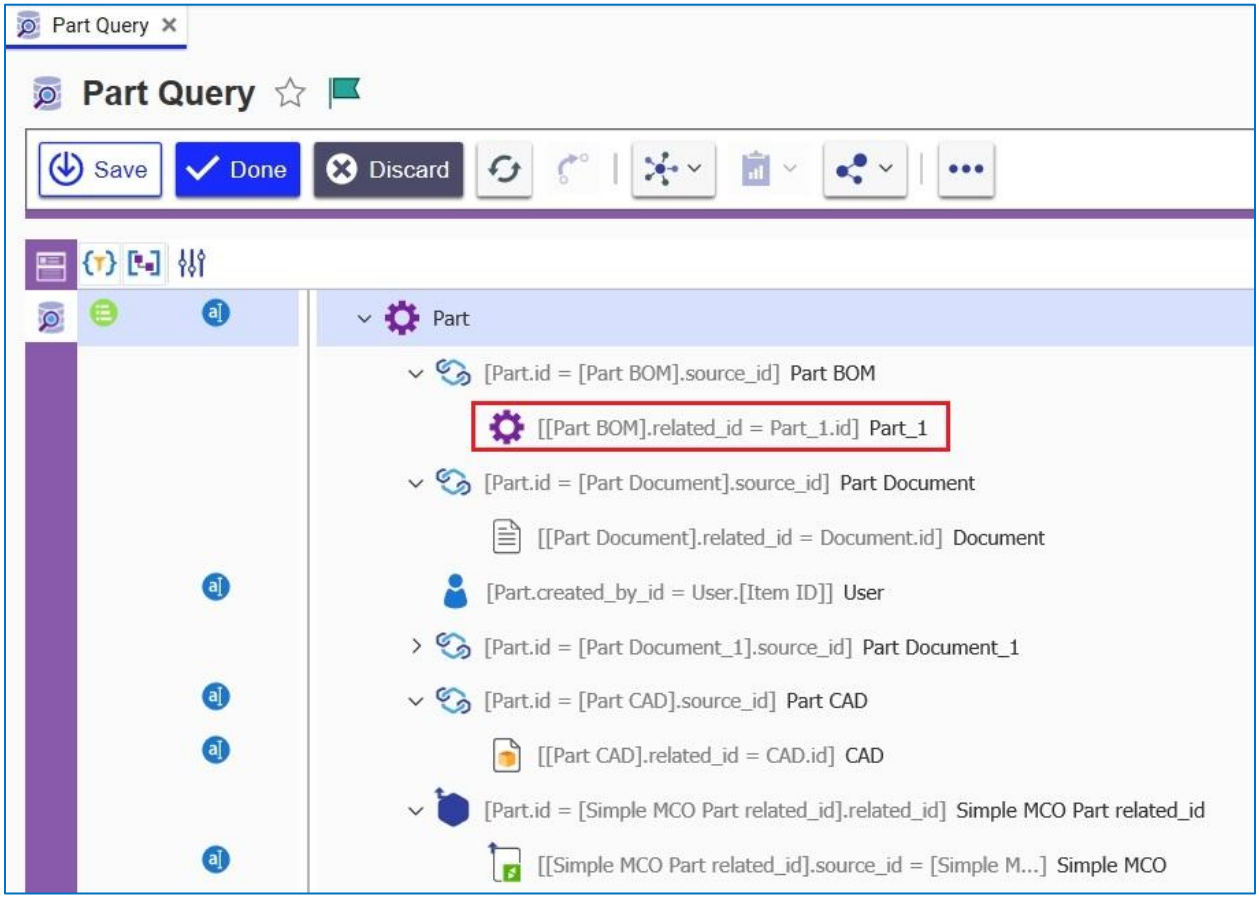

Figure 18.

2. For the BOM element, select the following properties:
   - `Sequence`
   - `Quantity`
   - `related_id`

3. Right-click on the child Part element and click **Reuse Query Element Definition**.

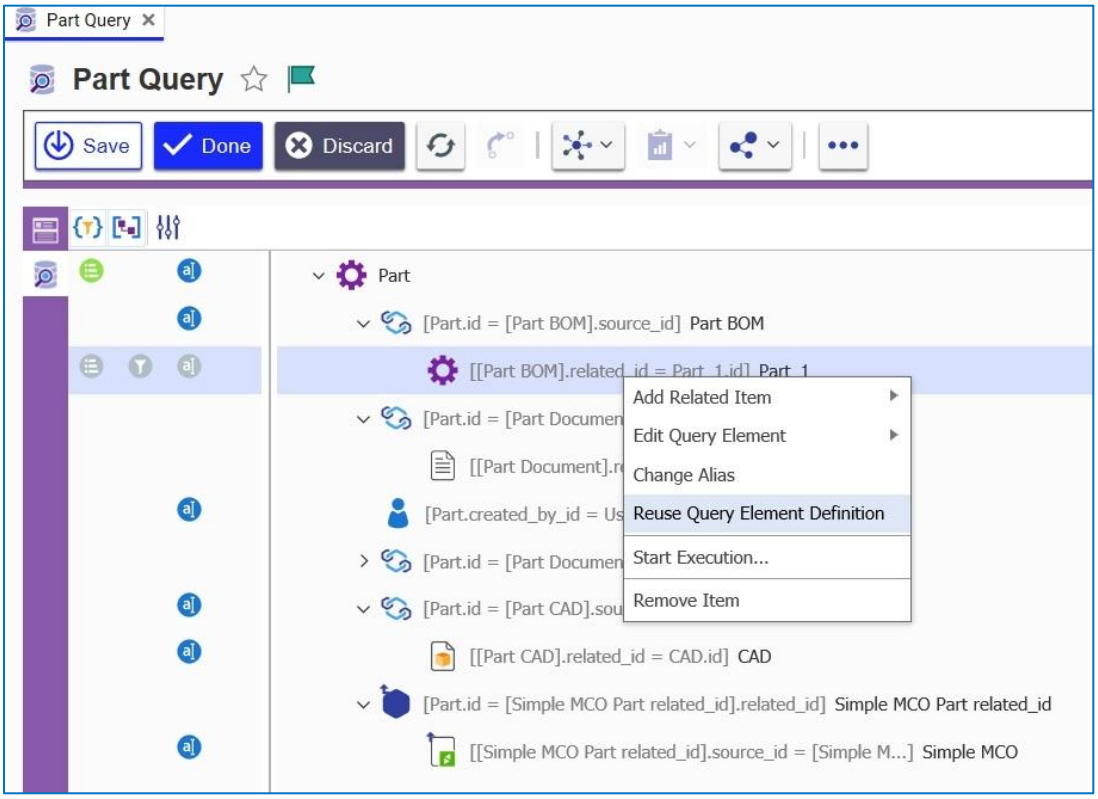

Figure 19.

4. Select the top-level **Part** element and click **Add**.

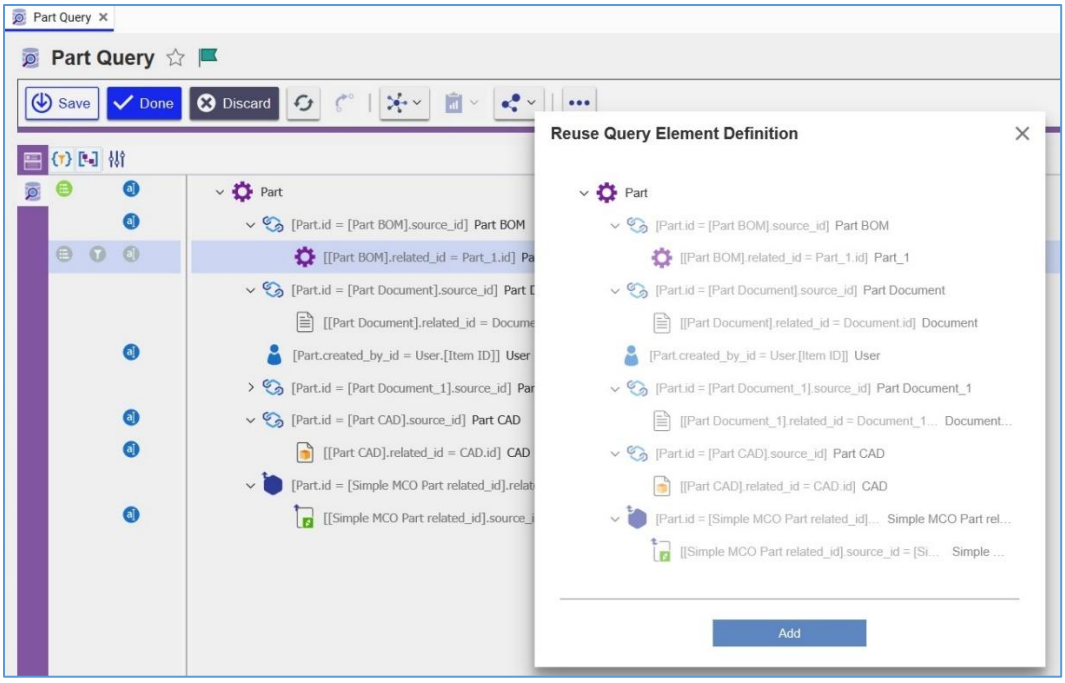

Figure 20.

5. Save the Query Definition item.

The child Part item now includes a search for the same elements as the top-level Part, including CADs, Documents, other Parts, etc. It also includes the same Properties and Order By logic as specified above.

2.3 Adding Search Conditions

You can specify Search conditions for every level of the Query Definition Structure. These search conditions can refine the set of data the Query returns. The query logic can include the components shown in Table 2.

Table 1: Query elements

Comparison Operators	Boolean Operators	Aggregate Functions
=	AND	Count()
!=	OR	
>	NOT	
>=		
<		
<=		
like		
is null		

Note: Query Builder does not support comparison of string and integer values. For example, you can use logic like `Count(…) = 3`, but you cannot use `Count(…) = "3"`.

Use the following process to add Conditions to the **Query Definition** item:

1. Right-click the **CAD** element and click **Edit Query Element** --> **Condition**.

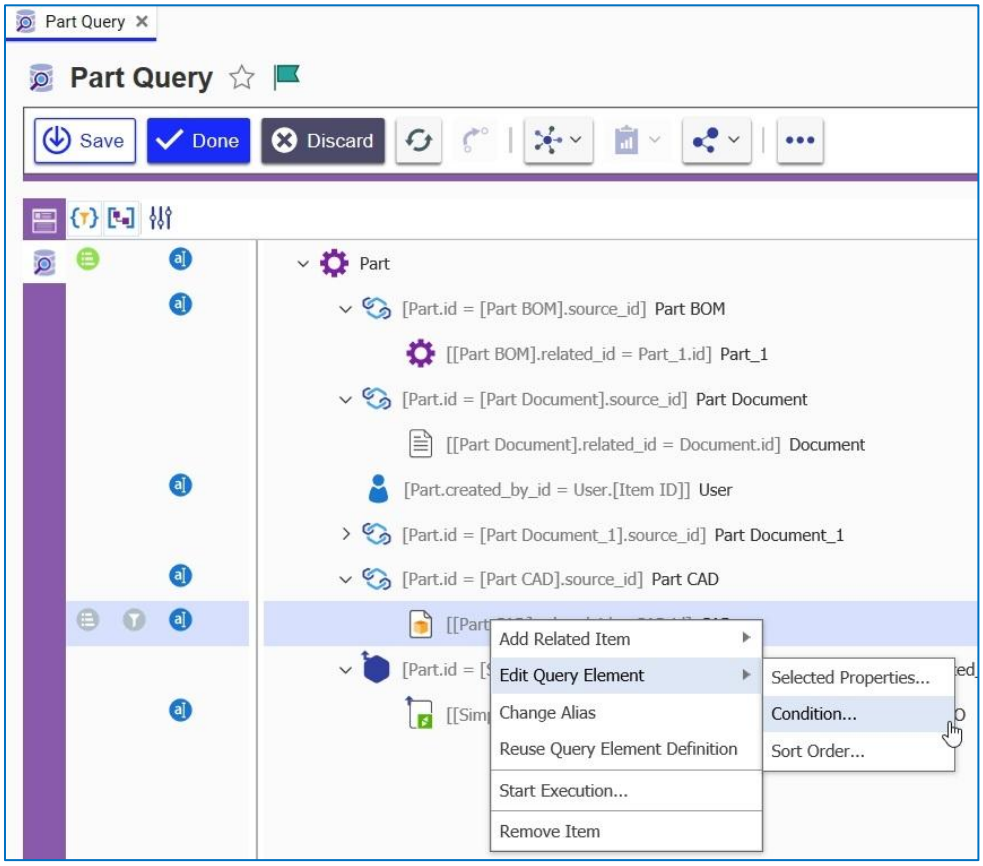

Figure 21.

2. Enter a search condition, such as `[Name] = 'CAD_1'Na`.

Figure 22.

Note: Conditions written in Query Builder must be separated by white space. For example, you can use the condition `Cost >= 0` but not `Cost>=0`.

3. Right-click the **Document** element and click **Edit Query Element** --> **Condition**.
4. Enter a search condition such as `[Document Number] like 'A27%'`.

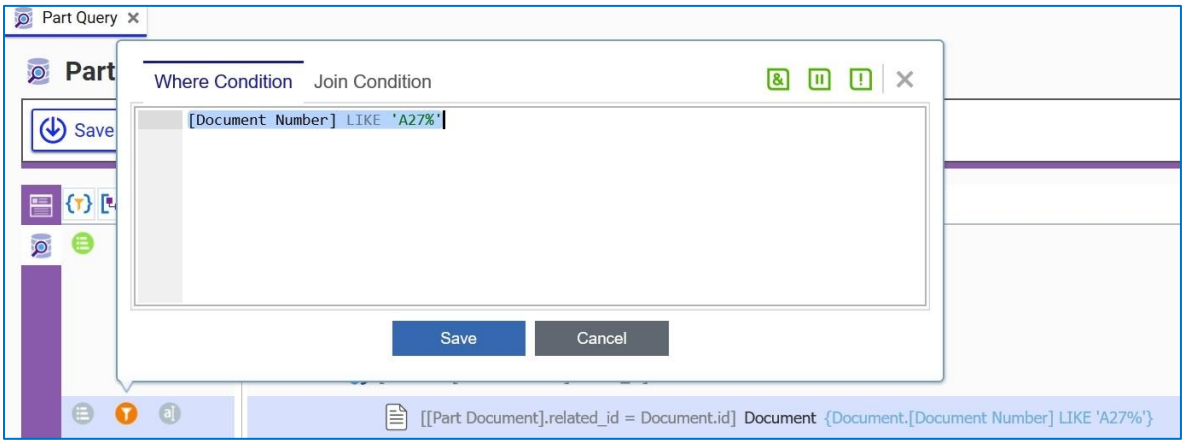

Figure 23.

Note: Brackets are placed around multi-word properties in the condition window of Query Builder. You may notice that these brackets automatically disappear for single-word properties.

5. Right-click the top **Part** element and click **Edit Query Element** --> **Condition**.
6. Enter a search condition that enforces the CAD and Document Relationships to exist:

```
(Count([[Relationship]] Part CAD) >= 1) AND (Count([[Relationship]]
Part Document) >= 1)
```

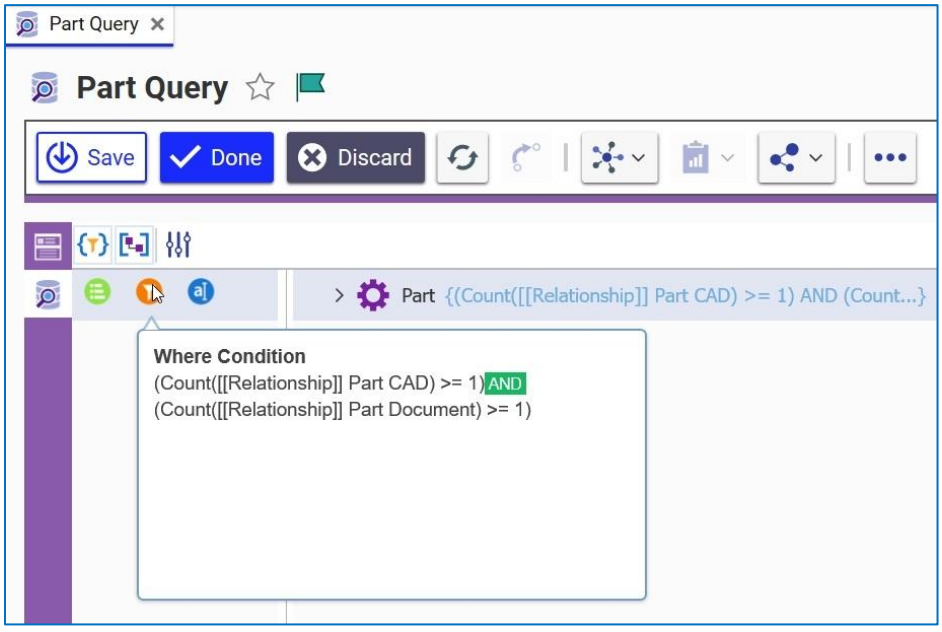

Figure 24.

7. Save the Query Definition.

2.4 Adding/Editing Query Parameters

The following example walks you through the process of defining query parameters. It uses Test Query as an example.

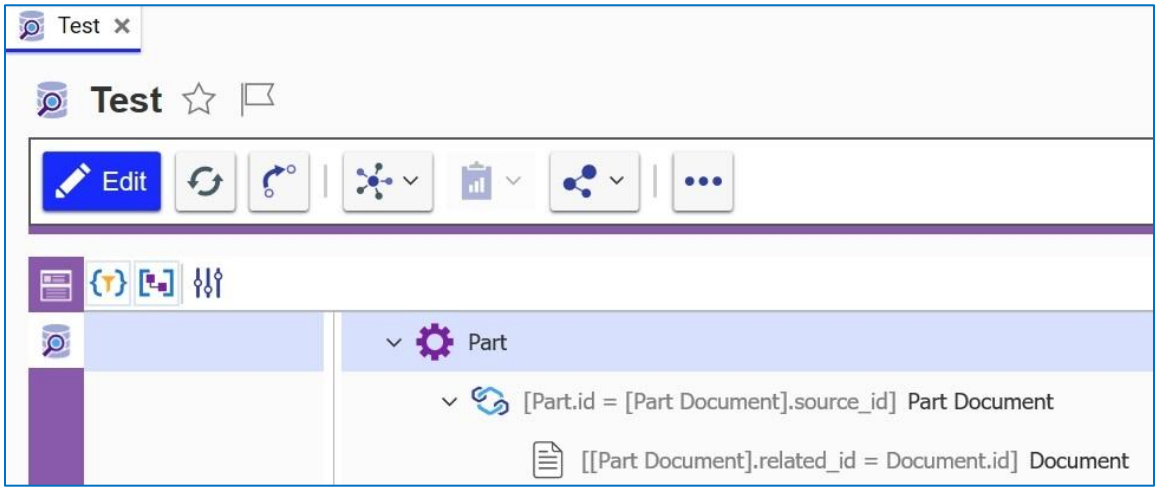

Figure 25.

1. Click the **Edit Parameters** button . The **Query Parameters** dialog box appears.

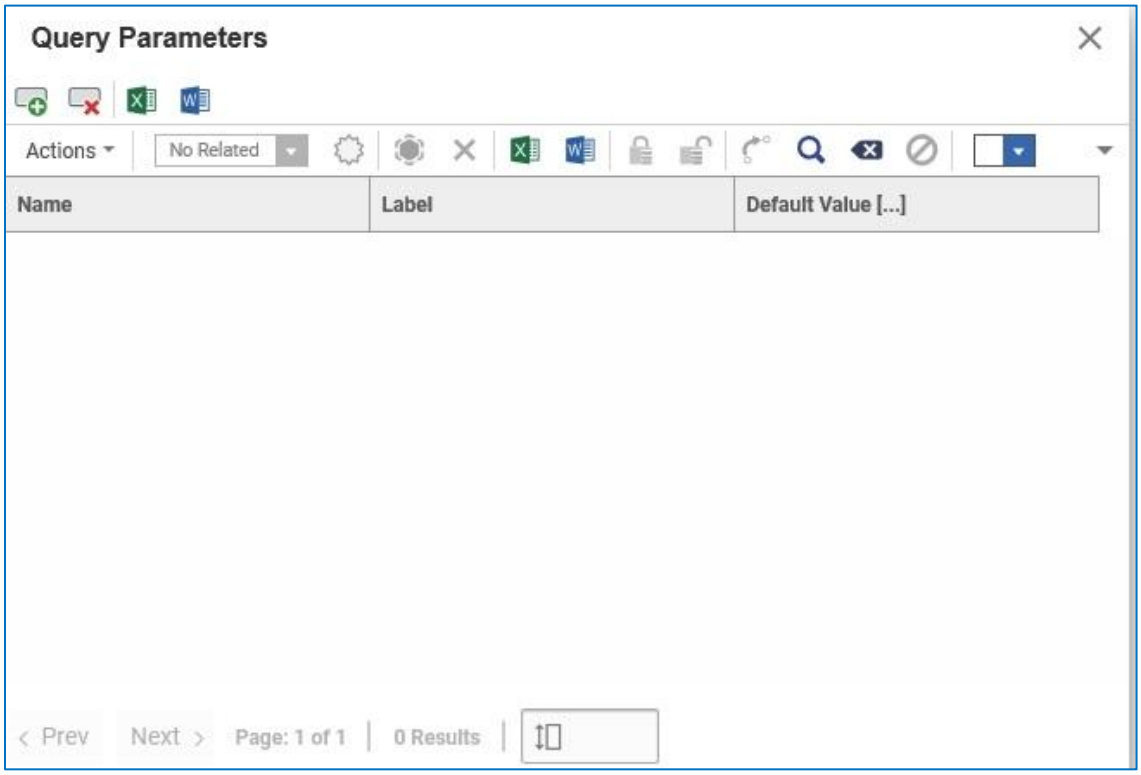

Figure 26.

2. Click the **New** button to add a parameter. Enter the following information:

Name: **doc_state**
Label: **Document State**
Default Value: ***

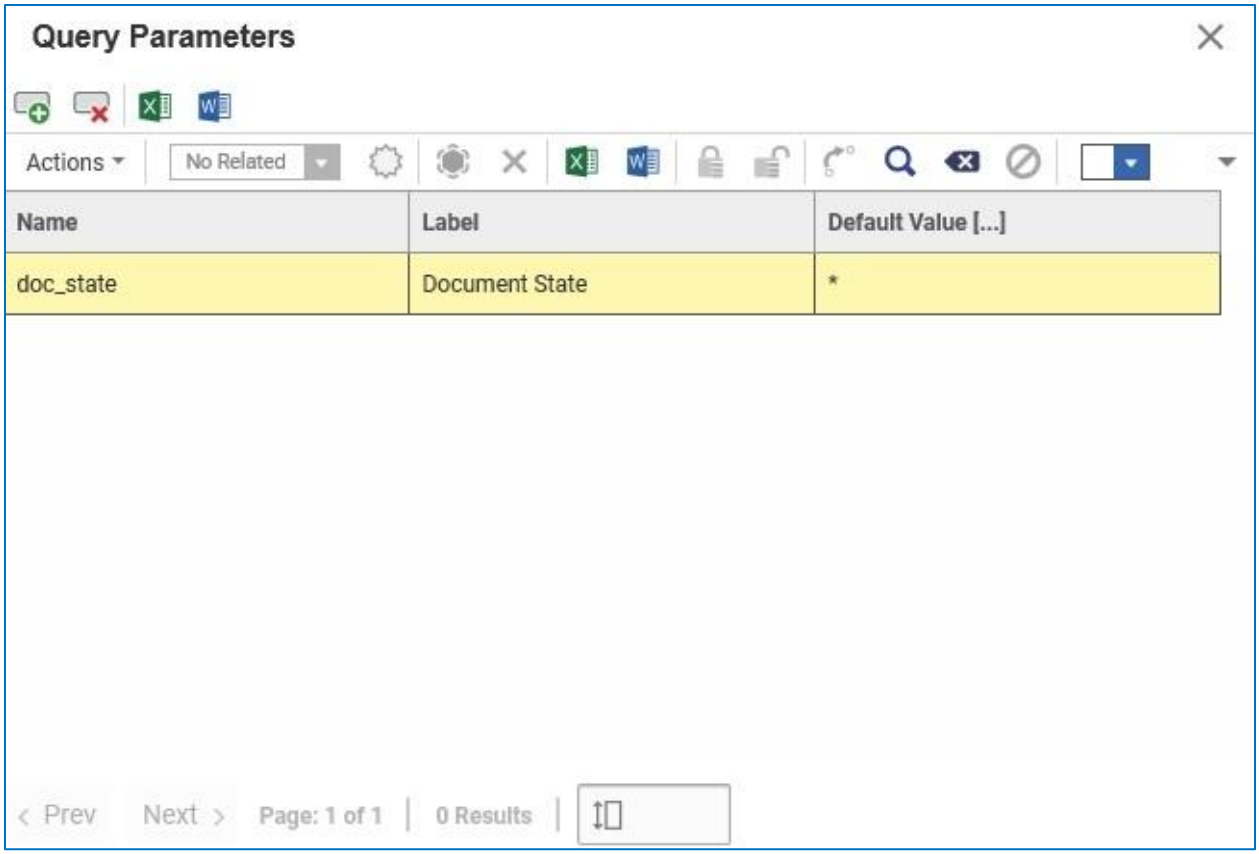

Figure 27.

3. Close the dialog box to save the parameter information.
4. Select the **Document** level of the relationship structure and click the associated **Filter** button to filter using the parameter you just created.
5. Set any other filters and sort orders that you need.
6. Save the Query definition.

The parameters that you add to the Query definition appear in the **Map Parameters** dialog box in the Tree Grid View.

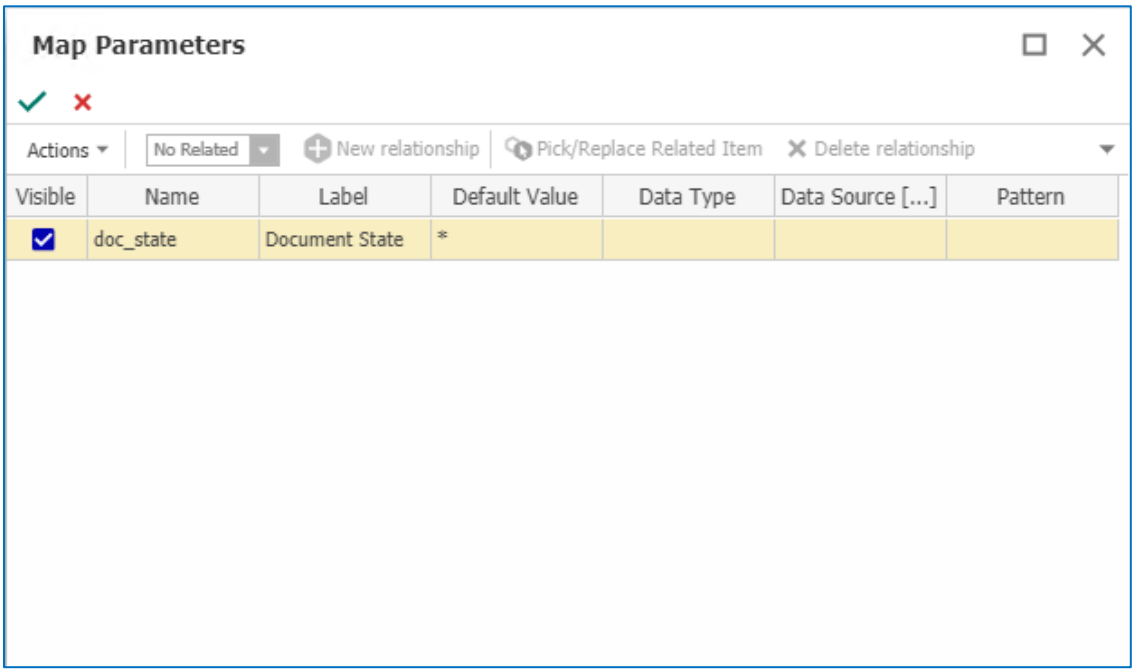

Figure 28.

Once mapped, the parameters become available in the Tree Grid View under the **Modify Parameters** button.

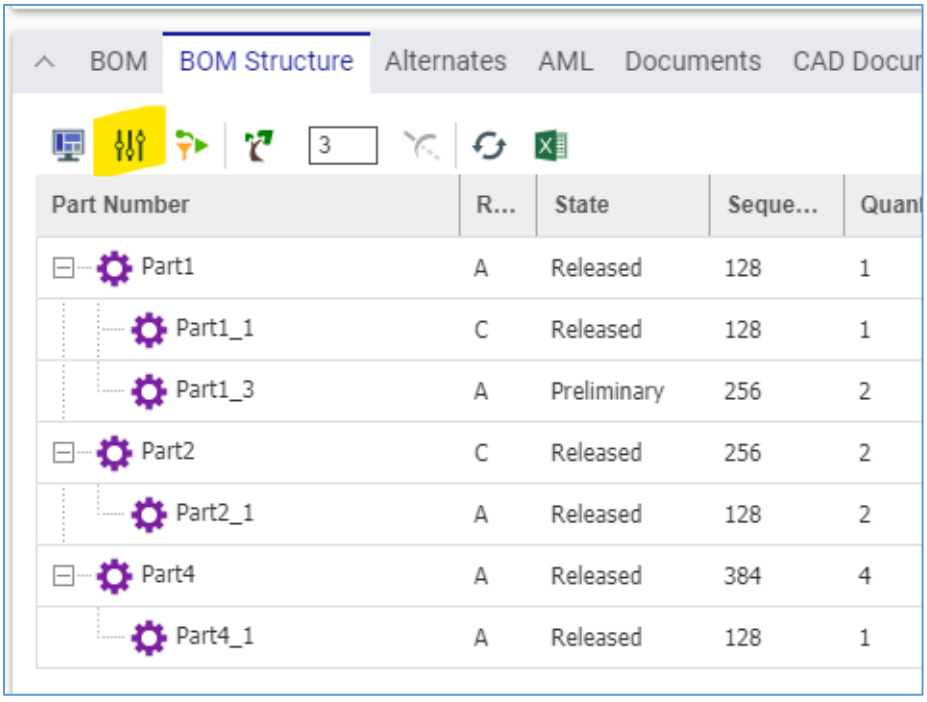

Figure 29.

This allows the entry or selection of the configured parameters from the Tree Grid View, setting query parameters in the underlying Query Definition from the Tree Grid User Interface at runtime.

Parameters can also be used for previewing query results when Executing a Query in the Query Builder. See Section 3.0 “Executing Queries” for an example of this.

2.4.1 Structure Resolution Parameter

For structures of versionable Items, like **Part** and **Part BOM** Structures, it is useful to examine query results in a Tree Grid View by either the latest versions, or by released versions only, or a combination where if no released components exist then the latest (un-released) version is displayed in the overall structure. The default is to display the structure as it is stored in the database without any optional resolution.

These Resolution Modes are provided to **Part BOM** Query Definitions (and Tree Grid Views) by a special Parameter which is pre-configured OOTB. Structure Resolution can be applied to any Item Relationship structure of versionable Items that follows the same rules as Part BOM structures: the **Released** Life Cycle State (**is_released** Boolean), incremental Revisions with interim versioned generations, etc.

2.4.1.1 Default Resolution Mode

The **Default** Resolution Mode uses the versions of related child Parts as stored on the BOM Relationships. This is the standard resolution in Aras Innovator.

Part Number	Revision	State
Child A	B	Preliminary
Child A.1	A	Preliminary
Child A.2	C	Released
Child A.3	C	Preliminary
Child B	B	Released
Child B.1	A	Released
Child B.2	B	Released
Child C	A	Preliminary
Child C.1	A	Preliminary
Child C.2	B	Released

Figure 30. The **Default** resolution mode

2.4.1.2 Latest Resolution Mode

When the **Latest** resolution mode is selected, the structure is resolved starting from the top using the latest versions of child items. When a child item is resolved to its latest version, then its child items are resolved using this resolution mode. This is done recursively.

The **Latest** resolution mode does not consider the released state of the child Parts.

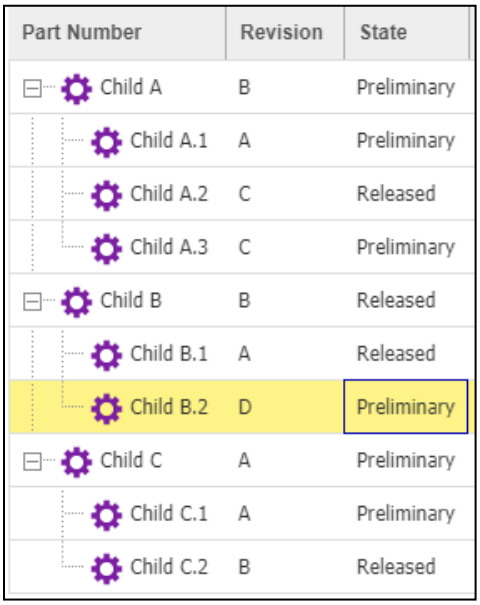

Figure 31. **Latest** resolution mode

The latest revision **D** of **Child B.2** (Preliminary) is shown where this Part version is in the **Preliminary** state, although the default structure under revision **B** of **Child A** (this structure) refers to revision **B** of **Child B.2** (Released) as stored in the database. In other words, the Latest Resolution Mode dynamically provides a view to in-work changes to released parts that are not yet committed to the released structure.

2.4.1.3 Latest Released Resolution Mode

When the **Latest Released** resolution mode is selected, the structure is resolved starting from the top using the latest versions of child items that are in Released state. When the child item is resolved to its latest Released version, then its child items are resolved using this resolution mode. This is done recursively.

If the related child item has no versions that are in Released state, then a blank row is displayed for that item. There is no further processing on that item.

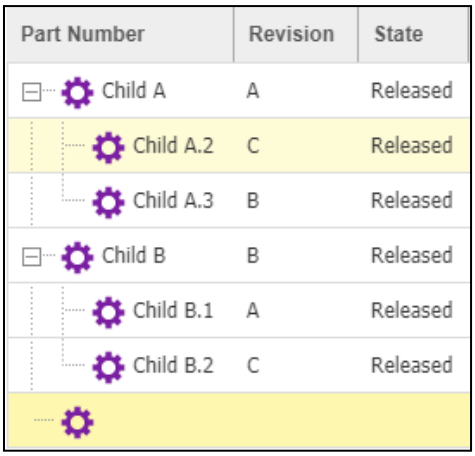

Figure 32. **Latest Released** resolution mode

The latest Released versions of child parts are resolved. A blank row is displayed for **Child C** because it has no version that is in the Released state.

2.4.1.4 Latest Released Or Latest Resolution Mode

When the **Latest Released or Latest** resolution mode is selected, the structure is resolved starting from the top using the latest versions of child items that are in Released state. If there are no versions that are in Released state, then the latest version is used. When the child item is resolved to a version, then its child items are resolved using this resolution mode. This is done recursively.

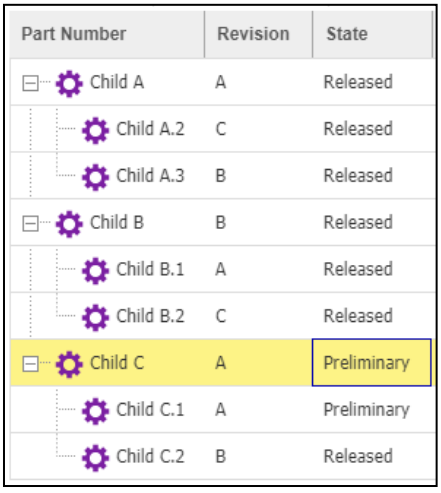

Figure 33. **Latest Released or Latest** resolution mode

The latest Released versions of child parts are resolved. **Child C** does not have any version that is in the Released state. Therefore, its latest Preliminary version is resolved.

2.4.2 Structure Resolution Configuration for Part BOM Structures

In Aras Innovator, the BOM Structure grid (a TGV under the **BOM Structure** tab in the **Part** form) is pre-configured with a Structure Resolution Parameter, thus it supports the **Latest**, **Released**, and **Released or Latest** modes out of the box.

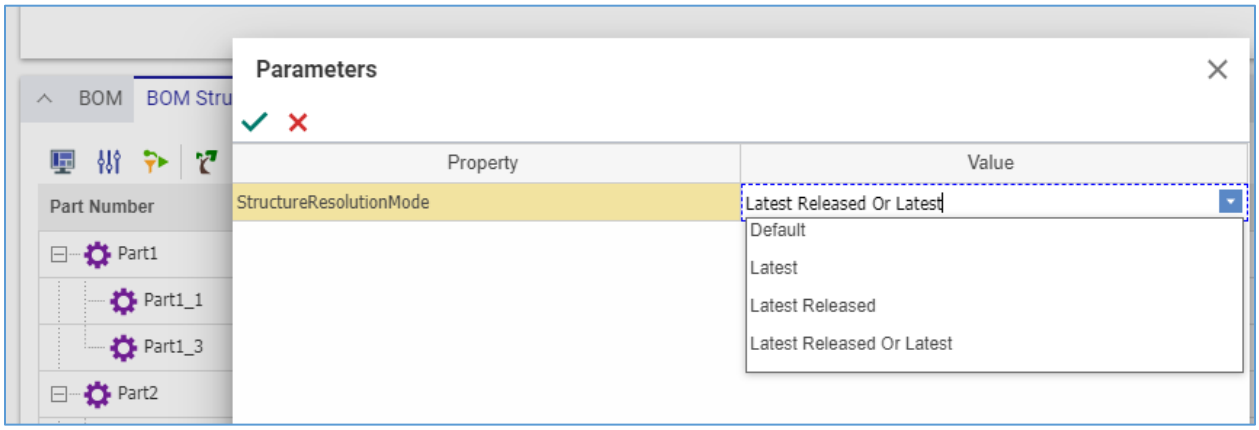

Figure 34.

In the event you need to enable Structure Resolution for your Query Definition, you will need to add a Parameter that has a default value of **Aras.Resolution.EntryPoint;Default**, and then also follow the additional detailed steps for configuring Structure Resolution Parameters described in Section 7.

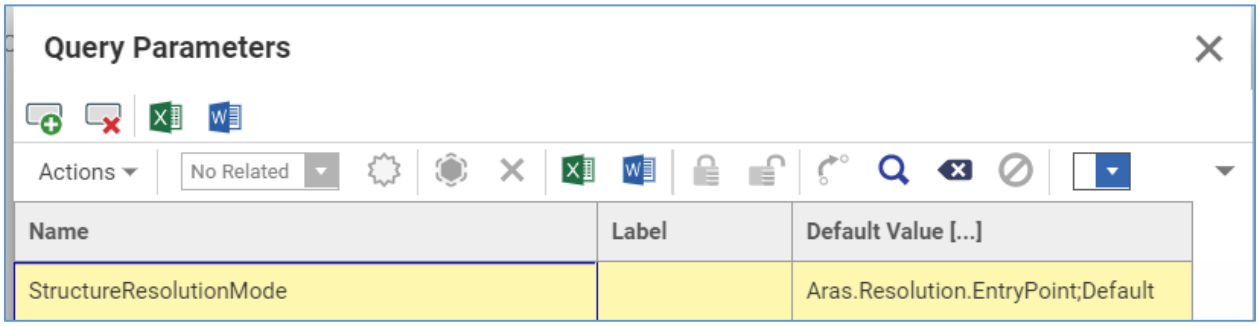

Figure 35.

For more details and extensive examples of how Resolution Modes are applied in BOM Structures, Effectivity, and 3D Visualization please refer to the following documents:

- *Aras 3D Visualization - Administrator Guide*
- *Aras Innovator Product Engineering - User's Guide*

3 Executing Queries

The query can be executed at any time by right-clicking on the Query structure or by selecting the option from the **Actions** menu.

Note: The results of the query are limited by the User's standard "Get" permissions. In the case of a table row that contains relationship and related_item information you must have "Get" permissions for the related item, otherwise the cells in the row that correspond to the related_item will appear as blank/empty.

Query Builder/Tree Grid View can be configured to use the "keyed_name" associated with the related_item. If there is no "Get" access to the related_item, the keyed_name value will appear as "Restricted."

Executing the query from the top-level element or from the **Actions** menu executes the entire query structure. However, executing the query from one of the child elements returns the results for that substructure only.

Running the **Start Execution** command results in a window that looks similar to the following:

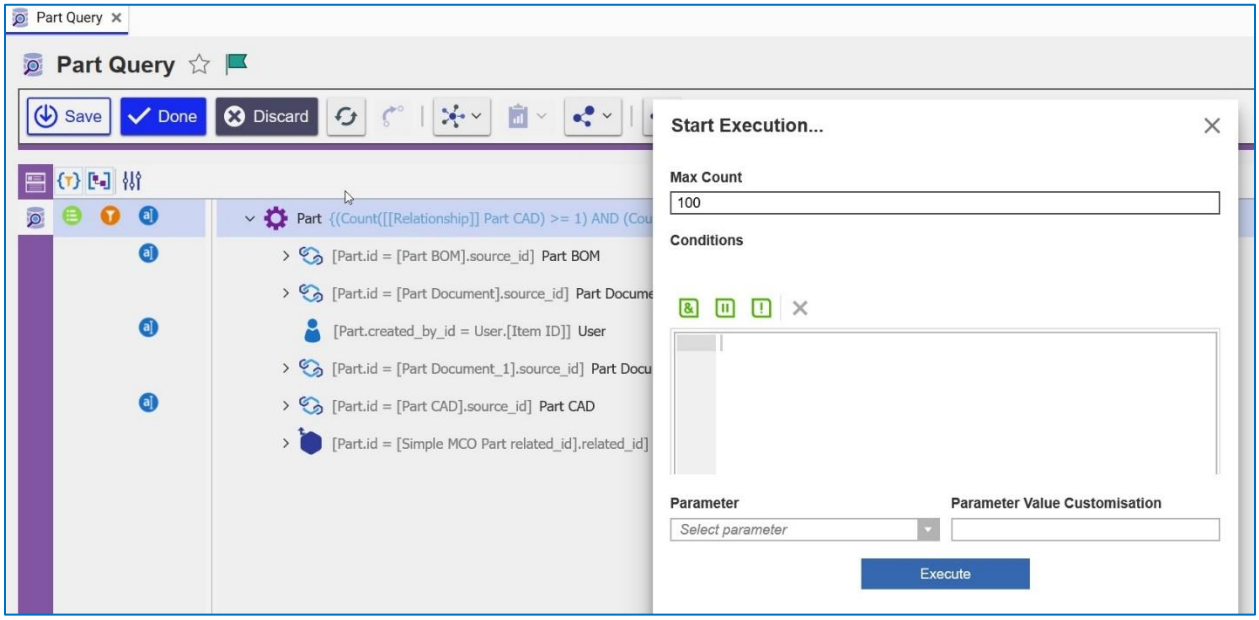

Figure 36.

The window offers an opportunity to add conditions on top of the ones defined by the search, enter parameter values, and set **Max Count**. If Conditions and/or Parameters are unspecified the query is executed without any filtering via conditions or parameters. The Max Count is intended to limit the number of top-level items returned by the search. These settings are recommended during development because the number of elements returned by the query could potentially be very large.

When ready, press the **Execute** command to run the search.

The search for the **Part Query** defined in Section 2 Developing Query Definitions would return a structure similar to the following:

```
<AML>
    <Item type="Part">
        <id keyed_name="part_7">82A0B82AAD1C43309075640D21117E90</id>
        <modified_by_id keyed_name="Innovator
Admin">30B991F927274FA3829655F50C99472E</modified_by_id>
        <name>part_7</name>
        <state>Preliminary</state>
        <item_number>part_7</item_number>
        <Relationships>
            <Item type="Part CAD">
                <sort_order>128</sort_order>
                <Relationships>
                    <Item type="CAD">
                        <current_state
keyed_name="Preliminary">A29C9720FF0D4950A140A90BEEA903DA</current_state>
                        <id keyed_name="cad_1">C94B5097E5894EDA8C39C310F735832B</id>
                        <name is_null="1" />
                        <item_number>cad_1</item_number>
                    </Item>
                </Relationships>
            </Item>
            <Item type="Part Documents">
                <sort_order>1</sort_order>
                <Relationships>
                    <Item type="Document">
                        <id keyed_name="Doc_01">FFB47BECC6E64E8BBB8254F0A61E9EEC</id>
                        <name>User Guide</name>
                        <state>Preliminary</state>
                        <item_number>Doc_01</item_number>
                    </Item>
                </Relationships>
            </Item>
            <Item type="Part Created By">
                <id keyed_name="Super User">AD30A6D8D3B642F5A2AFED1A4B02BEFA</id>
                <login_name>root</login_name>
                <first_name>Super</first_name>
                <last_name>User</last_name>
            </Item>
            <Item type="Simple MCO Part (related_id)">
                <Relationships>
                    <Item type="Simple MCO">
                        <created_on>2019-03-21T15:59:26</created_on>
                        <id keyed_name="MCO-100001">A5DCE160DE28465A8ECDFF37CEA6B75C</id>
                        <state>New</state>
                        <item_number>MCO-100001</item_number>
                    </Item>
                </Relationships>
            </Item>
        </Relationships>
    </Item>

</AML>
```

Note: All child elements of the top-level Part item are returned as `Relationships`, including **created_by_id** and **Simple MCO**.

3.1 Saving Query Results to a Local File

Using the Query Execution Dialog, users can opt to save the results returned to a local file instead of having the results displayed in a pop-up dialog. Saving the results in this manner is for convenience when users would like to review query results using a separate editor.

3.1.1 Save Query Results

To save the results from executing a Query Definition to a local file, select the check box labeled **Result as File** prior to executing the query.

Figure 37. Save Query Results to a File

Once the results are returned to the client, the web browser will provide an interface for the user to select a folder and file name to store the results. The default file name uses the Query Definition name with an '.xml' extension. For example, `MyQueryDefinition.xml`.

4 Sample Query with xProperties

This section describes using Query Builder to create a query using a Part item containing an xProperty.

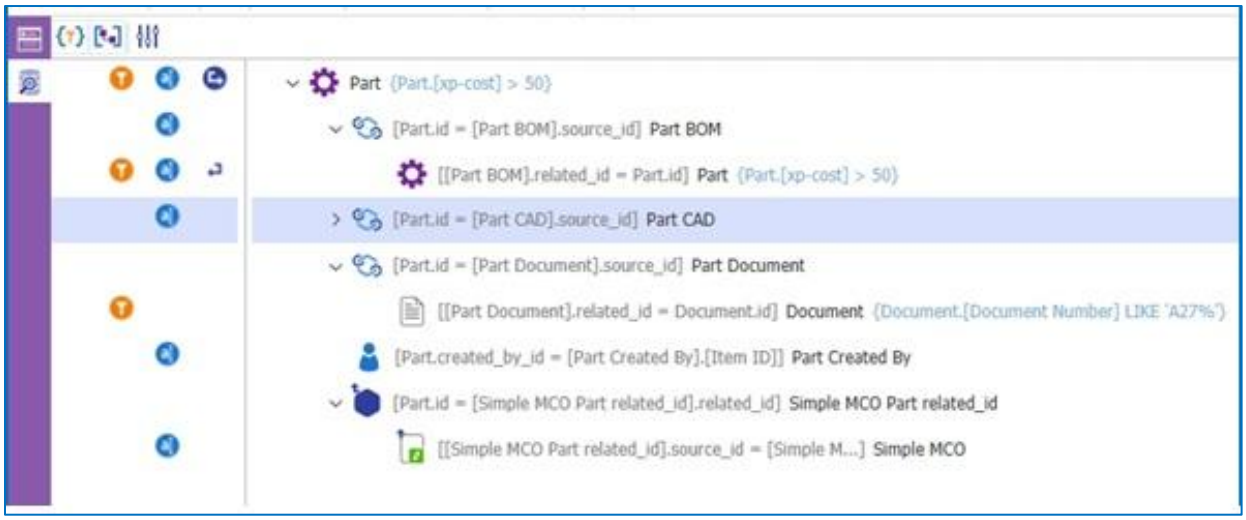

Figure 38.

The first level of the query contains a Part item that has the **xp-cost** property associated with it. Click the Part **Properties** button to view this xProperty.

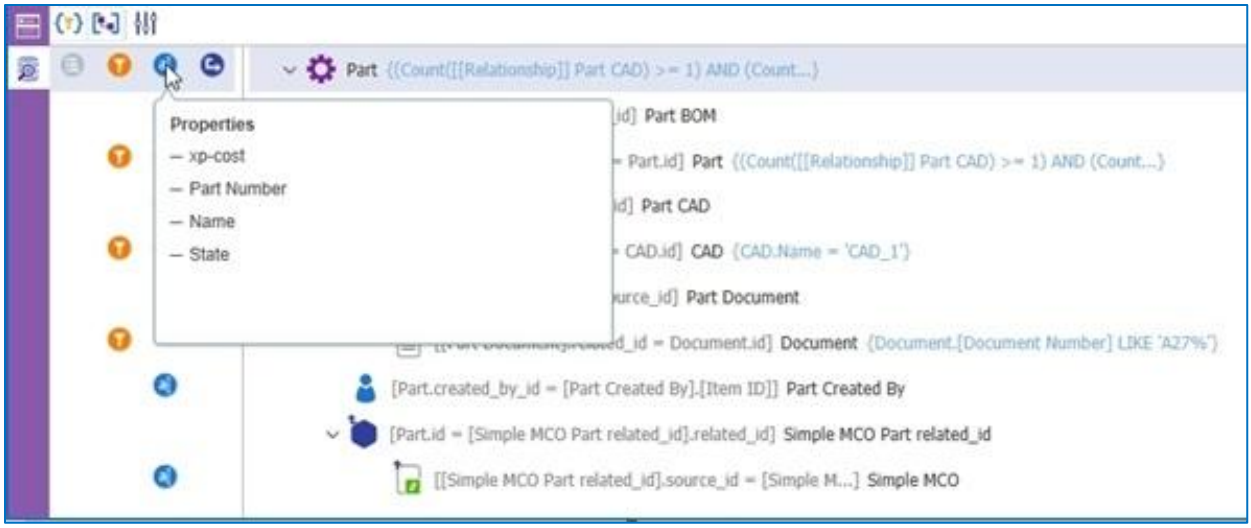

Figure 39.

4.1 Creating a Where Condition

You can create a Where condition using the **xp-cost** property by clicking the **Show Condition** button. The **Where Condition** box appears.

Figure 40.

1. Enter **[xp-cost] > 50** in the **Where Condition** dialog to return the number of items where the cost is greater than 50.

Figure 41.

2. Click **Save**.

3. Right-click **Part** and click **Start Execution**.

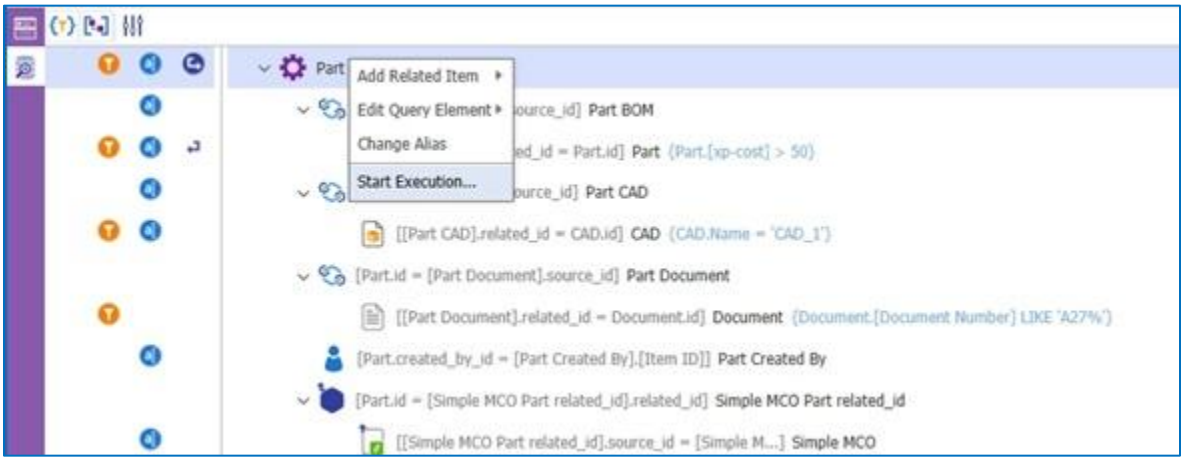

Figure 42.

The **Query Result** dialog appears.

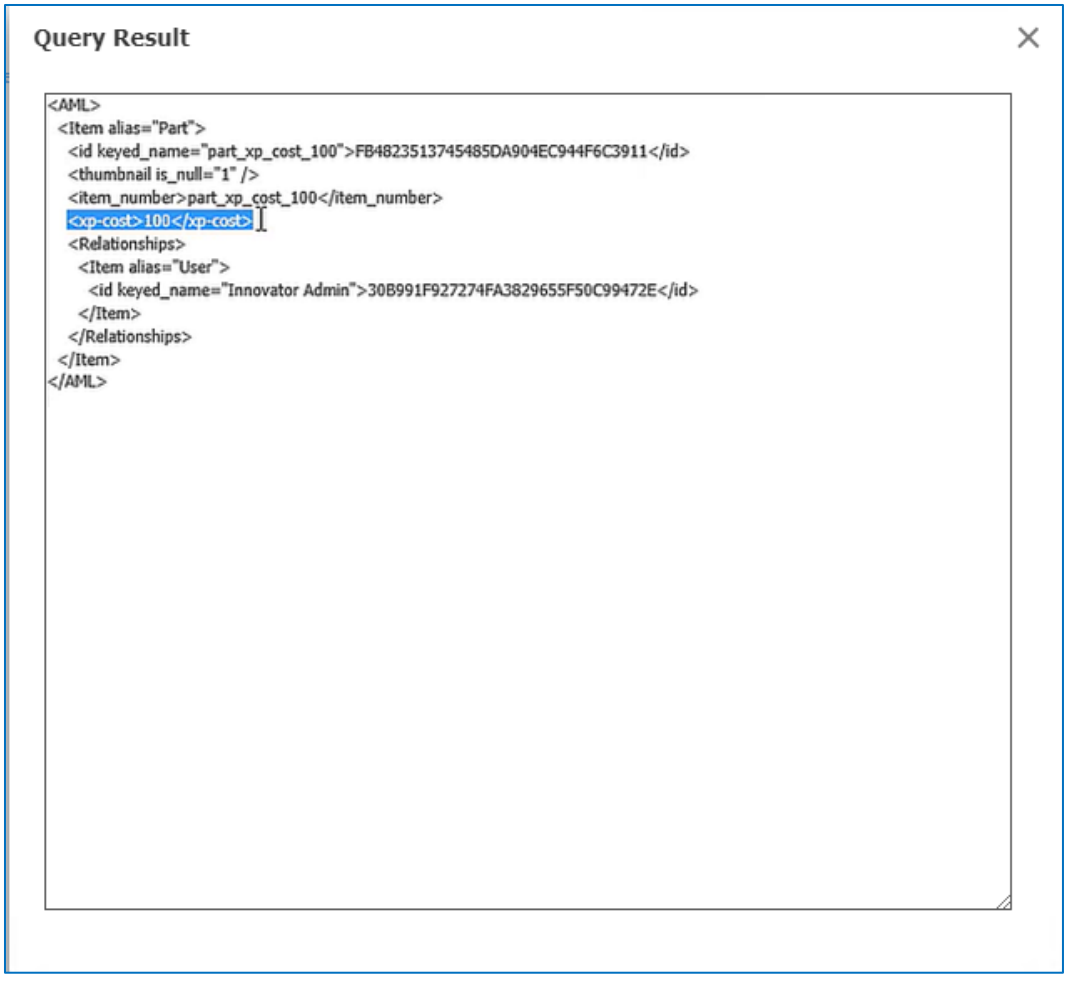

Figure 43.

As you can see, this query only contains one `part` item where the cost is greater than 50. You can modify the Where condition so **xp-cost > 100**. This returns two parts in the result.

Figure 44.

Both parts contain the **xp-cost** xProperty.

5 Using Max and Min Aggregate Functions in a Query

The following example demonstrates how to use the Max() aggregate function in a Where clause to identify the latest related part using the generation property. The Max() aggregate function returns the maximum value of a specified property. Similarly, the Min() aggregate function returns the minimum value of a specified property. You can use a statement like this in the conditional logic applied to either a Join or Where clause when defining a Query Definition.

1. Go to **Contents** --> **Administration** --> **Configuration** --> **Query Definitions**. The menu shown in Figure 45 appears.

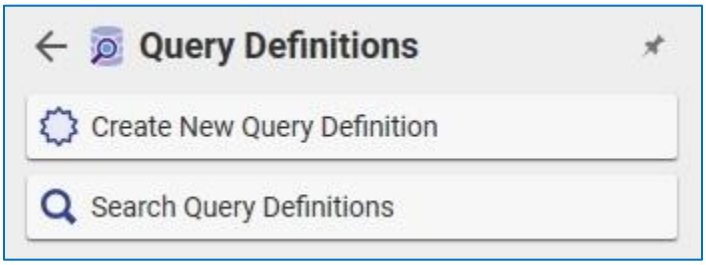

Figure 45.

2. Click **Create New Query Definition**. A blank **Query Definition** dialog appears.

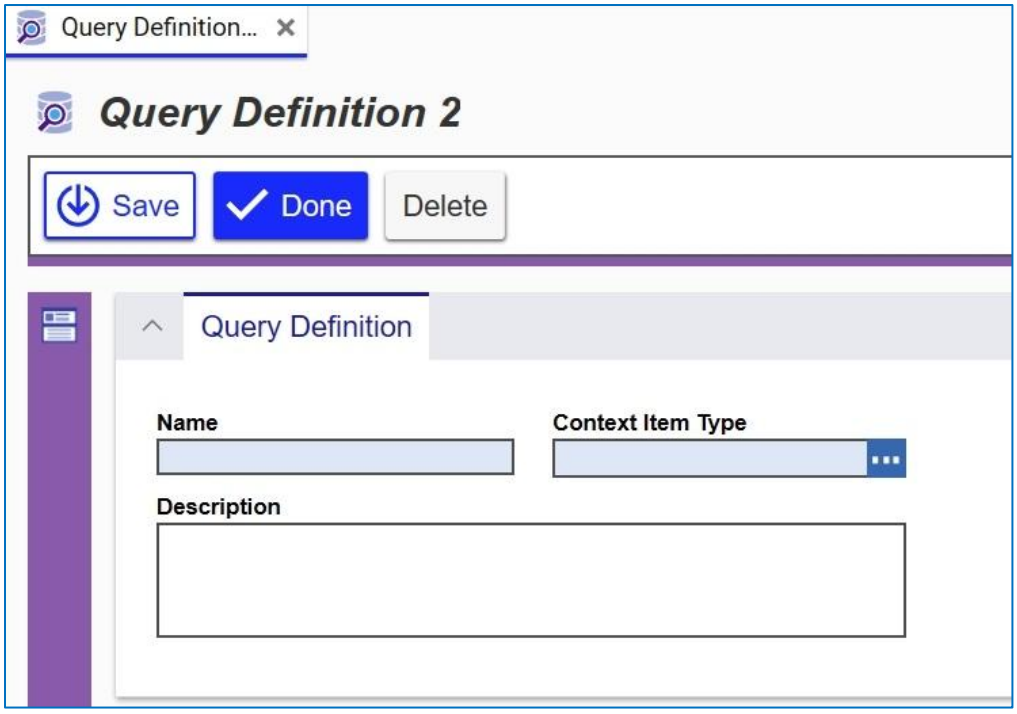

Figure 46.

3. Enter the **Query** title in the **Name** field.
4. Click the ellipses in the **Context Item Type** field and select **Part**.

5. Click 

. The **Show Editor** button appears in the ribbon.

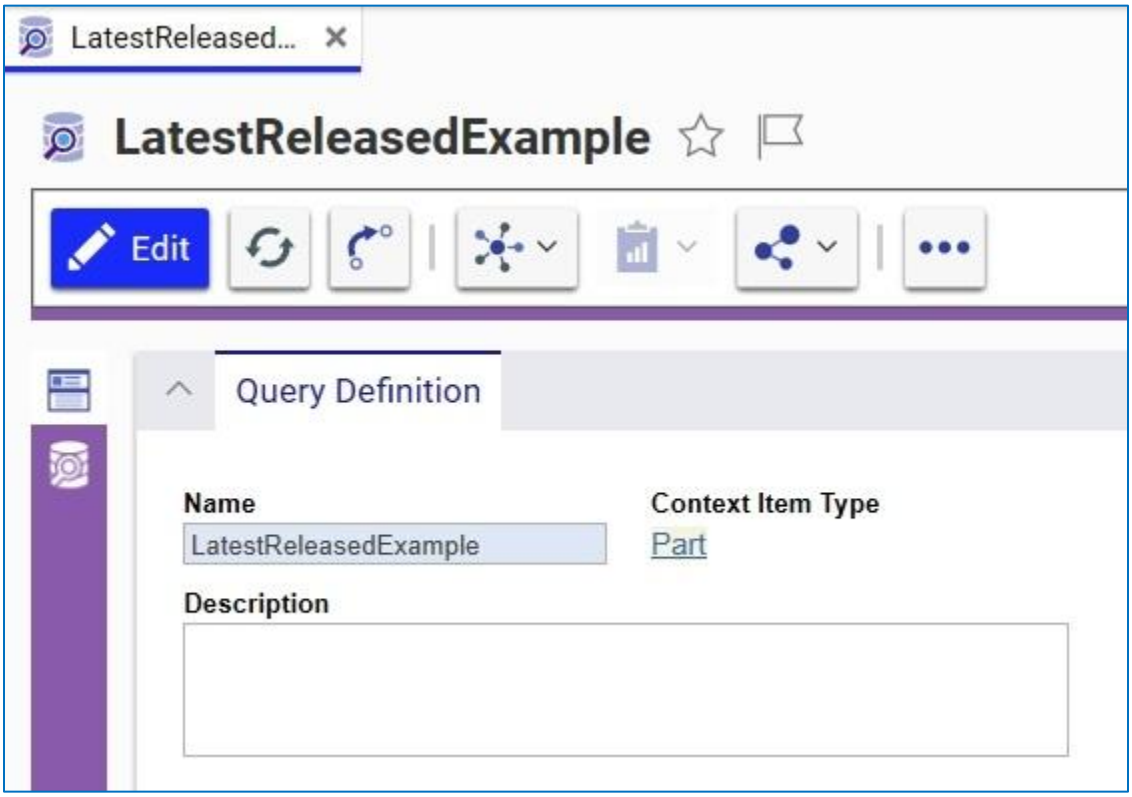

Figure 47.

6. Click the **Show Editor** button. The dialog appears displaying the context item.

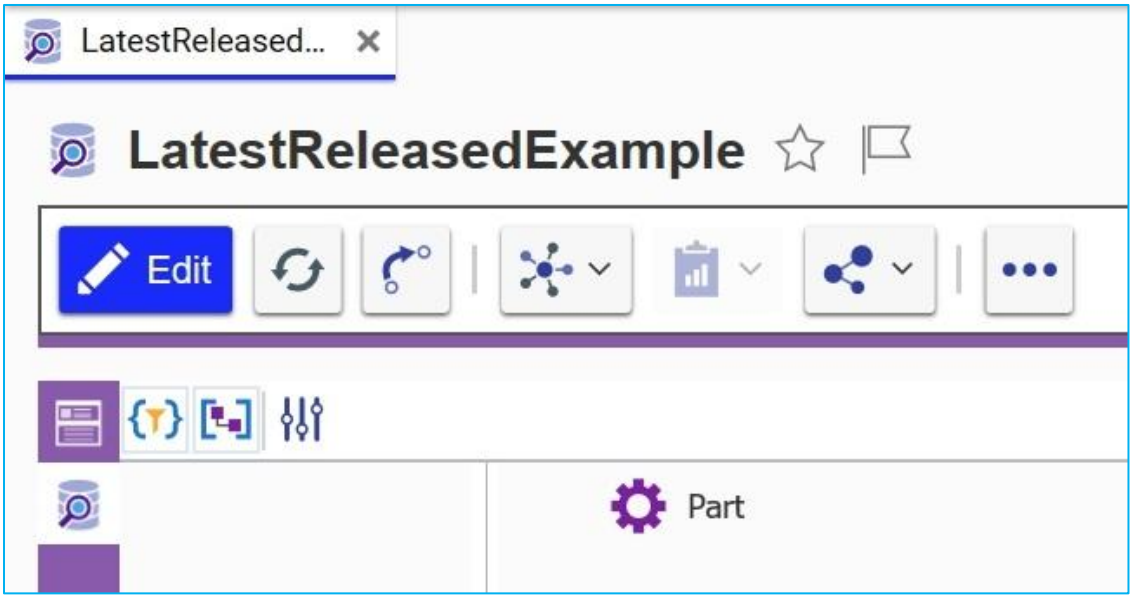

Figure 48.

7. Right-click on the context item and select **Add Related Item** --> **Using Item Property**.

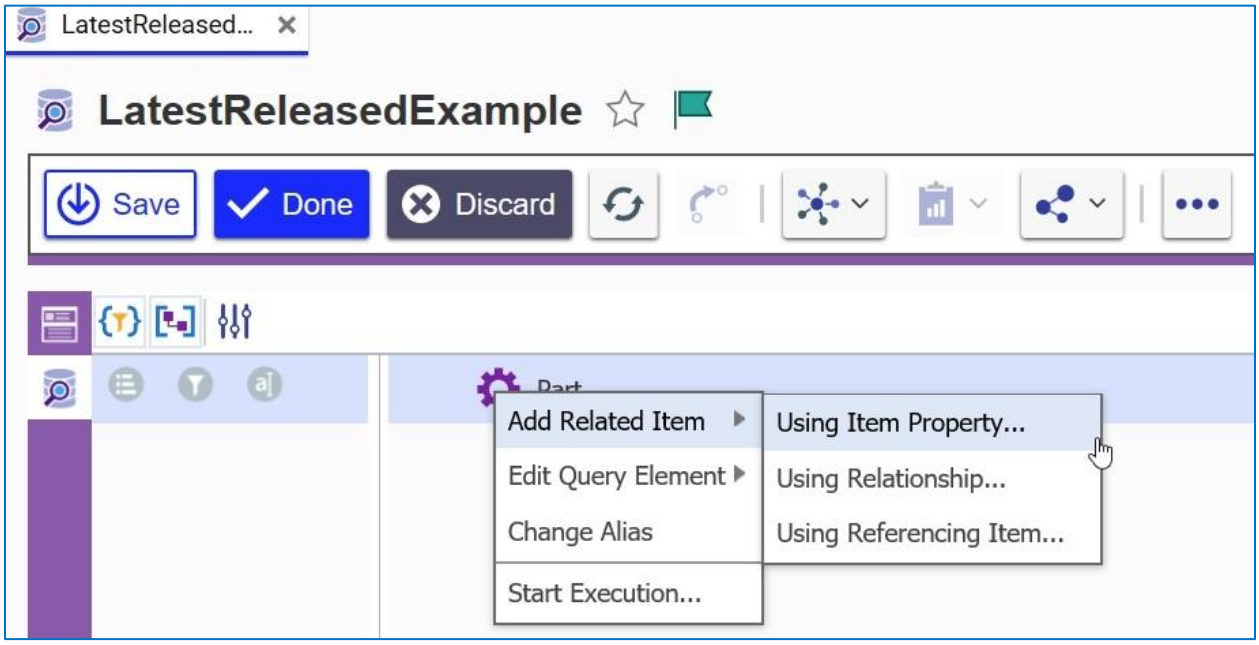

Figure 49.

The dialog box shown in figure 50 appears.

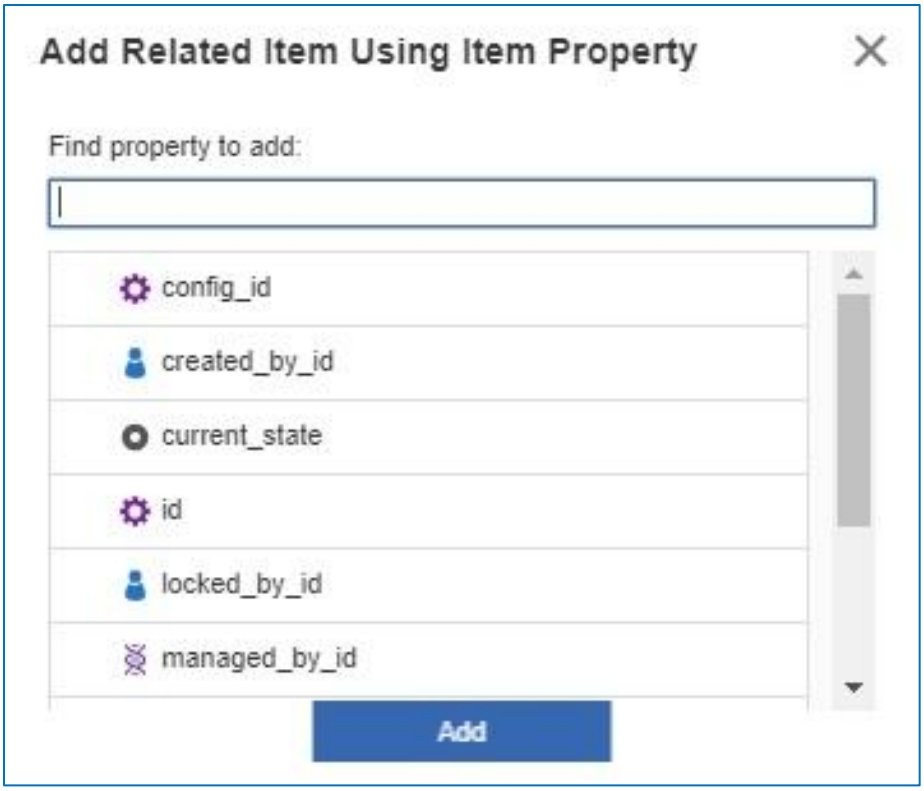

Figure 50.

8. Select **config_id** and click **Add**.

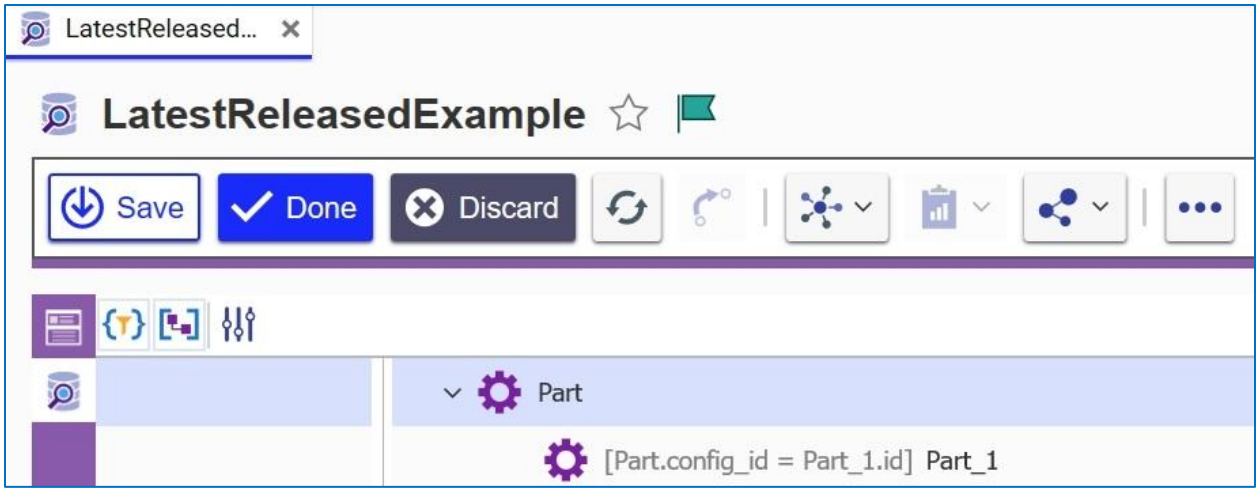

Figure 51.

9. Right-click the **Where** condition button and click **Change Alias** in the context menu:

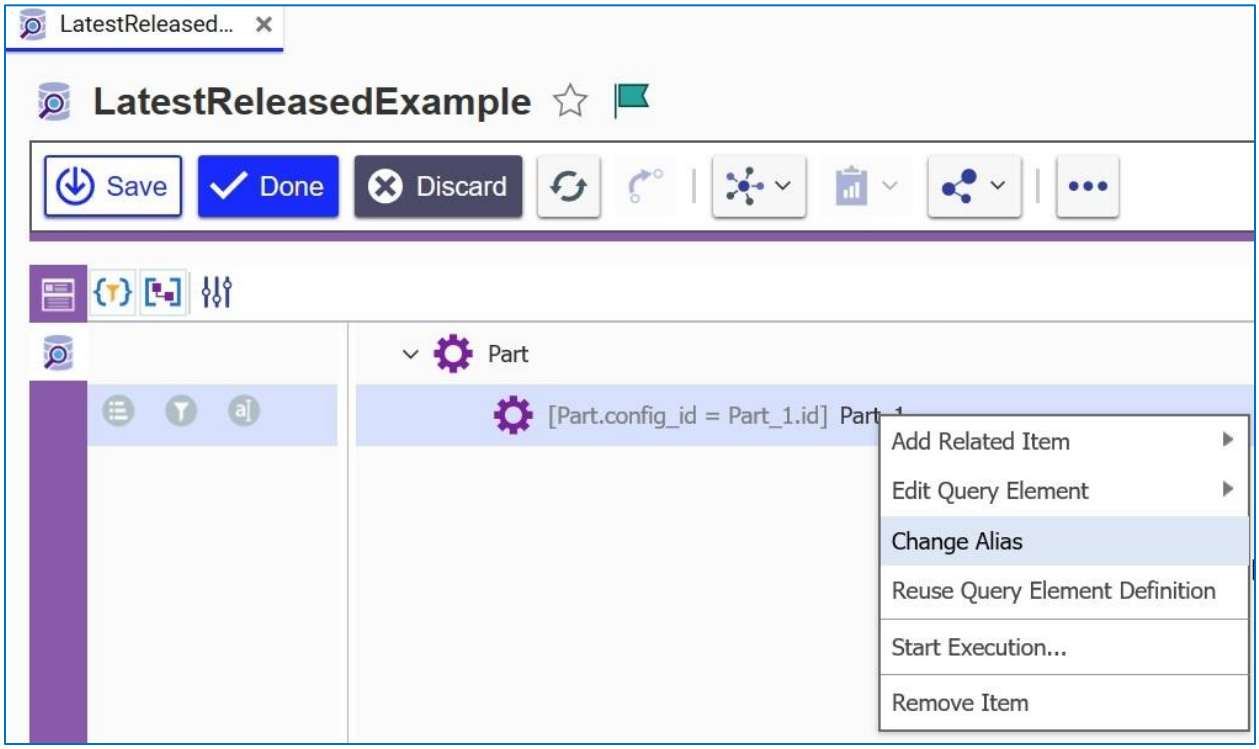

Figure 52.

The **Change Alias** dialog appears.

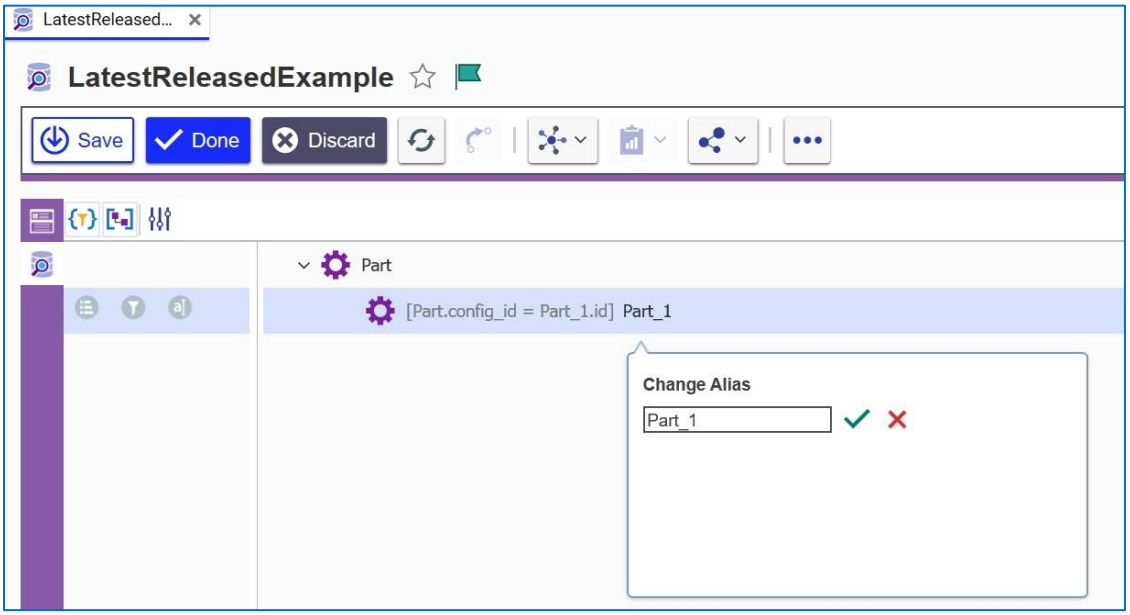

Figure 53.

10. Change the alias to **Part_LatestReleased_Filter** and click ✓.
11. Click the **Where** condition button and create a join condition for the child **Part** as shown in Figure 54.

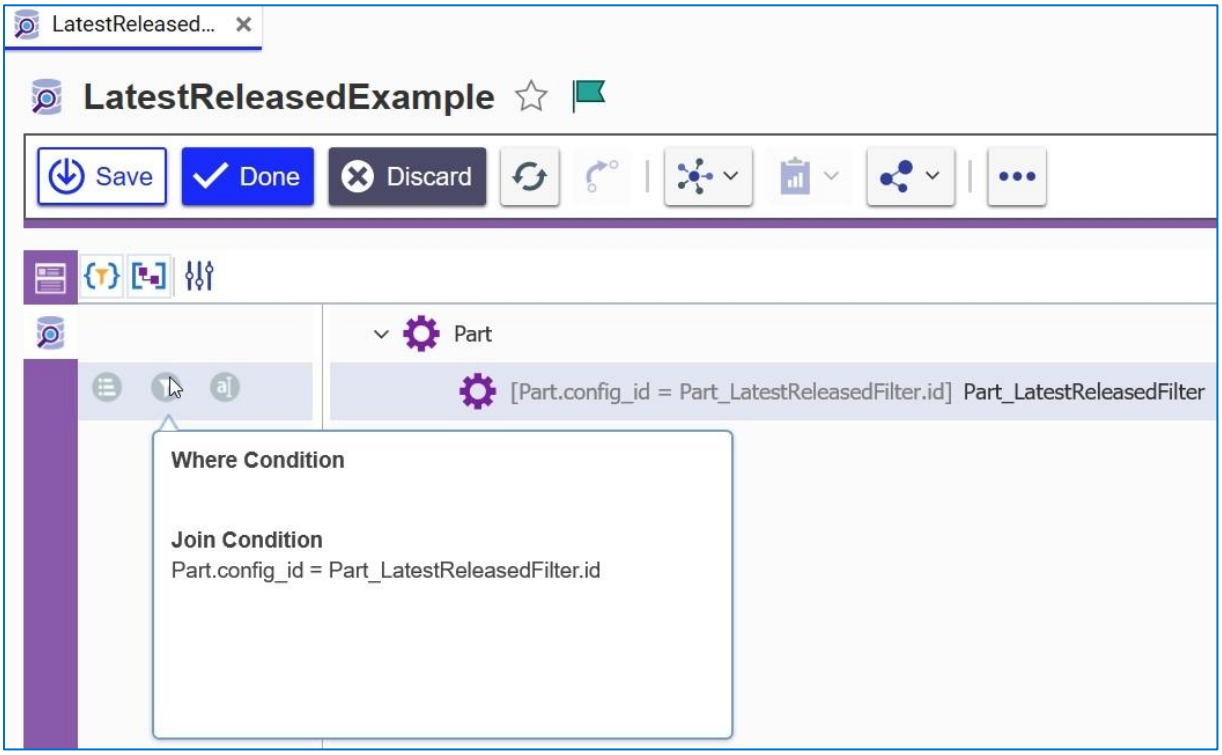

Figure 54.

12. Click the **Where** condition icon to create the where condition for the query as shown in Figure 55.

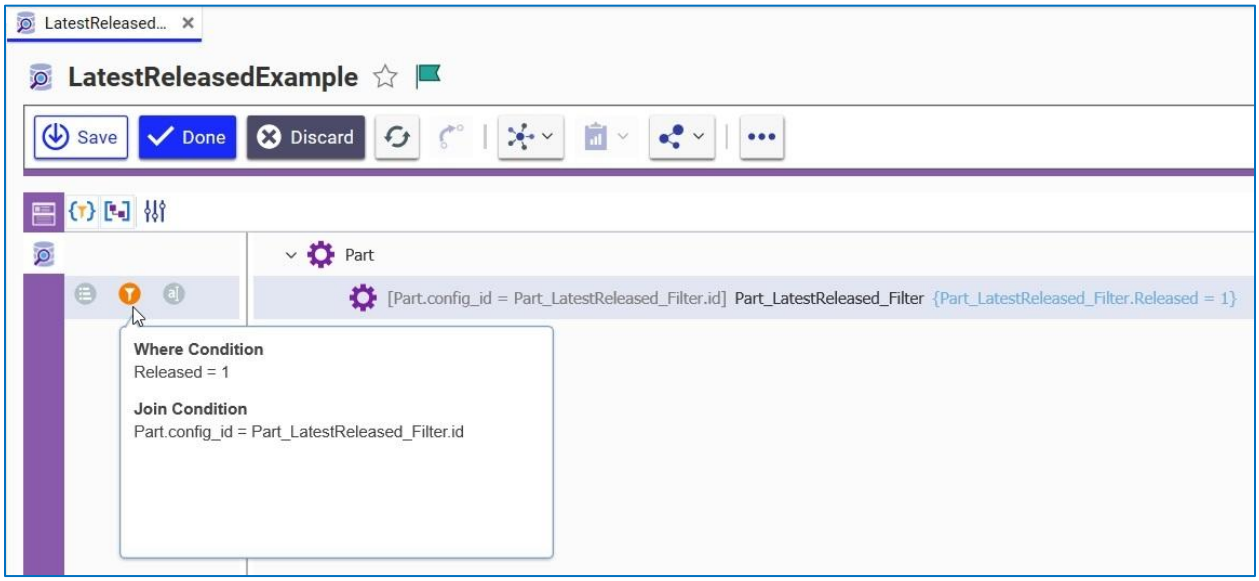

Figure 55.

13. Click **Save**.
14. Select the parent **Part** and click the **Where** condition button to create the expression as shown in Figure 56.

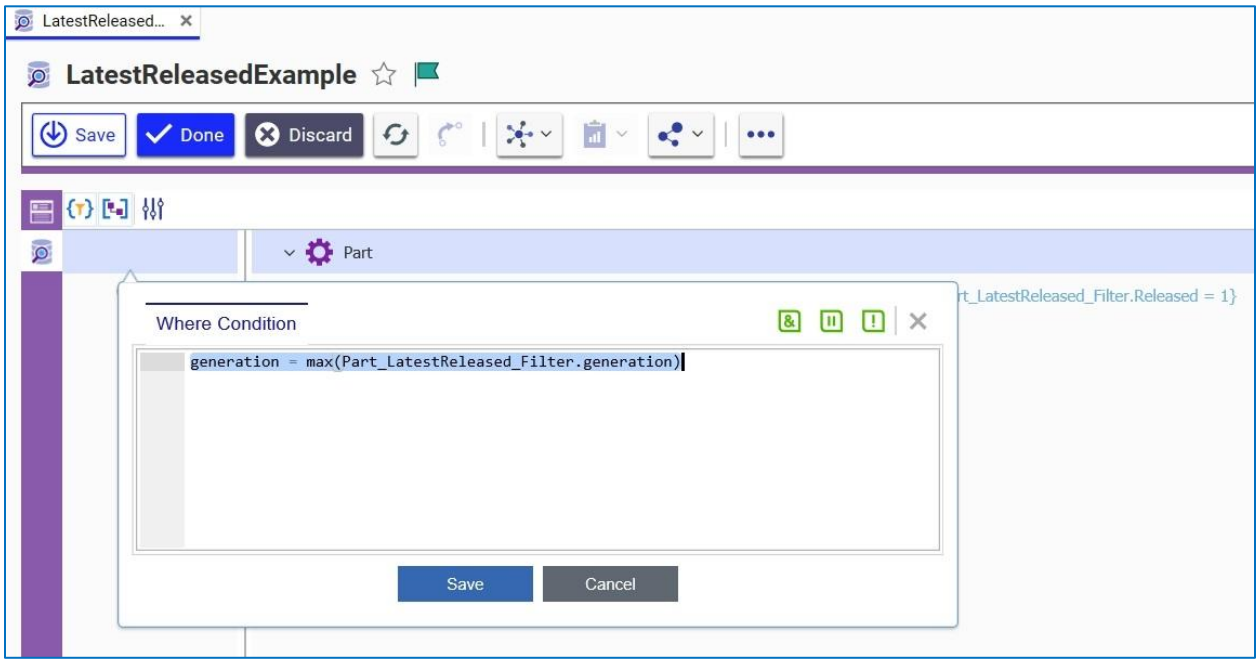

Figure 56.

15. Click **Save**. The **Show Editor** dialog displays the parent and child parts.

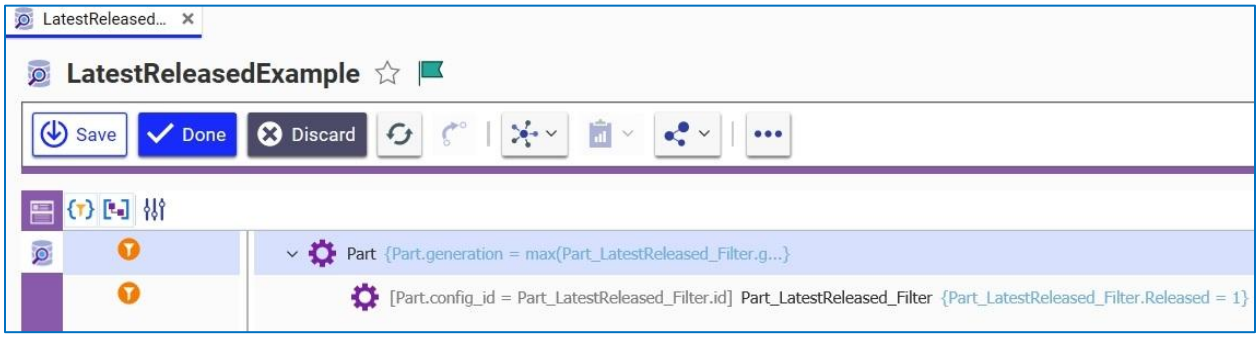

Figure 57.

The query returns the most recent version of released **Parts** associated with a particular **config_id**.

6 Sample Query with Exists() Function

The `Exists()` Function can be applied in a **WHERE** condition to filter Items based on the *existence* of related items. In essence, it's similar to a **LEFT INNER JOIN** in a SQL statement. To explain the use of such a feature, refer to the basic Query Definition shown in Figure 58.

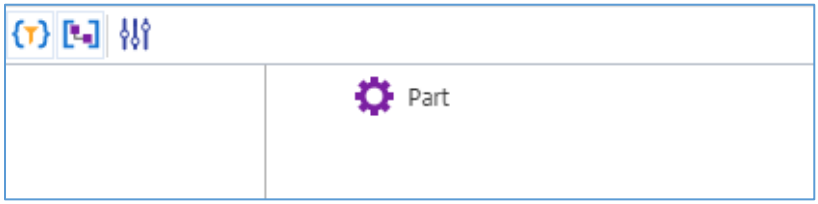

Figure 58.

This query will result in returning all Parts in the system. It's equivalent to the following SQL statement:

```
SELECT FROM Part
```

The addition to this Query Definition shown in Figure 59 will add related **Parts** through the **Part BOM** Relationship.

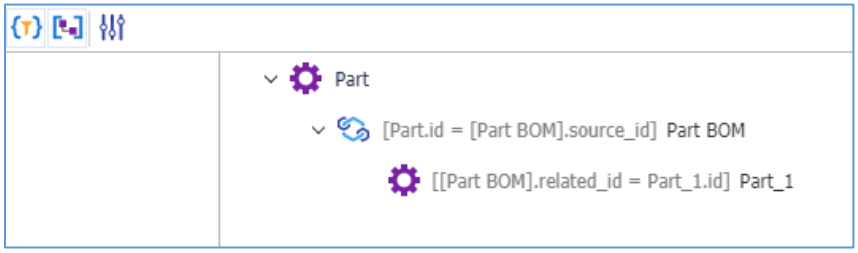

Figure 59.

This query will return all **Parts**, including those **Parts** that have a **Part BOM** Relationship along with the related (child) **Part**. It's equivalent to the following SQL statement:

```
SELECT FROM Part p
  LEFT JOIN Part BOM pb on …
  LEFT JOIN Part child on…
```

The addition to this Query Definition shown in Figure 60 will add the Condition `Exists([[Relationship]] Part BOM)` on the Context Part Query Item.

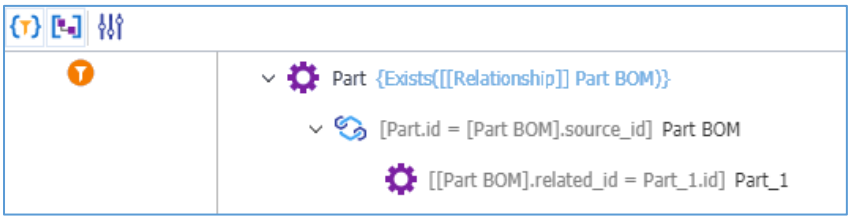

Figure 60.

This query will return all **Parts** that have a **Part BOM** Relationship along with the related (child) **Part**. It's equivalent to the following SQL statement:

```
SELECT FROM Part p
 INNER JOIN Part BOM pb on …
  LEFT JOIN Part child on…
```

The effect is to restrict the **Part** Items returned for the Context Query Item to only those that have related **Parts** using the **Part BOM** relationship. Any related Query item can also have conditions applied to it. For example, using the previous example, the **Part BOM** Query item can have a condition that restricts Items based on a `Quantity > 2`. In this case, the entire Query Definition will only return **Parts** that have related **Part BOM** Items that have a Quantity of at least 3.

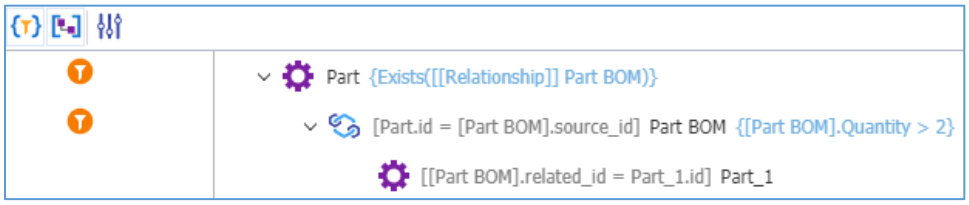

Figure 61.

The `Exists()` function can be used on any Query item with related content. Note that the Condition Editor will automatically offer valid `Exists()` statements. See Figure 62 for an example.

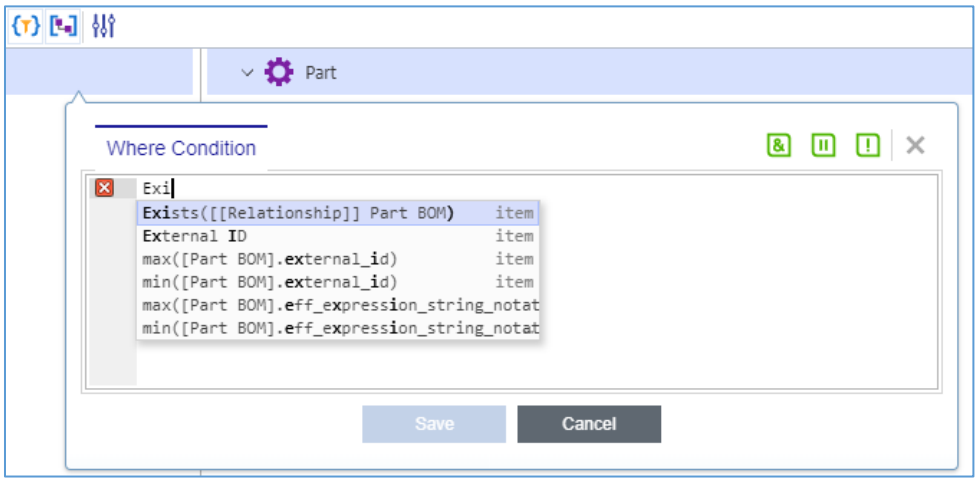

Figure 62.

In this case, beginning to type the `Exi` term will isolate the list of valid statements and include all valid ways to use `Exists()` for the selected Query item.

7 Structure Resolution Parameters

We mentioned Structure Resolution briefly in the Parameters section above. The Structure Resolution parameter sets query criteria to return a specific configuration of items with respect to Released state and version (generation).

Structure Resolution Modes:

- **Default**: a structure as physically stored in the database. Technically speaking, results will contain Items with ID values in `related_id` Property of Relationships in the structure without interpretation of LifeCycle State or availability of new versions (generations).
- **Latest**: uses the generation of the Item with the highest value.
- **Latest Released**: uses the generation of the Item with the highest value which is also Released.
- **Latest Released or Latest**: uses the generation of the Item with the highest value which is also Released; otherwise, the Latest if there are no Released generations.

By default, the structure is only displayed as stored in the database. Without dynamic view parameters users would need to spend more time examining work-in-progress changes to components in a structure. Dynamic structure resolution is important feature for BOM structure, Effectivity, 3D Visualization, and other Aras Innovator features.

In this section we describe in detail how Structure Resolution can be configured for a Query Definition using the Query Builder.

Note: In Aras Innovator 12.0 SP2 and SP3, there was an error in process Structure Resolution Queries for the Dynamic Viewer in that it would assign the Structure Resolution filter to the root CAD Item. Thus, if a Structure Resolution query was meant to include all Latest Released CAD Items (and the root CAD Item was not Released, the query would return nothing. This has been fixed in 12.0 SP4 and later releases.

7.1 OOTB Example: BOM Structure Resolution

Structure Resolution comes pre-configured for Part BOM Structures OOTB. If you wish to enable Structure Resolution for your own Query Definition its useful to examine the OOTB configuration used in Query Definition PE_BomStructure for reference. Other than naming conventions and base query structure you will follow the same procedure as used for the OOTB implementation when adding Structure Resolution to your Query.

7.1.1 Display Hidden Tabs Under Query Definition

By default, Tabs under the Query Definition itemtype are hidden. Query Definition Itemtype have Relationships to Parameters and Event handlers. We will need them displayed to examine and configure Structure Resolution.

As administrator, edit the Itemtype qry_QueryDefinition and change the Default Structure View to “Tabs On” as shown on right. If the tabs do not appear for Query Definition form, then you may need to exit the Query Definition and re-open to refresh the form.

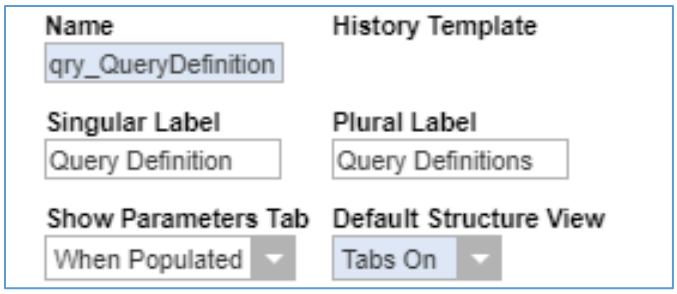

Figure 63.

7.1.2 Check Parameter Tab

Once displayed, you will have access to relationships under the Query Definition as Tabs. Recall from **Section 2.4** that Parameters are essentially runtime variables configured in the Query Builder to provide runtime variables (i.e. search criteria) to the underlying query.

Under qry_QueryParameter tab, you will see all Query Parameters defined for this query. For Bom Structure OOTB, Parameter “StructureResolutionMode” is pre-configured with the default value “Aras.Resolution.EntryPoint.Default” as shown below.

If adding Structure Resolution to your custom query, you will need to add a Parameter and assign the Default Value “Aras.Resolution.EntryPoint.Default” as shown below. This procedure is detailed in Section 7.2.

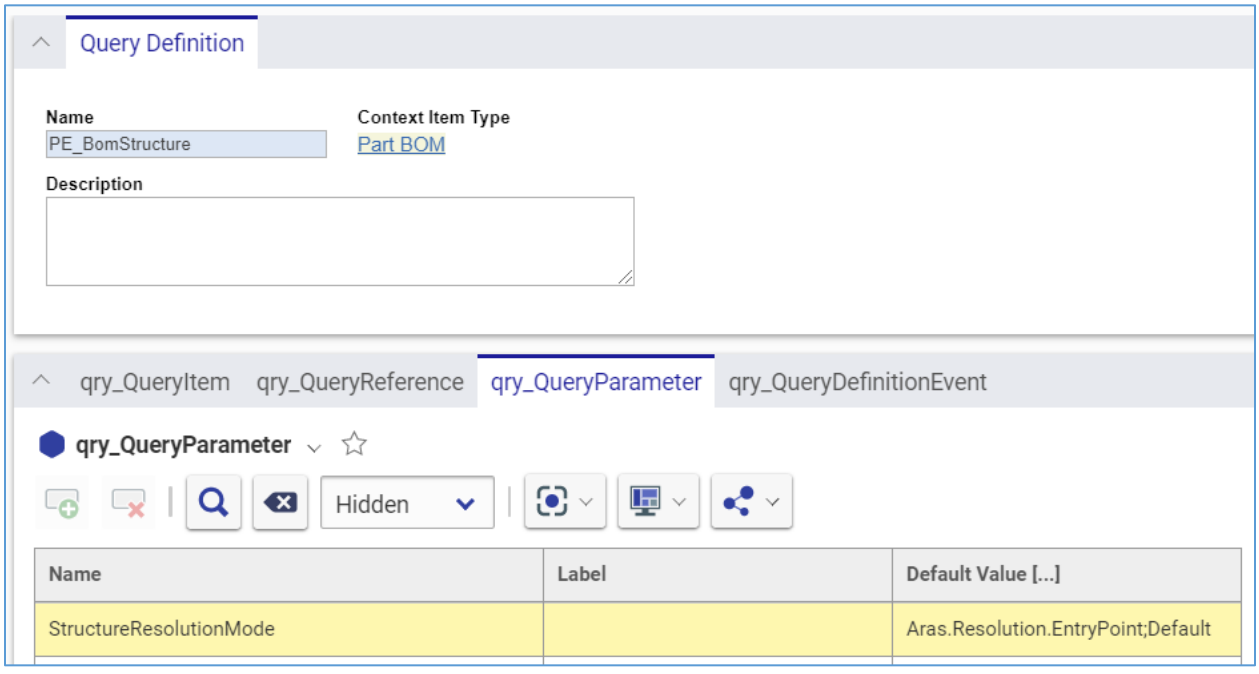

Figure 64.

7.1.3 Query Definition Events Tab

Beside the Parameters tab in the Query Definition form is the *qry_QueryDefinitionEvent* tab. In the OOTB Bom Structure Query Definition you will see an Event Handler that tells the query to run a method named "*qry_ResolveStructureEntryPoint*" before query execution (event "OnBeforeExecute"). This method is provided OOTB, providing the logic to resolve structures according to the selected Parameter value (Latest, Latest Released, Latest Released or Latest).

When adding Structure Resolution to your custom query, you will need to add this event handler and define the method and event name exactly as set in the BOM Structure query definition.

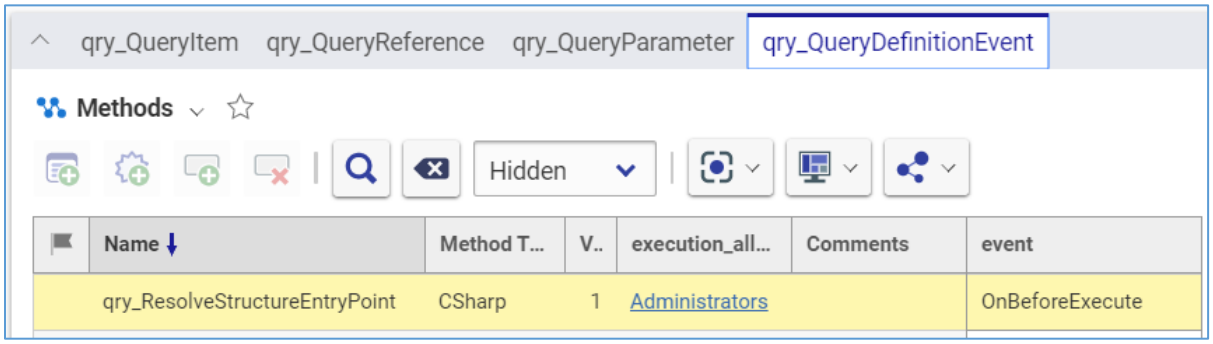

Figure 65.

In Summary, the Parameter and Event Handler tells the Query Definition to (a) allow runtime entry of Latest/Released criteria and (b) specifies method logic to execute to enforce Structure Resolution behavior on execution of a query.

7.1.4 Reference to Parameter in Join Condition in Query

Once the Parameter is added to the Query Definition and the Event Handler set, the Parameter can be dereferenced in the Join Condition of the query item to be resolved using the `$Name` syntax below. In the BOM Structure example, the resolved query item is the Part node, therefore the Join condition is edited to use `'$StructureResolutionMode.parent.related_id' = Part.id`.

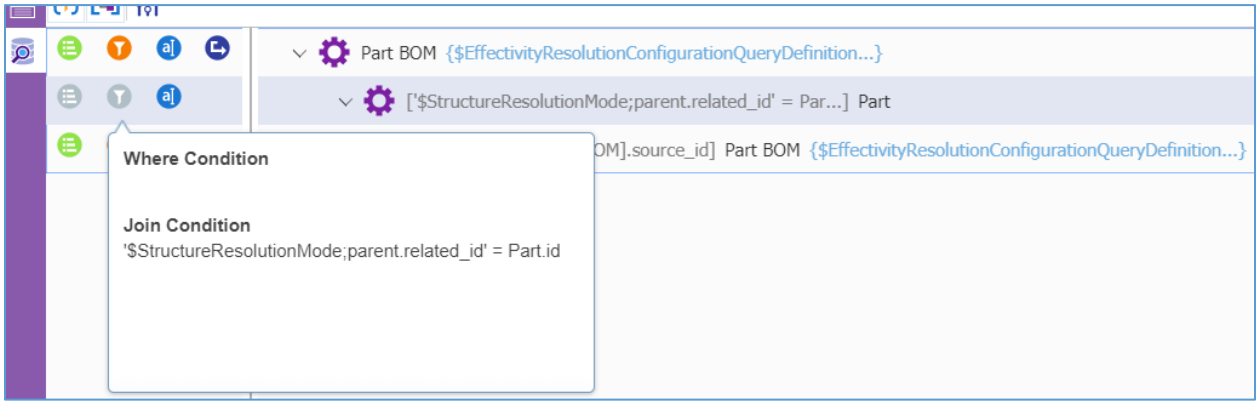

Figure 66.

Sections 7.1.1 – 7.1.4 describe the same configuration settings you would use to enable Structure Resolution on your custom query, or another Query Definition applied to a structure of related versionable items such as CAD Structures. The next section covers this procedure in detail.

7.2 Enabling Structure Resolution in a Query Definition

Structure Resolution can be applied to any Item Relationship structure of Versionable items that follows the same rules as Part BOM structures – Life Cycle State "Released" (is_released Boolean), incremental Revisions with interim versioned generations, etc.

- CAD Structure is an example of such a structure, and Structure Resolution can be applied to Query Definitions against this data model in the same manner as Part BOM structures. We use CAD Structure as an example of this procedure which is documented in greater detail in the *Aras 3D Visualization - Administrator Guide* document.

Per the Part BOM Query Definition we examined in Section 7.1, Structure Resolution requires a Parameter with the Resolution Entrypoint, an Event Handler must be added to the Query, and the Parameter must be referenced in the Join Condition for the resolved itemtype.

Adding the Structure Resolution configuration to a CAD Structure Query will involve all of these steps as well.

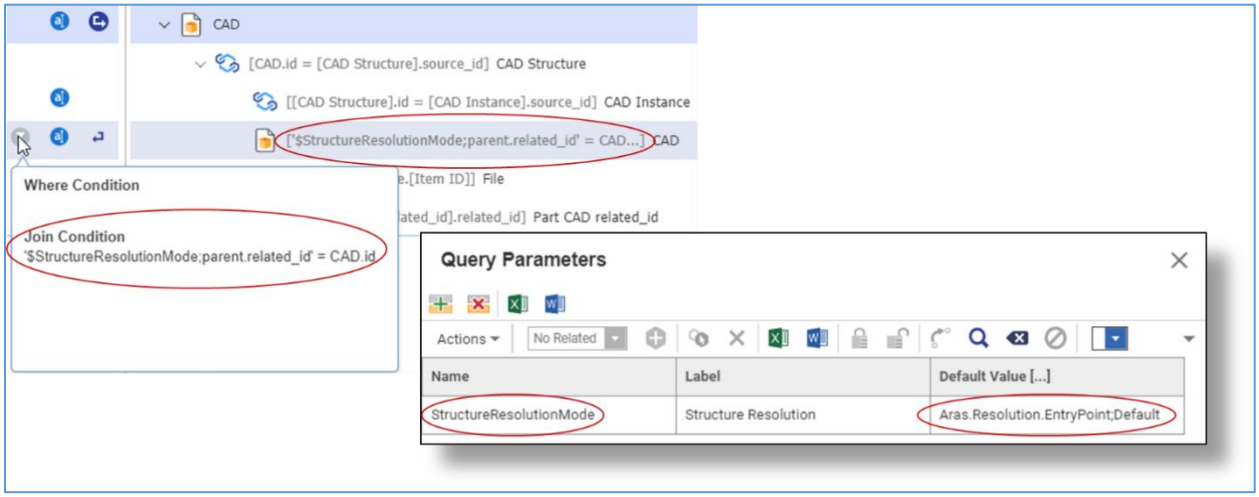

Figure 67.

Step 1: Create Structure Resolution Mode Parameter

Applying a specific Structure Resolution Mode ('Default', 'Latest', etc.) is done using a Query Parameter. Using the Query Parameters Dialog, add a Parameter to be used for Structure Resolution Mode. For example '`StructureResolutionMode`' and assign the default value '`Aras.Resolution.EntryPoint;Default`'

Note: Be sure to check the syntax of the Default Value for the Parameter. It is used to select a specific List Item as set in the Tree Grid View Definition and described below in Step 4.

Step 2: Incorporate the Query Parameter in Join Condition

The added Query Parameter must be incorporated into the Query Definition in order for it to be used/exposed to the end-user. Add it as part of the Join Condition of the Child CAD Query Item as shown in **7.1.4**. Replace the default Join Condition – e.g. '`[Part BOM].related_id = Part.id`' with the following – '`'$StructureResolutionMode;parent.related_id' = Part.id`'

Part BOM and Part are used for this example, but it could just as well be '[myLinkage].related_id = myNode.id' being replaced by '`'$StructureResolutionMode;parent.related_id' = myNode.id`'.

Note: Be sure to check the syntax of the Join Condition. It must be as specified above (without the outer quotes). This assumes the value StructureResolutionMode was used for the Query Parameter Name. If you named the Parameter differently, e.g. myParam, then you would naturally need to dereference it as $myParam in the expression.

The Structure Resolution Mode Query Parameter can only be inserted once in a Query Definition. It cannot be used/applied to more than one Query Item.

Step 3: Add Structure Resolution Method to Query Event Handler

Show the Relationship Tabs for the Query Definition in the Form View if they are not shown. Tabs On/Off is set in the Query Definition itemtype (qry_QueryDefinition).

Select the `qry_QueryDefinitionEvent` Relationship Tab. A Method needs to be added to the `OnBeforeExecute` event. To do this, select the **Select Items Icon** to create a Relationship and select the `qry_ResolveStructureEntryPoint` Method and assign it to the `OnBeforeExecute` event as shown in **Error! Reference source not found.**.

Figure 68.

Step 4: Map the Query Parameter in the Tree Grid View Definition

In order for the Query Parameter to be used in a **Tree Grid View**, it must be enabled (made visible) in the Tree Grid View Definition.

Open the Map Parameters Dialog in the Tree Grid View Definition within the Editor View. All Parameters defined in the Query Definition used by the Tree Grid View will be listed.

Select the **Visible** check box for the `StructureResolutionMode` row. The **Data Type** is a `List` and the **Data Source** is the List `qry_StructureResolution`.

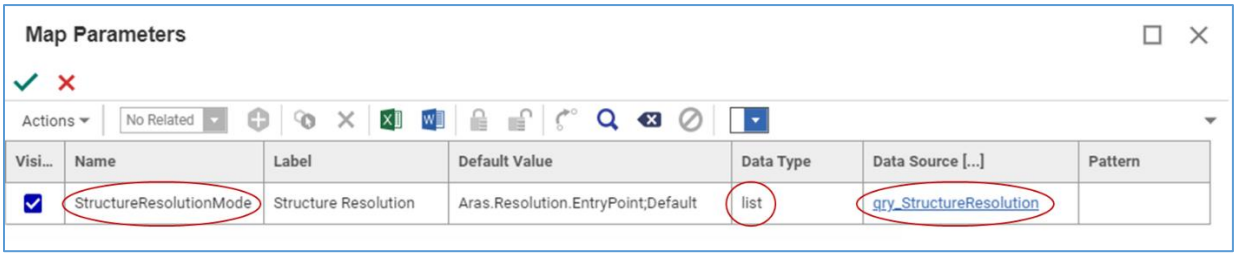

Figure 69.

With this Configuration in place, users will be able to select specific Resolution Modes by opening the Query Parameters Dialog in the Tree Grid View used within the 3D Viewer.

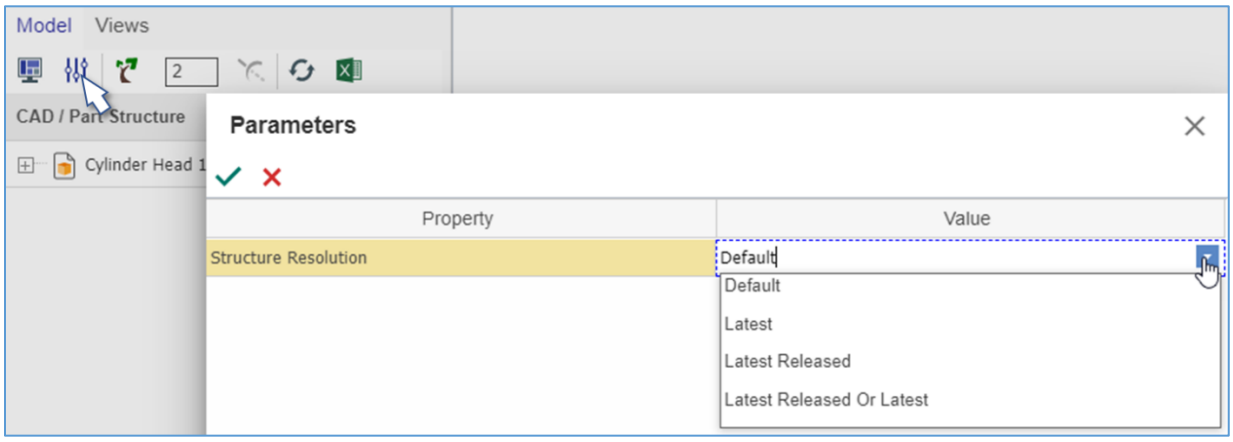

Figure 70.

7.2.1 Applying a Where Condition Against Resolved Items

In Aras Innovator 12.0 SP4 and later releases it is possible to add a Where Condition to a Query Item that is used for Structure Resolution (e.g., CAD in this example). Doing so allows the Administrator to apply an additional condition (filter) when resolving the Structure of Items with fixed Relationships (such as CAD / CAD Structure). For example, assuming the CAD Structure in the following diagram:

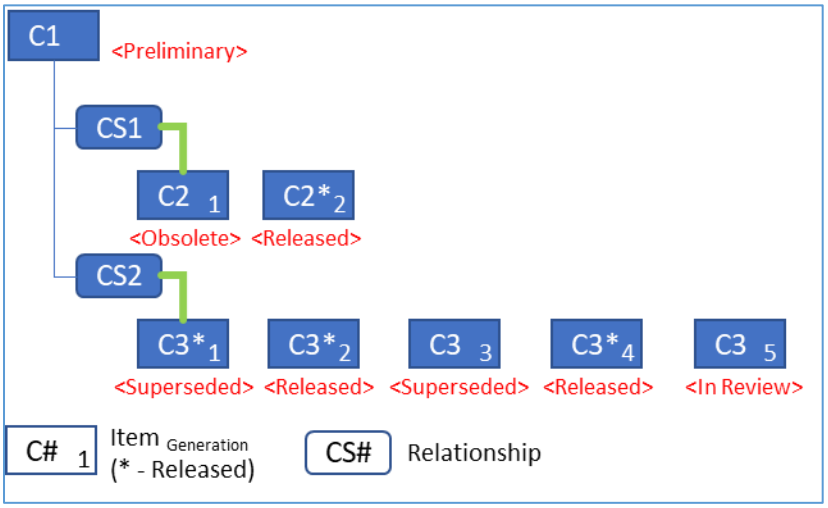

Figure 71.

If the Administrator would like to use Structure Resolution of 'Latest' and apply an additional condition to only include CAD Items that were *In Review* (that is, with a CAD.State = 'In Review') then Where and Join conditions like the following could be added to the Query Item:

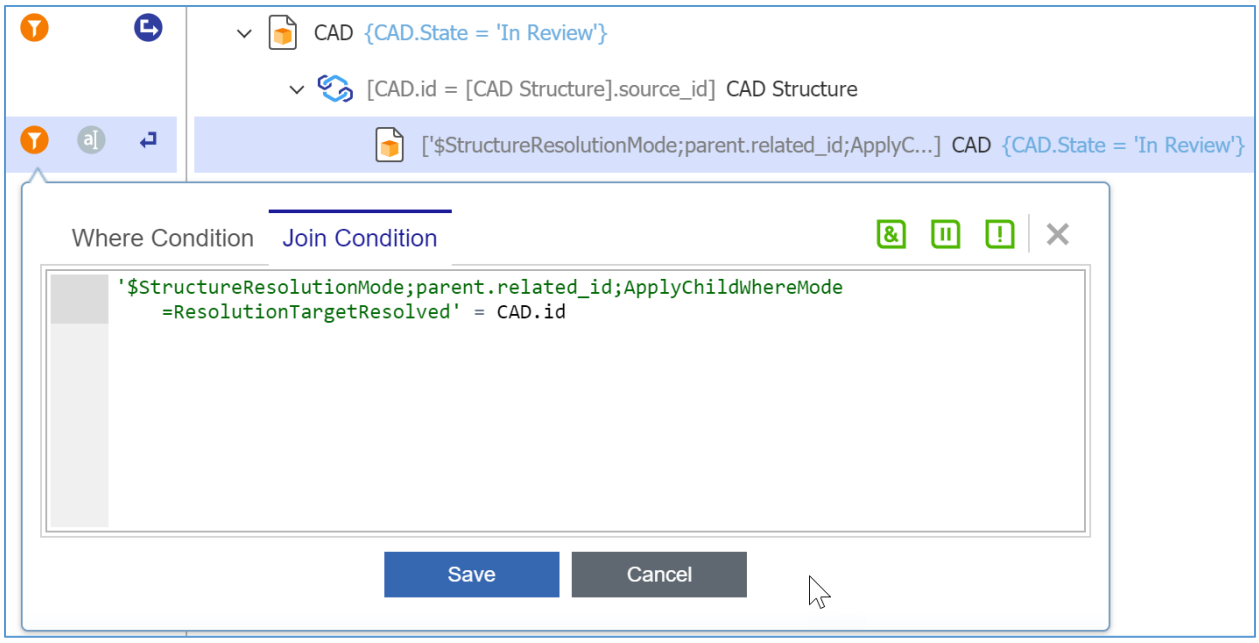

Figure 72.

Note the addition of the left-side of the Join Condition:

`'$StructureResolutionMode;parent.related_id;ApplyChildWhereMode=ResolutionTargetResolved' = CAD.id]`

Executing this Query Definition, with Structure Resolution of `'Latest Released'` with return:

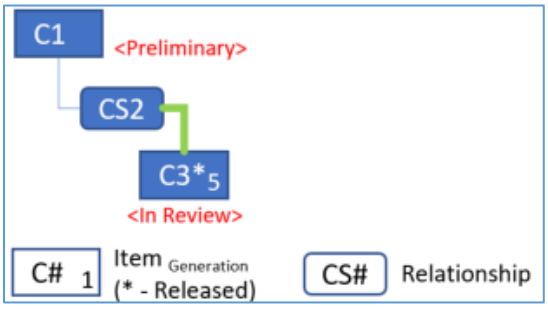

Figure 73. Structure Resolution Logic with Condition Result

In this example, only the CAD Item – CS3, version 5 is returned because of the added Where Condition of `'CAD.State = 'In Review'` applied to the resolved results of the Structure Resolution Join Condition using the selected Mode parameter.

For more details and extensive examples of how Resolution Modes are applied in BOM Structures, Effectivity, and 3D Visualization please refer to the following documents:

- *Aras 3D Visualization - Administrator Guide*
- *Aras Innovator Product Engineering - User's Guide*