The KWQL Query Builder - User's Manual

The KQB User Interface

The KWQL Query Builder user interface includes the complete functionality to edit visKWQL queries and provide help to the user while he is working with the system.

The user interface consists of the following parts:

1) The Menu Bar The Menu Bar allows access to all the different visKWQL constructs, which can be used to form a query. They are grouped into Resources, Qualifiers, Operators, Other items, and Examples (please see section for an explanation of those constructs). The menu is a simple drop-down menu, which displays items or sub-menus when the mouse is hovered over it. The Qualifiers are grouped after the Resources they can belong to (see Fig. 2) to help select appropriate Qualifiers for the right Resources. When a menu item is clicked, the appropriate visKWQL box is created and added to the workspace

2) Undo & Redo These buttons allow to undo or redo the last action. There is no limit on how many steps can be undone or redone, and every action, like dragging and dropping a box, or entering a character into a text field, is considered one step

3) The Workarea This is the main workspace, on which visual queries are displayed and edited

4) The Tooltip Pane This area displays help about a specific visKWQL element, when the mouse is moved over it

5) The Hint Pane The Hint Pane provides help in the form of hints of what to do next, or error explanations and suggestions of how to solve an error or problem that might exist in the currently displayed query

6) KWQL Output In this field, the textual KWQL query is displayed. It is updated after every user action. If the current visual query is syntactically incorrect, an error message will be displayed in its stead

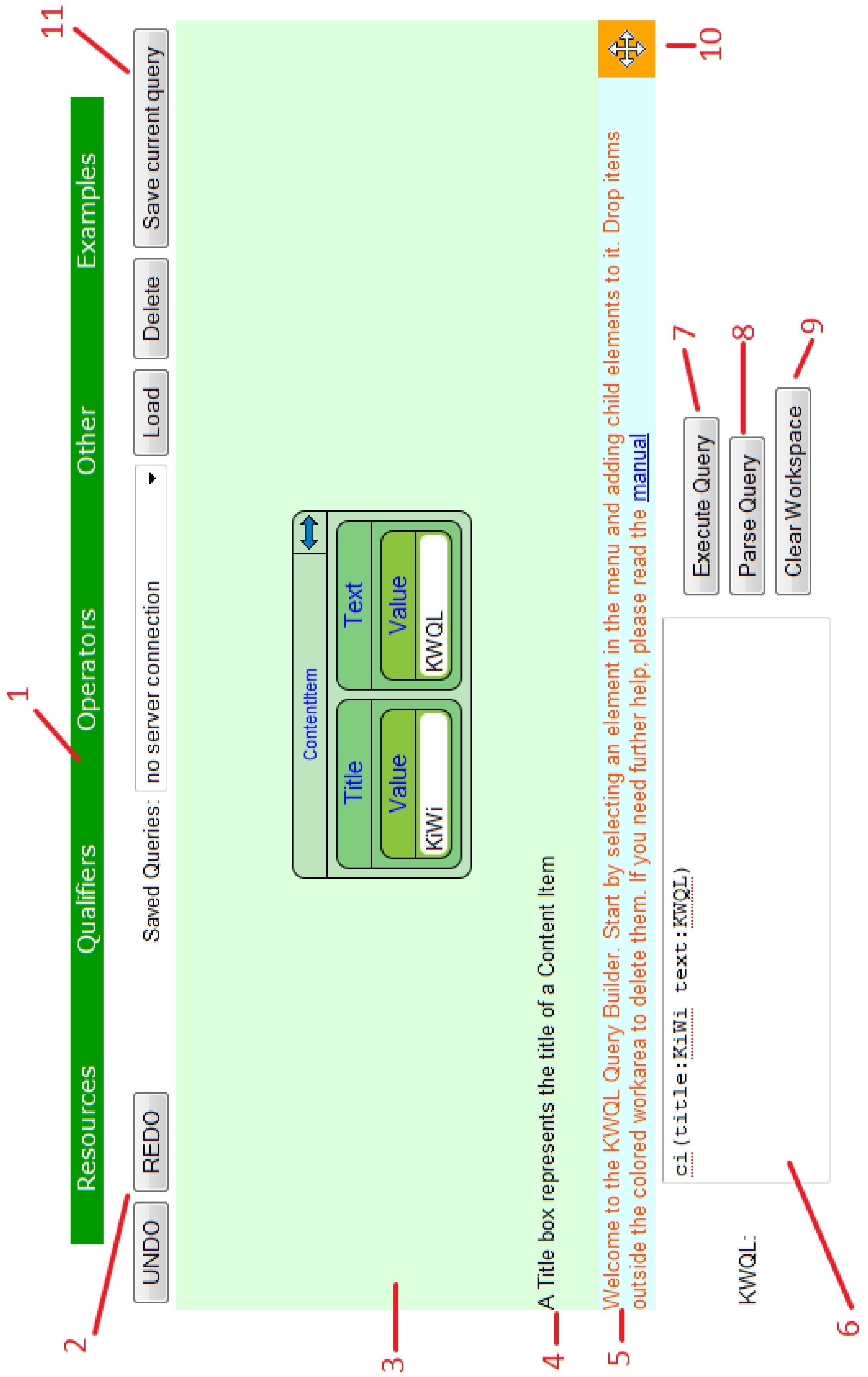
7) Execute Query Button Pressing this button will execute the current query (assuming it does not contain an error), and open the result page

8) Parse Query Button This button will parse the textual query currently in the KWQL Output text field (which the user can enter or edit) and replace the current visual query with it

9) Clear Workspace Button The Clear Workspace Button simply deletes both the visual and textual query. This action can be undone with the Undo Button

10) The Resize Box This box can be dragged with the mouse to change the size of the Workarea

11) Saved Queries When logged into KiWi, the drop down box displays a list of saved queries. When a list entry is selected, and the Load button is pressed, that query will be loaded and replace the current query in the Workarea. The Delete button will remove the current list entry and delete the saved query. The Save button will ask for a name, and then save the currently displayed visual query under that name on the server



A Title box represents the title of a Content Item

Welcome to the KWQL Query Builder. Start by selecting an element in the menu and adding child elements to it. Drop items outside the colored workarea to delete them. If you need further help, please read the manual

```
ci(title:KiWi text:KWQL)
```

Figure 1: KWQL Query Builder GUI

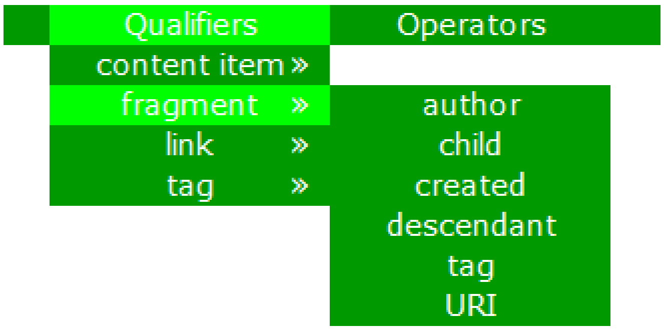


Figure 2: A part of the KQB menu

(vis)KWQL elements

Every visual element of visKWQL corresponds to one textual element of KWQL. To be able to create meaningful visKWQL queries, it is thus important to understand the meaning of those elements. While the KQB tooltips give a short explanation of all elements, so that the user is not required to read a manual to be able to construct useful queries, a better understanding of the underlying conceptual model and the meaning of different language elements will be helpful, so that a full listing of all elements is given here.

Resources

In KiWi's conceptual model, the wiki structure is made up of four different resources. These resources are directly queryable by KWQL and visKWQL:

Content Item (ci): KiWi's most basic and most important structure is the *Content Item* (the lower case or abbreviated name, like ci, of an element as given first in the list is always how the element is named in KWQL). Content Items are holding textual or multimedia data, and correspond to wiki pages. While each Content Item can be seen as one wiki page, a Content Item can also include other Content Items, which in turn can include other Content Items, so that each Content Item can be a tree of other Content Items. An example would be a wiki page for "Elephant" with a textual description and a photo of an elephant. While the whole wiki page would be one unique Content Item, this Content Item would include one Content Item with the textual description of the elephant, and one Content Item with the photo. When the photo is clicked in the "Elephant" wiki page, the elephant photo's own wiki page (Content Item) will be displayed. One Content Item can thus include a series of other Content Items, however, there are neither overlap nor cycles in Content Item compositions

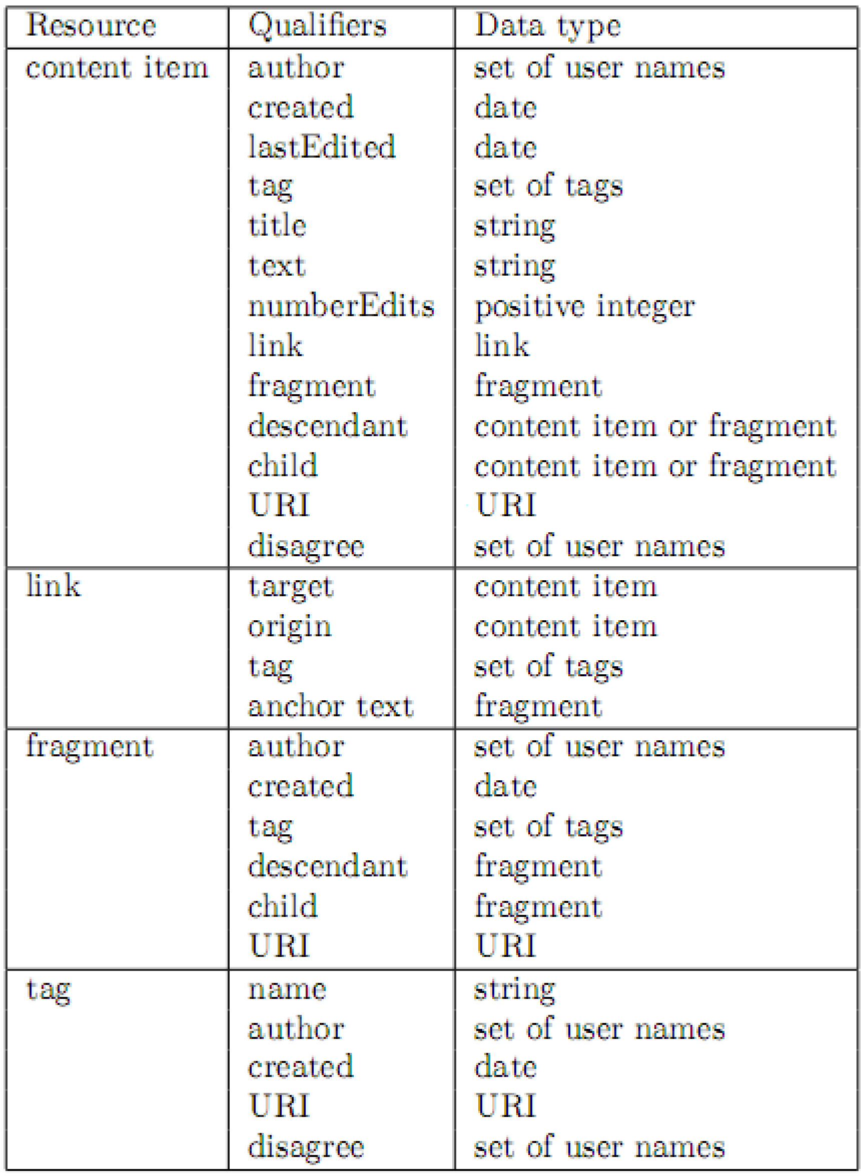
fragment: A *Fragment* is a piece of textual data within a Content Item. Its length is arbitrary, it can be one word, one sentence, one paragraph, or any other meaningful portion of text that a user might want to annotate. Fragments can include other Fragments, but there is no overlap, and one Fragment can not include text from more than one Content Item

Link: *Links* in KiWi are the same as hypertext links on web pages. They have a source and a target Content Item, and an anchor. When a user clicks the anchor on the source wiki page, he will be shown the target wiki page

tag: *Tags* are the annotations in KiWi. Users can annotate Content Items, Fragments and Links. Each Tag has a name, which is free-form information or meta-data which the user gives to some resource. For example, an "Elephant" Content Item could be annotated with the Tags "animal", "lives in Africa", "grey" or any other fitting information

Qualifiers

Each Resource has accompanying data and meta-data, some of which is automatically created by the system (like the creation date of a Content Item), and some of which is created by users (like the text of a Content Item). These are called Qualifiers of the Resources, and are also directly queryable by (vis)KWQL:



Resource	Qualifiers	Data type
content item	author	set of user names
	created	date
	lastEdited	date
	tag	set of tags
	title	string
	text	string
	numberEdits	positive integer
	link	link
	fragment	fragment
	descendant	content item or fragment
	child	content item or fragment
	URI	URI
	disagree	set of user names
link	target	content item
	origin	content item
	tag	set of tags
	anchor text	fragment
fragment	author	set of user names
	created	date
	tag	set of tags
	descendant	fragment
	child	fragment
	URI	URI
tag	name	string
	author	set of user names
	created	date
	URI	URI
	disagree	set of user names

Figure 3: Resources and accompanying Qualifiers

author: Content Items, Fragments and Tags each have an Author qualifier. This Qualifier is a list of the names of the users who have authored that particular Resource

created: The Created Qualifier is the date that a Content Item, Fragment or Tag was first created

lastEdited: This Qualifier is the date a Content Item was last edited

tag: The list of Tags associated with a Content Item, Link or Fragment

title: The title of a Content Item

text: The text of a Content Item

numberEdits: The number of times that a Content Item has been edited

link: A Link within a Content Item

fragment: A Fragment within a Content Item

child: A Content Item or Fragment that is a direct child of another Content Item or Fragment, meaning it is directly included in it

descendant: A Content Item or Fragment that is a direct or indirect child of another Content Item or Fragment, either directly included in it, or included within another child

URI: A Unique Resource Identifier is associated with every Content Item, Fragment and Tag. Through an URI, each of these Resources is uniquely identifiable and reachable

disagree: The Disagree Qualifier is a list of user names who do not agree with the content of a Content Item or with a Tag

target: The Content Item a Link points to

origin: The source Content Item of a Link

anchorText: A Fragment within a Content Item, to which a Link is anchored

name: The Name Qualifier specifies the text of a Tag

Operators

Operators in (vis)KWQL are associated with one or more other language elements and allow the creation of complex queries:

AND: The AND operator forms the logical conjunction of two or more other elements. In visKWQL, all children of an AND box must be true in a query. For example, a query in which author:Mary and author:Carl are both children of an AND within a Content Item, will only return Content Items which have been written by both Mary and Carl together, but will not return ones where only one of the two was the author (visual examples for the use of operators will be given in the next section)

OR: The OR operator forms the logical disjunction, where only one of its children must be true, but not all of them

NOT: The logical negation. For example, if a visKWQL query includes author:Mary as the child of the NOT operator within a Content Item, the query will return all Content Items where Mary was not the author

COUNT: This operator can only have a variable as its child, and will return the number of values within that variable. This operator is only meaningful in the construct part of a Rule

ALL: The ALL operator will return (in the construct part of a rule) all values saved in a variable. It has two children: the variable whose values it extracts, and a string which is used to delimit those values. For example, if an ALL operator has a variable child containing the values x, y and z, and a delimiter child “ - “, it will return the string “x - y - z”

SOME: The SOME operator works like the ALL operator, only that it has an additional numerical child, which sets the maximum number of values to return. The COUNT, ALL and SOME operators are all used to construct textual values, like a Content Item's Text or Title, within rules

OPTIONAL: The OPTIONAL operator allows the specification of an optional element within the query part of a rule. It has three children: The first child is a default value. The second child is the non-optional part of the query, and the third child is a variable assignment to some Qualifier. When the Qualifier is present and has a non-empty value, this value will be bound to the variable. If it is not present or has an empty value, the default value will be bound to the variable

Other elements

Some additional elements of (vis)KWQL do not fall into the above categories. These are:

Value: A simple string, which is either the value of some Qualifier or Operator, or a Keyword on its own. As a value of a Qualifier within a query, it can be seen as a value restriction. A query consisting only of the Qualifier author with the value "Mary" will only return those Content Items which have an author with the name Mary

Rule: A rule allows the creation of new Content Items. It consists of two parts: a query and a construct part. The first child (in visKWQL) is the construct part, and specifies a Content Item that is to be created. The second child is a normal query, which usually contains variable definitions which are used in the construct part (please see the next section for an explained visual example)

Variable: A variable definition to be used in a rule. In the query part, a Qualifier can have a variable definition as a child, instead of a value. In this case, the Qualifier's value is bound to this variable (or the Qualifiers' values are bound to it, if the query results in multiple items). In the construct part of the rule, these values can than be extracted with the ALL or SOME operators

Binding: A (variable) Binding is a combination of value restriction and variable binding. In visKWQL, a Binding contains two children. The first is a value, which restricts the Qualifier the Binding is associated with. The second child is a Variable, to which the actual value will be bound. This way it is possible, for example, to bind all Content Item Titles which include the word "wiki" to a variable and construct a new Content Item with links to all those wiki-themed Content Items, with the link anchor being the full title of the specific Content Item

SPARQL: KWQL allows the inclusion of SPARQL queries. In visKWQL, this can be achieved by entering the textual SPARQL query into a SPARQL box

Compound: The Compound is an element specific to visKWQL. In KWQL, a Qualifier in a query can only have one child (a Value, Variable, or Binding), but in a rule's construct part, some Qualifiers like Text and Title can have multiple children. Qualifiers in visKWQL only have one child, so to allow them to have multiple children within a rule's construct part, multiple elements can be put into a Compound box, which can then be added as a child of the Qualifier

Query Construction

Creating Queries with Drag & Drop

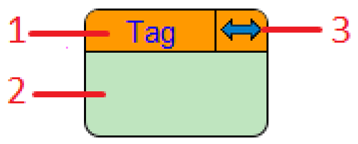


Figure 4: A visKWQL box

All queries in visKWQL are created by nesting boxes by dragging and dropping them onto each other.

Boxes represent all language elements, and work in the same way, no matter if they represent a Resource, an Operator, or something else.

Figure 4 shows an empty box. All boxes have a *Label* (1) and a *Body* (2). Some boxes additionally have a *Resize Box* (3).

The box label always shows the name of the box, which is the KWQL element (Resource, Qualifier, Operator, ...) it represents. The box body can hold child elements, which are either other boxes or text boxes, in which keywords, values, or variable names can be entered.

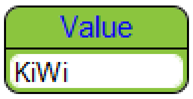


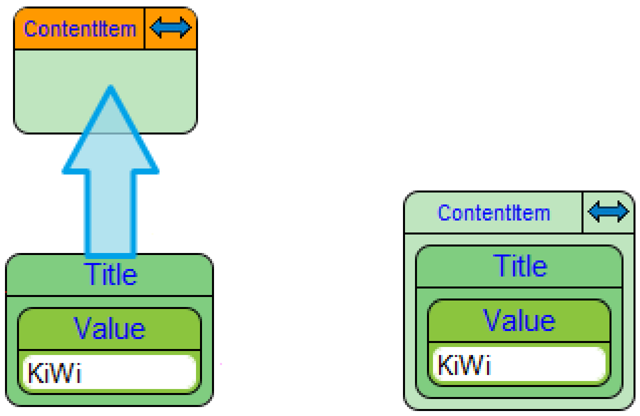
Figure 5: A keyword query

The simplest kind of query in visKWQL is a keyword query, which results in a search for the keyword over the entire wiki.

For a keyword query, select "Value" in the "Other" category of the menu bar. Clicking it will create a new Value box in the workarea. Now enter a keyword into its text field, and the query is finished. Figure 5 shows a keyword query for "KiWi". When executed, this query will result in a list of all Content Items (which are the default resource if the query doesn't explicitly declare one) which contain the word "KiWi" in any Qualifier.

The real power of (vis)KWQL, however, lies in its capability to directly query the structure of KiWi, specific Resources and Qualifiers.

To create a more specific query in visKWQL, a box can be added as a child box to another box, by dragging and dropping one box on another box's body (see Figure 6).



(a) Dragging into the body of another box

(b) The result

Figure 6: Adding a childbox

Assume that we want to find all wiki pages which have the word "KiWi" in their title.

First select "ContentItem" (since a Content Item represents a wiki page) in the Resource menu, and "Title" in the Qualifier menu, to add them to the workarea. Note that for convenience, Qualifiers like Title already include a keyword (Value) child when created.

Next, enter the keyword we are searching for, "KiWi", into the text box of the Title.

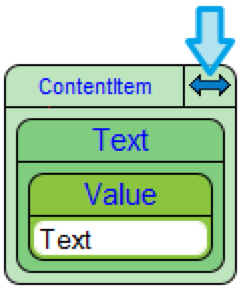
Now add the Title box as a child of the ContentItem box. To do this, "grab" the Title box by clicking the mouse down on its label, and drag it over the body of the ContentItem box (see Figure 6a). Let the mouse button go to drop it there. Figure 7c shows the result of the action. If the query is executed, it will return all wiki pages with the word "KiWi" in their title. You can also search for wiki pages where the title does not include "KiWi", but where the title is exactly "KiWi", by putting the keyword into quotation marks.

Note that, since Content Item is the default Resource, the Title box alone would produce the same query results as the Title box within a Content Item box.

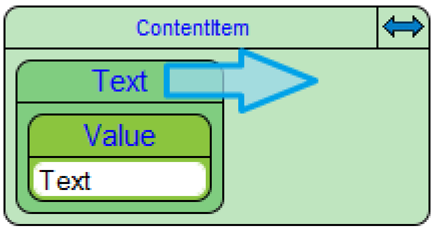
When You create queries this way, the system will check if the result of a drag & drop operation is valid, and will prevent You from dropping a box when that would result in an invalid query.

Also note that, when there are multiple (not nested) boxes on the workspace, only the box that was put first on the workarea is considered the current query and will be shown in its textual form in the KWQL Output field, and executed when the Execute Query Button is pressed.

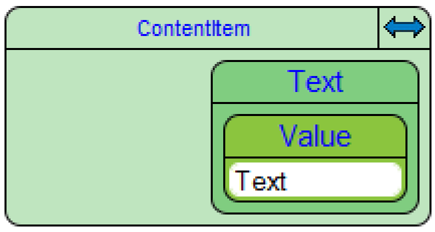
To remove boxes from the workarea, simply drop them outside the workarea to delete them. You can also delete everything by pressing the "Clear Workspace" Button. If You accidentally delete something, You can use the Undo Button to restore it.



(a) Clicking the Resize Box



(b) Dragging inside the box



(c) The result

Figure 7: Expanding and dragging within a box

Some boxes, like those representing Content Items, have a Resize Box in their label, indicated by a two-headed arrow. Boxes with such a Resize Box can have multiple children.

To add a second child to such a box, click the Resize Button (Figure 7a) to expand it. Now the box is wider, and has a free place in which another child box can be dragged.

It is also possible to arrange the children of a box in a new order, by dragging them within their parent box. As the example shows, in an expanded box, You can simply drag one child to a free position to place it there (Figures 7b and 7c). Now a second child can be dropped in the place the Text box has occupied before. You can click the resize box again at any time to hide the free space.

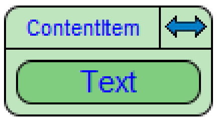
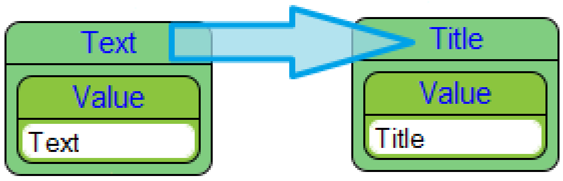


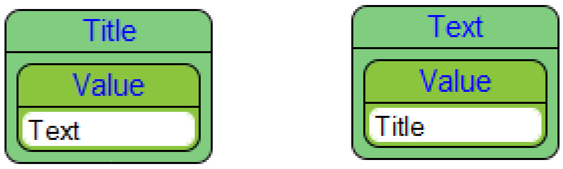
Figure 8: Hiding the body of a box

If You create more complex queries, they might get quite large and unclear. To hide parts of a query, You can click the label of any box to fold it together, to hide its body and all child boxes. Figure 8 shows the result of clicking on the label of the Text box of the query in Figure 9b. Clicking it again will show the body and child boxes again. It is possible to hide arbitrarily large parts of the query this way, making working with large queries easier.

The KWQL Query Builder also allows dropping a box on the label of another box, to switch the two boxes.



(a) Dragging and dropping on the label



(b) The result

Figure 9: Switching boxes

Figure 9 demonstrates this. Assume we want to search the Text of a Content Item instead of its Title for a specific keyword. One way to do this would be to drag the Title box out of the Content Item, then drag and drop the Text box on its body, then enter the keyword in the Text box.

A faster way to do this is to switch the Title and Text boxes. To do this, simply drag the Text box, and drop it on the label of the Title box (Figures 9a and 9b). Note that not the complete boxes change place, but that they switch their type. The child boxes stay in place. The example shows the type switch between two free-standing boxes, but the switch is possible no matter on what level of a query the two boxes are, or how many children they have.

If You drag a box that contains a text input field, like Value or Variable, onto the label of another such box, their text values are switched.

A requirement for a successful switch is that the two boxes are compatible. Qualifiers with textual values, like Title and Text in the example, can be switched, as can an AND with an OR operator. Switching a Qualifier with an Operator box, however, would result in an invalid query.

The system thus checks the validity of all drop actions, and will only accept meaningful switches and prevent forbidden ones.

Problems that the system can not directly prevent will be shown to You through red or orange colored box labels. In such a case, hover Your mouse over a to read the error message, and pay attention to the Hint Pane, which will inform You about the cause of the error and how to correct it.

Advanced Examples

In addition to the Resources and Qualifiers shown in the examples of the previous section, visKWQL also supports all KWQL operators.

The first kind of operators are connectives (the AND and OR operators) and negation (the NOT operator).

Assume You want to search for all wiki pages which do not include “KiWi” in their title. To create a query for this, put a Title box inside a NOT box, and then that NOT box inside a ContentItem box, instead of putting the Title box inside the Content Item box, which would be a query to return all wiki pages that do contain “KiWi” in the title.

If You want to search for all wiki pages which were either written by Mary or by Carl, put two Author boxes, with the values “Mary” and “Carl”, inside an OR box. Note that in boxes that can hold multiple children, like a Content Item, an implicit AND is assumed, so that putting the two Author boxes inside an AND box within a Content Item would be equal to putting them both directly in the Content Item, to search for all wiki pages written by Mary and Carl together.

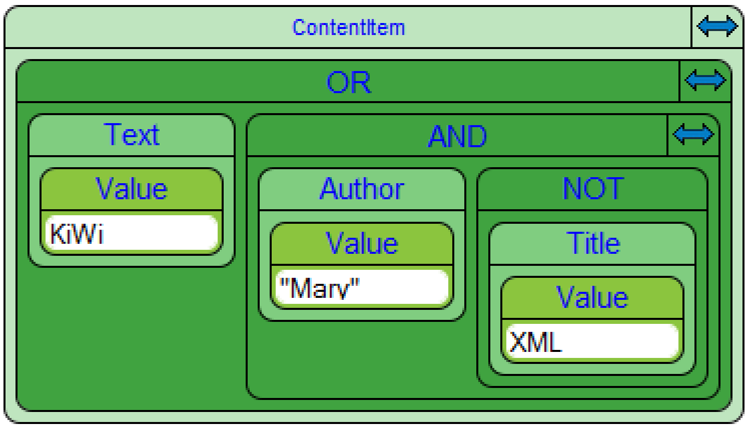


Figure 10: Using operators

Figure 10 shows a query that makes use of the AND, OR and NOT operators, which can be arbitrarily nested to produce the desired query. In the example, the resulting query returns all wiki pages with contain “KiWi” in their text, or which were written by Mary and do not contain “XML” in the title.

The other operators of visKWQL, ALL, SOME and OPTIONAL, can only be used within rules. Rules are used to create new content items. They consist of a query part, and a construct part which creates new data, based on the results of the query part.

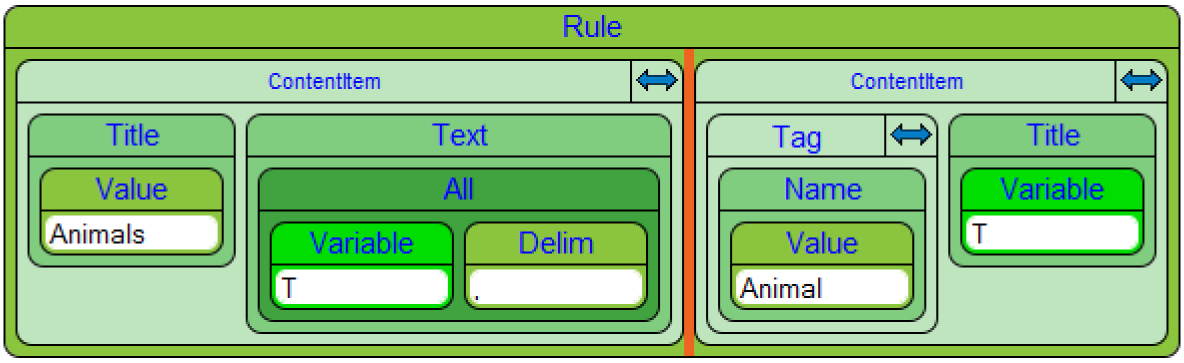


Figure 11: A visKWQL rule

Figure 11 shows a complete rule, which creates a new wiki page with a listing of all animals for which a wiki page exists.

The right side of the rule, the query part, contains a query as used before, only that it now contains a variable definition. In this example, the query will search through all wiki pages, and for those pages that have an “Animal” tag, it will save the title in the variable T (as mentioned before, when selected in the menu, Qualifiers come with a Value child. To put a Variable box in its place, select a Variable from the “Other” menu, and switch it with the Value child).

The left part of the Rule is the construct part. It creates a new wiki page with the title “Animals”, and then uses the ALL operator to construct its text. The ALL operator extracts all bindings of its variable child. Its first child is the variable, the second child a textual string that is used to delimit the values of the variable. The resulting text in the example will thus be a comma separated list of all the titles of wiki pages which were tagged “Animal”.

The SOME operator works very similar, only that it contains an additional child in which a number can be entered, and it will not return all but only that number of values.

Instead of the Variable box, You can also use a Binding box to restrict the variable binding further. A Binding box contains a Variable child and a Value child, and will only bind values to the variable if they fit with the Value child. For example, if a Binding with the value “gray” would be used in the example instead of the Variable, the resulting wiki page would only list the title of those pages with a tag “Animal” where the title also contains the word “gray”, for example “Gray Mouse”, or “Gray Wolf”.

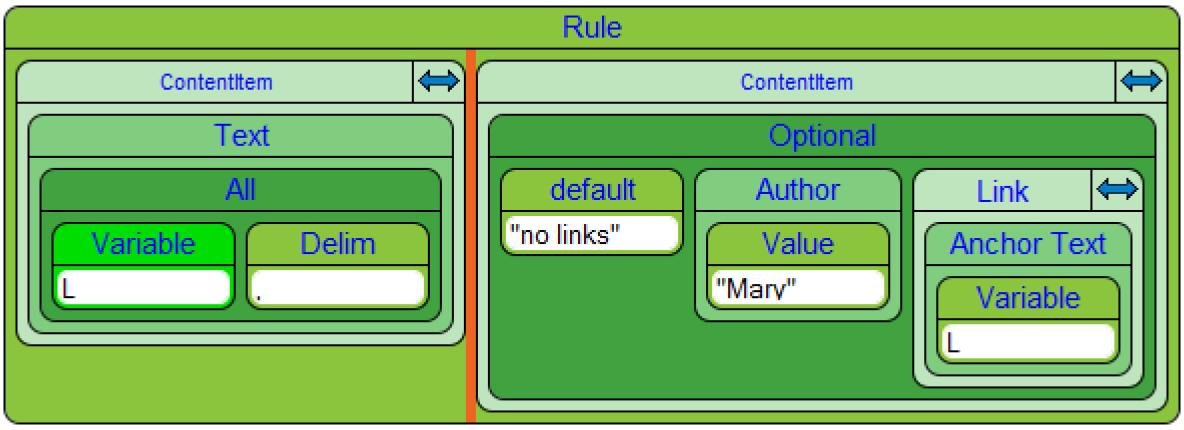


Figure 12: A visKWQL rule with the OPTIONAL operator

The last visKWQL operator is the OPTIONAL, which can only occur in the query part of a rule.

Figure 12 shows a rule that uses it. The first child of the OPTIONAL operator is a default value, the second child is the non-optional part, and the third child the optional part of the query. The query searches all wiki pages written by Mary. If a found page contains a link, the link’s anchor text is assigned to the variable L. If it contains no link, the string “no links” is assigned to L. The construct part simply creates a new wiki page with a comma-separated list of the anchor texts of links in Mary’s wiki pages, or the text “no links” if there are no links on her pages.