jump-sqi

Simple query interface for JUMP

License

Revision History

Revision 0.1.1 (15 jan 2004)

Abstract

This document presents jump-sqi, a small query editor, its main capabilities and how to use it.

Table of Contents

1. Introduction

1.1. What is jump-sqi ?

Jump-sqi is a small query editor made for JUMP workbench. A simple and user-friendly GUI make it possible to query the current task data layers and to return the result as a new layer, or as a feature selection. It allows :

- Spatial queries
- Attribute queries

1.2. What can I do with jump-sqi ?

1.2.1. Context

The GUI environment has the following capabilities :

- record / reload a query
- comboboxes to help query edition
- attribute filters to select the query-type (geometric, numeric or string queries)
- a check-box for case-sensitivity

1.2.2. Queries

Jump-sqi query system allows :

- to query one or several layers at a time
- to perform queries on any type of attribute but Object type
- to perform spatial queries (intersects, includes, is included, touches,...)
- to apply string functions to string attributes (substring extraction, trim,...)
- to apply geometric function to features (ex. buffer, centroid, length, area...)
- to use parameters with some functions as substring or distance
- to use regular expression on strings (for match and contains operators)

1.2.3. Results

Three kinds of results can be asked through the GUI :

- queried features are selected in the layer view
- an attribute table containing the result is opened
- a new layer containing the result is created

1.3. Jump-sqi limitations

Jump-sqi limitations are :

- queried features are wholy selected whith all their attributes (no way to define a target schema different from the queried layer one)
- It is not possible to define several conditions with "and" or "or" (for the "and" case, just query the result of the previous query).
- no way to make joins
- no SQL-type editor

1.4. Installation

1.4.1. Plugin Installation for using jump-sqi with jump

You just have to copy the `jump-sqi.jar` and the `buoy.jar` files in the directory dedicated to plugins (default is `ext`).

1.4.2. Development

To improve jump-sqi or to adapt it to your own needs, use the source code included in the jump-sqi.jar also including the binaries. To compile it, you'll need jump library, jts library, and buoy library.

1.4.3. Internationalization (I18N)

Located at the jar file root, you'll find the `QueryDialog_i18n_ll_CC.properties` file containing the GUI strings. If the file for your language is not there, copy the default `QueryDialog_i18n.properties` file, rename it with your language and country extensions (ex. _de_DE) and replace english strings by your language ones.

2. Query interface configuration

The upper half of the query pannel has some options to configure the query environment.

2.1. The query manager

The query manager contains two buttons, Open and Save as..., allowing to open an existing query or to save the current one.

While opening a query file, an error may occur int these cases :

- the layer defined by the query does not exist in the current task
- the attribute used by the query does not exist in the queried layer

Queries are recorded in a properties file (text file with key/value pairs). Attention, it may be difficult to handwrite or to modify such a file by hand, because special characters have to follow the specifications of java properties file.

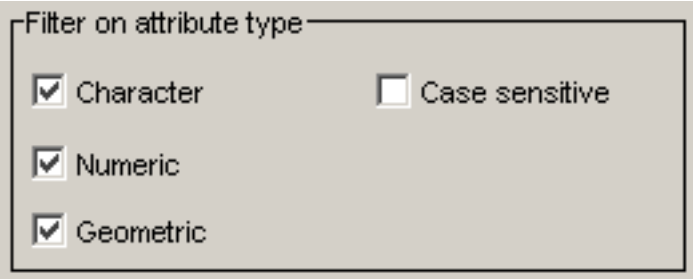
2.2. Filtering attributes

The second sub-menu make it possible to filter visible attributes, using one or several criteria among the following :

- character-based attributes
- numeric attributes
- geometric attributes

Only the attributes of the selected types are proposed in the query editor.

Moreover, selecting the character type make another checkbox visible to define if character-based queries must be case-sensitive or case-insensitive.

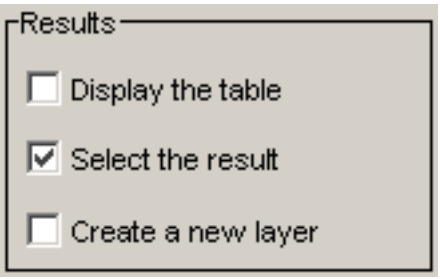


In the here above case, all attributes related to the choosen layer are proposed in the query editor, and string comparisons are case-insensitive.

Querying a layer to find features which name is "New-York" will also find new-york, NEW-YORK or New-york...

2.3. Presentation of the result

A query result may be defined by the following options.



1. Display the table : the query result will appear as an attribute table with all the objects of the defined the layer(s) and in accordance with the defined condition.

2. Select : queried objects are selected in the layer view. They may belong to several layers. As the query editor make it possible to use the selection as queried collection or as the target collection (geometric queries), this query mode make it easy to run through several queries very efficiently.

3. Create a new layer : the result is copied in a new layer or in several new layers if the query is performed on several layers at a time. The name of the new layer is the name of the original layer followed by an underscore and the value located in the last combo box of the query editor.

2.4. The query editor

The query editor contains five fileds :

2.4.1. The "layer" field

This field is used to select the feature collection or feature collections to query on. It may be :

- All the layers : the query will scan all the features from all the layers. Layers where the query are meaningless are ignored (ex. the attribute defined in the query is absent from this layer).

- Selection : queried features are those selected in the active layer view. They may be-

long to several layers. These features may also result from the previous query, run with the checkbox select set to true.

- Selected layers : query only the selected layers of the layer manager.
- USER_LAYER : after these 3 special sources, comes the enumeration of all the user layers which may be selected individually.

Warning

Every time a new layer is selected, the list of available attributes is updated, along with available functions and operators. In the same way, when an attribute or a function is changed, the following comboboxes are updated. It is the reason why it is recommended to define the query parameters from left to right.

2.4.2. Attribute field

In this combo box, you'll find all the available attributes. The list content depends on the choosen layer(s), along with the options checked in the "Filter on attribute type" sub-menu.

Attribute names are immediately followed by their geometric types. At the first place comes the "Geometry" attribute, other attributes appear following the alphbetical order.

An attribute selection determine the list of available functions (the next combo box). For character-based attributes, a list of possible values is computed in the last combo box (the value combo box). This list may be complete (as for enumerated attribute types) or not complete (when there are many possible values, the program stop after the 12th value).

2.4.3. Function field

The choosen attribute may be transformed with a function before the final condition is evaluated. The available list of function depends on the type of the selected attribute.

2.4.3.1. Geometric functions

Functions you can use if you choosed the Geometry attribute to build your condition are :

- : no value means you want to use the feature geometry
- **length** : return the length of the feature geometry. The test condition will use this length
- **area** : return the geometry area. The test condition will use this area
- **number of points** : return the geometry total number of points
- **number of parts** : return one or the number of parts for geometry collections
- **buffer** : return the buffer geometry computed from the feature geometry. This function needs an argument (buffer offset). To make it possible to change the argument, this combo box value is editable. Take care to follow the syntax given as an example
- **centroid** : return a point located inside the geometry area (or on the geometry for linestrings or points)
- **empty** : return "true" if this is an empty geometry, and "false" otherway
- **simple** : return "true" if the geometry is "simple", and "false" otherway
- **valid** : return "true" if the geometry is valid and "false" otherway

The last three functions refer to the "Simple Feature Specification" document from OGC and to the JTS API documentation.

Geometric functions may return a geometry, a numeric value or a boolean (true/false)

2.4.3.2. Numeric function

No numeric function is defined.

2.4.3.3. Text functions

- : the text is let as it is (no modification)
- **trim** : starting white spaces and ending white spaces are removed. This function is very useful to compare shapefiles attributes where texts attributes are filled in with white spaces when their length is under the maximum attribute length.
- **substring** : to extract a string from an attribute value. This function may take two arguments (substring option the text field is editable). Follow the example to edit your query : first argument is the 0 based index of the first letter to extract and second argument is the index of the letter following the string to extract. For example, "New-York" substring(0,3) will return "New"
- **length** : return the string length

Text functions are applied to string, enumeration or character attributes, and may return a string or a numeric value.

> **Tip**
>
> In the next section, you'll find a very powerful mean to do string queries using regular expressions. You'll learn how to use the match operator or the find operator. The regular expression to match or to find has then to be defined in the value combo box (see § 2.4.4.3 et § 2.4.5.4)

2.4.4. Operator field

Available operator list depends on the attribute type or on the value type returned by the function if this type is different from the attribute type. So, operators may be divided up by type :

2.4.4.1. Geometric operators

- **intersectes** : select every feature intersecting any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer).
- **contains** : select every feature containing any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer).
- **is included** : select every feature included in any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer)..
- **within distance** : this is an operator using an argument (the text area is editable so as to input a distance argument using the syntax shown as an example). Select every feature located within [distance] from any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer)..

- **touches** : select every feature touching any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer)..
- **cross** : select every feature crossing any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer)..
- **overlap** : select every feature overlapping any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer).
- **is disjoint** : select every feature being disjoint from any feature from the collection defined in the value combo box (selected faetures, all the layers, a choosen layer)..

> ## Warning
>
> with the disjoint operator, if a feature is disjoint from only one feature of the target collection (and touches or overlaps all the other features) it will be selected in the query. This is due to the fact that the query engine works on a per feature basis. To have a more intuitive way to use disjoint, you may have to union geometries first.

2.4.4.2. Numerical operators

Numerical operators areLes opérateurs numériques sont :

- **=** : equals
- # : different
- **<** : lesser than
- **>** : greater than
- **<=** : lesser or equals
- **>=** : greater or equals

Integer, Double and BigDecimal are proceesed in a uniform way (they are first converted into a double before being compared to the reference value).

2.4.4.3. Operators comparing character strings

Operators to compare strings are :

- **equals** : test if the string attribute (or the attribute transformed into a string) is strictly equal to the value entered in the value field (see hereafter). If the checkbox Case sensitive is checked (see 2.2), the comparison will be case sensitive.
- **different** : check if string attribute is different from the reference value
- **starts with** : tests if first letters of the attribute (or the attribute transformed by the function) are the same as the string in the value field.
- **ends with** : tests if last letters of the attribute (or the attribute transformed by the function) are the same as the string in the value field. It may be a good idea to use the trim function on the attribute before applying this request.
- **matches** : it's a very powerful operator making it possible to compare the attribute to a regular expression. If the whole attribute string matches the regular expression, the condition is checked. The operator takes into account the case sensitivity option. You'll

find some basic examples of regular expressions hereafter (§ 2.4.5)

- **finds** : as the previous one, this operator uses regular expressions. Unlike the previous one, the condition is checked if any part of the attribute string matches the regular expression in the value field.
- **before** : find every attribute before the string entered in the value field following the alphabetical order.
- **after** : find every attribute before the string entered in the value field following the alphabetical order.

2.4.5. Reference value

There are different kinds of reference value, according to the nature of the attribute/function/operator :

2.4.5.1. Reference for geometric operators

For geometric operators, the reference is the collection of features defined by one of the following :

- the selection
- any other feature from the layer
- another feature from this task (or any feature of any layer)
- a feature from a particular layer

2.4.5.2. Reference for numerical operator

For numerical operators, enter a number in the value field. Use the . (dot) symbol for decimal numbers.

2.4.5.3. Reference for boolean operators

Geometric functions may return boolean values. In this precise case, the only possible operators are "equals" or "different" and the only possible values are "true" or "false".

2.4.5.4. Reference for character strings operators

For character string operators, the reference is either a string (equals, different, starts with, ends with, before, after) or a regular expression (matches, contains). Neither one or the other needs quotes.

To avois any confusion, when one of the two last operators is choosen, the words `regular expression` appear in the value textfield.

Here are some regular expression samples :

- matches \A\s+\S+\s+\Z : string containing something else thant spaces (\S+), but starting with (\A) or ending with (\Z) one or several (+) spaces (\s)
- matches S(ain)?te[-\s]?Th[eéè]r[eéè]se : this regex finds the name "Sainte-Thérèse" written "Ste Thérèse" or "Sainte Thérèse", with or without hyphen and with different accents.

- finds [\d]+([\.,][\d]+)? : finds every string including an interger or a decimal number, using the decimal separator "." or ",".

2.5. Querying

When your query is defined, click on Valid to execute it.

During query execution, you can see :

- The name of the layer being processed and a progress bar indicating the number of features already processed.
- A message located just under the query editor showing that the query is being processed

When the querying process has ended, the progress bar is reset to 0, and the message displays the number of features found.

At the bottom of the GUI, you'll find a Refresh button you will have to activate if your data model has been updated (if you added or removed layers) since the last query.

A. Historique des changements

- **Version 0.1.1** (15 jan 2005)
  - now, the query editor is always in front of the JUMP windows
  - combo boxes are no more re-initialized each time one selected value changed
- **Version 0.1.0** (04 dec 2004)
  - Version initiale