# Filter Language

2025-07-14

# Contents



The Biovista Reference Filter Language documents the Biovista's query language.

The documentation can be found also in PDF printable form here.

# Chapter 1

# Biovista Reference Filter Language (BRFL)

## 1.1 Introduction

The filter language augments the Biovista search engine by exchanging speed for flexibility. Without the filter language many queries would be impossible or they would require to generate a custom database.

Here is an example of such a case. We would like to get all references of CCA1 (RE) where the word 'circadian' is mentioned but the word flowering' is not. The query representation in Biovista References Filter Language (BRFL) is:

```
Text matches 'circadian' and Text !matches 'flowering'
```

Furthermore the language provides logical operators such as AND, OR and the necessary grouping and precedence with parenthesis, thus making possible to perform unique and innovative questions that weren't possible before.

## 1.2 Language specification

A BRFL query consists of a domain, a search operator and the input to match. Multiple queries can be combined with boolean logic thanks to logical operators and parenthesis.

The available domains are:

- Entity
- Text
- Claim
- Journal

- Year
- PMID
- CT:Phase
- CT:Type
- CT:Enrollment
- CT:Results

The available search operators are:

- Equals (=)
- Not Equals (!=)
- Matches (matches)
- Not Matches (!matches)
- Contains (contains)
- Not Contains (!contains)
- Starts with (startsWith)
- Ends with (endsWith)
- Less or Equal (<=)
- Greater or Equal (>=)

**Note:** Not all search operators are applicable to all domains.

BRFL also sports a vectorized query form in order to give to the compiler hints for more aggressive optimizations. The vectorized form is denoted with parenthesis and the elements are separated with commas.

### 1.2.1 Entity

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Matches (matches)
- Not Matches (!matches)

The operator **matches** is matching the entities based on the same algorithm used by the disambiguation panel.

In the vectorized form operators = and **matches** are filtering based on the union of the provided entities (any i.e or). When the negation operator is used the papers are matched against all entities (any i.e or) so a paper is filtered when it contains any of the entities mentioned in the vector. Note that in case the argument to the search operator is a vectorized list proper quoting should be used.

#### 1.2.1.1 Examples

How to keep all papers mentioning the entity cca1

```
Entity = cca1
```

How to filter out all papers mentioning the entity cca1

```
Entity != cca1
```

How to keep all papers mentioning the entity cca1 or lhy

```
Entity = ('cca1', 'lhy')
```

Slower equivalent of the above is

```
Entity = cca1 or Entity = lhy
```

How to filter out all papers mentioning the entity cca1 or lhy

```
Entity != ('cca1', 'lhy')
```

How to filter out all papers mentioning the entity cca1 and lhy

```
Entity != cca1 and Entity != lhy
```

In order to disambiguate entities per entity type it is possible to prefix the entity with its entity entity type like

```
Entity = 'Gene|cca1'
```

Note that if you use the entity type qualification the argument must be properly quoted.

### 1.2.2 Text

Available search operators are:

- Matches (matches)
- Not Matches (!matches)
- Contains (contains)
- Not Contains (!contains)

The **matches** operator is using the database's indexed search operation and so it matches whole words. The **contains** operator allows substring matching at lower speeds. The filter allows vectorized queries where the arguments are combined with the **or** operator. The vectorized form should be avoided when negation is used because the semantics of the operation are unlikely to match user's wish.

#### 1.2.2.1 Examples

How to keep all papers mentioning the word photophobic

```
Text matches 'photophobic'
```

### 1.2.3 Journal

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Contains (contains)
- Not Contains (!contains)
- Starts with (startsWith)
- Ends with (endsWith)

In this filter the matching is case insensitive. In the vectorized form operators EQUAlS or CONTAINS are filtering based on the union of the provided entities (ANY i.e OR). When the negation operator is used each paper's journal is matched against all terms (ANY i.e OR) so a paper is filtered when its journal is one (any) of the journals mentioned in the vector.

### 1.2.4 Year

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Less or Equal (<=)
- Greater or Equal (>=)

#### 1.2.4.1 Examples

How to keep all papers after 2004

```
Year >= 2004
```

### 1.2.5 PMID

Available search operators are:

- Not Equals (!=)

The PMID accepts simple strings, quoted strings (single or double), and quoted lists.

#### 1.2.5.1 Examples

How to exclude one specific paper

```
PMID != 1203434
```

How to exclude specific papers

```
PMID != ('1203434', '1203434')
```

### 1.2.6 CT:Phase

This filter is available only for ClinicalTrials.gov corpora.

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Contains (contains)
- Not Contains (!contains)
- Starts with (startsWith)
- Ends with (endsWith)

In this filter the matching is case insensitive. In the vectorized form operators EQUAlS or CONTAINS are filtering based on the union of the provided entities (ANY i.e OR). When the negation operator is used each clinical trials's phase is matched against all terms (ANY i.e OR) so a clinical trial is filtered when its phase is one (any) of the phases mentioned in the vector.

Available phases are:

- NA
- " (empty)
- PHASE2
- PHASE1
- PHASE3
- PHASE4
- PHASE1/PHASE2
- PHASE2/PHASE3
- EARLY_PHASE1

### 1.2.7 CT:Type

This filter is available only for ClinicalTrials.gov corpora.

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Contains (contains)
- Not Contains (!contains)
- Starts with (startsWith)
- Ends with (endsWith)

In this filter the matching is case insensitive. In the vectorized form operators EQUAlS or CONTAINS are filtering based on the union of the provided entities (ANY i.e OR). When the negation operator is used each clinical trials's type is matched against all terms (ANY i.e OR) so a clinical trial is filtered when its type is one (any) of the types mentioned in the vector.

Available types are:

- INTERVENTIONAL
- OBSERVATIONAL

- EXPANDED_ACCESS
- ” (empty)

### 1.2.8 CT:Enrollment

This filter is available only for ClinicalTrials.gov corpora.

Available search operators are:

- Equals (=)
- Not Equals (!=)
- Less or Equal (<=)
- Greater or Equal (>=)

#### 1.2.8.1 Examples

How to keep all clinical trials with en enrollment >= 100

```
CT:Enrollment >= 100
```

### 1.2.9 CT:Results

This filter is available only for ClinicalTrials.gov corpora.

Available search operators are:

- Equals (=)
- Not Equals (!=)

#### 1.2.9.1 Examples

How to keep all clinical trials with results

```
CT:Results = true
```

### 1.2.10 Quotation

In some cases the arguments contain special characters utilized by the BP mini query language, so the unprotected use of them results to BP search engine syntax errors. To overcome this problem the search argument requires extra protection. Below is the table containing the most frequent illegal character uses and the specific protection measure that should be employed.

| **Special character** | **Search Engine protection** | **Example** |
|---|---|---|
| Space ’ ’ | Enclose the argument in single quotes or double quotes | Entity = ‘cca1 protein’ |

| **Special character** | **Search Engine protection** | **Example** |
|---|---|---|
| Parenthesis ‘()’ | Enclose the argument in single quotes or double quotes | Entity = ‘(6-4)-PHOTOLYASE’ |
| Single quotes “’” | Enclose the argument in double quotes | Entity = “2’,5’-OLIGOADENYLATE SYNTHETASE |
| Double quotes ‘“’ | Enclose the argument in single quotes | Entity = ‘“VENTILATION” PNEUMONITIS’ |
| Single and double quotes | Enclose the argument in single quotes and use the backslash character to protect (escape) the single quotes | Entity = ‘“5\’-RACE system for rapid amplification of cDNA ends”’ |

# Chapter 2

# Filter Console

**Note:** This feature is only available for Java Clients

The filter language console facility offers access to advanced features of the Biovista search engine empowering the user to ask complex questions that are not covered by the standard filter user interface (UI). The main reason the UI may not be enough for all user needs is that is not always possible to specify the precedence of the requested operations.

The user can learn the usage of the console and the syntax of the language either be studying the available help thoroughly or by examining the output of the console after every filter operation. All questions are first formulated in BRFL before actual execution so the user can learn by example. The previous commands are easily recalled by the previous button so the user can hand edit them to their liking adding the missing parenthesis or request for even more advanced queries.

## 2.1 Tips and Tricks

- History of previous commands is accessible through the up and down arrows of the keyboard so it is not required to break for the mouse in order to get a previously executed query.

# Chapter 3

# Questions?

For more information you may visit the Frequently Asked Questions page or you may contact Biovista at support@biovista.com