## Searching in GRIN-Global



Revision Date

December 21, 2022

|  |  |
| --- | --- |
| image2449.png | The Search Engine has evolved in GRIN-Global. This documentation refers to the Search Engine used in server release 1.9.9.2 or higher.  Please send any questions related to [feedback@ars-grin.gov](mailto:feedback@ars-grin.gov). |

The Appendix contains [change notes](#change_notes) pertaining to this document.

Author

Martin Reisinger

TOC

[Search Engine and Search Tool Overview 4](#_Toc122517983)

[Overview 4](#_Toc122517984)

[Search Engine Evolution - Enhancements in Server Release 1.9.9.2 5](#_Toc122517985)

[Speed Improvements 6](#_Toc122517986)

[Public Website 6](#_Toc122517987)

[Advanced Searches 7](#_Toc122517988)

[What Does the Search Engine Search? 8](#_Toc122517989)

[List Search on the Public Websites 9](#_Toc122517990)

[List Search in the Search Tool 10](#_Toc122517991)

[Extension of the List Search 11](#_Toc122517992)

[Search Comments 12](#_Toc122517993)

[Curator Tool – Searching via Dynamic Folders 12](#_Toc122517994)

[Dynamic Folders 12](#_Toc122517995)

[IDs & Lookups 13](#_Toc122517996)

[Wildcards 14](#_Toc122517997)

[Using Quotes 14](#_Toc122517998)

[Reserved Words & Wildcards -- Examples 16](#_Toc122517999)

[Wildcard / Operator / Reserved Words 16](#_Toc122518000)

[IS NULL / IS NOT NULL 17](#_Toc122518001)

[IN / NOT IN 17](#_Toc122518002)

[LIKE 17](#_Toc122518003)

[BETWEEN 17](#_Toc122518004)

[Date Fields 18](#_Toc122518005)

[GETDATE() 18](#_Toc122518006)

[DATEDIFF() 18](#_Toc122518007)

[WHERE 18](#_Toc122518008)

[COUNT(\*) 18](#_Toc122518009)

[Subqueries 19](#_Toc122518010)

[DISTINCT 19](#_Toc122518011)

[LEN function 19](#_Toc122518012)

[EXCEPT 20](#_Toc122518013)

[INTERSECT 21](#_Toc122518014)

[--DUMPSQL 22](#_Toc122518015)

[Full Text Indexing 24](#_Toc122518016)

[Considerations 25](#_Toc122518017)

[Extended SQL Support 27](#_Toc122518018)

[WHERE 27](#_Toc122518019)

[NOT 27](#_Toc122518020)

[BETWEEN 27](#_Toc122518021)

[INTERSECT 27](#_Toc122518022)

[EXCEPT FUNCTION 30](#_Toc122518023)

[LEN function 30](#_Toc122518024)

[DateDiff function to find recent viabilities 30](#_Toc122518025)

[Subqueries 30](#_Toc122518026)

[DISTINCT 31](#_Toc122518027)

[NOT EXISTS 31](#_Toc122518028)

[Displaying the SQL: --DUMPSQL 32](#_Toc122518029)

[Appendix A: Fields used in the GG Searches 33](#_Toc122518030)

[Autofields 33](#_Toc122518031)

[SQL to List the “Autofields” Used in the Search Box 33](#_Toc122518032)

[Full Text Indexing 34](#_Toc122518033)

[SQL to List the Fields Having Full Text Indexes 34](#_Toc122518034)

[Appendix B: SQL Queries on the Public Website 35](#_Toc122518035)

[Overview 35](#_Toc122518036)

[3 Basic Components 35](#_Toc122518037)

[Public Website Searches Using the @ 36](#_Toc122518038)

[Appendix C: Administrator Notes on Sorting Search Results 37](#_Toc122518039)

[Appendix D: Document Change Notes 39](#_Toc122518040)

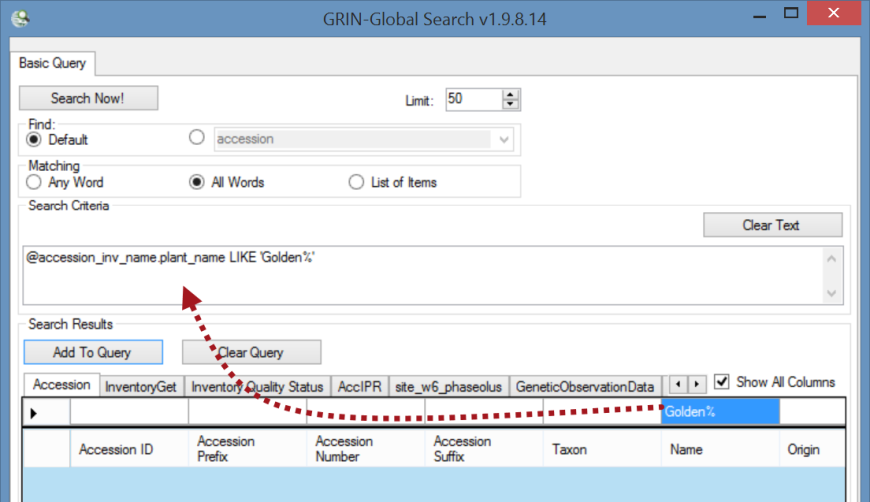
#### Search Engine and Search Tool Overview

##### Overview

The Search Tool and the Public Website use the same customized GG search engine (SE). The GG search engine and Search Tool have both evolved and are continuing to be developed with further enhancements.

SQL Server Full-Text Indexing allows searching for single words in large text fields (such as notes) without specifying wildcards. At the U.S. National Plant Germplasm System (NPGS), Microsoft SQL’s Full-Text Indexing feature was implemented simultaneously with server release 1.9.9.2.

Several notes:

* The Search Tool is a stand-alone program
* Search Tool & Public Website use the same search *engine*, but the PW search capabilities are supplemented by some PW code
* The Search Tool has two distinct modes
  + Text Box
  + Query By Example grid (“QBE”) Recommended ***method***  
    

|  |  |
| --- | --- |
| image2449.png | Think of the search engine as using a “wide net.” At first glance, it may not be obvious why some records are returned by the search. The “odd” results is most likely due to the search finding matches in multiple fields.  In this example, the search string was ‘Van deman’ In the search results, it is obvious why the first and third accessions are listed, but why the second?  Looking at the accession’s details, the Narrative mentions “Van Deman.” On the PW, the Narrative comes from the Accession **Note** field, and in this database, that field was one that the DBA had indexed – hence it was searched. |
|  |  |

##### Search Engine Evolution - Enhancements in Server Release 1.9.9.2

The Search Engine (SE) has evolved, and in server release 1.9.9.2, the functionality has expanded. For example, the search engine now has extended SQL support. This SE fixes many of the issues between the PW and SE regarding visible, active, and available check boxes.

The new version of the SE implemented a completely new way for the PW to filter by these status values. Other changes:

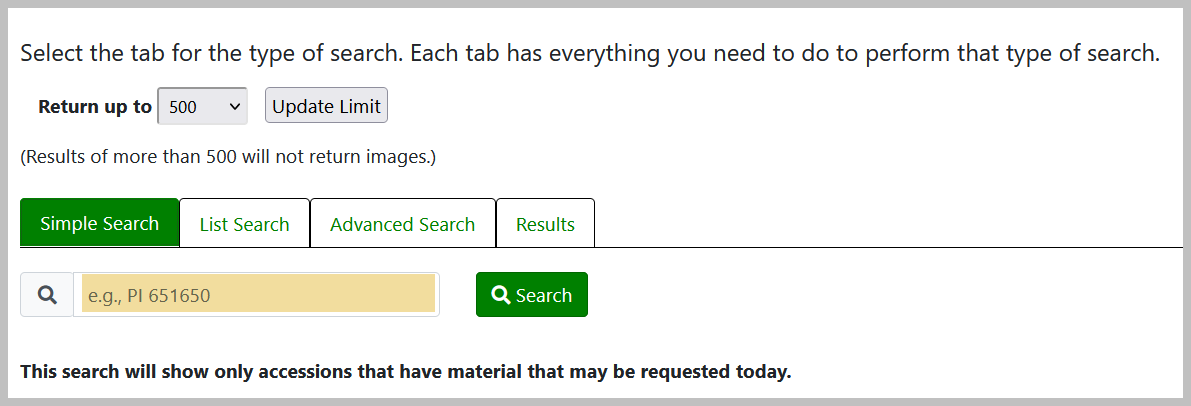
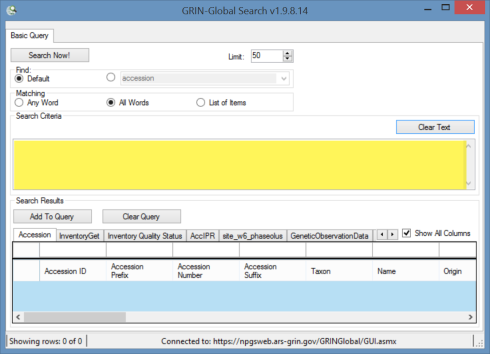
* Speed increases on simple searches
* Full text indexing
* **List of Items** Change
* Changes to Public Website queries
* Extended SQL Support to additional key words: BETWEEN, EXCEPT, UNION, INTERSECT, NOT IN, …

##### Speed Improvements

Checking for web visibility or availability was slowing down simple PW searches such as **PI 500000** because the search’s formatted section might be only @accession.is\_web\_visible = ‘Y’ -- this would return 800,000 results, taking a few seconds to complete, whereas now the SE examines the freeform section first and converts the results into a criteria to combine with the formatted section.

##### Public Website

Basically…  
the **Search For** box on the Public Website is equal to the text box in the Search Tool.

=

The difference is that in some cases the Public Website uses additional logic to handle the **is\_web\_visible** flags and other issues specific to the PW.

There are three levels of sort on the output of the public website searches:

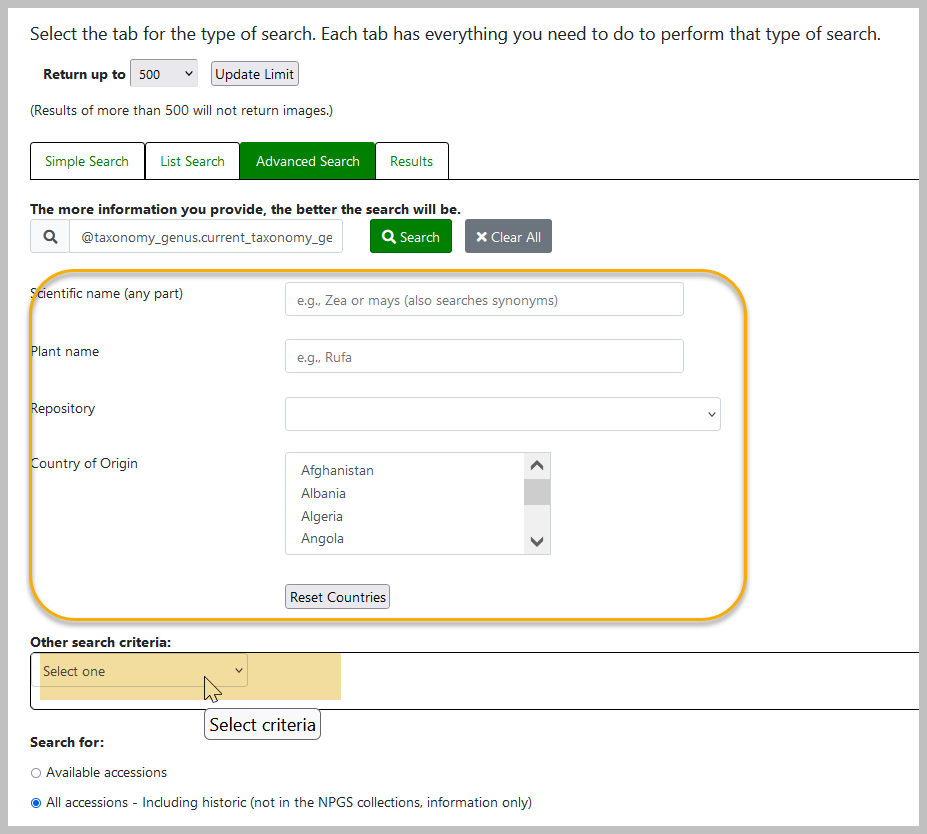
1. The highest weighted field is found first (genus hits before others)
2. Accessions with PI prefixes before Non-PIs\*
3. Most-recently received accessions are found first

Organizations other than NPGS that are running GRIN-Global may set the preferred prefix from “PI” to their organizations preferred prefix. Notes for administrators relevant to this are in Appendix B.

So if there are more than 500 (or whatever your limit is set to) accessions that are genus hits on PI numbers, the most recent of those is first. If there are less than 500 PI records for the genus you are going to see recent non-PI genus hits further down the list and recent PI non-genus hits even further down. That is not all recent accessions will be at the top because the other sorts have a higher precedence.

|  |  |
| --- | --- |
| **image2449.png** | The Public Website has a dual personality. External users, not genebank organizational personnel, use it to search for and order accessions. Internal staff, whose Public Website logins have been associated by the GG Admin to their CT login, have additional features, including the ability to run SQL queries against the database. Refer to the appendix section [*SQL Queries for Searching the Database*](#extended_sql_support) on using the Public Website to search the database using SQL queries. |
| A picture containing light, drawing  Description automatically generated | The search text formatted in the Search Tool text box used by internal users can also be copied and used in the Search box on the Public Website. This may be handy when an external user requests assistance obtaining information from the database that is not available via any Public Website options. An internal staff user can format the query in the Search Tool, send the query text to the external user and explain how to drop the query text in the search box. (See Public Website [search constructs](#search_constructs).) |

##### Advanced Searches

On the **Advanced Search** tab, additional criteria may be included to supplement the text inputted in the search box:   


#### What Does the Search Engine Search?

The search engine (SE) has three main code sections:

1. Formatted
2. Lists
3. Unformatted
4. **Formatted: What it’s told to search for**  
   The user creates formatted searches from QBE with SQL-like syntax starting with the at sign (@)   
   Ex: **@accession.accession\_number\_part1 = 'PI' AND @site.site\_id IN (3)**
5. **Lists: Identifier (ID) lists**   
   The Search List function looks for patterns in the text provided in the listed items. It first determines the number of blocks of text separated by spaces (also known as “tokens”).

|  |  |  |
| --- | --- | --- |
| **Number of Tokens** | **Search Engine Assumes** | **Example** |
| 4 | Inventory identifier | NA 51425 .001 PL |
| 3 | Accession identifier | GMAL 3764 .a |
| 1 (text) | Accession identifier | CZ12345twery |
| 1 (numeric) | Order Request identifier | 345102 |
|  | Plant Names |  |

When there are 4 tokens – the SE assumes the items are inventory items, since the inventory identifier may have up to four items (prefix, number, suffix, and inventory type form). When there are three tokens, it assumes these are the three parts of the accession identifier. The List Search is also programmed to use a single token and look for accessions matching the one text string (some genebanks use only the accession prefix field to contain the entire accession identifier).

1. **Unformatted - Freeform Searches**Enter words (and/or numbers) and the SE tries to find them as best it can
2. It will first search IDs such as PI 500000   
   (using either accession or inventory ID)
3. Each word is checked for an exact match on 22 fields (determined by the DBA using the **sys\_search\_autofield** table) (see [autofields](#autofields))
4. Words are also checked in any existing full-text indexes.   
   The GG DBA can index any text field, usually large fields such as **Note** fields (comments) to meet an organization’s requirements. The GG table **sys.fulltext.indexes** lists these fields. (see [Full-Text Indexing](#fti))

|  |  |
| --- | --- |
| image2449.png | Users can [combine formatted and freeform criteria or append formatted criteria](#list_and_freeform) to the end of a list search. Searches work best when the formatted text is appended after the list of items (since that is where the PW tacks it on). |

##### List Search on the Public Websites

The original GRIN-Global Public Website had a checkbox that need to be selected in order to use the List Search. In the current Public Website, the List search has its own tab.

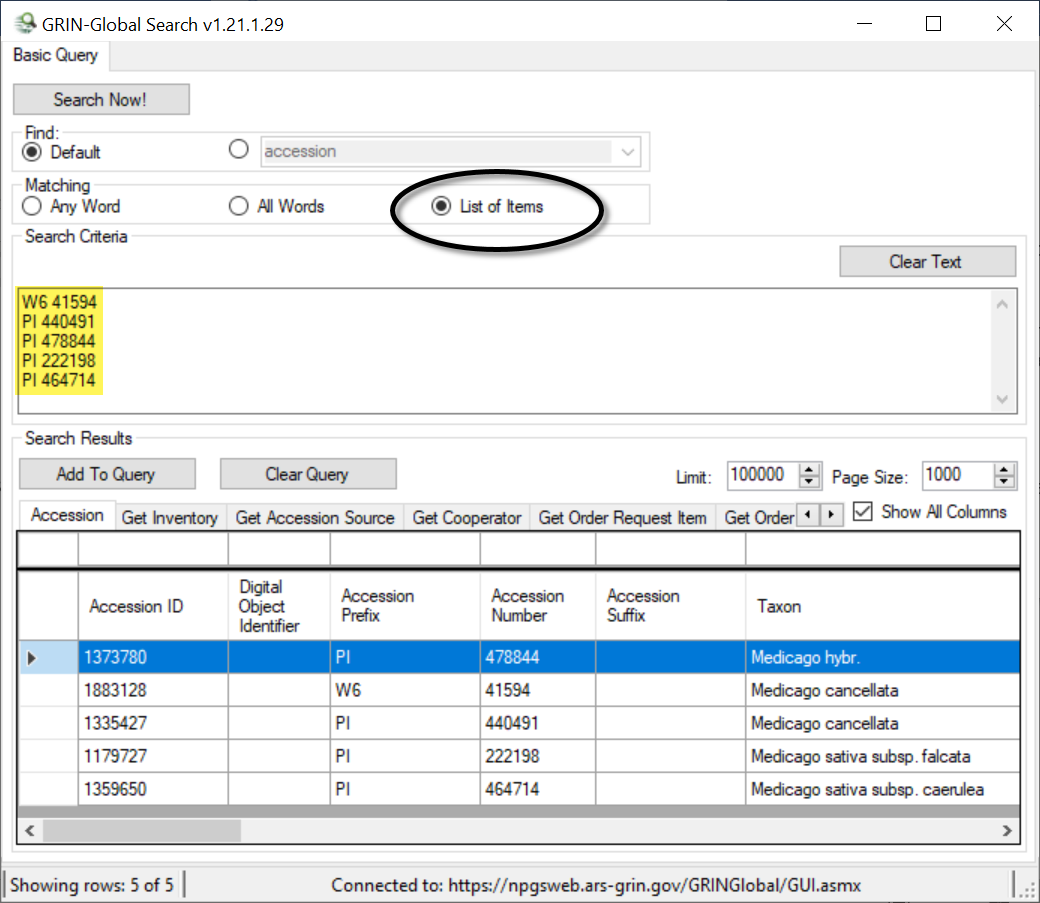
|  |  |
| --- | --- |
| A picture containing light, drawing  Description automatically generated | On the public website, if you enter a valid order ID in the List Search box, the search will return the accessions included in the request. |

###### Original PW

###### The checkbox “**Alternative Search method...**” initiates the List Search Current Public Website List Search

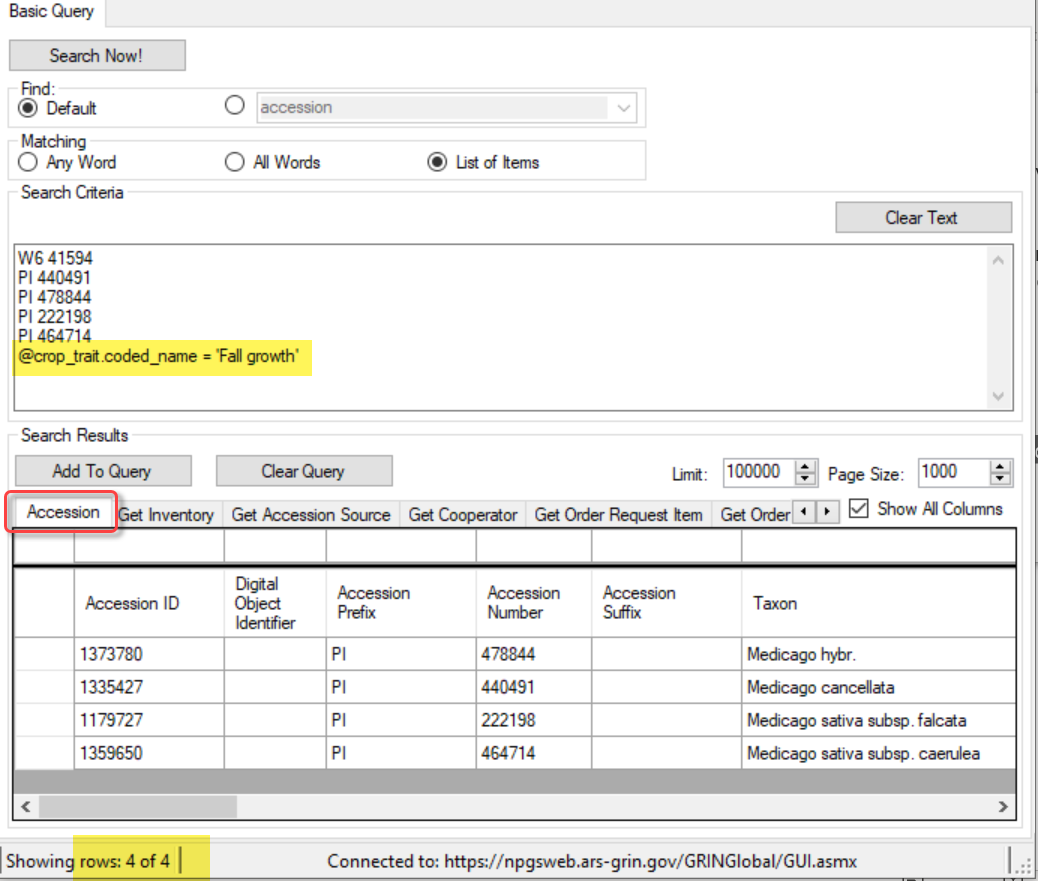
##### List Search in the Search Tool

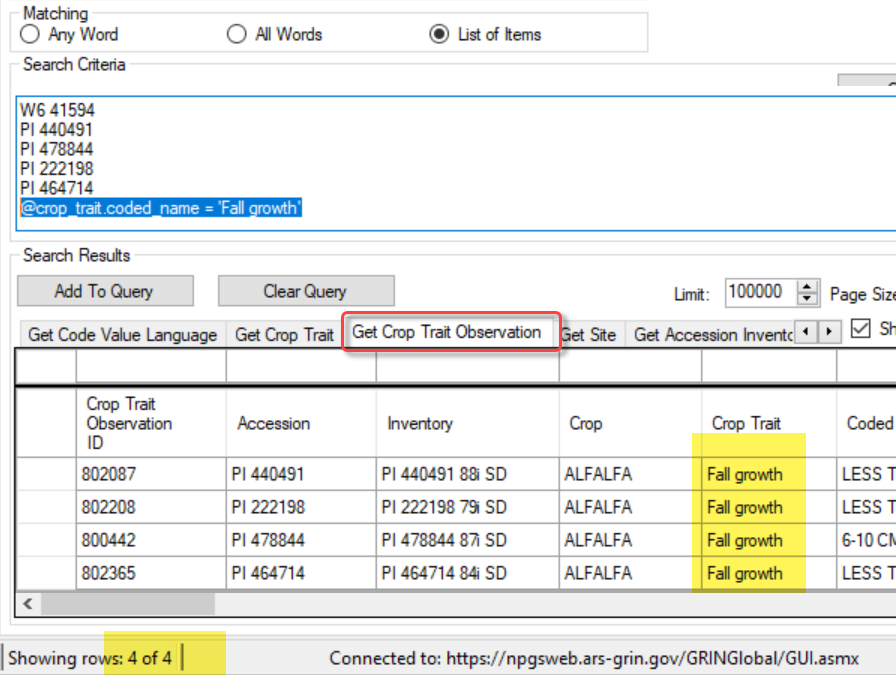
|  |  |
| --- | --- |
| image2449.png | In releases prior to server 1.9.8.2, the search would work with a list in the text box even when this radio button wasn’t selected. Now it must be selected for a list of IDs. |



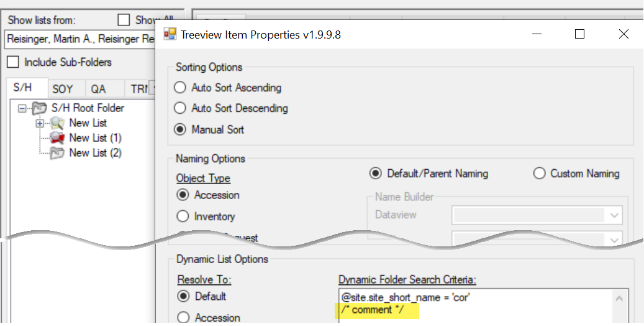
|  |  |
| --- | --- |
| A picture containing light, drawing  Description automatically generated | Remember to switch radio buttons after a List Search. Otherwise, a typical search will fail. |

##### Extension of the List Search

It is possible to append a formatted search string to a list of items. For example, the following example is a valid search:  


In the search without the @crop\_trait.coded\_name = 'Fall growth' statement, five accessions were found, but with it, four. When switching to the **Crop Trait Observation** dataview and *re-running the search*, the reason is more apparent:  


#### Search Comments

When using the Search Tool, you can include comments. This is helpful when copying the search statement to the Curator Tool to build a Dynamic Folder:  
 

* when you use a double dash -- on a line, anything after the double dash is treated as a comment
* to comment multiple lines, start with **/\*** and then end your comment with **\*/**

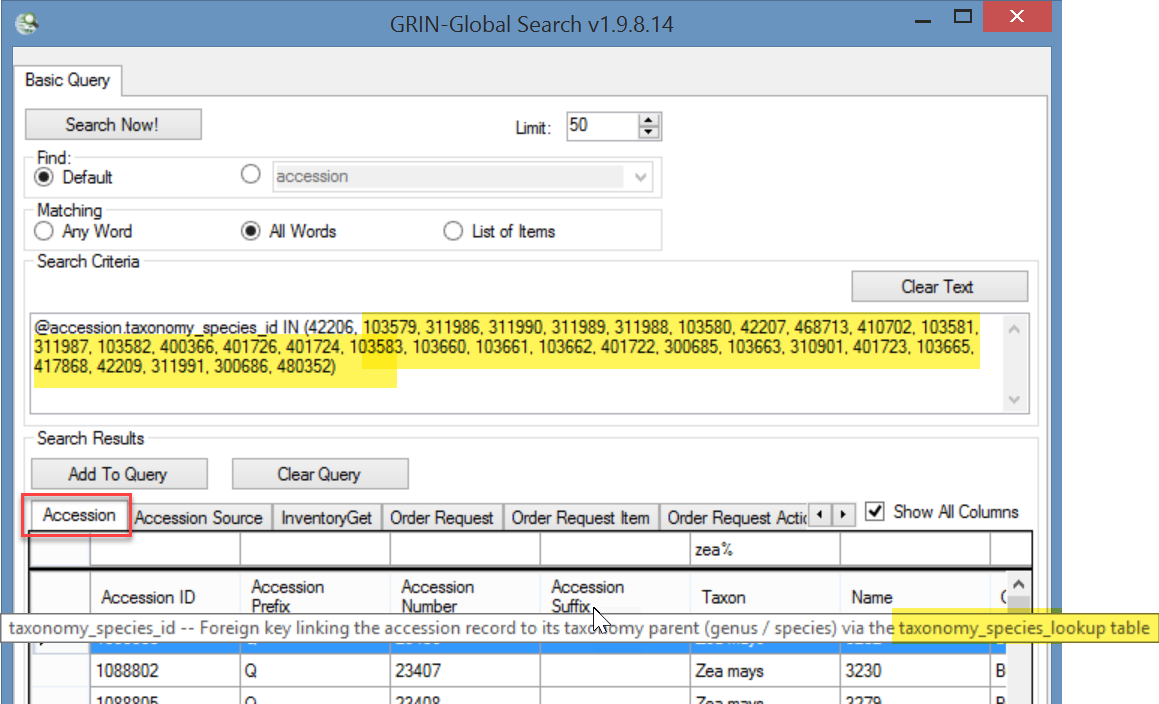
#### Curator Tool – Searching via Dynamic Folders

##### Dynamic Folders

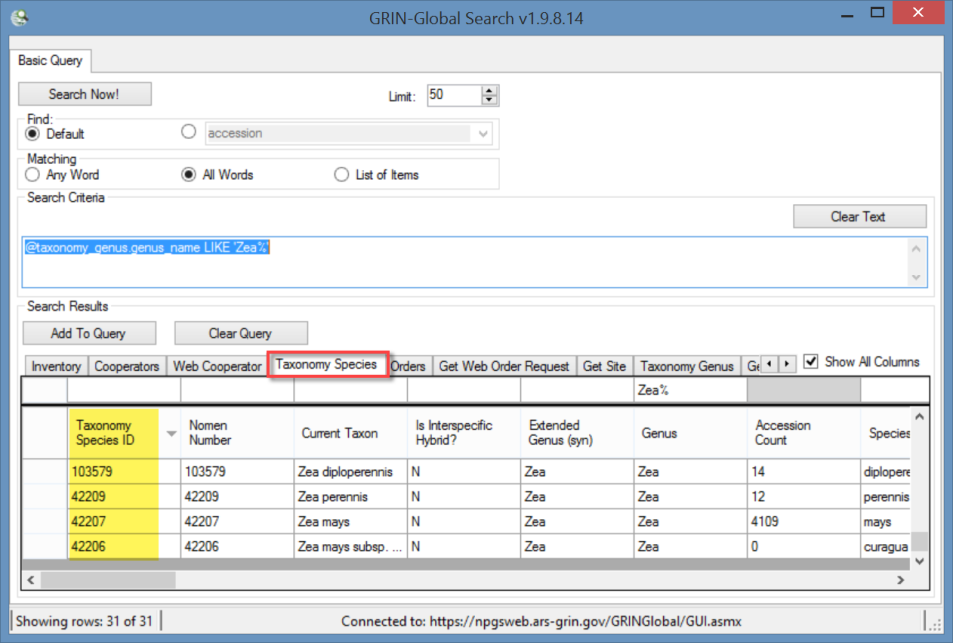
The Curator Tool has two types of folders, static and dynamic. Dynamic folders (also referred to as dynamic queries) are basically stored queries in a CT user’s List Panel that originate from the generated text from a Search Tool query. A big advantage of setting up a dynamic folder is that the folder retains your search criteria; it eliminates the steps of switching back and forth to the search tool and dragging desired records into the Curator Tool.

A complete Dynamic Folders guide is online: <https://www.grin-global.org/docs/gg_dynamic_folders.pdf>

#### IDs & Lookups

When searching on a field that uses LOOKUP IDs, the ID numbers are listed in the search statement:  


In the screen above, there were 31 species records matching “Zea.”

The following screen shows the corresponding 31 species records:  


#### Wildcards

|  |  |
| --- | --- |
| image2449.png | % and \* behave differently in the Public Website. Full text indexing will handle asterisks at the end of the word (\*), whereas if you use a trailing % sign, only the autofield search is used.  Example: **borde%**  vs. **borde** In the QBE, an \* **converts to LIKE ‘%’** |

##### Using Quotes

Using quotes ensures that the full term is searched. Two examples below, with and without quotes - and the number of found records:

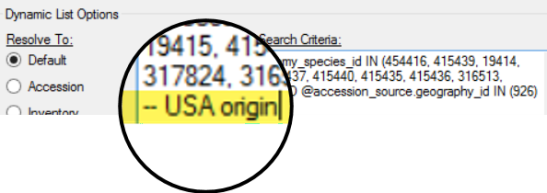
|  |  |  |
| --- | --- | --- |
| **Search string** | **Records Found** | **What the Search Engine is Looking For** |
| ‘yellow rain’ | 0 | the two words **yellow rain** - exactly as entered |
| yellow rain | 66 | either word, **yellow**, or **rain**, in any of the fields that are searched |
| ‘rain’ | 638 | any occurrence of the word **rain** in any of the fields being searched |
| rain | 638 | any occurrence of the word **rain** in any of the fields being searched |

Extended SQL Support

Additional SQL terms can be used now:

* BETWEEN
* WHERE
* EXCEPT
* INTERSECT
* GETDATE()
* DATEDIFF()
* COUNT
* DISTINCT

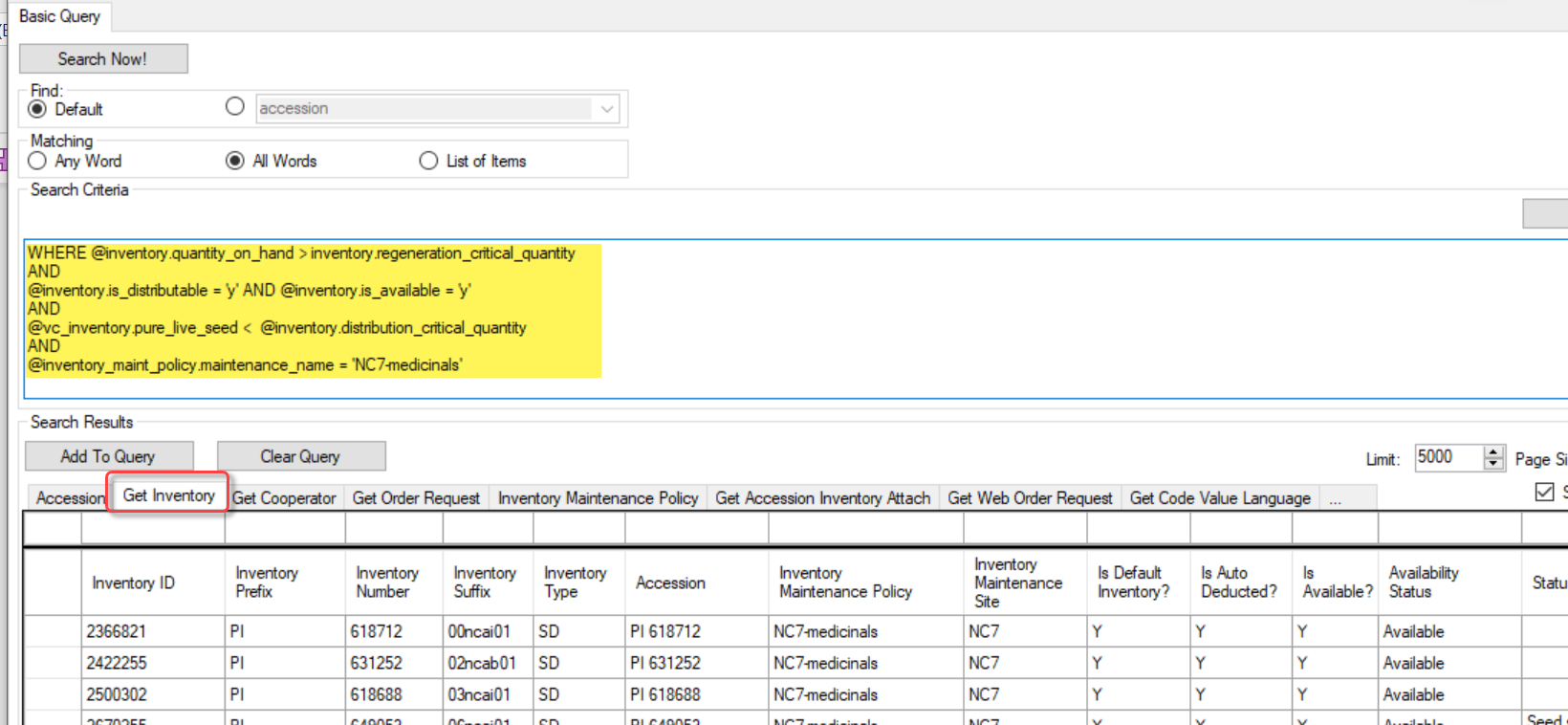
|  |  |
| --- | --- |
| image2449.png | The ST can’t handle an entire SQL select statement, only a statement beginning with a SQL WHERE clause. |

* comments are valid (double dash) --   
  also valid w/ dynamic folders:  
  
* WHERE may be used in the Search Tool:

**WHERE accession.accession\_number\_part2 BETWEEN 500000 AND 500050**

#### Reserved Words & Wildcards -- Examples

| Wildcard / Operator / Reserved Words | **Examples /  Notes** |
| --- | --- |
| %  (percent symbol)  \* (asterisk – not recommended)  In 1.9.6.41 it is recommended to only use the % rather than the \*. (Date searches work with %, but not with \* - this is a known (reported) bug. ) | Use to broaden searches, especially when the exact spelling is unknown. The field must be a text field. Either wildcard (% or \*) allows a match of any string of any length (including zero length)  Examples:  **Rubus%**  **Prunus%var** will locate any Prunus with “var” included;  **%var%** will locate any accessions with the text “var” as part of its taxon  **‘2015%’** |
| \_  (underscore) | The wild card underscore character **\_**  Represents any *single* character. Multiple underscores may be used if needed. The field must be a text field.  **Solanum\_x%** will find:  **Solanum x** doddsii and  **Solanum x** sucrense  If you need to search for the underscore character rather than have it act as wildcard, enclose it in brackets, such as: @inventory\_action.action\_name\_code LIKE 'INS[\_]%'  (in this example, the 4th character must be an underscore character) |
| <> (not equal to) | Can be used to indicate “not equal to.” The field can be either a text or numeric field.   * when the field is a *text* field, the criterion must be enclosed by quotes – single quotes: ‘PI’ or double quotes: “PI” * when the field is a *numeric* field, the criterion *is not* enclosed in quotes   For details, refer to [Searching with <>](#search_w_gt_lt) |
| IS NULL /  IS NOT NULL | NULL values represent missing unknown data. By default, a table column can hold NULL values.  Note: NULL and 0 are not equivalent.   For details, refer to [Searching for NULL or NOT NULL](#null) |
| IN / NOT IN | Used when the criterion field is using a lookup table. (Lookups generate an **IN (…)** clause.) The numbers in the parentheses are the Lookup Key values in the database.  For details, refer to [Searching with IN and NOT IN](#searching_withIN) |
| LIKE | The LIKE operator is used to search for a specified pattern.  Example: **LIKE ‘CAPSICUM%’**  In this case the QBE is saying find any text that begins with “Capsicum.” The trailing percent symbol indicates that any records with any text after “capsicum” should be included if found. |
| BETWEEN | Prior to server version, 1.9.9.2, BETWEEN was *not* a valid operator. When a range of values is needed, construct your criteria using a range.  For example: @order\_request.ordered\_date > ‘2015-01-31’ AND  order\_request.ordered\_date < ‘2015-03-01’  (finds the orders for February, 2015)  Same results, using BETWEEN @order\_request.ordered\_date  BETWEEN '2015-01-31' AND '2015-03-01'  Note: BETWEEN can be used with text as well, such as searching for a range between ‘GBK-0100’ and ‘GBK-0200’ |
| Date Fields | Searching for dates can be tricky because the date field includes the time of day as well. Refer to Date Fields for details.    The following are valid searches:  **@accession.created\_date like '2015%' @accession.created\_date like '2015-09-%'**  **@accession.created\_date like '2015-09-05%'**  **@accession.created\_date like '2015-%-05%'** |
| GETDATE() | Retrieves database current date/time in SQL Server |
| DATEDIFF() | Calculates the difference between two dates |
| WHERE | The ST can’t handle an entire SQL select statement, but it can handle parts a SQL WHERE clause. The Search Engine looks at which fields you use so it knows which table to join when it builds the FROM clause. And the dataview definition specifies which fields get selected.  **WHERE taxonomy\_genus.genus\_name like 'Triticum%'**  **AND NOT EXISTS (SELECT \* FROM accession\_source acs**  **WHERE accession.accession\_id = acs.accession\_id**  **AND acs.source\_type\_code = 'COLLECTED')** |
| COUNT(\*) | A query using COUNT to find rows with many inventories (from one accession)  in the Search Tool or dynamic folder:  **@ taxonomy\_genus.genus\_name = 'Zea'**  **AND (SELECT COUNT(\*) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 32** |
| Subqueries | A subquery is a query within a query – the inner query is resolved first.  Can be used in various ways, such as to search by specific owner  Example:  **@accession.owned\_by IN (SELECT cooperator\_id FROM cooperator WHERE last\_name = 'Millard')**  Example: A query using COUNT to find rows with many inventories (from one accession)  in the Search Tool or dynamic folder:  **@ taxonomy\_genus.genus\_name = 'Zea'**  **AND (SELECT COUNT(\*) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 32** |
| DISTINCT [server >= 1.9.9.2] | The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.  Example: List More than 2 inventory owners  **WHERE taxonomy\_genus.genus\_name = 'Zea'**  **AND (SELECT COUNT(distinct i.owned\_by) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 2** |
| LEN function [server >= 1.9.9.2] | The LEN function determines the string length. This could be used to find long plant names  **WHERE LEN(accession\_inv\_name.plant\_name) > 36** |
| EXCEPT [server >= 1.9.9.2] | Returns any distinct values from the query to the left of the EXCEPT operator that are not also returned from the right query.  The following EXCEPT query is used to track orders not yet completed (**order\_request.completed\_date IS NULL**) when a curator has been alerted (**action\_name\_code = 'CURALERTED')** about an NC7 order **(site\_id = 16)**, but he has not cleared it and the order is still pending (the curator hasn’t cleared the order (**action\_name\_code = 'CURCLEARED').**  **EXCEPT @site.site\_id IN (16) AND @order\_request.completed\_date IS NULL AND @order\_request\_action.action\_name\_code = 'CURALERTED' AND @order\_request\_action.cooperator\_id IN (122186)** |
| INTERSECT [The GG server release must be >= 1.9.9.2] | The INTERSECT operator is used to combine like rows from two queries.  It returns rows that are in common between both results.  For example, using the search tool, find accessions with specific observation values for two different traits. Example: find *kernel color*  **White** and *primary race* **Corn Belt Dent**.  **@crop.name = 'Maize' AND @crop\_trait\_lang.title = 'Primary Race' AND @crop\_trait\_code\_lang.title = 'Corn Belt Dent'**  **INTERSECT**  **@crop.name = 'Maize' AND @crop\_trait\_lang.title = 'KERNEL COLOR' AND @crop\_trait\_code\_lang.title = 'White'**  **INTERSECT**  **@site.site\_id IN (16) AND @inventory.is\_distributable = 'Y' AND @inventory.is\_available = 'Y'**  A similar, but faster version of the query, using the trait IDs:  **@crop\_trait\_observation.crop\_trait\_id = 89001 AND @crop\_trait\_code\_lang.title = 'Corn Belt Dent'**  **INTERSECT**  **@crop\_trait\_observation.crop\_trait\_id = 89027 AND @crop\_trait\_code\_lang.title = 'White'**  **INTERSECT**  **@site.site\_id IN (16) AND @inventory.is\_distributable = 'Y' AND @inventory.is\_available = 'Y'** |
| --DUMPSQL [server >= 1.9.9.2] | With –DUMPSQL, the search engine has an option to deliberately throw an error and show the SQL it generated when the first line of the query is this comment: **--DUMPSQL** (See [DUMPSQL](#dumpsql).) |

**Maintenance Policy**, such as in the following example:  


The code above:

**WHERE @inventory.quantity\_on\_hand > inventory.regeneration\_critical\_quantity**

**AND**

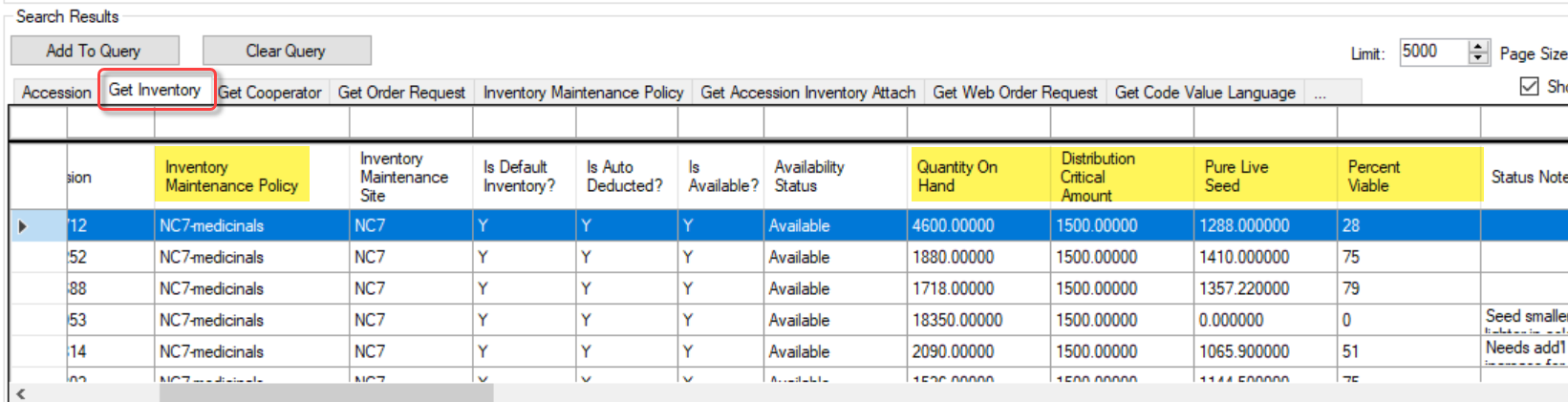
**@inventory.is\_distributable = 'y' AND @inventory.is\_available = 'y'**

**AND**

**@vc\_inventory.pure\_live\_seed < @inventory.distribution\_critical\_quantity**

**AND**

**@inventory\_maint\_policy.maintenance\_name = 'NC7-medicinals'**

In the query above, 13 inventory lots were identified as having quantities of viable seeds that were less than the desired distribution quantities.  


#### Full Text Indexing

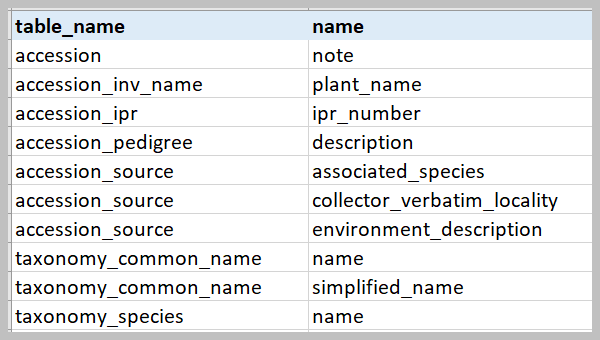
A full text index will have an entry in a generated index for each term or word found in a specified table field. These indexes are established by the genebank’s GG administrator for specific fields in the database; additional fields can be indexed over time. This feature provides significant changes to the Public Website users’ searches.

Administrator’s Note: Full text indexing requires the GG administrator to use SQL Server’s Full Text Indexing methodology. See also [Appendix A](#appendix_fields_searched).

Example:

|  |  |
| --- | --- |
| **Releases pre- 1.9.9.2** | **Release 1.9.9.2** |
| PW: ‘%weedy red rice%’ | PW: weedy red rice |

###### NPGS:

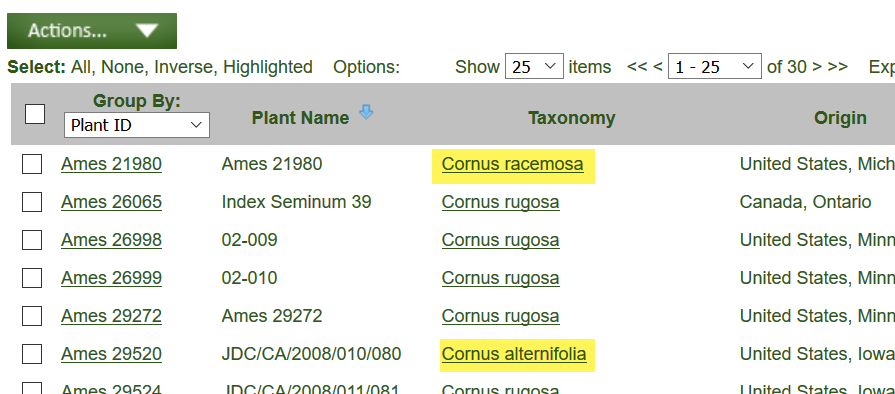
In Release 1.9.9.2, the following fields are now set to full text indexing:   


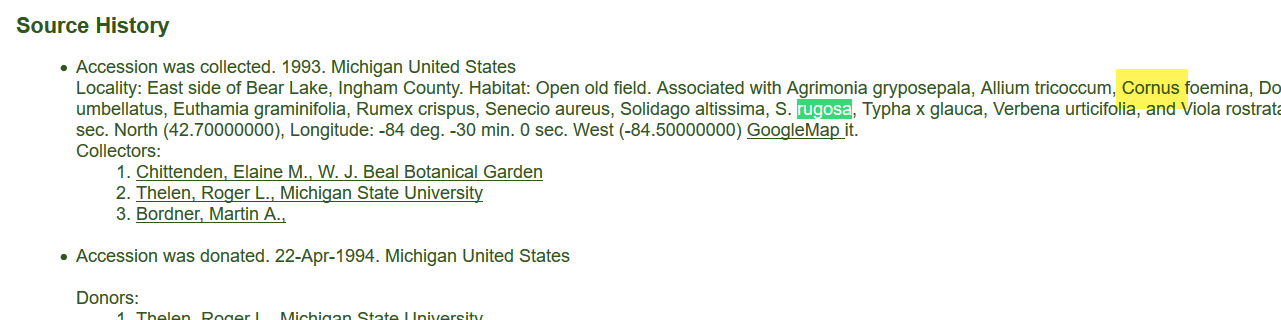
Some stop words (such as "the" and "and") that are both common and typically not meaningful are ignored by the search. ([sample stop words](https://www.textfixer.com/tutorials/common-english-words.txt))

How would you know what fields are indexed? When logged into the Public Website, run the following SQL:

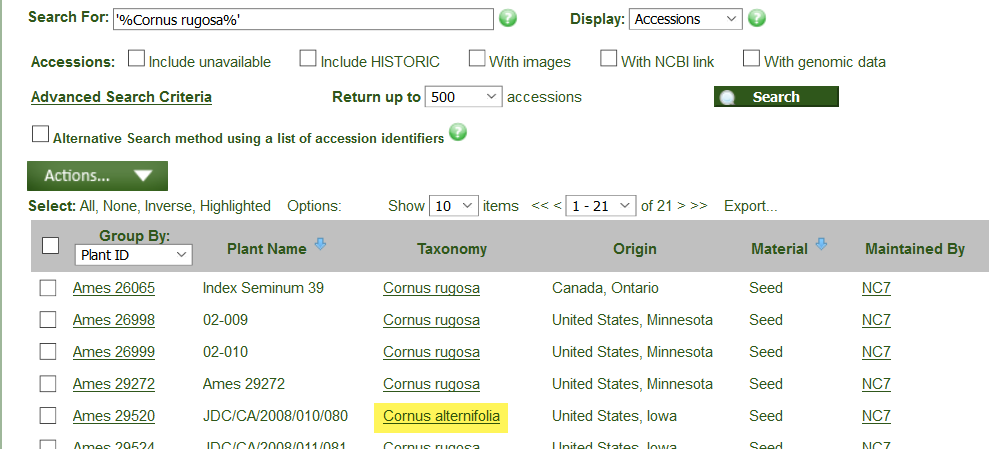
**SELECT DISTINCT  
 object\_name(fic.[object\_id])as table\_name,  
 [name]  
FROM  
 sys.fulltext\_index\_columns fic  
 INNER JOIN sys.columns c  
 ON c.[object\_id] = fic.[object\_id]  
 AND c.[column\_id] = fic.[column\_id]**

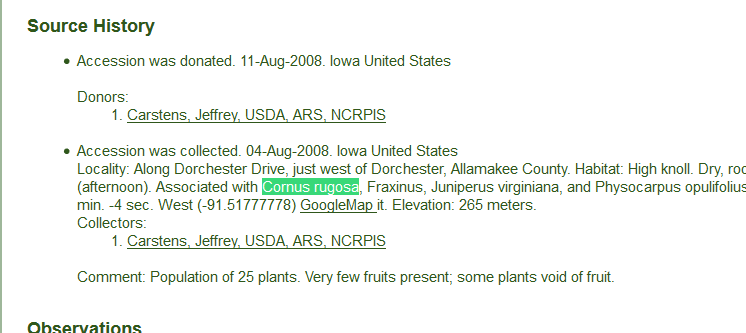
##### Considerations

A Public Website search for **%Cornus rugosa%** may find accessions which at first glance in the list may seem like not a valid match. In this example, the following displays in NPGS’s database:  


Using the Ames 21980 accession, the detail page shows:  


The search is basically asking for *either* word to be found, ***Cornus***, or ***rugosa***. Any words specified between the %...%

When the same string is used, but in quotes – **‘%Cornus rugosa%’** – the list of records will not include that record:  


Using the Ames 29520 accession, the detail page shows:  


Using the quotes ensures that the full term is searched, in this case, ***Cornus rugosa***.

Two examples, with and without quotes - and the number of found records:

|  |  |  |
| --- | --- | --- |
| **Search string** | **Records Found** | **What the Search Engine is Looking For** |
| ‘yellow rain’ | 0 | the two words **yellow rain** - exactly as entered |
| yellow rain | 66 | either word, **yellow**, or **rain**, in any of the fields that are searched |
| ‘rain’ | 638 | any occurrence of the word **rain** in any of the fields being searched |
| rain | 638 | any occurrence of the word **rain** in any of the fields being searched |

#### Extended SQL Support

##### WHERE

SQL WHERE clauses work in the Search Tool. However, since the search engine doesn’t use table aliases, use full table names when constructing statements.

**@taxonomy\_genus.genus\_name LIKE 'Glycine%'** equals

**WHERE taxonomy\_genus.genus\_name LIKE 'Glycine%'**

In the following example, a comment (text preceded with -- ) is also illustrated.

The following code can be used in the Search Tool:

**-- Find accessions owned by Esther which are active, but not available**

**WHERE accession.owned\_by=107186**

**AND accession.status\_code = 'ACTIVE'**

**AND NOT EXISTS (SELECT \* FROM inventory WHERE accession\_id = accession.accession\_id**

**AND is\_distributable = 'Y' AND is\_available = 'Y' and owned\_by=107186)**

##### NOT

…now allowed in freeform queries:

Ex: **Bahamas AND NOT gossypium**Ex: **Malus NOT (KAZ or Canada or USA or GBR)**

##### BETWEEN

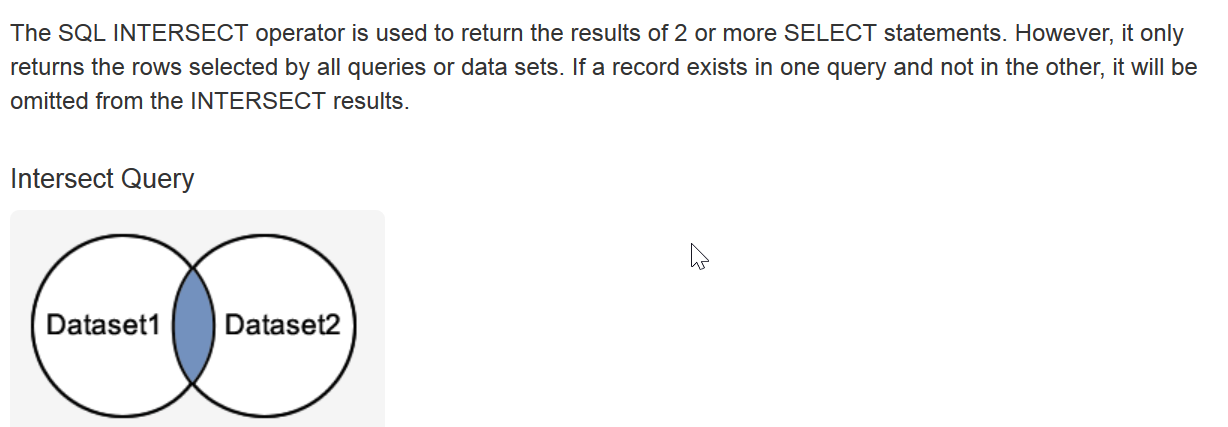
**@accession.accession\_number\_part2 BETWEEN 500000 AND 500050**

**@order\_request.ordered\_date BETWEEN '2015-01-31' AND '2015-03-01'**

##### INTERSECT

https://www.techonthenet.com/sql/intersect.php

The INTERSECT operator is used to combine like rows from two queries.  It returns rows that are in common between both results.



For example, using the search tool, find accessions with specific observation values for two different traits. Example: find *kernel color*  **White** and *primary race* **Corn Belt Dent**.

**@crop.name = 'Maize' AND @crop\_trait\_lang.title = 'Primary Race' AND @crop\_trait\_code\_lang.title = 'Corn Belt Dent'**

**INTERSECT**

**@crop.name = 'Maize' AND @crop\_trait\_lang.title = 'KERNEL COLOR' AND @crop\_trait\_code\_lang.title = 'White'**

**INTERSECT**

**@site.site\_id IN (16) AND @inventory.is\_distributable = 'Y' AND @inventory.is\_available = 'Y'**

A similar, but faster version of the query, using the trait IDs:

**@crop\_trait\_observation.crop\_trait\_id = 89001 AND @crop\_trait\_code\_lang.title = 'Corn Belt Dent'**

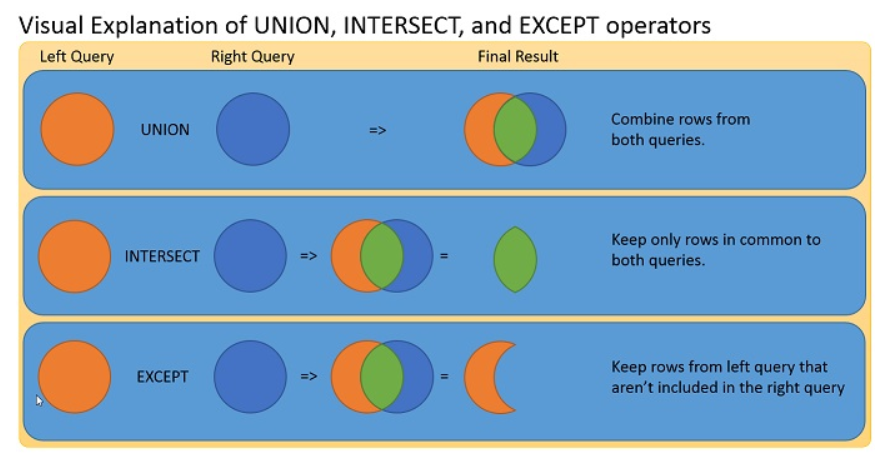
**INTERSECT**

**@crop\_trait\_observation.crop\_trait\_id = 89027 AND @crop\_trait\_code\_lang.title = 'White'**

**INTERSECT**

**@site.site\_id IN (16) AND @inventory.is\_distributable = 'Y' AND @inventory.is\_available = 'Y'**

Besides INTERSECT, UNION and EXCEPT can be used to fine tune searches.



INTERSECT Example: Looking for aronia accessions that have an available inventory and have an inventory with an image attached, available or not. That requires fancier SQL, such as an INTERSECT.

This search produces incorrect results:

**@taxonomy\_genus.genus\_name = 'aronia'**

**and @accession.status\_code = 'ACTIVE'**

**and @accession.is\_web\_visible = 'Y'**

**AND @inventory.is\_distributable =  'Y'**

**AND @inventory.is\_available = 'Y'**

**AND @accession\_inv\_attach.category\_code = 'IMAGE'**

With INTERSECT, the search produces correct results:

@taxonomy\_genus.genus\_name = 'aronia'

and @accession.status\_code = 'ACTIVE'

and @accession.is\_web\_visible = 'Y'

INTERSECT @inventory.is\_distributable =  'Y' AND @inventory.is\_available = 'Y'

INTERSECT @accession\_inv\_attach.category\_code = 'IMAGE'

##### EXCEPT FUNCTION

Returns any distinct values from the query to the left of the EXCEPT operator that are not also returned from the right query.

The following EXCEPT query is used to track the orders when a curator has been alerted (**action\_name\_code = 'CURALERTED')** about an NC7 order **(site\_id = 16)**, but he has not cleared it and the order is still pending (the curator hasn’t cleared the order (**action\_name\_code = 'CURCLEARED').**

**@site.site\_id IN (16) AND @order\_request.completed\_date IS NULL AND @order\_request\_action.action\_name\_code = 'CURALERTED' AND @order\_request\_action.cooperator\_id IN (122186)**

**EXCEPT**

**@order\_request\_action.action\_name\_code = 'CURCLEARED' AND @order\_request\_action.cooperator\_id IN (122186)**

**-- Millard is 122186**

##### LEN function

The LEN function determines the string length. This could be used to find long plant names

**WHERE LEN(accession\_inv\_name.plant\_name) > 36**

##### DateDiff function to find recent viabilities

**WHERE datediff(day, inventory\_viability.tested\_date, getdate() ) < 180**

**@inventory\_viability.inventory\_viability\_id LIKE '%'**

**AND datediff(day, inventory\_viability.tested\_date, getdate() ) < 180**

##### Subqueries

A subquery is a query within a query – the inner query is resolved first.

Can be used in various ways, such as to search by specific owner   
Ex: **@accession.owned\_by IN (SELECT cooperator\_id FROM cooperator WHERE last\_name = 'Millard')**

Ex2: A nested subquery for site name:

**@accession.owned\_by IN**

**(SELECT cooperator\_id FROM cooperator WHERE site\_id =**

**(SELECT site\_id FROM site WHERE site\_short\_name = 'NC7'))**

Ex3: A query using COUNT to find rows with many inventories (from one accession)

in the Search Tool or dynamic folder:

**@ taxonomy\_genus.genus\_name = 'Zea'**

**AND (SELECT COUNT(\*) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 32**

in a SQL query:

**WHERE taxonomy\_genus.genus\_name = 'Zea'**

**AND (SELECT COUNT(\*) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 32**

Ex4: A query using a dataview’s calculated field COUNT to determine the number of orders with a specified number of items for a specified year:

**@order\_request.completed\_date LIKE '%2019%'   
AND @site.site\_short\_name = 'NC7'**

**AND (SELECT count(\*) FROM order\_request\_item WHERE order\_request\_id = order\_request.order\_request\_id) >=250**

##### DISTINCT

The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

Example: List More than 2 inventory owners

**WHERE taxonomy\_genus.genus\_name = 'Zea'**

**AND (SELECT COUNT(distinct i.owned\_by) FROM inventory i WHERE i.accession\_id = accession.accession\_id) > 2**

##### NOT EXISTS

Similar to EXCEPT…

The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns true if the subquery returns one or more records. If a subquery returns any rows at all, EXISTS *subquery* is TRUE, and NOT EXISTS *subquery* is FALSE.

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE EXISTS  
(SELECT *column\_name* FROM *table\_name* WHERE *condition*);

Example: Find records without a recent viability test

**WHERE**

**inventory.inventory\_id IS NOT NULL  /\* necessary if resolving outside inventory \*/**

**AND NOT EXISTS**

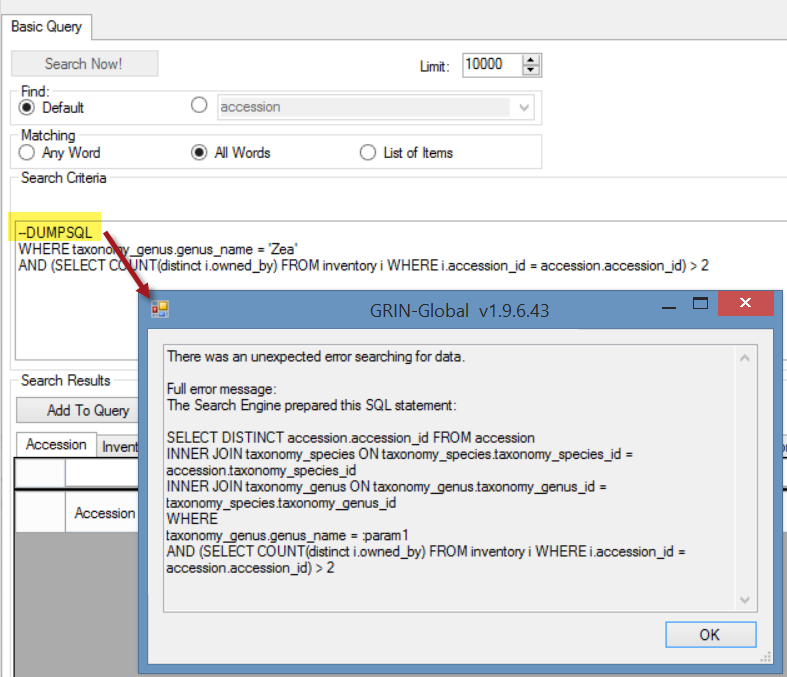
**(SELECT \* FROM inventory\_viability iv**

**WHERE iv.inventory\_id = inventory.inventory\_id    -- link subquery to main select**

**AND datediff(day, iv.tested\_date, getdate() ) < 365)**

##### Displaying the SQL: --DUMPSQL

* SE4 has an option to deliberately throw an error and show the SQL it generated when the first line of the query is this comment: **--DUMPSQL**

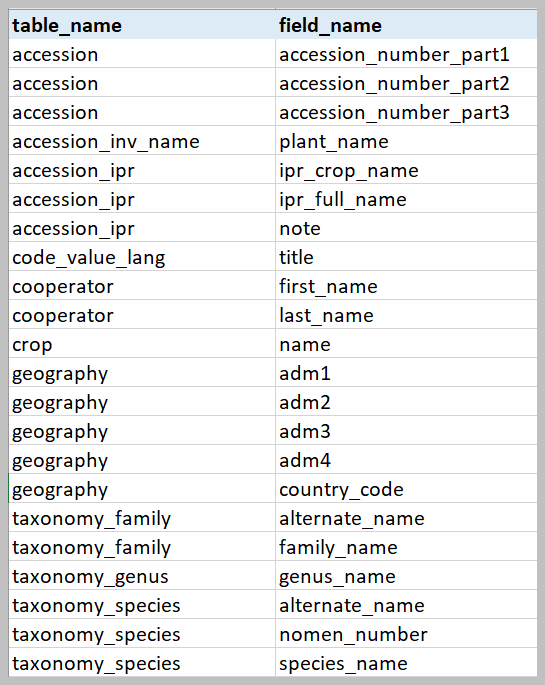


### Appendix A: Fields used in the GG Searches

The GRIN-Global administrator can determine which fields are to be searched using two different approaches. GG “Auto” fields may be designated in the **sys\_search\_autofield** table. The second method requires the GG administrator to use SQL Server’s Full Text Indexing methodology.

#### Autofields

The following fields were designated by the National Plant Germplasm System (NPGS) GG administrator to be used for text box searches. (Every GG genebank can determine what fields are to be included.)



##### SQL to List the “Autofields” Used in the Search Box

**SELECT table\_name, field\_name**

**FROM sys\_search\_autofield ssa**

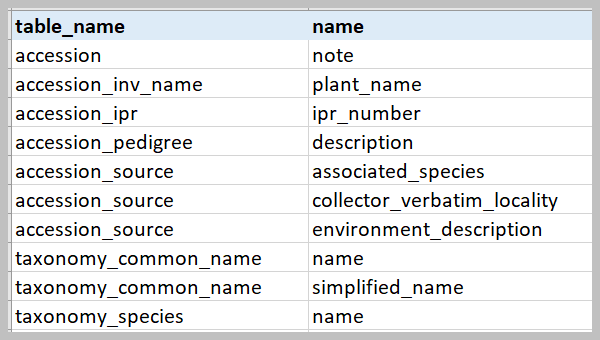
**JOIN sys\_table\_field stf ON stf.sys\_table\_field\_id = ssa.sys\_table\_field\_id**

**JOIN sys\_table st ON st.sys\_table\_id = stf.sys\_table\_id**

**ORDER BY 1,2**

#### Full Text Indexing

The fields listed below were indexed by the National Plant Germplasm System (NPGS) GG administrator.

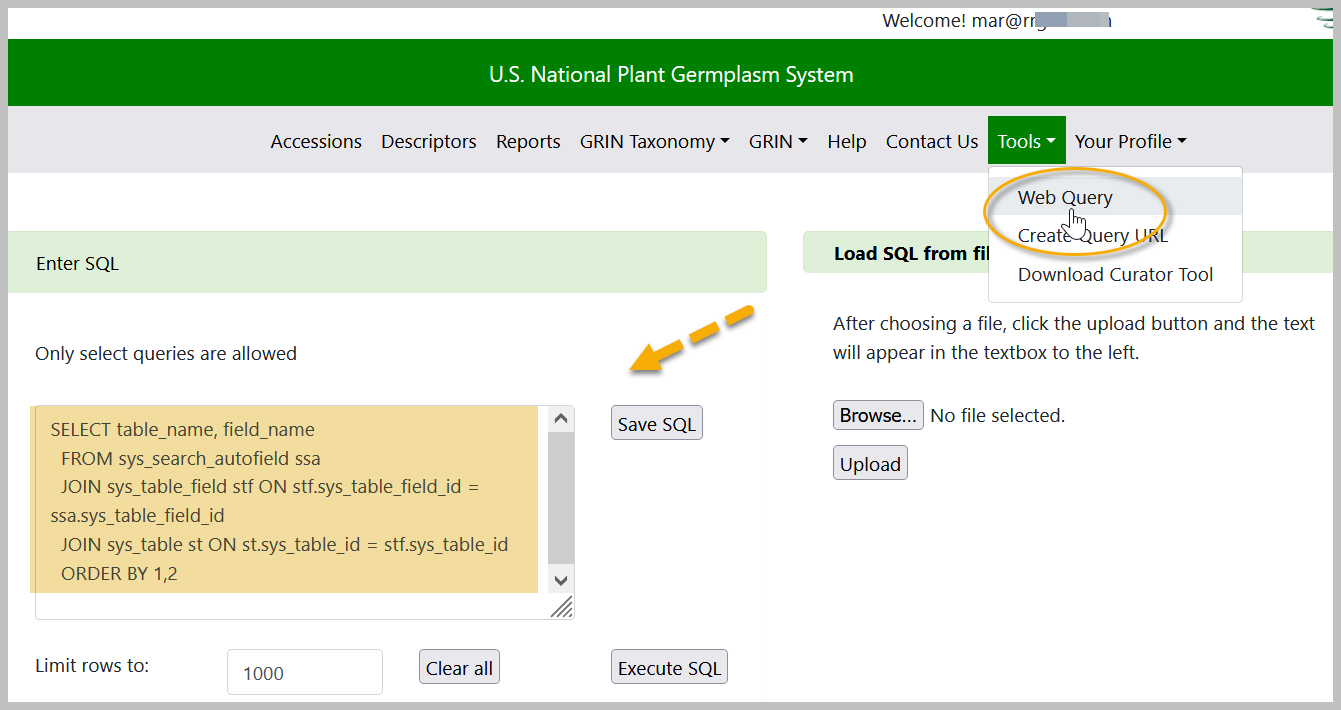


##### SQL to List the Fields Having Full Text Indexes

SELECT DISTINCT  
 object\_name(fic.[object\_id])as table\_name,  
 [name]  
FROM  
 sys.fulltext\_index\_columns fic  
 INNER JOIN sys.columns c  
 ON c.[object\_id] = fic.[object\_id]  
 AND c.[column\_id] = fic.[column\_id]

### Appendix B: SQL Queries on the Public Website

##### Overview

Genebank staff who have had their Public Website account connected to their Curator Tool account by their GG administrator, when logged into the Public Website, will have the **Tools** option visible on the menu. From there, select **Web Query** to display the box for inputting SQL. Log in; select **Tools** | **Web Query** You can copy or type valid SQL in the box as shown:  


You can open a .txt or Word file in which SQL has been stored and cut in paste into the query box, or use the PW feature to Create a query SQL text file.

##### 3 Basic Components

In general, in GRIN-Global, most SQL statements will use these three words.

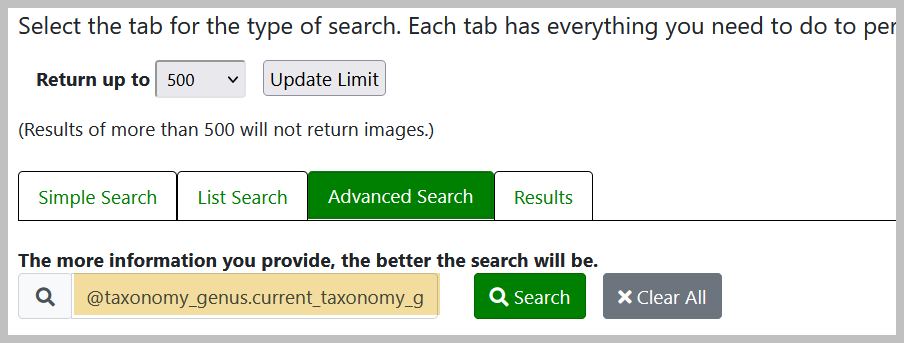
SELECT – what columns to display  
FROM – what tables to search  
WHERE – what criteria

In a valid SQL command, indicate what data you want to display and the conditions. In the GRIN-Global Public Website, a user cannot modify data – only read. Statements such as INSERT or DELETE do not work on the PW page.

Online there are multiple documents, tutorials, and examples on how to use SQL queries on the Public Website. See <https://www.grin-global.org/sql_examples.htm>.

##### Public Website Searches Using the @

On the Public Website, you can also use @ search constructs. While it is not user friendly, if you know the actual table and field names, using these searches provides more search capabilities on the Public Website. Also, internal genebank staff can share these constructs with external users when appropriate.



### Appendix C: Administrator Notes on Sorting Search Results

There are three levels of sort on the output of Public Website searches:

1. Highest weighted field hits first (genus hits before others)
2. Accessions with PI prefixes are listed before Non-PIs\*
3. Most recently received accessions are listed first

\* Organizations other than NPGS that are running GRIN-Global may set the preferred prefix from “PI” to their organizations preferred prefix.

If there are more than 500 (or whatever your maximum limit is set to) accessions that are genus hits on PI numbers, the most recent of those is first. If there are less than 500 PI records for the genus you are searching, going to see recent non-PI genus hits further down the list and recent PI non-genus hits even further down. That is, not all recent accessions will be at the top because the other sorts have a higher precedence.

If as administrator of a GG system you want to change any of that behavior, you’ll need to know how the sorting is controlled.

The first sort is by the weights of the freeform text fields, controlled by the **get\_search\_autofields** dataview. The weights assigned to autosearch fields can be adjusted in the following CASE clause:

  , CASE

      WHEN field\_name IN ('genus\_name', 'title') THEN 0

      WHEN field\_name IN ('species\_name', 'adm1',  
 'accession\_number\_part1') THEN 1

      ELSE 2

END AS weight

Note that “title” refers to the country name from the code lang translation. So hitting on genus name or country name are equally weighted, then species name, state name, or accession prefix for the next level, then the rest of the autosearch fields and finally the full text index hits (not controlled by the dataview).

The other two levels of sort are controlled by the PW dataviews web\_search\_overview\_2 and web\_search\_overview\_noimages\_2 with an ORDER BY clause at the end of the dataview:

-- Put PI numbers first, then sort by date received

ORDER BY CASE WHEN a.accession\_number\_part1 = 'PI' THEN 0 ELSE 1 END, COALESCE(a.initial\_received\_date, a.created\_date) DESC, pi\_number

Actually there is a fourth level of sort by PI number if the received date is exactly the same. Another system could change that ORDER BY to whatever suits them.

The Search Tool retrieves the data in a different fashion. So the sort order as described above for the Public Website doesn’t apply.

### Appendix D: Document Change Notes

– December 21, 2022

* added Note regarding BETWEEN
* also, corrected BETWEEN example

– June 21, 2022

* added Appendix B and details regarding PW sort priority preferences

– February 25, 2022

* added a dynamic query section with a link to the online Dynamic Query guide

– January 10, 2022

* mainly formatting changes

– July 12, 2021

* enhanced the section regarding calculating the actual quantities of viable seeds

– April 20, 2021

* formatted the table headings for the reserved words; therefore the headings are now included in the TOC

– February 2, 2021

* elaborated on the three search types; added screen examples

– October 1, 2020

* added note on comments
* enhanced notes on using search text on the Public Website

– September 20, 2020

* enhanced List Search notes

– August 12, 2020

* expanded information on BETWEEN

– February 29, 2020

* added use case searching using Live Seed (a calculated field)

– April 24, 2019

* changed example and wording for the WHERE clause

– December 17, 2018

* changed example and wording for the WHERE clause