



First Look: SQL Search with InterSystems IRIS

Version 2018.1
2018-01-31

First Look: SQL Search with InterSystems IRIS

InterSystems Version 2018.1 2018-01-31

Copyright © 2018 InterSystems Corporation

All rights reserved.



InterSystems, InterSystems Caché, InterSystems Ensemble, InterSystems HealthShare, HealthShare, InterSystems TrakCare, TrakCare, InterSystems DeepSee, and DeepSee are registered trademarks of InterSystems Corporation.



InterSystems IRIS Data Platform, InterSystems iKnow, Zen, and Caché Server Pages are trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

First Look: SQL Search with InterSystems IRIS.....	1
1 Why SQL Search Is Important	1
2 How InterSystems IRIS Implements SQL Search	2
3 Trying SQL Search for Yourself	2
4 For More Information about SQL Search	4

First Look: SQL Search with InterSystems IRIS

This quick start guide introduces you to InterSystems IRIS Data Platform™ support for SQL text search, which provides semantic context searching of unstructured text data in a variety of languages. It covers the following topics:

- [Why SQL Search Is Important](#)
- [How InterSystems IRIS Implements SQL Search](#)
- [Trying SQL Search for Yourself](#)
- [For More Information about SQL Search](#)

This First Look guide presents an introduction to SQL context-aware text searching and walks through some initial tasks associated with indexing text data for searching and performing SQL Search. Once you've completed this exploration, you will have indexed text in an SQL column for text searching and performed several types of searches. These activities are designed to use only the default settings and features, so that you can acquaint yourself with the fundamentals of the feature. For the full documentation on SQL Search, see the [SQL Search Guide](#).

A related, but separate, tool for handling unstructured texts is [Natural Language Processing \(NLP\)](#). SQL Search presupposes that you know what you are looking for. NLP text analysis allows you to analyze the contents of texts with no prior knowledge of the text contents.

1 Why SQL Search Is Important

The ability to rapidly search unstructured text data is fundamental to accessing the content of the huge volume of text commonly stored by many companies and institutions. Any search facility for such data must have the following functionality:

- **Fast search:** InterSystems IRIS™ SQL Search can rapidly search large quantities of data because it is searching a generated optimized index to the data, rather than sequentially searching the data itself.
- **Word-aware search:** SQL Search is not a string search, it is a search based on semantic structures in the text. The most basic semantic structure for SQL Search is the word. This reduces the number of false positives that result when a string search finds a string embedded in another word, or when a string bridges two words.
- **Entity-aware search:** SQL Search takes into account multiple words that are grouped by semantic relationship to form entities. It can thus search for multiple words in a specified order (a positional phrase), words appearing within a specific proximity to each other (regardless of sequence), and words found at the beginning or at the end of an entity. This enables you to narrow a search to a word (or phrase) found in a specified context of other words.
- **Language-aware search:** identifying semantic relationships between words is language-specific. SQL Search contains semantic rules (language models) for ten natural languages. It also provides support for other languages. It does not require the creation or association of dictionaries or ontologies.
- **Pattern matching:** SQL Search provides both wildcard matching and regular expression (RegEx) matching to match character patterns.

- Fuzzy matching: SQL Search provides fuzzy search for near-matches that take into account a calculated degree of variation from the search string. This enables matching of spelling errors, among other things.
- Derived matching: SQL Search can use decompounding to match root words and component words. SQL Search can use synonym tables to match synonym words and phrases.

2 How InterSystems IRIS Implements SQL Search

SQL Search can search text data found in a column in an SQL table. In order to do this, you must create an SQL Search index for the column containing the text data. InterSystems implements a table column as a property in a persistent class.

There are three levels of index available, each supporting additional features as well as all of the features of the lower levels: Basic, Semantic, and Analytic:

- *Basic* supports word search and positional phrase search, including the use of wildcards, ranges between words in a phrase, regular expression (RegEx) matching, and co-occurrence search.
- *Semantic* supports all of the Basic functionality, and also supports InterSystems IRIS Natural Language Processing (NLP) entities. It can search for entities, and words or phrases that begin an entity or end an entity. It recognizes NLP attributes, such as negation.
- *Analytic* supports all of the Semantic functionality, and also supports NLP paths. It can also search based on NLP dominance and proximity scores.

Populating the index. Like all SQL indices, you can either build the index directly after the table has been populated with data, or have SQL automatically build the index entries as you insert records into an empty table. In either case, SQL automatically updates this index as part of subsequent insert, update, or delete operations.

You perform an SQL search you write a SELECT query in which the WITH clause contains `%ID %FIND search_index()` syntax. The `search_index()` function parameters include the name of the SQL Search index and a search string. This search string can include wildcard, positional phrase, and entity syntax characters. The search string can also include AND, OR, and NOT logical operators.

3 Trying SQL Search for Yourself

It's easy to use InterSystems IRIS SQL Search. This simple procedure walks you through the basic steps of searching text data stored as a string in an SQL table column.

1. Preliminaries

You need to have an InterSystems IRIS instance that is up and running and has an active license key. (You can view the licence key from the Management Portal: select **System Administration** then **Licensing**.)

This documentation uses the Aviation.Event SQL table, which is available on GitHub at <https://github.com/interSystems/Samples-Aviation>. (You do not need to know anything about GitHub or have a GitHub account.) To install these samples, InterSystems recommends that you create a dedicated namespace called (for example) TESTSAMPLES and then load the samples into that namespace (or you can use an existing namespace; however, you cannot use the %SYS namespace). To create a namespace, use the Management Portal options **System Administration** -> **Configuration** -> **System Configuration** -> **Namespaces**. For the general process of downloading from GitHub, see [Downloading Samples for Use with InterSystems IRIS](#). After you download a sample, be sure to open the README file and follow the setup instructions.

2. Create an index for SQL Search. The following class definition example creates a Basic SQL Search index:

```

Class Aviation.TestSQLSrch Extends %Persistent [ ClassType=persistent,
  DdlAllowed,Owner={UnknownUser},ProcedureBlock,SqlRowIdPrivate,
  SqlTableName=TestSQLSrch ]
{
Property UniqueNum As %Integer;
Property Narrative As %String(MAXLEN=100000) [ SqlColumnNumber=3 ];
Index NarrBasicIdx On (Narrative) As %iFind.Index.Basic(INDEXOPTION=0,
  LANGUAGE="en",LOWER=1);
Index UniqueNumIdx On UniqueNum [ Type=index,Unique ];
}

```

Make sure to build this class definition in the TESTSAMPLES namespace. This example creates a persistent class (table) that contains a Narrative property (column). It defines an SQL Search Basic index for this property. Because this is a new class, you must populate this table with text data.

3. Populate the table with text data and build the SQL Search index. An SQL Search index is built and maintained like any other SQL index.

The following example populates the new table with text data from an existing SQL table. In this example, the SQL Search index is automatically built as each record is added:

```

ZNSPACE "TESTSAMPLES"
SET in1="INSERT OR UPDATE INTO Aviation.TestSQLSrch (UniqueNum,Narrative) "
SET in2="SELECT %ID,NarrativeFull FROM Aviation.Event WHERE %ID < 100"
SET myinsert=in1_in2
SET tStatement=##class(%SQL.Statement).%New()
SET qStatus=tStatement.%Prepare(myinsert)
  IF qStatus=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT}
SET rset=tStatement.%Execute()
WRITE !,"Total rows inserted=",rset.%ROWCOUNT

```

For performance reasons, you may wish to use the %NOINDEX option to defer building indices until the table is full populated, and then build the SQL Search index (and any other defined indices) using the %Build() method.

Alternatively, you could add an SQL Search index to an existing persistent class that already contains text data, and then populate the SQL Search index using the %Build() method.

4. Use SQL Search as a WHERE clause condition of a **SELECT** query. The WHERE clause can contain other conditions associated by AND logic. Run the following SQL Query in the TESTSAMPLES namespace:

```

SELECT %iFind.Highlight(Narrative,'"visibility [1-4] mile*" AND "temp* ? degrees"')
FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'"visibility [1-4] mile*" "temp* ? degrees"',0,'en')

```

- The **search_index()** function specifies a *search_index* parameter. This is a defined SQL Search index for the property (column) to be search. It can be a Basic, Semantic, or Analytic index.
- The **search_index()** function specifies a *search_item* parameter.

This example defined the *search_item* as "visibility [1-4] mile*" "temperature ? degree*". This returns all records that contain both positional phrases, in any order:

"visibility [1-4] mile*" returns phrases with from 1 to 4 words between the words “visibility” and “mile”. Because mile* specifies a wildcard, it could match either mile or miles. For example, “visibility less than 1 mile”, “visibility 10 miles”, “visibility approximately 20 statute miles”, “visibility for many miles”.

"temp* ? degrees" returns phrases with a word beginning with “temp” and ending in 0 or more non-space wildcard characters, a single missing word, and then the word “degrees.” Thus it would return records with the phrase “temperature 20 degrees”, “temp. 20 degrees”, “temperature in degrees”, and also the (probably unintended) “temporarily without degrees”.

- The **search_index()** function can optionally specify a *search_option* parameter.
- This option can apply an optional transformation to the search, as follows: 1=stemmed search applies a stemmer to match words or phrases based on their stem form. 2=decompounding search applies decompounding to compound

words. 3=fuzzy search applies a specified degree of fuzziness (number of character differences) to the search. 4=regular expression search allows searching using RegEx matching. This example specifies the default, 0, meaning no search transformation.

- The **search_index()** function can optionally specify a *search_language* parameter. You can specify a language, or specify ' * ' to invoke automatic language identification, supporting searching texts containing multiple languages. This example specifies the default, 'en' (English).

This example also highlights the returned text by applying the same *search_item* to the returned records. This highlights every instance of either of these phrases by delimiting them with and tags.

This example is provided to give you some initial experience with InterSystems IRIS SQL Search. You should not use this example as the basis for developing a real application. To use SQL Search in a real situation you should fully research the available choices provided by the software, then develop your application to create robust and efficient code.

4 For More Information about SQL Search

InterSystems has other resources to help you learn more about SQL Search:

- InterSystems Documentation: [Using InterSystems SQL Search](#)