InterSystems SQL Search の使用法

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

InterSystems SQL Search の使用法
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

1

InterSystems SQL Search ツール

この章では、コンテキスト認識テキスト検索操作を実施するためのツールである InterSystems SQL Search 機能について説明します。InterSystems SQL Search を使用するには、検索したいテキストを含む各列に対して SQL Search インデックスを定義する必要があります。そのうえで、InterSystems SQL Search 構文を含む WHERE 節にて標準 SQL クエリを使用すれば、テキスト・レコードを検索することができます。クエリは、指定された検索項目 (複数可) が含まれるすべてのレコードを返します。返されるレコード内の一致テキストをハイライト表示することもできます。

1.1 開始の前に

InterSystems IRIS® データ・プラットフォーム・インスタンスを起動し、実行している必要があります。また、InterSystems IRIS 自然言語プロセッサ (NLP) にアクセスするための有効な InterSystems IRIS ライセンス・キーが必要です(ライセンス・キーは管理ポータルで表示できます。**[システム管理]** の次に **[ライセンス]** を選択します)。

このドキュメント内のサンプルでは、Aviation.Event SQL テーブルからのデータが使用されています。このサンプル・データを使用する場合は、https://github.com/intersystems/Samples-Aviation で入手できます (GitHub の知識や GitHub アカウントは必要ありません)。**README.md** ファイルの内容を確認します。このファイルは、GitHub リポジトリに含まれるファイル名とディレクトリの下に表示されています。**README.md** ファイルの下部にある “Setup instructions” セクションまでスクロールして、その手順を完了します。

1.2 SQL Search が重要な理由

構造化されていないテキスト・データをすばやく検索する機能は、多くの企業や機関に一般的に保存されている膨大な量のテキストの内容にアクセスするための根幹となるものです。こうしたデータに対する検索機能には、以下のような機能が必要です。

- 高速検索: InterSystems IRIS SQL Search では、データそのものをシーケンシャル検索するのではなく、データに対する生成済みの最適化されたインデックスを検索するので、大量のデータをすばやく検索できます。
- 単語認識検索: SQL Search は文字列検索ではなく、テキストの意味的構造に基づく検索です。SQL Search の最も基本的な意味的構造は単語です。これにより、文字列検索が別の単語に埋め込まれた文字列を見つけた場合や、1 つの文字列が 2 つの単語をブリッジしている場合に起こる誤検出の数が減少します。
- エンティティ認識検索: SQL Search では、意味的関係性によってグループ化されてエンティティを形成する複数の単語が考慮に入れられます。このため、指定された順序の複数の単語 (位置句)、(順序に関係なく) 相互の特定の近似範囲内で表示される単語、およびエンティティの先頭または末尾にある単語を検索できます。これにより、その他の単語の指定されたコンテキスト内にある単語 (または語句) に検索を絞り込むことができます。

- 言語認識検索: 単語間の意味的関係性の識別は、言語に固有のものです。SQL Search には、10 個の自然言語の意味規則 (言語モデル) が含まれています。その他の言語もサポートしています。辞書やオントロジを作成または関連付けする必要はありません。
- パターン・マッチング: SQL Search では、文字パターンの照合のために、ワイルドカード・マッチングと正規表現 (RegEx) マッチングの両方を実行できます。
- ファジー・マッチング: SQL Search では、ほぼ一致するものに対するファジー検索ができます。ここでは、検索文字列からの計算された自由度が考慮に入れられます。これにより、特にスペル・エラーの照合が可能になります。
- 派生マッチング: SQL Search は複合語分解を使用して、語幹と構成単語を照合します。SQL Search では同義語テーブルを使用して、同義語や同義の語句を照合できます。

1.3 SQL Search のためのソースのインデックス作成

SQL Search を使用して、%String データ型または %Stream.GlobalCharacter (文字ストリーム) データ型のテキストを検索できます。

SQL 検索を実施するには、検索する列が定義済みの SQL Search ビットマップ・インデックスを有する必要があります。SQL Search インデックスには、4 段階のレベルが存在します。これらのレベルは入れ子にしたサブクラスにて定義します。各インデックス・レベルでは、前段レベルの全機能に加えて、そのレベル特有の追加 SQL Search 機能を備えます。以下の SQL Search インデックスのうちの任意のタイプを作成できます。

- 最小インデックス (%iFind.Index.Minimal): ワイルドカードによる SQL 単語検索および語句検索、ファジー検索、および正規表現をサポートします。同時検索、位置句検索、および検索結果のハイライト表示はサポートしません。
- 基本インデックス (%iFind.Index.Basic): SQL 単語検索およびワイルドカードによる語句検索をサポートします。同時検索、位置句検索、および検索結果のハイライト表示をサポートします。
- 意味的インデックス (%iFind.Index.Semantic): NLP エンティティの SQL 検索、および必要に応じて否定属性をサポートします。
- 分析インデックス (%iFind.Index.Analytic): 意味的インデックスに関する NLP 機能をすべてサポートします。パス、近似、および優位性情報もサポートします。

各インデックス・レベルでは、前段レベルのパラメータをすべてサポートしており、さらに追加パラメータが 1 つ以上加わります。指定がなければ、パラメータは既定の値となります。

以下のクラス定義の例では、Narrative プロパティ (列) に意味的インデックスでテーブルを作成します。インデックスの付いたプロパティのデータ型は、%String または %Stream.GlobalCharacter にできます。

Class Definition

```
 Class Aviation.TestSQLSrch Extends %Persistent [
    DdlAllowed,Owner={UnknownUser},SqlRowIdPrivate,
    SqlTableName=TestSQLSrch ]
{
Property UniqueNum As %Integer;
Property CrashDate As %TimeStamp [ SqlColumnNumber=2 ];
Property Narrative As %String(MAXLEN=100000) [ SqlColumnNumber=3 ];
Index NarrSemanticIdx On (Narrative) As %iFind.Index.Semantic(INDEXOPTION=0,
   LANGUAGE="en",LOWER=1);
Index UniqueNumIdx On UniqueNum [ Type=index,Unique ];
}
```

いずれのタイプの SQL Search インデックスでも、以下のパラメータがサポートされます。

- IGNOREPUNCTUATION は、句読点文字を無視するかどうかを指定します。%iFind.Index.Minimal の場合、既定値は 1 です。句読点は無視されます。他のすべての SQL Search インデックス・タイプの場合、既定値は 0 です。句

読点は検索結果に影響します。テキスト内の先頭および末尾の句読点が検索文字列内の同じ句読点と一致する必要があります。

- INDEXOPTION は、インデックスで語幹解析または複合語分解を使用可能にするかどうかを指定します。これらの操作をサポートするインデックスの作成では、インデックスのサイズがきわめて大きくなるため、語幹解析または複合語分解を使用する可能性が高くない限り、INDEXOPTION=0 と指定することをお勧めします。既定は 0 です。
- LANGUAGE は、レコードのインデックス作成で使用する言語を指定します。例えば、"en" は英語を指定します。自動言語識別を有効にするには、"*" を使用します。既定は "en" です。
- LOWER は、クエリ検索で大文字と小文字を区別するかどうかを指定します。既定で、InterSystems SQL Search インデックス作成では大文字と小文字が区別されません。SQL Search では、テキストのインデックス作成前にそのテキストが小文字に正規化されます。LOWER パラメータは、この小文字への正規化を実施するかどうかを指定します (既定は LOWER=1 で、小文字に正規化されます)。一般的に言語規則では、文の先頭で、またはタイトルでの使用時に単語を大文字にするため、小文字への正規化は大半のアプリケーションに推奨されます。LOWER=0 と指定すると、クエリの search_items 文字列では大文字と小文字が区別されます。例えば、LOWER=0 と指定すると、クエリの search_items 文字列 'turkey' に一致するのは turkey のみで、Turkey は一致しません。LOWER=1 と指定すると、クエリの search_items 文字列 'turkey' には turkey と Turkey の両方が一致します。
- USERDICTIONARY では、インデックス作成前にテキストに適用されるユーザ定義の UserDictionary の名前を指定できます。このパラメータはオプションで、上級者専用です。

サポートされるパラメータすべてのリストは、"インターシステムズ・クラス・リファレンス" の "%iFind.Index.Basic" を参照してください。

意味的インデックス (%iFind.Index.Semantic) では、以下のオプションのパラメータもサポートされます。

- IFINDATTRIBUTES では、テキストで否定を識別して否定属性を格納するかどうかを指定できます。IFINDATTRIBUTES=1 と指定すると、否定が識別され、否定のインデックスが作成されます。既定は IFINDATTRIBUTES=0 です。

1.3.1 テーブルへのデータ移入

あらゆる SQL インデックスと同様に、定義済み SQL Search インデックスは (既定で) 新しいテーブルへのデータの移入時に構築され、その後データを挿入、更新、または削除するときに保持されます。%NOINDEX を使用してテーブルへのデータ移入時のインデックス構築を延期し、後で %Build() メソッドを使用してインデックスを構築することができます。既にデータが移入されているテーブルにインデックスを追加してから、そのインデックスを構築することができます。詳細は、"インデックスの定義と構築" を参照してください。

以下の例では、Aviation.Events テーブルから Aviation.TestSQLSrch テーブルを生成します。あらゆる定義済み SQL Search インデックスが、自動的に構築されます。この例では大量のテキストが挿入されるため、実行時間が 1 分ほどかかる場合があります。

SQL

```
INSERT OR UPDATE INTO Aviation.TestSQLSrch (UniqueNum,CrashDate,Narrative)
    SELECT %ID,EventDate,NarrativeFull FROM Aviation.Event
```

この例では、フィールドが一意キーで定義されている INSERT OR UPDATE を使用し、繰り返しの実行によって重複レコードが作成されないようにします。

1.4 SQL Search の実施

SQL クエリの WHERE 節で SQL Search 構文を使用して、1 つ以上のテキスト・アイテムに対するテキスト検索を実施します。これらのテキスト・アイテムは単語、単語の羅列 (基本インデックス)、または NLP の意味的エンティティ (意味的インデックス) とすることができます。複数のテキスト・アイテムは暗黙的な AND 検索となり、すべての指定項目を順不同でテキストに出現する必要があります。SQL Search 検索の構文は以下のようになります。

```
WHERE %ID %FIND
search_index(indexname,'search_items',search_option,'language','synonym_tables')
```

- indexname は、特定の列に対して定義された SQL Search インデックスの名前です。
- search_items は、検索するテキスト・アイテム (単語または NLP エンティティのいずれか) の一覧であり、引用符で囲みます。テキスト・アイテムは空白で区切られます。アイテムは英数字文字列およびオプションのワイルドカード構文文字で構成されます。既定では、テキスト・アイテムでは大文字と小文字が区別されません (LOWER パラメータを参照してください)。使用可能な search_items 構文については以下で説明しています。
- search_option は、実施する検索タイプを指定するためのインデックス・オプション整数です。使用可能な値には、0 (構文検索)、1 (語幹解析による構文検索)、2 (複合語分解と語幹解析による構文検索)、3 (ファジー検索による構文検索)、4 (正規表現による構文検索) があります。search_option=4 の場合、search_items には 1 つの正規表現文字列が含まれていると想定されます。詳細は、"ObjectScript の使用法" の "正規表現" の章を参照してください。パフォーマンス上の理由により、SQL Search では一部の高度な正規表現の構文形式がサポートされていません。こういった構文形式は、$LOCATE ObjectScript 関数でサポートされています。
- language は、適用する NLP 対応の言語モデルで、2 文字の文字列で指定します。例えば、'en' は英語を指定します。'*' を指定すると、クエリでは自動言語識別が実施されます。
- synonym_tables は、同義語テーブルの名前が含まれる文字列、または同義語テーブルのコンマ区切りのリストです。

基本インデックス検索の実施時、SQL Search では 1 つ以上の空白文字の存在により単語を認識します。文の句読点 (後に空白が続くピリオド、コンマ、セミコロン、またはコロン) は無視されます。SQL Search では、他の句読点はすべてリテラルとして扱われます。例えば、SQL Search は “touch-and-go“ を単一の単語として扱います。ハイフンや数値内の小数点などの句読点はリテラルとして扱われます。引用文字およびアポストロフィを指定する必要があります。その場合は、一重引用符を 2 つ用いて指定します。

既定では、テキストは小文字に正規化されます (大文字と小文字は区別されません)。文字幅のみが異なる日本語文字の正規化など、言語固有のその他の正規化も実行されます。

意味的インデックスを使用して基本インデックス検索 (単語、同時、位置句) を実行できます。基本インデックスを使用して意味的インデックス検索を実行しようとすると、SQLCODE -149 エラーが返されます。

1.4.1 SQL search_items 構文

基本インデックス search_items には、以下の構文を含めることができます。

単語検索 :

引数	説明
word1 word2 word3	テキスト内のいずれかの場所において、これらと同一の単語が存在しなければならないことを指定します (順不同)。(論理 AND)。1 つの単語を指定することも、任意の数の単語をスペースで区切って指定することもできます。
word1 OR word2 NOT word3 word1 OR (word2 AND word3)	search_items には AND、OR、および NOT 論理演算子を含めることができます。AND は単語をスペースで区切ることと同じです (暗黙的な AND)。NOT は論理的に AND NOT と同じです。search_items では、論理演算子をグループ化するために括弧を使用することもできます。明示的な AND は、グループ化括弧内の複数の単語を指定する際に必要です。例 : (word2 AND word3)。明示的な AND が省略された場合、(word2 word3) は位置句として解釈されます。エスケープ文字「¥」を使用して AND、OR、NOT を論理演算子ではなくリテラルとして指定できます。例 : ¥and
?word word? w?rd w??d	疑問符ワイルドカードは、任意のタイプの 1 つの非スペース文字を指定します。接頭語や接尾語として、または単語内で 1 つ以上の ? ワイルドカードを使用できます。? ワイルドカードと * ワイルドカードを組み合わせることができます。エスケープ文字「¥」を使用してリテラルとして「?」を指定できます。例 : ¥?
word word *word* w*d	アスタリスク・ワイルドカードは、0 個以上の任意タイプの非空白文字を指定します。アスタリスクは接頭語、接尾語として、または単語内で使用できます。エスケープ文字「¥」を使用してリテラルとして「*」を指定できます。例 : ¥*

同時単語検索 :

引数	説明
[word1,word2,...,range]	同時検索。range で指定された近接ウィンドウ内で、これらと同一の単語が存在しなければならないことを指定します (順不同)。任意の数の単語または複数語句を指定できます。複数語句では、区切り句読点がなくスペースで区切られた単語として指定されます。単語 (または位置句) はコンマで区切られ、最後のコンマ区切り要素はオプションの数値の range です。単語はアスタリスク・ワイルドカードを指定できます。 rangeは min-max として、または単純に min に既定の 1 が指定された max として指定できます。例えば、1-5 や 5 です。range はオプションです。省略すると、既定により 1-20 となります。範囲の数値は指定した単語すべてにおいてその数値を含みます。 同時検索は、search_option=4 (正規表現) を指定した場合には使用できません。

位置句検索：

注釈　二重引用符 "word1 word2 word3" または括弧 (word1 word2 word3) を使用して、位置句を区切ることができます。括弧は論理演算子のグループ化にも使用されるため、二重引用符の使用をお勧めします。

引数	説明
″word1 word2 word3″	これらと同一の単語が指定順にて連続して存在する必要があります。単語はスペースで区切られます。意味分析は実施されないことに注意してください。例えば、“句“内の単語は文の最終単語および次文の先頭単語となる場合があります。アスタリスク・ワイルドカードは、句の個々の単語に適用できます。search_items のリテラル括弧文字は引用符で囲む必要があります。
″word1 ? word3″ ″word1 ? ? ? word5″	疑問符は、句内の指定単語の間に 1 つの単語があることを示します。それぞれをスペースで仕切れば、複数の疑問符を指定できます。
″word1 ?? word6″	二重疑問符 (間にスペースなし) は、句内の指定単語の間に 0 ～ 6 個の単語があることを示します。
″word1 [1-3] word5″	角括弧は、句内の指定単語の間の単語の間隔数を示します。min-max。この間隔は、変数の範囲として指定されます。この場合には 1 ～ 3 個の単語が不足しています。

意味的インデックス search_items では、基本インデックス構文に加えて、以下の NLP エンティティ検索構文を含むことができます。

フル・エンティティ検索および部分エンティティ検索：

引数	説明
{entity}	NLP エンティティの同一文言を指定します。アスタリスク・ワイルドカードは、エンティティの個々の単語に適用できます。
<{entity}	< 記号の接頭語は指定の単語 (複数可) で終わる NLP エンティティを指定します。指定の単語の前に現れるエンティティ内には、1 つ以上の単語が存在する必要があります。
{entity}>	> 記号の接尾語は指定の単語 (複数可) で始まる NLP エンティティを指定します。指定の単語の後に現れるエンティティ内には、1 つ以上の単語が存在する必要があります。

空白で区切ることにより、複数の search_items を指定することができます。これは暗黙的な AND テストとなります。例えば以下のようになります。

SQL

```
SELECT Narrative FROM Aviation.TestSQLSrch WHERE %ID %FIND
search_index(NarrSemanticIdx,'<{plug electrode} "flight plan" instruct*',0,'en')
```

これは、Narrative テキストに、“plug electrode” で終わる 1 つ以上の SQL Search エンティティ、位置句 “flight plan”、ワイルドカード接尾辞が指定された単語 “instruct” (つまり、“instructor”、“instructors”、“instruction”、“instructed” などが可能) が含まれている必要があることを意味します。これらの項目のテキスト内での順序は任意です。

1.4.2 SQL 検索項目文字列の検証

%iFind.Utils.TestSearchString() メソッドを使用すると、search_items 文字列を検証できます。このメソッドを使用することで、論理演算子の構文エラーおよび曖昧な使用を検出できます。例えば、`"word1 AND word2 OR word3"` は、論理的に曖昧なため検証で不合格となります。括弧を追加し、`"word1 AND (word2 OR word3)"` または `"(word1 AND word2) OR word3"`.のいずれかにしてこの文字列を明確にします。

次の例は、この SQL Search ユーティリティを SQL 関数として呼び出します。

SQL

```
SELECT %iFind.TestSearchString('orange AND (lemon OR lime)')
```

TestSearchString() は、%Status の値を返します。有効な search_items 文字列はステータス 1 を返します。無効な search_items 文字列の場合、0 で始まり、その後にエンコードされたエラー情報が続くオブジェクト式を返します。

1.4.3 ファジー検索

検索文字列と “おおまか“ に一致している要素 (単語またはエンティティ) を含むレコードを照合するファジー検索を InterSystems SQL Search はサポートしています。ファジー検索は、記述方法のわずかな違い (color と colour) や誤字 (collor と color)、および文法的に異なる形式 (color と colors) を見つけるために使用できます。

SQL Search では、2 つの単語間のレーベンシュタイン距離を比較して、ファジー一致を評価します。レーベンシュタイン距離とは、1 つの単語を他の単語に変更するために必要な単一文字編集の最小数 (挿入、削除または置換) です。必要な単一文字編集の最大数は、最大編集距離として知られています。InterSystems SQL Search の最大編集距離は、既定では 2 文字です。最大編集距離は検索文字列の各要素に個別に適用されます。SQL Search 基本インデックスでは、検索文字列の各単語に適用されます。SQL Search 意味的インデックスでは、検索文字列の各 NLP エンティティに適用されます (以下の例では、SQL Search 基本インデックスを想定しています)。

例えば、“analyse programme behaviour” という語句は、最大編集距離=2 の場合、ファジー検索で “analyze program behavior” と一致します。これは、検索文字列内の各単語が次のように、(最大) 2 文字の編集距離だけ異なるためです。analyse=analyze (1 文字置換)、programme=program (2 文字削除)、behaviour=behavior (1 文字削除)。

最大編集距離以下である単語の場合、最大編集距離の値以下の文字数を含むすべての単語がファジー検索で一致します。例えば、編集距離が 2 で、単語が “ab” の場合、任意の 2 文字の単語 (2 置換)、任意の 1 文字の語 (1 置換、1 削除)、“a” または “b” のいずれかを含む任意の 3 文字の単語 (1 置換、1 挿入)、“a” および “b” の両方をこの順序で含む任意の 4 文字の単語 (2 挿入) が一致することになります。

- ファジー検索は、基本、意味的、および分析のすべての SQL Search インデックス・タイプでサポートされています。基本インデックスでは、個別の単語に対してファジー検索が実行されます。意味的インデックスでは、個別の NLP エンティティに対してファジー検索が実行されます。
- ファジー検索はワイルドカード検索と組み合わせることはできません。

search_index() のファジー検索を有効にするには、既定の編集距離が 2 のファジー検索で search_option を 3 と指定します。または編集距離が n 文字と指定されているファジー検索では、3:n と指定します。以下の例は、編集距離が 4 のファジー検索が指定された SQL Search を示しています。

SQL

```
SELECT Narrative FROM Aviation.TestSQLSrch WHERE %ID %FIND
search_index(NarrBasicIdx,'"color code" program','3:4','en')
```

3:1 と設定すると、編集距離=1 に設定されます。これにより、英語ではほとんど (すべてではありません) の単数および複数形の単語を照合できます。3:0 に設定すると、編集距離=0 に設定されます。これは、ファジー検索なしの SQL Search と同じです。

SQL Search メソッドのファジー検索を指定するには、pSearchOption = $$$IFSEARCHFUZZY と設定します。

1.4.4 語幹解析と複合語分解

語幹解析と複合語分解は、基本インデックス、意味的インデックス、および分析インデックスのすべてでサポートできます。語幹解析と複合語分解は単語ベースで、NLP エンティティベースの操作ではありません。SQL Search インデックスを定義する際に語幹解析と複合語分解を有効にする必要があります。インデックスで語幹解析対応の検索を使用できるようにするには、INDEXOPTION=1 を指定します。語幹解析対応の検索と複合語分解対応の検索の両方を使用できるようにするには、INDEXOPTION=2 を指定します。

語幹解析 (1) または語幹解析と複合語分解 (2) に対応するよう定義されている SQL Search インデックスの場合、search_option の値を設定することによって search_index() クエリでこれらの機能を使用することができます。

1.4.4.1 語幹解析

語幹解析により、各単語の語幹形式が識別されます。語幹形式により、同じ単語に対する複数の文法的な形式が統一されます。クエリの実行時に search_option=1 を使用すると、SQL Search は、実際のテキスト形式ではなく、単語の語幹形式を使用して、検索処理とマッチング処理を実行します。search_option=0 を使用すると、通常の (語幹形式を使用しない) 検索に、同じインデックスを使用できます。

語幹解析が有効な場合、検索語の語幹形式を特定し、その語幹形式を使用してテキスト内の単語と照合することによって、検索とマッチングが実行されます。例えば、`doctors` という検索語は、テキスト内の `doctor` または `doctors` と一致します。語幹解析が有効なときには、検索リスト内の 1 つの単語を引用符で囲むことによって、検索語をテキスト内の完全一致の単語のみと一致させることができます。`"doctors"` という検索語は、テキスト内の `doctors` のみと一致します。

1.4.4.2 複合語分解

複合語分解では複合語を構成単語に分割します。SQL Search は常に、複合語分解と語幹解析を組み合わせます。単語が構成部分に分解されると、それぞれの部分は自動的に語幹抽出されます。複合語分解検索 (search_option=2) を使用すると、SQL Search は、検索語の複合語分解する語幹と、インデックス作成されたテキスト・フィールド内の複合語分解された単語の語幹を比較します。複合語分解された構成単語のいずれかの語幹が、その検索語のすべての構成単語と一致する場合、SQL Search は複合語分解されている単語のみをマッチングします。

例えば、検索語が “thunder“、“storm“、または “storms“ の場合は、すべてが “thunderstorms“ と一致します。ただし、検索語が “thunderstorms“ の場合は、“thunder“ とは一致しません。これは、もう 1 つの構成単語 (“storm“) が一致しないためです。

InterSystems SQL Search の複合語分解アルゴリズムは、可能性のある構成単語を識別する言語固有のディクショナリを使用します。このディクショナリには、%iKnow.Stemming.DecompoundingUtils クラスを通じてデータを入力する必要があります。例えば、インデックス作成の前に、このクラスがテキスト列をポイントするようにします。特定の単語には、複合語分解を適用しないこともできます。個別の単語、文字シーケンス、およびトレーニング・データの単語リストに対する複合語分解の適用を除外するには、%iKnow.Stemming.DecompoundUtils を使用できます。

1.4.5 InterSystems IRIS 自然言語プロセッサでサポートされない言語

SQL Search 基本インデックスを使用すると、対応する NLP 言語モデルが存在しない言語のテキストにインデックスを作成して検索できます。

語幹解析は、NLP の意味的インデックス作成に依存しないため、語幹解析機能が使用できる場合は、単語の語幹形式に対する基本インデックスの単語検索も実行できます。語幹検索を実行するには、INDEXOPTION=1 または INDEXOP-

TION=2 を指定する必要があります。例えば、イタリア語は NLP でサポートされていない言語ですが、InterSystems IRIS にはイタリア語に対応した %Text 語幹解析機能が用意されています。

NLP でサポートされていない言語の SQL Search には、以下の制限と注意事項が適用されます。

- この機能を使用するには、InterSystems IRIS 自然言語プロセッサのライセンスが必要です。
- 言語は、スペースを使用して単語が区切られている必要があります。単語の区切り文字を使用しない言語は検索できません。ただし、日本語 (単語の区切り文字を使用しない言語) では NLP が日本語言語モデルを提供しているため、検索できます。
- アポストロフィでは、単語が区切られません。NLP は、短縮形 (“can’t” など) と動詞の短縮形 (“there’s” など) を認識して、2 つの単語に分割します。その一方で、所有を表す (“John’s”) など、その他の目的で使用されるアポストロフィは無視します。NLP のサポートがない場合、SQL Search は短縮形を別々の単語に分割できません。これは、テキストを前処理して、アポストロフィの前後に必要に応じて空白スペースを挿入することで補えます。
- SQL Search インデックス作成前に UserDictionary をテキストに適用することはできません。

詳細は、″InterSystems SQL の使用法″ の “データベースの問い合わせ” の章にある ″フリー・テキスト検索を呼び出すクエリ″ を参照してください。

1.4.6 同義語テーブル

同義語テーブルを実装するには、**iFind.Synonym** 抽象クラスを拡張する永続クラスとしてそのテーブルを定義します。

このクラスでは、2 つのプロパティ FromTerm と ToTerm が定義されます。FromTerm プロパティと ToTerm プロパティのペアによって、ToTerm が FromTerm の同義語として定義されます。SQL Search では、クエリに FromTerm が含まれる場合、ToTerm を使用してそのクエリが拡張されます。

クエリでは、このクラスの GetMatch() メソッドを使用して、同義語テーブル内の同義語がクエリ用語に対して検索されます。

クエリの実行中、SQL Search は、単一の単語単位または位置検索句に対する同義語がないかどうか確認します。例えば、2 つの同義語ペア (″persons″,″people″) および (″walk″,″run″) が同義語テーブルに定義されていて、句 ″persons walk″ に対して SQL Search クエリを実行するとします。同義語テーブルがクエリに関連付けられている場合、SQL Search は元のクエリに一致するドキュメントだけでなく、″persons run″、″people walk″、および ″people run″ のいずれかのクエリに一致するドキュメントも返します。

ただし、search_items 文字列が `'"persons walk"'` の場合、クエリは拡張されません。SQL Search では、位置句検索内の単語が拡張されることはないからです。位置句自体は、クエリ拡張の最小単位です。ただし、(″persons walk″,″persons walk and run″) のような同義語ペアを定義した場合、SQL Search ではクエリ `'"persons walk"'` が `'"persons walk and run"'` に拡張されます。SQL Search は、ToTerm に複数の単語が含まれる場合、これを位置句として扱います。ToTerm には、あらゆる有効な位置句を指定でき、ワイルドカードの * または ? を含めることができます。

注釈　　同義語テーブルを正規表現検索 (search_option=4) と共に使用することはできません。

1.5 ハイライト表示

search_items 構文を使用して、返されるテキスト内の単語をハイライト表示することができます。ハイライト表示構文は、以下のとおりです。

```
(text,search_items,search_option)
```

search_items：ハイライト表示では、検索と同じ search_items 構文を使用します。これにより、レコードを返す処理と、それらのレコードが返される要因となったレコード内の文字列をハイライト表示する処理の両方で、同じ search_items 値を使用できるようになります。また、TestSearchString() メソッドを使用して、ハイライト表示の search_items 構文を検証することもできるようになります。ただし、ハイライト表示は、一致ごとに各インスタンスに適用されるため、ハイライト表示では、search_items 文字列内の search_items 構文の AND、OR、および NOT 論理演算は無視されます。

search_option：オプションの search_option には、0 (既定値) または 4 (正規表現) を指定できます。

以下のいずれかを使用して、ハイライト表示を適用できます。

- SELECT 項目ハイライト表示：

  SQL

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

- ユーティリティ・メソッド・ハイライト表示：

  %iFind.Utils.Highlight() メソッドを使用して、SQL Search を実施し、結果にハイライト表示を適用できます。

既定で、ハイライト表示では、文字列内の該当する場所に および (太字) XML タグが挿入されます。既定では、ハイライト表示で大文字と小文字は区別されません。

以下の例で示しているように、ハイライト表示は、正規表現検索 (search_option=4) を含め、あらゆる search_option と共に使用できます。

ObjectScript

```
SET x="Time flies like an arrow. other stuff. Fruit flies like a banana."
WRITE ##class(%iFind.Utils).Highlight(x,"\p{LU}(\p{L}|\s)+",4)
```

単語検索と共に使用する場合、このメソッドでは、指定された各単語が出現するたびに個別にハイライト表示されます。

位置句検索と共に使用する場合、このメソッドでは、位置句が出現するたびにハイライト表示されます。

1.6 SQL Search の例

以下の例では、任意のタイプの SQL Search インデックスにて、SQL Search 基本インデックス構文を使用することができます。SQL Search 意味的インデックス構文では、意味的または分析インデックスが必要となります。

これらの例では、この章で前述の "SQL Search のためのソースのインデックス作成" で説明されているとおりに、Aviation.TestSQLSrch テーブルを作成して、データを入力しておく必要があります。

表示を簡略化するため、これらの例では、レコード・テキストそのものではなく、レコード・カウントを返しています。これらのカウントは、検索基準と一致するレコードの数であり、レコード内で検出された一致数ではありません。レコードは複数の一致を含んでいることもありますが、カウントされるのは 1 回のみとなります。

1.6.1 基本検索の例

以下の例は、基本インデックス検索を使用して Aviation.TestSQLSrch テーブルを検索します。

単語 “electrode”、“plug”、および “spark” (順不同) のインスタンスの 1 つ以上を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'electrode plug spark',0)
```

これは単語検索であり、文字列検索ではないことに注意してください。したがって、以下の例では異なる結果が返り、実際に前の例よりも多くの結果が返る場合があります。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'electrodes plug spark',0)
```

“electrode” で始まる単語 (electrode、electrodes)、および語句 “spark plug” (順不同) のインスタンスの 1 つ以上を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'electrode* "spark plug"',0)
```

6 つの単語の同時近接ウィンドウ内に “electrode” で始まる単語 (electrode、electrodes)、および語句 “spark plug” (順不同) を含むレコードを検索します。同時検索で単語および語句を指定するために使用する句読点に注意してください。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'[electrode*,spark plug,1-6]',0)
```

2 つの異なる語句 `normal wear` および `"normal" wear` を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'"normal wear"',0)
```

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'"\"normal\" wear"',0)
```

文字列 seal (seal、seals、unseal、sealant、sealed、previously-sealed など) を含む単語を最低 1 つ、および語句 “spark plug“ を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'*seal* "spark plug"',0)
```

ワイルドカード語句 “wind from ? ? at ? knots” を含むレコードを検索します。考えられる値としては、“wind from the south at 25 knots” および ”wind from 300 degrees at ten knots“ があります。連続する 2 つの疑問符の間にスペースがある場合 (? ?)、ワイルドカードは、正確に 2 つの単語を表します。2 つの疑問符の間にスペースがない場合 (??)、ワイルドカードは、0 ～ 6 個の単語を表します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'"wind from ? ? at ? knots"',0)
```

以下の例では、基本インデックスを正規表現検索 (n=4) と共に使用します。これは、“January 10“ から “January 29“ までの間の日付を指定する文字列の出現を含んだレコードの検索となります。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrBasicIdx,'"January [1-2][0-9]"',4)
```

詳細は、″ObjectScript の使用法″ の ″正規表現″ を参照してください。

1.6.2 意味的検索の例

以下の例は、意味的インデックス検索を使用して Aviation.TestSQLSrch テーブルを検索します。

NLP エンティティ “spark plug electrodes“ を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrSemanticIdx,'{spark plug electrodes}',0)
```

“spark plug” または “spark plugs” で終わる NLP エンティティを含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrSemanticIdx,'<{spark plug*}',0)
```

“spark plugs“ で終わる NLP エンティティと NLP エンティティ “spark plugs“ の両方を含むレコードを検索します。

SQL

```
SELECT COUNT(Narrative) FROM Aviation.TestSQLSrch
WHERE %ID %FIND search_index(NarrSemanticIdx,'<{spark plugs} {spark plugs}',0)
```

2

SQL Search 用 REST インタフェース

この章では、InterSystems SQL Search 用の REST インタフェースについて説明します。

2.1 SQL Search 用 REST 構文

SQL Search には、SQL Search インデックス作成結果にアクセスするための REST API が用意されています。

エンドポイント構文は `"/table/:TableName/search"` です。TableName は、1 つ以上の SQL Search インデックスが含まれるテーブルです。

アクセス・パスのサンプルとしては、
`http://localhost:52773/api/iKnow/v1/user/table/iFind.Table/search` のようになります。

エンドポイントへのアクセスには、"POST" を使用する必要があります。要求の本文に、JSON オブジェクトとして検索パラメータを入れることができます。

2.1.1 検索パラメータ

サポートされるパラメータは以下のとおりです。

```
{
  "query":"string",
  "index":"string",
  "option": 0,
  "distance": "string",
  "language":"string",
  "includeText": 0,
  "columns": ["string"],
  "highlightSpec": {
     "tag": "<b>",
     "limit": 0,
     "name": "Highlighted"
     },
  "rankSpec": {
    "name": "Rank"
    }
}
```

`"query"` は、SQL Search クエリを指定します。

`"index"` は、検索する SQL Search インデックスを指定します。インデックスを指定しないと、SQL Search では最初に見つかったインデックスが使用されます。

"option" は、実施する検索タイプを指定する整数です。使用可能な値には、0 (構文検索)、1 (語幹解析による構文検索)、2 (複合語分解と語幹解析による構文検索)、3 (ファジー検索による構文検索)、4 (正規表現による構文検索) があります。

"language" は、テキストの言語を指定する 2 文字の文字列を指定します。例えば、"en" は英語を指定します。

"includeText" と "columns" は、返す列を指定します。"includeText" に 1 を指定した場合、SQL Search インデックスの付いたフィールドが返されます。"columns" は、JSON 配列の文字列として他の列名を指定するために使用できます。

"highlightSpec {name}" は、ハイライト表示の列に列エイリアス名を指定します。

"rankSpec {name}" は、ランク表示の列に列エイリアス名を指定します。

返される結果の形式は、JSON オブジェクトです。返される行は、以下の ″rows″ JSON 配列の形式になります。{ "rows":[{},{},{}] }