2019-06-05 - SLPG Meeting

Date & Time
20:00 UTC Wednesday 5th June 2019

Teleconference Details
To join the meeting please go to https://snomed.zoom.us/j/471420169.
Further information can be found at SLPG meeting information.

Goals
- Review actions from last meeting
- Proposed enhancements to expression constraint language
  - Term searching
- Proposed new language features for mapping

Attendees
Chair: Linda Bird
Project Group: Daniel Karlsson, Michael Lawley, Anne Randorff Højen, Ed Cheetham, Rob Hausam

Agendas and Meeting Notes

<table>
<thead>
<tr>
<th>Description</th>
<th>Owner</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome and apologies</td>
<td>Linda</td>
<td>Bird</td>
</tr>
<tr>
<td>Actions from last week</td>
<td>Linda</td>
<td>Bird</td>
</tr>
<tr>
<td>Introducing term searching to ECL</td>
<td>Linda</td>
<td>Bird</td>
</tr>
</tbody>
</table>

**Term Search Type**

1. Wild Card Match (collation) - e.g.
   - `{{ term = wild:**"heart"* }}`
   - `{{ term = wild (sv):"hjärta"* }}`
2. Perl Regex - e.g.
   - `{{ term = regex:**"heart.*"* }}`
3. Word Prefix Any Order - e.g.
   - `{{ term = match:**hear att"* }}`
4. Default (word prefix any order) - e.g.
   - `{{ term = "hear att" }}`
   - `{{ term = "**heart"* }}`

**Potential Examples**

- `<< 64572001 | Disease| {} |term = "heart"]
  and << 64572001 | Disease| {} |term = "hjärta", languageCode = "sv"]`
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] OR << 64572001 | Disease| {} |term = "hjärta", languageCode = "sv"]`
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] AND << 64572001 | Disease| {} |term = "hjärta", languageCode = "sv"]`
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] OR (term = "hjärta", languageCode = "sv"]`)
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] OR (term = "hjärta", languageCode = "sv"]`)
- `<< 64572001 | Disease| {} |term = "cardio", languageCode = "sv"]`)

**Recommendation to be made on (based on investigation of grammar):**

- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] AND (term = "hjärta", languageCode = "sv"])`
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] OR (term = "hjärta", languageCode = "sv"])`
- `<< 64572001 | Disease| {} |term = "heart", languageCode = "en"] MINUS (term = "hjärta", languageCode = "sv"])`

Use Cases
- Intentionally define a reference set for chronic disease. Starting point was ECL with modelling; This misses concepts modelled using the pattern you would expect. So important in building out that reference set.
- Authors quality assuring names of concepts
- Checking translations, retranslating. Queries for a concept that has one word in Swedish, another word in English
- AU use case would have at most 3 or 4 words in match
- Consistency of implementation in different terminology services
- Authoring use cases currently supported by description templates
- A set of the “ectomy”s and “itis”s

**Questions**

- Do we include 'typeId' - e.g. << 64572001 |Disease| {{ D.term = "heart", typeId = 90000000000013009 |Synonym| }}
  - NO
- Do we include 'type' - e.g. << 64572001 |Disease| {{ D.term = "heart", D.type = synonym }}
  - NO
- Do we include 'languageCode' - e.g. << 64572001 |Disease| {{ D.term = "heart", D.type = synonym, D. languageCode = "en" }}
  - YES
- Do we include 'caseSignificanceId' - e.g. << 64572001 |Disease| {{ D.term = "Heart", D.caseSignificanceId = 900000000000017005 |case sensitive| }}
  - NO
- Do we include 'caseSignificance' - e.g. << 64572001 |Disease| {{ D.term = "Heart", D.caseSignificance = sensitive }}
  - NO
- Do we include 'language' and 'version' - e.g. << 64572001 |Disease| {{ term = "heart" }} VERSION = http://… |LANGUAGE = (999001881000000108|Gastro LRS|, |GB English|)
  - NO
- Do we include syntactic sugar - e.g.
  - NO
- Packaging - How do we package this extension to ECL
  - A new version of ECL - e.g. 2.0 (or 1.47?) same specification document
  - An optional extension to ECL for SNOMED authors/content developers - e.g. ECL++ / ECLv1.3++ An appendix on the ECL document? (with Filter Language)
  - A subset/profile of the Query Language A separate document that defines SNOMED filters, which can be added to any version of ÉCL
  - Filter Language
Executing maps

<table>
<thead>
<tr>
<th>Linda Bird</th>
<th>Reverse memberOf function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What refsets is a concept a member of?</td>
</tr>
<tr>
<td></td>
<td>Proposed extension to ECL (same packaging questions as above) to support the execution of maps</td>
</tr>
<tr>
<td></td>
<td>Example use cases</td>
</tr>
<tr>
<td></td>
<td>• Mapping from international substance concepts to AMT substance concepts</td>
</tr>
<tr>
<td></td>
<td>• Anatomy structure and part association reference set - e.g. find the anatomical parts of a given structure</td>
</tr>
<tr>
<td></td>
<td>Potential syntax to consider</td>
</tr>
<tr>
<td></td>
<td>• Functional</td>
</tr>
</tbody>
</table>
|            | • mapTarget ((Anatomy structure and part association refset), << |Upper abdomen structure|)
|            | • Return the map targets from the given map refset, where the referencedComponent matches the condition |
|            | • mapSource ((Anatomy structure and part association refset), << |Liver part|)
|            | • Return the referencedComponent from the given map refset, where the targetId matches the condition |
|            | • Dot notation + Attribute refinement |
|            | • [Anatomy structure and part association refset] . [mapTarget] |
|            | • [Anatomy structure and part association refset] . [referencedComponent] (Same as ^ [Anatomy structure and part association refset]) |
|            | • [ (Anatomy structure and part association refset): [referencedComponent] = << |Upper abdomen structure] . [mapTarget] |
|            | • [(Anatomy structure and part association refset): [mapTarget] = << |Upper abdomen structure] . [referencedComponent] |
|            | • Specify value to be returned |
|            | • ?[mapTarget] ? [Anatomy structure and part association refset] |
|            | • ?[mapTarget] ? [Anatomy structure and part association refset] ( [referencedComponent] = << |Upper abdomen structure]) |
|            | • ?[mapTarget] ? [Anatomy structure and part association refset] : [referencedComponent] = << |Upper abdomen structure] |

Returning attributes

<table>
<thead>
<tr>
<th>Michael Lawley</th>
<th>Proposal from Michael:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Currently ECL expressions can match (return) concepts that are either the source or the target of a relationship triple (target is accessed via the 'reverse' notation or 'dot notation', but not the relationship type (ie attribute name) itself.</td>
</tr>
<tr>
<td></td>
<td>For example, I can write:</td>
</tr>
<tr>
<td></td>
<td>&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td>&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td>But I can't get all the attribute names that are used by &lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td>• Perhaps something like:</td>
</tr>
<tr>
<td></td>
<td>• ? R.type ? (&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td>• This could be extended to, for example, return different values - e.g.</td>
</tr>
<tr>
<td></td>
<td>• ??</td>
</tr>
</tbody>
</table>

URI Standard

| Linda Bird | Finalize and publish language and language instance URIs |
Examples: version and language

- `<< 64572001 |Disease| [{term = "*heart*"}] VERSION http://snomed.info/sct/900000000000207008 /version/20180131`
- `<< 64572001 |Disease| [{synonym = "*heart*"}] VERSION http://snomed.info/sct /900000000000207008/version/20180131`
- `<< 64572001 |Disease| [{FSN = "*heart*"}] VERSION http://snomed.info/sct/900000000000207008/ve
rion/20180131`
- `<< 64572001 |Disease| [{FSN = "*heart*"}] VERSION http://snomed.info/sct/900000000000207008/ve
rion/20180131, LANGUAGE W`
- `<< 64572001 |Disease| [{preferredTerm = "*heart*"}] VERSION http://snomed.info/sct /
900000000000207008/version/20180131, LANGUAGE Y`
- `<< 64572001 |Disease| [{acceptableTerm = "*heart*"}] VERSION http://snomed.info/sct /
900000000000207008/version/20180131, LANGUAGE Y`
- `"(* {term = "*heart*"}) VERSION http://snomed.info/sct/900000000000207008/version/20180131, LANGUAGE W"
- `X MINUS Y WHERE X = *, Y = (* [{term = "*heart*"})] VERSION http://snomed.info/sct /
900000000000207008/version/20180131, LANGUAGE W`

Notes

- Allow nested where, version, language
- Scope of variables is inner query

Examples: where

- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547)`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547) VERSION http://snomed.info/sct /
900000000000207008/version/20180131`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456) VERSION http://snomed.info/sct /
900000000000207008/version/20180131, LANGUAGE Y`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456), LANGUAGE 900000000000508004 [GB English]`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456), LANGUAGE 900000000000508004 [GB English]`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456), LANGUAGE 900000000000508004 [GB English]`
- `X MINUS 1 WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456), LANGUAGE 900000000000508004 [GB English]`

Notes

- Allow nested variable definitions, but recommend that people don't due to readability
- Scope of variables is the inner query
- No recursion e.g X WHERE X = 1234 MINUS X
  - i.e can't use a variable in its own definition
  - i.e X is only known on the left of the corresponding WHERE, and not on the right of the WHERE
Keywords for Term-based searching:

- **D.term**
  - D.term = "*heart*"
  - D.term = wild:"*heart*"
  - D.term = regex:"*heart*"
  - D.term = match:"hear att"
  - D.term = (sv) wild: "*heart*"

- **D.languageCode**
  - D.languageCode = "en"
  - D.languageCode = "es"

- **D.caseSignificanceId**
  - D.caseSignificanceId = 900000000000448009 [entire term case insensitive]
  - D.caseSignificanceId = 900000000000170005 [entire term case sensitive]
  - D.caseSignificanceId = 900000000000020002 [only initial character case insensitive]

- **D.caseSignificance**
  - D.caseSignificance = "insensitive"
  - D.caseSignificance = "sensitive"

- **D.typeId**
  - D.typeId = 900000000000003001 [fully specified name]
  - D.typeId = 900000000000013009 [synonym]
  - D.typeId = 900000000000550004 [definition]

- **D.type**
  - D.type = "FSN"
  - D.type = "fullySpecifiedName"
  - D.type = "synonym"
  - D.type = "textDefinition"

- **D.acceptabilityId**
  - D.acceptabilityId = 900000000000549004 [acceptable]
  - D.acceptabilityId = 900000000000548007 [preferred]

- **D.acceptability**
  - D.acceptability = "acceptable"
  - D.acceptability = "preferred"

**Additional Syntactic Sugar**

- **FSN**
  - FSN = "*heart"
    - D.term = "*heart", D.type = "FSN"
    - D.term = "*heart", D.typeId = 900000000000003001 [fully specified name]
    - FSN = "*heart" LANGUAGE X
    - D.term = "*heart", D.type = "FSN", D.acceptability = "LANGUAGE X"
    - D.typeId = 900000000000003001 [fully specified name], acceptabilityId = "LANGUAGE X"

- **synonym**
  - synonym = "*heart"
    - D.term = "*heart", D.type = "synonym"
    - D.typeId = 900000000000013009 [synonym]
    - synonym = "*heart" LANGUAGE X
    - D.typeId = 900000000000013009 [synonym], acceptabilityId = "LANGUAGE X"
    - D.typeId = "synonym", D.acceptability = "LANGUAGE X"
    - D.typeId = "synonym", (D.acceptabilityId = 900000000000549004 [acceptable] OR D.acceptabilityId = 900000000000548007 [preferred]) LANGUAGE X

- **synonymOrFSN**
  - synonymOrFSN = "*heart"
    - synonymOrFSN = "*heart" OR FSN = "*heart"
    - D.typeId = "synonym" OR D.type = "fullySpecifiedName"
    - synonymOrFSN = "*heart" LANGUAGE X
    - synonym = "*heart" OR FSN = "*heart" LANGUAGE X
    - D.typeId = "synonym" OR D.type = "fullySpecifiedName", D.acceptability = "LANGUAGE X"

- **textDefinition**
  - textDefinition = "*heart"
    - D.typeId = "definition"
    - textDefinition = "*heart" LANGUAGE X
    - D.typeId = "definition", D.acceptability = "LANGUAGE X"
    - D.typeId = "definition", D.acceptabilityId = "LANGUAGE X"

- **Unacceptable Terms**
  - (D.term = "*heart") MINUS (D.term = "*heart", D.acceptability = "LANGUAGE X")
Language preferences using multiple language reference sets

- LRSs that use the same Language tend to use 'Addition' - i.e. child LRS only includes additional acceptable terms, but can override the preferred term
  - E.g. Regional LRS that adds local dialect to a National LRS
  - E.g. Specialty-specific LRS
  - E.g. Irish LRS that adds local preferences to the en-GB LRS
- LRSs that define a translation to a different language tend to use 'Replacement' - i.e. child LRS replaces set of acceptable and preferred terms for any associated concept
  - E.g. Danish LRS that does a partial translation of the International Release
    - 999999 [Danish language reference set] ELSE [GB English reference set]

<table>
<thead>
<tr>
<th>Other topics</th>
<th>Linda Bird</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Any other topics?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Confirm next meeting date /time</th>
<th>Linda Bird</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLPG meeting will be suspended until September, due to a series of travel, holiday and work commitments.</td>
<td></td>
</tr>
</tbody>
</table>

---

**File**

Microsoft Excel Spreadsheet RegexCheat.xlsx

**Modified**

2019-Jun-04 by Linda Bird