2019-05-22 - SLPG Meeting

Date & Time
20:00 UTC Wednesday 22nd May 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 schedule for October business meeting
- 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: Michael Lawley, Daniel Karlsson, Ed Cheetham, Anne Randorf Højen, Peris Brodsky

Apologies

Agenda 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 Bird</td>
<td>- Question - Are there any conflicts in the current meeting schedule for October business meeting?</td>
</tr>
</tbody>
</table>
| Actions from last week       | Linda Bird | - Actions from last week:  
  - Additional use cases for term searching in ECL?  
  - Search types? Include regex? Does anyone have a regex ABNF?  
  - Language features required in ECL? |


### Syntax

\[
\text{term} = \{ \text{termSearchType: } \text{String} \}
\]

### Term Search Type

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

### Potential Examples

- \{\text{term = "heart"}
- \{\text{term = "hear att"}
- \{\text{term = "*heart*"}
- \{\text{term = "hear att"}
- \{\text{term = "*heart*"}

### 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 'typedep' - e.g. \{\text{D.term = "heart"}, typdep = 90000000000013009 [Synonym] \}
- Do we include 'type' - e.g. \{\text{D.term = "heart"}, D.type = synonym \}
- Do we include 'languageCode' - e.g. \{\text{D.term = "heart"}, D.type = synonym, D.languageCode = "en" \}
- Do we include 'caseSignificanceId' - e.g. \{\text{D.term = "heart"}, D.caseSignificanceId = 90000000000017005 [case sensitive] \}
- Do we include 'language' and 'version' - e.g. \{\text{D.term = "heart"} \}
- Do we include syntactic sugar - e.g.
  - \{\text{D.term = "heart"}, languageRefSet = en-gb \}
  - \{\text{D.term = "heart"}, languageRefSet = en-gb \}
- Do we use require the "D" at the start of "term"?
- How does this relate to description templates? How does this relate to SNOSTORM implementation?
- Packaging - Do we (over time) extend ECL with all the new Query Language features, and define a set of example ECL profiles - e.g. "Basic ECL", "ECL with basic term searching", "ECL with filters".

### Term Search Type

- **Wild Card Match (collation)** - e.g.
  - \{\text{term: wild: "heart"}
  - \{\text{term: wild (sv): "hjärta"}
- **Perl Regex** - e.g.
  - \{\text{term: regex: ".*heart.*"}
  - \{\text{term: wild (sv): "*hjärta"}
- **Word Prefix Any Order** - e.g.
  - \{\text{term: "heart"}
  - \{\text{term: "*heart*"}
- **Default (word prefix any order)** - e.g.
  - \{\text{term: "hear att"}
  - \{\text{term: "*heart*"}

### Potential Examples

- \{\text{term = "heart"}
- \{\text{term = "hear att"}
- \{\text{term = match: "hear att"}
- \{\text{term = regex: ".*heart.*"}
- \{\text{term = wild: "*hjärta"}
- \{\text{term = wild: "*heart*"}

### 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 'typedep' - e.g. \{\text{D.term = "heart"}, typdep = 90000000000013009 [Synonym] \}
- Do we include 'type' - e.g. \{\text{D.term = "heart"}, D.type = synonym \}
- Do we include 'languageCode' - e.g. \{\text{D.term = "heart"}, D.type = synonym, D.languageCode = "en" \}
- Do we include 'caseSignificanceId' - e.g. \{\text{D.term = "heart"}, D.caseSignificanceId = 90000000000017005 [case sensitive] \}
- Do we include 'language' and 'version' - e.g. \{\text{D.term = "heart"} \}
- Do we include syntactic sugar - e.g.
  - \{\text{D.term = "heart"}, languageRefSet = en-gb \}
  - \{\text{D.term = "heart"}, languageRefSet = en-gb \}
- Do we use require the "D" at the start of "term"?
- How does this relate to description templates? How does this relate to SNOSTORM implementation?
- Packaging - Do we (over time) extend ECL with all the new Query Language features, and define a set of example ECL profiles - e.g. "Basic ECL", "ECL with basic term searching", "ECL with filters".
<table>
<thead>
<tr>
<th>Executing maps</th>
<th>Linda Bird</th>
<th>Proposed extension to ECL to support the execution of maps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Example use cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mapping from international substance concepts to AMT substance concepts</td>
</tr>
<tr>
<td></td>
<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></td>
<td>• Potential syntax to consider</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mapTarget (Anatomy structure and part association refset), &lt;= Upper abdomen structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mapSource (Anatomy structure and part association refset), &lt;= Liver part</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dot notation + Attribute refinement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>•</td>
</tr>
</tbody>
</table>
|                |            |     • |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 |
|                |            |   • Dot notation + Filters                                 |
|                |            |     • ( |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| ( |referencedComponent| = <= Upper abdomen structure ) |
|                |            |   • ?mapTarget? |Anatomy structure and part association refset| : |referencedComponent| = <= Upper abdomen structure |

<table>
<thead>
<tr>
<th>Returning attributes</th>
<th>Michael Lawley</th>
<th>Proposal from Michael:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<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></td>
<td>For example, I can write:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<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></td>
<td>? R.type ? (&lt;&lt; 404684003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This could be extended to, for example, return different values - e.g.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>??</td>
</tr>
</tbody>
</table>

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

- `<http://snomed.info/sct/900000000000207008/20180131>` `<< 64572001 |Disease| {{ term = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131`
- `<http://snomed.info/sct/900000000000207008/20180131>` `<< 64572001 |Disease| {{ synonym = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131`
- `<http://snomed.info/sct/900000000000207008/20180131>, LANGUAGE W` `<< 64572001 |Disease| {{ FSN = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE W`
- `<http://snomed.info/sct/900000000000207008/20180131, LANGUAGE Y>` `<< 64572001 |Disease| {{ preferredTerm = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE Y`
- `<http://snomed.info/sct/900000000000207008/20180131, LANGUAGE Y>` `<< 64572001 |Disease| {{ acceptableTerm = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE Y`
- `<http://snomed.info/sct/900000000000207008/20180131, LANGUAGE W>` `{{ term = "heart*" }} VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE W`

Notes

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

Examples: where

- `X MINUS >! X WHERE X = (<< 1234 : 5678 = << 6547)`
- `X MINUS >! X WHERE X = (<< 1234 : 5678 = << 6547) VERSION` `http://snomed.info/sct/900000000000207008/20180131`
- `X MINUS >! Y WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456) VERSION` `http://snomed.info/sct/900000000000207008/20180131`
- `X MINUS >! X WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456) VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE 999001881000000108|GB clinical extension LRS|, LANGUAGE 900000000000508004|GB English|`
- `X MINUS >! X WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456) VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE 999001881000000108|GB clinical extension LRS|, LANGUAGE 900000000000508004|GB English|`
- `X MINUS >! X WHERE X = (<< 1234 : 5678 = << 6547), Y = (<< 1456) VERSION` `http://snomed.info/sct/900000000000207008/20180131, LANGUAGE 999001881000000108|GB clinical extension LRS|, 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. `WHERE` `X = 1234 MINUS X`
  - `ie can't use a variable in its own definition`
  - `ie 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 = 900000000000000000 [entire term case insensitive]
  - D.caseSignificanceId = 900000000000000001 [entire term case sensitive]
  - D.caseSignificanceId = 900000000000000002 [only initial character case insensitive]

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

- **D.typeId**
  - D.typeId = 900000000000000001 [fully specified name]
  - D.typeId = 900000000000000002 [synonym]
  - D.typeId = 900000000000000003 [definition]

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

- **D.acceptabilityId**
  - D.acceptabilityId = 900000000000000004 [acceptable]
  - D.acceptabilityId = 900000000000000007 [preferred]

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

Additional Syntactic Sugar

- **FSN**
  - FSN = "*heart"
  - D.term = "*heart", D.type = "FSN"
  - FSN = "*heart" LANGUAGE X
  - D.term = "*heart", D.type = "FSN", D.acceptability = " * LANGUAGE X"

- **synonym**
  - synonym = "*heart"
  - D.term = "*heart", D.type = "synonym"

- **synonymOrFSN**
  - synonymOrFSN = "*heart"

- **textDefinition**
  - textDefinition = "*heart"

- **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]

Other topics

- Linda Bird
  - Any other topics?

Confirm next meeting date /time

- Linda Bird
  - The next SLPG meeting will be held in 2 weeks at 20:00 UTC on Wednesday 5th June.