TMQL path language proposal

Leipzig, November 2009

The proposal

- This is a proposal for a replacement to the current TMQL draft path language
- It's based on the proposal posted on LMG's blog, but modified
  - http://www.garshol.priv.no/blog/205.html
- This is an informal presentation of the language in order to
  - judge what the community thinks of the proposal, and
  - get feedback on alternative design choices in the language

Topic references (issue 1362)

- We use the same syntax as CTM for this
  - ids are item identifiers
  - qnames are subject identifiers
  - same way to define prefixes
  - etc etc
- *Maybe* subject identifier and locator references as in CTM
- Not providing any way to refer to topics by name

Basics

- foo # a topic reference
- ... / axis::type [filter] # a navigation step
- ... / axis:: [filter] # same, without type filtering
- ... / type # shorthand navigation step

- lmg / email # all my email addresses
- lmg / occurrence::email # the same
- lmg / email @ private # email addresses in “private” scope
- lmg / email [. / scope:: = private] # fully expanded version

Some more examples

- Path expressions starting with “/” start from the topic map item
  - / person # all person topics
  - / default::person # full expansion
  - / employed-by # all employed-by associations
  - / person / email # all email addresses of all persons
  - / person [. / email] # all persons which have an email occ.

Alternative syntax

- Inge Henriksen points out that technically, only the first slash is needed
  - the first slash is needed so you know whether to start from the topic map or not
- The result would be
  - / person email
  - Img / employee association::employed-by employer *
- instead of
  - / person / email
  - Img / employee / association::employed-by / employer / *

Filters

- The [filter] contains a boolean expression
  - simple path expression: true if it produces at least one value
    - / person [. / email] really means / person [exists(. / email)]
  - comparison expression: <, >, <=, >=, !=, =
  - AND, OR, NOT
- NOT
  - / person [not . / email] # one possible syntax (not is operator)
  - / person [not(. / email)] # another (not is function)
- . /
  - lets us distinguish topic references from path navigation steps
  - / person [. / start-date = . / end-date]
  - / person [. / employed-by(employee -> employer) = tmlab]
  - / person [not(email)] # true if email topic exists
  - / person [not(. / email)] # true if person has email

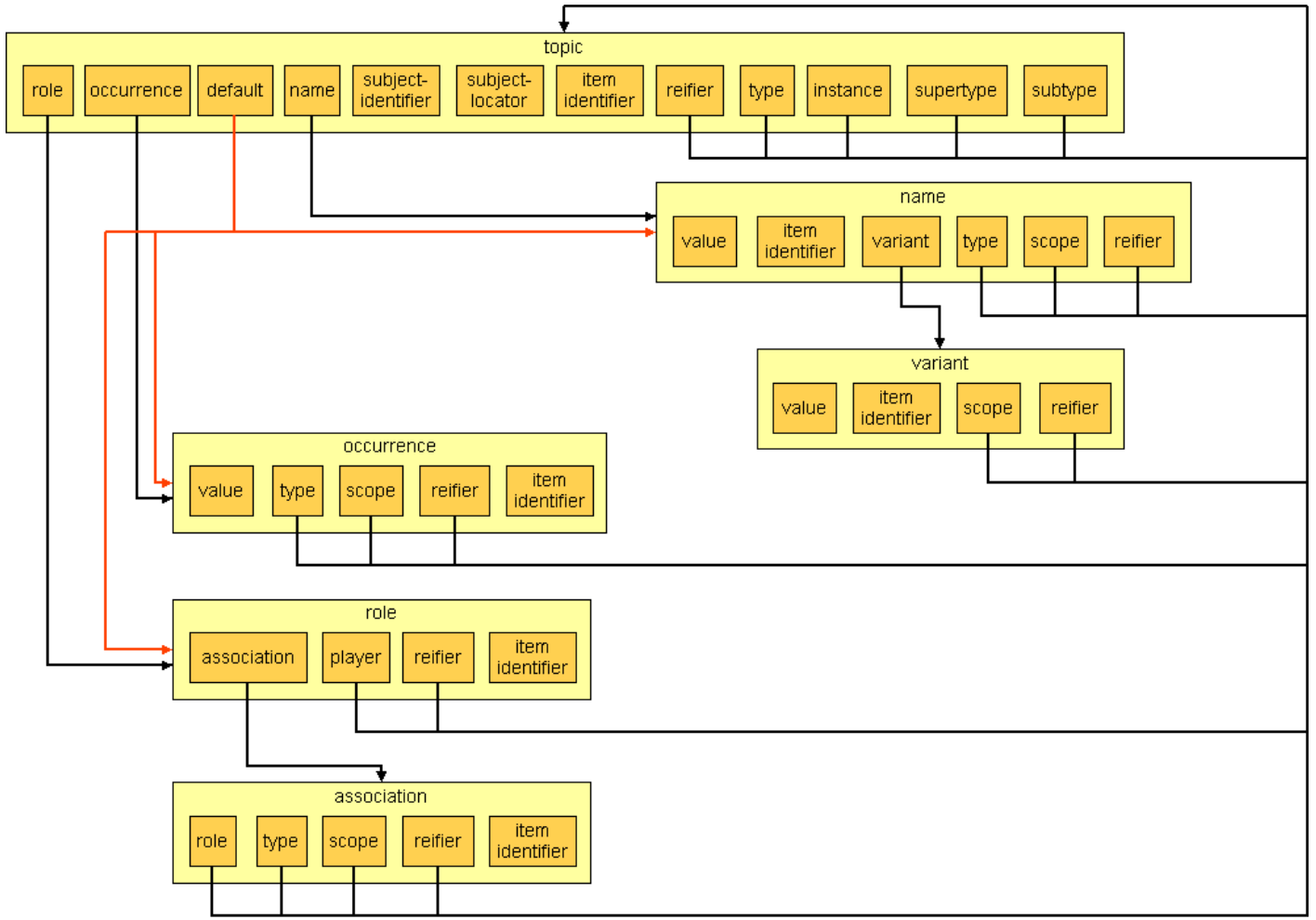
The axis syntax

- We are not 100% satisfied with the axis::type notation
- Problems with it
  - lmg / occurrence::email # fine
  - lmg / occurrence:: # all occurrences of any type
  - lmg / subject-identifier:: # not as pretty, perhaps?
  - $c / occurrence::tmcl:card-min # lots of colons there...
- Alternative #1
  - lmg / occurrence::email
  - lmg / occurrence # ambiguous, unfortunately
- Alternative #2
  - lmg / occurrence(email) # looks like function call. problem?
  - lmg / occurrence() # of any type
  - lmg / subject-identifier()
  - $c / occurrence(tmcl:card-min) # no colon collision any more

The axes

- topic
  - default
  - name
  - occurrence
  - role
  - subject-identifier
  - subject-locator
  - item-identifier
  - reified
  - type
  - instance
  - supertype
  - subtype
- association
  - role
  - type
  - scope
  - reifier
  - item-identifier
- role
  - type
  - player
  - association
- name
  - type
  - value
  - ...

The axes


topic
role
occurrence
default
name
subject-identifier
subject-locator
item identifier
reifier
type
instance
supertype
subtype
name
value
item identifier
variant
type
scope
reifier
variant
value
item identifier
scope
reifier
occurrence
value
type
scope
reifier
item identifier
role
association
player
reifier
item identifier
association
role
type
scope
reifier
item identifier

Association traversal

- Somewhat cumbersome
  - lmg / employee / association::employed-by / employer / *
- Built-in operator for traversal
  - lmg / employed-by(employee -> employer)
  - handles symmetric associations automatically
- Supports additional constraints
  - lmg / employed-by(employee -> employer) @ past
  - lmg / employed-by(employee -> employer) [. / reifier:: / start-date < “2000-01-01”]

Constraints on roles

- Supported via filters
  - $company / represented-by(represented -> representative) [. / at = tmra]
- However, one could also reuse association predicate syntax:
  - $company / represented-by(represented -> representative, at : tmra)
  - benefits
    - makes association traversal look like association predicates (familiar)
    - shorter
  - disadvantages
    - makes association traversal look like association predicates (which they are not)
    - bigger language

Variables in predicates

- If we reuse the predicate syntax, what about
  - $company / represented-by(represented -> representative, at : $event)
  - $event is unbound
- Possible answers
  - this is confusing, so we shouldn't use this syntax
  - path expressions can't bind variable values, so this is an error
  - path expressions *can* bind variable values, so this is OK
- Note that the last answer raises a bigger issue
  - should it be possible to use path expressions on their own?
  - should path expressions return just a set, or a more complex result?

Allowing any type

- Original draft had *
  - $person / occurrence::* # all occurrences of person
- We now use a blank
  - $person / occurrence:: # same

Boolean and set operators

- Booleans
  - / person or dog # not allowed
  - / topic:: [. / type = person or . / type = dog] # allowed
- Set operators
  - / person UNION / dog # would have worked
  - however, we don’t allow set operators in the path language
  - these are left for SELECT/FLWR statements

Taxonomometry

- Given
  - lmg isa person . person ako creature . creature ako subject .
  - / creature # returns lmg
- The problem is querying for the types
  - lmg / type:: # returns person, creature, subject
  - person / supertypes:: # returns creature, subject
- How to get only what is said explicitly in the topic map?
  - lmg / tm:type-instance(tm:instance -> tm:type) # do it explicitly
  - lmg / direct-type:: # add extra axis
  - # requires direct-type, direct-instance, direct-supertype & direct-subtype
- A more general proposal
  - lmg / type(0) # lmg
  - lmg / type() # person
  - lmg / type(1) # person
  - lmg / type(1) / supertype(1) # creature
  - lmg / type(1) / supertype(0..*) # person, creature, subject

Taxonometry (2)

- Rani says:
  - “I don't like that ‘type’ refer both to types and tosupertypes. This seems not clean to me.”
- Proposed solution
  - Img / parent-type() # person, creature, subject
  - Img / type() # person

Functions

- concat(str, str)
- starts-with(str, str)
- ends-with(str, str)
- contains(str, str)
- substring-before(str, str)
- substring-after(str, str)
- substring(str, int, int?)
- string-length(str)
- normalize-space(str)
- translate(str, str, str)
- find(str, str)
- ceiling(float)
- floor(float)
- round(float)
- count(resultset)

Functions and sets

- func(pathexpr) # passes result set to function
  - count(/ person) # counts person topics in TM
  - string-length(/ person / email / value) # gets length of *one* email
- Not possible to do make a list of the lengths of all emails
  - in the path language, that is!
  - in SELECT statements this *is* possible

Type conversion

- Implicit in comparisons
  - / person [. / email = “larsga@bouvet.no”] # occurrence converted
- Implicit in function calls
  - / person [string-length(. / email) > 20] # occurrence converted
- Explicit elsewhere
  - string(lmg) # converts ‘lmg’ topic to string
- Note that if we do this we can drop the value axis
  - or should we keep it for completeness?

Slicing

- In the existing TMQL draft slicing is supported in two ways
  - / foo / bar [1]
  - / foo / bar [1 .. *]
  - select ... offset 50 limit 25
- Inge Henriksen suggests we should preserve this feature

Tuple constructors

- In the existing TMQL draft path expressions can produce tabular results, as follows:
  - / person (. / name, . / email)
- This would produce a set of (name, email) pairs
- This has been deliberately omitted in the current proposal