

Topic Map Relational Query Language - TMRQL

Graham Moore
Kal Ahmed

Copyright © 2005 Networked Planet Limited

Table of Contents

| | |
|--|----|
| Introduction | 2 |
| Conventions Used | 2 |
| The Relational Views | 3 |
| tm_assoc | 3 |
| tm_assoc2 | 3 |
| tm_assocScope | 4 |
| tm_directInstanceOf | 4 |
| tm_name | 4 |
| tm_nameScope | 5 |
| tm_nameValue | 5 |
| tm_nameVariant | 5 |
| tm_nameVariantResource | 6 |
| tm_nameVariantValue | 6 |
| tm_occur | 6 |
| tm_occurResource | 7 |
| tm_occurScope | 7 |
| tm_occurValue | 7 |
| tm_si | 8 |
| tm_sl | 8 |
| tm_topic | 8 |
| tm_src | 8 |
| tm_src | 9 |
| tm_variantScope | 9 |
| Built in Functions | 9 |
| tm_displayName | 9 |
| tm_instanceOf | 10 |
| tm_sortName | 10 |
| tm_subClasses | 11 |
| tm_superClasses | 11 |
| tm_tc (Transitive Closure) | 12 |
| TMDM Relationship | 12 |
| SQL | 12 |
| Benefits of this approach | 12 |
| Developer Familiarity | 13 |
| Sound Conceptual Foundation | 13 |
| Flexibility in Outputs | 13 |
| Prevents repeating previous mistakes | 13 |
| Increases Adoption and Use | 13 |
| Many query utilities and semantics already defined | 13 |
| Weaknesses Of This Approach | 13 |
| Complex to implement | 13 |
| Inconsistencies in SQL support | 14 |
| Conclusion | 14 |
| Appendix A - TMQL Use Case Queries | 14 |

Introduction

Topic Map Relational Query Language (TMRQL) has been designed in order to provide a sound foundation for querying topics maps. To this end it does not define an entire new language but instead presents a core set of abstract relational views.

The relational model provides a firm foundation for the development of a topic map query language. Development in this direction would lead to a more accessible and usable language by a greater number of developers than a new and bespoke language. Developers would be familiar with the concepts and their existing tools would work with the data structures returned. To them, the topic map data model would appear as just another schema or view. In order that the TMRQL language is not bound to a single implementation schema, nor even, bound to a relational database implementation we define a set of Relational Views that provide an abstract relational model of the topic map data model. This abstract data structure is independent of any particular implementation yet provides a foundation to use the full power of the SQL language and helps with portability of TMRQL queries. This abstract model can be used by in-memory implementations, small database applications and large corporate solutions. The relational views can be seen to be roughly equivalent to the predicate structures in Tolog or the XPath abstractions in TMAPath. We present a similar approach using the relational model. The fact that both of these approaches have gained some traction indicates that the principle is sound, but the TMRQL approach takes the concept to the widest possible audience in terms of current query technology. In this paper we describe the abstract relational views and how these views can be used by a full or partial SQL implementation. We conclude by discussing the core benefits of this approach including how developers benefit and how the different semantics required in a query language are provided without having to re-develop them.

Conventions Used

This document uses a number of typographical conventions to highlight particular types of words or sections of text.

- Namespace, class and interface names are all displayed in a monospaced font. e.g `NetworkedPlanet.TMAPI`, `ITopic`.
- Method names are displayed in a monospaced font. When referring to a particular overload of a method, the method parameters are displayed, otherwise they are omitted. The first time a method is referred to in the text, the method name is qualified by the interface or class that it is a member of e.g. `ITopic.CreateTopicName()`
- File names are shown in bold monospaced font e.g. **`App.config`**

In addition the following conventions are used to highlight sections of the text:

Note

This is a note. A note highlights or expands on something mentioned in the text.

Warning

This is a warning. A warning alerts you to a critical piece of information.

```
// This is a program listing
// It contains a code snippet which is relevant to the text
// Lines are numbered for references from the text
```

This is computer output
It shows the text output that results

from running a command-line application.

The Relational Views

The following relational views provide an abstract and normalised view of the topic map data model. The views are optimised to support searching by for example already having join roles with associations and occurrences with scope etc. As these are conceptual views applications are not required to implement them directly. These views are proposed as a basic conceptual building block such that when writing queries these view should be present and usable. The intended implementation platform is of course SQL but there is nothing preventing other higher level language utilising these tables. An example would be an XML database exposing these views of topic map data stored as XML.

One of the inherent principles that is embodied in the views approach is that every topic map object (topic, name, occurrence etc) has some kind of system identity. The important property is touched upon in the Topic Maps Data Model (TMDM) but in general it is something that has not been utilised to its full extent. There is no definition here of what this system identity must be. It could be the XML DOM Node object identity, an id in a database column, or an identifier assigned programmatically by the topic map processor. The topic map views approach brings this system id to the fore as the unique unambiguous identity of constructs within the system.

tm_assoc

Binds an association with the topic that defines the association type, and the type and role player of a single role in that association. Useful when querying for unary topic associations.

Table 1. tm_assoc view

| Column | Type | Description |
|---------------------|------|---|
| topicmap | int | The ID of the containing topic map |
| association_id | int | The ID of the association |
| association_type | int | The ID of the topic that types the association |
| association_role_id | int | The ID of the association role |
| r1p | int | The ID of a role player in the association |
| r1t | int | The ID of the topic that types the role played by the role player |

tm_assoc2

Binds an association with the topic that defines the association type, and the type and role player of two separate roles in that association. An association with two roles is represented as two rows in this view. This view displays all possible combinations of two different association roles from all associations. Associations with only one association role will not appear in this view at all. Associations with N roles will be represented by $(N-1)2$ rows representing all possible combinations of two roles in that association. The result of this redundancy is that it is possible to query for an association by role player or role type without concern for whether it appears as the first or second role in this table (as it will appear as both).

Table 2. tm_assoc2 view

| Column | Type | Description |
|--------|------|-------------|
|--------|------|-------------|

| | | |
|------------------|-----|--|
| topicmap | int | The ID of the containing topic map |
| association_id | int | The ID of the association |
| association_type | int | The ID of the topic that types the association |
| r1_id | int | The ID of the first association role |
| r1p | int | The ID of the first role player in the association |
| r1t | int | The ID of the topic that types the role played by the first role player |
| r2_id | int | The ID of the second association role |
| r2p | int | The ID of the second role player in the association |
| r2t | int | The ID of the topic that types the role played by the second role player |

tm_assocScope

Binds an association with each of the topics in its scope. This view contains one row for each topic in the scope of each association in the system. An association which is unscoped will not appear in this view at all. An association with N topics in its scope will appear in the view N times (once for each scoping topic).

Table 3. tm_assocScope view

| Column | Type | Description |
|------------------|------|--|
| topicmap | int | The ID of the containing topic map |
| assoc_id | int | The ID of the association |
| scoping_topic_id | int | The ID of a topic which is in the scope of the association |

tm_directInstanceOf

Binds a topic with the topic(s) which define its type. This view only presents the direct types of the topic. To query topic types taking the XTM-defined superclass-subclass association into account the function tm_instanceOf is also defined.

Table 4. tm_directInstanceOf view

| Column | Type | Description |
|----------|------|---|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the topic that is the instance |
| type_id | int | The ID of the topic that types the instance |

tm_name

Binds a topic with its names. This view contains one row for each topic name in the system and provides access to the container hierarchy of the topic name. To extract the string value of the name, use the view tm_nameValue.

Table 5. tm_name view

| Column | Type | Description |
|----------|------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| name_id | int | The ID of the topic name |

tm_nameScope

Binds a topic name with the topics in the scope of the topic name. This view contains one row for each topic in the scope of each topic name in the system. A topic name which is unscoped will not appear in this view at all. A topic name with N topics in its scope will appear in the view N times (once for each scoping topic).

Table 6. tm_nameScope view

| Column | Type | Description |
|------------------|------|--|
| name_id | int | The ID of the topic name |
| scoping_topic_id | int | The ID of a topic in the scope of the topic name |

tm_nameValue

Binds a topic with its names and their string values. This view contains one row for each topic name in the system and provides access to the container hierarchy for the topic name and the string value of the topic name.

Table 7. tm_nameValue view

| Column | Type | Description |
|------------|---------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| name_id | int | The ID of the topic name |
| name_value | varchar | The string value of the topic name |

tm_nameVariant

Binds a topic with its names and name variants. This view contains one row for each variant name in the system. The columns of the view provide access to the complete container hierarchy for the variant name. To query variants by their value, use either the view tm_nameVariantValue to query string-valued variants or tm_nameVariantResource to query locator-valued variants.

Table 8. tm_nameVariant view

| Column | Type | Description |
|----------|------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |

| | | |
|------------|-----|--------------------------|
| name_id | int | The ID of the topic name |
| variant_id | int | The ID of the variant |

tm_nameVariantResource

Binds a topic with a name, a variant of that name and the address pointed to by the variant. This view contains one row for each variant name in the system with a non-null locator value. The columns of the view provide access to the complete container hierarchy for the variant name and to the locator value of the variant name. To query variants by their string value use the view tm_nameVariantValue.

Table 9. tm_nameVariantResource view

| Column | Type | Description |
|------------------|-------|---|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| name_id | int | The ID of the name |
| variant_id | int | The ID of the variant |
| variant_resource | vchar | The address of the locator value of the variant |

tm_nameVariantValue

Binds a topic with a name, a variant of that name and the string value of the variant. This view contains one row for each variant name in the system with a non-null string value. The columns of the view provide access to the complete container hierarchy for the variant name and to the string value of the variant name. To query variants by their locator value use the view tm_nameVariantResource.

Table 10. tm_nameVariantValue view

| Column | Type | Description |
|---------------|-------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| name_id | int | The ID of the name |
| variant_id | int | The ID of the variant |
| variant_value | vchar | The string value of the variant |

tm_occur

Binds a topic with each of its occurrences and the occurrence type only. This view contains one row for each occurrence in the system. The view provides access to the container hierarchy for the occurrence and to the type of the occurrence. To access the string or locator value of the occurrence, use either tm_occurValue for string-valued occurrences or tm_occurResource for locator-valued occurrences.

Table 11. tm_occur view

| Column | Type | Description |
|----------|------|------------------------------------|
| topicmap | int | The ID of the containing topic map |

| | | |
|------------|-----|---|
| topic_id | int | The ID of the containing topic |
| occur_id | int | The ID of the occurrence |
| occur_type | int | The ID of the topic that types the occurrence |

tm_occurResource

Binds a topic with an occurrence, the occurrence type and the address of the resource referenced by the occurrence. This view contains one row for each occurrence in the system with a non-null resource reference locator. The view provides access to the container hierarchy of the occurrence, the occurrence type and the address pointed to by the locator value of the occurrence. To query occurrences with a string value use the view tm_occurValue.

Table 12. tm_occurResource view

| Column | Type | Description |
|----------------|---------|---|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| occur_id | int | The ID of the occurrence |
| occur_type | int | The ID of the topic that types the occurrence |
| occur_resource | varchar | The address of the occurrence locator value |

tm_occurScope

Binds an occurrence with the topics in the scope of the occurrence. This view contains one row for each topic in the scope of each occurrence in the system. An occurrence which is unscoped will not appear in this view at all. An occurrence with N topics in its scope will appear in the view N times (once for each scoping topic).

Table 13. tm_occurScope view

| Column | Type | Description |
|------------------|------|--|
| occur_id | int | The ID of the occurrence |
| scoping_topic_id | int | The ID of a topic in the scope of the occurrence |

tm_occurValue

Binds a topic with an occurrence, the occurrence type and the string value of the occurrence. This view contains one row for each occurrence in the system with a non-null string value. The view provides access to the container hierarchy for the occurrence, the type of the occurrence and the string value of the occurrence. To query occurrences with a locator value use the view tm_occurResource.

Table 14. tm_occurResource view

| Column | Type | Description |
|------------|------|---|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the containing topic |
| occur_id | int | The ID of the occurrence |
| occur_type | int | The ID of the topic that types the occurrence |

| | | |
|-------------|---------|------------------------------------|
| occur_value | varchar | The string value of the occurrence |
|-------------|---------|------------------------------------|

tm_si

Binds a topic to its subject identifiers. This view contains one row for each subject identifier in each separate topic map in the system. Because this view spans multiple topic maps, it is possible for the view to contain multiple rows with the same subj_id column value, but the combination of subj_id and topicmap must be unique due to topic map merging constraints.

Table 15. tm_si view

| Column | Type | Description |
|----------|---------|---------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the topic |
| subj_id | varchar | The subject identifier address string |

tm_sl

Binds a topic to its subject locators. This view contains one row for each topic subject locator in each separate topic map in the system. Because this view spans multiple topic maps, it is possible for the view to contain multiple rows with the same subj_loc column value, but the combination of subj_loc and topicmap must be unique due to topic map merging constraints.

Table 16. tm_sl view

| Column | Type | Description |
|----------|---------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the topic |
| subj_loc | varchar | The subject locator address string |

tm_topic

Binds a topic with the topics that define its types. This view contains one row for each type of each topic in the TMCORE System. Untyped topics are represented by a single row with the type_id column set to NULL.

Table 17. tm_topic view

| Column | Type | Description |
|----------|------|------------------------------------|
| topicmap | int | The ID of the containing topic map |
| topic_id | int | The ID of the topic |
| type_id | int | The ID of the topic type |

tm_tsrc

Binds a topic to its source locators. This view contains one row for each topic source locator in each separate topic map in the system. Because this view spans multiple topic maps, it is possible for the

view to contain multiple rows with the same `src_loc` column value, but the combination of `src_loc` and `topicmap` must be unique due to topic map merging constraints.

Table 18. tm_tsrc view

| Column | Type | Description |
|-----------------------|----------------------|------------------------------------|
| <code>topicmap</code> | <code>int</code> | The ID of the containing topic map |
| <code>topic_id</code> | <code>int</code> | The ID of the topic |
| <code>src_loc</code> | <code>varchar</code> | The source locator address string |

tm_src

Binds a topic map object to its source locators. This view contains one row for each topic map object source locator in each separate topic map in the system. Because this view spans multiple topic maps, it is possible for the view to contain multiple rows with the same `src_loc` column value, but the combination of `src_loc` and `topicmap` must be unique due to topic map merging constraints.

Table 19. tm_src view

| Column | Type | Description |
|--------------------------|----------------------|--|
| <code>topicmap</code> | <code>int</code> | The ID of the containing topic map |
| <code>tmo_id</code> | <code>int</code> | The ID of the topic map object |
| <code>src_loc</code> | <code>varchar</code> | The source locator address string |
| <code>object_type</code> | <code>char</code> | A single character object type code. The following codes are used: A = Association R = Association Role N = Topic Name O = Occurrence T = Topic M = Topic Map V = Variant Name |

tm_variantScope

Binds a variant name with the topics in the scope of the variant name. This view contains one row for each topic in the scope of each variant name in the system. A variant name which is unscoped will not appear in this view at all. A variant name with N topics in its scope will appear in the view N times (once for each scoping topic).

Table 20. tm_variantScope view

| Column | Type | Description |
|-------------------------------|------------------|--|
| <code>variant_id</code> | <code>int</code> | The id of the variant name |
| <code>scoping_topic_id</code> | <code>int</code> | The id of a topic in the scope of the variant name |

Built in Functions

In order to assist query writers we provide some basic topic map query operations, such as `transitive closure`, and `displayname`. Some functions are pure utility returning a suitable string for a name, others return tables that can be further joined with other `tm_<views>`.

tm_displayName

Returns a display string for a topic selected from the topics variant names and topic names.

This function applies the following algorithm:

1. If the topic has a variant name with a scope including a topic with the subject identifier `http://www.topicmaps.org/xtm/1.0/core.xtm#display`, return the string value of that variant name.
2. If the topic has a topic name with a scope including a topic with the subject identifier `http://www.topicmaps.org/xtm/1.0/core.xtm#display`, return the string value of that topic name.
3. Return the string value of a topic name of the topic.

If any of those steps match multiple names, then the most recently added name is returned. If none of the above steps match any names, NULL is returned.

Table 21. tm_displayName Function

| Parameter | Type | Description |
|-----------|--------|---------------------|
| topic_id | int | The id of the topic |
| return | string | The display name |

tm_instanceOf

Binds a topic to the classes that it is an instance of. This function observes the XTM standard super-class/subclass association.

If the topic_id parameter is not NULL, and the class_id parameter is NULL, then this method returns all direct types and supertypes of the direct types of the topic_id. If the topic_id parameter is NULL and the class_id parameter is not NULL, then this method returns all instances of class_id and all instances of all subclasses of class_id. If both topic_id and class_id are not NULL, this method returns a single row containing topic_id and class_id if and only if class_id is a direct type of topic_id or is a supertype of a direct type of topic_id. If both topic_id and class_id are NULL, this method calculates all direct types and superclasses for all topics in the topic map identified by topicmap. In this mode, the operation has the potential to be highly resource-intensive and so use should be avoided where possible.

Table 22. tm_instanceOf Function

| Parameter | Type | Description | | |
|-----------|-------|------------------------------|------|--------------------------|
| topicmap | int | The id of the topicmap | | |
| topic_id | int | The id of the instance topic | | |
| class_id | int | The id of the class | | |
| return | table | Column | Type | Desc |
| | | topic_id | int | The id of instance topic |
| | | class_id | int | The id of class topic |

tm_sortName

Returns a sort string for a topic selected from the topics variant names and topic names.

This function applies the following algorithm:

1. If the topic has a variant name with a scope including a topic with the subject identifier <http://www.topicmaps.org/xtm/1.0/core.xtm#sort>, return the string value of that variant name.
2. If the topic has a topic name with a scope including a topic with the subject identifier <http://www.topicmaps.org/xtm/1.0/core.xtm#sort>, return the string value of that topic name.
3. Return the string value of a topic name of the topic.

If any of those steps match multiple names, then the most recently added name is returned. If none of the above steps match any names, NULL is returned.

Table 23. tm_sortName Function

| Parameter | Type | Description |
|-----------|--------|---------------------|
| topic_id | int | The id of the topic |
| return | string | The sort name |

tm_subClasses

Binds a topic representing a class to the topics that represent all subclasses of that class. This function uses the standard XTM-defined superclass-subclass association and association role types to return all subclasses of the topic identified by the parameter cls. The return from this function is a table with a single column listing the ID of each topic which is either a direct subclass of cls (i.e. plays the role 'subclass' in a 'superclass-subclass' association with cls playing the role 'superclass') or an indirect subclass (i.e. It is a subclass of a subclass of cls).

Table 24. tm_subClasses Function

| Parameter | Type | Description | | |
|-----------|-------|--|------|--|
| topicmap | int | The id of the topicmap | | |
| class_id | int | The base class to find all subclasses of | | |
| return | table | Column | Type | Desc |
| | | subclass | int | the ID of the topic that is a direct or indirect subclass of @class_id |

tm_superClasses

Binds a topic representing a class to the topics that represent all subclasses of that class. This function uses the standard XTM-defined superclass-subclass association and association role types to return all superclasses of the topic identified by the parameter cls. The return from this function is a table with a single column listing the ID of each topic which is either a direct superclass of cls or an indirect superclass (i.e. It is a superclass of a superclass of cls).

Table 25. tm_superClasses Function

| Parameter | Type | Description |
|-----------|------|--|
| topicmap | int | The id of the topicmap |
| class_id | int | The base class to find all subclasses of |

| | | | | |
|--------|-------|------------|------|---|
| return | table | Column | Type | Desc |
| | | superclass | int | the ID of the topics that are a direct or indirect superclass of @class_id. |

tm_tc (Transitive Closure)

Calculates and returns the transitive closure formed by walking an association from a starting point traversing specific types of association and association role. A transitive closure is computed as a set of paths starting with START. Each step in the path is found by traversing an association role of type FROMROLETYPE to an association of type ASSOCTYPE and traversing an association role of type TOROLETYPE to another topic. Each row in the return table represents one path in the transitive closure. Only the starting point and the end point of the path are included in the results - intermediate topics are omitted. This function is particularly useful for traversing hierarchies of related topics.

Table 26. tm_tc Function

| Parameter | Type | Description | | |
|----------------|-------|--|------|--|
| start | int | The ID of the topic to start the transitive closure from | | |
| assoc_type | int | The ID of the topic defining the type of association to be traversed. | | |
| from_role_type | int | The ID of the topic defining the type of association role to be walked *from* the edge of the transitive closure. | | |
| to_role_type | int | The ID of the topic defining the type of the association role to be walked *to* across the edge of the transitive closure. | | |
| return | table | Column | Type | Desc |
| | | start_id | int | the id of the start topic in the transitive closure path |
| | | end_id | int | the id of the end topic in the transitive closure path |

TMDM Relationship

The Topic Map Data Model presents a infoset and UML representation of the underlying data model of topic maps. The views presented here present the TM as a set of relationships between entities with identity. All of the properties and relations defined in the TMDM can be found in the views defined. This ensures that all aspects of the topic maps model can be queried using this approach.

SQL

It seems hardly worth saying but the reason for creating this set of relational views is so that they can be used by the SQL query language. The views and functions provide a level of abstraction away from any specific implementation and thus aid interoperability of queries. The real interoperability issue is with compatible SQL versions. The basic views presented here do not suppose any extensions or additional query capabilities than those defined in SQL-92.

Benefits of this approach

Developer Familiarity

Developers are familiar with SQL as a language and the concepts of views and functions. Developers are also well tuned to using a variety of tools that work with SQL databases and relational data structures. Using `tm_<views>` in conjunction with SQL as a query language provides developers with a familiar environment without compromising the flexibility and power that should come from querying topic maps.

Sound Conceptual Foundation

Using views, or the relational model in conjunction with SQL provides a solid conceptual foundation that has been researched and developed over many years. Implementations of SQL have been refined and improved as lessons have been learnt to continually improve performance. Basing the language on this technology means that as these other standards evolve that TMRQL will benefit.

Flexibility in Outputs

One of the requirements of the TMRQL use cases are the production of XML and XTM output. Most SQL implementations now provide native operations to support such activities and it is felt that defining the exact mechanism for how this should be achieved goes beyond the basic requirements of TMRQL. XML operations will become standardised and as they do so TMRQL can make recommendations on how best to use those features. In addition, producing XML results from relational data is something that can easily be accomplished by even the most novice developer who has SQL and XML experience.

Prevents repeating previous mistakes

The development of a query language is a non-trivial undertaking and creating a new language from scratch or one that borrows from ideas that have not been proven in the large puts at risk the adoption and success of a standard. Topic Maps should look to embrace the existing standards in order to maximise adoption and support. In addition, using mature technology to support things such as searching means that the topic map community benefits from all the work done by the existing database community.

Increases Adoption and Use

New standards take time to be adopted and one of the major factors in adoption is the tool support that is available and the skill set required to use the tools. Creating a new query language with a new syntax means that users will have to wait while early technology producers create first implementations. The number of software vendors will be small and their software largely untested. By having a language based on existing standards the bar to implementation is much lower and companies already have trained staff that can use the technology.

Many query utilities and semantics already defined

SQL already defines many functions and utilities that would need to be defined in a new language, things such as GROUP BY, ORDER etc. Redefining these semantics in ways that are obvious to users from other contexts is time consuming for the standards body, implementors and those using the language. These features have been developed and standardised as a matter of course as SQL has evolved - why do it all again.

Weaknesses Of This Approach

Complex to implement

Full support for SQL-92 is not a trivial coding task. However we believe that there are three important points to consider in relation to this objection.

1. The majority of scaleable implementations of the TMDM are currently implemented on relational database technology. The implementation of TMRQL for these systems would be trivial.
2. Although we recommend the use of SQL-92 as a query language, we feel that the primary contribution of TMRQL is the definition of the *views* that the query language operates on. These views could be mapped onto the underlying storage mechanism and accessed by native query languages. For example in an XML database, the views could be mapped to indexes or to separate XML documents that could then be queried using XQuery or the query language of the underlying database system.

By treating TMRQL as operations on relational views, it would also be possible for the TMQL standard to define a subset of SQL as the required operations for minimal conformance.

3. All query languages are non-trivial to implement. Regardless of which language is chosen as the ISO TMQL, it is likely to require some significant research and developer effort to implement effectively.

Inconsistencies in SQL support

In the appendix to this article we show the use of SQL to query the TMRQL relational views in order to address the use cases provided by the TMQL working group. These queries were written and executed against Microsoft SQL Server. It is well-known that support for SQL constructs varies between database implementations and that even databases with full SQL-92 support will provide additional non-portable features which make it more difficult to write completely portable queries.

We believe that by recommending that developer's restrict themselves to standard SQL-92 constructs (or by choosing a reasonable subset), the question of query portability can be addressed.

Conclusion

This approach shows how a core set of relational views can provide a solid basis for a topic map query language based on SQL. The basic argument for this as an approach to topic map querying is that relational technology and theory has matured over many years and many developers are familiar with the concepts. We believe it is a risk to topic map adoption to attempt to create a query language from scratch. The only advantage we see of a new and bespoke language is that the syntax could be made more elegant but the sample queries in the appendix illustrate that SQL topic map queries are concise and readable.

Appendix A - TMQL Use Case Queries

```

-----
-- TMQL Queries
-----

-- GENERAL NOTE:
-- These implementations return full source locators where the requirements
-- ask for identifiers as ids are not maintained by the TMDM!

-- First some initialisation.
-- NOTE: These are simply used as a convenient short cut for a
-- join / subselect against the tm_tsrc view, and could be replaced
-- with either an in-line join / subselect or with a function call.
DECLARE @tmid int
DECLARE @author int
DECLARE @en int
DECLARE @de int
DECLARE @tutorial int
DECLARE @person int
DECLARE @authorof int

```

```
DECLARE @opus int
DECLARE @pubdate int
DECLARE @document int
DECLARE @subclasses int
DECLARE @superclass int
DECLARE @subclass int
DECLARE @download int
DECLARE @email int
DECLARE @abstract int
DECLARE @language int
DECLARE @paper int
DECLARE @pepper int
DECLARE @influencedby int
DECLARE @influence int
DECLARE @influenced int

SET @tmid = (SELECT ID FROM topicmap WHERE MAPNAME='tmql')
SET @author = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#author')
SET @en = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#en')
SET @de = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#de')
SET @tutorial = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#tutorial')
SET @person = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#person')
set @authorof = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#is-author-of')
set @opus = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#opus')
set @pubdate = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#publication-date')
set @document = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#document')
set @subclasses = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#subclasses')
set @superclass = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#superclass')
set @subclass = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#subclass')
set @download = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#download')
set @email = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#email')
set @abstract = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#abstract')
set @language = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#language')
set @paper = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#paper')
set @pepper = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#steve-pepper')
set @influencedby = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#is-influenced-by')
set @influence = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#influence')
set @influenced = (SELECT topic_id FROM tm_tsrc
  WHERE src_loc = 'http://www.networkedplanet.com/tmql#influenced')

--
-- Retrieve all author names, i.e. the name of a topic which plays the role
-- author in an is-author-of association.
-- Expected Results:
-- "Holger Rath"
-- "Michel Biezunski"
-- "Steve Pepper"
-- "Steve Newcomb"
```

```

SELECT name_value from tm_assoc join tm_nameValue on rlp=topic_id
  where rlt=@author

--
-- Retrieve the name of all authors, this time ordered by the sort name
-- provided in person topics.
--
-- Expected Results:
-- "Michel Biezunski"
-- "Steve Newcomb"
-- "Steve Pepper"
-- "Holger Rath"

SELECT name_value from tm_assoc join tm_nameValue on rlp=topic_id
  where rlt=@author order by dbo.tm_sortName(rlp)

--
-- Retrieve the titles of all tutorials. The titles should be preferably
-- those in scope en, de or in the unconstrained scope, in that order.
--
-- Expected Results:
-- "Making topic maps more colourful"
-- "Euler, Topic Maps und Revolution"

DECLARE @names table(id int, name varchar(512))
INSERT INTO @names SELECT tm_nameValue.topic_id, name_value FROM
  tm_topic JOIN tm_nameValue on tm_topic.topic_id=tm_nameValue.topic_id
  JOIN tm_nameScope on tm_nameValue.name_id=tm_nameScope.name_id
  WHERE
    tm_topic.type_id = @tutorial AND tm_nameScope.scoping_topic_id=@en
INSERT INTO @names SELECT tm_nameValue.topic_id, name_value FROM
  tm_topic JOIN tm_nameValue on tm_topic.topic_id=tm_nameValue.topic_id
  JOIN tm_nameScope on tm_nameValue.name_id=tm_nameScope.name_id
  WHERE
    tm_topic.topic_id NOT IN (select id from @names) AND
    tm_topic.type_id = @tutorial AND tm_nameScope.scoping_topic_id=@de
INSERT INTO @names SELECT tm_nameValue.topic_id, name_value FROM
  tm_topic JOIN tm_nameValue on tm_topic.topic_id=tm_nameValue.topic_id
  JOIN tm_nameScope on tm_nameValue.name_id=tm_nameScope.name_id
  WHERE
    tm_topic.topic_id NOT IN (select id from @names) AND
    tm_topic.type_id = @tutorial AND tm_nameScope.scoping_topic_id IS NULL
SELECT name from @names

--
-- Retrieve the names of all persons who have not authored anything.
-- Expected Results:
-- "John Smith"

SELECT dbo.tm_displayName(topic_id) from tm_topic where
  type_id=@person
  and topic_id NOT IN (SELECT rlp from tm_assoc where rlt=@author)

--
-- Retrieve a list of all author names together with the title of their
-- publications.
--
-- Exected Results:
-- "Holger Rath", "Making topic maps more colourful"
-- "Steve Pepper", "Euler, topic maps, and revolution"
-- "Steve Pepper", "Navigating haystacks and discovering needles"
-- "Steve Pepper", "The TAO of Topic Maps"
-- "Steve Newcomb", "XML topic maps: finding aids for the Web"
-- "Michel Biezunski", "XML topic maps: finding aids for the Web"

SELECT dbo.tm_displayName(rlp), dbo.tm_displayName(r2p) from tm_assoc2
  WHERE association_type = @authorof AND rlt = @author and r2t = @opus

```

```

--
-- Retrieve a list of all titles of documents which are tutorials (i.e. are a
-- direct or indirect instance of class tutorial) sorted by publication date,
-- descending.
--
-- Expected Results:
-- "Making topic maps more colourful", 2000
-- "Euler, topic maps, and revolution", 1999

select dbo.tm_displayName(tm_topic.topic_id), occur_value from
tm_topic join tm_occurValue on tm_topic.topic_id = tm_occurValue.topic_id
WHERE
    tm_topic.type_id=@tutorial and
    tm_occurValue.occur_type=@pubdate
order by cast(cast(occur_value as varchar(5)) as int) DESC

--
-- Retrieve a list of documents, sorted by publication date (ascending),
-- only number 3 to 5, inclusive, together with this order number.
--
-- Expected results:
-- 3,"The TAO of Topic Maps", 2000
-- 4,"Making topic maps more colourful", 2000
-- 5,"XML topic maps: finding aids for the Web", 2001
--
-- NOTE: The following query returns all items. Would need to page
-- either in software or possibly using a CURSOR

select dbo.tm_displayName(topic_id), occur_value from tm_occurValue
WHERE
    occur_type = @pubdate
AND
    topic_id in (
        select topic_id from tm_topic
        where (type_id=@document) or
        (type_id in (
            select to_id from dbo.tm_tc(@document, @subclasses,
                @superclass, @subclass)
        ))
    )

--
-- Retrieve all topic identifiers of documents which have a download URL.
--
-- Expected Results
-- pepper99a
-- pepp00
-- d-topicmaps-color

select src_loc from tm_tsrc where topic_id in
(select topic_id from tm_occur where occur_type=@download)

-- Retrieve a list of author's email addresses where the author has authored
-- documents for which no download URL exists. Include the topic identifiers
-- of these documents.
--
-- Expected Results
-- pepper@n0spam.ontopia.net, pepper99b
-- srn@n0spam.coolheads.com, bienew01
-- mb@n0spam.coolheads.com, bienew01

select email.occur_resource, src_loc from
tm_assoc2 join tm_occurResource as email on rlp=email.topic_id
join tm_tsrc on r2p=tm_tsrc.topic_id
where
    r1t=@author and r2t=@opus and r2p not in (
        select topic_id from tm_occur where occur_type=@download

```

```

    )

--
-- Retrieve a list of author names where the author has written more than
-- 1 document.
--
-- Expected Results
-- "Steve Pepper"

select dbo.tm_displayName(topic_id) from tm_topic where
  (select count(distinct association_id) from tm_assoc2 where
    rlt=@author and rlp=topic_id group by rlp) > 1

--
-- A list of author names where the author has not written a single document
-- with someone else. As the person must be an author, she must have written
-- at least one document.
-- Expected Results:
-- "Steve Pepper"
-- "Holger Rath"

select dbo.tm_displayName(topic_id) from tm_topic where topic_id in (
  select rlp from tm_assoc where rlt=@author and rlp not in (
    select auth1.rlp from tm_assoc2 as auth1
      join tm_assoc2 as auth2 on auth1.r2p=auth2.r2p
    where auth1.rlt=@author and auth2.rlt=@author and
          auth1.rlp <> auth2.rlp
  )
)

--
-- Retrieve all topic identifiers of documents and their URLs which have a
-- non-working URL at query time.
--
-- Expected Results:
-- pepp00, http://www.broken.example.com/

-- This query would require separate application processing.

--
-- Retrieve all topic identifiers for documents for which the abstract in
-- english (i.e. the occurrence of type abstract in scope en) contains the
-- phrase 'topic map' or 'topic maps', case-insensitive.
-- Expected Results:
-- pepp00
-- bienew01

select src_loc from tm_tsrc
  where topic_id in (
    select occurrence.topic_id from tm_occur as occurrence
      join tm_occurvalue as abstractvalue on
        abstractvalue.occur_id=occurrence.occur_id
      join tm_occurscope as abstractscope on
        abstractscope.occur_id=occurrence.occur_id
    where occurrence.occur_type=@abstract and
          abstractscope.scoping_topic_id=@en and
          (abstractvalue.occur_value like '%topic map%' or
            abstractvalue.occur_value like '%topic maps%')
  )

-- Retrieve all documents which have a title in german (i.e. a basename
-- in the scope de).
-- Expected Results:
-- [ empty list ]

select topic.topic_id from tm_topic as topic
  join tm_name as tname on tname.topic_id=topic.topic_id
  join tm_namescope as tnamescope on tnamescope.name_id=tname.name_id

```

```

where tnamespace.scoping_topic_id=@de and topic.type_id=@document

--
-- Retrieve all topic identifiers of the pairs of documents which share
-- at least one word in the title, ignoring stopwords like 'and', 'of',
-- 'the'. No duplicates in this list are allowed.
-- Note: Duplicates of the form (a, b) and (b, a) may be allowed.
-- The result should be sorted by the identifiers.
--
-- Expected results:
-- d-topicmaps-color, bienew01
-- pepp00, bienew01
-- pepp00, d-topicmaps-color
-- pepper99a, bienew01
-- pepper99a, d-topicmaps-color
-- pepper99a, pepp00

-- This is a corner-case implying that a query language should provide
-- features normally associated only with full-text indexed systems.
-- Some RDBMS implementations have the facility to handle a query like
-- this, but standard SQL-92 would probably require additional applicatio
-- code.

--
-- Retrieve the identifiers of all topics which represent information
-- resources on the ontopia.net server(s).
--
-- Expected Results:
-- pepper99a
-- pepp00

select distinct topic.topic_id from tm_topic as topic
  join tm_sl as subjectlocator on subjectlocator.topic_id=topic.topic_id
  where
    subjectlocator.subj_loc like '%ontopia.net%' and
    topic.topicmap=@tmid

-- Retrieve the topic identifiers of all documents which are not written
-- in the language English (or where there is no information about that),
-- i.e. where there is no occurrence of type language.
--
-- Expected Results:
-- pepper99b
-- pepp00
-- d-topicmaps-color

select distinct topic.topic_id from tm_topic as topic where
  topic.topic_id not in
  (select occur.topic_id from tm_occur as occur
    join tm_occurvalue on occur.occur_id = tm_occurvalue.occur_id
    where occur.occur_type=@language and occur_value like 'english'
  )
  and topic.type_id in
  (select to_id from
    dbo.tm_tc(@document, @subclasses, @superclass, @subclass))
  and topic.topicmap=@tmid

-- Retrieve the topic identifiers of all documents which are directly or
-- indirectly influenced by something which
-- Steve Pepper wrote (i.e. interpreting "influenced by" as non-reflexive,
-- but transitive relation).
-- Copy the topic identifiers of the influential papers to the output.
--
-- Expected Results:
-- pepper99b,'influenced by', pepper99a
-- d-topicmaps-color,'influenced by', pepper99b
-- d-topicmaps-color,'influenced by', pepper99a
-- bienew01,'influenced by', pepper99b

```

```

-- bienew01, 'influenced by', pepper99a

declare pepper_papers CURSOR for
  select r2p from tm_assoc2 where association_type=@authorof and rlp=@pepper
declare @influence_table table(influencer int, influenced int)
declare @p int
OPEN pepper_papers
FETCH NEXT FROM pepper_papers into @p
WHILE @@FETCH_STATUS = 0 BEGIN
  insert into @influence_table select * from
    tm_tc(@p, @influencedby, @influence, @influenced)
  FETCH NEXT FROM pepper_papers into @p
END
CLOSE pepper_papers
DEALLOCATE pepper_papers

select influenced.src_loc as influenced, influencer.src_loc as influence from
  tm_tsrc as influenced
  join @influence_table as i on influenced.topic_id = i.influenced
  join tm_tsrc as influencer on influencer.topic_id = i.influencer

--
-- Retrieve all identifiers of topics which are either (direct or indirect)
-- instances of paper or tutorial.
-- Duplicates should be suppressed.
--
-- Expected Results :
-- pepper99a
-- pepper99b
-- d-topicmaps-color
-- bienew01

select distinct topic_id from tm_topic where
  type_id=@tutorial or
  type_id in (
    select to_id from dbo.tm_tc(@paper, @subclasses, @superclass, @subclass)
  )

--
-- Retrieve all titles of all publications (regardless their scope) together
-- with their most specific class,
-- i.e. that class where this publication is directly (and not indirectly)
-- an instance of.
--
-- Expected results:
-- Euler, topic maps, and revolution, conference-paper
-- Euler, topic maps, and revolution, tutorial
-- Euler, Topic Maps und Revolution, conference-paper
-- Euler, Topic Maps und Revolution, tutorial
-- Navigating haystacks and discovering needles, journal-paper
-- The TAO of Topic Maps, article
-- Making topic maps more colourful, tutorial
-- XML topic maps: finding aids for the Web, journal-paper

select topicname.name_value, topic.type_id from tm_topic as topic
  join tm_namevalue as topicname on topicname.topic_id=topic.topic_id
  where
    topic.type_id in (
      select to_id from
        dbo.tm_tc(@document, @subclasses, @superclass, @subclass)
    )

--
-- Retrieve all authors (i.e. all topic items which play the role author in
-- an is-author-of association).
-- Expected results:
-- [ holger-rath ]
-- [ michel-biezunski ]

```

```
-- [ steve-pepper ]
-- [ steve-newcomb ]

select distinct rlp from tm_assoc2
  WHERE association_type = @authorof AND rlt = @author

--
-- Retrieve all basename items of topics which play the role author in an
-- is-author-of association).
--
-- Expected Results
-- [ "Holger Rath", unconstrained-scope ]
-- [ "Michel Biezunski", unconstrained-scope ]
-- [ "Steve Pepper", unconstrained-scope ]
-- [ "Steve Newcomb", unconstrained-scope ]

select name_id from tm_name where
  tm_name.topic_id in (
    select distinct rlp from tm_assoc2 WHERE
      association_type = @authorof AND rlt = @author
  )

--
-- Retrieve a list of all occurrence items being of type email.
--
-- Expected Results:
-- [ email, srn@n0spam.coolheads.com, unconstrained-scope ]
-- [ email, mb@n0spam.coolheads.com, unconstrained-scope ]
-- [ email, pepper@n0spam.ontopia.net, unconstrained-scope ]
-- [ email, holger.rath@n0spam.empolis.com, unconstrained-scope ]
-- [ email, j.smith@example.com, unconstrained-scope ]

select tm_occur.occur_type, occur_resource, scoping_topic_id from tm_occur
  join tm_occurResource on tm_occurResource.occur_id = tm_occur.occur_id
  join tm_occurscope on tm_occurscope.occur_id = tm_occur.occur_id
  where
    tm_occur.occur_type=@email
```