	Evaluation of LRA Technical Model Query Requirements			
	Programme	NPFIT	Document Record ID Key	
	Sub-Prog Project	Informatics Data Standards Programme	NPFIT-FNT-TO-DPM-0950.05	
	Prog. Director	D. Perry	Status	FINAL
	Owner	S. Bentley	Version	1.0
	Author	A. Greenhalgh	Version Date	

Evaluation of LRA Technical Model Query Requirements

Amendment History:

Version	Date	Amendment History
0.1	2009-05-20	First draft for comment
0.2	2009-08-21	Second draft for review
0.3	2009-09-20	Third draft for external review
1.0	2009-11-02	Final version following external review

Approvals:

This document must be approved by the following:

Name	Signature	Title / Responsibility	Date	Version
Steve Bentley and/or Laura Sato		Head of LRA	26/11/2009	1.0

Document Status:

This is a controlled document.

Whilst this document may be printed, the electronic version maintained in FileCM is the controlled copy. Any printed copies of the document are not controlled.

Related Documents:

These documents will provide additional information.

Ref no	Doc Reference Number	Title	Version
1	NPFIT-SHR-QMS-PRP-0015	Glossary of Terms Consolidated.doc	<enter latest>
2		LRA Production Environment Business Context	0.5
3		LRA Technical Model Infrastructure Specification Part 1: Care Components	0.1.1
4	NPFIT-FNT-TO-DPM-0933.03	Terminology Binding Technical Specification	0.4
5		Austin T, Kalra D, Tapuria A, Lea N, Ingram D, 2008, 'Implementation of a query Interface for a generic record server' <i>IJMI</i> 77: 754-764	
6	https://www.portal.nss.cfh.nhs.uk/sites/idsp/LRAteam/LRA%20Core%20Team%20Documents/Forms/AllItems.aspx?RootFolder=/sites/idsp/LRAteam/LRA%20Core%20Team%20Documents/GPES%20-%20General%20Practice%20Extraction%20Service&View={A2652BBC-D1DB-440D-897B-2CEA07A0616F}	GPES Customer Requirements	0.1
7		Kalra,D., Singleton,P., Milan,J., MacKay,J., Detmer,D., Rector,A.,	

		Ingram,D, 2005, 'Security and confidentiality approach for the Clinical E-Science Framework (CLEF)' <i>Methods of Information in Medicine</i> 44(2): 193-197	
8	http://www.openEHR.org/wiki/display/spec/openEHR+Query+Specifications	openEHR Query Specifications	0.8
9		Terminology Binding Requirements and Principles	
10	http://www.connectingforhealth.nhs.uk/systemsandservices/ssd/products_and_services/vaprodmquest	MIQUEST Technical Specification	4.2.3
11		Health informatics — Electronic health record communication — Part 4: Security	EN 13606-4:2007
12		Health informatics — Time standards for healthcare specific problems	EN 12381:2005
13	https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/ref/ISO_21090/	Health informatics – Harmonized data types for information interchange	ISO 21090
14		Information technology. Database languages. SQL. Part 2. Foundation (SQL/Foundation)	BS ISO/IEC 9075-2:2007
15	http://www.w3.org/TR/xquery-operators/	XQuery 1.0 and XPath 2.0 Functions and Operators	23 January 2007
16	http://www.w3.org/TR/xmlschema-2/	XML Schema Part 2: Datatypes Second Edition	W3C Recommendation 28 October 2004
17	NPFIT-FNT-TO-TIN- 0427.09	Information Governance Requirements for ESP Systems	4.0
18		GPES-I Design & Overview of Version 1	1.0
19		James F. Allen: Maintaining knowledge about temporal intervals. In: Communications of the ACM. 26/11/1983. ACM Press. S. 832-843, ISSN 0001-0782	
20		SNOMED CT Bindings for Common Recording Patterns	0.5
21		GELLO: A Common Expression Language,	Release 1, 26/5/2005
22		Mulya M, Van der Aalst, Wil, Peleg M, 2007, 'A Pattern-based Analysis of Clinical Computer-interpretable Guideline Modeling Languages' <i>JAMIA</i> 14(6): 781-787	

23		Tu <i>et al</i> , 2007, 'The SAGE Guideline Model: Achievements and Overview' <i>JAMIA</i> 14(5)	
24		ISO/TS 18308:2004 – Health informatics -- Requirements for an electronic health record architecture	
25	http://www.omg.org/technology/documents/formal/ocl.htm	OCL Specification	2.0

Glossary of Terms:

List any new terms created in this document. Mail the NPO Quality Manager to have these included in the master glossary above.

Term	Acronym	Definition

Contents

0	About this Document.....	6
0.1	Purpose.....	6
0.2	Audience.....	6
0.3	Content.....	7
1	Introduction	7
1.1	Document Conventions and Terminology	7
1.1.1	Requirements terminology	7
1.1.2	Query Examples	8
2	Background	8
2.1	LRA Business requirements.....	8
2.2	Standards	9
2.2.1	EN13606.....	9
2.2.2	openEHR Query Specification	10
2.2.3	HL7	10
2.2.4	GELLO	10
2.3	Existing Query Systems.....	11
2.3.1	MIQUEST	11
2.3.2	GPES.....	13
2.3.3	Decision Support Systems.....	13
2.4	Scope and Assumptions.....	14
2.4.1	Retrieval	14
2.4.2	External Data	15
2.4.3	Care Record Modification	15
2.4.4	Query review and reuse infrastructure	15
2.4.5	Clinical Safety and Assurance	16
2.4.6	Information Governance.....	16
2.4.7	Security	17
3	Survey of Querying and Information Retrieval for Primary and Secondary Uses.....	17
3.1	Use Cases in Primary Use	17
3.1.1	Query Sources.....	18
3.1.2	Queries providing display of clinical information	18
3.1.3	Queries resulting in a clinical statement	19
3.2	Queries in Secondary Use	20
3.2.1	Query Sources.....	20
3.2.2	Cohort generation.....	21
3.2.3	Data collections	21
4	Querying and Data Retrieval Involving both Structural and Terminological Components	21
4.1.1	SNOMED CT	22
4.1.2	UCUM.....	24
4.1.3	Link types.....	24
4.1.4	Mappings to other vocabularies	25
4.2	Evaluation of Care Record Querying and Information Retrieval Requirements	25
4.3	Logical Query Model.....	26

4.4	Query Construction.....	26
4.4.1	Logical Query Syntax.....	26
4.4.2	Query Input.....	27
4.4.3	Query Result	27
4.4.4	Meaning-Based Retrieval.....	28
4.4.5	Query Metadata.....	29
4.4.6	Enumerations.....	29
4.5	Query Logic.....	29
4.6	Temporal Query.....	30
4.6.1	Temporal Relations	32
4.7	Functions and Operations.....	34
4.7.1	Mathematical and statistical functions and operators.....	36
4.8	Query Tooling	37
4.8.1	Query Repository	37
4.8.2	Query Builder	37
4.8.3	Example EHR	38
5	Requirements Summary	38
Appendix A	42
Example Queries	42
Query Class Model	48
Package Ira.technical.query	48
Class EHR	49
Class EHR_Provider_System	50
Abstract Class Q_AbstractQuery	50
Abstract Class Q_Filter	52
Abstract Class Q_Inference	53
Class Q_Metadata	54
Abstract Class Q_Parameter	54
Abstract Class Q_Query	55
Package Ira.technical.query.example	56

0 About this Document

0.1 Purpose

The purpose of this document is to determine the requirements that must be satisfied by LRA terminology bound Technical Models and the formalism used to specify computable query and data abstraction definitions in order to represent fully the types of query used to retrieve data from care records.

0.2 Audience

The intended audience of this specification is any individual, group or organisation involved in the development or use of the LRA.

0.3 Content

This document comprises the following sections:

- Section 1: Introduction
- Section 2: Background
- Section 3: Survey of Querying and Information Retrieval for Primary and Secondary Uses
- Section 4: Querying and Data Retrieval Involving both Structural and Terminological Components
- Section 5: Evaluation of Care Record Querying and Information Retrieval Requirements
- Section 6: Requirements Summary

1 Introduction

Information retrieval is an important aspect of the LRA, and there is a need to specify scope and requirements for the LRA information retrieval mechanism. The requirements SHALL be determined with particular regard to

1. the types of query and information retrieval that implemented systems¹ are required to support for primary uses (i.e. to support direct care of the patient or service user) as well as for secondary uses (i.e. for analyses); and
2. the need to be able to specify query and data abstraction definitions which are computable in a finite time given finite computational power and with predicates including both structural and terminological model elements.

1.1 Document Conventions and Terminology

The following sections should be noted when making use of this document.

1.1.1 Requirements terminology

The following keywords are used throughout the document to specify the extent to which an item is a requirement for the LRA.

MUST/SHALL – This means that the item is an absolute requirement.

SHOULD – This word means that there may exist valid reasons not to treat this item as a requirement, but the full implications should be understood and the case carefully weighed before discarding this item.

MAY/MIGHT/COULD – This means that an item deserves attention, but further study is needed to determine whether the item should be treated as a requirement.

Normative requirements are numbered and highlighted to enable tracking and clarity.

¹ This includes both clinical systems and systems dedicated to performing analyses on care records data.

1.1.2 Query Examples

Query examples consisting of an English description and query expression are provided and have been used within the analysis leading to requirements. Although these examples reflect existing queries they are non-normative and serve only to illustrate the analysis. This document does not presuppose the adoption of the formalism used for the illustrative examples as the query syntax, model, or presentation to be used in the LRA. The illustrative model in use in examples is detailed in Appendix A. Furthermore, example query expressions are based on broad assumptions about the use of reference models to produce models which are largely unavailable at the time of writing, and so should not be taken to represent modelling requirements or recommendations. Where examples are derived from a specific domain usage the source is noted.

2 Background

2.1 LRA Business requirements

Business Requirements for the LRA Production Environment, of which the Technical Model and query package form part, have been set out in [2]. Several of these requirements have a bearing on information retrieval and so are briefly restated here.

REQ-009	The ability to create longitudinal views of patient care from data derived from multiple sources
REQ-011	The ability to specify the data requirements for identifying patients with potential risks...in order to enable automated recommendations for targeted follow-up
REQ-022	Data standards defined by the LRA SHOULD be "either patient- or service user-centred or derived from personal care records data (e.g. for population analyses)".
REQ-001	For care professionals to "audit their own practice and that of their service or organisation with a view to provide comparisons with peers using data that is clinically meaningful, and that supports the objective of improving care outcomes. This would include the recording of patient data that can be compared against pathway expectations, as well as against 'markers' for desired (or undesired) clinical outcomes".
REQ-004	To process patient data from a variety of sources – i.e. the ability for clinicians to "aggregate, integrate and/or selectively view current and historical patient data from a variety of sources, in a timely, reliable, safe and meaningful way".
REQ-006	Provide clinicians with decision support "based on the particular requirements of an individual patient. When more standardised clinical data is available, both positive and negative alerts to clinicians may be supported in applications, based on algorithms that take into account multiple clinical data points at the same time".

Table 1: Relevant LRA PE business requirements, from [2]

2.2 Standards

Given the IDSP design principle of adopting established standards wherever appropriate [2] it is important to evaluate whether existing standards in this area might meet LRA requirements.

There are several query languages and standards in wide use. The most notable standards include:

- SPARQL; W3C Recommendation
- SQL; ISO 13249
- XQuery. W3C Recommendation

These and others like them are generic standards designed for expressing queries against a concrete dataset of a very specific type (graph, relational and hierarchical data respectively). They are therefore not suitable for specifying queries in the LRA. However, they are useful and will be referred to as exemplary models. Furthermore, given that these particular standards are the most likely implementation mechanisms for LRA queries it will be important to recognise their capabilities.

There are currently no standards directly pertaining to query at the logical level, however there are several standards in the wider healthcare domain which may impact or inform query requirements, or which could similarly be used in implementations.

Terminology standards are considered separately in Section 4 below.

2.2.1 EN13606

The LRA Care Components technical model conforms, with noted exceptions, to EN13606-1. While this standard does not specify direct requirements for query, a number of requirements for the broader EHR Architecture (EHRA) specified in ISO/TS 18308:2004 [24] are relevant:

“The EHRA shall be amenable to querying for the purpose of data aggregation to support information gathering required for population and public health initiatives, surveillance, and reporting.”

“The EHRA shall support selective retrieval and customized views of the same information for specific needs (e.g. decision support, data analysis).”

“The EHRA shall enable the storage of multiple values of the same measurement taken at closely proximate times at the same contact, or at different contacts and at different locations. The context of these measurements shall be preserved – such as who took the measurement, what method was used etc. These values should be able to be returned in a query and ordered in different ways”

“The EHRA shall support the continuity of a clinical process, the ability to query the status of a process, modify an existing process, and verify that a process has been completed.”

“The EHRA shall support the recording and querying of data to enable the measurement of operational and clinical performance, to ensure compliance with standards of care, to ensure quality process, and to measure outcomes.”

“The EHRA shall support the ability to review information of all types recorded in the past, including via the use of query and filter facilities, during the data capture process.”

As the LRA has no stated compliance with this requirement set these are not taken on as explicit requirements in this document. However the requirements presented above are considered to be aligned with the high-level requirements of LRA query.

2.2.2 openEHR Query Specification

The openEHR foundation has developed a query specification [8]. There are currently two candidate solutions for this specification: The XPath-based A-path, and Archetype Query Language. Of these the latter is referenced in EN13606 as a likely avenue for query development within that standard. The LRA has no current requirement to interoperate at the logical level with openEHR, therefore the openEHR Query Specifications provide no direct requirements. However, it is foreseeable that future LRA-compliant systems may be based on openEHR, and so LRA queries MIGHT be transformed into openEHR Query Specification-compliant language. The requirements outlined in the openEHR Query Specification are similar to many of those outlined in this document, and so the design may influence the implementation of an LRA query language.

2.2.3 HL7

Alignment with HL7 version 3 as used in the NHS CFH Message Implementation Manual (MIM) has been considered an important goal within the LRA. HL7 V3 specifies a general structure for queries and their response. These methods are then applied within functional domains to develop specific query/response pairs for those domains' needs. In particular, there is no implication that a specific system must support generalized queries to comply with the HL7 Standard. Rather, these transactions provide a format by which a system may support queries to the extent it desires. As such HL7 Queries require that a data owner must decide that it would like to make information available via a query and decide precisely what data will be made available and how it will be offered. The data owner specifies exactly which input variables the data requestor can use to control the data that the data owner agrees to return. The complete specification of what data are available, how the data will be returned, and what variables can be set or constrained need to be provided for both the query and the response. However, the operation and logic of the query itself is not defined.

This is that case for the MIM and supporting business analysis, which specify only the structure and content of queries and responses. The HL7 Query structure is in use only within the PSIS, PDS and LR domains. Where queries defined in the MIM might need to process several document requests and responses an optional batched structure is used to enable more efficient messaging.

The HL7 query infrastructure provides a useful interface specification mechanism and development methodology which will be important when considering LRA query implementations, but it provides no specific requirements at this stage.

2.2.4 GELLO

A potentially very useful component of HL7 is the GELLO expression language. As described in the specification [21], GELLO is based on the Object Constraint

Language (OCL). OCL is well-developed as a constraint language and is used as such within the LRA. However, it also has a number of features that make it desirable for use as an expression language. GELLO incorporates most of OCL's functionality with the exception of some unneeded constraint focussed capabilities which have been removed. However, GELLO is not an exact subset of OCL, having extensions and incompatibilities at a number of points.

While GELLO is a part of the HL7 standard and usage examples focus on use in this model, the syntax of the GELLO language can be used with any object-oriented data model. This would include as the LRA care record model, and examples exist of its experimental use in querying an EN13606-based model [http://wiki.medical-objects.com.au/index.php/Clinical_Decision_Support].

GELLO is strongly typed and object-oriented, and includes a set of basic built-in data types in order to maximize the ability to share expressions such as in queries. Although this is not explicitly stated these basic types are the UML/OCL kernel datatypes as used to define the ISO 21090 datatypes in use within the LRA care record model.

GELLO presents a compelling option for use as a query expression language, and should be evaluated as such against the requirements set out in this document.

2.3 Existing Query Systems

A number of systems implement querying within healthcare. These range from legacy systems to those in current development. As these are not in themselves standards they will be considered here as examples rather than sources of requirements. The systems are described in brief with particularly relevant elements drawn out for consideration within the LRA requirements.

2.3.1 MIQUEST

MIQUEST is a system for carrying out data collection from GP systems. It was developed over fifteen years ago and is now a mandatory part of the General Medical Practice Computer Systems - Requirements for Accreditation (RFA99) scheme. The system is useful to examine here as it provides an example of a highly successful system for executing queries and aggregating results from distributed heterogeneous healthcare systems.

The system specification [10] sets out:

- A common form for expressing enquiries;
- A common form for the responses to these enquiries;
- A common view of core data elements in GP systems and the relationships between them;
- A process for submitting enquiries and receiving responses to those enquiries. This includes:
 - Preparation of enquiries and their submission to GP Systems
 - Processing by GP Systems (including facilities for viewing and authorisation)

- Receipt, aggregation and analysis of responses from GP Systems.
- Restrictions on the availability of identifiable patient-related information that conform with current legal and ethical guidelines.
- Software to write and submit queries and receive and aggregate responses prior to analysis in any widely available spreadsheet, statistics package or database.
- Conformance tests for GP systems that support the MIQUEST Specification.

MIQUEST is on the whole is highly specific to querying GP systems; however two particular elements of the system are of particular interest: the confidentiality and data sharing model, and the query syntax itself.

The system provides a useful confidentiality model which allows queries reporting patient identifying data to be run on local systems with practice approval and for queries aggregating results with identifying detail removed to be requested remotely. It also allows details of the query author and the data collection scheme under which the results are being collected to be specified in the query header. Metadata about the person authoring the query and receiving the results is provided as a header to the query. This can include information on any data collection agreement under which the query is being carried out.

Queries themselves are specified using a bespoke query language, Health Query Language (HQL) which merits separate consideration.

2.3.1.1 *Healthcare Query Language*

HQL provides an example of a query expression language for the healthcare domain. It is a superset and extension of SQL, which has been simplified for use by non-technical users. The HQL grammar is introduced in the MIQUEST specification [10], an example is given below.

```
REPORT
PRINT DATE, VALUE1, VALUE2
FROM JOURNALS(LATEST_FOR_PATIENT)
WHERE CODE IN (".246%")
#{.246%} is the Clinical Term for "Blood pressure taken"
# Note that the two elements of the blood pressure are considered to be part of
# the same record.
```

Box 1: HQL query reporting the latest systolic and diastolic blood pressure for each patient, from [10]

There are three query types defined in HQL:

- *ANALYSE* queries which count patients or events meeting particular criteria. An analysis may be subdivided into bands by age and sex and/or by other criteria.
- *REPORT* queries which report subsets of the records of a group of patients. The records reported are selected by one or more sets of selection criteria.

- *SUBSET* queries which select a subset of patients to whom further queries may be applied. The subset of patients is selected by applying a set of selection criteria to the practice population or to an existing subset of that population.

Coded data is supported, with the coding scheme specified either as an enumeration of possible code values (e.g. for encoding sex), as a reference to a coding scheme (e.g. Clinical Terms) or a reference to a body responsible for providing the code values or identifiers required (e.g. NHS for GP numbers, referral speciality codes, etc.).

As well as these simpler vocabularies, clinical coding schemes are used in the *CODE* record attribute. However, there are a number of clinical coding schemes in use within GP systems (e.g. READ2, CTV3, SNOMED). More than one clinical code system is permitted in a single query, and these can be identified in the header. Alternative values in different coding schemes can be provided directly within the queries, with no mapping occurring at the GP system level. Operators exist to allow specification of a range of terms or selection of a subset from a hierarchy of terms.

Numeric values such as height are provided in the optional record attributes *VALUE1* and *VALUE2* which are the values relating to the *CODE* attribute. There are a standard set of values which must be available to query. These required values include Body Mass Index which, the requirement states, if calculated for display in the GP system the same calculation should be performed to return the BMI when codes relating to the BMI are referenced in a query.

2.3.2 GPES

The General Practice Extraction Service (GPES) Project is planning to implement a means of extracting data from general practice system databases for a range of secondary uses. Hence, it aims to provide a replacement for MIQUEST. Requirements have been gathered for the GPES system which is currently at the procurement stage. The version 1 interface specification (GPES-I) [18] is at the core of the system and is the item of interest here as it defines the query syntax.

GPES version 1 will use HQL to specify queries. In making this determination XQuery was discounted as none of the GP system databases are XML-based. Likewise the Archetype Query Language was discounted as GP systems are not archetype-based. SQL was assessed as requiring further detailed analysis to determine the fit with version 1 needs. It is recognised that HQL will not meet the long-term needs of GPES, and therefore there is an argument that HQL should be enhanced, in part to align fully with HL7.

Like MIQUEST, GPES faces a wide range of code systems in use within GP systems. GPES will deal with this by defining separate queries for each code system and mapping between queries with the same semantics within the query library. This is a departure from the MIQUEST/HQL approach of mapping by providing alternative codes within the query at the individual term level.

2.3.3 Decision Support Systems

In linking clinical observations with encoded clinical knowledge, Clinical Decision Support Systems (CDSS) necessarily form queries, both over the patient record, and on the clinical knowledgebase. Due to the breadth of clinical knowledge and the costs

involved in encoding it CDSS are highly domain-specific, although there are a small number of languages commonly used across these systems. For these same reasons research in this area has tended to focus on encoding guidelines rather than querying the patient record. A description and evaluation of the guideline modelling languages Asbru, GLIF, and PROforma is provided in [22]. Another example, the SAGE Guideline Model, is of particular interest as it uses GELLO as the foundation of its expression language [23].

Another useful facet to examine is the process of encoding and review of clinical knowledge technical representations. The requirement for expression authoring and review by clinicians will be one shared with the LRA query system. Various methods for structuring or transformation of expressions into a format accessible by non-technical personnel have been used. These should be examined for potential for application here.

This is an area worthy of greater and more detailed investigation as there are highly informative examples of patient records query systems. The approaches used in these systems should be evaluated and taken forward within query implementations and authoring and review process.

2.4 Scope and Assumptions

This section states limitations on scope and basic assumptions, some of which produce basic requirements for query. Assumptions are derived from the overall aims and high level architecture of the LRA. Use cases presented below draw on the assumptions and presumptive requirements set out in this section.

The scope of the Logical Query Model is illustrated in Table 3 below using the dimensions of query input and output type:

In scope: ✓ / X Example	Direct Query	Derived operation	Direct inference	Derived inference
Patient	✓ DOB	✓ BMI	✓ HRG Obesity	X ICD Coding
Linked (related) Patients	✓ Foetal crown rump length	✓	✓ Gestation stage	X
Population	✓ All vaccinated patients	✓ Count/average	✓ Herd immunity	X

Table 2: Query model scope

2.4.1 Retrieval

The primary functionality of query within the LRA will be meaning-based information retrieval. Further analysis and display of this information is expected and must be supported but these are outside the scope of query. As such, some LRA use cases

making use of query will require further manipulation of query results. It is expected that clinical and analysis applications making use of the LRA will provide this functionality where required.

2.4.2 External Data

The care record is the central concern of the LRA, and so the scope of querying in the LRA is limited to queries on this, with any requirement to query data external to the care record excluded from consideration here.

This exclusion includes information on the care provider and the care setting, for example, although it is recognised that this information is often required by indicators. The care components model includes identifiers for these associated objects and so it would be possible to meet these requirements externally by joining LRA query results to external datasets.

2.4.3 Care Record Modification

Querying as specified here is regarded logically as a single operation that is side effect free, i.e. execution of the query does not affect the state of the care record. However, this does not preclude other operations invoking a query and adding its output to the care record as new content.

QRY 1 A Query MUST NOT allow the data of a care record to be modified.

2.4.4 Query review and reuse infrastructure

Queries are assumed to be complex and time consuming to develop, and therefore sharing and reuse is to be built into the query system. A mechanism for storing and sharing queries alongside associated metadata should be developed in support of this. The basis of the search functionality has yet to be decided, as unlike expression constraints, which have a focus object and a well defined structure, there is no obvious basis for organising queries.

QRY 2 Queries MUST be uniquely identified.

QRY 3 Queries SHOULD be able to be persisted in a searchable repository with appropriate access control restricting reading editing and authoring.

QRY 4 Queries SHOULD have a name and/or short human-readable rendering that can be displayed in response to a search.

QRY 5 Queries MUST have some human-readable format such that they can be reviewed against requirements to determine if they are suited to a particular use or reuse. Complex queries such as those made up of several sub-queries MAY have some level of drill-down so that more or less detail can be viewed.

QRY 6 Query expressions SHOULD be able to include notes or comments inline clarifying the usage or operation of a particular section.

QRY 7 The query repository SHOULD support versioning, change control and tracking of query authorship and modifications.

QRY 8 Queries MUST allow specific terms to be marked as parameters which may be set at the point at which they are run.

QRY 9 Queries MUST allow their input to be the output of another query.

QRY 10 The output type of all queries MUST be clearly specified.

Given this complexity and the potential importance of queries in patient care and safety, this also creates a need for a query development and assurance process.

QRY 11 A query development and assurance process MUST be available for use where query development is required.

QRY 12 Query metadata MUST record the development status and assurance process stage of a query.

QRY 13 Query metadata MUST record the point at which the query was reviewed with respect to the care record model. Where the model is subsequently changed this MUST be reviewed against dependant queries.

While dependencies on other queries are relatively easy to record and/or resolve, it remains unclear how query dependencies on the care record model can be recognised. This is an area which requires further consideration.

2.4.5 Clinical Safety and Assurance

Note: Prior to consideration of individual queries themselves, the LRA query requirements and development process as set out in this document and any subsequent products require clinical assurance. This has not yet occurred and so none of these products should be used in a clinical environment.

Given that queries may inform clinical decisions at the level of individual patient care clinical safety is of particular importance. The assurance requirement specified above must include clinical assurance as a central component. Queries may be sensitive to context and usage through incorporating assumptions which may be valid in one context but not in another. Therefore this requirement for review must also apply when reusing a query as a sub component in other queries.

QRY 14 Queries MUST be available in some representation accessible to clinicians for review.

QRY 15 Query metadata MUST enumerate any assumptions made by the query author for the purpose of the query.

QRY 16 Query metadata MUST separately record instances of query reuse.

QRY 17 Where queries are reused, this reuse MUST be included within the query assurance process

2.4.6 Information Governance

Information Governance issues are a clear concern in querying patient data, and consideration of Information Governance issues specific to the query should be included in the query development process specified above. An indication of the IG requirements is provided in [17]. Three particular issues relate to the technical query package specified here. These issues are not further developed into formal requirements in this document as this is a matter for wider consideration in respect of the LRA as a whole:

- The means of identifying patient consent for a specific query or use.

- The effects of some element of the care record being unable to query through consent being withheld or an individual care record element being protectively marked on queries returning aggregate information.
- Given the stated LRA aim of integrating secondary and primary datastreams which leads to a possible requirement for a pseudonymisation and/or anonymisation service as part of the query implementation. Approaches such as that used in the project reported in [7] may provide solutions for this.

A mechanism is defined in *EN13606 Part 4* for including access policy information within the care record. This is intended to inform the information recipient about the wishes of the subject of care and of healthcare providers for how future access requests for the data should be managed, and therefore provides a likely avenue for future work within the LRA.

2.4.7 Security

The LRA explicitly excludes definition of security requirements [2] and so it is assumed that authentication and access are handled transparently outside of the query package. However, it is recognised that this may have direct impact on query where role-based access being granted to users would restrict the type of query which can be run and/or the information returned based on the user's role and/or relationship to the patient.

3 Survey of Querying and Information Retrieval for Primary and Secondary Uses

Requirements for querying the LRA Technical Model have been gathered through meetings with LRA stakeholders identified in project documents. From these meetings a number of use cases have been developed which aim to be representative of the range of queries carried out across stakeholders.

The LRA aims to integrate secondary and primary data collection streams, and therefore makes no fundamental distinction between primary and secondary data. A more useful distinction in this context is not the purpose of data collection, but whether a query is over a single subject of care or a population. The requirement outcome of this is that:

QRY 18 Queries MUST be able to be defined over a population or specific patient records.

3.1 Use Cases in Primary Use

Queries in primary use are defined here as those directly in support of individual patient care. It is important to make a distinction between queries providing a filtered view on the patient care record to the clinician, for example showing body temperature over the last hour, and queries resulting in a clinical finding, such as a query showing a history of miscarriage or observation such as a query calculating BMI from previous values.

3.1.1 Query Sources

Common clinical application functionality has been extensively researched within the NHS Common User Interface (CUI) Programme. While the Programme output is naturally centred on providing requirements for a common look and feel, the research is highly informative on common clinical application requirements in the whole, and as such describes a basic set of use cases for queries in primary use. The following analysis draws on the CUI Guidance Catalogue, specifically the medication search and prescribe, and display of clinical data workflows.

3.1.2 Queries providing display of clinical information

The analysis is based around supporting a secondary care clinical interface as exemplified in the CUI demonstrators. These are queries which present a filtered view of existing record entries without inferring further information. Example 1 in Appendix A provides an illustrative example of this type of query.

It is important to distinguish where queries inform or direct clinical actions. This is for a number of reasons: firstly this requires a greater standard of assurance than other queries; secondly the information that a clinician had available when an action is carried out should be auditable for medico-legal purposes; finally legal responsibility for a unit of information must be made clear through attestation.

QRY 19 Where they have been used directly in the care of a particular patient, results of specific queries **MUST** be able to be stored within the care record of an individual patient alongside clinical entries.

Queries may form a part of a distinct operation on patient records. Where this is the case it is assumed that the operation workflow including the query within it must be modelled. In this case it should be the operation and result where appropriate which is recorded within the care record. The modelling of such operations are clearly outside the scope of this document but any queries used within must be modelled and persisted consistent with the requirements set out here, and so specific further requirements arise.

Ordering of values is clearly of importance, and it is anticipated that results of LRA queries will be displayed interactively within information systems, allowing the users to adjust the display of information to suit their needs. This might include filtering, ordering and grouping information returned from a query. In terms of deriving requirements though scope limitations are significant. The requirement for ordering objects must be to support retrieval based on the semantics of ordered values, rather than sorting items for display. Temporal and numerical order may be semantically significant and therefore must be supported in queries, but ordering by name or text content is not. Likewise grouping and aggregation should be carried out at a layer of abstraction above retrieval.

Datatypes with comparators can be totally ordered, such that every (unique) item has an order relation to every other item. ISO 21090 defines T.max(T) and T.min(T) comparators on some of the datatypes, shown in Table 2.

CO	<i>Coded ordinal</i>
INT	<i>Integer</i>
MO	<i>Monetary amount</i>

PQ	<i>Physical quantity</i>
REAL	<i>Real number</i>
TS	<i>Timestamp</i>

Table 3: ISO 21090 datatypes with $T.max(T)$ and $T.min(T)$ defined

Sorting other datatypes will be result in a partial order, based on the equality of the items such that items with the same value will be placed together but will be otherwise unordered. In these cases sorting in ascending or descending order are meaningless, and either should produce the same output.

Following the functional style, sort operations will not affect the order of the underlying collection but will result in a new appropriately typed instance of the collection.

QRY 20	Collections retrieved from care records or returned from operations MUST be able to be sorted by one or more attributes.
QRY 21	Collection sorting MUST be based on natural order and must apply to datatypes with a <i>min</i> operation defined.
QRY 22	All other types MUST be partially ordered based on the equality semantics defined in the specification. Ordering of items with equal value is arbitrary, and MUST NOT be relied upon within queries.

Order must take the possibility of null flavors into account. Ordering is effected by greater-than or less-than comparators. However, it is clearly stated that the *equals* comparator applied to two instances with the same null flavour should return a null flavour and so, by extension, should greater-than or less-than comparisons. Expected behaviour is not clear therefore. The SQL standard, for example, does not define an explicit sort order for nulls, and includes the NULLS FIRST or NULLS LAST optional clause in the ORDER BY clause. Similarly XPath allows *empty least* or *empty first* to be specified, which affects the sort order of null-like values.

The default should be for null flavors to be sorted to the end of a collection. This seems an appropriate default in that null flavored values will appear at the end of reports and UI lists.

QRY 23	Queries MUST allow the sort order of nulls to be specified. When sorting ascending order and not otherwise specified, queries SHOULD default to sorting null flavors after non-null flavored data.
--------	--

3.1.3 Queries resulting in a clinical statement

Queries might abstract from existing information and broader clinical knowledge such that the result should be considered a new clinical statement. New clinical statements can be derived from existing statements in a number of ways:

- A key advantage of a structured terminology is that the structure may be used to generalise information. Inferences here might be based on simple concept subsumption, such as inferring that the left index finger is a part of the left hand.
- Existing general clinical knowledge might also be used to abstract or generalise: a pulse rate recorded as found to be normal might infer that a value of between 60 and 80 bpm was observed.

- Assumptions might be made about the practice of recording information in the care record. For example if an adult patient has no recorded diagnosis of diabetes, it may be appropriate to assume that they are not diabetic.

Examples 2 and 3 in Appendix A are examples of this type of query.

There are a wide range of calculations, generalisations and inferences which might be made from the care record information. All of these forms are dependant, to a varying degree, on:

- The structure of the care record model;
- The specific information available in the care record;
- The general clinical knowledge encoded in the query;
- The specific clinical context;

Links between the clinical statement resulting from this kind of query and each of these items must be made. Responsibility for these statements must also be made clear through appropriate attestation.

Calculations and inferences based on further interpretation of care record data must be carried out at a level of abstraction above information retrieval. Considerations when performing information retrieval are quite different to those when performing further analysis, and so these should be carried out separately. Likewise once created these two levels may be managed as distinctly different objects: the retrieval layer will be tightly bound to the information model, while the abstraction layer will have dependencies on the clinical knowledge, classification scheme or analytical processes represented. For this reason the model must be able to reference these separately.

QRY 24	Where query results represent a new clinical statement through calculating, generalising or inferring based on existing care record information, the query mechanism MUST be able to store this as a clinical statement in the care record, with links to the information on which the result is based.
QRY 25	Where a query creates clinical statements, these statements MUST be marked as attested by the query system, along with other appropriate audit information.
QRY 26	A query abstraction MUST define any aggregations and the logic of calculations and inferences based on record data selected by a query.

3.2 Queries in Secondary Use

These are queries made for a purpose other than care of a specific patient.

3.2.1 Query Sources

The primary source of information on existing queries has been the NHS Information Centre. Many of the following queries draw on the IC Indicator Library, which is currently in development. The structure of the use cases is based around existing LRA analysis work on query user stories.

3.2.2 Cohort generation

The goal of type of query case is to determine a cohort of patients matching specific criteria over multiple care settings and systems. An example is provided an Example 4 in Appendix A.

Having identified a research cohort, researchers will carry out further analysis either with the patient directly or using their records. The entire care record for the identified patients will therefore need to be available.

QRY 27 Queries SHALL be able to return the entire care record for a patient matching a set of clinical and demographic characteristics

3.2.3 Data collections

Data collections based on queries form a significant proportion of analytical work. While queries must be able to be defined over the entirety of the care components model, including both care records and administrative data, queries requiring joining patient data to external datasets have been ruled out of scope.

Where queries are designed to produce output for a specific purpose such as mapping to another vocabulary or joining to another dataset, this further use should be recorded. Linkage between technical artefacts including queries and business requirements, which exist in the knowledge aspect of the LRA is an important pre-existing requirement for the LRA, and will be met by the interface objects. Any requirement here for linkage between implementation and end use therefore is considered to covered by this higher level requirement and should be met by the interface artefact described elsewhere.

4 Querying and Data Retrieval Involving both Structural and Terminological Components

As the *Terminology Binding Requirements and Principles* [9] states, the key requirement of any care record entry reuse is the need to retrieve particular items of information reliably and consistently, irrespective of how the data were entered and stored. Retrieval of care record entries that encompasses the information model and terminology contributions to meaningful representation of clinical information must therefore address the interdependencies between these two components.

Where there is scope for structural and terminological representations of care record content to overlap within LRA models either one permissible representation must be specified or both representations must be permitted and the rules for losslessly transforming between them specified. Developing a set of authoring guidelines which will produce consistent models in this vein is vital to the ability of a query system to meet requirements.

While SNOMED CT is clearly a core part of the LRA other simpler types of coded expression are in use alongside it, and bring their own requirements. Significant terminologies in use within the LRA are examined below.

It is also important to recognise that a range of healthcare terminologies exist and are in wide use outside the LRA. As queries represent an interface between the LRA and

external uses there may therefore be a requirement to interoperate with these systems.

4.1.1 SNOMED CT

The use of SNOMED CT within the LRA is well described in *Terminology Binding Requirements and Principles* [9] and its subsequent application in *SNOMED CT Bindings for Common Recording Patterns* [20]:

The LRA model specifies that entries in clinical records that need to be retrieved and reused based on clinical meaning shall be represented using SNOMED CT Expressions. A SNOMED CT Expression consists of references to one or more SNOMED CT ConceptIds. A pre-coordinated expression contains only one ConceptId. A post-coordinated expression contains two or more ConceptId and these are related to one another in a way that is specified by the SNOMED CT Concept Model.

The main aspects of SNOMED CT post-coordination that are required to support the LRA are:

- The context wrapper
 - The SNOMED CT context model explicitly represents the situation/context that applies to individual ENTRYs.
- The clinical focus concept
 - A clinical finding, observable, event, procedure or social context.
- Refinements
 - Values applied to SNOMED CT concept model attributes that add detail and specificity to the focus concept.
 - Refinement includes adding specificity to an existing defining relationship and applying values to other attributes sanctioned by the concept model.
 - Refinement also includes 'indirect' use of attributes such as the application of laterality to a finding or procedure. In this case, the laterality logically applies to the finding or procedure site.

The *Terminology Binding Technical Specification* [9] specifies a mechanism for expressing constraints on SCT expressions, termed *Expression Constraints*, which is in use to describe LRA model constraints.

4.1.1.1 Analysis

The ability to query entries based on SCT expressions is a core requirement of querying in the LRA. SNOMED expressions exist in care records as instances of the CD datatype.

The datatype specification states that equality of two CDs only takes account of the code and codeSystem literal values and must not consult the semantic meaning of the content to determine whether the same concept is intended. The *implies (other : CD):BL* operation is defined on CD which returns true “if this code/codeSystem is a specialization of the other code/codeSystem or has the same meaning”. Thus two isomorphic forms of the same SNOMED expression will imply each other and hence the *implies* relation will hold for both pre- and post-coordinated expressions.

QRY 28	Queries MUST be able to express literal SNOMED CT Expressions as query terms.
--------	---

QRY 29 Queries MUST implement the *implies* operation on CD as defined in the datatypes specification, such that where two SNOMED CT expressions have the same normal form they imply each other.

It is explicitly stated that Expression Constraints may also be used to specify query selection criteria, and several examples the use of this within retrieval criteria are provided [20]. Expression Constraints can refer to either the semantics of a SNOMED expression (termed a *semantic expression constraint*) or its form (a *literal expression constraint*). As the focus within the query requirements is on meaning-based retrieval, no use case has yet been identified for the use of a literal expression constraint. Nonetheless, the existing expression constraint mechanism should ideally be used within queries unchanged, and so both forms of constraint are required:

QRY 30 Queries MUST be able to test if an instance of a SNOMED expression conforms to a SNOMED Expression Constraint.

As specified in the *Technical Specification*, [9], the test for conformance with a *semantic constraint* is:

Does the result of a SNOMED CT normal form transform applied to the *field* value conform to the *expression constraint*?

While the test for conformance with an *expression-structure constraint* is:

Does the literal (untransformed) *field* value conform to the *expression constraint*?

Caution should be used when querying using literal expression constraints as this could easily result in false negative results where semantically equivalent terms expressed in different forms are discarded.

A number of expression serializations are specified, and queries should be able to handle all of these. However, as transformations are available for each this is not an absolute requirement.

QRY 31 Queries MUST be able to represent SNOMED Expression Constraints as query terms.

QRY 32 Queries SHOULD be able to accept all serializations of Expression Constraints as query terms.

Examples above provide use cases for constructing a SNOMED expression and expression constraint within a query:

- Constructor bindings might be used to help test equivalence between structural and terminological representations.
- Abstracting meaning from query results requires the construction of a SNOMED Expression carrying that meaning.
- Parameterised queries which select care record objects based on conformance to an Expression Constraint may require that terms in the Expression Constraint can be passed as parameters.

There may be further use cases not yet identified.

The concept of a *Constructor Binding* was introduced in the *Terminology Binding Technical Specification* and this would fit the requirement for building SNOMED expressions and, it is assumed, could be extended to build expression constraints:

Constructor bindings are used to specify the way in which a set of related *fields* contribute to a valid SNOMED CT *expression* that encapsulates the relevant context and refinements. Strictly speaking, the terminology component of a *constructor binding* is not a constraint but a blueprint. This blueprint determines how multiple data points should be transformed into a single *expression* that fits within a specified common pattern for representing a particular type of information.

Queries will be required to support constructor bindings; however, the operation of constructor bindings is not currently specified in this version of the document:

QRY 33 Queries SHOULD be able to construct SNOMED Expressions and Expression Constraints based on care record objects and query parameters.

4.1.2 UCUM

The Unified Code for Units of Measure (UCUM) is a code system concerned with specifying units of measure. Units of counting, such as beats per minute can be expressed using UCUM notation, but do not carry unit semantics. UCUM is the core vocabulary for expressing quantities within the LRA. The Physical Quantity datatype PQ is constrained to using UCUM, while PQV is reserved for physical quantities expressed using another code system. SNOMED concepts for many units of measure, but these are not currently used within the LRA.

As it is currently expected that UCUM will meet all the requirements for representing physical quantities within the LRA there is no requirement for conversion between PQ and PQV.

4.1.2.1 Analysis

There is a need to be able to include literal UCUM expressions as query terms, and hence create a new instance of a PQ.

Comparison between values of type PQ expressed in different but commensurate units (such as 1 metre =100 centimetres) is supported in the ISO 21090 specification [13]. While automatic unit conversion for the purpose of value comparison will therefore occur within the query using datatype semantics already defined, query output may be required in a specific unit.

QRY 34 Queries SHALL be able to express terms using literal UCUM expressions

QRY 35 Queries SHALL be able to return results in a specified unit where values are held or derived in a different, commensurate unit.

4.1.3 Link types

Links between care record components are represented by a composite of LINKS and COMPONENT_RELATIONSHIP_ELEMENTS, as described in the Care Record Model Specification. The current content of SNOMED CT supports an insufficiently wide range of link assertion concepts to meet the requirements of the LRA and so an LRA-specific vocabulary has been developed to meet this need, along with a number

of supporting vocabularies. Although this currently uses the UML enumeration mechanism the result is a relatively complex vocabulary which should be considered alongside the terminologies described above.

4.1.3.1 Analysis

The LraComponentRelationshipElementMeaning terms are marked as transitive and symmetric as appropriate and terms may have parent and inverse relationships with other terms. In addition the pattern used to represent linked elements in the information model is relatively complex.

Queries will be required to access and query linked elements. Queries will also need to test the meaning of links between elements. While it would be possible to navigate over and test linked elements in the same way as other classes and properties in the model this would not be efficient and would not make full use of the rich features of the vocabulary. A simplified method for querying and navigating linked elements and the ability to query on relationship type equivalence are therefore required:

QRY 36	Queries SHALL provide a simple method for testing existence and type of component relationships.
QRY 37	Queries SHALL be able to test for equivalent component relationships types taking account of subsumption, inverses, symmetry and transitivity defined in the vocabulary.
QRY 38	Queries SHALL provide a simple method for accessing linked components, including selecting them by type.

4.1.4 Mappings to other vocabularies

A number of query use cases need to provide output using vocabularies other than SCT. The level of inference necessary to form these mappings has been ruled out of scope for query. This use case highlights that query results may be further transformed

4.1.4.1 Analysis

Mappings from Read 2, CTV3, OPCS 4, ICD-9-CM and ICD-10-CM to SNOMED already exist. Systems should be free to implement querying based on these or other terminologies or classifications but it should be their responsibility, with the appropriate implementation guidance, to provide the necessary translation. However, at the logical level SNOMED CT will be the only clinical terminology in use.

As stated above, where queries are in use as part of a process involving external mapping to other terminologies to meet a specific requirement this usage must be recorded. The requirement to do so is covered by that for maintaining links between queries and business requirements.

4.2 Evaluation of Care Record Querying and Information Retrieval Requirements

This section describes the features of a Logical Query Model based on an evaluation of the requirements identified above. Finally requirements for tooling supporting this model are identified.

4.3 Logical Query Model

This section draws requirements identified in earlier sections together to determine the necessary features of a Logical Query Model.

The role of a Logical Query is to specify the operation of a query, rather than to be directly executable. The latter is delegated to one or more implementation query languages to which the logical syntax can be mapped and executed through an interpreter.

QRY 39 A query **MUST** define its logic and output type but **MUST NOT** restrict implementation.

It is important that the model can demonstrate the linkage between requirements for a specific domain and the data elements and queries fulfilling those requirements. Requirements and (candidate) data elements are expressed as knowledge artefacts. In order that queries can be linked to these artefacts a query interface artefact should be developed to provide a shared artefact between knowledge and technical areas. The query interface element should be described using the data elements and candidate data elements also in the interface layer. The interface artefact must make visible:

- The data elements input as parameters
- The data elements used or accessed in the query
- The logic of any expressions used in the query
- The data elements output

Query expressions must be interpretable by knowledge modellers, and so an expression rendering should be defined which can be used by knowledge modellers. Existing LRA tooling is UML-based, and so a UML diagram type should be used for this representation. The activity diagram is considered to meet the requirement to represent queries. This rendering should be created through a transform from technical artefacts.

QRY 40 A transformation from technical artefacts forming a query to an interface artefact including a UML activity diagram forming **SHOULD** be available.

QRY 41 A query interface artefact **MUST** be available in order to represent the requirements specified in the knowledge space for a particular domain.

4.4 Query Construction

4.4.1 Logical Query Syntax

Any Logical Query Model must enable the construction of a declarative statement structure, e.g. `SELECT <query result> FROM <query input> WHERE <query conditions>`. This follows the pattern common to query implementation languages.

QRY 42 A query **MUST** be able to specify the:

- identity or location of the record component attribute to which it applies;
- the logical comparator used to test the value of the attribute for

- inclusion in the result; and
- value of the predicate.

4.4.2 Query Input

This document uses the term *query set* to refer to the set of patient records on which a query operates. To support the definition of query sets using a combination of results of other queries, the query set of a query needs to be able to be defined in terms of a combination of other query results.

QRY 43 Queries SHOULD allow definition of query sets based on results of other queries combined with basic set operations.

A consistent representation of the care record information model is required for queries to be able to operate. The Care Components Model describes an information model for care records, but does so by specifying a model for a care record extract, containing part or all of the health record information extracted from an EHR Provider system for the purposes of communication. As queries need to be run against complete care records, a model abstraction representing a patient or service user's complete EHR is required. The model of encoding information within the record is well specified and so the RECORD_COMPONENT and associated classes and the Participations model should be used within the EHR unchanged.

QRY 44 A model abstraction representing the complete EHR of an individual patient or service user MUST be available against which to design and specify queries. This COULD be realised by an appropriately constrained instance of `Ira.technical.en13606.extract.EHR_EXTRACT`.

QRY 45 The EHR model MUST use the same model for representing clinical statements and associated auditing, attestation and participant identification as the Care Components Model.

The EN13606 versioning and revision mechanism is intended for additions usually to be made by the original author within a short time frame, and not for recoding an evolving clinical story, and so only a very small number of queries for medico-legal purposes will require access to previous EHR versions.

QRY 46 Queries MUST, by default, be run against the latest version of an EHR.

4.4.3 Query Result

Although the result of a query is unknown before it is run, a user defining a query must be able to predict the structure and type of the results, and so query specifications must include this.

QRY 47 Queries MUST specify their result type.

The requirements identified earlier indicate that a query result may return any of the following:

- one (or possibly more) extracts each containing part of the EHR of a patient or service user (i.e. directly in support of individual patient or service user care); or
- one or typically more extracts each containing the complete EHR of a patient or service user (i.e. cohort generation); or

- the aggregate result of a population or cohort study (e.g. indicators and data collections).

The EHR extract may consist wholly or partly of generated content such as a collection of records assembled from one or more encounters with a given patient or service user e.g. a discharge summary or problem list; or a collection of specified elements such a set of one or more measurements recorded during a single encounter; or a computed value such as a BMI from height and weight. Such content is distinguished from original record content using the RECORD_COMPONENT.synthesised attribute.

QRY 48 A query result SHALL contain either one or more EHR_EXTRACT instances each containing part or all of the EHR of an individual patient or service user or one or more aggregate result instances.

It is envisaged that a combination of reusable filter queries could be used as the query set for application-specific queries which will select only the data items needed for a particular purpose. Application-specific queries will only rarely be used as the input to other queries.

4.4.4 Meaning-Based Retrieval

A *conformsTo* function is specified within the Terminology Binding Technical Specification [9] for testing a SNOMED expression against an Expression Constraint, and implementation of this is the required mechanism for testing meaning within queries.

Expression Constraints are currently referred to using UUIDs, and are to be stored as files identified by this UUID in a managed repository. However, the specification notes the possibility of these IDs moving to SCTIDs. It is also noted that it the identifier might be replaced with the full text of the constraint. The various parameter types can most easily be handled by overloading the function.

QRY 49 Data type CD.CV.SCT MUST implement the logical comparator function *conformsTo(constraintRef: UUID, effectiveTime: TS): BL* which accepts a UUID that uniquely identifies the constraint expression constraint and an optional timestamp specifying the version.

QRY 50 Data type CD.CV.SCT SHOULD implement the logical comparator function *conformsTo(constraintRef: II, effectiveTime: TS): BL* which accepts a SNOMED CT instance identifier that uniquely identifies the constraint expression constraint and an optional timestamp specifying the version.

QRY 51 Data type CD.CV.SCT MUST implement the logical comparator function *conformsTo(constraint: ST): BL* which accepts an instance of a SNOMED CT constraint expression.

QRY 52 Query implementations MUST handle dereferencing of expression constraints through retrieval from the constraint repository.

These terms MUST use the constraint format specified in [9]. It MUST be possible to express these terms using the Human Readable Grammar although these MAY be persisted in any form.

A requirement to support constructor bindings has been identified above. The Technical Specification states that:

A *constructor binding* can be represented using the SNOMED CT compositional grammar with the individual concept identifiers replaced by references to relevant data points in the Care Components model.

This will be assumed to include the extended compositional grammar. This enables the concept of constructor bindings to be extended to outputting expression constraints.

A constructor binding model requires further analysis once the full range of use cases has been identified. One potential application of constructor bindings that warrants further investigation is their use in testing equivalence between structural and terminological representations within LRA models.

In order that they can be used within examples in this document, the model described in appendix A uses simple string substitution based on the C/Java formatted string (sprintf) method. This provides operations for substituting placeholder tokens in Expression Constraints and SNOMED Expressions with SNOMED Expressions passed as parameters. This will mean that the 'template' expression or constraint may not be a valid expression/constraint.

4.4.5 Query Metadata

The query metadata will provide a link to the LRA Knowledge Model Artefacts relating to the query. The metadata for queries should use the same metadata model as used for maintaining information on other LRA artefacts. However, this is currently not specified, and so an interim approach has been developed. Example queries in this document are documented and annotated using specially formatted comments, using a format and tooling based on Javadoc. This means that the metadata is visible and searchable with the current limited tooling.

QRY 53 Queries SHOULD use a common LRA metadata standard for storing, searching and accessing metadata on specific queries.

4.4.6 Enumerations

Many of the reference model class attributes are typed as an enumeration and these are often important in retrieval. Reference enumerations are defined in the *datatypes::uml::derived* package. It is the named meaning of the enumeration that is significant rather than its underlying code or value, therefore:

QRY 54 Queries MUST be able to query using the enumerated values of record component attributes.

4.5 Query Logic

Rather than using a SQL-like 3-value logic, the LRA query logic is complicated by a range of null flavours in use which query logic must be able to handle appropriately. This is to reflect the need for handling incomplete data which will be encountered in EHRs. The nullFlavor attribute is defined on the ANY datatype, the base of all ISO 21090 types, and so all LRA datatypes can carry a nullFlavor unless otherwise

constrained. The ANY datatype defines the isNull() and notNull() operations which are required to be implemented by the query mechanism:

QRY 55 Queries SHALL be able to test if an instance of ISO 21090-derived LRA datatype is a nullFlavor.

The datatypes specification gives guidance on performing operations on nullFlavored types, and specifically compares this to the OCL model. Operations on nullFlavors must still apply, though the result of these operations will be some flavor of null: as the standard points out, this behavioural similarity is why the property is named 'nullFlavor'. In handling nullFlavors, these operations must take into account the semantics of the particular nullFlavor. This guidance is summarised as:

- If the operation takes no parameters, it will return a nullFlavor. Usually the nullFlavor will be NA, but other nullFlavors may be appropriate, depending on the semantics of the nullFlavors. When performing operations upon null values, the semantic meaning of the nullFlavor SHALL be considered.
- If the operation takes parameters, and any of the parameters are a UML/OCL null, the result will be a UML/OCL null
- If the operation takes parameters, and any of the parameters are a nullFlavor, the result of the operation will be the first common generalisation of the two flavours

QRY 56 Where appropriate, query operators MUST accept ISO21090 datatypes with null flavors, and follow the behaviour for returning a null flavour defined in the datatypes specification.

Querying for the absence of data or negatives might be particularly problematic, both from a records-keeping perspective where the lack of a recorded finding of a condition for example does not imply that a condition does not exist, and as the EHR being queried cannot practically be assumed to be a complete record. Facts that are not specifically negated in the care record should therefore to be considered unknown rather than false.

4.6 Temporal Query

Time is an important variable in healthcare and therefore requires particular attention here. Temporal expressions exist in a number of places within the LRA Care Record Model. Timestamps held against audit and attestation information as well as the creation and constraint times for the *EHR_EXTRACT* are relatively unproblematic. However, the temporal information within the clinical content is more so. This is primarily because temporal expressions can potentially exist in four different places in the record:

- As a time point or period within the *ELEMENT.obs_time* attribute, which represents the time at which the statement represented by the *ELEMENT* actually occurred or was true, if different from the session time of the *COMPOSITION*.
- As a time point or period within the *COMPOSITION.session_time* which represents the time during which the clinical encounter or documentation session occurred.

- As the | temporal context | attribute of a SNOMED CT expression within the *meaning* attribute of an *ELEMENT*, which represents the relationship between the time of occurrence of a | clinical finding | or | procedure | and the time when the information was recorded.
- Potentially as a time point, period, series or duration in the *value* of an *ELEMENT*, representing the data value of that *ELEMENT*.

Guidance and rules for the usage of these attributes when representing a given clinical situation are given in *SNOMED CT Bindings for Common Recording Patterns* [20] on which this section draws. The modelling rules and recommendations relating to temporal expressions can be summarised for each of the preceding items as:

- The *obs_time* field represents the time at which or period during which the clinical statement indicated by the *meaning* expression was true and had the value represented by the *value* field.
 - In the case of goals, risks or expectations, the *obs_time* represents a future time at which the specified *value* is intended or expected to apply.
- The *session_time* attribute of the containing *COMPOSITION* provides a reference point if the *obs_time* is omitted.
 - The temporal context indicates a time relative to the *session_time* (e.g. current, immediate or past).
- The SNOMED temporal context given in *meaning* should primarily be used to indicate that an entry contains retrospective (e.g. past history), without specifying an actual date of an occurrence.
 - If the entry specifies a date or time for a | clinical finding | or | procedure |, the | temporal context | need not be stated and can be ignored.
 - The default value for this attribute is: 410512000 | current or specified |.
 - Permissible values for this attribute are the descendant subtypes of the 410510008 | temporal context value | concept.
- The *value* field should not be used for points in time or specific periods of time.
 - Data items such as 'date of last menstrual period' or 'expected date of delivery' could be considered as properties with date values.
 - Date and time based properties should instead be represented using the *obs_time* field of a relevant finding.
 - Use of the | finding context | attribute allows future expected, planned or intended dates to be captured using the same approach.
 - It is possible that this approach may change in cases where a clear principled rule distinguishes time and date properties from the time and date of a clinical finding.
 - This advice does not preclude the use of the *value* field to represent duration as a quantity using time units.

These rules are manifested in a set of patterns for common clinical information, some of which include retrieval examples. However, representation of events and episodes

has not been in the scope of the current modelling work, which precludes a broader temporal abstraction being defined.

The ISO standard *EN12381 Health informatics — Time standards for healthcare specific problems* [12] specifies a set of representational primitives and semantic relations required for representation of explicit time-related expressions in care records. It is assumed that this standard fully describes the functional requirements for representation of temporal aspects of clinical information and hence the requirements of the LRA. This assumption will be used here as the basis for the requirements for queries based on temporal relations.

The LRA query mechanism must have as a minimum requirement the capacity to order situations, events and episodes in three major ways:

- by relating situations to a calendar;
- by relating situations to “reference” situations;
- by relating events together in “before- and after-” chains.

These situations, events and episodes are specified by one of a number of temporal expression types. Representations of each of these types are possible using the ISO21090 temporal datatypes as shown in table 2 below:

EN12381 Primitive	ISO 21090 Datatype
Time point expression, TP	Timestamp, TS
Time interval expression, TI	Interval of time, IVL<TS>
Frequency expression, FQ Rate expression, RT Timeseries expression, TS	Periodic interval, PIVL or Event based interval, EIVL or Bounded periodic time interval, GTS.BoundedPIVL or QSET<TS>
Duration expression, DR	Physical quantity of time, PQ.Time

Table 4: Mapping from ISO 12381 temporal expression type to ISO 21090 representation

Simple arithmetic and comparative operators are defined on many of these types within the LRA datatype specification. However, the comparators defined on these types are insufficient to meet the requirements of ISO 12381, which introduces a requirement for a set of temporal relation operators.

4.6.1 Temporal Relations

The temporal relations defined in [12] are shown in the figures below:

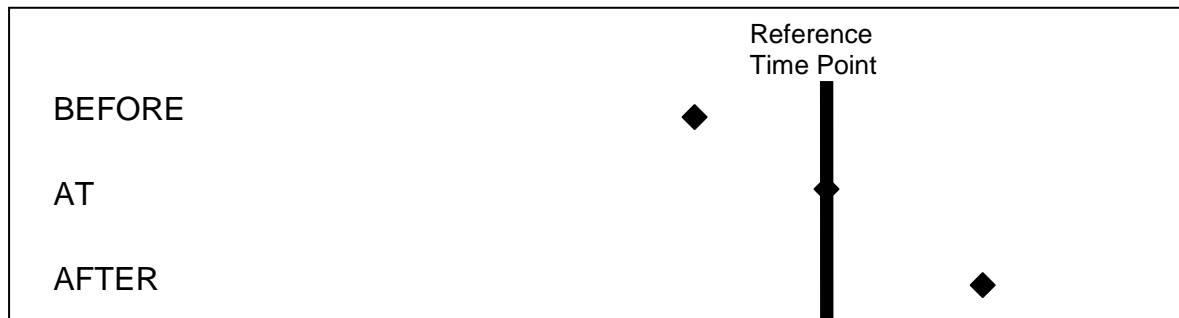


Figure 1: Time point to time point relations

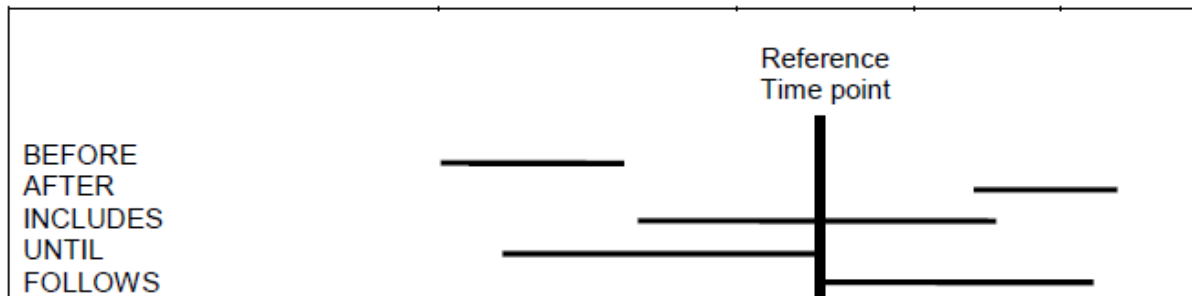


Figure 2: Time interval to time point relations

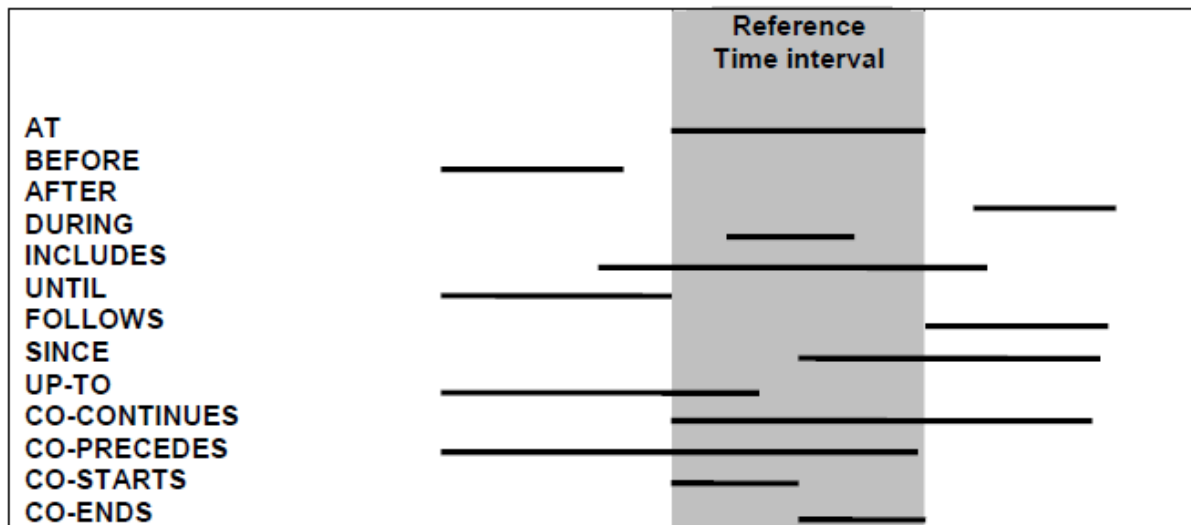


Figure 3: Time interval to time interval relations

These relations can be mapped to GELLO, which assumes that the reference model provides Allen's temporal relations on time intervals [19].

EN12381 Temporal Comparator	Allen Relation
AT	Equals
BEFORE	Before
AFTER	After
DURING	During
INCLUDES	Contains

UNTIL	Meets
FOLLOWS	Met-by
SINCE	Overlapped-by
UP-TO	Overlaps
CO-CONTINUES	Started-by
CO-PRECEDES	Finished-by
CO-STARTS	Starts
CO-ENDS	Finishes

Table 5: Mapping of ISO 12381 comparators to Allen relations

As events and episodes have not yet been modelled within the LRA these comparators will be used only on absolute values of the temporal datatypes shown in Table 2 above. Comparison between two temporal types must be made at the lowest common level of precision.

QRY 57 Queries MUST provide the temporal comparators defined in ISO 12881 for the absolute temporal datatypes TS and IVL(TS).

Comparison will need to be made with the current date/time, and so this must be available as a reference time point. There is an existing requirement for queries to be able to create new instances of TS and its specializations based on a literal date value, and so this will be extended to require that the current time instance is available. This might be implemented as a separate function or the default value of the TS constructor.

QRY 58 Queries MUST be able to create an instance of TS containing the current point in time for use as a query term.

Comparison of only particular date or time elements may need to be made regardless of the specific point in time.

QRY 59 Queries MUST be able to compare individual time and date components of a timestamp.

The EIVL (Event-Related Periodic Interval of Time) datatype class specifies a periodic interval of time where the recurrence is based on activities of daily living, and can hold an offset for an event. EIVL could only be meaningfully compared with other EIVLs. While some relations can be discerned between these values (*after breakfast is after before lunch*, for example) they are too complex and inconsistent to model or query and so no additional relations will be required for this type.

4.7 Functions and Operations

Operations are defined on many of the ISO 21090 datatypes and there is an existing requirement in the LRA Datatypes Specification that an information processing entity (such as the query mechanism) shall define operations on ISO 21090 based datatypes. The operators specified in the LRA Datatypes Specification are required to be implemented as specified:

QRY 60 The LRA query mechanism MUST implement the operators defined on

the ISO 21090 datatypes used in the LRA.

The operations defined in ISO 21090 deliver some basic functionality but are insufficient to provide the required query functionality. The operators defined on these types will therefore be extended beyond the current datatypes specification to meet the additional requirements specified below.

Two primitive forms of datatype are in use in the LRA. Complex ISO21090-derived datatypes are used in the reference model classes. These contain attributes which may themselves be complex or can be simple types derived from the UML/OCL kernel. In some cases, such as the UML integer and ISO 21090 INT, the types are effectively equivalent, although there is no mapping between these. In order to maintain the semantics of these data, data will not be considered or compared at a level below the ISO 21090 datatype, with the semantics as defined in the datatypes specification. Accordingly query operations will accept only ISO 21090 datatypes, and operations on UML primitives will be disallowed within queries.

QRY 61 Query operations MUST accept ISO 21090 datatypes as parameters and MUST return an ISO 21090 datatype.

QRY 62 Queries MUST NOT make use of operations on UML types.

The ISO 21090 collection datatypes (*GLIST*; *SLIST*; *COLL* and its subtypes) are currently defined by the LRA Datatypes Specification as out of scope. However, the structures defined in EN13606-1 can contain the record of only a single patient and so SET will need to be used to contain the records or results for more than one patient. Collections of values will also be required for calculation of aggregate values, which will require a broader range of collection datatypes to be brought into scope.

There is a need to allow literal values to be used as query predicates. These instances of datatypes are to be used as query terms only and will not be persisted as part of the query. As this is the case these constructed values need hold enough information in their attributes to carry their value and be valid instances of their type, and so no optional attribute should be provided unless it affects the value of the item. This is to be interpreted as whether an attribute affects the *T.equals(T)* relation. Appropriate system-specific defaults can be used for some attributes such as language. It is not possible to create an instance of an abstract datatype such as QTY. No use case has yet been found for creating literal null flavors as query terms. However creating instances of some classes (such as FINDING_OBSERVATION_ELEMENT) requires an attribute (such as FINDING_OBSERVATION_ELEMENT.value) to be set to the null flavour, and so a requirement exists to support this.

GELLO requires that the mechanisms for creating variables as instances of classes are delegated to the underlying data model in order to preserve GELLO as a side-effect-free language. This is specified as following the pattern of a class factory for each type taking with the value specified in a single string argument (note that the examples provided in the specification are inconsistent with this pattern). While this approach may be suitable for basic value types such as string (ST) or physical quantity (PQ) it is insufficient for creating instances of more complex classes such as RECORD_COMPONENTS. In the LRA query model therefore a constructor is required to be defined for each type allowing multiple typed parameters.

QRY 63 Queries MUST allow literal values of concrete datatypes to be used as

query terms.

QRY 64 Queries MUST provide a mechanism for creating instances of null flavors.

While the *ST.toReal*, *ST.toInteger()* and *ST.toTimestamp()* operations are defined and could be used as mechanisms to create instances of the respective types from string literals it may be useful to provide shortcuts to these constructors defined alongside other constructors.

4.7.1 Mathematical and statistical functions and operators

As stated above the primary purpose of query in the LRA is information retrieval rather than analysis. Accordingly the operators and functions specified as required are those which will be used in retrieval.

Mathematical operators must be able to handle the LRA numeric datatypes INT and REAL as well as PQ and MO, which add a requirement for these operators to handle units. CO are numeric in a sense: coded ordinals in EN21090 represent scale results which have an implied order. As recognised in the datatypes specification, this is a difficult subject as the nature of coded ordinals is variable. Interpretation of an assessment scale result as a continuum does depend both on the precise nature of the scale and on the circumstances and the use case for which that interpretation is being used. Thus in the LRA coded ordinals are used to meet specific analysis use cases, and must be permitted as parameters to mathematical functions.

Mathematical and physical constants which may be required are accessible as UCUM expressions, and so no further requirement for these is specified.

Aggregate functions required here operate on a Collection of numbers and return the appropriate numeric type.

Function Name	Description
exp(x: T):T	Returns the value of e raised to the specified power.
log(x: T):T	Returns the natural (base e) logarithm of the argument.
log10(x: T):T	Returns the common (base 10) logarithm of the argument.
pow(x: T):T	Returns the first argument raised to the power of the second argument.
sqrt(x: T):T	Returns the square root of the argument.
count(x: T, c: COLL (T)): INT	Returns the number of occurrences of object in a collection.
sum(c: COLL(T)):T	Returns the total of all the elements in a collection.
avg(c: COLL (T)):T	Returns the average (arithmetic mean) of the numerical elements in a collection
stdev(c: COLL (T)):REAL	Returns the standard deviation of the numerical elements in a collection.
variance(c: COLL (T)):REAL	Returns the variance of the numerical elements in a collection

median(c: COLL (T)):T	Returns the median of the numerical elements in a collection
mode(c: COLL (T)):T	Returns the most frequently occurring value in a collection.
max(c: COLL (T)):T	Return the largest value in the collection c.
min(c: COLL (T)):T	Return the smallest value in the collection c.

Table 6: Mathematical function requirements

QRY 65 Queries SHALL support the mathematical functions described in Table 6: Mathematical function requirements.

Early analysis suggests that the functions required above are available or can be implemented in likely implementation languages. More advanced analysis than presented in the use cases above may be required in a small number of cases. It may therefore be useful to have the option of extending the built in functions or delegate the necessary additional functions to the query framework itself. Queries relying on extensions may not be interoperable or implementable.

QRY 66 The query mechanism MAY define a method of extending the built in function set.

4.8 Query Tooling

In order to support development a set of tools could be developed to provide a query development environment alongside the repository. Tooling requirements for terminology binding are specified in section 6 of the Terminology Binding Technical Specification [9]. As Expression Constraints are in use in both terminology bindings and queries, this same set of tools should be used in developing both artefact types. Outside terminology, query development tooling might include in addition the following components.

If OCL is selected as the query expression language then components from existing OCL tools could be shared with the LRA constraint editing mechanism.

4.8.1 Query Repository

Constraints can be attached to their respective UML classifier as a note with a <<constraint>> stereotype as shown in the Appendix A. If query expressions are implemented as constraints would persist the query expression alongside query classes within the LRA model.

4.8.2 Query Builder

A query authoring tool would include the following components, presented in increasing degree of complexity:

- Editor
With a formal query grammar in place a query editor could be provided to support the coding of queries. This could offer code colouring and suggestions, and could be based on existing tools.
- Syntax checker;

A syntax checking component would check that query expressions are well formed against the query grammar.

- Validator

This would ensure queries are valid against the LRA model by ensuring that operations and attributes referred to in the query exist in the model, and that queries used as the input query set exist.

- User interface binding

UI binding could be used to create examples of queries in the context of a realistic clinical system. This could be used in the clinical validation of queries.

4.8.3 Example EHR

A test execution environment with an example EHR containing synthesised data would enable the generation of query output examples.

QRY 67 A query development environment MIGHT be developed to support the authoring and testing of queries.

5 Requirements Summary

The requirements identified inline in the sections above are collated in the table below:

QRY 68	A Query MUST NOT allow the data of a care record to be modified.
QRY 69	Queries MUST be uniquely identified.
QRY 70	Queries SHOULD be able to be persisted in a searchable repository with appropriate access control restricting reading editing and authoring.
QRY 71	Queries SHOULD have a name and/or short human-readable rendering that can be displayed in response to a search.
QRY 72	Queries MUST have some human-readable format such that they can be reviewed against requirements to determine if they are suited to a particular use or reuse. Complex queries such as those made up of several sub-queries MAY have some level of drill-down so that more or less detail can be viewed.
QRY 73	Query expressions SHOULD be able to include notes or comments inline clarifying the usage or operation of a particular section.
QRY 74	The query repository SHOULD support versioning, change control and tracking of query authorship and modifications.
QRY 75	Queries MUST allow specific terms to be marked as parameters which may be set at the point at which they are run.
QRY 76	Queries MUST allow their input to be the output of another query.
QRY 77	The output type of all queries MUST be clearly specified.
QRY 78	A query development and assurance process MUST be available for use where query development is required.

QRY 79	Query metadata MUST record the development status and assurance process stage of a query.
QRY 80	Query metadata MUST record the point at which the query was reviewed with respect to the care record model. Where the model is subsequently changed this MUST be reviewed against dependant queries.
QRY 81	Queries MUST be available in some representation accessible to clinicians for review.
QRY 82	Query metadata MUST enumerate any assumptions made by the query author for the purpose of the query.
QRY 83	Query metadata MUST separately record instances of query reuse.
QRY 84	Where queries are reused, this reuse MUST be included within the query assurance process
QRY 85	Queries MUST be able to be defined over a population or specific patient records.
QRY 86	Where they have been used directly in the care of a particular patient, results of specific queries MUST be able to be stored within the care record of an individual patient alongside clinical entries.
QRY 87	Collections retrieved from care records or returned from operations MUST be able to be sorted by one or more attributes.
QRY 88	Collection sorting MUST be based on natural order and must apply to datatypes with a min operation defined.
QRY 89	All other types MUST be partially ordered based on the equality semantics defined in the specification. Ordering of items with equal value is arbitrary, and MUST NOT be relied upon within queries.
QRY 90	Queries MUST allow the sort order of nulls to be specified. When sorting ascending order and not otherwise specified, queries SHOULD default to sorting null flavors after non-null flavored data.
QRY 91	Where query results represent a new clinical statement through calculating, generalising or inferring based on existing care record information, the query mechanism MUST be able to store this as a clinical statement in the care record, with links to the information on which the result is based.
QRY 92	Where a query creates clinical statements, these statements MUST be marked as attested by the query system, along with other appropriate audit information.
QRY 93	A query abstraction MUST define any aggregations and the logic of calculations and inferences based on record data selected by a query.
QRY 94	Queries SHALL be able to return the entire care record for a patient matching a set of clinical and demographic characteristics
QRY 95	Queries MUST be able to express literal SNOMED CT Expressions as query terms.
QRY 96	Queries MUST implement the implies operation on CD as defined in the datatypes specification, such that where two SNOMED CT expressions

	have the same normal form they imply each other.
QRY 97	Queries MUST be able to test if an instance of a SNOMED expression conforms to a SNOMED Expression Constraint.
QRY 98	Queries MUST be able to represent SNOMED Expression Constraints as query terms.
QRY 99	Queries SHOULD be able to accept all serializations of Expression Constraints as query terms.
QRY 100	Queries SHOULD be able to construct SNOMED Expressions and Expression Constraints based on care record objects and query parameters.
QRY 101	Queries SHALL be able to express terms using literal UCUM expressions
QRY 102	Queries SHALL be able to return results in a specified unit where values are held or derived in a different, commensurate unit.
QRY 103	Queries SHALL provide a simple method for testing existence and type of component relationships.
QRY 104	Queries SHALL be able to test for equivalent component relationships types taking account of subsumption, inverses, symmetry and transitivity defined in the vocabulary.
QRY 105	Queries SHALL provide a simple method for accessing linked components, including selecting them by type.
QRY 106	A query MUST define its logic and output type but MUST NOT restrict implementation.
QRY 107	A transformation from technical artefacts forming a query to an interface artefact including a UML activity diagram forming SHOULD be available.
QRY 108	A query interface artefact MUST be available in order to represent the requirements specified in the knowledge space for a particular domain.
QRY 109	A query MUST be able to specify the: <ul style="list-style-type: none"> • identity or location of the record component attribute to which it applies; • the logical comparator used to test the value of the attribute for inclusion in the result; and • value of the predicate.
QRY 110	Queries SHOULD allow definition of query sets based on results of other queries combined with basic set operations.
QRY 111	A model abstraction representing the complete EHR of an individual patient or service user MUST be available against which to design and specify queries. This COULD be realised by an appropriately constrained instance of <code>Ira.technical.en13606.extract.EHR_EXTRACT</code> .
QRY 112	The EHR model MUST use the same model for representing clinical statements and associated auditing, attestation and participant identification as the Care Components Model.

- QRY 113 Queries MUST, by default, be run against the latest version of an EHR.
- QRY 114 Queries MUST specify their result type.
- QRY 115 A query result SHALL contain either one or more EHR_EXTRACT instances each containing part or all of the EHR of an individual patient or service user or one or more aggregate result instances.
- QRY 116 Data type CD.CV.SCT MUST implement the logical comparator function conformsTo(constraintRef: UUID, effectiveTime: TS): BL which accepts a UUID that uniquely identifies the constraint expression constraint and an optional timestamp specifying the version.
- QRY 117 Data type CD.CV.SCT SHOULD implement the logical comparator function conformsTo(constraintRef: II, effectiveTime: TS): BL which accepts a SNOMED CT instance identifier that uniquely identifies the constraint expression constraint and an optional timestamp specifying the version.
- QRY 118 Data type CD.CV.SCT MUST implement the logical comparator function conformsTo(constraint: ST): BL which accepts an instance of a SNOMED CT constraint expression.
- QRY 119 Query implementations MUST handle dereferencing of expression constraints through retrieval from the constraint repository.
- QRY 120 Queries SHOULD use a common LRA metadata standard for storing, searching and accessing metadata on specific queries.
- QRY 121 Queries MUST be able to query using the enumerated values of record component attributes.
- QRY 122 Queries SHALL be able to test if an instance of ISO 21090-derived LRA datatype is a nullFlavor.
- QRY 123 Where appropriate, query operators MUST accept ISO21090 datatypes with null flavors, and follow the behaviour for returning a null flavour defined in the datatypes specification.
- QRY 124 Queries MUST provide the temporal comparators defined in ISO 12881 for the absolute temporal datatypes TS and IVL(TS).
- QRY 125 Queries MUST be able to create an instance of TS containing the current point in time for use as a query term.
- QRY 126 Queries MUST be able to compare individual time and date components of a timestamp.
- QRY 127 The LRA query mechanism MUST implement the operators defined on the ISO 21090 datatypes used in the LRA.
- QRY 128 Query operations MUST accept ISO 21090 datatypes as parameters and MUST return an ISO 21090 datatype.
- QRY 129 Queries MUST NOT make use of operations on UML types.
- QRY 130 Queries MUST allow literal values of concrete datatypes to be used as query terms.
- QRY 131 Queries MUST provide a mechanism for creating instances of null flavors.

- QRY 132** Queries SHALL support the mathematical functions described in Table 6: Mathematical function requirements.
- QRY 133** The query mechanism MAY define a method of extending the built in function set.
- QRY 134** A query development environment MIGHT be developed to support the authoring and testing of queries.

Appendix A

This section presents an example query model and illustrative examples of some of the query types identified in the document. The examples use the UML model presented here and the Object Constraint Language (OCL) [25] to define the logic and operation of queries. OCL has the power of first order predicate logic but without its unintuitive syntax. Where OCL is insufficient to meet the requirements specified, this functionality could be delegated to the underlying implementation.

OCL is in use already within the technical model to specify constraints. Queries are, in essence, constraints on instances of a model, and so the use of a constraint language here also seems not inappropriate. While OCL may seem an utterly different paradigm from the existing and well-used query languages mentioned in section 3.2, OCL transforms and code generators for languages including query languages are used and available. The structure and capability of these likely implementation languages has been borne in mind where the query model has defined operations to extend the capability of OCL.

Example Queries

Note on examples: *The formalism used in this and further examples is for illustrative purposes only. This document does not presuppose its adoption as the query syntax, model, or presentation to be used by the LRA. The illustrative model in use in examples is detailed in Appendix A.*

Query Type	Clinical query
Source	Care Components Model specification
Question	Get the value of blood pressure observations and time taken recorded for an identified patient during the current episode.
Notes	
Code	<pre> package query::example context Q_get_patient_bp_against_time /** *Get the value of blood pressure observations and time taken recorded for an *identified patient during the current episode. *Episode is not currently modelled. Assumes episode exists as a *GENERAL_ACTIVITY_ELEMENT with an ops_time.high = null (i.e. ongoing), and *that observations have a link inContextOf this episode. *Assumes a valid BP reading MUST have both diastolic and systolic parts * *@author Aled Greenhalgh </pre>

```

*@version 0.1, 2009-08-19
*@param id An II containing patient identifier
*@return a Tuple containing systolic: PQ, diastolic: PQ, time: TS
*/
let bpSemantic: SnomedCtExpression =
  SnomedCtExpression("{463a0ca7-5ce1-11de-8a39-0800200c9a66}")
in
let systolicObservationSemantic : SnomedCtExpression =
  SnomedCtExpression("{463a33b8-5ce1-11de-8a39-0800200c9a66}")
in
let diastolicObservationSemantic : SnomedCtExpression =
  SnomedCtExpression("{463a33b7-5ce1-11de-8a39-0800200c9a66}")
in
let episodeSemantic: SnomedCtExpression = SnomedCtExpression("
  #episode semantic placeholder#
")
in
--assumes bpMeasurement has a link inContextOf an episode
--querySet should be a set of 1 EHR
def: solution: Set(TupleType(systolic: PQ, diastolic: PQ, time: TS)) =
  querySet.all_compositions->iterate(
    c: COMPOSITION;
    result: Set(TupleType(systolic: PQ, diastolic: PQ, time: TS))=|
    c.content.items->collect(bp|
      if (
        conformsTo(bp.meaning, bpSemantic)
        and
        links(bp)->select(episode|
          --bp observation is has a link in contextOf
          linkEquivalentTo(bp, episode, "inContextOf")
          and
          --the object inContextOf is an episode
          conformsTo(episode.meaning, episodeSemantic)
          and
          --the current time is within episode clinical time
          within(TS(), episode.obs_time)
        )->notEmpty()
        and
        links(bp)->select(pressureObs|
          --Has 2 parts
          links(bp, "HasPart")->size()=2
          and
          --
          links(pressureObs)->select(part|
            conformsTo(
              part.meaning,
              systolicObservationSemantic
            ).value
            or
            conformsTo(
              part.meaning,
              diastolicObservationSemantic
            ).value
          )->size() = 2
        )->notEmpty()
      )
    then
    let systolic:PROPERTY_OBSERVATION_ENTRY = links(bp)->select(pressureObs|
      links(pressureObs)->select(part|
        conformsTo(

```

```

        part.meaning,
        systolicObservationSemantic
    )
)
)first()
in
let diastolic:PROPERTY_OBSERVATION_ENTRY = links(bp)->select(pressureObs|
    links(pressureObs)->select(part|
        conformsTo(
            part.meaning,
            diastolicObservationSemantic
        )
    )
)first().value
in
result.including(
    Tuple{
        systolic: PQ = systolic.value,
        diastolic: PQ = diastolic.value,
        time: TS = systolic.obs_time
    }
)
)
)

```

Example 1**5.1.1.1 Example**

Query Type	Clinical query
Source	Care Components Model specification
Question	For a selected patient or service user, retrieve the most recent body mass index (BMI). This may be recorded explicitly or computed
Notes	
Code	<pre> package query::example context Q_patient_bmi /** *Get value of latest BMI, or if more recent hight and weight measurements *available then the calculated value. Height and weight measurements must have *been made in the same session. Assumes weight observations have not mbeen made *while pregnant * *@author Aled Greenhalgh *@version 0.1, 2009-08-19 *@param id NHS number of the patient */ --filter querySet to a specific patient record using another qry --SCT referring to BMI let bmiSemantic: SnomedCtExpression = SnomedCtExpression(" SnomedCtExpression(" </pre>

```

243796009 | situation with explicit context | :
{ 246090004 | associated finding | = << 60621009 | body mass index |
, 408729009 | finding context | = 36692007 | known |
, 408731000 | temporal context | = 410512000 | Current or specified |
, 408732007 | subject relationship context | = 410604004 | Subject of record | }
")
in

--SCT referring to body weight
let bodyWeightSemantic : SnomedCtExpression =
  SnomedCtExpression("
243796009 | situation with explicit context | :
{ 246090004 | associated finding | = ^ 971000000138 | Weight |
, 408729009 | finding context | = 36692007 | known |
, 408731000 | temporal context | = 410512000 | Current or specified |
, 408732007 | subject relationship context | = 410604004 | Subject of record | }
")
in

--SCT referring to height
let heightSemantic : SnomedCtExpression =
  SnomedCtExpression("
243796009 | situation with explicit context | : ^ 1371000000138 | Height |
{ 246090004 | associated finding | = ^ 971000000138 | Weight |
, 408729009 | finding context | = 36692007 | known |
, 408731000 | temporal context | = 410512000 | Current or specified |
, 408732007 | subject relationship context | = 410604004 | Subject of record | }
")
in

--the most recent bmi observation for the patient
let bmiFindingComposition : COMPOSITION =
  querySet.all_compositions->select(c|
    --compositions containing a bmi observation
    c.content.items.conformsTo(meaning,bmiSemantic)->notEmpty()
  )->sortedBy(c|
    --sort by the maximum of session_time and obs_time
    c.session_time.max(
      c.content.items.conformsTo(meaning,bmiSemantic)->first().obs_time
    )
  )->last()
in

--the last BMI observation in the session
let bmiFinding : PROPERTY_OBSERVATION_ELEMENT =
  bmiFindingComposition->select(c|
    c.content.items.conformsTo(meaning,bmiSemantic)->last()
  )

--most recent session containing both height and weight observations
let heightAndWeightComposition : COMPOSITION =
  querySet.allCompositions->select(c|
    c->select(conformsTo(content.items.meaning, bodyWeightSemantic).value)->notEmpty()
    and
    c->select(conformsTo(content.items.meaning, heightSemantic).value)->notEmpty()
  )->sortedBy(c|c.session_time)->last()
in

def: solution: PROPERTY_OBSERVATION_ELEMENT =
  --if height and weight observations are more recent than bmi observations

```

```

if after(
    heightAndWeightComposition.session_time,
    --the maximum of session_time and obs_time
    bmiFindingComposition.session_time.max(
        bmiFindingComposition.session_time,
        bmiFinding.obs_time
    )
)
then
    --calculate the bmi
    let height: PQ = heightAndWeightComposition.content.items->select(item|
        conformsTo(item.meaning, heightSemantic).value
    )->last()
    in
    let weight: PQ = heightAndWeightComposition.content.items->select(item|
        conformsTo(item.meaning, bodyWeightSemantic).value
    )->last()
    in
    let bmiValue: PQ = weight / pow(height,2) in
    let bmiSemanticLiteral = CD.CV.SCT("
{60621009|body mass index|,
408729009 | finding context | = 36692007 | known |,
408731000 | temporal context | = 410512000 | Current or specified |,
408732007 | subject relationship context | = 410604004 | Subject of record | }
")
    in
    let bmi: PROPERTY_OBSERVATION_ELEMENT =
        ELEMENT(bmiValue, weight.obs_time, bmiSemanticLiteral,
            null, null, null, null, null, null)
    in
    -- return the bmi element as the result
    bmi
else
    --result is the bmi finding
    bmiFinding

```

Example 2

Query Type	Clinical query
Source	Care Components Model specification
Question	For a selected patient or service user, retrieve the medical history
Notes	Retrieves only items explicitly recorded
Code	
<pre> package query::example context Q_history_of_disease_by_parameter /** *An inference query returning a recorded history of the concept identified by *the parameter named 'finding'. * *@author Aled Greenhalgh *@version 0.1, 2009-08-19 *@param finding A string identifying a snomed concept referring to the disease *to find a history for *@return a set of ENTRIES representing findings of history, linked to explicit *findings and previous diagnoses of ?disease */ </pre>	

```

--A string containing the constructor expression.
--C is the token that will be substituted. The format string 'C' formats the
--CD appropriately.
let diseaseMatchString : String = "
    %C :
    { 408731000 | temporal context | << 410513005 | past |
      , 408729009 | finding context | << 410515003 | known present }
"
in

--the concept to find a history of, passed as a CD
let disease : CD = self.parameters->select("name = finding").first()
in

--put the concept passed as a param into
let history : SnomedCtExpression = SnomedCtExpression(diseaseMatchString, disease)
in

--solution is the set of all EHRs with an element with meaning the constraint
--defined above
def: solution: Set(EHR) =
    querySet->select(allCompositions.content.items.conformsTo(meaning, history))

```

Example 3

Query Type	Research
Source	Kalra et al [7]
Question	Find the age and gender of patients who have been diagnosed with Hodgkin's disease, where the initial diagnosis occurred between the ages 50 and 70 inclusive.
Notes	Currently assumes that any finding of Hodgkin's is a diagnosis.
Code	<pre> context HodgkinsQuery::solution: Set(EHR) /** *Patients who have been diagnosed with Hodgkin's disease, where the initial *diagnosis occurred between the ages 50 and 70 inclusive. * *@author Aled Greenhalgh *@version 0.1, 2009-08-19 */ derive: --time period representing age between 50 and 70 yo. let diagnosisPeriod: IVL<TS> = IVL(querySet.demographic_information.entity.birthTime + PQ(50, "[years]"), querySet.demographic_information.entity.birthTime + PQ(70, "[years]")) in --TODO:this is not the correct semantic let hodgekinsSemantic: CD.CV.SCT = Factory.SnomedCtExpression("<< 1234324 Hodgkins (finding)") in --finding of hodgekins </pre>

```

let hodgekinsDiagnosis: FINDING_OBSERVATION_ENTRY =
  querySet.all_compositions.content.items-> select(item|
    conformsTo(item.meaning, hodgekinsSemantic)
  )->first()
in
--return the solution
querySet.select(
  --there is a diagnosis of Hodgkins
  hodgekinsDiagnosis->notEmpty()

  and
  --the first diagnosis is within diagnosisPeriod
  within(hodgekinsDiagnosis.obs_time,diagnosisPeriod)
)

```

Example 4

Query Class Model

This specification was generated from the UML source model using the RTF Class Model Specification template (v1.0.4) at 2009-11-27 12:24:36.

Package Ira.technical.query

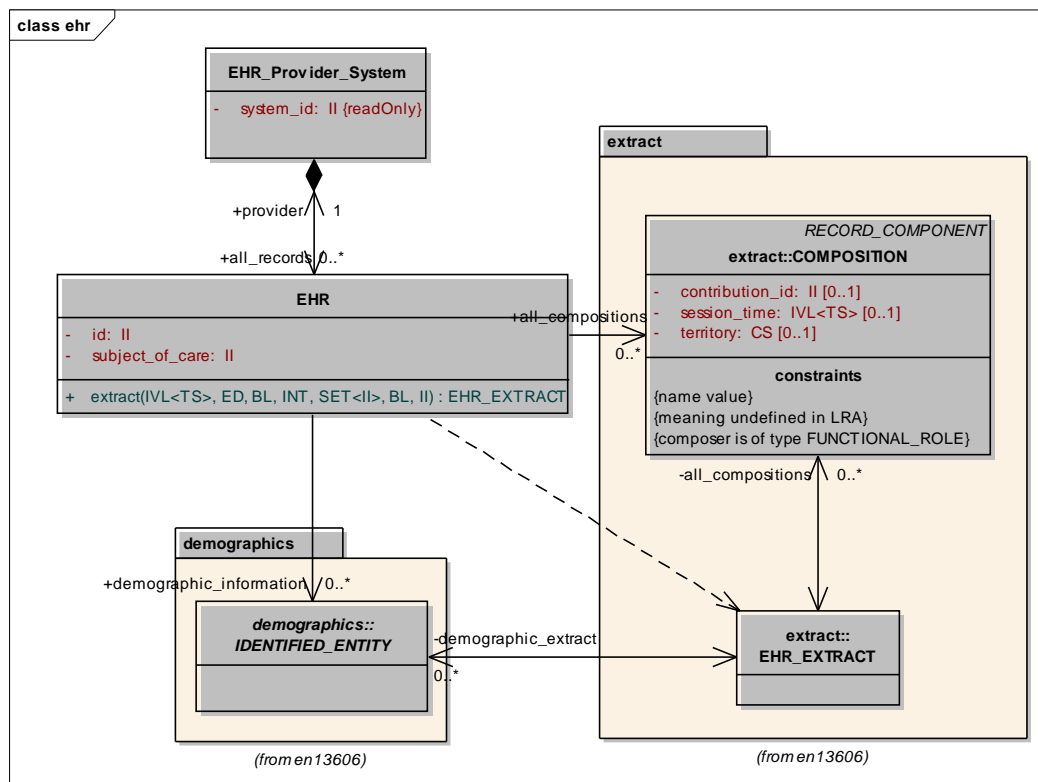


Figure 1: ehr

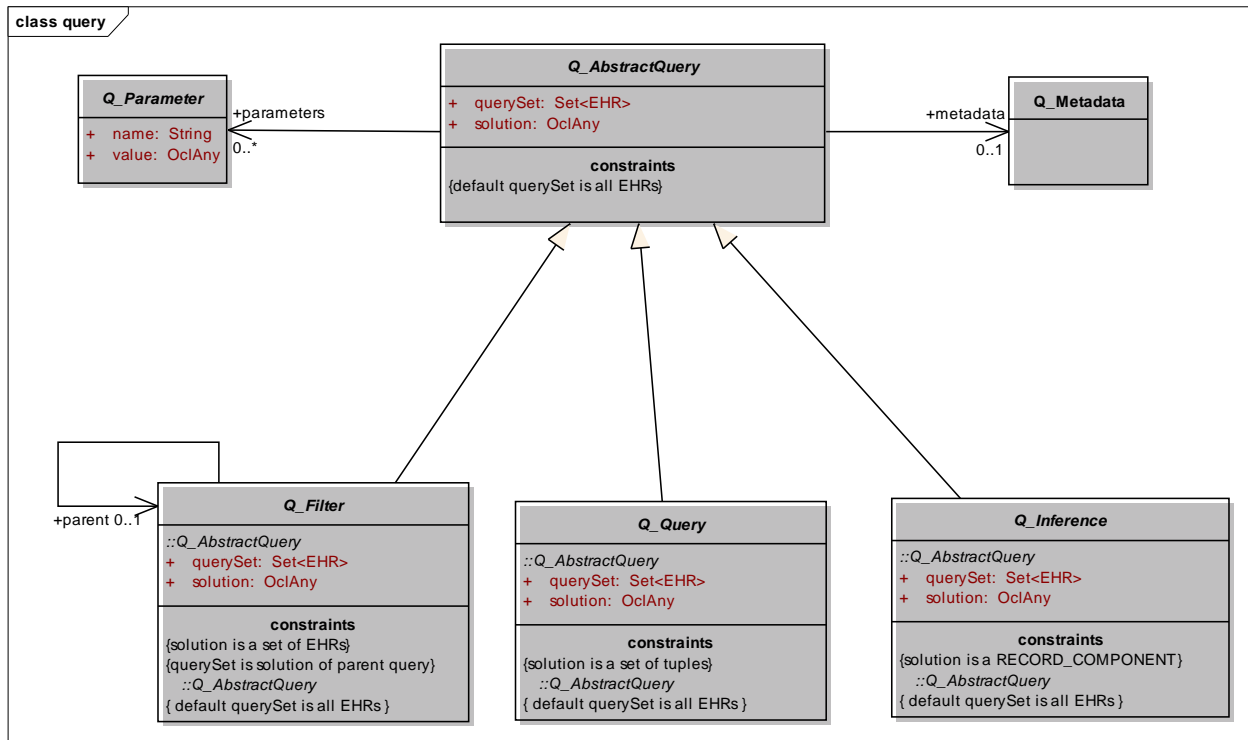


Figure 2: query

Class EHR*Specialises:**Realises:*

This class represents a complete Electronic Health Record, relating to a single person, identified by subject_of_care. This follows the model of the EN13606 EHR_EXTRACT, but with all notions of an extract removed. A set of all instances of EHR form the default queryset of LRA queries.

Attributes

Attribute	Description
id : II [1..1]	The id of each EHR. This id is unique within each EHR system.
subject_of_care : II [1..1]	Unique identifier for the subject of the EHR.

Relationships

Relationship Type	Source	Target	Description
Association	all_records EHR 0..*	provider EHR_Provider_System 1	The EHRs make up the EHR_System
Dependency	EHR	EHR_EXTRACT	The EHR effectively implements EHR_EXTRACT, with

Relationship Type	Source	Target	Description
			extract-specific attributes removed.
Association	EHR	demographic_information IDENTIFIED_ENTITY 0..*	Demographic information relating to the subject_of_care. Demographic information is provided by the LRA Participations model.
Association	EHR	all_compositions COMPOSITION 0..*	The records of clinical encounters making up the clinical content of the EHR are accessed through this association. extract::COMPOSITION and all associated classes are as described in the care components model.

Class EHR_Provider_System

Specialises:

Realises:

A class representing a concrete EHR system.

Attributes

Attribute	Description
system_id : II [1..1]	The unique identifier of the EHR system.

Relationships

Relationship Type	Source	Target	Description
Association	all_records EHR 0..*	provider EHR_Provider_System 1	The EHRs make up the EHR_System

Abstract Class Q_AbstractQuery

Specialises:

Realises:

An abstract class representing an LRA query. Implementations specialize one of Q_AbstractQuery's specializations adding constraints on querySet , parameters and solution.

This class also defines operations to be realised by the query implementation. For the sake of organisation these are separated by function into nested classes.

Attributes

Attribute	Description
querySet : Set<EHR> [1..1]	The data over which the query runs. Equivalent to the SQL FROM clause, and can be the result of another query. The querySet presumes the query runs against an EHR which conforms to the EHR_EXTRACT model. For the purpose of providing an initial value, the default is all EHRs
solution : OclAny [1..1]	A derived attribute containing the solution to the query.

Relationships

Relationship Type	Source	Target	Description
Nesting	Q_StringOperation	Q_AbstractQuery	
Generalization	Q_Filter	Q_AbstractQuery	
Association	parameters Q_Parameter 0..*	Q_AbstractQuery	The number, name and type of a specific individual query's parameters must be specified in a constraint.
Association	Q_AbstractQuery	metadata Q_Metadata 0..1	Metadata is marked as optional, but should be provided for all queries.
Nesting	Q_InferredResult	Q_AbstractQuery	
Nesting	Q_SCT	Q_AbstractQuery	
Nesting	Factory	Q_AbstractQuery	
Generalization	Q_Query	Q_AbstractQuery	

Relationship Type	Source	Target	Description
Nesting	Q_Aggregate	Q_AbstractQuery	
Nesting	Q_UnitConversion	Q_AbstractQuery	
Nesting	Q_StringComparison	Q_AbstractQuery	
Nesting	Q_TemporalRelation	Q_AbstractQuery	
Nesting	Q_TimeComponentExtraction	Q_AbstractQuery	
Generalization	Q_Inference	Q_AbstractQuery	
Generalization	Q_SCTConstructor	Q_AbstractQuery	

Constraints

Constraint Type	Name	Details
OCL	default querySet is all EHRs	init:self.querySet= EHR->allInstances()

Abstract Class Q_Filter*Specialises:* Q_AbstractQuery*Realises:*

An abstract query returning a filtered set of unmodified EHRs.

Relationships

Relationship Type	Source	Target	Description
Generalization	Q_Filter	Q_AbstractQuery	
Generalization	Q_get_patient_by_id	Q_Filter	
Association	Q_Filter	parent Q_Filter 0..1	A parent query, the solution for which provides the querySet for this query. If no parent query is given the querySet is the set of all instances of EHR.

Constraints

Constraint Type	Name	Details
Invariant	solution is a set of EHRs	inv:self.solution.ocllsTypeOf(SetType(EHR))
Invariant	querySet is solution of parent query	init: self.querySet=(if self.parent->size() > 0 then self.parent->first().solution)

Abstract Class Q_Inference*Specialises:* Q_AbstractQuery*Realises:*

Result is a record component which if not already preexisting MUST be persisted to the Care Record by the query mechanism, with appropriate audit info and attestation set.

Relationships

Relationship Type	Source	Target	Description
Generalization	Q_Inference	Q_AbstractQuery	

Constraints

Constraint Type	Name	Details
Invariant	solution is a RECORD_COMPONENT	inv:self.solution.ocllsTypeOf(RECORD_COMPONENT)

Class Q_Metadata*Specialises:**Realises:*

Metadata describing the query and providing a link to the knowledge model.

The LRA metadata model is not currently defined but the following are starting categories for metadata definition taken from the LRA Artefacts Overview:

Data element type (fixed categories – e.g. data class, data value)

Data element name (?naming protocol)

Data element business definition

Definition source

SNOMED Clinical Term binding (if applicable)

XML Tag Name (if applicable)

Approval status

Implementation status

Reference to related Business Objective and/or Requirement Statement (if applicable)

Reference to related Business Model Component (if applicable)

Reference to models dependent on data element

Relationships

Relationship Type	Source	Target	Description
Association	Q_AbstractQuery	metadata Q_Metadata 0..1	Metadata is marked as optional, but should be provided for all queries.

Abstract Class Q_Parameter*Specialises:**Realises:*

A named parameter used within the query which is useful to have

Attributes

Attribute	Description
name : String [1..1]	Name is a string by which the query can internally refer to the parameter, and must only be unique within the query.

Attribute	Description
value : OclAny [1..1]	A literal value used within the query. Can be supplied at runtime or set in a specialized query. Value can be of any type: either UML or ISO13606.

Relationships

Relationship Type	Source	Target	Description
Association	parameters Q_Parameter 0..*	Q_AbstractQuery	The number, name and type of a specific individual query's parameters must be specified in a constraint.
Generalization	Q_Parameter_patient_id	Q_Parameter	
Generalization	Q_Parameter_sct_disease	Q_Parameter	

Constraints

Constraint Type	Name	Details
Invariant	No null parameters	inv: nameoclIsDefined() and value.oclIsDefined

Abstract Class Q_Query*Specialises:* Q_AbstractQuery*Realises:*

An abstract query returning a set of tuples. OCL version 2.0 provides the tuple datatype, which is useful here as it provides a bridge to the relational model. The datatypes contained in the tuple are not constrained. This means that collections can be included in the result tuples. A further specialization might restrict the contents of the result tuples to UML primitives, making a mapping from this result type to a table more direct.

Relationships

Relationship Type	Source	Target	Description
Generalization	Q_Query	Q_AbstractQuery	
Generalization			

Relationship Type	Source	Target	Description
	Q_history_of_disease_by_parameter	Q_Query	

Constraints

Constraint Type	Name	Details
Invariant	solution is a set of tuples	inv:self.solution.ocllsTypeOf(SetType(TupleType))

Package `Ira.technical.query.example`

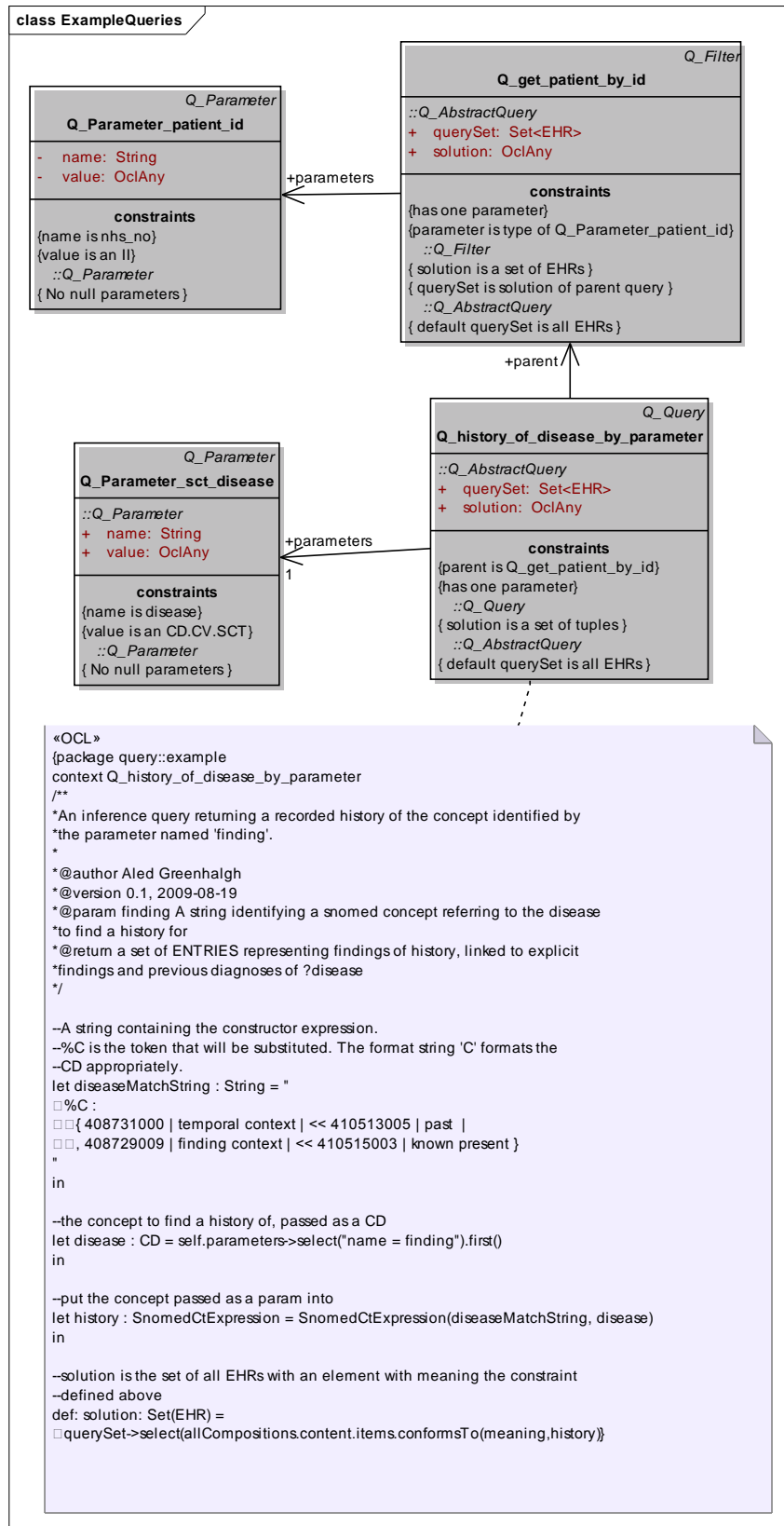


Figure 4: ExampleQueries

Class Q_Parameter_patient_id

Specialises: Q_Parameter

*Realises:***Attributes**

Attribute	Description
name : String [1..1]	
value : OclAny [1..1]	

Relationships

Relationship Type	Source	Target	Description
Generalization	Q_Parameter_patient_id	Q_Parameter	
Association	Q_get_patient_by_id	parameters Q_Parameter_patient_id	

Constraints

Constraint Type	Name	Details
Invariant	name is nhs_no	inv:name='nhs_no'
Invariant	value is an II	inv:value.ocIsTypeOf(II)

Class Q_Parameter_sct_disease*Specialises:* Q_Parameter*Realises:***Relationships**

Relationship Type	Source	Target	Description
Association	Q_history_of_disease_by_parameter	parameters Q_Parameter_sct_disease 1	
Generalization			

Relationship Type	Source	Target	Description
	Q_Parameter_sct_disease	Q_Parameter	

Constraints

Constraint Type	Name	Details
Invariant	name is disease	
Invariant	value is an CD.CV.SCT	

Class Q_get_patient_by_id*Specialises:* Q_Filter*Realises:***Relationships**

Relationship Type	Source	Target	Description
Association	Q_history_of_disease_by_parameter	parent Q_get_patient_by_id	
Generalization	Q_get_patient_by_id	Q_Filter	
Association	Q_get_patient_by_id	parameters Q_Parameter_patient_id	

Constraints

Constraint Type	Name	Details
Invariant	has one parameter	inv:parameters->size()=1
Invariant	parameter is type of Q_Parameter_patient_id	inv:parameters->first().oclIsTypeOf(Q_Parameter_patient_id)

Class Q_history_of_disease_by_parameter*Specialises:* Q_Query*Realises:***Relationships**

Relationship Type	Source	Target	Description
Association	Q_history_of_disease_by_parameter	parameters Q_Parameter_sct_disease 1	
NoteLink	<anonymous>	Q_history_of_disease_by_parameter	
Association	Q_history_of_disease_by_parameter	parent Q_get_patient_by_id	
Generalization	Q_history_of_disease_by_parameter	Q_Query	

Constraints

Constraint Type	Name	Details
Invariant	parent is Q_get_patient_by_id	
Invariant	has one parameter	

No index entries found.