HARMOSEARCH
the future of information services

FP7-SME-1
Project no. 262289

HARMOSEARCH

Harmonised Semantic Meta-Search in
Distributed Heterogeneous Databases

SEVENTH FRAMEWORK PROGRAMME

D3.1
Ontology for the query model

Due date of deliverable: 2011-05-31
Actual submission date: 2011-05-31

Start date of project: 2010-12-01 Duration: 24 month

Project funded by the European Commission within the Seventh Framework Programme

Dissemination Level		
PU	Public	**X**
PP	Restricted to other participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	

HARMOSEARCH
the future of information services

PROJECT ACRONYM: **HARMOSEARCH**

Project Title: Harmonised Semantic Meta-Search in Distributed Heterogeneous Databases

Grant Agreement: 262289

Starting date: December 2010 **Ending date:** November 2012

Deliverable Number: D3.1, Version 1.0

Title of the Deliverable: Ontology for the query model

Lead Beneficiary: TU-WIEN

Task/WP related to the Deliverable: WP 3, Task 3.1

Type (Internal or Restricted or Public): Public

Author(s): Adriano Venturini, Albert Rainer, Alessandro Forti, Christoph Herzog, Claudio Prandoni, Federico Galeazzi, Sabine Schneider

Partner(s) Contributing: eCTRL, CPR, TU-WIEN

Contractual Date of Delivery to the CEC: May 31, 2011

Actual Date of Delivery to the CEC: May 31, 2011

PROJECT CO-ORDINATOR

Company name:	[X+O]
Name of representative:	Manfred Hackl
Address:	Hamburgerstrasse 10/7, A-1050 Vienna, Austria
Phone number:	+43-676-842755-100
Fax number:	+43-676-842755-599
E-mail:	manfred.hackl@xpluso.com
Project WEB site address:	www.harmosearch.org

TABLE OF CONTENTS

1 INTRODUCTION 4

1.1 PURPOSE OF THE DOCUMENT 4
1.2 DEFINITIONS OF TERMS AND ABBREVIATIONS 4
1.3 RELATIONSHIP WITH OTHER DOCUMENTS 4
1.4 STRUCTURE OF THE DOCUMENT 5

2 USER REQUIREMENTS AND TEST CASES 6

3 TECHNICAL REQUIREMENTS 13

3.1 USER REQUIREMENTS ANALYSIS 13
3.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS 13
3.3 ADDITIONAL REQUIREMENTS 15

4 HARMOSEARCH QUERY REFERENCE MODEL 18

5 HARMOSEARCH QUERY MANAGEMENT 23

5.1 QUERY BY EXAMPLE APPROACH 23
5.2 TWO LAYERS APPROACH 24

6 LIST OF FIGURES 26

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

One of the biggest challenges in the HarmoSearch project is the mapping of search queries to enable search on different data bases with different query languages in use. Different data base systems, in fact, define the same query in a different way, as a typical part of their data access and manipulation logic.

In many cases queries can be very complex and might be combined with constraints based on geographical data (hotels not more than 500 metres from the Spanish Steps), price ranges (not more than EUR 100 per night), etc. HarmoSearch queries must be able to be sufficiently expressive to model the customer's requirements, either directly through a single complex query that enumerates all constraints, or through a sequence of simpler queries that narrow down result sets.

In order to translate from one query language to another, it needs some reference model to describe the concept of a query language. Purpose of this document is to model the concepts to allow to represent a Harmonise query in an abstract form. This model should be powerful enough to represent the desired query constructs, but simple enough to be able to support the variety of query capabilities which are exposed by the data providers. Some will expose powerful constructs, others very limited, so the right compromise should be identified to be able to cover most of the cases without losing in flexibility.

A specific model explicitly studied to represent queries that can then be translated in the query language of each data provider will be presented hereinafter, starting from the collection and analysis of user requirements and test cases.

1.2 DEFINITIONS OF TERMS AND ABBREVIATIONS

Harmonise: name of the existing technological solution. The current version is Harmonise 2.0, which includes the Harmonise Ontology, Harmonise Service Centre and the Harmonise Portal.

Metasearch: HarmoSearch component which provides distributed search capabilities to the integrated data sources.

Query Processor: HarmoSearch component which translates a query from one query language to another.

1.3 RELATIONSHIP WITH OTHER DOCUMENTS

Inputs to this document come from the deliverable D2.1 Use Case Specification, which defines the use cases that the system is supposed to support, and in particular from the metasearch scenario, which drives the identification of the user requirements for the query model.

This document poses the basis for the development of the query language (D4.1) and provides some guidelines and recommendations for the implementation of the Query Processor (D4.2) and of the Metasearch Application (D4.3).

1.4 STRUCTURE OF THE DOCUMENT

This document is structured in the following main sections:

- User requirements and test cases, which summarises all the user requirements gathered during the first months of the project pertaining to the query process.
- Technical requirements, which present the technical requirements to be taken into consideration for the development of the HarmoSearch query language, starting from the analysis of the user requirements and test cases collected and defined in the previous chapter.
- HarmoSearch query reference model, which describes the concepts to allow to represent an Harmonise query in an abstract form, starting from the technical requirements identified in the previous chapter.
- HarmoSearch query management, which analyses how to possibly handle a HarmoSearch query, providing therefore some inputs to the implementation of the query language and of the Metasearch component.

2 USER REQUIREMENTS AND TEST CASES

This section presents all the user requirements gathered during the first months of the project which pertain to the query process, including examples and possible test cases.

For each user requirement the following information is provided:

- ID: unique identifier of the requirement
- Author: partner who provided the requirement
- Group: category which the requirement belongs to
- Action: system functionality which the requirement refers to
- Requirement: brief description of the requirement
- Description: detailed description of the requirement
- Comment: additional notes which are useful for the implementation of the requirement
- Priority: importance of the requirement
    - High (i.e. mandatory)
    - Medium (i.e. desiderata)
    - Low (i.e. for the future)
- Dependency: relation to other requirements
- Examples: sample queries which refer to the requirement

ID	SEARCH001
Author	Afidium
Group	Search
Action	Quick Search
Requirement	It should be possible to search for specific items by specifying the name or a unique code
Description	Sometimes items are identified by a unique code or by a key; it should be possible for the user to search by specifying one of these fields in order to get a short list of results (or even just one result) without inserting many search parameters.
Comment	--
Priority	High

Dependency	--
Examples	Find the exhibition which has a particular code. Find a museum by its complete name

ID	SEARCH002
Author	Afidium, eCTRL, SPK, EC3, [x+o]
Group	Search
Action	Basic Search
Requirement	It should be possible to search for items by specifying a single criterion or a combination of criteria
Description	It should be possible to query different data providers by specifying one or more search criteria. The search results have to match either all the different conditions (AND) or at least one of them (OR) or a combination of them. The data types of the search fields include numbers, texts, dates, etc.
Comment	In the user interface, the search criteria should be organised and grouped by category in order to improve the usability
Priority	High
Dependency	--
Examples	Find all exhibitions in Rome or Napoli on a given date Find all accommodations of category 3, 4 and 5 stars in a given city

ID	SEARCH003
Author	SPK, eCTRL, [x+o]
Group	Search
Action	Basic search with enumeration values

Requirement	It should be possible to fill in some of the search criteria by choosing their values from enumerated value domains
Description	Some of the search parameters are not free text or numeric fields, but drop-down lists whose values must be selected by choosing among a set of predefined values. In order to translate a query from one query language to another, these reference lists have to be translated too
Comment	It would be useful to have a component to manage the reference lists, i.e. to dynamically retrieve, add, edit or remove values of an enumerated value domain.
Priority	High
Dependency	Depends on SEARCH002
Examples	Find all the exhibitions pertaining to "modern art" Find all the "3 stars" accommodations

ID	SEARCH004
Author	Afidium, EC3, SPK, [x+o]
Group	Search
Action	Basic search with geographical data
Requirement	It should be possible to search by specifying geographical data and/or the indication of a specific area of interest
Description	In some cases the user has the need to find items which are located in a particular area or close to a specific point of interest
Comment	The search engine should be able to handle geographical data and to compute distances between them. It requires a geographic hierarchy holding several levels of geographic entities and their relations. It requires geocodes of each geographic entity, with distance calculation functionality.
Priority	Medium
Dependency	Depends on SEARCH002

Examples	Find all the accommodations close to the centre of Berlin Find all the exhibitions within 1 km from a certain place

ID	SEARCH005
Author	Afidium
Group	Search
Action	Basic search with flexible dates
Requirement	It should be possible to get back not only the results which match exactly the specified dates, but also the ones which are available one or two days before and after
Description	The user searches for a specific item on a specific date and gets back among the results also the items matching the search criteria which are available one or two days before or after the specified date
Comment	
Priority	Medium
Dependency	Depends on SEARCH002
Examples	Find all the exhibitions which will take place on May 1st or close to this date

ID	SEARCH006
Author	SPK, [x+o]
Group	Search
Action	Basic search with priority criteria
Requirement	It should be possible to distinguish between criteria which are mandatory and criteria which are optional
Description	From the user's point of view some of the search criteria could be mandatory, while others could be just preferred, i.e. it is not necessary that the search results match these latter criteria but it is a desiderata

Comment	The search engine should handle criteria with different priorities and show to the user the search results ranked according to these priorities
Priority	Low
Dependency	Depends on SEARCH002
Examples	Find all the accommodations in Berlin that possibly have the swimming pool Find all the exhibitions pertaining to "modern art" which take place possibly in Berlin

ID	SEARCH007
Author	SPK, [x+o]
Group	Search
Action	Advanced search
Requirement	It should be possible to combine basic searches applied to different objects
Description	The advanced search is used to handle nested queries, i.e. queries which are composed of multiple requests whose search results must be combined or where the results of the first query will become a search criteria to run the following one
Comment	In the user interface, the search criteria should be organised and grouped by category in order to improve the usability
Priority	High
Dependency	Depends on SEARCH002 and SEARCH003 (possibly also SEARCH004 and SEARCH005)
Examples	Find all the accommodations which are located in the same city as the exhibition X Find all the accommodations which are close to exhibitions of "modern art"

ID	SEARCH008

Author	Afidium, EC3
Group	Search
Action	Search with information on the query context
Requirement	It should be possible to add to a query some information on the query context in order to find results which better fit to user's needs or preferences
Description	This search definition adds to the search criteria specified by the user some additional information related to the environment of the query, such as an identifier of the initiator, intended receivers, preferred language and the intentions of the requester, i.e. what the requester wants to achieve by sending the request and how a response should look like.
Comment	Additional information on the query context can be added to the query as "hidden" fields, such as the user actual position, the device type from which the user is searching, the user profile; the user's device can influence the output format of the result data (HTML, XML, Mobile, etc.). It requires a context analyzer aware of actual user properties (like position, locale, browsing device) as well as user pre-configurations (i.e., receivers, result filter, etc..).
Priority	Medium
Dependency	Depends on SEARCH002 and SEARCH003 (possibly also SEARCH004 and SEARCH005)
Examples	Find all the exhibitions near the users actual position which best fit to the user's interests Find all the "modern art" exhibitions located in Berlin and send the search results to a specified email address

ID	SEARCH009
Author	Afidium, eCTRL, [x+o]
Group	Results management
Action	Sort results

Requirement	It should be possible to sort the results of a query by one or more criteria
Description	The user can sort the results of a query according to different sort criteria, such as by price, by location, by alphabetical order, by mark (in case of ranking provided), by matching value (close or far from search criteria), etc.
Comment	Search results should be paginated in order to be easily browsed by the user
Priority	High
Dependency	Depends on SEARCH002
Examples	Find all the exhibitions sorted for price ascending Find all the accommodations sorted in alphabetical order

ID	SEARCH010
Author	Afidium, SPK, [x+o]
Group	Results management
Action	Simple result filters
Requirement	It should be possible to specify which particular field or fields should be returned in responses
Description	The user is not always interested in every single detail of the results; it would be useful to specify in the query which are the fields that he wants to receive
Comment	The selection of which information to display can be done also by the user interface
Priority	Medium
Dependency	Depends on SEARCH002
Examples	List of the city names where a "modern art" exhibition is going to take place on May 1st

3 TECHNICAL REQUIREMENTS

3.1 USER REQUIREMENTS ANALYSIS

This section summarises the results of the analysis of the user requirements and test cases collected and defined in the previous chapter. The outcomes of this analysis are a number of requirements to be taken into consideration for the development of the HarmoSearch query language. Such requirements have been divided into two main categories:

- Functional and Non-functional Requirements that pertain to the query language
- Additional Requirements that address issues related to the query process in general rather than to the query language

3.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

This paragraph describes the system requirements for the implementation of the HarmoSearch query language. They are divided in Functional Requirements, which describes what the query language must be able to do, and Non-functional Requirements, which specify additional characteristics such as usability, availability, reliability, supportability, testability, maintainability, etc.

For each system requirement the following information is provided:

- ID: unique identifier of the requirement
- Requirement: brief description of the requirement
- Description: detailed description of the requirement
- User Req: user requirement(s) which the system requirements refers to
- Priority: importance of the requirement
    - High (i.e. mandatory)
    - Medium (i.e. desiderata)
    - Low (i.e. for the future)

Functional Requirements				
ID	**Requirement**	**Description**	**User Req**	**Priority**
FR1	Simple data types	• Number • String • Date • Enumeration	SEARCH001 SEARCH002	High
FR2	Complex data types	• Geographical data	SEARCH004	Medium

FR3	Basic operations	• =, <, <=, >, >=, <> • STATIC IN (predefined set)	SEARCH001 SEARCH002	High
FR4	Complex operations	• JOIN • DYNAMIC IN (nested queries)	SEARCH007	High
FR5	Logic operations	• AND • OR • NOT	SEARCH002	High
FR6	Data specific operations	• NEAR -> geographical data • LIKE %% -> string • BETWEEN -> date • "Flexibility" -> date	SEARCH002 SEARCH004 SEARCH005	High/ Medium
FR7	Condition operations	• Mandatory vs. optional fields/conditions	SEARCH006	Low
FR8	Selection operations	• <fields list> • DISTINCT • AS	SEARCH010	Medium
FR9	Sorting the results	• ORDER BY (multiple sort criteria)	SEARCH009	High
FR10	Operations on the results	• AVG • MIN • MAX • COUNT	--	Low

Non-functional Requirements				
ID	**Requirement**	**Description**	**User Req**	**Priority**
NFR1	Domain independence	The query language should be independent from the domain for which it is designed, it should be general and easily adaptable to other domains	..	High

NFR2	Extensibility	The query language should be easily extendible in case additional requirements will be added which are not covered by the first implementation	--	High
NFR3	Robustness and security	The query language should handle unexpected situations without any consequence for the data which are queried	--	High
NFR4	Human readable	The query language should be easily readable by humans	--	Medium
NFR5	Easily mappable and convertible	The translation and mapping process between this query language and another one should be as simple as possible	--	High
NFR6	Open standard/not vendor specific	The query language should be based on open standards and it should not be dependent on vendor specific definitions or modules	--	Medium

3.3 ADDITIONAL REQUIREMENTS

Among the user requirements collected in Chapter 2, a couple of them refer to issues related to the query process in general rather than to the query language:

- How to encode in the query information on the context, such as an identifier of the initiator, intended receivers, preferred language and the intentions of the requester, i.e. what the requester wants to achieve by sending the request and how a response should look like (User Requirement SEARCH008)
- How to query metadata and schema, including reference lists management, i.e. the possibility to dynamically retrieve, add, edit or remove values of an enumerated value domain (User Requirement SEARCH003)

The following paragraphs will analyse more in details these two additional requirements.

3.3.1 Context of the query

The query context denotes the environment of a query, such as an identifier of the initiator, intended receivers, preferred language, and the intentions of the requester, i.e., what the requester wants to achieve by sending the request and how a response should look like. For instance, a request could originate from an interactive communication where a human actor sits in front of a computer and waits for the response, i.e., the response should be quick, should be sorted, and appropriate to the screen device, e.g., HTML and paging of result list. On the other hand, a request

from a data collecting and analyzing application may have a lower priority, no paging, no ranking, and the result should be a single XML document.

A major function of the query context is to aid in finding suitable service providers for a given request. A properly declared context can improve both, precision and recall of retrieved service providers from the registry. Precision means the ratio of relevant providers to retrieved providers with respect to a request while recall means the ratio of successfully retrieved relevant providers to all providers that are relevant to a given query.

The following table summarises which are the requirements to be considered to include in the query language the information on the query context. In order to keep all the requirements homogeneous, the same notation is used as for the system requirements described in the previous paragraph.

Additional Requirements – Context of the query				
ID	**Requirement**	**Description**	**User Req**	**Priority**
AR1	Identifier of the sender	Propagate to the receivers an identifier of the actor that initiates a query request	SEARCH008	Medium
AR2	Intended receivers	Specify the data providers which to constrain the search to	SEARCH008	Medium
AR3	Type of request	Specify the type of request (e.g. sending an ad-hoc query to a number of data providers or batch transfer of static data)	SEARCH008	Medium
AR4	Reference to the domains or collections	Reference to the particular domains of the actual request	SEARCH008	Medium
AR5	Geographical region of interest	Specify the geographical region which to constrain the search to	SEARCH008	Medium
AR6	Preferred language	Specify a preferred language for the content of the results	SEARCH008	Medium
AR7	Priority	Specify the priority of the request (e.g. for accounting)	SEARCH008	Low
AR8	Preferred output	Specify how the response will look like (e.g. preferred output format, return address)	SEARCH008	Medium

AR9	Paging preferences	Maximum number of items that should be displayed on a single page	SEARCH008	Medium

3.3.2 Query metadata and schema

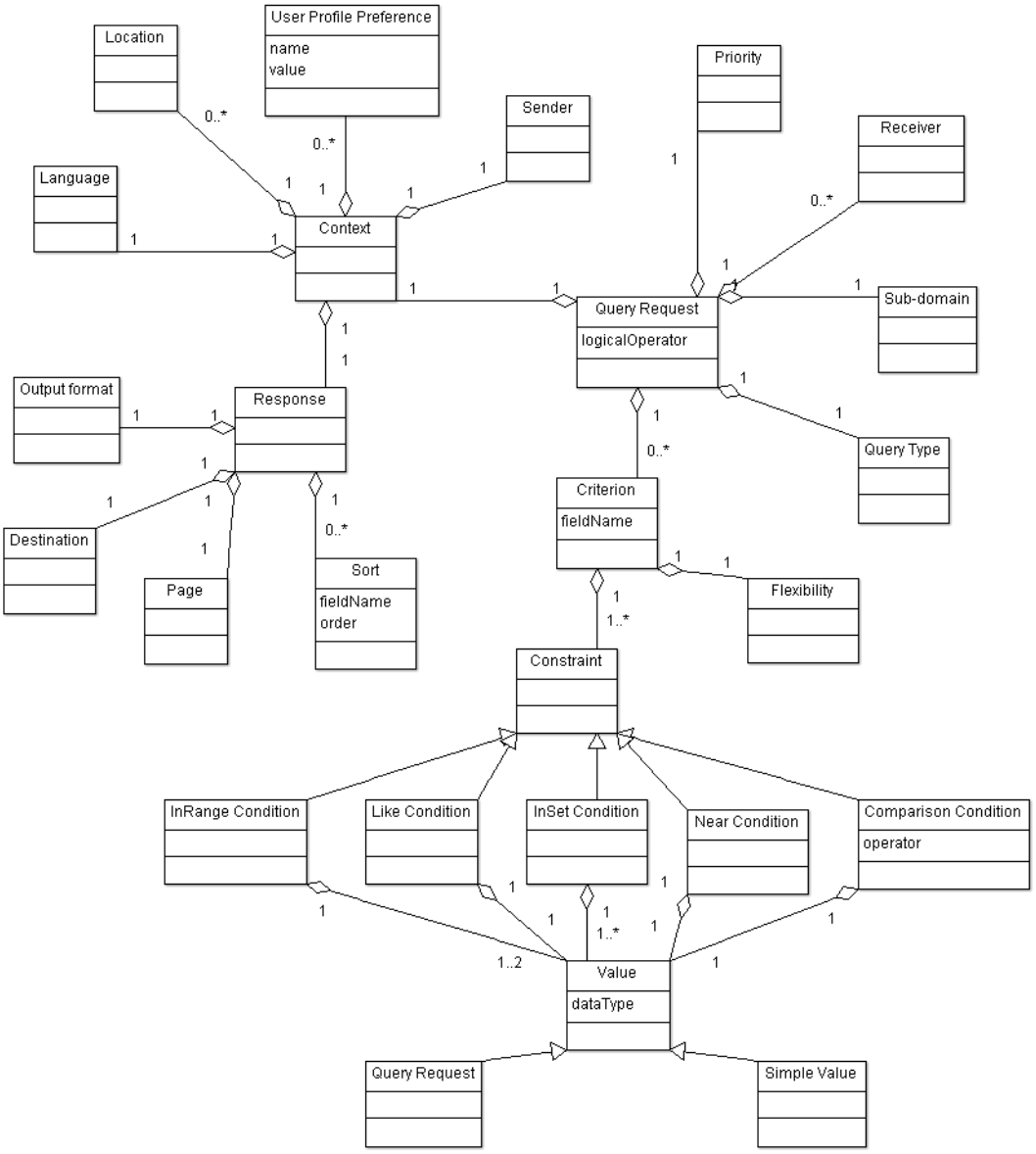
In order to develop effective services by exploiting the HarmoSearch query service, it should be possible to dynamically retrieve information about the type of data which are managed by HarmoSearch (metadata information).

The query language should be able to access metadata. This is similar to what some relational databases provide, i.e. the possibility to exploit the same query language (e.g. SQL) to access schema and metadata information.

Additional Requirements – Query metadata and schema				
ID	**Requirement**	**Description**	**User Req**	**Priority**
AR10	Get all available data providers	Which are the available content providers?	SEARCH003	Medium
AR11	Get all available collections	Which are the available collections? I.e. which type of content is available (e.g. accommodations, events, etc.)?	SEARCH003	Medium
AR12	Get collections of a given data provider	Which are the available collections of a given content provider?	SEARCH003	Medium
AR13	Get fields of a given collection	For each collection, which are the available elements (fields)?	SEARCH003	Medium
AR14	Get type of a given field	For each field, which is the type?	SEARCH003	Medium
AR15	Get pre-defined values of a given enumerated type	For each enumerated type, which are the possible values?	SEARCH003	Medium
AR16	Get primary and foreign keys	For each field, is it a primary key? Or a foreign key? Related to what?	SEARCH003	Medium

4 HARMOSEARCH QUERY REFERENCE MODEL

This section presents the HarmoSearch query reference model, which describes the concepts to allow to represent an Harmonise query in an abstract form. Starting point are the technical requirements identified in the previous chapter. This model will be the basis for the development of the HarmoSearch query language (D4.1).



The main concepts of HarmoSearch query model are:

- ***Query Request***. A *Query Request* is a container that holds the context as well as the workload, i.e. the actual query. Different search criteria can be combined according to the value of the *logical Operator* attribute. The search results have to match either all the different conditions (AND) or at least one of them (OR).

- ***Query Type***. HarmoSearch supports different use cases, such as sending an ad-hoc query to a number of data providers or batch transfer of static data (see D2.1 Use Case Specification). The *Query Type* concept is used to denote the type of request. For instance, in case of meta-search the element has the value "MS-1" which refers to the respective use case.
- ***Sub-domain***. HarmoSearch allows in principle query requests from any domain. Therefore it is necessary to have a reference to the particular domain of the actual request. For instance, a query submitted to find actual art exhibitions in Austria does not fit to find an accommodation in the same country. Domain standard types are to be developed by the HarmoSearch community. To go on with the example, the domain type *Art_Exhibition_Light* or *Art_Exhibition_Medium* would refer to two particular versions of a community standard that denotes the fields for requests and the expected outcome. Data providers could then declare their conformity with such a community standard, requesters can refer to the standard, and HarmoSearch can use this reference in order to match the query request with appropriate providers. The usage of the *Sub-domain* element is sketched on Figure 1. From this reference appropriate data providers can be inferred.

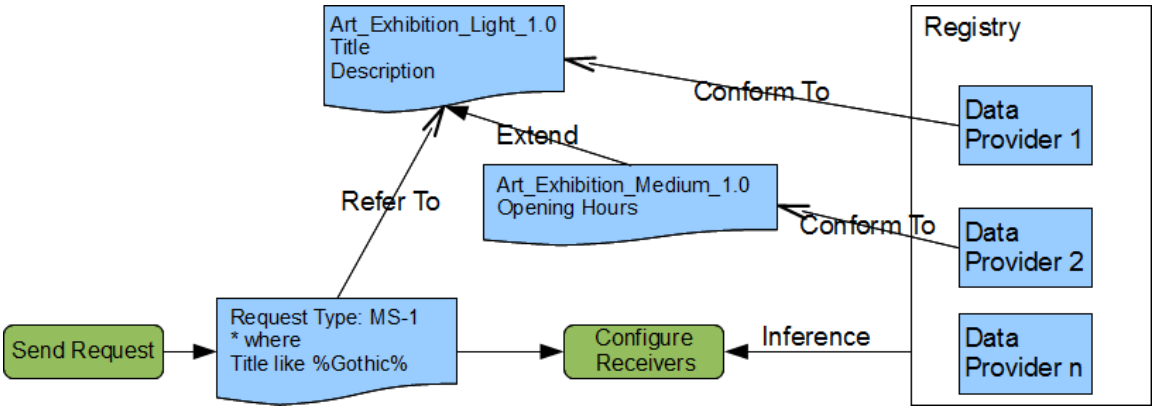


Figure 1: Query requests have a reference to a sub-domain; Query Type "MS-1" refers to Use Case Meta-Search-1 from D2.1 Use Case Specification.

- ***Receiver***. From the originators point of view, a receiver is an abstract actor that is the addressee of the originators request. The actual receivers are chosen by the HarmoSearch platform according to properties from the request that are matched with properties of services retrieved from the registry component. A requester may want to specify the data provider directly in order to constrain the search to dedicated providers. Receivers may be pre-configured in order to reuse service configurations and have a faster receiver configuration at hand. For instance, all data providers that have a focus on a particular country or a particular topic of interest could be preconfigured and then be referenced by a sender. Receivers can have modifiers. A modifier shows whether the receivers are optional or mandatory, or it denotes a minimum respectively maximum set of preferred receivers. For instance, a set of named providers with a Mandatory and

Max=unbounded modifier would imply that these named providers are required to be contacted and if additional relevant providers can be found these should be contacted, too. In contrast, an Optional modifier on a set of named providers would imply that only providers from the set should be considered and then only those that can cope with the request. A typical Use case would be that a requester has a relation with those providers such as an agreement that permits the requester to use the service.

- ***Priority***. While HarmoSearch in its current state does not consider accounting of requests it could be of interest for the data provider and the requester. Request with low priority could be cheaper than high priority jobs.
- ***Context.*** The *Context* denotes the environment of a query, such as an identifier of the initiator, geographical area of interest, preferred language, and the intentions of the requester, i.e., what the requester wants to achieve by sending the request and how a response should look like.
- ***Sender***. A sender is the actor that initiates a query request. A sender must have a unique ID within the HarmoSearch environment. This ID is NOT the user name of the actor, but an identifier that is used to propagate a reference from the actor to the receivers (including HarmoSearch) for accounting and reporting.
- ***Location***. A requester may specify that only data from a certain geographical region are of interest (e.g. a country, a province or a city). From these preferences it can then be inferred which data providers are relevant to handle the request.
- ***Language***. Data providers may support different languages for the content of the results and a requester may denote a preferred language. The content provider would then return the results either in the preferred language - if it supports this feature and the specified language - or in the default language if it does not.
- ***User Profile Preference***. A requester may specify additional preferences according to his interests and to what he wants to achieve by sending the request. Such preferences may be derived also from the user's profile. From these preferences it can then be inferred which data providers are relevant to handle the request.
- ***Response***. A *Response* in the context of a meta-search request is a list of result items. A requester may want to specify how a response should look like, i.e. in terms of preferred output (HTML, XML, Mobile phone format), return address for asynchronous results, etc.
- ***Output Format***. A standard procedure how HarmoSearch processes the list of result items is to return it to the requester as a single XML document as soon as all data providers have returned their respective result or a time out has occurred. Such a default procedure would not be suitable to every use case. The *Output Format* element allows to specify which is the preferred output format of the response. A specific template may be provided by the requester to be filled with the results. Or it may be possible to specify which

particular field or fields should be returned in responses. Such different handling of the response can be specified in the request.

- ***Destination.*** *Destination* denotes the return address of the response. Per default this is just the address of the originator of the request, that is, "back to sender" in a synchronous communication fashion. Such a default procedure would not be suitable to every use case. For instance, in an interactive mode with a human user it is more appropriate to return a possibly partial list as soon as possible. On the other hand, for later data processing it should be possible to send the result not back to the requester but to a dedicated addressee such as a data analysis application. If this element is present, the result is sent to the value declared in the *Destination* element. This must be some service that is capable to receive the result. Typically this will be a Web Service that can process the result list or a data storage application in order to keep the result for later processing.
- ***Page***. Paging means the split up of the result list into smaller chunks that fit on a screen. If the element is present, it denotes the maximum number of items that should be displayed on a single page.
- ***Sort***. The requester may want to specify how the list of result items will look like and get the results of a query sorted according to different criteria, such as by price, by location, by alphabetical order, by mark (in case of ranking provided), by matching value (close or far from search criteria), etc.
- ***Criterion***. This element models the different search criteria, i.e. how to constrain the search for each of the specified fields.
- ***Constraint***.
    - *Comparison Condition*. It represents a comparison condition and has an attribute *operator* (=, <, <=, >, >=, <>). The content of the *value* element is the selected value by the user.
    - *Like Condition*. It represents a partial match condition for symbolic features. The content of the *value* element is the substring for partial matching.
    - *InSet Condition*. It represent a membership condition which allows to determine whether the value of an expression is equal to any of several *values* in a specified list.
    - *InRange Condition*. It represents a range condition. The content of the *from Value* element is the lower range value and, analogously, the content of the *toValue* element is the upper range value.
    - *Near Condition.* It represents a geographical condition which allows to search for items which are located in a particular area or close to a specific point of interest.
- ***Value***. This is the value entered by the requester which to constrain the search to. It can be a *Simple Value* (e.g. a number, a string, a date, ...) or a *Query Request* itself. The latter is to allow users to perform nested queries, i.e. queries which are composed of multiple requests whose results must be

combined (e.g. give me all 4* hotels with available single rooms within 1 km from an exhibition in Berlin about modern art).

- ***Flexibility***. From the user's point of view some of the search criteria could be mandatory, while others could be just preferred, i.e. it is not necessary that the search results match these latter criteria but it is a desiderata. With these elements the requester may distinguish between criteria which are mandatory and criteria which are optional. For instance, if a date range is not mandatory, he can get back not only the results which match exactly the specified dates, but also the ones which are available one or two days before or after that.

5 HARMOSEARCH QUERY MANAGEMENT

When designing a query language it needs to be aware of how a query will be handled and processed to provide to the users the requested results.

Even if the detailed design and specifications of the whole query process will be provided in WP4 – D4.1 Query language, D4.2 Query Processor and D4.3 Metasearch Application – some basic ideas are anticipated hereinafter which serve as guidelines and recommendations for the implementation of the HarmoSearch query language and Metasearch component.

Starting point is how to manage the process of mapping a query from one query language to another. This is a complex issue which involves:

- Mapping of the search fields
- Mapping of the reference lists
- Mapping of the search options/operators/conditions

To handle query transformation and processing, two approaches have been taken into consideration – the Query by Example (QBE) approach and a two layers approach. The following paragraphs will analyse more in details these two possibilities.

5.1 QUERY BY EXAMPLE APPROACH

"Query by example" (QBE) is a database query language for relational databases developed by IBM in the 1970s in parallel to what was to become SQL.

In the context of information retrieval, QBE allows a user to submit a document, or several documents, and ask for "similar" documents to be retrieved from a document database. In other words, the user supplies example result sets that can formulate constraints or other selection criteria in addition to typical string values. Examples can often be built through graphical user interfaces.

To conduct a search for field data matching particular conditions, the user enters criteria into the search form, creating search conditions for as many fields as desired. A query is automatically generated to search the database for matching data. When looking for a hotel, for example, the client would specify basic hotel characteristics (e.g. name, category, location, etc.) and the system would on that basis return a suitable set of hotels matching those characteristics. Only items with search values filled in are used to "filter" the results.

In order to successfully handle query transformation and processing in HarmoSearch, one possibility is to encode the query as a simple QBE, based on the HarmoSearch ontology, and to translate it by reusing or extending the data mapping provided by the organization to be queried.

QBE is sufficient to handle a large number of applications; nevertheless it is by itself not enough to handle the complexity of the user requirements which has been described in the previous chapters, in particular in cases of complex queries such as nested queries which join different search criteria or queries which enumerate a number of non linear constraints.

5.2 TWO LAYERS APPROACH

A complete approach to query languages, i.e. relational algebra, can basically be split in two groups of operations: primitive operations based on some sort of selection (SELECT) and operations based on a joining of retrieved values (JOIN). For querying different data providers, it makes sense to handle the two parts (SELECT and JOIN) separately.

By definition data providers have a limited view of the whole data space in this system. Therefore they cannot successfully apply JOIN operations with restrictions on their data. For example think of a query asking for hotels which are near to some (any) large music festival. Even if an accommodation provider also has information about (some) events, he cannot correctly fulfil the query since he would have to have knowledge of all events in the data space.

Therefore it appears useful to only transport the SELECT portion of the query to the actual data providers for answering. This allows us to move the logic for further (JOIN) operations onto a separate layer "above" the basic SELECT functionality.

This logic layer has the task to handle complex queries (joins or nested queries) by transforming them into a combination of simple (SELECT) queries issued to the metasearch engine. The logic layer then collects the results from the simple queries and processes them according to the request of the user who performed the query.

There can be several concurrent implementations of this JOIN layer according to what functionality is actually required by the use cases to be covered. While a full relational complete query language can be implemented, for the current use cases a simpler approach appears sufficient. Additional capabilities can then be added as a new or extended logic layer as the need arises.

Therefore, the implementation should assure that it is possible to concurrently have different logic layer implementations and access them through search interfaces. Also, skipping the logic layer and accessing the SELECT layer directly should be possible.

5.2.1 SELECT Layer

For the Harmonise requirements, it appears feasible and desirable to implement the basic SELECT layer in a Query-By-Example (QBE) approach, as described in the previous paragraph. In this way, a relatively simple and easy-to-use query language can be made available which nevertheless covers a large number of applications.

These simple QBE queries are also the input for the metasearch component, which distributes the query to the appropriate recipients and collects the results. Only if more complex operations are required on the Harmonise level (JOIN), then the collected results are passed to a JOIN layer for further handling.

5.2.2 JOIN Layer

One possible way to implement the JOIN layer is to provide a mechanism which allows to create new ad-hoc collections of data ("VIEWS") by defining how they map to the already existing collections. Complex queries can be then treated as simple (SELECT) queries on these VIEWS. This approach implies:

- To define a new collection by specifying that it is actually created by joining other collections which are already available in the Harmonise space.
- To specify the JOIN condition, i.e. which are the fields of the existing collections that will be used to join them.
- To specify the FILTER condition, i.e. which are the constraints which apply to the joined collections.
- To specify which fields of the new collection should be mapped to which fields of the joined collections.

6 LIST OF FIGURES

Figure 1: Query requests have a reference to a sub-domain; Query Type "MS-1" refers to Use Case Meta-Search-1 from D2.1 Use Case Specification. 19