FACULTY OF ENGINEERING
AND ARCHITECTURE

Discovering Digital Art Collections using Link-Traversal-based Query Processing

Martijn Bogaert
Student number: 01706456

Supervisors: Prof. dr. Pieter Colpaert, Prof. dr. ir. Ruben Verborgh
Counsellors: Bryan-Elliott Tam, Ir. Wout Slabbinck

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2022-2023

GHENT
UNIVERSITY

Acknowledgements

TODO

Notes related to the master's thesis

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Disclosure of confidentiality (only if applicable)

To this end, consult the information on the faculty website - **Note related to the format of the master's thesis (all studies)**

TODO

Abstract

TODO

Conference Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

This document is a model and instructions for LaTeX. Please observe the conference page limits.

II. Ease of Use

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections III-A–III-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

Identify applicable funding agency here. If none, delete this.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not

"Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a] Sample of a Table footnote.

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.

Contents

List of Figures

List of Tables

List of Code Fragments

1

Introduction

TODO

2

Related work

2.1 Linked Data

This section presents a comprehensive exploration of Linked Data, encompassing its fundamental principles, data modeling, syntax, query interfaces, and the associated challenges and advantages. In Section 2.1.1, the concept of Linked Data and its principles are introduced, highlighting the significance of unique URIs, dereferencing, and data interlinking. Section 2.1.2 focuses on the Resource Description Framework (RDF) as the cornerstone for representing relationships and knowledge connections within Linked Data. Section 2.1.3 provides an overview of RDF syntax, including popular formats such as XML, Turtle, N-Triples, and JSON-LD, which facilitate the flexible expression and exchange of RDF data. Lastly, Section 2.1.4 briefly introduces SPARQL, the query language for RDF data This comprehensive examination serves as a solid foundation for the subsequent discussions on Linked Traversal-based Query Processing.

2.1.1 Introduction and Principles

To better understand the origins of the idea behind Linked Data, it is important to examine the origins of the World Wide Web. For example, its first, but still rather primitive, underlying technology was introduced in 1989 at CERN. Tim Berners-Lee was the man responsible for its development. By using HyperText Markup Language (HTML), it enabled scientists, and later the rest of the world, to publish documents that could contain links to other documents. This helped create a mesh of documents and information. However, since these documents in fact contained nothing more than raw data dumps and links between documents represented simply an indication of how to reach the document, these documents and their relationships lacked semantics. Figure 2.1 illustrates what a web of documents without unambiguous indications of what their contents and the links between them represent, might look like. It is necessary to note here that the used icons are not the contents of their respective documents, but only a representation of their contents. Nevertheless, in themselves, they prove the weakness of such web as much as when the effective content of the documents had been represented. After all, just from the raw content of documents and their mutual links, a person cannot clearly infer exactly what their constellation represents, let alone a computer. From that deficiency, therefore, emerged the idea of Linked Data. (Jacksi and Abass, 2019) (Bizer et al., 2011)

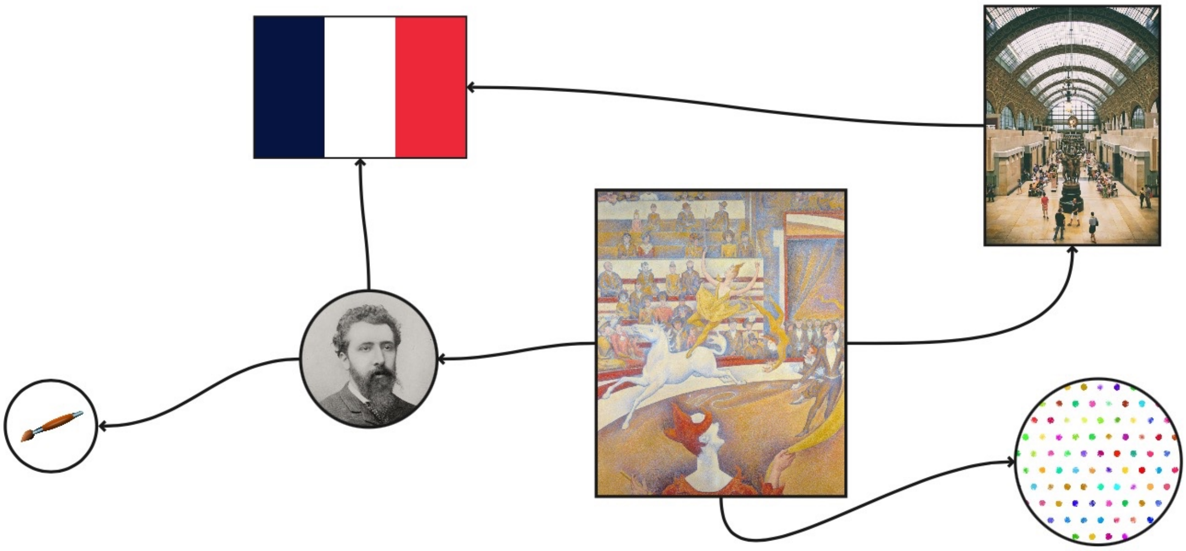

Figure 2.1: Representation of a web of documents without unambiguous indications of what the documents and the links between them represent

Simply put, data coming from different sources can be labeled as Linked Data as soon as they are linked by typed links. In other words, links are no longer just an indication of how to reach another document. Indeed, within the Linked Data story, they also contain information about what exactly the link in question represents. Linked Data thereby ensures the meaning of data is explicitly defined, in turn rendering the data machine-readable. Figure 2.2 represents the same web of documents as Figure 2.1, but this time in accordance with the idea of Linked Data. Indeed, the documents have been given an unambiguous indication of what they represent, and their mutual semantics have also been clarified thanks to the labeling of their links. (Bizer et al., 2011)

Although several technologies exist to achieve the goals of Linked Data, the use of URIs is essential. After all, since URIs are unique, they can unambiguously reference a particular entity. Practically speaking, the URIs that appear in a Linked Data document can be dereferenced using the HTTP protocol in order to retrieve the underlying entities. For instance, `https://stad.gent/id/concept/530010539`, is a URI that can be dereferenced using the HTTP(S) protocol.

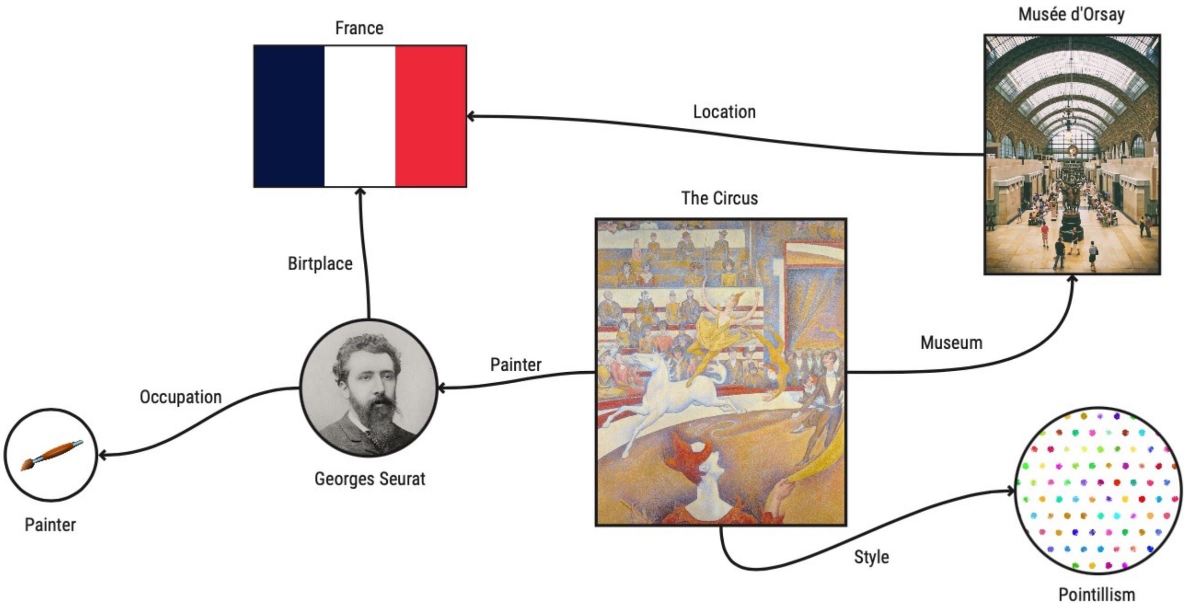

Figure 2.2: Representation of a web of documents composed according to the spirit of Linked Data

By dereferencing URI after URI in this way, little by little a - what could be called - *field of information* unfolds, whose semantics can be unambiguously determined by both man and machine. (Bizer et al., 2011)

To clarify the concept of Linked Data, Berners-Lee (2006) put forth four principles to be taken into consideration.

1. **Use URIs as names for things**

   The principle of using URIs has already been discussed above.

2. **Use HTTP URIs so that people can look up those names**

   The principle of using the HTTP protocol to dereference URIs was also touched on above. Nevertheless, it is important to reiterate its importance, as there are other protocols besides HTTP for dereferencing URIs. However, these will technically differ from the HTTP protocol, each in its own different ways. For example, not using the ubiquitous Domain Name System (DNS), is, among others, a common practice among alternative protocols. However, in light of clarity and uniformity, as well as for other technical reasons, the HTTP protocol should be adhered to. (Berners-Lee, 2006)

3. **When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)**

   Obviously, it would not fit within the spirit of Linked Data to obtain a raw data dump when dereferencing a URI that was included from another document as a *Linked Data link*. The obtained data itself must comply with Linked Data principles. Therefore, there are some standards that clearly indicate how ontologies can be described. Consequently, to enable the construction of applications that deal with Linked Data, it goes without saying that a Linked Data document should be built according to the principles of an existing standard. RDF is the most common such standard and is therefore discussed further in Sections 2.1.2. In addition, Section 2.1.4 introduces the SPARQL query interface. After all, large datasets are expected to also provide such interface. (Berners-Lee, 2006)

4. **Include links to other URIs so that they can discover more things**
   The fourth and final principle, too, is rather obvious. After all, by definition, one can only speak of Linked Data when a document refers to at least one other document. In addition, to help advance the cause of transforming the World Wide Web in its current form into a semantic World Wide Web, aided by the concepts of Linked Data, it is preferable to also include links to documents belonging to other sites. (Berners-Lee, 2006)

In conclusion, Linked Data plays a crucial role in giving meaning to the Web by enabling the interconnection and integration of diverse data sources. By adhering to the principles of unique URIs, dereferencing, linking, and using standardized formats, Linked Data fosters a more structured and interconnected web of knowledge. Examples such as DBpedia[1], which provides a structured representation of Wikipedia data, and Friend of a Friend (FOAF), which allows for the description of people and their relationships, illustrate how publishing data as Linked Data benefits from enhanced data discoverability, interlinking with other datasets, and enabling novel applications and insights. Local initiatives like Collections of Ghent (CoGhent[2]), which digitizes art collections from cultural houses in Ghent and will be further discussed in Section 2.4, similarly demonstrate the potential of Linked Data for local organizations in contributing to the broader web of knowledge. (Auer et al., 2007) (Golbeck and Rothstein, 2008) (Van de Vyvere et al., 2022)

2.1.2 Resource Description Framework

The idea behind Linked Data is interesting in itself, but does not yet describe exactly how to get started with it. Therefore, this section introduces the Recourse Description Framework (RDF). Developed under the auspices of the World Wide Web Consortium (W3C), RDF is an infrastructure that allows for the construction of Linked Data datasets and their metadata. Consequently, this not only allows data publishers to lay out their data as Linked Data, but also gives data consumers clear guidance on how the data can be understood. Note here that data consumers can be both individuals and computer applications. (Miller, 1998)

An interesting way to understand RDF is to first make a jump to the English language. Take the sentence below:

[1] https://www.dbpedia.org
[2] https://www.collections.gent

The birthplace of Georges Seurat is France.

According to English grammar, the *who* or *what* around which a sentence revolves, is called the subject of the sentence. Therefore, when looking at the sentence above, *Georges Seurat* is its subject. In addition, the part of a sentence that gives more information about the subject, is referred to as the predicate, making *the birthplace* the predicate in the above sentence. Finally, the matching value complementing the predicate and completing the sentence, is also of importance. Logically, in the case of the sentence above, that would be *France*. Together, these three components form the most basic building blocks of a sentence. In fact, no matter their lengths, combined, they will always establish a piece of knowledge, exactly what RDF also seeks to accomplish. (Powers, 2003)

The building blocks of RDF data are basically exactly the same as those of linguistic sentences. After all, they are also three in number and even partly share the same names. Moreover, much like with sentences, combined, they form a single yet very clear piece of knowledge. Unlike the English language, however, they are not referred to as sentences. Rather, they are called triples. (Powers, 2003)

- **Resource**

  Miller (1998) defines a resource as any object that is uniquely identifiable by a URI. This enables it to come in different forms: as a web page, as an entire website or simply as any resource on the Web that conveys information in one way or another. (Candan et al., 2001)

  To make the comparison with the English language again, in a triple, the resource corresponds to the subject in a sentence. Moreover, in practice, the term *subject* is often preferred over *resource*. (Powers, 2003)

- **Property Type**

  A property type, or simply a property, introduces a specific aspect, characteristic, attribute, or relationship of a resource. A property type always expects a value to ultimately define the piece of knowledge represented by a triple. (Candan et al., 2001) (Miller, 1998)

  As for property types, in practice, the corresponding term from the English language, *predicate*, is also frequently used as opposed to the more theoretical *property type*. (Powers, 2003)

- **Value**

  A value resolves the concept or relationship initiated by a property type. In this way, it captures the knowledge conveyed by the triple. Values can be represented as text strings, numbers, or any atomic data. However, they

> can also be resources themselves. This characteristic allows triples therefore to be the building blocks of a web of knowledge. (Miller, 1998)
>
> It is evident that a value in a triple corresponds to a value in an English sentence. However, in practice, the term *object* is often preferred. (Powers, 2003)

While triples convey a clear and distinct piece of knowledge, a collection of triples can naturally convey a more comprehensive knowledge. Such a collection of triples, interconnected by values that are themselves resources, is also referred to as an *RDF description*. Figure 2.3 illustrates what such an RDF description might look like. Additionally, it is important to note that each of its components, whether it be a resource, property type, or value, does not necessarily have to be a digital concept. After all, Web assets can perfectly represent real-life concepts. (Miller, 1998) (Candan et al., 2001)

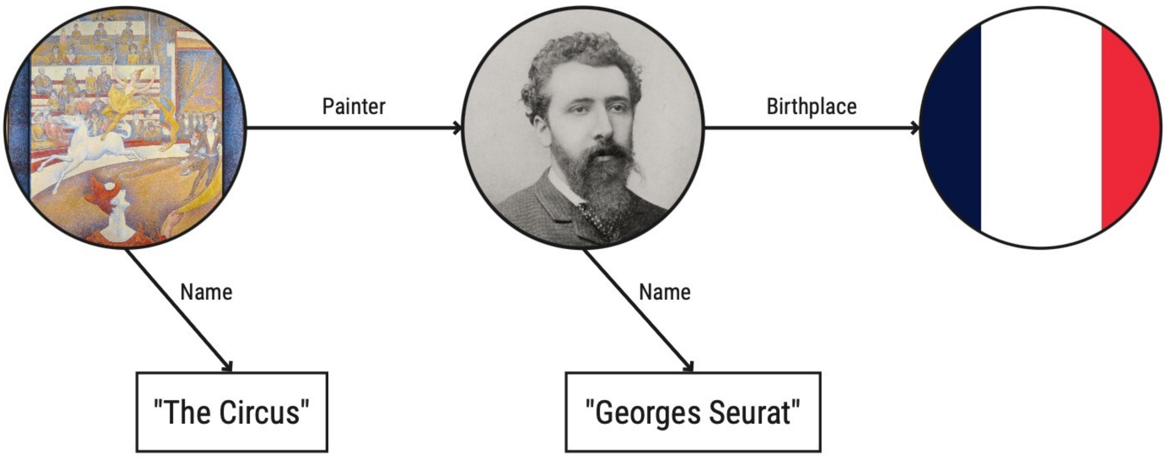

Figure 2.3: Representation of an RDF description

Circles represent resources, arrows represent property types and values are situated at the end of arrows

Clearly, different terms exist to denote the same RDF concepts. For instance, in addition to the synonyms mentioned above, in literature, the term *statement* is sometimes preferred over *triple*. However, in light of uniformity and clarity, throughout the rest of this text, the terms *triple*, *subject*, *predicate* and *object* will be used instead of their counterparts. (Candan et al., 2001)

2.1.3 Resource Description Framework Syntax

What constitutes RDF exactly, should be clear by now, but the question of how to actually write down RDF descriptions, still remains to be answered. Therefore, this section introduces some RDF syntaxes. However, since they are not the focus of this research, they will not be discussed in detail. Instead, their outlines will be illustrated by presenting the RDF description from Figure 2.3 in the syntax in question. Incidentally, since the schema presented in Figure 2.3 also has clear guidelines on how to be used, in itself, it also qualifies as an RDF syntax, albeit a graphical one. (Miller, 1998)

All the syntaxes to be discussed are instantiations of the RDF Model and Syntax Specification, providing concrete implementations. However, the first syntax stands apart from the rest as it primarily serves as a notation recommendation for humans to express RDF descriptions in a manner that is unambiguous yet simple. Unlike the other syntaxes, this particular one is not intended for machine consumption. Code Fragment 2.1 demonstrates how the RDF description, as schematically depicted in Figure 2.3, can be represented using this human-centric syntax. In this representation, resources are enclosed in straight brackets, while property types are represented by arrows. Furthermore, the representation of values varies depending on their types. As denoted, resources are encapsulated within brackets. However, if the values are atomic in nature, they are simply enclosed in quotation marks. (Miller, 1998)

```
[The Circus] ------name--------> "The Circus"
[The Circus] ------painter-----> [Georges Seurat]
[Georges Seurat] --name--------> "Georges Seurat"
[Georges Seurat] --birthplace--> [France]
```

Code Fragment 2.1: RDF description depicted using a human-centric RDF syntax

The example from Code Fragment 2.1 is easy to read, but at the same time rather confusing. Indeed, certain resource names correspond to certain atomic values. One could of course try to give the resources a more generic name to indicate what exactly the resource in question means. However, that would make little sense given the way the following machine-readable RDF syntaxes refer to resources. After all, they use URIs, allowing for a more clear distinction between resources and atomic values.

- **N-Triples**

  Code Fragment 2.2 depicts the representation of the RDF description using N-Triples. In this syntax, each line corresponds to a triple, wherein the subject, predicate, and object are delimited by spaces or tabs. The triple is terminated by a period and a new line character. (Beckett, 2014)

```
<http://example.org/The_Circus> <http://example.org/name> "The Circus" .
<http://example.org/The_Circus> <http://example.org/painter> <http://example.org/Georges_Seurat> .
<http://example.org/Georges_Seurat> <http://example.org/name> "Georges Seurat" .
<http://example.org/Georges_Seurat> <http://example.org/birthplace> <http://dbpedia.org/resource/France> .
```

Code Fragment 2.2: RDF description depicted using the N-Triples syntax

Furthermore, absolute URIs are employed to denote resources, while atomic values are enclosed within quotation marks. With that in mind, it is important to note that if a value itself contains a quotation mark, it must be properly escaped to ensure correct interpretation. (Beckett, 2014)

- **N3**

  Parsing an RDF description in N-Triples syntax is relatively straightforward for computers, but it can be challenging for humans to comprehend at a glance. The use of absolute URIs in N-Triples can lead to visual clutter and hinder readability. To address this, the N3 syntax builds upon N-Triples by introducing the concept of relative URIs. (Beckett, 2014)

  In N3, it is possible to specify a base URI by including a `@base <URI>` directive at the beginning of the document. When a relative URI is encountered elsewhere in the document, the parser appends it to the specified base URI. This allows for a more concise representation of URIs. (Berners-Lee and Connolly, 2011)

  However, RDF descriptions may contain URIs with different base URIs, making a single base URI insufficient. To overcome this limitation, N3 allows the document to be preceded by one or more `@prefix prefix: <URI>` directives. These directives associate prefixes with URIs, and the parser appends any relative URI preceded by a prefix to the corresponding base URI associated with that prefix. This mechanism enables the use of multiple base URIs within the same document and enhances the flexibility and expressiveness of the N3 syntax. Code Fragment 2.3 illustrates the use of prefixes for the N3 syntax. (Berners-Lee and Connolly, 2011)

```
@prefix ex: <http://example.org/> .
@prefix dbp: <http://dbpedia.org/resource/> .

ex:The_Circus ex:name "The Circus" .
ex:The_Circus ex:painter ex:Georges_Seurat .
ex:Georges_Seurat ex:name "Georges Seurat" .
ex:Georges_Seurat ex:birthplace dbp:France .
```

Code Fragment 2.3: RDF description depicted using the N3 and Turtle syntaxes

- **Turtle**

  The Turtle syntax is very similar to N3. In fact, Turtle is a subset of N3. Specifically, Code Fragment 2.3 can be processed by a Turtle parser just as well. However, while N3 allows for more expressiveness in principle, Turtle keeps things simpler, making it a popular choice for human readability. (Berners-Lee and Connolly, 2011) (Beckett et al., 2014)

Providing an exhaustive list of the precise differences between the two syntaxes would exceed the scope of this text since the intricacies of RDF syntaxes are not the primary focus here.

- **RDF/XML**

  RDF/XML is one of the earliest RDF syntaxes and remains widely used. To introduce this syntax, Code Fragment 2.4 serves as a guide.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/"
         xmlns:dbp="http://dbpedia.org/resource/">
  <rdf:Description rdf:about="http://example.org/The_Circus">
    <ex:name>The Circus</ex:name>
    <ex:painter rdf:resource="http://example.org/Georges_Seurat"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/Georges_Seurat">
    <ex:name>Georges Seurat</ex:name>
    <ex:birthplace rdf:resource="http://dbpedia.org/resource/France"/>
  </rdf:Description>
</rdf:RDF>
```

Code Fragment 2.4: RDF description depicted using the RDF/XML syntax

The RDF description in RDF/XML is enclosed within `rdf:RDF` elements, where necessary prefixes can also be defined. While an XML declaration like `<?xml version="1.0"?>` can precede the RDF/XML document, it is optional and omitted in Code Fragment 2.4 to focus primarily on the basics of RDF syntaxes. (Gandon et al., 2014)

Upon encountering the `rdf:RDF` tag, a parser recognizes that it should process an RDF description. In RDF/XML, such an RDF description is constructed using one or more `rdf:Description` elements. In fact, each `rdf:Description` element represents a subject, and its optional `rdf:about` attribute denotes the subject's URI. Consequently, the triples associated with the subject are enclosed within the corresponding `rdf:Description` tags. Predicates on the one hand, whether represented using a prefix or not, have their own elements. The representation of subjects, on the other hand, depends on their nature: for atomic values, they can simply be placed between opening and closing subject tags, while for resource subjects, their URIs are included as the value of an `rdf:resource` attribute within the subject tag. (Gandon et al., 2014)

Once again, it is important to note that the Code Fragments used in this section provide only an introductory glimpse of the proposed syntaxes. They cover only a small portion of the potential scope of a syntax. Code Fragment 2.4, in particular, demonstrates that RDF/XML syntax can obscure simplicity, especially when dealing with more extensive RDF descriptions. Consequently, RDF/XML is not commonly used for human-readable purposes but rather as a syntax primarily intended for machine consumption. (Dongo and Chbeir, 2019)

- **JSON-LD**

  The final RDF syntax introduced is called JSON-LD. Similar to RDF/XML, JSON-LD builds upon an existing syntax for representing data on the web. However, JSON-LD representations are generally more human-readable. As most resources and examples in the following text will be presented in JSON-LD, a slightly more comprehensive overview of this syntax is provided compared to the previous ones. Nevertheless, what follows is not an exhaustive listing of all the intricacies of the syntax. Instead, it aims to offer readers a concise introduction to JSON-LD without prior knowledge, making the rest of the text more easily comprehensible. For those seeking more in-depth information about JSON-LD, it is recommended to consult other sources[3].

  It is evident that the same data can be represented in various ways, and this applies to RDF data as well. While the visual representation of an RDF description, as depicted in Figure 2.3, is relatively straightforward, converting it into a fully textual format poses certain choices to be made. After all, there are numerous possibilities regarding the exact data representation. In the introduction of previous syntaxes, a specific representation was chosen each time. However, in this section, three different approaches for representing the same set of data using the JSON-LD syntax are presented.

  To start off, Code Fragment 2.5 closely resembles the previous examples, using nesting to store all the data in a single JSON-LD document. However, some may question whether it is appropriate to make the `George_Seurat` resource a child of `The_Circus` resource, implying a hierarchical relationship that may not be relevant.

  Subsequently, in Code Fragment 2.6, the data is split into two JSON-LD documents. Utilizing URIs, the documents can still refer to each other uniquely, without suggesting any hierarchical relationship between the resources.

  Finally, Code Fragment 2.7 takes a distinct approach by using the `@graph` property. This allows listing the necessary resources in a JSON array, placing them on equal footing within a single document. However, this method introduces extra clutter and overhead compared to the previous approaches. (Sporny et al., 2020)

[3] The W3C JSON-LD 1.1 Recommendation provides very in-depth information about the JSON-LD syntax.

```
{
  "@context": {
    "ex": "http://example.org/",
    "dbp": "http://dbpedia.org/resource/"
  },
  "@id": "ex:The_Circus",
  "ex:name": "The Circus",
  "ex:painter": {
    "@id": "ex:Georges_Seurat",
    "ex:name": "Georges Seurat",
    "ex:birthplace": "dbp:France"
  }
}
```

Code Fragment 2.5: RDF description with nested objects depicted using the JSON-LD syntax

```
Document 1:
{
  "@context": {
    "ex": "http://example.org/"
  },
  "@id": "ex:The_Circus",
  "ex:name": "The Circus",
  "ex:painter": "ex:Georges_Seurat"
}

Document 2:
{
  "@context": {
    "ex": "http://example.org/",
    "dbp": "http://dbpedia.org/resource/"
  },
  "@id": "ex:Georges_Seurat",
  "ex:name": "Georges Seurat",
  "ex:birthplace": "dbp:France"
}
```

Code Fragment 2.6: RDF description spread over two documents depicted using the JSON-LD syntax

Ultimately, the choice of representation depends on the specific use case and the desired balance between simplicity and expressiveness. Each approach has its advantages and trade-offs, showcasing the flexibility of the JSON-LD syntax in accommodating different data representation needs.

```
{
  "@context": {
    "ex": "http://example.org/",
    "dbp": "http://dbpedia.org/resource/"
  },
  "@graph": [
    {
      "@id": "ex:The_Circus",
      "ex:name": "The Circus",
      "ex:painter": {
        "@id": "ex:Georges_Seurat"
      }
    },
    {
      "@id": "ex:Georges_Seurat",
      "ex:name": "Georges Seurat",
      "ex:birthplace": {
        "@id": "dbp:France"
      }
    }
  ]
}
```

Code Fragment 2.7: RDF description as a graph depicted using the JSON-LD syntax

Understanding Code Fragments 2.5, 2.6, and 2.7 becomes relatively straightforward after having discussed the previous syntaxes. However, two aspects deserve further attention: the use of `@id` and `@context` keywords in JSON-LD.

Firstly, the `@id` keywords uniquely identify the proposed resources using URIs. Indeed, in the given examples, the id's do exactly that. (Sporny et al., 2020)

Secondly, the `@context` keyword plays a crucial role in JSON-LD. It introduces specifics that can be taken for granted in the actual data, reducing the need for repetitive information and cleaning up the actual JSON. While Code Fragments 2.5, 2.6, and 2.7 use the context in a straightforward way by introducing prefixes, in practice, it can do more than that. Essentially, the context maps terms to URIs. These terms can be freely chosen to enhance human readability. (Sporny et al., 2020)

W3C's JSON-LD Recommendation[4] offers a valuable example of how the context is typically used, as illustrated in Code Fragment 2.8. The provided context clearly indicates that when the key `name` appears in the data, it refers to `http://schema.org/name`. Similarly, for `image` and `homepage`, their respective values are *expanded* into objects that hold additional information. The `@type` keyword is also used in the example to indicate the type of the final value. In Code Fragment 2.8, it shows that the `image` and `homepage` keys are followed by an `@id`, representing unique resources. Moreover, JSON-LD supports various other types, and custom types can be defined to suit specific requirements. (Sporny et al., 2020)

```
{
  "@context": {
    "name": "http://schema.org/name",
    "image": {
      "@id": "http://schema.org/image",
      "@type": "@id"
    },
    "homepage": {
      "@id": "http://schema.org/url",
      "@type": "@id"
    }
  },
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Code Fragment 2.8: Example of context use in JSON-LD, proposed by Sporny et al. (2020)

To further enhance the cleanliness of a JSON-LD document, one can opt to store the context as a separate resource rather than embedding it directly in the document. Using this approach, the JSON-LD document includes the URI that references the context as the value for the `@context` key. Storing the context separately allows for greater modularity and reusability, making it easier to manage and maintain complex JSON-LD documents. The use of separate

[4] https://www.w3.org/TR/json-ld11/

contexts can significantly improve the organization and readability of JSON-LD data, enhancing its compatibility with RDF and Linked Data principles. (Sporny et al., 2020)

To finish off this section on JSON-LD, it is interesting to note that when the JSON-LD document presented in Code Fragment 2.8 is *expanded*, the data takes on its typical RDF form, adhering fully to the Linked Data principles. This expansion, as shown in Code Fragment 2.9, reveals the underlying structure of the data and its connection to other resources. (Sporny et al., 2020)

```
[{
  "http://schema.org/name": [{"@value": "Manu Sporny"}],
  "http://schema.org/url": [{ "@id": "http://manu.sporny.org/" }],
  "http://schema.org/image": [{ "@id": "http://manu.sporny.org/images/manu.png" }]
}]
```

Code Fragment 2.9: Example of an expanded JSON-LD document, proposed by Sporny et al. (2020)

In summary, the `@id` and `@context` keywords in JSON-LD contribute to the readability, expressiveness, and flexibility of representing RDF data, enabling a more human-friendly approach to data serialization.

Before concluding this section on RDF syntaxes, it is crucial to reiterate that the explanations provided are not exhaustive. Only a surface-level overview of these syntaxes was covered, and there is much more to explore and learn about them. This section serves as a reference for those with limited or no prior knowledge of RDF syntaxes, aiming to facilitate their understanding of the remaining text. In the following sections, several RDF examples will be presented, with the majority of them using the JSON-LD syntax. However, there will be no further elaboration on new elements that are specific to each syntax unless they are essential for a clear understanding of the text. For readers seeking a more in-depth understanding of the syntaxes, additional resources are recommended to further explore their intricacies and capabilities.

2.1.4 SPARQL

TODO

2.2 Link-Traversal-based Query Processing

TODO

2.3 Comunica

TODO

2.4 Collections of Ghent

TODO

2.4.1 Linked Data Event Streams

2.4.2 Example Queries

TODO

```
PREFIX cidoc:<http://www.cidoc-crm.org/cidoc-crm/>
PREFIX adms:<http://www.w3.org/ns/adms#>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX la:<https://linked.art/ns/terms/>
SELECT ?title ?note ?image ?objectname ?objectnumber ?associatie
       ?creator ?plaats ?timespan ?techniek ?materiaal
WHERE {
    # Title
    ?o cidoc:P102_has_title ?title.
    # Description
    ?o cidoc:P3_has_note ?note.
    # Image
    ?o cidoc:P129i_is_subject_of ?image.
    # Objectname
    ?o cidoc:P41i_was_classified_by ?classified.
    ?classified cidoc:P42_assigned ?assigned.
    ?assigned skos:prefLabel ?objectname.
    # Association
    ?o cidoc:P128_carries ?carries.
    ?carries cidoc:P129_is_about ?about.
    ?about cidoc:P2_has_type ?type.
    ?type skos:prefLabel ?associatie.
    # Objectnumber
    ?o adms:identifier ?identifier.
    ?identifier skos:notation ?objectnumber.
    # Creator
    ?o cidoc:P108i_was_produced_by ?production.
    ?production cidoc:P14_carried_out_by ?producer.
    ?producer la:equivalent ?equivalent.
    ?equivalent rdfs:label ?creator.
    # Place
    ?o cidoc:P108i_was_produced_by ?produced.
    ?produced cidoc:P7_took_place_at ?tookplace.
    ?tookplace la:equivalent ?plaatsequivalent.
    ?plaatsequivalent skos:prefLabel ?plaats.
    # Date
    ?o cidoc:P108i_was_produced_by ?produced.
    ?produced cidoc:P4_has_time-span ?timespan.
    # Technique
    ?o cidoc:P108i_was_produced_by ?produced.
    ?produced cidoc:P32_used_general_technique ?technique.
    ?technique cidoc:P2_has_type ?hastype.
```

3

CoGhent Data and Link Traversal

The primary focus of this research is the development of tools for constructing queries that target specific properties of CoGhent Human-Made Objects. These queries can either be confined to data within the CoGhent LDESs or extend beyond them by employing Link Traversal to follow links and traverse the corresponding documents. This approach facilitates the acquisition of new insights into the CoGhent data by not only enhancing the understanding of specific Human-Made Objects but also enabling their comparison in novel ways.

In the subsequent sections of this research, Comunica's link traversal capabilities will be utilized, as its modularity allows for the creation of link traversal engines tailored to the structure of the CoGhent data and the specific needs of this research. However, it is important to note that link traversal, despite its potential, remains an active area of research and can be configured in various ways.

This chapter therefore aims to explore the use of link traversal for discovering properties of Human-Made Objects, starting from the CoGhent LDESs. The chapter begins by providing an overview of the available data sources that can serve as starting points for the link traversal process. It then delves into the development of a link traversal engine optimized for the objectives outlined above. Finally, the chapter examines the most pertinent and intriguing types of resources to which the

CoGhent Human-Made Objects link. These resources will be crucial for achieving the goal of broadening the knowledge of the CoGhent data.

3.1 CoGhent Data Sources

CoGhent provides a set of LDESs for each participating institution. These LDESs are accessible through specific endpoints, as listed in Table 3.1

Table 3.1: CoGhent LDES endpoints as published by CoGhent (2022)

Publishing organisation	Endpoint URI
Design Museum Gent (DMG)	https://apidg.gent.be/opendata/adlib2eventstream/v1/dmg/objecten
Huis van Alijn (HVA)	https://apidg.gent.be/opendata/adlib2eventstream/v1/hva/objecten
Industriemuseum	https://apidg.gent.be/opendata/adlib2eventstream/v1/industriemuseum/objecten
STAM	https://apidg.gent.be/opendata/adlib2eventstream/v1/stam/objecten
Archief Gent	https://apidg.gent.be/opendata/adlib2eventstream/v1/archiefgent/objecten

3.1.1 URI Redirection

When accessing any of the URIs listed in Table 3.1, it is resolved to the same URI but with an additional query parameter `generatedAtTime`. For example, accessing the LDES from Industriemuseum results in the original URI being extended with `?generatedAtTime=2023-08-17T00:07:32.016Z`[1].

This behavior is confirmed by running the following command:

```
curl -i "https://apidg.gent.be/opendata/adlib2eventstream/v1/industriemuseum/objecten"
```

This returns an HTTP `302 Found` response code and a `Location` header with the extended URI, indicating a redirect to that link. Eventually, when a client (e.g. a browser or Comunica) sends a `GET` request to the updated link, the server returns the last (most recent) page of the requested LDES in JSON-LD format. (MDN Web Docs, 2023)

[1] Since the query parameter's value is time-dependent, this specific value serves only as an example of how it is structured.

3.1.2 Non-deterministic results

When configuring a query engine, any or multiple of the CoGhent endpoints can be chosen as data sources, depending on the specific data of interest. Naturally, due to the nature of LDESs, the same query should never be assumed to yield the same results across multiple executions. However, even when running the same query multiple times in a row with the certainty that the LDES hasn't updated yet, the results will still differ in terms of content and order. This variability is attributed to the nature of LTQP and Comunica's implementation of it. After all, results are influenced by the order in which links are pushed to the link queue, which in turn is influenced by the time it takes for the corresponding HTTP requests to get resolved.

This phenomenon is demonstrated by running the query displayed in Code Fragment 3.1[2] twice, using Design Museum Gent's LDES as data source. Tables 3.2 and 3.3 show, for both executions respectively, each result's IIIF Manifest URI, as well as the order in which the results were returned. Comparing both outputs clearly proves the results from the two executions differ in both content and order.

```
PREFIX iiif: <http://iiif.io/api/presentation/2#>
PREFIX cidoc:<http://www.cidoc-crm.org/cidoc-crm/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX w3-exif: <http://www.w3.org/2003/12/exif/ns#>
PREFIX w3-oa: <http://www.w3.org/ns/oa#>
SELECT ?manifest ?height ?image
WHERE {
  # Manifest URI
  ?human_made_object cidoc:P129i_is_subject_of ?manifest.
  # Image height
  ?manifest iiif:hasSequences/rdf:first/iiif:hasCanvases/rdf:first/w3-exif:height ?height.
  # Image URI
  ?canvas iiif:hasImageAnnotations/rdf:first/w3-oa:hasBody ?image.
}
LIMIT 10
```

Code Fragment 3.1: SPARQL query fetching ten Human-Made Object's IIIF Manifest URIs, image heights and image file URIs

For similar reasons, the order in which CoGhent endpoint URIs are given to the engine as data sources does not necessarily imply that one endpoint's data has priority over the other. This is illustrated by running the same query (see Code Fragment 3.1) with the Design Museum Gent LDES first and the Huis Van Alijn LDES second, and then reversing the order. The

[2] The query's specifics are discussed in Section 3.3.1.

Table 3.2: (Part of) results after **first** execution of query displayed in Code Fragment 3.1

	IIIF Manifest URI
1	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3086_3-5
2	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:1992-0068
3	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3130
4	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:1990-0051_0-5
5	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3054
6	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3124
7	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0284
8	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0296
9	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0305
10	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0281_21-21

Table 3.3: (Part of) results after **second** execution of query displayed in Code Fragment 3.1

	IIIF Manifest URI
1	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3075
2	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0305
3	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:3054
4	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:1563
5	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:1987-0447
6	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:1987-1127_1-2
7	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0271
8	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0284
9	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2018-0296
10	https://api.collectie.gent/iiif/presentation/v2/manifest/dmg:2990_0-4

results from both executions, as shown in Tables 3.4 and 3.5 respectively, once again show variations in content and order, yet most importantly don't seem to show any notable correlation to the order in which the endpoints were given to the engine.

Table 3.4: (Part of) results after execution of query displayed in Code Fragment 3.1 with Design Museum Gent (**DMG**) LDES endpoint as **first** data source and Huis Van Alijn (**HVA**) LDES endpoint as **second** data source

	IIIF Manifest URI
1	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-015
2	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2015-024-001
3	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:3223
4	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:3086_3-5
5	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:1563
6	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-001
7	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:1987-1127_2-2
8	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-002
9	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:1987-0447
10	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-003

3.1.3 Duplicate Human-Made Objects

It is also important to note that, since updates to an LDES object are performed by adding a new version of the object to the LDES, it is possible to receive multiple results for the same Human-Made Object. As discussed in Secion 2.4.2, a potential workaround would be to use a combination of `distinct` and `order by` clauses in the query itself, to only retrieve the newest versions. However, since ordering can only occur when all results are in, this approach prevents them from appearing in a *streaming* manner. A more efficient solution, therefore, is to let the application that initiated the query, keep track of Human-Made Object URIs while the results are coming in. That way, when the application encounters duplicate Human-Made Objects, it can decide to only retain the latest version's results. Since implementing such a solution is considered trivial, the issue will not be discussed further in this research.

Table 3.5: (Part of) results after execution of query displayed in Code Fragment 3.1 with Huis Van Alijn (**HVA**) LDES endpoint as **first** data source and Design Museum Gent (**DMG**) LDES endpoint as **second** datasource

	IIIF Manifest URI
1	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-002
2	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-001
3	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2014-031-003
4	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2009-018-568
5	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2009-018-568
6	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2015-024-004
7	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:2018-0261
8	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:2990_4-4
9	https://api.collectie.gent/iiif/presentation/v2/manifest/**dmg**:2018-0260
10	https://api.collectie.gent/iiif/presentation/v2/manifest/**hva**:2015-024-001

3.1.4 Conclusion

In conclusion, the CoGhent LDES endpoints work perfectly well to initiate the Link Traversal-based Querying process from. Each institution having a separate LDES is an added bonus, as this gives users the flexibility to choose which institutions' data to query. However, it is essential to be aware that the results and order of results are not predictable due to the nature of LDESs as well as Comunica's LTQP implementation. Additionally, Human-Made Objects are spread over multiple pages in the LDES, which needs to be taken into consideration when building the Comunica link traversal engine configuration.

3.2 Comunica Link Traversal Engine Configuration

As discussed in Section 2.3, the Comunica engine offers a wide range of configurability for link traversal. Numerous link traversal-specific actors have been developed. Some of those have already matured, while others are still in active development. In this section, some of these actors will be considered for configuring a Comunica link traversal engine that meets the requirements of this research, as well as performs up to a standard that is acceptable for real-world use. The resulting configuration should ultimately determine the engine used throughout the rest of this research.

3.2.1 Base Configuration

The Comunica Link Traversal repository[3] already provides several predefined configurations[4] that are *out of the box* available to Comunica users to kick-start with LTQP. A common feature of these configurations is the initial import of `config-base.json`[5]. This configuration file imports all actors and mediators necessary for the basic functionality of a Comunica Link Traversal engine, such as HTTP fetching, query operations, and RDF parsing. In other words, such a base configuration is essential to having a working link traversal engine. However, since this research does not focus on these basic functionalities, the intricacies of setting up a base configuration won't be discussed further. Rather, as is the case with the predefined configurations, the configuration specific to this research will also start with importing the `config-base.json` file.

3.2.2 Basic Link Extractors

The most important type of actors that should be considered when setting op a link traversal engine, are arguably the link extractors. When a new RDF document is encountered during the link traversal process, these actors determine which links from that document should be added to the link queue. In other words, they are the ones deciding which resources should be queried.

The most basic link extractor is the *All Extract Links Actor*[6]. This actor essentially implements the *cAll* criterion, as discussed in Section 2.2. Simply put, it adds all links it encounters to the link queue. However, this approach is not suitable for the purposes of this research, as it may lead to traversing too many documents that will most certainly not aid in resolving the query at hand, in turn leading to impractical execution times.

As already discussed, this research focuses on queries that fetch data specific to Human-Made Objects. This means that the specific paths to follow - starting from a Human-Made Object and ending in the object of interest - are known beforehand. In other words, the queries already specify these *sequences of predicates*, allowing for a more targeted approach. Therefore, another interesting link extractor to consider, is the *Quad Pattern Query Extract Links Actor*[7]. Essentially, this actor is an implementation of the *cMatch* criterion that was discussed in Section 2.2. It adds only those links to the link queue that are part of quads that match at least one quad pattern in the query. Given the knowledge of the starting subject - Human-Made objects - and the specific sequence of predicates to follow, this actor should better *guide* the engine in the right direction,

[3] https://github.com/comunica/comunica-feature-link-traversal

[4] https://github.com/comunica/comunica-feature-link-traversal/tree/master/engines/config-query-sparql-link-traversal/config

[5] https://github.com/comunica/comunica-feature-link-traversal/blob/master/engines/config-query-sparql-link-traversal/config/config-base.json

[6] https://github.com/comunica/comunica-feature-link-traversal/tree/master/packages/actor-extract-links-all

[7] https://github.com/comunica/comunica-feature-link-traversal/tree/master/packages/actor-extract-links-quad-pattern-query

leading to faster results. However, it is possible for certain documents to, by change, contain quads that don't lead to the data the query was set up for, still leading to *wrong* documents being visited.

3.2.3 Extracting Links based on Predicates

Having in mind that sequences of predicates are already known beforehand, the most promising link extractor is the *Predicates Extract Links Actor*[8]. This type of link extractor was not discussed before, but its workings are straightforward. Essentially, for every quad in a document, the actor only considers objects. Apart from the object naturally needing to be a URI, the only links that are added to the link queue are those objects' links that have a predicate matching one of the regexes set in the actor's configuration. In other words, the sequences of predicates that define the queries considered in this research, can literally serve as the regexes this actor uses to evaluate predicates. Additionally, the *rules* can even be tightened by obliging every quad's subject to match the URI of the document currently being processed. This extra requirement further narrows down the selection of links to follow, potentially speeding up the querying process even further.

To test this approach, a Comunica link traversal query engine is built using the configuration as depicted in Code Fragment 3.2, in turn tasked with resolving the query displayed in Code Fragment 3.1. Once again, the data source is set to the Design Museum Gent LDES endpoint. As can be seen in Code Fragment 3.2, the configuration's second import is a custom configuration file. This file is displayed in Code Fragment 3.3 and not only tasks the engine being built to use the Predicates Extract Links Actor, but also instructs this link actor to only consider object links whose predicates match the query's predicates and whose subjects match the current document's URI. The keys that specify these settings are respectively called `predicateRegexes` and `checkSubject`.

However, after building the engine and instructing it to resolve the query, no results are returned. To uncover the reason for this failure, the logs[9] outputted by the engine during execution and displayed in Code Fragment 3.4, can be consulted. From these logs, it can be inferred that the engine initially fetches the provided data source, in this case, the Design Museum Gent LDES. Then, it retrieves the documents referenced in the context of the LDES in order to expand the LDES. Finally, once this expansion is completed successfully, the LDES is marked as *identified*. As for the rest of the logs, there are no significant actions taking place. In other words, no other documents are identified, let alone requested. From this, it can be deduced that no links are being added to the link queue while traversing the LDES. This suggests that the configuration of the Predicates Extract Links Actor needs to be reviewed.

[8] https://github.com/comunica/comunica-feature-link-traversal/tree/master/packages/actor-extract-links-predicates

[9] Logging can be enabled as explained here: https://comunica.dev/docs/query/advanced/logging/.

```
{
  "@context": [
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/config-query-sparql/^2.0.0/components/context.jsonld",
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/config-query-sparql-link-traversal/^0.0.0/components/context.jsonld"
  ],
  "import": [
    "ccqslt:config/config-base.json",
    "./actors/extract-links-predicates-custom.json",
  ]
}
```

Code Fragment 3.2: Comunica link traversal engine configuration using Predicates Extract Links Actor

Through debugging, it can be determined that only two quads pass the test comparing their subject URIs to the URI of the current document, in this case the LDES. These quads in question are both TREE-related quads - as mentioned in Section 2.4.1, LDESs are built on the TREE specification. It comes as no surprise that these quads fail the subsequent test that compares predicates with the provided regexes. However, the fact that only these two quads pass the first test, and every other quad fails, highlights why the configuration of the Predicates Extract Links Actor, as shown in Code Fragment 3.3, doesn't work for the query presented in Code Fragment 3.1: since the *starting point* of the query is expected to be a Human-Made Object subject - the first predicate `cidoc:P129i_is_subject_of` achieves this as only Human-Made Object subjects have this predicate in the LDES - these subjects will never match the URI of the LDES. As a result, the Predicates Extract Links Actor will disregard these quads.

One possible solution is to modify the query by providing the LDES page itself as the *starting point* and extending the sequence of predicates to *bridge the gap* between the LDES root node and the Human-Made Objects. However, as part of the aim of this research is to assist people without a technical background in constructing and better comprehending queries, making the queries unnecessarily long and complex is not desirable. Consequently, the decision is made to set the `checkSubject` key in the configuration of the Predicates Extract Links Actor to `false`. This ultimately leads to the configuration presented in Code Fragment 3.5.

```json
{
  "@context": [
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/runner/^2.0.0/components/context.jsonld",
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/actor-extract-links-predicates/^0.0.0/components/context.jsonld"
  ],
  "@id": "urn:comunica:default:Runner",
  "@type": "Runner",
  "actors": [
    {
      "@id": "urn:comunica:default:extract-links/actors#predicates-common",
      "@type": "ActorExtractLinksPredicates",
      "checkSubject": true,
      "predicateRegexes": [
        "http://www.cidoc-crm.org/cidoc-crm/P129i_is_subject_of",
        "http://iiif.io/api/presentation/2#hasSequences",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#first",
        "http://iiif.io/api/presentation/2#hasCanvases",
        "http://www.w3.org/2003/12/exif/ns#height",
        "http://iiif.io/api/presentation/2#hasImageAnnotations",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#first",
        "http://www.w3.org/ns/oa#hasBody"
      ]
    }
  ]
}
```

Code Fragment 3.3: Comunica Predicates Extract Links Actor configuration with predicate regexes set to predicates from query displayed in Code Fragment 3.1 and subject checking **enabled**

```
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             dmg/objecten
             { ... , method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
...
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/cultureel-erfgoed-object-ap.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/persoon-basis.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/cultureel-erfgoed-event-ap.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/organisatie-basis.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/generiek-basis.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Requesting
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             context/dossier.jsonld
             { ..., method: 'GET', actor: 'urn:comunica:default:http/actors#fetch' }
[...]  INFO: Identified as file source:
             https://apidg.gent.be/opendata/adlib2eventstream/v1/
             dmg/objecten?generatedAtTime=2023-08-12T00:01:27.217Z
             { actor: 'urn:comunica:default:rdf-resolve-hypermedia/actors#none' }
...
```

Code Fragment 3.4: (Cleaned up) logs outputted during execution of engine configured by files displayed in Code Fragments 3.2 and 3.3

```json
{
  "@context": [
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/runner/^2.0.0/components/context.jsonld",
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/actor-extract-links-predicates/^0.0.0/components/context.jsonld"
  ],
  "@id": "urn:comunica:default:Runner",
  "@type": "Runner",
  "actors": [
    {
      "@id": "urn:comunica:default:extract-links/actors#predicates-common",
      "@type": "ActorExtractLinksPredicates",
      "checkSubject": false,
      "predicateRegexes": [
        "http://www.cidoc-crm.org/cidoc-crm/P129i_is_subject_of",
        "http://iiif.io/api/presentation/2#hasSequences",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#first",
        "http://iiif.io/api/presentation/2#hasCanvases",
        "http://www.w3.org/2003/12/exif/ns#height",
        "http://iiif.io/api/presentation/2#hasImageAnnotations",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#first",
        "http://www.w3.org/ns/oa#hasBody"
      ]
    }
  ]
}
```

Code Fragment 3.5: Comunica Predicates Extract Links Actor configuration with predicate regexes set to predicates from query displayed in Code Fragment 3.1 and subject checking **disabled**

3.2.4 Comparing Link Extractors

In an attempt to compare the discussed link extractors not only in terms of functionality but also in terms of performance, a small experiment is conducted. Similar to before, the query shown in Code Fragment 3.1 is used, with the Design Museum Gent LDES serving as the data source. The first engine utilizes the All Extract Links Actor, the second one employs the Predicates Extract Links Actor, and the third utilizes the Predicates Extract Links Actor in the configuration outlined in Code Fragment 3.5. Consequently, the final engine configurations corresponded to the existing *Follow All*[10] and *Follow Match Query*[11] configurations present in the Comunica Link Traversal GitHub repository, along with the custom configuration as illustrated in Code Fragment 3.2. To ensure reliability, each engine executes the query consecutively three times, with the engine's complete HTTP cache being invalidated after each run. The outcomes of the experiment are presented in Table 3.6.

Table 3.6: Results from experiment comparing different Comunica link traversal engines

Engine	Total time (s)	Average time single execution (s)
Follow All	Runtime error	Runtime error
Follow Match Query	66.08	22.03
Custom (using configuration displayed in Code Fragment 3.5)	54.34	18.11

The results immediately indicate that the Follow All engine struggles to execute the query successfully. It is important to note that the success rate is subject to a variety of factors, encompassing both client and server circumstances, such as the machine's specifications and the state of the internet connection. However, in this specific instance, the runtime error that emerged following unsuccessful link traversal was attributed to an excessive number of listeners assigned to a TLS socket. This situation may be associated with an overflow of HTTP requests. The combination of this issue with the absence of any valid results even after a considerable time span underscores that, for the objectives of this research, the Follow All engine, without additional configuration or the integration of supplementary actors, is unsuitable.

Fortunately, both the Follow Match Query and Custom engines were able to successfully execute their tasks. It is noteworthy, however, that the average times to resolve the query differ by only a few seconds. As expected, the custom engine performs better, but the marginal time saved initially might not seem significant compared to the drawback of having to adjust its configuration for each query. Nevertheless, it is reasonable to expect that the custom engine's advantage will become more

[10] https://github.com/comunica/comunica-feature-link-traversal/blob/master/engines/config-query-sparql-link-traversal/config/config-follow-all.json

[11] https://github.com/comunica/comunica-feature-link-traversal/blob/master/engines/config-query-sparql-link-traversal/config/config-follow-match-query.json

pronounced when handling queries that target data distributed across multiple documents and situated at deeper levels. Moreover, it is entirely feasible to develop a user-friendly application that constructs the necessary configuration based on the specific query before executing the engine. This way, the configuration complexity can be abstracted from the end-users, providing a smoother user experience while harnessing the benefits of the custom engine's efficiency.

3.2.5 Traversing LDES Pages

Despite the custom engine's capacity to retrieve highly targeted data, its present form only accounts for a fraction of the available dataset. This limitation arises from the fact that an LDES comprises multiple pages, technically TREE nodes, necessitating both forward and backward *browsing* to encompass the entirety of the dataset. However, the predicates leading to the objects providing access to these other TREE nodes are presently absent from the regex array in the configuration of the Predicates Extract Links Actor.

While incorporating these predicates is straightforward, an even more effective approach involves introducing a second link extractor to the custom engine configuration. The *Extract Links Tree Actor*[12] possesses the capability to introduce links to the preceding and succeeding LDES *pages* – as identified by the *greater than* and *less than* relationships as defined within the TREE specification – into the link queue. This modest addition to the configuration profoundly enhances the capabilities of the resultant engine.

The revised configuration, as presented in Code Fragment 3.6, not only facilitates finely targeted searches for the requested data but also encompasses the complete dataset of the specified CoGhent institution(s) by leveraging the Extract Links Tree Actor.

3.2.6 Conclusion

In summary, an engine constructed using the Follow Match Query configuration, which utilizes the Quad Pattern Query Extract Links Actor, effectively addresses specific query requirements without necessitating additional actor configuration. However, for queries demanding more extensive traversal across documents or encompassing data distributed across multiple documents, a tailored configuration that integrates both the Predicates Extract Links Actor and the Extract Links Tree Actor can significantly enhance performance.

It is important to acknowledge that this approach does require a specific configuration outlining the predicates for each query. Nevertheless, this configuration complexity can be effectively abstracted from end-users through the development

[12] https://github.com/comunica/comunica-feature-link-traversal/tree/master/packages/actor-extract-links-extract-tree

```json
{
  "@context": [
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/config-query-sparql/^2.0.0/components/context.jsonld",
    "https://linkedsoftwaredependencies.org/bundles/npm/
        @comunica/config-query-sparql-link-traversal/^0.0.0/components/context.jsonld"
  ],
  "import": [
    "ccqslt:config/config-base.json",
    "./actors/extract-links-predicates-custom.json",
    "ccqslt:config/extract-links/actors/tree.json",
  ]
}
```

Code Fragment 3.6: Comunica link traversal engine configuration using Predicates Extract Links Actor and Extract Links Tree Actor

of tools that manage the technical intricacies behind the scenes. This approach ultimately strikes a balance between query performance optimization and user accessibility, aligning with the overarching goals of the research.

3.3 Links to Follow

TODO

3.3.1 IIIF Manifest

TODO

3.3.2 Wikidata

TODO

3.3.3 Stad Gent

TODO

3.3.4 Getty Vocabularies

TODO

3.4 Conclusion

4

Tools for Query Building

TODO

4.1 Building Queries from Predicate Sequences

TODO (see LateX comments for notes)

```
objectname: [
    '?o cidoc:P41i_was_classified_by ?classified.',
    '?classified cidoc:P42_assigned ?assigned.',
    '?assigned skos:prefLabel ?objectname.',
]
```

Code Fragment 4.1: WHERE clause statements to query for *objectname* stored as elements in array

```
PREFIX cidoc:<http://www.cidoc-crm.org/cidoc-crm/>
PREFIX adms:<http://www.w3.org/ns/adms#>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX la:<https://linked.art/ns/terms/>
```

Code Fragment 4.2: All possible PREFIX statements of original CoGhent Query Builder

```
objectname: [
    { prefix: 'cidoc', predicate: 'P41i_was_classified_by' },
    { prefix: 'cidoc', predicate: 'P42_assigned' },
    { prefix: 'skos', predicate: 'prefLabel' },
]
```

Code Fragment 4.3: Prefixes and predicates for WHERE clause statements to query for *objectname* stored as elements in array

```
# Objectname
?o cidoc:P41i_was_classified_by ?s1.
?s1 cidoc:P42_assigned ?s2.
?s2 skos:prefLabel ?s3.
```

Code Fragment 4.4: WHERE clause statements with object variable names constructed using numbers

```
# Objectname
?o
    cidoc:P41i_was_classified_by
        ?o_cidoc_P41i_was_classified_by.
?o_cidoc_P41i_was_classified_by
    cidoc:P42_assigned
        ?o_cidoc_P41i_was_classified_by_cidoc_P42_assigned.
?o_cidoc_P41i_was_classified_by_cidoc_P42_assigned
    skos:prefLabel
        ?o_cidoc_P41i_was_classified_by_cidoc_P42_assigned_skos_prefLabel.
```

Code Fragment 4.5: WHERE clause statements with object variable names constructed from preceding statements

```
# Place
?o cidoc:P108i_was_produced_by ?produced.
?produced cidoc:P7_took_place_at ?tookplace.
?tookplace la:equivalent ?plaatsequivalent.
?plaatsequivalent skos:prefLabel ?plaats.
# Date
?produced cidoc:P4_has_time-span ?timespan.
```

Code Fragment 4.6: WHERE clause statements without overlapping statements

```
# Place
?o cidoc:P108i_was_produced_by ?produced.
?produced cidoc:P7_took_place_at ?tookplace.
?tookplace la:equivalent ?plaatsequivalent.
?plaatsequivalent skos:prefLabel ?plaats.
# Date
?o cidoc:P108i_was_produced_by ?produced.
?produced cidoc:P4_has_time-span ?timespan.
```

Code Fragment 4.7: WHERE clause statements with overlapping statements

```
const properties = {
    title: [
        { prefix: 'cidoc', predicate: 'P102_has_title' }
    ],
    description: [
        { prefix: 'cidoc', predicate: 'P3_has_note', object_variable_name: 'note' },
    ],
    objectname: [
        { prefix: 'cidoc', predicate: 'P41i_was_classified_by' },
        { prefix: 'cidoc', predicate: 'P42_assigned' },
        { prefix: 'skos', predicate: 'prefLabel' },
    ],
    association: [
        { prefix: 'cidoc', predicate: 'P128_carries' },
        { prefix: 'cidoc', predicate: 'P129_is_about', object_variable_name: 'about' },
        { prefix: 'cidoc', predicate: 'P2_has_type' },
        { prefix: 'skos', predicate: 'prefLabel' },
    ],
};

const prefixes = {
    cidoc: 'http://www.cidoc-crm.org/cidoc-crm/',
    skos: 'http://www.w3.org/2004/02/skos/core#',
};
```

Code Fragment 4.8: Properties and prefixes ready to be consumed by query building function

```
PREFIX cidoc:<http://www.cidoc-crm.org/cidoc-crm/>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>

SELECT ?title ?note ?objectname ?association

WHERE {
    # title
    ?human_made_object cidoc:P102_has_title ?title.

    # description
    ?human_made_object cidoc:P3_has_note ?note.

    # objectname
    ?human_made_object
        cidoc:P41i_was_classified_by/cidoc:P42_assigned/skos:prefLabel
            ?objectname.

    # association
    ?human_made_object cidoc:P128_carries/cidoc:P129_is_about ?about.
    ?about cidoc:P2_has_type/skos:prefLabel ?association.
}
```

Code Fragment 4.9: SPARQL query generated from input displayed in Code Fragment 4.8

```
const properties = {
    title: {
        statements: [
            { prefix: 'cidoc', predicate: 'P102_has_title' }
        ],
    },
    description: {
        statements: [
            { prefix: 'cidoc', predicate: 'P3_has_note', object_variable_name: 'note' },
        ],
        filters: { string: 'luchter', language: 'nl' },
        optional: true,
    },
};
```

Code Fragment 4.10: Example of properties dictionary to illustrate use of filters and optionals

```
PREFIX cidoc:<http://www.cidoc-crm.org/cidoc-crm/>

SELECT ?title ?note

WHERE {
    # title
    ?human_made_object cidoc:P102_has_title ?title.

    # description
    OPTIONAL {
        ?human_made_object cidoc:P3_has_note ?note.

        FILTER(REGEX(?note, "luchter", "i"))
        FILTER(LANG(?note) = "nl")
    }
}
```

Code Fragment 4.11: SPARQL query generated from input displayed in Code Fragment 4.10

4.2 A New Query Builder

TODO

4.3 Discovering Predicate Sequences

TODO

5

Handling Query Results

TODO

5.1 Visualizing Query Results

TODO

5.2 Saving Query Results

TODO

Conclusion

TODO

Ethical and social reflection

TODO

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*, pages 722–735. Springer.

Beckett, D. (2014). RDF 1.1 n-triples. W3C recommendation, W3C. https://www.w3.org/TR/2014/REC-n-triples-20140225/.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 turtle. W3C recommendation, W3C. https://www.w3.org/TR/2014/REC-turtle-20140225/.

Berners-Lee, T. (2006). Linked data-design issues. *http://www.w3.org/DesignIssues/LinkedData.html*.

Berners-Lee, T. and Connolly, D. (2011). Notation3 (n3): A readable rdf syntax. W3C team submission, W3C. http://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/.

Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global.

Candan, K. S., Liu, H., and Suvarna, R. (2001). Resource description framework: metadata and its applications. *Acm Sigkdd Explorations Newsletter*, 3(1):6–19.

CoGhent (2022). Linked data event streams. https://coghent.github.io/apiendpoints.html.

Dongo, I. and Chbeir, R. (2019). S-RDF: A New RDF Serialization Format for Better Storage Without Losing Human Readability. In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences*, 28th International Conference on COOPERATIVE INFORMATION SYSTEMS, pages 246–264, Rhodes, Greece. Springer International Publishing.

Gandon, F., Schreiber, G., and Becket, D. (2014). RDF 1.1 XML syntax. W3C recommendation, W3C. https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

Golbeck, J. and Rothstein, M. (2008). Linking social networks on the web with foaf: A semantic web case study. In *AAAI*, volume 8, pages 1138–1143.

Jacksi, K. and Abass, S. M. (2019). Development history of the world wide web. *Int. J. Sci. Technol. Res*, 8(9):75–79.

MDN Web Docs (2023). 302 found. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/302.

Miller, E. (1998). An introduction to the resource description framework. ディジタル図書館, 13:3–11.

Powers, S. (2003). *Practical RDF: solving problems with the resource description framework.* O'Reilly Media, Inc.

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., Champin, P.-A., and Lindström, N. (2020). JSON-ld 1.1. W3C recommendation, W3C. https://www.w3.org/TR/2020/REC-json-ld11-20200716/.

Van de Vyvere, B., D' Huynslager, O. V., Atauil, A., Segers, M., Van Campe, L., Vandekeybus, N., Teugels, S., Saenko, A., Pauwels, P.-J., and Colpaert, P. (2022). Publishing cultural heritage collections of ghent with linked data event streams. In *Metadata and Semantic Research: 15th International Conference, MTSR 2021, Virtual Event, November 29–December 3, 2021, Revised Selected Papers*, pages 357–369. Springer.

Appendices

Appendix A

TODO

Appendix B

TODO