Unit-process data modelling within the 'Industrial Ecology Data Commons'

Thomas R. A. Millross

Under the supervision of

Arjan de Koning

Leiden University, Faculty of Science, Institute of Environmental Sciences (CML)

&

Martijn Warnier

TU Delft, Faculty of Technology Policy and Management, Systems Engineering

This dissertation is submitted for the degree of

Master of Science

In the field of

Industrial Ecology

Document date: 2019-02-06

Thesis defence scheduled: 2019-02-13

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Thomas R. A. Millross

Acknowledgements

I owe huge thanks to my super supervisors Martijn and Arjan. I invariably looked forward to our frequent meetings (except that pesky midterm!), where I received challenging and fair critique. You've been friendly, professional and immeasurably helpful with excellent feedback and guidance, helping us to progress from kick-off to defence in less than 20 weeks. I strongly recommend you both to any other students that are considering relevant thesis topics, and hope we may work together again in the future.

To Kiki the coordination king, thank you for your hard work and brilliant organisation with warmth and patience. Also, to the other heroes behind the scenes at CML including Paula, Susanne and Joyce, who enable the functioning of everything else.

To Stefan Pauliuk, thank you for our challenging conversations in-person, video-call and email. And for providing access to not only the early versions your manuscript, but also direct database access! In flying the banner for open-science in Industrial Ecology, you clearly practise what you preach and are an example to others.

And finally, to my friends, family and partner. When I decided to give up my career in London to study a niche science in the Netherlands, I had little idea what to expect. Your support, encouragement and friendship have helped to make this one of the best decisions I ever made. Immense gratitude to you all ☺

Summary

The Industrial Ecology Data Commons (IEDC) is a relational database designed as an open "platform to exchange industrial ecology datasets" (Pauliuk et al., forthcoming). It is positioned as an implementation of the 'general data model (GDM) for socioeconomic metabolism', which the authors state "can be used to structure all data that can be located in the industrial system". To evaluate the representative capabilities of the GDM and IEDC, a source dataset of *unit-processes* is mapped to their datamodel, and imported into the database. The source data selected for this purpose is a version of the ecoinvent database, which has been transformed into the *minimum consensus knowledge model* for LCA data (Kuczenski et al. 2016).

Three primary insights have been found through this research. Firstly, it is possible to represent unit-process data within the IEDC. However, there are numerous shortcomings, which lead to the evaluation that this representation is not *effective*. Second, it was found that the GDM is probably not described sufficiently clearly and explicitly to allow for the development of alternative interoperable implementations, using technologies other than those used within the IEDC. Finally, the IEDC acts simultaneously as both a *datamodel repository,* and a *database* for storing the data which conforms to those models; a design pattern which permits the actual data modelling process to be deferred to the data-loading phase. Relational databases function best when their datamodel (and hence structure) is determined *before* the loading of data. As such, this report recommends that further work is required to satisfy the full design requirements of an effective platform for IE dataset exchange.

List of Figures

List of Tables

Table of contents

Chapter 1 Research overview

1.1 Introduction

This research aims to contribute to the science of *industrial ecology* through the synthesis of knowledge and recommendations on the topic of domain-specific data modelling. It builds primarily upon one recent work, a paper titled "A General Data Model for Socioeconomic Metabolism and its Implementation in an Industrial Ecology Data Commons Prototype" (Pauliuk et al., forthcoming)[1]. The development of this 'general data model' (**GDM**) was motivated by the same perspective as this thesis research: a recognition of the suboptimal data modeling and management practises within the field. The 'industrial ecology data commons' (**IEDC**) is a practical prototype of the GDM, in the form of a relational database. It aims to address a "data integration and exchange problem", which is caused by the "lack of cross-method data formats and platforms".

The GDM and IEDC are recent conceptions which have seen limited usage, testing and independent assessment. At the time of writing, they have not been used in any practical applications or case studies. Once datamodels and databases are adopted, software and users become dependent on them. Any modifications may be difficult for the users to accept or adjust to after their research practises become interconnected with the data. In the words of Tupper (2011): "it is preferable to change a designed structure before a system is built, since

[1] An early-version of the paper is quoted throughout this report, with permission of the lead-author. This version was accepted to the Journal of Industrial Ecology with major revisions in December 2018. Those revisions are ongoing at the time this report is submitted, hence any quotations may not appear in the final published version. However, the main discussion items and conclusions of this work focus on the IEDC database, and are unlikely to be significantly affected by manuscript edits.

design changes generally cost significantly less than application code changes". As such, the timing is ideal for a critical appraisal of the datamodels, as presented in this work.

This introductory chapter begins by providing background context on two related areas: the usage of data, information and systems in industrial ecology; and the movement within the field toward *open science* and *data*. The identified knowledge gap stems from statements in the literature regarding the capabilities of the datamodels. The next subsection describes the research specification, including the objective, research questions and scope. The chapter concludes with a sturctural overview of the remainder of the thesis.

1.2 Research background

Industrial ecology is a science which models sociometabolic systems using a wide variety of information and data. This background section begins by describing how these concepts interrelate, to help position the work within the wider context. The standard working practises of the research community are addressed next, within the context of the broad movement in science toward increased openness, and specifically *open data* within IE. The GDM and IEDC which are the focus of this reseach are intended as open datamodels and databases for socioeconomic metabolism (SEM) data. Futher theoretical background is provided within the first subsection of each later chapter: section 2.1 on *data modelling* theory; section 3.1 on *socioeconomic metabolism;* and section 4.1 on *unit-processes*.

1.2.1 Data, information and systems in industrial ecology

Two cliched yet pertinent terms are commonly used to characterise the present era: the "Anthropocene" and 'Information Age'. Earth's environmental systems are undergoing accelerating anthropogenic transformation (Steffen et al., 2015) at the very same time as the information describing these systems becomes copiously available and distributed (Castells, 1996). Industrial ecology is at the intersection of these developments. It aims to understand the complex societal drivers of this environmental transformation in a quantitative way. To

satisfy this aim from a scientific perspective requires vast amounts of structured and reliable data and information[2].

Accounting methods have been developed to guide the systematic selection, measurement, processing and communication of this data. Some of the most commonly applied methods are lifecycle assessment (LCA), material flow analysis (MFA) and environmentally extended input-output analysis (EIOA). The case studies developed by following these methodologies involve the modelling of a system with a network structure. Structured data is an essential component for every applied case study which follows these methodologies.

An example of the components of these methods is shown in Figure 1.1 (from Müller et al., 2018). This displays a hierarchical pyramid of the various components of MFA case studies, with each level of the pyramid fully dependent on the levels below. Often, case study research is performed to contribute to scientifically informed governmental policy relating to production, consumption, trade and general economic and industrial policy. The policies are often normatively oriented toward addressing anthropogenic environmental problems. As visible in the pyramid, this high-level decision making has foundations in multiple lower levels. This research focuses on the deeper layers of the pyramid: the systems and data.

[2] When *data* and *information* are contrasted: data are obtained via observations, whilst information is obtained via analysis or interpretation. In common usage and for convenience, the terms are often used interchangeably

Figure 1.1: the position of data in relation to systems, models, and other elements – the MinFuture pyramid from Müller et al. (2018)

Raw measurements and observations are almost always aggregated in IE research, and these aggregations are what researchers refer to as *data*. This is necessary because of the size and/or complexity of the systems being described. *Unit-processes* are an example of this aggregation process, which are the focus of this research and described in Chapter 3. In order for this data to become useful information, it must be contextualised as part of a system-level analysis, with clearly delineated scope and boundaries. The location of each aggregated data point within the system should be specified, with minimal ambiguity (more on this in Chapter 2).

1.2.2 Open data

The data, information and system descriptions which are introduced in the previous sections concern issues which are of relevance to many people throughout society. Despite this, they are often not widely accessible to everyone. The scientific data released in journals and through other sources is often protected by intellectual property licenses. Even when the data is released, it may be in formats or datamodels which are difficult to understand, or read and process with common software (see Chapter 2). There is a transdisciplinary movement acting across science towards improving this situation, using the term *open data*, which is one component of a wider movement toward *open science*.

The field of IE has made some progress in this direction (Davis et al., 2010; Bollinger et al., 2015; Pauliuk et al., 2015; Hertwich et al., 2018; Kuczenski, 2018). A Data Transparency Task Force' (DTTF) was setup by the *International Society for Industrial Ecology* (ISIE) to focus on this issue. They conducted a small survey, which identified strong support for this direction ("SI3. Survey results on data openness requirements" in Hertwich et al. [2018]). However, in general, the community is yet to comprehensively embrace the change in working practises required to evolve the open-data ecosystem which is envisioned in these publications. In order to facilitate these changes, Hertwich et al., (2018) states that "structured data from a wide spectrum of studies could be integrated into a common database so that researchers have the opportunity to query multiple relevant data sets at the same

time". The IEDC is an attempt at creating this common database. It is intended to act as a central part of the open-data ecosystem (Zuiderwijk et al., 2014) enabling research data sharing within IE.

To function effectively within this role, the specification and implementation of any community database should satisfy a series of design requirements. A community-sourced transdisciplinary set of principles could be used as a starting point for these requirements, as described here. The "FAIR Guiding Principles for scientific data management and stewardship" were developed to "improve the infrastructure supporting the reuse of scholarly data" (Wilkinson, 2016). FAIR stands for Findable, Accessible, Interoperable and Reusable. The recommendations were summarised and emphasised by the ISIE DTTF in Hertwich et al. (2018) as:

"(1) findable: indexing or archiving (meta)data with unique identifiers (e.g., digital object identifiers [DOIs]) at a searchable resource;

(2) accessible: (meta)data use an open standard for machine readability and are made permanently available.

(3) Interoperable: (meta)data use standard data vocabularies, in a formal, open, and broadly applicable language, and include references to connected data.

(4) reusable: (meta)data are defined with relevant attributes for"

What is unclear, is the extent to which alternative infrastructure designs are able to support these guidelines for the IE community specifically. Can the GDM and IEDC effectively facilitate the IE community to embrace *open data*?

1.3 Knowledge gap: representation of unit-process data in the GDM and IEDC

The GDM and IEDC are designed primarily for the representation and storage of data relating to SEM. To evaluate their utility for this purpose, it is important to test their capabilities and

design. Some specific claims have been made in this regard, listed and elaborated below. The statements and elaborations are then summarised as a specific knowledge gap. Quotations are from Pauliuk et al. (forthcoming) except where otherwise noted.

1. "Complex datasets, such as an EcoSpold unit-process descriptions or MRIO tables, can be broken down in the basic data types[3] of the IEDC"

This suggests that the 'basic data types' of the IEDC are at a lower-level than the source data, which they aim to represent.

2. "Metadata specification varies substantially across fields … future work needs to identify how these rich but differently formatted metadata can be brought into a general structure."

This statement suggests that the metadata which describes the source datasets must also be represented by the GDM. This metadata is in addition to the *data* or *datasets* from item 1. It is unknown what should be defined as metadata for particular sources.

3. "What we did for now is to demonstrate that the different available data formats fit into a common format. The conversion we did…is coarse… we only show that it is possible in principle. What is unresolved yet is to take an entire LCA database…, identify the different data sets contained in there, and show how they are translated in to the common data model." (Pauliuk, personal communication, 2018).

This final quotation helps to clarify the context and motivation of this thesis research. It shows the current state of affairs at the beginning of this research effort. LCA databases contain primarily unit-process data, which is described at depth in Chapter 3.

The research gap is summarised as:

[3] *Data types* here refers to an IEDC concept (described in 3.2.3), not the typical meaning of a data type such as Integer or Boolean (see section 2.1.1).

The extent to which the GDM and IEDC are capable of effectively representing unit-process data is unknown. The designers of these systems believe that this is possible, but it has not yet been demonstrated in theory or practise.

1.4 Research specification

The research specification is divided into three sections. The *objective* section explains what this thesis intends to test and demonstrate, plus the broader significance of this work. The research questions which guide the thesis research are then presented. The final subsection describes which research is included as within and beyond the scope of this work.

1.4.1 Objective: contributing to state-of-the-art data modelling and management in IE

The broad purpose of this work is to contribute to the effectiveness of IE research, through the synthesis and communication of suitable and accessible knowledge and recommendations on the topic of data modelling and management. Improved practises in these areas will lead to more efficient research, higher quality science, and a deeper understanding of systems under study across the field. This could benefit society via improved scientifically-informed decision making, on issues at the intersection of society and the environment.

More specifically, this research aims to evaluate the GDM and IEDC through an assessment of their ability to effectively represent complex unit-process datasets, such as those used in background databases for LCA. Identified shortcomings and recommendations may then be translated into iterative improvements or redesigns of their datamodels. In the words of Tupper (2011): “models allow us to map existing data to the structures and evaluate their appropriateness. If the model structure does not support the available data, then knowledge is gained without excessive expenditure of money or time, and without impact on applications”. The GDM and IEDC are not yet widely used, and have not had significant money or time expended on them by actors beyond the original developers. Hence this research is presented at an opportune moment.

1.4.2 Research questions: can the GDM and/or IEDC effectively represent unit-process data?

Is it possible to effectively represent unit-process data in the 'general data model for SEM' and/or the IEDC and what are the challenges and limitations?

This is broken down into the following sub-questions.

1. *Which datamodels are used in the IEDC and GDM for SEM, and what is their relationship?*
2. *Which datamodel is archetypal for unit-processes, and which data following this model can be used to evaluate the IEDC?*
3. *How can the selected source datasets be mapped and transformed into the datamodel of the IEDC?*

Two words in the main research question require explanation: "effectively" and "represent". *Represent* implies that the data in the IEDC is able "to serve as the counterpart or image of" the original data source (MerriamWebster, 2019b). The term *effective* implies that the unit-process representation should be "ready for service or action" and "producing a decided, decisive, or desired effect" (MerriamWebster, 2019a). Hence if the GDM/IEDC datamodel can "effectively represent" unit-processes, then the data stored in their structure is ready to satisfy the needs of its users. The expected use-cases are discussed in Chapter 4.

1.4.3 Scope: analysis of the GDM and IEDC, using one source dataset

As will be argued in section 6.2.1, the GDM theoretical descriptions are difficult to assess. The IEDC database design and the data it contains are therefore the main targets of assessment within the scope of this research. A single unit-process source dataset is selected to transform into the datamodel of the IEDC. The selection of this source is justified, and its contents are analysed in some depth. The transformation of this source dataset into the destination datamodel is a process that is used to generate knowledge about the IEDC. In order to understand the value of this knowledge, a theoretical basis from the art and science of data modelling is presented in Chapter 2.

A number of topics that are important for any community data infrastructure (such as the GDM/IEDC) are beyond the scope of this research. The sociotechnical systems context within which they exist is not emphasized, which includes avoiding legal and licensing issues wherever possible. Other areas beyond scope relate to the database development life cycle: data submission workflow process design; version controlling of data or code; and database user and permissions management. Finally, practical redesigns of any analysed systems are not attempted, although recommendations are offered toward this end.

1.5 Report structure

This thesis report is divided into eight chapters, as visually depicted in Figure 1.2. After this introductory chapter, Chapter 2 introduces essential basic knowledge required to understand the research: data modelling and transformation theory. Chapter 2 concludes with the method and requirements which provide context for the subsequent three chapters of analysis. Chapter 3 presents a detailed original description of the GDM and IEDC database which are the focus of the work. Chapter 4 focuses on the unit-process data which is used as a source dataset for testing the IEDC. Chapter 5 then describes the conceptual and detailed mapping of the source data from Chapter 4, into the destination database of Chapter 3. It also describes how these mappings are implemented in practise. Chapter 6 presents a discussion of the results and discoveries of the research, and places them in context. The research questions are answered formally in the conclusions of Chapter 7. The final Outlook chapter suggests an alternative approach to community data infrastructure, distinct from the one assessed in this research.

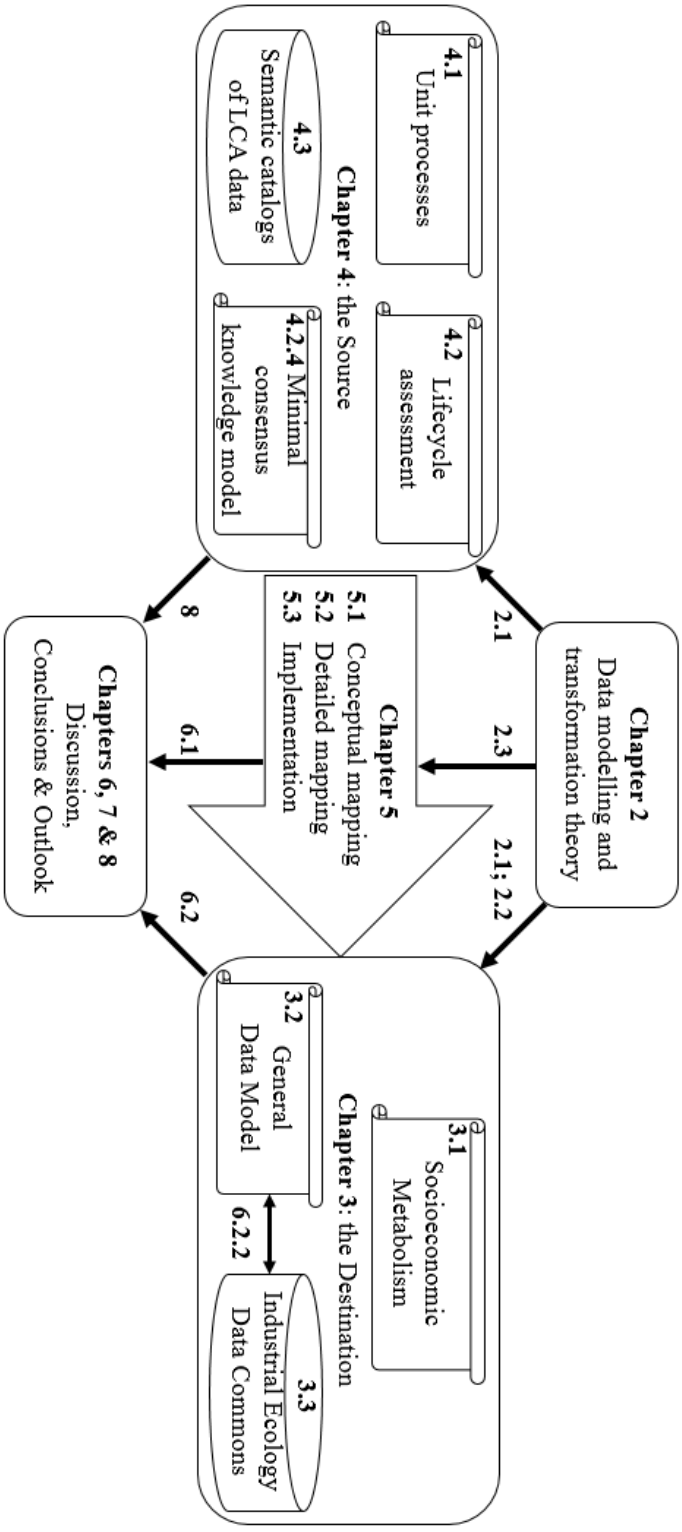

Figure 1.2: structure of this thesis report

Chapter 2 Data modelling and transformation theory and method

This chapter introduces the necessary background knowledge for an IE audience to understand the method and results of the research. Section 2.1 provides a broad overview of data modelling, including its meaning, purpose, and the *constructs* which are used to build up datamodels. Section 2.2 attempts to identify markers of datamodel *quality*, which are used as the basis for an assessment of the GDM and IEDC in Chapter 3. Section 2.3 describes how data which conforms to one datamodel can be converted into data which conforms to a completely different model. This conversion process has a number of pre-requisite requirements, that provide the basis for the gathering and development of knowledge in Chapters 3 and 4.

2.1 Background: data modelling definitions and purpose

Datamodels may be confused with a number of related concepts, such as *data types, data structures, data/file formats* and *databases*. Definitions are offered for these concepts, to ensure they are clearly delineated from one another and to properly position this work. The purpose of datamodels is then explored: what do they do and why is that important? This section concludes by describing the basic building blocks of datamodels: their *constructs.*

2.1.1 Definitions and delineation

Definitions for **datamodels** vary from the abstract to the applied. Tupper (2011, Ch.10) describes them as “a symbolic or abstracted representation of something real or imagined”. Tsichritzis & Lochovsky (1982, p. 10) argue that datamodels “define the rules according to which data are structured”. Oppel (2010) offers two definitions which describe the way models are frequently used in modern practise: they “describe how the data in an information

system is represented and accessed" (p. 4) and are "abstractions of existing or proposed databases" (p. 4). These definitions have no mutually exclusive elements, and are all considered valid for this research. One usage of *datamodel* which is not intended for this research, is used to describe a particular way of representing data, such as a: "flat model" (two-dimensional array of elements); "relational model" (tables); or "a graph model" (documents with references) (Simsion & Witt, 2004, p. 31).

Data modelling is a process undertaken by individuals or groups to create a data model. The fundamental task of this process is the discovery, encoding, and communication of *structures*, *rules* and *tendencies* which accurately describe the domain of knowledge which is modelled. This process may be improved by the inclusion of relevant stakeholders such as subject-matter-experts, users of the model being defined, and data modelling professionals.

Data structures are the "way in which data are stored for efficient search and retrieval" (NIST, 2004). Examples include *queues*, *trees*, *linked lists*, and *dictionaries*. They are made up of primitive **data types**, such as *integer, character*, or *boolean*. When a data type is specified for a variable or function, the permissible values it may assume are constrained. Data structures and types are researched and developed by computer scientists. Generally, industrial ecologists and others may simply use them when required, without concern for their implementation.

The term **data format** is ambiguous. Perhaps the most common meaning, and that which is most relevant to this research, is **file format**. When a datamodel is used in practise, the resultant data instances must be saved in some particular file format, for storage in computer files. The software which processes these files expects the information content in a pre-defined structure. A *file format specification* defines this structure, based upon a set of requirements, which depend on the intended use-cases. If the file formats are also intended for human-readability (rather than solely computer-readability), the format is usually a type of text file, made up of lines of character strings. IE research typically uses a limited range of file formats which are custom-designed for specific research areas. Examples include the XML-based ecoSpold2 file format which was designed for ecoinvent (Meinshausen et al., 2016); the '.LCA' files used by the software CMLCA; and the '.zmfa' format of the STAN software, for material flow analysis (MFA).

Databases are collections of organized data which are organised for search and retrieval (Date, 2006). Datamodels can be transformed into database structural code to provide useful functionality to users. The datamodel used in a database should be clearly visible in the names of the tables, and the relationships defined between them. Within each table, the entity attributes are visible in the columns, which are constrained by *data types*, as described above. Databases are stored on disk in computer files described by a data/file format; but users (for instance scientists and their software/models) do not typically interact with databases through these files. Rather, users and applications use a **database management system** (DBMS). These software implementations of databases enable the performance of standard operations on data and metadata, such as *create, read, update* and *delete*, and a wide range of other functionality. Popular DBMS examples include SQL Server, Oracle, and MySQL.

Finally, **datamodel repositories** are defined by Tupper (2011, p. 196) as "a storage bank of datamodels". They are the location where metadata and formal descriptions of an organisation's datamodels are stored, harmonised and maintained. The constructs (section 2.1.3) of each model are described, such that those shared between multiple models are easily identified. Each datamodel found within the model repository can be instantiated as a *separate database or schema*, which would then hold the actual data which fits the specific model. Many organisations do not maintain formal datamodel repositories, so additional conceptual work is required when they harmonise and integrate data.

2.1.2 Purpose: communication with high specificity

Datamodels are used for the accurate and specific representation and communication of information. Prior to the modelling of data, a domain of knowledge may be described in *natural language* as part of a narrative, or in another semi-structured form. In the words of Tupper (2011), "the modelling process reduces undisciplined nonmathematical narrative to algebraic regularity, formal symbols and statements". People and computers can then logically process these symbols and statements in a consistent way with a comparatively low degree of ambiguity.

Data modelling can help to bring conceptual disagreements into the open. Use of a common model increases the likelihood of users to reach a common level of understanding of the

knowledge which is modelled. Datamodels make explicit the relationships between the basic entities which are defined within the ontology of a particular field. If the conceptualisation of one group does not match that of another, their datamodels will necessarily differ. Oppel (2010, p. 13) states that "datamodels are a vehicle for pointing out common entities and attributes across an organization" [4]. The process of data modelling effort also requires the naming of things, and hence the development of commonly agreed naming standards, improving communication through standardisation.

Datamodels can be described using "words, pictures, or any combination of media" (Tupper, 2011). However, for clear communication it is hard to argue against Oppel (2010), who states that "diagrams are the very best medium". Particularly when these visual diagrams are enforced with supporting information, such as accompanying text to clarify potential ambiguities. Each visualisation represents a datamodel and communicates their main concepts, enabling an evaluation of the completeness and accuracy of the model in reference to the domain it attempts to represent. Popular diagrams are the entity-relationship diagram (Chen, 1976) and the table-based diagrams which are based on the relational database model (Codd, 1990). Examples of these visual techniques are shown later, in Figure 4.5 and Figure 3.2 respectively.

Once a datamodel is in widespread usage, actors can easily share data which fits these models. The models form the basis of a shared conceptualisation, which may be stable over a period of time. *Instances* of data following the model can be shared across space and time, maintaining a consistency of interpretation. The model itself fades into the background, and people are instead able to focus on the meaning of the data that are represented in these models. Updates to the models may become necessary. In these cases, it is helpful for users if backwards-compatibility is maintained, allowing them to keep using their existing data instances within the new model. However sometimes this is not possible.

[4] Much of the data-modelling literature focuses on the *organization* or *business* context, where most professionals working in this area are employed. Nonetheless, the ideas and theories are readily transferrable to the scientific domain of this research.

2.1.3 Constructs: entities, attributes and relationships

A number of constructs are used for almost all datamodels: *entities, relationships* and *attributes*, as shown in Figure 2.1. Alternative names may sometimes be used for these constructs, depending on the modelling paradigm or language.

An **entity** is "a thing or object of significance, whether real or imagined, about which some information needs to be known or held" (Tupper, 2011). For naming entities, a common convention is to use a noun in singular form. It should be "as meaningful as possible to reflect the information it is maintaining".

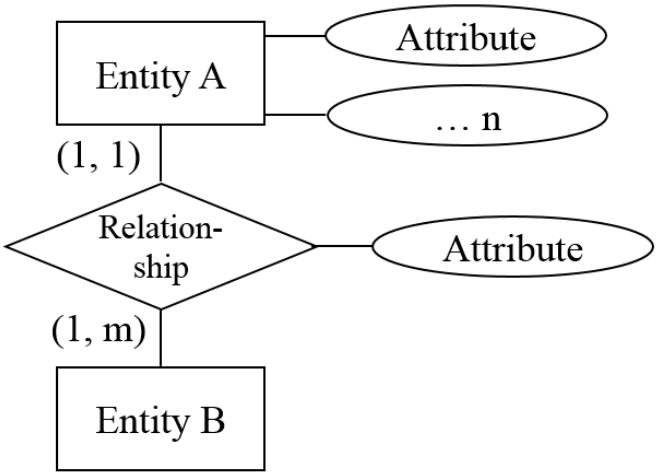

Figure 2.1: data model constructs: entities, attributes and relationships

An **attribute** (or property) **"**holds a particular piece of information about an entity" (Silverston & Agnew, 2009, p. 18). Thus, entities are usually described or specified by a number of attributes, which may be *required*, or *optional*. If an attribute is required, this is considered a *constraint* of the data model.

A **relationship** "defines how two entities are associated with each other" (Silverston & Agnew, 2009, p. 18). It is important to understand the nature of the relationship between entities, to help in interpretation of the significance and meaning of the data. There are many types of relationship, important types for IE are: *hierarchies*, *aggregations*, and *recursive relationships*. Recursive or peer-to-peer relationships are "semantic connections between many objects of the same class" (Silverston & Agnew, 2009, p. 134). Each relationship has a cardinality, which specifies the required number of entities on each side of the relationship. For instance, the biological *child-to-parent* relationship in most animals has the cardinality *one-to-two,* for any specific child.

When modelling a particular domain, the choice of which construct should be used to represent each real-world concept may not be obvious. Legitimate alternative perspectives may be held by stakeholders in the modelling process and the Verelst (2004) refers to the alternative available decisions as *construct variability*. As such, perfect data models do not exist. But some are more effective than others for their intended tasks, and are said to be of *better quality*.

2.2 Characteristics of datamodel quality

This research aims to provide commentary and recommendations on the use of particular datamodels within the IE research community. To do this, it is necessary to identify characteristics which relate to the broadly-defined *quality* of a data model. Oppel (2010, p. 14-16) provides some suggestions: validity of the representation; ease of understanding; appropriate enforcement of business rules; and the level of flexibility and adaptability. Each characteristic is explored using the example of modelling a 'unit-process', which is a key component of this research and described in Chapter 4.

2.2.1 Representation validity

It may be clear that a data-model is of low quality, because it cannot be used to describe the phenomena being modelled. In the words of Tupper (2011) "the essence of a model lies in its efficient representation of the business problem area". For instance: unit-processes may have many inputs and outputs. Any datamodel which permits only one input or output per unit-process is clearly invalid, and of no practical use in this domain.

Tupper further specifies that efficient representation is "achieved by eliminating unnecessary detail" (ibid.). If a datamodel includes superfluous detail, this does not necessarily mean it is invalid. For instance, if a unit-process requires that the '*invention date*' of the technology of the process is specified, then this would be an extra burden placed on all data gathered, and of limited use to most users of the data. The meaning of this attribute may also be unclear, leading to inaccurate interpretations.

2.2.2 Ease of understanding

As described previously, much of the value of a datamodel comes from its ability to aid communication between users of the model. Therefore, it is important that the model can be understood by the intended users. Models that require significant mental effort to understand them are unlikely to achieve widespread usage. If they do, there is a greater likelihood that users will apply the model in different ways. This creates incompatible conceptualisations, data integrity violations, and general uncertainty. Cooperation and collaboration under these circumstances would be challenging.

2.2.3 Promotion of data reusability and integration

Within any data model, there are particular entities and attributes that could serve many purposes. Data reusability is realised when these are identified and isolated. For instance, the international ISO 3166 provides a standard for representing all countries and their subdivisions as shortened codes. This list of codes is reused in the datamodels in many systems in many contexts around the world. This saves time for the developers, as they need not recreate the list when they model a *country* entity. It also increases inter-operability between systems, as it is only necessary to transfer the representative country code in order to communicate meaning with fairly low ambiguity.

Data integration also relates to the circumstances when data can and should be used from multiple different sources. When developing a data model, an opportunity is available to integrate the data into common structures. The process of analysing the original data sets provides insight into the structures of each source. Data can be integrated more easily if it follows explicit and tightly defined rules specifying which data is permissible in a set. This includes the data type, precision, and whether values must come from a pre-defined list.

2.2.4 Enforcement of domain rules

From a scientific perspective and in the context of datamodels, domain rules are constraints and assertions that are derived from the established body of knowledge. They aid the clear and accurate description of objects and concepts which are modelled. For instance, a process

may have a thermodynamic efficiency. This efficiency can never be greater than 100%, and may be further restricted depending on the type of process under consideration. In this case, thermodynamic equations may determine the values which are possible for particular attributes of the process entity. Violation of this constraint is theoretically impossible, so any study using a datamodel which permits violations is open to the possibility of inaccurate results and conclusions.

Domain rules may be enforced at a number of levels, which each have advantages and disadvantages: at the data level, the software level, or human level. The simplest approach from the perspective of the software developers and data modellers is also the least reliable: this involves trusting the users of the model to only enter data which is valid (and hence not in violation of the domain rules). The constraints can also be applied at the software level. Here, the datamodel allows invalid values, but the software and algorithms running on the data are able to identify these violations at some point in the processing. This implies the requirement for extensive validation and testing on the software developers and quality assurance experts. Finally, the definition of the datamodel itself may enforce particular domain rules, making them harder to circumvent. Common constraints include the *cardinality* of a relationship, the *uniqueness* of value, the *permissible range* of a value, and whether a value is *required* or optional. *Formulas* may also be used, such as using the mass-balance principle to derive a rule for unit-process datamodels: the sum of mass flowing into a process is equal to the sum of mass flowing out.

The ideal rules to enforce at the level of the datamodel and database are those that are unlikely to be changed. If the rules do change, this may compromise the stability of the model. Rules that change occasionally can be handled by software. And rules which change all the time are not really rules at all. These cases usually represent the type of ambiguity and patterns that humans are typically able to deal with more effectively than computers. To conclude, the decision of which domain rules to enforce within the datamodel has a significant impact at subsequent stages, when others attempt to build software to interact with the datamodel, and people begin to use the data and software. These choices cannot be avoided when developing a datamodel.

2.2.5 Specialisation vs generalisation

An important trade-off inherent in the data modelling process is that of specialisation versus generalisation. A specialised model implies high suitability for a very specific purpose or function. Whereas more generalised models are likely to be somewhat suitable for modelling a wider variety of phenomena. An example relating to unit-processes is shown in Figure 2.2.

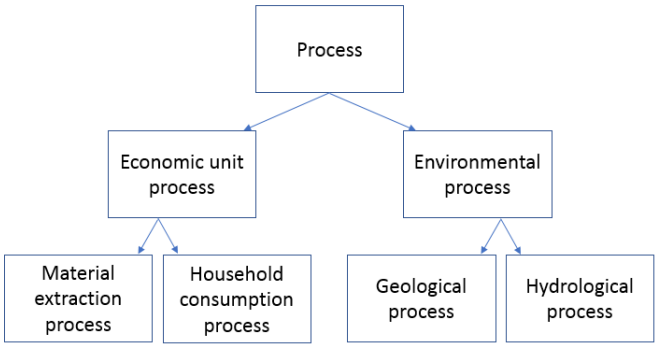

Figure 2.2: a process-focused example of a super-class and subclasses.

This research is focused on modelling *economic unit-processes*, which are a subset of the broader category *process*. This category also includes *environmental processes* and many other types. Both sub-categories could be broken down further, as shown. A datamodel which effectively captures salient features at one vertical level is unlikely to be fully suitable for representing another. Concepts at the same horizontal level are related but distinct; a datamodel which accurately represents one concept will not be appropriate for its neighbours. Thus, a generalised datamodel is likely to require adaptation and specification, in order to be functionally useful at multiple levels of abstraction.

2.3 Method: mapping datamodels, from source to destination

Building upon the background knowledge presented in this chapter, the focus now shifts to the method used in this thesis, to answer the research questions and generate new knowledge: *data mapping*. Haq (2016) defines data mapping as "the process of creating a link between two distinct datamodels". The two datamodels are referred to throughout this research as the *source* and the *destination* (or target). The source is the unit-process data described in

Chapter 4. And the destination is the 'general datamodel of socioeconomic metabolism'. However as shown in Chapter 3, the IEDC prototype must be used as a surrogate for the GDM. This is because the GDM is not defined with sufficiently specific detail within the currently available information to act as a target datamodel for the mappings.

This section describes two different types of data mapping: the high-level *conceptual* data mapping, and the more precise *detailed* data mapping. There are pre-requisites required before these mappings can begin, which are satisfied through chapters 3 and 4. Once the prerequisites are handled and the mappings are complete, they can be practically implemented or programmed, as algorithms. In the language of databases, this is called an extract-transform-load (ETL) process. ETL "involves dealing with the specificities of information at very low levels of granularity including transformation rules at the attribute level" (Luján-Mora et al., 2004). As such, it relies upon the detailed mappings more than the conceptual mappings. Execution of an ETL procedure results in the actual population of the destination database.

2.3.1 Conceptual and detailed data mappings

Datamodels can be viewed from various levels of abstraction. As argued by Silverston & Agnew (2009, p. 13): "in the data modelling industry, there does not appear to be a common, single, universal understanding regarding the purpose and definitions of conceptual models, business model, logical datamodels, and physical datamodels". This research requires a mapping across heterogeneous datamodels and formats, from a file-based json-ld source using an entity-relationship model (section 4.3.1) to a MySQL relational database target (see section 3.3.1). For this reason, the two levels of abstraction which are considered relevant in this research are referred to as 'conceptual' and 'detailed'. *Conceptual* is chosen in preference to *logical*, to emphasise the organisational semantics, without reference to the actual technologies in use. Related reasoning applies to the choice of *Detailed* over *physical*. The detailed mapping is concerned with the actual attributes and columns of data in the source and destination models, but without consideration for how they are physically stored on disk.

The conceptual model of the source database is described using entity-relationship model, which consists of entities, and the relationships between them (section 2.1.3). The attributes

of these entities are beyond the high-level conceptual model. The relational model of the destination is conceptually organised using tables, which are related via keys. In this case, columns are equivalent to attributes. Hence the conceptual mapping will consist of a mapping from an entity-relationship diagram without attributes, to a table-based diagram without columns.

The detailed data mapping follows from the conceptual mapping, with additional detail focused on specific attributes, properties, records, rows and columns. Detailed transformation rules are required here to clearly convey how to process the source data into to make it suitable for the destination model (Haq, 2016). These can be written in formal mathematical or programming languages which can be copied directly into the ETL design, or as pseudo-code including if-else statements, for later conversion.

A key result of the detailed mapping is the *data mapping document,* which complements the diagrams of the conceptual mapping (Haq, 2016, ch. 9). This document contains a number of components, including: an overview list; a legend to explain colour schemes or other design choices; a table of actual data mappings, including many mandatory columns of information; the loading order dependencies; and sometimes a reference data list and change log (in the typical case of an iterative development process). This document is usually designed and completed in spreadsheet software. These detailed mappings are then the focus of the implementation for the ETL process (section 2.3.3). The data mapping document created by this research is visible in 0.

2.3.2 Prerequisites to data mapping

The data mapping effort described above cannot begin until certain pre-requisites are in place. The process of satisfying these pre-requisites prepares the data-mapper, forcing them to develop a detailed understanding of the source and destination datamodels. This reduces the likelihood that they will make conceptual errors, which would lead to significant time wastage in implementing an incorrect ETL process. In the worst case, the mistakes would not be noticed until the destination data is in widespread usage. The following list of prerequisites must be understood and documented (Haq, 2016, ch. 7):

1. The source conceptual data model: entities, relationships and descriptions
2. The source physical data model: the attributes, plus their constraints, data structures and data types
3. The destination conceptual data model: tables, relationships and descriptions
4. The destination physical data model: the columns, keys, data types and constraints
5. Knowledge of the unique identification of entities and instances in the source and destination
6. Knowledge of the cardinality of the relationships
7. All source data, or a representative production-quality subset. The data mapper will query this data and base the transformation rules on the results. If the data is not complete or representative, the rules will be wrong
8. A subject matter expert, whom can clarify uncertainties regarding the data

2.3.3 Implementation of data mappings: extract, transform, load

The detailed data mapping document can be used to implement actual procedures to load the IEDC with data. In the language of databases, this is known as an **ETL** process: Extract, Transform and Load (Figure 2.3). ETL processes are commonly used in the construction of data warehouses and database migrations. Their design "is driven by the semantics of the data sources and the constraints and requirements of the data warehouse application" (Skoutas & Simitsis, 2006). Building and executing an ETL system creates two main artefacts as an outcome: source code for the ETL algorithms (which enable loading of data into the IEDC), and the *actual* population of the IEDC data warehouse with unit-process data from the chosen source. The three steps of extract, transform and load, are described briefly below, followed by some statements regarding the simplified scale of ETL intended for this research.

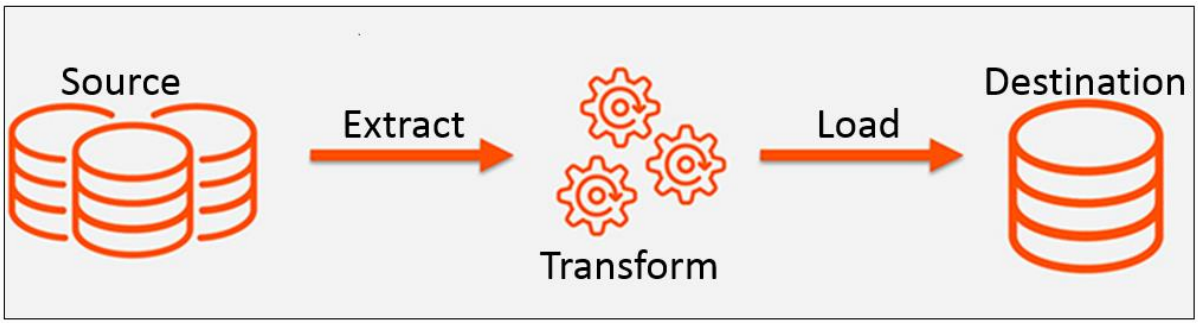

Figure 2.3: visualisation of an ETL process: source datasets undergo transformation for loading into a destination

To *extract* means “reading and understanding the source data and copying the data needed into the ETL system for further manipulation” (Kimball & Ross, 2013, p. 19). Once this data is available, there are many possible types of transformations possible, such as “cleansing the data (correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats), combining data from multiple sources, and deduplicating data. (ibid, p. 20). The ETL process concludes with the physical loading of data into the destination data structures. From here, the data can now be queried alongside other data sources that are represented in the same destination datamodel.

ETL processes can be incredibly intricate and complex. Kimball & Ross (2013, ch. 19) argue that an ETL system has 34 sub-systems, from data-profiling, and workflow-monitoring, to error-handling and metadata management. As this is scientific research, the practical implementation of the ETL is not emphasised in this report. The aim is to answer the research question via a proof-of-concept demonstration, to confirm whether the unit-process data can be represented in the destination data model. As such, the ETL is kept simple, and includes few of these purported sub-systems.

This concludes the section on mapping datamodels from source to destination, and the data modelling chapter. The requirements described here are gathered in the subsequent three chapters. For this research, the destination database for the mappings is the GDM and/or IEDC. These are the focus of the next chapter.

Chapter 3 The destination: a general data model for socioeconomic metabolism and the industrial ecology data commons

This chapter presents an analysis based on two sources which were briefly introduced in Chapter 1: a manuscript, and a database. The manuscript is "A General Data Model for Socioeconomic Metabolism and its Implementation in an Industrial Ecology Data Commons Prototype", by Pauliuk, Heeren, Hasan and Mueller (forthcoming, likely 2019). This includes a theoretical section describing a *general data model* (GDM), and a more practical second part describing the implementation of the GDM as the *industrial ecology data commons* (IEDC): an open database which is available now for download[5].

To assess the GDM and IEDC, it is important to understand their relationship. Pauliuk and colleagues (2019) state in reference to the GDM that: "the data model can be implemented in different ways, including spreadsheet-formatted data, relational databases, or array-shaped data in programming environments". The IEDC is then framed as "a relational database built on the general data model". From these two quotes, the following hypothesis is presented regarding their relationship:

The IEDC is a technology-specific implementation of the GDM, and other hypothetical implementations of the GDM should be interoperable with the IEDC.

To satisfy this *interoperability* requirement, the GDM must be described with enough explicit detail to enable data to be exchanged easily between alternative implementations. They would need to share a common conceptual data model, but only differ in the details of their implementation. The second section of this chapter suggests that the GDM is not yet

[5] Full source code for the project is available online (GitHub, 2018a). A prototype instance of the database is currently hosted on a public-facing webserver (Industrial Ecology Freiburg, 2018). As of December 2018, there is not yet evidence of usage beyond the developers.

described with the level of explicit detail required to enable this interoperability of implementations. For that reason, the IEDC is used as the only subject of detailed analysis in this research, and the exclusive destination for the data-mappings described in Chapter 5. This topic is returned to in the discussion section 6.2.

The 'GDM for SEM' uses the notion of *socioeconomic metabolism*. It is necessary to understand what is meant by SEM in order to understand what the GDM and IEDC are attempting to model, and hence their design requirements. As shown in Figure 3.1, this is the subject of the first section of the chapter.

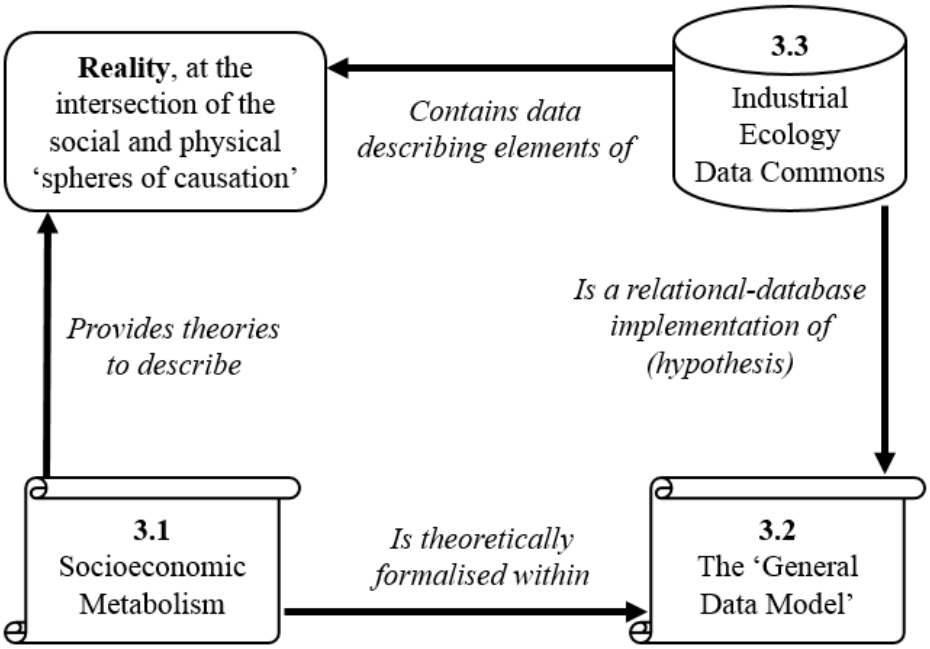

Figure 3.1: relationships between the topics of Chapter 3

3.1 Socioeconomic metabolism

The datamodels evaluated in this research aim to represent data for 'socioeconomic metabolism' (SEM). An introductory discussion of SEM is presented here. A summary of the relationship between IE and SEM is included in Appendix C. Society's metabolism is broadly recognised as the overlap between the social and biophysical spheres of causation, emphasizing the flows of energy and materials between *society* and *nature* (Fischer-kowalski & Weisz, 1999). The word 'metabolism' was first used at the societal scale as a *biological*

metaphor. Nowadays, the word is often used in a literal rather than metaphorical sense (Wassenaar, 2015). Although others disagree with this usage (Golubiewski, 2012).

Pauliuk & Hertwich (2015) argue that the 'biophysical basis of society' can be explained and researched by considering SEM together with a related concept, *society's biophysical structures*. They offer the definition*:* "Socioeconomic metabolism constitutes the self-reproduction and evolution of the biophysical structures of human society. It comprises those biophysical transformation processes, distribution processes, and flows, which are controlled by humans for their purposes. The biophysical structures of society ('in use stocks') and socioeconomic metabolism together form the biophysical basis of society." The *biophysical structures* are the static component of society, the objects which surround us and can be built-up as stocks (Haberl et al., 2004). These concepts relate only to the physical or tangible world, and hence exclude the ephemeral or intangible elements of society such as institutions, interpersonal relations, or software. When these intangible elements are also included, this may be referred to as the *technosphere*.

Motivations for the study of SEM are diverse, including exhaustion of resources, pollution, and the inefficiency with which the economy delivers services to satisfy the needs and desires of the population (Fischer-Kowalski & Hüttler, 1999). In a special edition of the Journal of Industrial Ecology dedicated to SEM, Schandl et al. (2015) argue that "*The growing impact of socioeconomic metabolism research on the sustainability policy domain can be attributed to the strength of its conceptual framework [which] allows for analysis of the linkages between traditionally fragmented research domains ... and the feasibility of establishing meaningful data"*. It is thus a multi-disciplinary field, interested in economy-environment intersection, and driven by data.

3.2 The 'general data model' of socioeconomic metabolism

The GDM of SEM is framed as a solution to the "lack of a generic structure for IE data". It is described solely by one information source: the manuscript which introduces it (Pauliuk et al., forthcoming). With regards to the GDM, the manuscript describes the following elements:

- the systems context of the data
- the relationship between *system dimensions* and *data aspects*, plus a table with examples
- a definition of the datamodel which aims for formality. This consists of six statements (labelled D1 to D6 in the source) including two equations.
- six pre-defined *data categories,* and suggested names for some *data types* which are stated to belong in those categories
- a "proposal for the required and optional aspects of [six] central data types" (one per *category*).

This section summaries and analyses these components of the GDM theory. It begins with some quotations to highlight the concepts and objects which the datamodel is intended to describe, as well as the acknowledged limitations of the model. The approach used in the GDM for representation of specific *facts* (or observations / data points) is then described. Followed by an overview of the GDM concept *data types*, which are intended to specify "the required and optional *aspects"* for each type of fact which can be structured within the model. The *data types* are grouped into *categories*, which are critically assessed in the section 3.2.3. These critiques flow into a section which highlights how the GDM cannot effectively be used as a destination for a conceptual data mapping, due to the lack of clarity in the existing descriptions. This chapter continues with an analysis of the IEDC, and these points are returned to later in the Discussion.

3.2.1 What does the GDM describe?

The supporting information of Pauliuk et al. (forthcoming) contrasts the GDM with the EcoSpold data format (see Appendix K). The GDM is stated to be "more comprehensive", representing additional data types, including: "product lifetimes, breakdown of products into components and materials, process capacities, population, economic statistics, trade flows, or economy-wide and sector-specific material flow accounts". These and other concepts constitute what the authors refer to as "the objects and events in the industrial system, or socioeconomic metabolism". The authors suggest that "quantitative information on processes, stocks, flows, etc… has three components", which are all handled within the proposed data model:

1) "The actual numerical information, including unit and uncertainty

2) The information needed to locate the data in the systems context, i.e., the link between data and the system dimensions (process, time, region, material, …)
3) The metadata, including provenance, source document, author and version information, and license"

The second item is particularly emphasised, as it is often insufficiently documented in existing IE research and "specific to systems science". Descriptions and analysis of each of these components as implemented in the IEDC are described in sections 3.3.5, 3.2.2 and 3.3.3 respectively. Two important GDM concepts related to the first and second of these items are *data types* and *data categories*, which are described in the next sub-section.

Only one limitation or shortcoming of the GDM or IEDC is noted in the paper (ibid.): "the structure presented here cannot efficiently accommodate high-resolution data…that describe a continuum, such as high-resolution time series or satellite images". As no other limitations are acknowledged, the paper appears to position the GDM and IEDC as fairly comprehensive solutions to the "data integration and exchange problem" which they seek to address. This thesis research identifies a number of additional shortcomings and limitations, which are highlighted in this chapter and reiterated where relevant in the discussion and conclusion chapters.

3.2.2 Representation of 'facts' as tuples

Each data item within the GDM represents a specific fact which describes part of the system. The system-location of this *fact* is described by a tuple of aspects, with a numerical value for quantification of this part of the system. Each element within the tuple is an *aspect* to be defined. The *data type* (next section) of the fact determines the *required* or *optional* aspects. The example given in Pauliuk et al. (forthcoming) is:

$$data\_item(aspect\ 1,\ aspect\ 2,\ aspect\ 3,\ \ldots) = value$$
$$trade\_flow(cars, Japan, USA, 2016, number\ of\ units) = 2540000\ units$$

The permitted values for each aspect are pre-determined. For instance, an aspect for *country* is selected from a list of all countries. These permitted values are the *classifications* for aspects. As described in Pauliuk et al. (2016), those classifications should ideally adhere to

the *H-MECE* properties: hierarchical, mutually-exclusive and collectively exhaustive. Following these rules, each tuple should be unique within a *datagroup* (see section 3.3.2).

3.2.3 Data types and categories

Two related concepts are described in the GDM manuscript for grouping and classifying the data which it stores: *types* and *categories*. *Data types* have a many-to-one relationship to *data categories*: each type must be a member of only one category, and each category can include any number of data types. This subsection critically assesses both of these concepts, beginning with data types.

The authors state that "all phenomena or properties described in the system are modelled as one of the defined data types". The paper states that "each *data type* has a specific *datamodel* that prescribes which aspects are required and optional". Clearly this is a different usage of the term *data type* compared to that which was introduced earlier. The unorthodox relationship between *data types* and *models* implied by this statement is critiqued in sections 6.2.1 and 6.2. Some examples are given for the *datamodels* of the *data types*, in a form the authors refer to as a "defining string". In the following example [brackets] denote aspects, (*) implies optional, and the connecting-words between the brackets are intended to specify the relationships between the aspects.

*"**Material composition of products (category 3)**: The material composition is an intensive object property describing the proportion of a material in a good/substance: [Material content] of [material] in [good/substance] of [age-cohort (*)] in [production region (*)] is [value/uncertainty] for [layer]. The layer can be mass but also volume."*

The use of natural-language predicates such as 'in', 'of' and 'is' without further specification introduces significant ambiguity into these *defining strings*. This ambiguity means it is difficult to write formal constraints and specifications for each *data type,* based only on the available examples. An alternative approach to these *defining strings* would be to use more explicit specification standards, such as UML class diagrams or entity-relationship models.

With regards to *data categories*, the paper (ibid.) attempts to group the data types into a higher level of organisation by focusing on the *properties* that each *type* describes. Properties of physical objects in sciences including SEM can be divided in two useful ways, into

(*intrinsic* or *extrinsic*) and (*intensive* or *extensive*) (Pauliuk et al., 2016). In the GDM, the intrinsic properties of objects should be "recorded along with the dimensional classification items" (ibid.). The remaining *extrinsic* properties are divided into six *data categories*. As shown in Table 3.1, they are first divided into the "two broad categories of description": *objects* and *processes*[6], which are also divided into *intensive* and *extensive*, creating a 2x2 matrix.

	Objects	**Processes**
Extensive	Extensive object properties… Flows (1) Stocks (2)	Extensive process properties (5)
Intensive	Intensive object properties (3)	Intensive process properties (4)
	General ratios (6)	

Table 3.1: overview of the six pre-determined GDM 'data categories'. Simplified version of *Table 2* in Pauliuk et al. (forthcoming)

Next, the *extensive object properties* are "divided further into stocks and flows", creating the fifth data category division[7]. The sixth and final data category is *general ratios*. These are described only briefly, as a derived category with examples: "data of categories 1-5 can be used to define ratios, like GDP per capita or per capita building stock". Examples for the *Intensive process properties* given in the source manuscript are also ratios, e.g. "Process operating costs per output". An explanation for the distinction between these *process* ratios,

[6] Contrast this with the #ontology paper, where the two broad categories were *objects* and *events,* with events divided into *processes* and *flows*. It is unclear why *processes* are now identified as a top-level category, but flows are not.

[7] This appears to claim that the fundamental modelling concepts of *stocks* and *flows* are now considered as "extensive object properties". This implies that the stock or flow (aggregation of objects) that an object belongs to, is a property of that object. This does not match typical usage of the term *property*, and seems illogical.

and the *general ratio* category is not provided. Based on the available information, it is unclear why these specific categories should be used, rather than another hypothetical approach.

3.2.4 Conceptual data mapping to the GDM is not viable in this research

To summarise, the definition of the GDM depends upon the *types* that constitute it. Examples are given for these *types*, but they are described in natural language rather than common standard datamodel description methods. The constructs used in the data types (e.g. material, region, age-cohort, yield, lifetime) are not formally defined. This means that people using the GDM require consistent implicit knowledge in order to use these constructs in the same way. And the cardinality of the relationships between these constructs is not specified. The types can be grouped into *categories*, but the rationale behind this additional layer of grouping is unclear.

For these reasons, it does not make sense to map the unit-process data source the GDM itself, as there is a barrier to understanding which cannot be surpassed using the limited descriptive information available. In order to perform the desired conceptual data mappings for this research it is necessary to instead reference the technology-specific implementation in the IEDC; an issue which is discussed further in section 6.2.1. The IEDC tangibly exists as a relational database, and therefore is explicitly described in data definition language (DDL). This will be the focus of the analysis and mappings for the remainder of the research.

3.3 The industrial ecology data commons

The goal of this section is to provide a basis for the conceptual and detailed mappings, by clearly detailing the structure, design decisions, and actual data contents of the IEDC via original research and analysis. The *Characteristics of datamodel quality* (section 2.2) provide context and guidance for this analysis.

The IEDC is an implementation of the GDM within the paradigm of relational data modelling (Codd, 1990). The relational model is implemented using MySQL, an open-source database

management system (DBMS) which is available in free and proprietary versions. The IEDC uses the free version. The authors state that they "chose a relational database model… as it is well established, easy to set up, and therefore suitable for a prototype". More traditional arguments for selecting a relational model include: referential integrity; performance (storage, CRUD operations); and confidence in the validity of a data model, enabling a high-likelihood for the pre-defined schema to remain stable. An alternative to using relational databases is explored in the Outlook (Chapter 8).

The design of the IEDC database can described in data definition language (DDL) which was implemented as part of the MySQL DBMS specification. Using this DDL code, instances of the database can be installed on ay MySQL installation. The first (and only currently known existing) internet-accessible version of the IEDC is hosted by the IndEcol group at Freiburg university. The process of remotely connecting to this Freiburg IEDC instance has been documented for public usage as part of this this research. Instructions for this are included in Appendix D. Instructions for creating an IEDC instance fully populated with the Freiburg datagroups and datasets are also included in Appendix E. By using this Freiburg database connection or instructions for creating a local clone, the analysis described in this chapter can be easily reproduced, confirmed or refuted.

3.3.1 Database structure

A relational database contains multiple tables which usually each store data about one specific entity. The relationships between tables determine the structure of the database. Relationships are defined by referencing the *primary key* of one table using a *foreign key* of another. Each relationship has a *cardinality* between the two tables, such as one-to-many, many-to-many, or one-to-zero-or-one.

The individual tables of the IEDC are described in the supporting information of Pauliuk et al. (forthcoming). As such, only descriptions that are necessary for the goals of this research are included here. A SQL select statement which can be executed to obtain formal descriptions of all relevant columns within the tables is included as Appendix F. The relationships between tables are not clearly described in the original work, and are hence emphasised here.

The schema of the IEDC is shown in Figure 3.2, which was created using MySQL Workbench model diagramming software. The cardinality of each inter-table relationship is indicated visually. The 'crow's feet' connections indicate that a table is the "many" side of a relationship, whereas the 'equals sign' connections indicate the "one" end. Table-pairs with multiple relationships defined between them are described in 3.3.4, and not shown in this diagram.

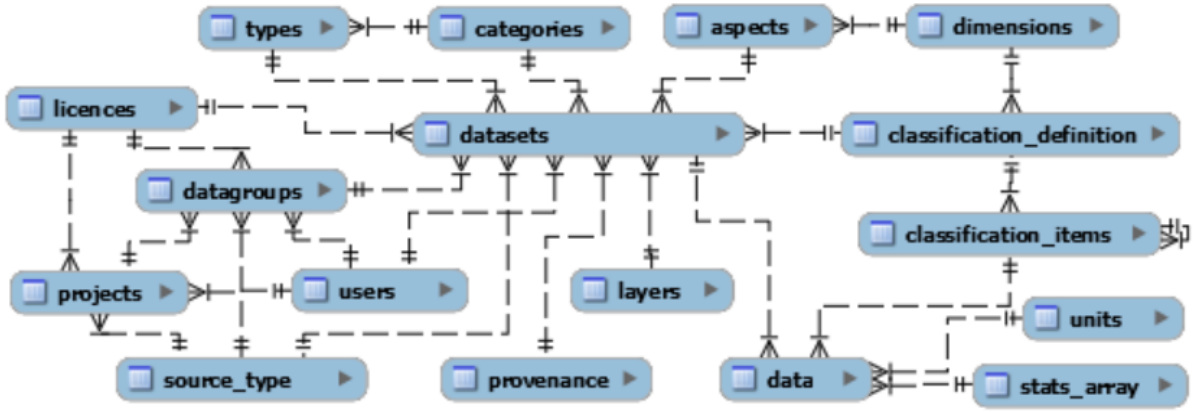

Figure 3.2: the full IEDC database schema

From this view of the schema, some observations can be made, which are relevant for the later mappings:

1. *Datasets* includes foreign keys to 10 other tables, and is referenced itself only by the *data* table
2. The *units* and *stats_array* tables are only related to the data table, implying that numerical or quantitative values are stored there
3. There is a group of highly-interconnected tables: *licenses*, *projects*, *datagroups*, *users* and *source_type*
4. *Datasets* each include a *type* and a *category*, and *types* are also related to *categories*
5. The only self-referencing table is *classification items*
6. The tables are named in a mixture of both singular and plural form

3.3.2 Primary storage hierarchy: project, datagroup, dataset, data

As shown in Figure 3.3, the hierarchical levels of *projects*, *datagroups* and *datasets* are used collectively to group observations of numerical facts or observations, which are stored in the *data* table. A collection of *data* records together constitutes a *dataset*. Each *dataset* has a *data type* which theoretically imposes the requirements on which aspects are compulsory and optional for each *data record* within that *dataset (*described in section 3.2.3). The datasets are grouped into *datagroups*, which can themselves be grouped into the final and highest level of grouping: *project*.

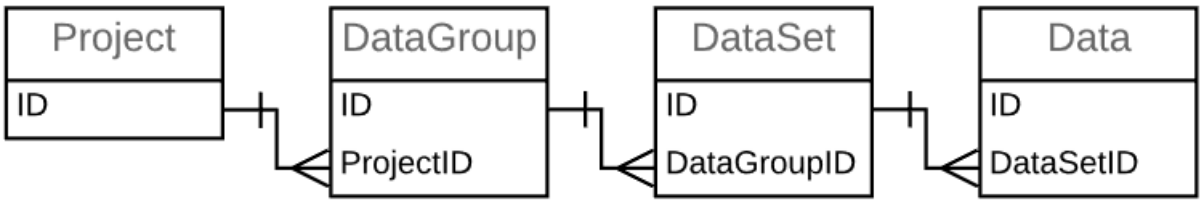

Figure 3.3: the hierarchical data groupings of the IEDC, with one-to-many relationships from left to right

3.3.3 Auxiliary metadata tables

The remaining tables briefly described here are: *licences*, *provenance*, *source_types*, and *users*. Complementary definitions can be found in the supporting information of Pauliuk et al. (forthcoming). *Licences* allows for tracking the licences under which each data source is made available. The *provenance* lookup table provides a list of possible methods via which the data group was constructed, for instance via scenarios[8], surveys, industry data, or expert judgement[9]. *Source_type* describes whether a dataset is from a publicly available or proprietary source[10]. *Users* is for management of the people and applications that load data into the IEDC and review the data as part of a workflow. These tables are of limited interest

[8] Note that 'scenario' is also listed as a 'system dimension', facilitating possible data integrity conflicts
[9] The complicated and sequential nature of provenance metadata is not well supported by this. E.g. see QUDT PROV data schema used for satisfying similar requirements https://www.w3.org/TR/prov-primer/
[10] Appears to have some overlap with the functionality provided by the aforementioned *licences* table, though they are not related via foreign keys in the model

to the mapping-focus of the research, and are not a core part of the data model. Omitting these from Figure 3.2 results in the simplified diagram shown below in Figure 3.4.

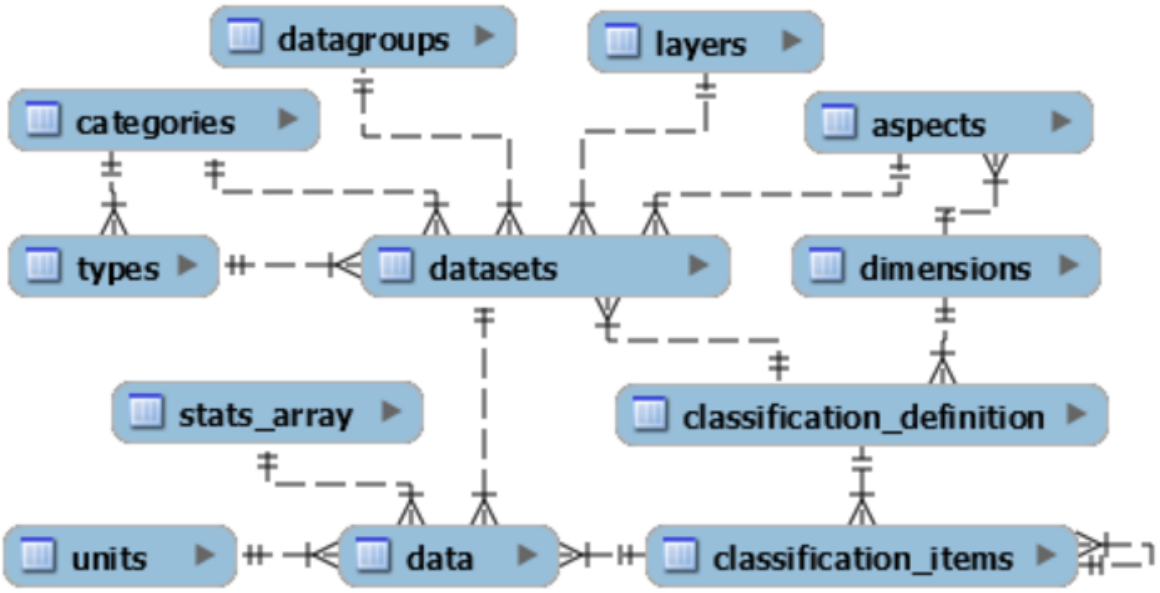

Figure 3.4: simplified IEDC database schema, following the removal of auxiliary tables

3.3.4 System location specification

The system location information for each quantitative value is specified using a combination of tables for *aspects* and *classifications*. As seen in Figure 3.4, the *aspects* and *classification_definition* tables each have a many-to-one relationship defined to *dimensions*. It is not clear precisely how that data is intended to be stored within these, based on their table or column names, documentation, or the data pre-stored in these tables. For instance, Appendix G shows how the records in *aspects* and *dimensions* are related and highlights some unclear issues. The SQL query to generate the results is also included.

The actual storage of records in the *data* and *datasets* tables do not relate directly to the dimensions table. This means this problem can be sidestepped for the remaining analysis. However, the aspects and classification tables are important for the upcoming mapping requirement, and are described in greater detail.

Typically, any pair of tables in a relational database have a single relationship defined between them. In contrast, the IEDC has three pairs of tables, which each have 12[11] relationships defined between the 2 tables in each pair. These are shown in Table 3.2, and a query to duplicate this result is available in Appendix G. The table *data* has 12 columns that reference *classification_items*. *Datasets* has 12 columns each referencing *aspects* and *classification_definition*. The 36 source columns have similar naming conventions with numbering from 1 to 12, and the column *id* (the primary key) of the referenced table is always used.

Table	**Columns with Foreign Keys**	**Referenced Table**	**Referenced Column**	**Key Count**
data	aspect[1….12]	classification_items	id	12
datasets	aspect_[1…12]	aspects	id	12
datasets	aspect_[1...12]_classification	classification_definition	id	12

Table 3.2: IEDC table-pairs with many relationships defined between them

The relationships described here are a centrally important component of the datamodeland are connected to one another. A 'tuple of aspects' specifies the system location of each data item (section 3.2.2). The 12 *aspect[1…12]* columns of the *data* table make up this tuple. The *classification_items* table they reference contains all of the possible values which an aspect may take. The *classification_definition*[12] used by each of the 12 *aspects* is stored in the *dataset* columns *aspect_[1…12)_classification*. This is a critical point to understand, which is not apparent from the implemented relational structure of the IEDC as in Figure 3.2. The 24 columns of the *datasets* table which reference *aspects* and *classification_definition*, are intended to specify the relationships (and hence meaning) of the records in the *Data* table.

Based on this insight, a new image of the core database schema which attempts to explain this design intention is shown in Figure 3.5. The central blue arrow indicates how records

[11] 12 is an arbitrary number, intended as the maximum number of aspects any data item is likely to be described by in practise.
[12] Classification_definition is the IEDC table for storing description of what are usually referred to as *classification systems*. The members of each classification system are stored in *classification_items*

stored in *dataset* describe the meaning of records in *data* which have foreign-keys defined to the *classification_item* table. The blue arrow in the bottom right indicates how *classification_definition* records define the meaning of records in *classification_items*. These are non-intuitive design pattern which increases the difficulty of querying the database and ensuring referential data integrity. An explanation for the cause of these unusual design patterns is proposed in section 6.2.

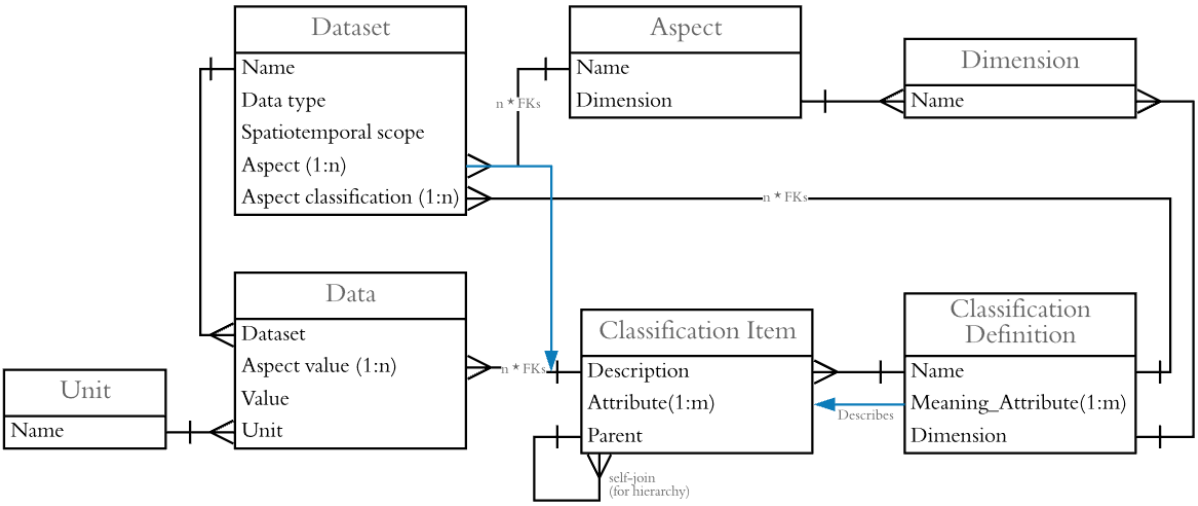

Figure 3.5: core IEDC tables, columns and relationships. IDs and other columns omitted for clarity. Blue arrows are *intended* relationships that are not actually implemented in the IEDC design. n and m are arbitrary integers (see footnote 11).

3.3.5 Storage of quantitative values

This subsection briefly describes the locations in the IEDC for storage of numerical values. Two queries have been written to identify all possible columns where numerical information can be stored, included as Appendix H. The results of this query are included in Table 3.3, which shows there are only five columns within the IEDC datamodel for this purpose. Appendix H includes a description of these locations, as this is necessary information to inform the mappings of Chapter 5. The most important location is for the storage of *facts* (as in section 3.2.2, this is "a tuple of aspects, associated with a value"). This value is stored with *double* precision in the *data* table, using column: *value*.

Table	Column	Type

data	value	double
data	stats_array_2	double
data	stats_array_3	double
data	stats_array_4	double
units	factor	double

Table 3.3: all columns for storage of numerical data within the IEDC

3.3.6 Data categories and types revisited

The evaluation of *data categories* and *types* in section 3.2.2 is continued here, and now extended with the increased specificity of the IEDC. As seen in Figure 3.4, the *datasets* table has many-to-one relationships defined to both the *categories* table and the *types* table. Each *dataset* record can therefore specify separately one data type and one data category, which applies to all of the *data* items contained within the dataset. But it is unclear what is gained by specifying a data category at all. As described in 3.2.2, the *data type* determines the required/optional *aspects*, without any reference to the *category*.

Further, a new option for data integrity conflicts occurs through the implementation. The specification of a *category* is possible in both the *datasets* and *types* tables. Hence if the two categories listed differ, there is no way to determine which should take precedence. This conflict occurs not only theoretically, but also in practise for 3 of the 122 datasets defined in the default data contained in the IEDC. The results demonstrating this and the query to generated those results are included in Appendix I.

To summarise, there are three distinct issues with the attempted data categorisation in the IEDC:

1. The 6 pre-determined categories of data (Table 3.1) in the GDM are not well-supported by existing literature, or justified with sufficient clarity and depth
2. The categories serve no apparent practical function in the IEDC. They increase the complexity of the data model, making it more difficult to understand
3. The IEDC implementation permits contradictory information to be stored regarding which *category* a *dataset* belongs to; in the pre-loaded data, this conflict occurs.

One final comment concerns the potential confusion introduced by usage of the term *data type*, which usually refers to a simpler concept. It is commonly used in programming, databases, and other areas of computer science to describe *strings, integers* and *floats* etc. These data types are defined by the operations that can be performed on them, and the values they can take. The concepts named *data types* in the IEDC include: “material composition”, “process capacity” and “unit-process inventory”. These require *datamodels* to define them, not *data types*. This line of reasoning is continued in the discussion section 6.2, which argues that many of the unusual design patterns encountered in this analysis can be explained with the observation that the IEDC attempts to simultaneously act as both a *database* and a *datamodel repository*.

Chapter 4 The source: unit-process datasets

The previous chapter provided a detailed analysis of the *destination* database, whereas this chapter describes, selects and analyses the *source*. The goal is to select and assess an archetypal example of unit-process data which can be effectively used to test the capabilities of the IEDC. Figure 4.1 summarises the many processing steps which the source data undergoes, prior to its usage in this research. The sections where these steps are described are highlighted in bold.

The first section of this chapter is focused on theoretically deriving data requirements for any datamodel which attempts to represent unit-process data. Lifecycle assessment (LCA) is described next, as a prominent method of IE which uses unit-processes to construct models. Research on a *minimum consensus knowledge model* for LCA data is then introduced. This knowledge model is used to build consistent *semantic catalogs* of the main LCA *background databases* (subsection 4.2.1). The catalog representation of *ecoinvent* (subsection 4.2.2) is selected for this research, specifically the *undefined system model* (subsection 4.2.3). This ecoinvent semantic catalog is assessed in the final part of this chapter, to function as the source dataset in the mappings of Chapter 5.

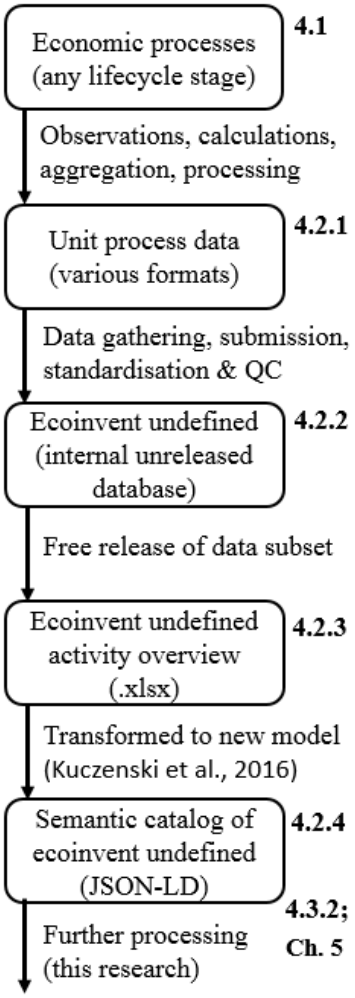

Figure 4.1: the processing steps for unit-process data selected for usage in this research

4.1 Theoretical background and data requirements

This section provides the theoretical basis for unit-processes which is necessary to understand the upcoming analysis and data mapping. An additional objective for this section is to elicit high-level requirements for unit-process datamodels. Seven categories of relevant data are identified in this section, and summarised at the end. The theory presented here applies to all nominal sociometabolic lifecycle stages, including: extraction, production, consumption, waste treatment, recycling and any other technosphere processes.

A unit-process is defined as the "smallest portion of a product system for which data are collected" (ISO 14040, 2006). They can sometimes be used to represent *environmental processes*, but for IE-style modelling, *unit-process* almost always refers to the *economic processes* of the technosphere. Hence these terms are used interchangeably. The *physical economy* or *socioeconomic metabolism* that IE researchers aim to understand cannot be directly observed at large scales (except to some extent using satellite imagery). Therefore, scientific research in this domain involves the construction of representative models. Unit-processes are the basic *building blocks* for this modelling effort, which is described further in section 4.2.

Unit-processes take *commodities* as inputs and generate *commodities* as outputs, as shown in Figure 4.2. Here, commodity means any object that can flow between processes, such as products, resources, services, energy, labour, wastes, or emissions (Heijungs, 1997, p. xvi). Purely *physical* representations of processes omit the intangible aspects, such as services and labour. However, this may reduce the usefulness of the unit-process description for particular modelling approaches which rely on this information. Unit-processes are under human control; people can choose to turn them on or off (ibid.). When they are turned on, the commodities are perceived as *flows*, as they are moving (or flowing) into and out of the processes. Commodities can be classified as either economic or environmental. Those that the unit-process modeller classifies as *environmental commodities* are said to have crossed the *environment-technosphere boundary*, by either entering or leaving human control. *Economic commodities* flow from/to other economic unit-processes; they remain in the technosphere and do not cross the aforementioned boundary.

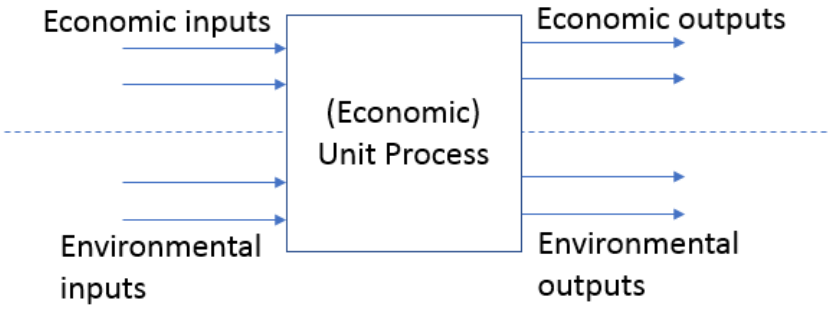

Figure 4.2: a unit-process with economic and environmental commodities flowing in and out

The unit-process conceptualisation makes no attempt to describe what happens *inside* a specific process. Rather, its borders are defined, and quantification of the physical commodities that cross those boundaries constitutes a sufficient and complete description. Nonetheless, additional properties of processes must also be recorded as metadata, to communicate the *system context* of unit-processes. This data is more varied, verbose, and hence open to misinterpretation, compared to the numerical data which somewhat unambiguously quantify the values. This system context data is required due to the intention to use the unit-processes in *economic modelling*. If the information about the original context of the process is not available, then it is not possible to validate the correct usage of the data in the new modelling context.

Unit-processes can be constructed from primary observations and measurements. The commodity flows in and out are observed, quantified, recorded and shared. Mass and energy balance principles apply. Assuming there is no build-up of stock, the flows must balance, including all chemical elements. Hence unobserved flows can also be calculated via stoichiometric principles (Hougen et al., 1954). Observations of economic unit-processes can be repeated and verified. For a subset of small unit-processes that are well-understood, the observed values are expected to conform to predictions from theoretical chemical equations. This foundation in positivist natural science affords researchers a high degree of confidence in some basic unit-process data.

However, most available unit-process datasets are not simple raw observations of this type, and cannot be theoretically predicted through chemical equations. Unit-process data almost always requires extensive processing. This is intended to make the data more representative and useful for its intended purposes. In the words of Heijungs (1997), unit-process datasets are "the result of certain manipulations to find, to correct, to complement, or to adjust data according to certain aims". To construct a representative unit-process data set, a process is observed for a period of time. Averages are usually then calculated based on this time series. The observations may span across a class of related processes or technologies, with the weighted average then intended as representative of the process class. The semi-structured notes describing this data gathering and processing are often included as *metadata*, alongside the numerical values of the flows. The aggregation of related datasets naturally leads to statistical uncertainty. There are many types of uncertainty which can be associated with unit-process data (Imbeault-Tétreault et al., 2013; Henriksson et al., 2014). A detailed discussion of uncertainty is beyond scope, except to highlight the necessity to represent this information, within a unit-process data model.

To summarise this section, the theoretically derived elements of data that are of relevance to any unit-process datamodel are:

1. Process data, which unambiguously identifies the process for humans and/or machines
2. Commodity/Flow data, which unambiguously identifies the flows of commodities
3. Relationship or network data, which enables the association of processes and flows
4. Numerical data, which quantifies the above flows
5. Uncertainty data, which statistically quantifies the uncertainty associated with the numerical data, often by specifying a distribution and parameter values
6. System context or location specification data, such as the spatial and temporal scope of the process
7. Data collection metadata, which describes the selection and processing of the reported observations, including system boundaries choices and other modelling decisions

4.2 Unit-processes in LCA

Unit-process data is used in variety of modelling approaches. Their most common application is in the lifecycle assessment (LCA) method for assessing the environmental impact of product systems (Guinée et al., 2002). All seven of the data elements listed in the previous section are required for high-quality LCA studies. The lifecycle inventory (LCI) analysis phase of an LCA involves constructing the supply chain of a product system from *cradle to grave*. This is usually visualised as a flowchart, which is an important aspect of the communication of each study (Anex & Lifset, 2014). Each *box* within the flowchart is a unit-process. These processes can be very small or large, depending on the resolution of the LCA case-study. Numerical data is then used to quantify or estimate the magnitude of the identified flows between processes within the system.

LCAs also have further data requirements in addition to the unit-processes which are used to construct the system models. These relate to later stages of the methodology: impact assessment and interpretative steps such as the weighting of impact categories (Guinée et al., 2002). The impact assessment phase aims "at understanding and evaluating the magnitude and significance of the potential environmental impacts of a product system" (ISO 14040, 2006). The prior LCI phase has already created a full inventory of flows into and out of the biosphere, for the product system being modelled. In order to evaluate the potential environmental impacts of these flows, the quantified potential impacts of each flow must also be *characterised*.

Standard LCA working practises involve the explicit modelling of *foreground product systems*, supported by generic data from *background databases* (also known as LCI databases). The ecoinvent database was selected as the source dataset for the mappings of Chapter 5, and is described in subsection 4.2.2. Like some other background databases, ecoinvent is available in a variety of *system models,* which create unit-process data for different LCA modelling purposes; these are described in subsection 4.2.3. Data from the background databases is released using standard data interchange formats (section 2.1). These are described in Appendix K, which focuses on the new json-ld (linked data) format used in the semantic catalogs. The final subsection describes research aimed at building a *knowledge*

model to represent all of the LCA background databases. This knowledge model is implemented using *semantic web linked-data* technologies to build the *semantic catalogs of LCA data* which are the focus of section 4.3.

4.2.1 Background processes and databases

Due to the widespread adoption of the LCA approach (Guinee et al., 2011), unit-process data are some of the most commonly available datasets on socioeconomic metabolism, available for many supply chains at high resolution. This data is created and gathered together by various organisations into large *background databases* (Kuczenski, 2015). The processes and other data available in each of them varies, although there is also overlap.

The term database here does not refer to a proper *relational* database or similar, as the providers almost never provide data in this way, even if they use databases internally. The data is released to the community in files, which use common formats (see Appendix K for more information). An online source which collates LCA datasets claims to host over 88,000 unit-processes (nexus.openlca.org, 2019). The site provides access to some of these for free and provide links to the proprietary databases or resell them directly in other cases. These data formats are sufficiently consistent to be used in a wide array of LCA software implementations. Despite well-known interoperability issues and limitations (Herrmann & Moltesen, 2015), these software systems are also widely used outside of academia (Seto et al., 2017), contributing to improved understanding of sustainability-related issues throughout the economy. The goal of this research is to test the ability of the IEDC to effectively represent archetypal unit-process data. These LCA databases are familiar to the vast majority of the IE scientific community. The data they contain has demonstrable utility and functionality and their network structure is broadly recognised and understood. Hence, they are generally appropriate to function as the source dataset for this research. Whilst it may be possible to import all background databases into the IEDC, this research focuses on a single one for only one for expediency: ecoinvent.

4.2.2 The ecoinvent database

The ecoinvent database has evolved from projects beginning over 20 years ago. It is now run as an independent not-for-profit foundation, with the goal to "ensure the further development of a consistent, transparent and trustworthy database for the LCA community" (ecoinvent.com, 2019b). Many of the unit-processes included in ecoinvent are provided by third-parties, via their data submission process (ecoinvent.com, 2019c). This enables users to submit their own datasets into a quality control and release procedure, and hence for the database to grow quickly. Metadata and contextual data are provided to describe the unit-processes in ecoinvent. These include *tags*, *technology levels*, *groupings*, *aliases*, *classifications*, *notes*, *spatiotemporal scopes* and other descriptive information.

The most recent major update was published in 2013 as Version 3, with additional updates every year leading to V3.5 in 2018 (ecoinvent.com, 2019a). There were three relevant methodological developments in version 3: the separation of raw unit-process data from system modelling choices, expansion of 'market' processes, and increased support for regionalised assessments (Wernet et al., 2016). Market processes are considered beyond scope for this research, and the emphasis is also not on regionalisation. Whereas, the separation of system modelling choices from raw data, is the focus of the next sub-section.

4.2.3 System models

Earlier versions of ecoinvent (and other databases) consisted of unit-process data with subjective modelling choices, pre-built into the data. These were caused by the necessity to *link* and *allocate* processes, according to a distinct set of rules, known as *system models* (Wernet et al., 2016). This made some of the data theoretically unsuitable for particular LCA epistemologies such as *consequential LCA* (Weidema et al., 1999). Ecoinvent v3 takes a theoretically improved approach, by separating out these modelling choices from the unlinked and unallocated unit-process data on which they operate. A result of this is that ecoinvent now offers a variety of different data sources to select between, for transformation into the IEDC. For this research, the decision was taken to use the **undefined** ecoinvent system. The context and reasoning for this decision is presented briefly here.

Figure 4.3 displays an adaptation of *Figure 1* from Wernet et al. (2016). The original describes the undefined system as the "primary raw unlinked data". Note that 'raw' is a relative term here, as even this data has already undergone significant processing (see section 4.1). Starting from this common point, *system models* can be applied to create a variety of epistemologically consistent linked datasets. The latest version of ecoinvent includes 3 system models (ecoinvent.com, 2019d). These are the 'cut-off', 'APOS' (allocation at the point of substitution), and 'Consequential' systems. The coded rules for some of these models are partially available through an open-source project called Ocelot, which includes ecoinvent as a partner (docs.ocelot.space, 2017).

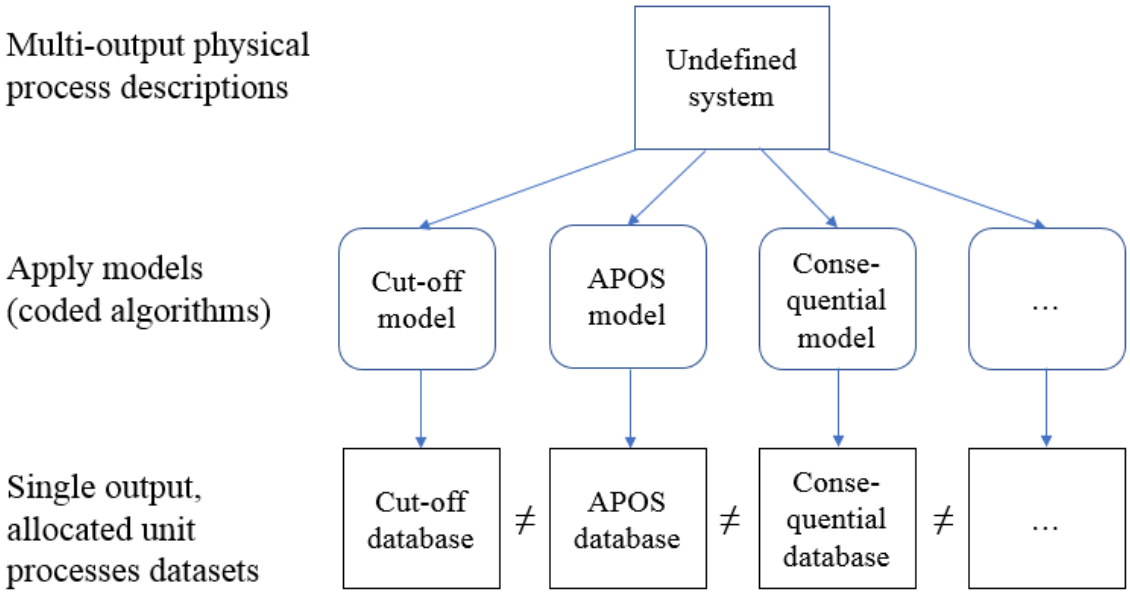

Figure 4.3: the construction of consistently allocated unit process databases via application of system models onto 'undefined' (unlinked and unallocated) unit-process data. Adapted from *Figure 1* in Wernet et al. (2016)

The *undefined* system was selected as the source data for loading into the IEDC because the data it contains is more fundamental than the others options. All other system models require subjective expert opinions, judgements, and decisions to be taken when they are being built. This makes them more useful for LCA modelling. However, the data in the IEDC is not likely to actually be used by practitioners conducting LCA case studies, but rather by researchers seeking accurate descriptions of processes of socioeconomic metabolism. Therefore, the undefined system makes most sense for the requirements of this research.

4.2.4 The minimum consensus knowledge model for LCA

As highlighted in the section 4.2.1, the heterogeneity of LCA background databases can make it challenging to identify their common structure. An attempt to solve this challenge is made by Kuczenski et al. (2016). Their paper describes the results of a workshop where "leading international domain experts" of LCA met with ontology engineers, "to develop a set of simple models called ontology design patterns (ODPs) for LCA information". They named their ODP "a minimal consensus model for LCA". Where *minimal consensus* means that they only included in their ontology, the elements which were present in all of the background databases they analysed. Their ontology reused pre-existing ontologies wherever possible, which is considered good practise on the semantic web (Antoniou & Van Harmelen, 2008). The main resources referenced and reused in this model are two commonly used high-level schemas, and two pre-existing LCA-specific ontologies:

- Schema.org (a top level semantic-web reference, available at https://schema.org/docs/schemas.html)
- QUDT (short for quantities, units, dimensions and data types. Available at http://www.qudt.org/ and highlighted again in the Outlook)
- "Specifying Spatiotemporal Scopes in Life Cycle Assessment" (Yan et al., 2015), available directly at http://descartes-core.org/ontologies/lca/1.0/stscope.owl
- The aforementioned OpenLCA linked-data schema (Ciroth & Srocka, 2015)

The knowledge model is displayed in Figure 4.4. See Kuczenski et al. (2016) for a full description, or section 4.3 for a code-based exploration of the implementation of this model.

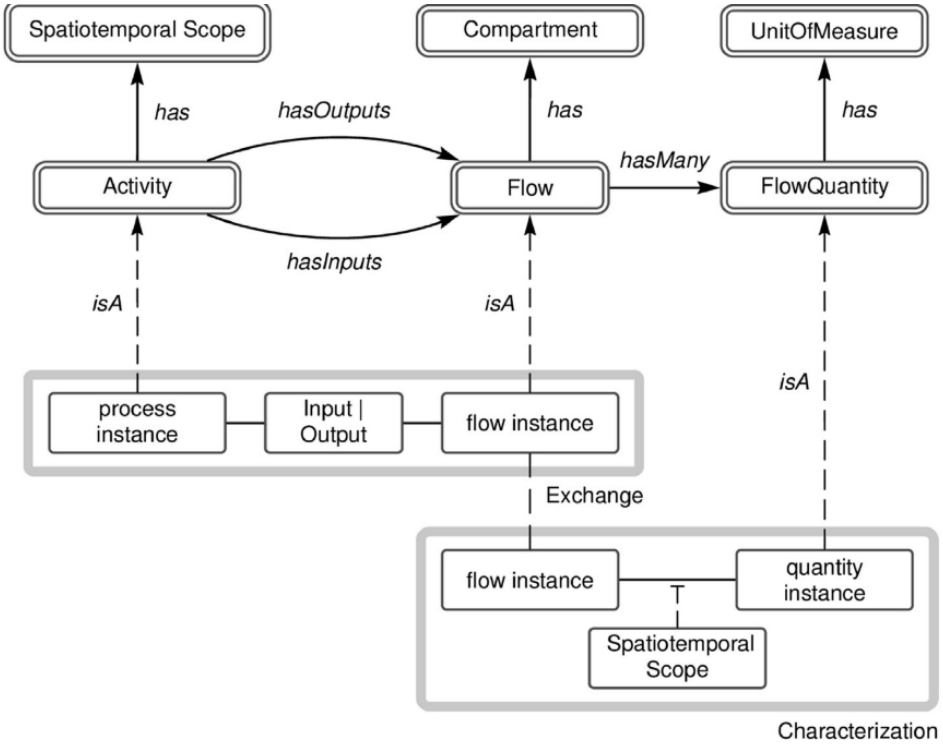

Figure 4.4: Caption in Kuczenski et al., (2016) Fig 1. states: "The consensus knowledge model showing three entity types and their defining properties. An exchange reports a relationship between activity and flow instances, and a characterization reports a relationship between flow and flow quantity instances."

This consensus model is used to represent a subset of the data contained in a number of prominent LCA *background databases.* These representations are described as Semantic Catalogs of LCA data, (hereafter, *catalogs*). These catalogs can be downloaded from the internet (GitHub, 2019b) in the .json-ld file format (Appendix K). Source code which can be used to generate the catalogs is also available (GitHub, 2019a), however this is beyond the scope of this research. In order for the catalogs to be released freely without licensing restraints, only freely available public data could be included. Ecoinvent is proprietary. However, a subset of the data is also released in 'activity overview' spreadsheets files, which are free and publicly available. The developers of the semantic catalogs chose these spreadsheets as their data source for ecoinvent, to avoid licensing concerns. The semantic catalogs were selected as a good example of LCA or unit-process data for the following four reasons:

1. Representativeness. If the authors claim of being a "minimal *consensus model* for LCA" is accurate; then the data is an archetypal example of unit-process data, and ideal for the purposes of this research.
2. Availability. The catalogs are easily discoverable online, and were downloaded with a single line of code (a 'git clone' to the URL of the repository)
3. Licensing. The authors of the catalogs completed the time-consuming work of collecting and sharing only the publicly available data from the LCA background databases. No proprietary or paid-for data was used, meaning that the dataset is appropriate for free scientific analysis and distribution without licensing concerns.
4. Usability. The data in the files is human and machine readable with relatively simple code. This means the transformations required are likely to require comparatively limited processing to satisfy the goals for the research.

4.3 The semantic catalogs of LCA data

This section presents an overview of the semantic catalogs based on the published article which describes them (Kuczenski et al., 2016), and code-based analysis of their contents. The original knowledge model (Figure 4.4) is then converted into a simplified entity-relationship diagram (Figure 4.5), to communicate the catalog structure as used in practise. The ecoinvent 3.2 undefined catalog is then explored in detail, as a pre-requisite requirement for the mappings of the next chapter.

4.3.1 Overview of the catalog contents and structure

The aforementioned semantic catalog representations are available for four main background databases (US LCI; GaBi 2016, with 22 extension databases; ELCD v3.2 and ecoinvent v3.2). Some of these catalogs are available in multiple system models and versions. With regards to the actual data included in the catalogs from each source, the authors state: "for data sources that are free, including USLCI and ELCD, the catalogs include the semantic, structural, and quantitative data (i.e. processes include lists of exchanges, and those exchanges include values). For data sources that are not free, including the GaBi professional

database and extensions, and the ecoinvent database, the catalogs include only the semantic information and partial structural information (a listing of reference exchanges or characterizations). They exclude non-reference exchanges and characterizations, and exclude all exchange values and characterization factors". However, *non-reference exchanges* actually appear be included for ecoinvent, as explored in section 4.3.2.

In Figure 4.4 above, both the *base classes*, and the *instances* of those classes are displayed. Definitions for these terms are available in the glossary in Appendix B. The actual data included in the catalog files is all *instance data*. These entity instances have a *context* which describes how they should be interpreted[13] and their semantic relationships and links to other objects on the semantic web (see Chapter 8) The most important instance data for the purposes of this research are shown in the *entity-relationship diagram* (Theodoulidis et al., 1992) of Figure 4.5, which is described below and followed by an example.

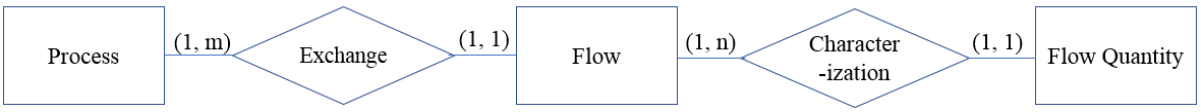

Figure 4.5: The 3 entity instances in a semantic catalog of LCA data (in squares), with their relationships (diamonds) and cardinalities (bracketed). Attributes not shown.

A process contains *1 to m* exchanges, which can be inputs or outputs. Each exchange points to one flow. Each flow contains *1 to n* characterizations. Each characterization points to one flow quantity. Exchanges can be identified by the unique combination of three items: the process they are contained within; the flow they point to using it's GUID; and their *direction*. Characterizations can be identified by the unique combination of: The flow they are contained within; and the flow quantity they point to using it's GUID. Both exchanges and characterizations can be labelled as the *reference* for their process or flow respectively. This

[13] The @context description for the catalogs is visible at https://bkuczenski.github.io/lca-tools-datafiles/context.jsonld

means that other *exchanges* and *characterizations* respectively are normalised to this reference value.

An example of this data structure in practise is shown in Figure 4.6 and described here with example values in bold. This **electricity mix** process has an exchange **input** of **9.6e-11** of the flow **Potassium Chloride.** This flow has the reference characterization of **1.0** of the flow quantity **Mass**, which has the unit **kg**.

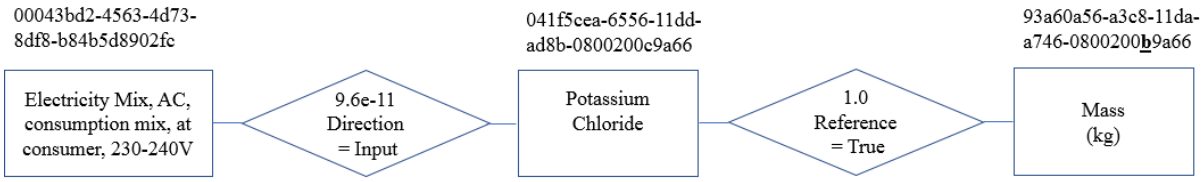

Figure 4.6: example of a related process, flow, and flow quantity. Entity names shown in boxes, with their GUIDs above. Relationship values and example attributes shown in diamonds.

4.3.2 Exploring the 'ecoinvent 3.2, undefined' semantic catalog

This section describes the selected source catalog dataset in detail. It is based on the actual data contents, and intended to provides the necessary pre-requisite knowledge for the mappings of Chapter 5. Appendix L includes the source-code used for this analysis. This was adapted from notebook-based examples provided by the researchers and developers that created the catalogs (GitHub, 2019c).

In this section, the capitalization of letters within the key names matches that which ids found in the data source. *Key* refers to the 'key : value pairs' of the JSON data format and the Python dictionary data structure (PythonDocs, 2019), rather than to the primary and foreign keys of relational databases. Table 4.1 provides an overview of the top-level keys in each catalog, which are described in the following subsections. These practical descriptions complement the theory and examples of Figure 4.4 and Figure 4.6.

	Process	Exchange	Flow	Characterization	Quantity
Common to all types	*entityType*	*entityType*	*entityType*	*entityType*	*entityType*
Common to primary entities	*Comment*		*Comment*		*Comment*
	entityId		*entityId*		*entityId*
	externalId		*externalId*		*externalId*
	Name		*Name*		*Name*
	origin		*origin*		*origin*
Common to relationships types		*isReference*		*isReference*	
		value		*value*	
Distinct per type	*Classifications*	*direction*	*CasNumber*	***quantity***	*Indicator*
	exchanges	***flow***	***characterizations***		*Category*
	SpatialScope		*compartment*		*Method*
	TemporalScope				*referenceUnit*
					UnitConversion

Table 4.1: top-level *keys* or *attributes* of the primary semantic catalog entities and their relationships. Attributes of particular entities listed vertically; recurring attribute names emphasised horizontally

4.3.2.1 Common data attributes

The only key which is common to all entities is *entityType*, which has as a simple text-string value stating its entity type, such as “process” or “exchange”. The primary entity types: process, flow and quantity, have 5 other keys in common. *Name* is used to help a human interpreter understand what the process does using natural languages. Whereas *entityId* is used for unambiguous identification by people or machines, via a GUID value (globally unique identifier). To demonstrate the non-uniqueness of names in this catalog: of the 13307 processes, there are only 6046 distinct process names. *externalId* is intended to hold data which can uniquely identify an entity within the source database, which may use a different entity identification convention to the semantic catalogs. In the case of the EI3.2, the *externalId* holds duplicate data to *entityId*, because ecoinvent also uses 32-character alphanumeric GUIDs, which means processes can easily be found (for example) by a Google search. The *origin* key:value pair holds the filename where the data was extracted from during the creation of the catalogs. It is hence metadata which describes part of the data provenance. Finally, *comment* is used to store extra descriptive text or category-based data, the details of which vary between each entity.

The two keys which are common to all entities and relationships are *isReference* and *value*. *isReference* is a Boolean value which indicates whether the exchange or characterization

relationship being described, is the reference. For an exchange, this is usually used for the functional flow of the process, the flow for which the process is intended to produce. For a characterization, it is the reference characterization to which other values are normalised. This is usually mass, such that other characterizations (e.g. global warming potential or volume) are recorded relative to the mass of the flow. The key-value pair named *value* is the only place within the catalogs where numerical data is expected to be stored. As described in section 4.3.1, this is not included for the proprietary ecoinvent data.

4.3.2.2 Distinct data keys per type

The process instances have four distinct data keys: *Classifications*, *exchanges*, *SpatialScope* and *TemporalScope*. The classifications are used to categorise the processes depending on their function. The top 5 most common classifications in the catalog are shown in Table 4.2 as examples. The *exchanges* entry holds a list containing every exchange which flows into and out of the process. The source code in Appendix L extracts these exchanges from the processes and creates separate data objects, such that they can be treated separately as part of this analysis. *SpatialScope* and *TemporalScope* reuse the ontological work of Yan et al. (2015), to describe the spatiotemporal scope for which the process is considered representative.

Process Classification	**Exchange Count**
Electric power generation, transmission and distribution	3015
Manufacture of basic chemicals	1099
Treatment and disposal of non-hazardous waste	980
Electric power generation, photovoltaic	395
Steam and air conditioning supply	343

Table 4.2: most common process classifications found in the ecoinvent semantic catalog

Exchange instances have two distinct data keys: *direction* and *flow*. *Direction* can be “Input” or “Output”. This specifies the orientation of the exchange, relative to the process. The *flow* key has a GUID as a value. This GUID points to a specific flow, such that the type of commodity flowing into or out of the process can be identified.

The Flow instances have three distinct data keys: *CasNumber*, *characterizations* and *Compartment.* CAS refers to the Chemical Abstracts Service. This is a widely used standard, which "uniquely identifies chemical substances on the basis of composition and structure" (Dittmar et al., 1976). The *characterizations* entry is a list, containing all of the characterizations which can be used to characterize the flow, via pointers to flow quantities. *Compartment* refers to the medium that contains flow. There are 25 distinct compartment categories listed for all flows in the ecoinvent catalog. Table 4.3 shows the most common. Characterizations have only one distinct data key: *quantity*. This simply points to the GUID of the Flow Quantity data which describes the flow that the characterization is a member of.

Compartment	**Flow Count**
Intermediate flow	2754
air	1723
water	1423
unspecified	677
soil	500

Table 4.3: most common flow compartment categories found in the ecoinvent semantic catalog

Flow Quantities are the final entity type, and have five distinct data keys: *indicator*, *category*, *method*, *referenceUnit* and *UnitConversion*. The *method* refers to the impact assessment method family used, such as "CML 2001". For readers unfamiliar with lifecycle impact assessment theory, refer to Guinée et al. (2002). For this catalog, there are 465 distinct *indicators* sourced from 39 different *methods*, grouped into 100 *categories*, using 58 *referenceUnits* for their measurement. Examples of common *referenceUnits* are shown in Table 4.4.

Reference Unit	Quantity Count
points	193
kg	115
kg 1,4-DCB-Eq	68
kg CO2-Eq	36
UBP	34

Table 4.4: most common flow quantity reference units found in the ecoinvent semantic catalog

This concludes the exploration of the ecoinvent 3.2, undefined system model, semantic catalog. The next chapter uses the knowledge gathered in this chapter to map this source dataset onto the IEDC database.

Chapter 5 Source to destination: mapping the datamodels

The previous two chapters collectively satisfy the pre-requisites to the data mapping which are listed in section 2.3.2. As such, the source and destination are now understood and documented, from a conceptual and detailed perspective This chapter will attempt to answer research sub-question 3, by demonstration. The assumption driving the mapping effort is that:

It is possible to comprehensively map all entities, relationships and attributes from the selected ecoinvent 3.2 undefined semantic catalog source dataset, into the relational datamodel of the IEDC database.

Figure 5.1 displays an overview of the requirements for the mapping, which are addressed through this chapter. It is based on Figure 4.5 and Figure 3.2. It is important to note that the mapping described in the subsequent sections is one of many possible mappings between this source and destination. Many decisions taken in this process have an element of subjectivity, and alternative researchers may prefer different approaches.

The chapter is divided into the following sections. First, the high-level conceptual data mapping is presented, giving a general outline and explaining the reasoning behind the most significant choices which were forced or made. Next, the complementary detailed mapping is described, with reference to the data mapping document included as Appendix A. The implementation of these mappings into an ETL process is then described in the next section. The result of this chapter is that the data can be represented into the IEDC, the assumption stated above is correct. This thesis research also aims to determine whether the representation is effective. This topic is addressed in section 6.1 of the Discussion.

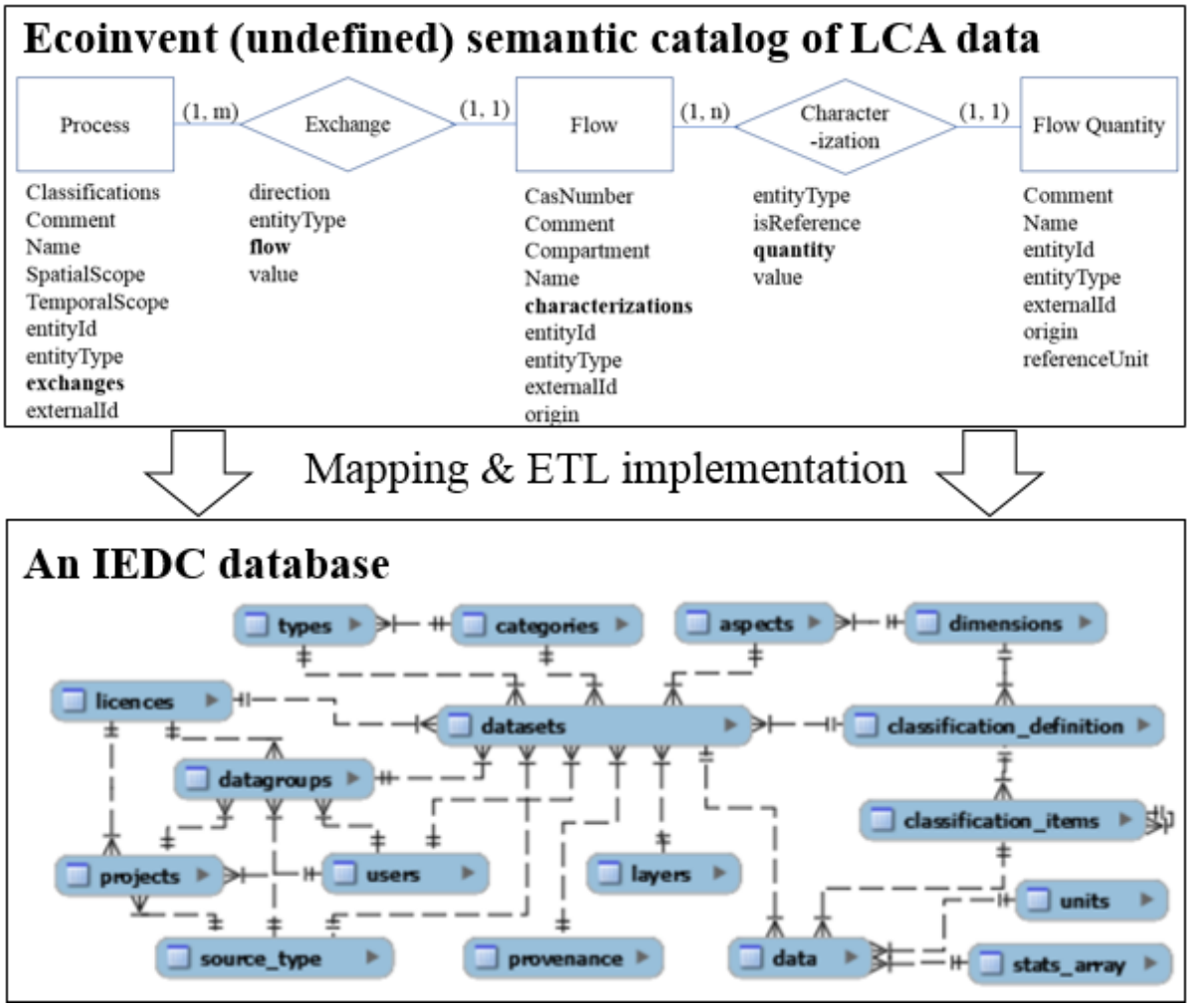

Figure 5.1: the mapping and ETL challenge, from semantic catalog source data to the IEDC destination tables

5.1 Conceptual data mapping

Figure 3.4 showed a simplified view of the IEDC schema, after removal of the auxiliary tables as explained in the accompanying text. The mappings of source data to these auxiliary tables are shown with comments in 0. To further simplify the mapping requirement on the destination side, some additional 'quick-win' mappings are described here:

1. Each semantic catalog is one *datagroup*. As only one semantic catalog was selected as the source data, only one entry is required into the datagroup table.
2. The *type* of each dataset is 'unit process'.

3. *Layers* described by the source data are ‘Mass’ and ‘Value per Mass’
4. Of the 6 pre-determined data *categories*, none of the options are fully appropriate (see section 3.3.6 and comment in the Mapping Document). Category “Flow” is selected as the closest match.
5. The *dimensions* pre-loaded into the IEDC are sufficient for the requirements of this data source, and each *aspect* and *classification* definition entry links to a different dimension (5.2.2)

Removing these easily-mapped concepts from the destination data model, leaves the relational model shown in Figure 5.2. The target for the challenging part of the mapping now consists of just 6 tables and 7 relationships (including a self-join); it is therefore more tractable that the full original schema.

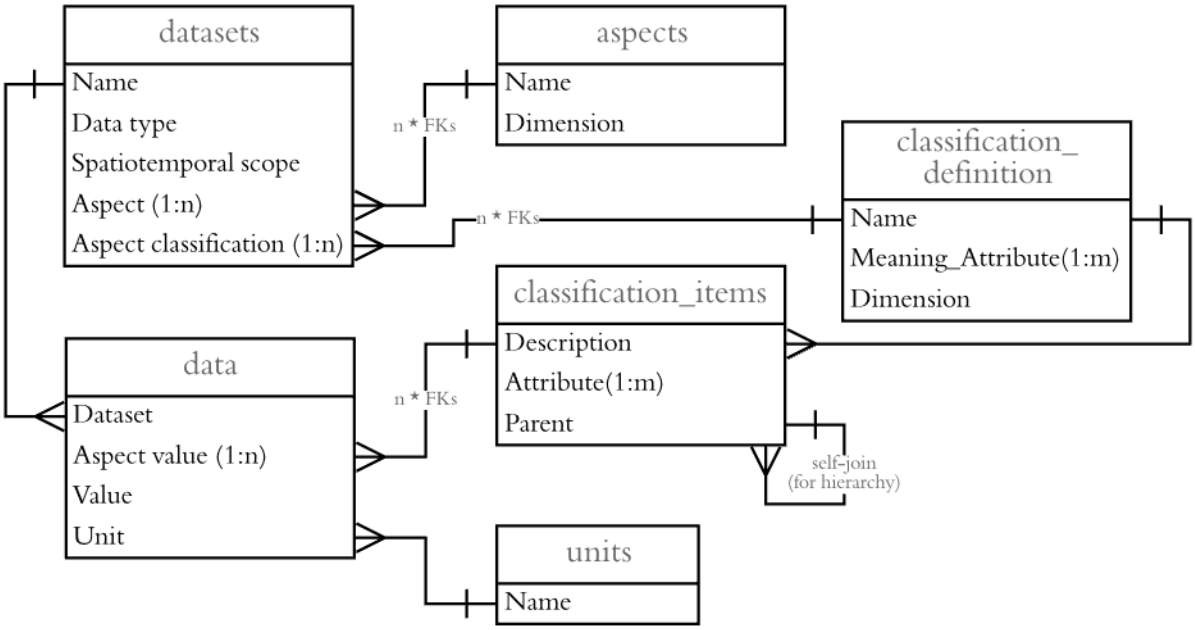

Figure 5.2: core IEDC schema, after removal of tables with simple mapping requirements

As described in section 3.3.5, a key *anchor point* in the IEDC, which directs the mapping of any data source, is the *data* table. Crucially, this is the only place that can store continuous numerical information (in the *value* column). In contrast, the source datamodel stores numerical information in two different places: the *value* fields of both the *exchange* and *characterization* relationship entities (e.g. see Figure 4.6). This leads to the question: which *value* should be stored in the data table?

If the *exchange value* is inserted into the data table, this would mean *characterizations* and *quantities* must be mapped somehow to the classification tables. If the *characterization values* are stored in the data table, then the *exchange* and *flow* data including their relative magnitudes would need to be stored in those tables. Neither of these solutions is satisfactory. An alternative approach is instead chosen for the conceptual mapping: transforming the source data by *flattening* it.

Flattening data de-normalizes it and creates additional redundancy by duplicating values across multiple rows. During flattening, the two numerical values are multiplied by each other to form a single value. If they had units, they would also be combined into a compound unit. However, neither the *exchange* nor *characterization* have a unit-of-measure directly specified. Rather, the reference unit is specified in the associated *flow quantity.* After flattening, this *unit* applies to the newly flattened row.

Based on this approach, an overview of the conceptual mapping is shown in Figure 5.3. The *process* concept of the source (along with some data from other objects as part of the flattening process) is mapped onto the *dataset* concept of the destination. *Flow quantities* of the source are broadly mapped onto units in the destination. The *exchange* and *characterization* relationships are combined and mapped onto the *data* table of the destination. If a *process* has **m** *exchanges*, and a *flow* has **n** *characterizations*, then **m*n** rows are entered into the *data* table for that process.

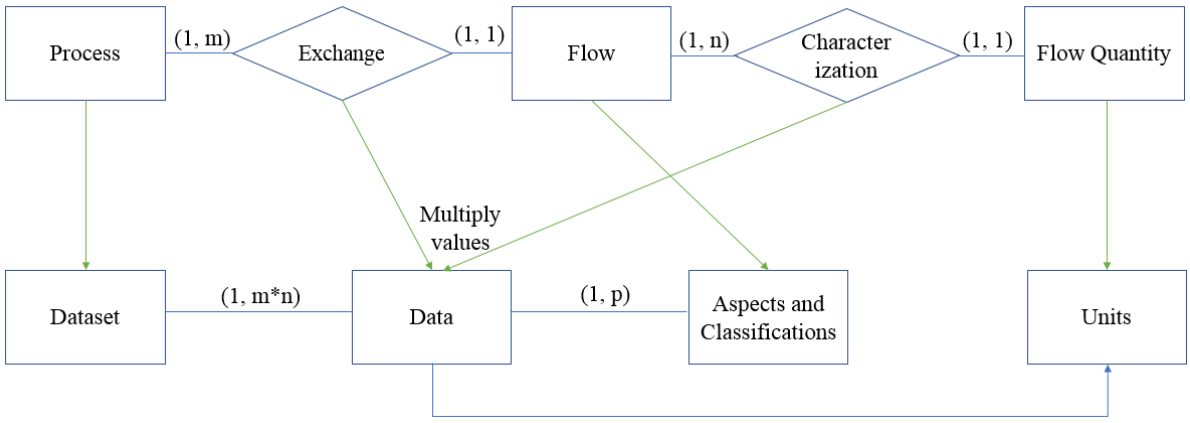

Figure 5.3: overview of the source to destination conceptual mapping

All the flows of the source dataset are modelled as a classification scheme in the *classification_definition* table. Each of the flows present in the source dataset become an entry in the *classification_items* table, including their attributes. The specific *flows* that are referenced by each of the flattened entries in the *data* table are referenced via the *aspects* table (see section 3.3.4). Three additional attributes from the source entities which do not fit into the Data table must also be specified using a combination of *aspects* and *classifications*: exchange["Direction"], exchange["isReference"] and characterization["isReference"].

5.2 Detailed data mapping

This section describes the *contents* of the Detailed Data Mapping document, duplicated in Appendix A. The description of its structure is included as Appendix O. This describes the different sheets, columns, mapping identification scheme, legends and formatting. This can be used to as guidance for a clear understanding of the meaning of the document. Whereas the section on the *contents* describes the important decisions taken in the actual data mapping.

5.2.1 Auxiliary mappings

The mapping document describes mappings which are labelled from 0 to 22. Mappings 0 to 10 are classified as auxiliary. These do not include reference to the source data, but rather are

hard-coded values that have been written based on the knowledge gathered and generated in earlier chapters. Of these, the IEDC already has appropriate data pre-loaded for 8 of the mappings, with just 3 auxiliary mappings requiring new data to be entered in the destination and hence *insert* statements (see section 5.3).

The entries have either been previously explained, or are simple and self-explanatory, based on the guidance above. Some further potential IEDC data integrity conflicts are highlighted in the *comments* column. For instance, *data_categories*, *data_types*, and *data_layers* are all specified for both the *projects* and *datagroups* levels of the data storage hierarchy (see section 3.3.2). The *comments* also include some recommendations on alternative approaches to modelling, some of which are highlighted in the Outlook.

5.2.2 Classification and aspect mappings

Of the 12 *core* mappings, 9 of these made up of 3 groups of 3 entries into the *aspects, classification_definition*, and *classification_items* tables. Each of these groups will be referenced in the *data* and *dataset* entries, as described in section 5.2.3. As summarised in Table 5.1: a single record for each of the three groups is inserted into *classification_definition* and the *aspects* tables. These entries are similar because the *aspects* entry is used to link each *dataset* unit-process entry to a *classification_definition* (section 5.2.3). *Classification_items* receives multiple inserts per group, as it contains all the items which belong to each of the three classifications which are defined in these mappings.

Mapping IDs	classification_definition	classification_items	aspects
12-14	Ecoinvent_Commodity_Flows	All distinct flows in the source data	Commodity_Flow
15-16	Boolean	2 records: True & False	Boolean
17-19	Direction	2 records: Input & Output	Direction

Table 5.1: classification and aspect mappings summary

The most interesting mapping group here is Ecoinvent_Commodity_Flows. Every *flow* entity from the source dataset becomes an entry in this classification system. As such, the *attributes* columns of the classification_items are used to store the source data. The first 4 optional attribute columns are used to store the source data fields: *entityID*, *Name*, *CasNumber* and *compartment*. As such, the classification system of the IEDC is quite capable of accepting all of the necessary source data, despite some shortcomings (see Discussion section 6.1.2).

5.2.3 Other core mappings: units, datasets and data

The remaining 3 core mappings are into the *units, datasets* and *data* tables. These are the most important mappings, and constitute the vast majority of the total quantity of data that is inserted into the IEDC during the ETL process.

As shown in Figure 5.3, the *Flow Quantity* concept of the semantic catalogs maps approximately onto the *units* table of the IEDC. The *Flow Quantity* attribute *referenceUnit* contains data which can be used in either/both the *unit_code* and *unit_name* columns of the destination table. Every distinct entity in the source dataset becomes a single record in the IEDC table.

The *datasets* table receives a single entry for every *process* in the source dataset. The *datagroup, category, type, layer, provenance, source_types, user* and *license* of every entry in the table is the same, with the specific values visible in the mapping document and justified where necessary. The *Name* attribute of the source becomes *dataset_name* in the destination table. The other *Process* attributes: *ISIC Number, Classifications, SpatialScope,*

TemporalScope and TechnologyLevel are mapped respectively to the *dataset* columns: *process_scope*, *product_scope*, *regional_scope*, *temporal_scope* and *description*. Four *aspects* and *aspect_classifications* are specified for every *dataset* entry. These are explained in section 5.2.2, and are used to specify which *classification* schemes each of the entries into the *data* table (see below) relates to. There are four groups entries here rather than the three explained in section 5.2.2, because the Boolean classification can be used twice, to specify both the *isReferenceExchange* and isReferenceCharacterization values.

The final mapping is to the IEDC *data* table. Each process entry into *datasets* will have many entries into this *data* table associated with it. The flattening transformation which creates these records is described in the conceptual mapping section above. This table receives attributes from four out of the five possible source entities: *Exchange*, *Flow*, *Characterization* and the unit from Flow *Quantity*. Aspects 1 to 4 in this table match with the entries of datasets, as described above and in the mapping document. This table permits the specification of two units: for both numerator and denominator, but only the numerator entity is required for this source dataset (with the denominator set to 1, as to have no effect). The source dataset has two comment entries which must be stored in this table, from the *Flow* and *Flow Quantity* entities. As such, these comments are appended to one another for mapping to the *comment* column, which has a suitably large data type to handle this.

5.3 Extract-transform-load implementation

It is possible to load data into the IEDC using a formatted *template* spreadsheet, and associated software-based *parser* and *importer*. However, preliminary exploration ruled out this option. Using the template would not provide sufficient flexibility for the complexity of the mapping under attempt. It would also require additional interfacing with the template and importer, which have not been tested beyond the IEDC development team. They state that "the documentation of the process of converting available data from the different sources, accounting routines, or model calculations into the IEDC template is the under the responsibility of the data provider, and no guidelines or standards exist". For these reasons, a custom ETL process was designed for this implementation.

In general, ETL implementations can be said to exist of five stages (Haq, 2016). A **source** data source is first extracted into a **landing area**. This data is loaded into memory or prepared in some other way into a space which can be referred to as the **staging area.** In this area, transformations are applied, until the data has been processed into a **load-ready** state. Finally, this data is inserted into the target database, completing the ETL implementation. An overview of these steps is shown in Figure 5.4

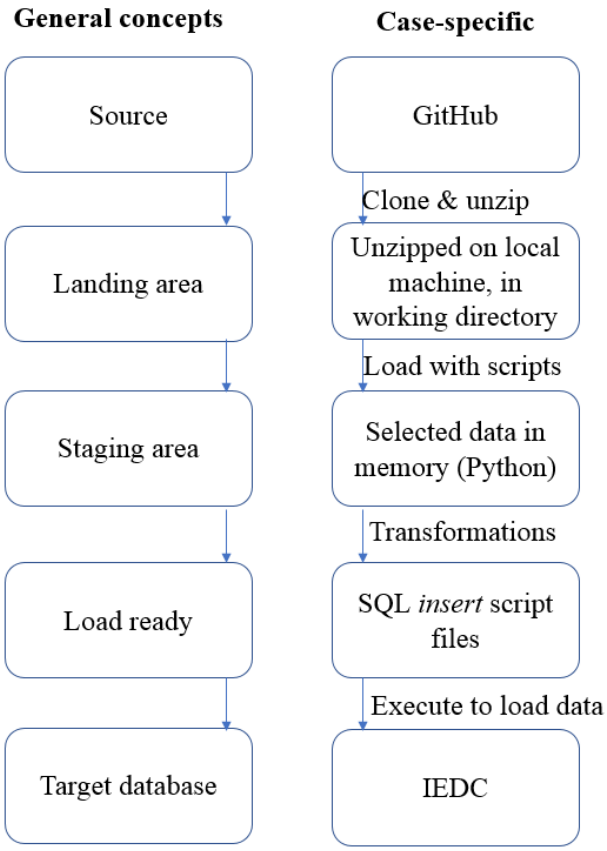

Figure 5.4: Flow chart for data extraction, transformation and loading process. General concepts on the left are based on Haq (2016). Application of concepts to this research method are shown on the right

For this research, the original **source** for the data is the LCA-Data-Tools GitHub repository (GitHub, 2019b). The repository is copied to a **landing area** on the local machine using a *clone* command of the Git version control system (GitHub, 2019c). From here they can be unzipped from the .gz format to the .json format and placed in an appropriate working directory. The json files are loaded into the memory of a Python kernel, which could be

considered a **staging area**. The code to handle this is available in Appendix L. The coded functions transform the catalog data into new objects which simplify further processing. Then later functions iterate over these objects, to generate strings of properly-formatted SQL code. These strings are saved as **load-ready** (.sql) files. The IEDC **target database** is loaded with the semantic catalog data by executing the scripts.

When loading data into the destination database, there are conditions which constrain the order that the tables must be loaded. These constraints are determined by the foreign key relationships between tables. When a foreign key relationship is defined on a column, the table which that key references must first be populated. Else the insert statements will raise errors and the data loading will fail. In general, the loading order dependency for a relational database can be derived by checking which tables have foreign keys defined to others. Tables with no foreign keys defined should be loaded first, as other tables will rely on the data they contain.

For the IEDC, the loading order dependency is included as the second column of 0. This result is applicable to any source dataset intended for loading, and is not related to the source unit-process data used for in research. Six separate levels (or steps) must be used during data loading, to guarantee the dependency constraints are not violated. The data loading within each level can be carried out in any order, including in parallel. The nine tables which have no FKs defined are defined as loading level 0. The tables which have FKs defined are ordered from levels 1 to 5. Three tables (aspects, classification_definition, and types) only have FKs defined to tables at level 0, so they can be loaded once level 0 is complete, and are labelled as level 1. This pattern continues until discovering the last table which can be loaded. This is the *data* table. An example of the *data* table's dependency chain is shown below, where -> means 'depends on the data in'.

Data -> datasets -> datagroups -> projects -> classification_items -> classification_definition -> dimensions

Chapter 6 Discussion

6.1 Summary and evaluation of the IEDC unit-process representation

The mappings described and implemented in the previous chapter demonstrate that it is possible to fill the IEDC with the unit-process data. This partially answers the question of the research, but is an insufficient conclusion on its own. The research question also aimed to answer whether the representation is *effective*. This section evaluates the effectiveness of the destination datamodel and database representation, broadly following the *Characteristics of datamodel quality* described in section 2.2. These were: representation validity; ease of understanding; promotion of data reusability and integration; and enforcement of domain rules.

6.1.1 Mapping successes

The most significant success of the research is to demonstrate that all of the data in the selected source dataset can be mapped to a specific location in the IEDC. This includes each of the five source entities, and all of the attributes they contain. The representation of the concepts *semantic catalog(s)* and *process* were comparatively clean. Each *catalog* becomes a single *datagroup*, and all of the semantic catalogs together can be grouped as one *project* (see 3.3.2). *Processes* are represented somewhat cleanly (see below) as *datasets*. The datasets table had many columns with flexible data types, which permits a satisfactory representation of all the necessary source data attributes with minimal processing.

It may be possible to recreate the original source data from the destination database via a series of reverse queries and operations. This is likely possible, as all entities and attributes from the source dataset are mapped onto at least one location in the destination database. Hence the ETL process can be completed without loss of information. Proving this would not be trivial, and is beyond scope.

6.1.2 Challenges and shortcomings

Despite these successes, there were many challenges encountered in the mappings and transformations. The destination representation of unit-process data does not broadly adhere to the characteristics of data model quality (section 2.2), and has a number of shortcomings. The issues described here are: the sacrifice of important relations in the source data model; the storage inefficiency; the absence of domain-rule and data-integrity enforcement; and the difficulty of retrieving the newly imported data.

The *flattening* of the source data (section 5.1) enabled the representation of many source entities: exchange, flow, characterization and flow quantity. However, there are significant downsides to this approach. The *minimal consensus knowledge model* for LCA data (section 4.2.4) made a convincing argument that these entities should be considered as separate and interrelated. The efficient modelling of these concepts which is the hallmark of the source datamodel is lost when represented in the IEDC data model. The destination tables that contain the data do not effectively describe their row-based contents, in a way that can be easily understood.

Storage inefficiencies in the IEDC occur in three areas: table, row, and column data type inefficiencies. Table inefficiencies relate to the high count of columns in many of the tables. Many of these are not used after the ETL of this research, and are rarely used in any of the pre-loaded data. Row-level inefficiencies relate to the necessity of the flattening process described above. This necessarily *denormalizes* the source data, creating many times more rows in the *data* table than were present in the source data (section 5.1). Column data-type inefficiencies relate to the *data type* and *precision* of each of the columns. Generally, good practise for efficient data storage requires that the modellers use the smallest possible data type that can reliably contain the expected values. The IEDC uses many column data types with higher precision than the range of values they contain. For an operational system which has performance requirements, these inefficiencies will become problematic. However, the IEDC is a prototype, so it could be reasonably argued that these inefficiencies are not a problem for its intended purpose.

The next issue is the absence of domain-rule and data integrity enforcement which has been mentioned a number of times previously (see sections 3.3.3, 3.3.4, 3.3.6, 5.2.1). When a user queries the database, they desire a clear and unambiguous response. The IEDC permits inconsistencies to be stored in a variety of places. A database that permits contradictory information to be stored, must also provide a method for resolving them. For instance: "when the *license* of a *project* and *datagroup* differ, the *datagroup license* takes precedence". Without this, the user cannot know how to interpret the results of their query, and will lose trust in the data source.

The final shortcoming relates to the difficulty of querying the data stored in the IEDC due to its attempt to follow a *general* datamodel. Relational database tables typically represent a single concept, and are named as a singular noun (section 2.1.3). The IEDC permits combinations of models to be included within the same tables (section 6.2). A single column may hold many different *concepts* and *attributes*, while their meanings are defined in a different column in a different table (section 3.3.4). This is likely to increase the difficulty of writing, validating, parameterizing and debugging queries. Inaccurate queries may lead to modelling and analysis mistakes.

It may be argued that these identified challenges and shortcomings are an artefact of the specific implementation in this research, rather than inherent issues with the IEDC. This is likely the case for the first item identified: the 'sacrifice of important relations in the source data model'. Via alternative mappings, it could be possible to retain some of the important relationships between the entities in the source. The 'storage inefficiency' issues are certainly present in the IEDC model and not an artefact of the mapping. This is also true of the 'domain-rule / data-integrity' problems, as clear from the fact that these issues were highlighted and demonstrated earlier in the research (Chapter 3), prior to any mapping attempt. The difficulty of querying data also likely cannot be avoided via any mapping approach. This is because a different aspect structure is used for every *data type*, and records adhering to all these different structures are stored alongside one another in the same *data* table.

6.2 Other high-level considerations regarding the GDM and IEDC

A broad objective of this research is to synthesise and communicate accessible knowledge and recommendations on the topic of data modelling and management. This discussion section consists of three related subsections. First, the question is posed as to whether the GDM qualifies as a datamodel as defined in the literature. Next, the relationship between the GDM and IEDC are explored, to determine if the IEDC is a technology-specific implementation of the GDM, or something else. Finally, it is argued that both the GDM & IEDC attempt to contain *datamodels* and *data* in parallel. And that this design pattern enables the deferral of actual data modelling to a later time.

6.2.1 Is the GDM actually a datamodel?

Three definitions from section 2.1.1 are re-introduced here in order to guide the assessment of whether the 'general data model' qualifies as a datamodel. Tsichritzis & Lochovsky (1982, p. 10) say datamodels must "define the rules according to which data are structured". In the GDM, the 'data types' concept is used to define the "required and optional aspects". Example proposals for 'data types' are given, but these are not pre-determined as part of the GDM. Pauliuk et al. (forthcoming) calls for "scholars from different modelling communities to enter a dialogue and reach consensus about the aspect structure and the semantics of the different data types". It is hence clear that the data-structuring rules are not defined as part of the GDM, but rather *within* it. Tupper (2011, Ch.10) defines a datamodel as "a symbolic or abstracted representation of something real or imagined". This definition is very broad and may include the GDM within its scope. However, it is unclear which "something" is represented, without referring again to the *data types*. Oppel (2010, p. 4) defines datamodels as "abstractions of existing or proposed databases". As discussed in the next subsection, the GDM may or may not be included under this definition. Based on these three assessments, it is remains unclear whether the GDM qualifies as a datamodel.

6.2.2 The relationship between the GDM and IEDC

Chapter 3 began by stating the hypothesis that "the IEDC is a technology-specific implementation of the GDM, and other hypothetical implementations of the GDM should be interoperable with the IEDC". Section 3.2 highlighted a number of ambiguities in the limited information available describing GDM and the previous section was unable to conclusively answer the question as to whether the GDM should qualify as a datamodel. Taken together, these issues suggest that the GDM require some additional work or redesign, before it is ready for widespread uptake by the community.

In contrast, the datamodel of the IEDC is clear, explicit, and not reliant on the natural language description in the manuscript. It tangibly exists as an open and accessible database. As such, it is formally described in DDL (data definition language) code. which explicitly describes this design. This code can be accessed through the instructions in Appendix E, and will create an identical database schema whenever it is run inside the MySQL DBMS. In order to reach this level of specificity, many important design choices were inevitably made, which cannot have been based only on the available written GDM theory.

This necessity for decision making means that other hypothetical GDM implementations (for instance using *NoSQL*, *graph* databases, or spreadsheet-based models) would also require a large number of conceptual and modelling decisions to be taken as part of their design. As such, not only will their physical datamodels differ to the IEDC, but also their conceptual datamodels. The result of this is that interoperability across implementations of the GDM be unlikely. Data stored in one implementation (e.g. the IEDC) would not be easily transferrable to others; they would not be interoperable. Therefore, the hypothesis is rejected.

One way that this interoperability problem could be avoided is by reformulating the relationship such that the IEDC is *the authoritative description* of the General Data Model for SEM. In this scenario, design choices required when creating other implementations of the GDM would reference the IEDC schema to inform the necessary decisions. However, in that case, it would no longer be true that the IEDC is an implementation of the GDM. Rather, a complete description of the GDM now requires reference to the IEDC implementation. This is summarised in Figure 6.1.

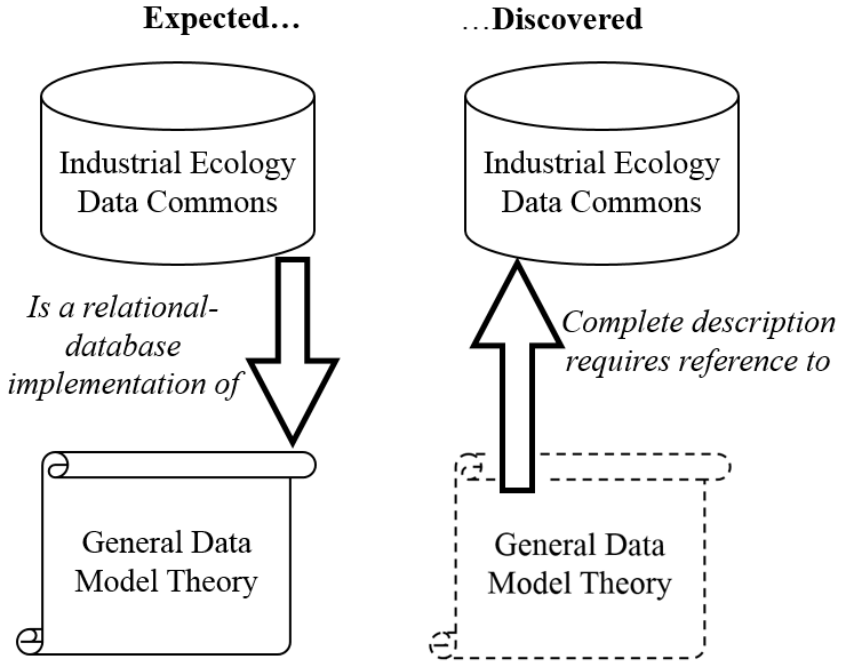

Figure 6.1: the expected and discovered relationship between the GDM and IEDC

6.2.3 Parallel descriptions of datamodels and data

This section concludes by highlighting a high-level critique from the perspective of *information architecture*. There appears to be a conflation of two distinct concepts in both the GDM and IEDC: *datamodel repositories* and *datamodels.* As introduced in section 2.1.1, datamodel repositories contain datamodels, and those datamodels can be instantiated as a single database (or schema) each. These schemas may be very large, they can reference one another and be queried together. The constructs of the datamodels are nouns, which should be visible in the table names.

To apply this idea to the organisation of the IE community: a *model repository* would host descriptions of datamodels used by systems such as ecoinvent (Wernet et al., 2016), Exiobase (Tukker, et al., 2018) the ProSUM project database (Straalen et al., 2015), amongst many other examples. An explicit attempt to build this model repository is a valuable contribution to the community (more on this in Chapter 8). Note however that this hypothetical model repository would not contain the *actual datasets* which conform to these models. The

recommended place for building this repository is on the web, using open standards, as described in Chapter 8.

As previously highlighted, *data types* are used in the IEDC to describe what are usually called *datamodels*. Each datamodel is entered as one row in an IEDC table. Examples pre-loaded include "flow", "population", "material composition" and "unit process inventory". All of the actual data items which are instances of these data types/models are also entered into the IEDC, in the table *Data*. Although the models and the data exist at a different level in any conceptual hierarchy, they are stored at the same level within the IEDC, within tables. This argument is summarised in Figure 6.2.

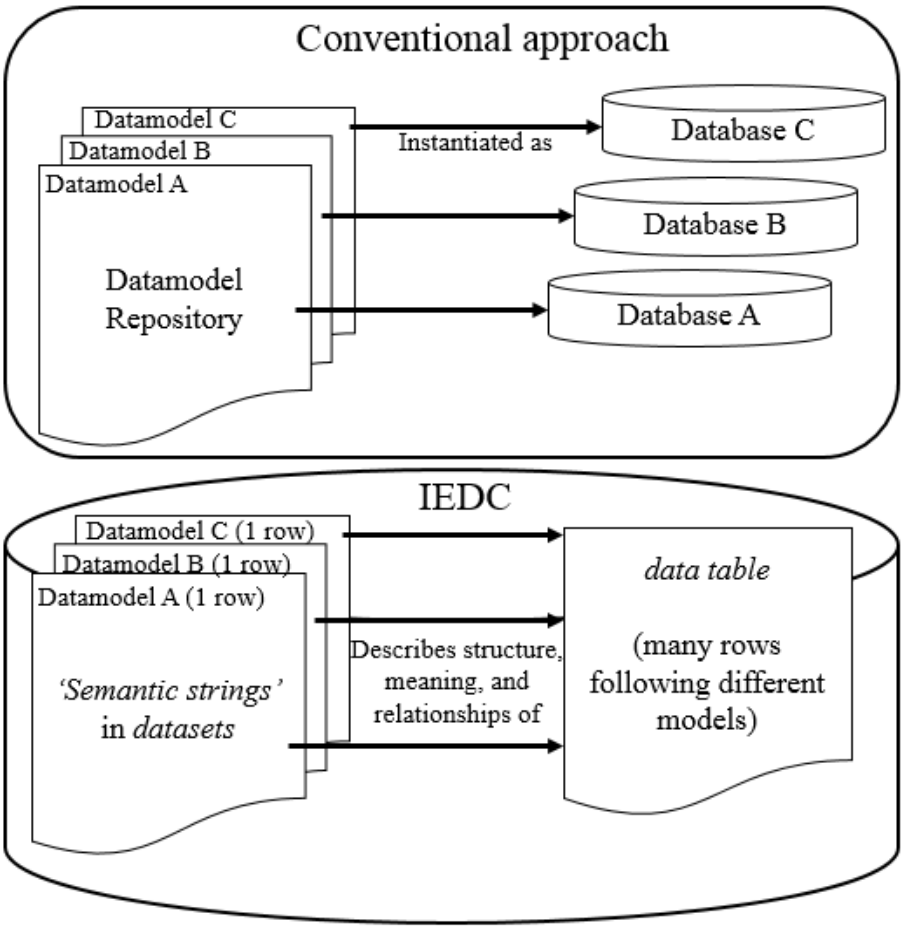

Figure 6.2: the conventional separation of datamodel repositories (for storing datamodels) and databases (for storing data), contrasted with the approach of the IEDC

To summarise: the IEDC and GDM are attempting to perform two functions in parallel which are usually kept separate within the information architecture of an organisation:

- a datamodel repository (where multiple interrelated datamodels are formally described), and
- a database for storing actual data (which usually follows only one datamodel, but in this case is an attempt to be 'general', and represent many distinct models simultaneously)

The choice to define datamodels within the database itself, enables the deferral of the actual data modelling to a later time. Specifically, when datasets are inserted. This is acknowledged within Pauliuk et al. (forthcoming), which states that [the datamodel] "allows scholars from different modeling communities to enter a dialogue and reach consensus about the aspect structure and the semantics of the different data types that are used across methods". This *deferral of modelling* is permitted in other data storage paradigms such as NoSQL, or graph databases. In these, the addition of data continuously defines the structure, so it does not need to be known in advance, and query languages (such as SPARQL or GraphQL) are designed with this expectation. This type of approach is also highlighted in the Outlook of Chapter 8.

But relational SQL databases are typically 'structure-first', rather than 'data-first'. The structure of the tables and relationships are pre-defined before they are loaded with data (Chmieliauskas et al., 2012). SQL stands for *Structured* Query Language: writing SQL queries usually requires advanced knowledge of the expected data structures. This is solved in the IEDC with the flexible *aspects* and *classification* concepts. Generic queries return all aspects for the *data* records in each *dataset*. In this case, it is not possible to apply data integrity constraints within the database design. Important data fields will always accept arbitrary *strings*, rather than (for instance) a pre-determined list of valid values. It becomes the responsibility of a user submitting data to ensure no *human-errors* are made. Considering the complexity of transformations that may be necessary to insert data into the database; this is a risky condition, which may lead to future data-quality problems.

6.3 Results in context

This research aims to independently assess the capabilities of the GDM and IEDC. Other literature available on this topic is limited to one scientific paper (currently in-review), plus the associated GitHub repository where development is ongoing. The results of this research provide original insight into the designs of these systems, and are the first to test the ability of the systems to represent unit-process data. The discussion offers a high-level critique of both the GDM and IEDC from multiple perspectives.

It is important to note that the IEDC is a prototype system. Its intended use is for demonstrating and testing the approach, and to function as a community data repository only if there is support and uptake. The design is completely transparent, due to the open and accessible working practises of the developers. This openness has enabled a level of detailed analysis which would not have been possible with a closed or proprietary system. Hence iterative and reflective learning can occur quickly for: the actors testing the system (e.g. this research); those that developed it; and other observers of the ongoing developments. This recognition supports the general approach of *open science* which is a key part of the context of the GDM and IEDC (section 1.2.2).

Chapter 7 Conclusions

This concluding chapter follows from the results and discussion. It begins with short answers to the research questions. Next, the scientific contribution of the research is summarised. The chapter ends with a summary of how this research contributes to the field of Industrial Ecology, and society more generally.

7.1 Answers to research questions

1. *Which datamodels are used in the IEDC and GDM for SEM, and what is their relationship?*

The datamodel of the IEDC is explicitly described in its tables, columns, and their relationships. This information is complemented by the pre-loaded data contents, which provides essential insight into the intended usage of the model. In contrast, the text-based description is of the GDM is limited. It is unclear whether this description is sufficient for the GDM to actually classify as a datamodel. Hypothetical alternative implementations of the GDM are unlikely to be interoperable with the IEDC implementation. Therefore, the actual relationship between the GDM and IEDC is unclear.

2. *Which datamodel is archetypal for unit-processes, and which data following this model can be used to evaluate the IEDC?*

Unit-process data consists of the following seven items: data describing the processes; data describing the commodities or flows; relationship or network data for the association of processes and flows; quantified numerical data and the associated uncertainty data; system context and location specification data; and other metadata describing the data collection and processing. The *minimum consensus knowledge model* ontology contains all of these

elements and may be considered archetypal for unit-process data. The *semantic catalogs of LCA data* follow this model, and were used in this research to evaluate the IEDC.

3. *How can the selected source datasets be mapped and transformed into the datamodel of the IEDC?*

Data sources can be mapped and transformed into the IEDC datamodel via the following stages. First, map the high-level concepts of the source entities and relationships onto the approximately equivalent concepts in the destination. This step necessarily introduces some subjectivity. Next, map the attributes of the source in detail onto the columns of the IEDC tables. Finally, implement these mappings as an algorithm which extracts the source dataset, transforms it, and loads the data into the IEDC database.

Main question: Is it possible to effectively represent unit-process data in the 'general data model for SEM' and/or the IEDC and what are the challenges and limitations?

It is possible to represent unit process data within the IEDC, but not within the GDM. The IEDC database has been designed for flexibility, and many different sources can be stored within its structures, including unit-process data. In contrast, the GDM is described only in natural language, and requires more explicit elucidation using standardised data modelling and communication techniques, such as entity-relationship diagrams.

There are a number of limitations to the IEDC representation of the unit process data. Many of the important relationships which are present in the source model are lost; the data is not stored efficiently; data integrity is not enforced via explicitly defined rules; and the data stored in the tables would be difficult to query due to the flexible meanings and purposes of the tables, columns, and foreign key relationships. These challenges collectively lead to the conclusion that the unit-process representation within the IEDC prototype is not *effective*, using the meaning established in section 1.4.2.

7.2 Unexpected findings

Two main findings, unrelated to the research questions, were discovered through the process of the research. As presented in section 6.2.2, it was found that the GDM is probably not described sufficiently clearly and explicitly to allow for the development of alternative interoperable implementations using technologies other than relational databases. Based on this finding, this research concludes that the GDM requires further formalisation and development in order to function effectively as a commonly shared conceptualisation of SEM for the IE community.

Secondly, section 6.2.3 argues that the IEDC acts simultaneously as both a *datamodel repository* and a *SQL database* for storing the data which conforms to those models. This permits the actual data modelling process to be deferred to the data-loading phase. Because of this, practically no automatic data validation is possible, and the data quality is dependent only on the humans entering the data. For this reason, some architectural redesign is recommended, before the IEDC is ready to function effectively as a datamodel repository and/or community database.

7.3 Scientific contribution

Prior to this research, the GDM for SEM and IEDC had received no direct independent evaluation. The ongoing peer-review process at the Journal of Industrial Ecology will determine whether the submitted manuscript is suitable for publication. In contrast, this work has tested specifically the capability of the systems to effectively represent unit-process data, which is a fundamental category of data within IE. Their capabilities in this regard have been demonstrated in practise. High-level novel critiques have been presented and assessed, based on ideas from information architecture and data modelling & transformation theory.

7.4 Field and societal contribution

This research has contributed a number of results and insights to the field of IE. Industrial ecologists are often data modellers by necessity, they frequently develop data sources in new structures and formats as part of their wider modelling efforts. Chapter 2 is a concise overview of basic data modelling and transformation theory, which could help with the development of high-quality models. This is a novel contribution to the field.

To the developers of the IEDC and others working on community data infrastructure for the support of open science, this work has contributed many suggestions for design improvements which could be applied to the next iteration of related work. The early timing of this contribution is critical: it is much more difficult to change datamodels once they are integrated into a wider sociotechnical system. Source code and documentation have also been made available, enabling others to validate or refute many of the claims of this report. This code may also be adapted for their own purposes and projects. The upcoming Outlook section also contributes a suggestion and argumentation for the future work on data/information architecture within the field.

These contributions may translate into contributions to society via the following steps. The results and recommendations of this work could lead to improvements in the next generation of IE data modelling and infrastructure. This would enhance data management and availability, enabling high-quality analysis and case studies to be conducted in less time. These studies lead to scientific knowledge about interrelated economic and environmental systems, enabling well-informed decision making. Society benefits via the improved outcomes of these decisions.

Chapter 8 Outlook: a 'community data commons' infrastructure native to the web

The Industrial Ecology Data Commons is a contribution toward the *open data* area of the broader movement toward *open science*. Within this area, the IEDC and GDM aim to address the "data integration and exchange problem". The IEDC developers state that "a general data format for the industrial system does not exist" and there is a "lack of cross-method data formats and platforms for exchanging IE data" (Pauliuk et al., forthcoming).

This final chapter follows on from the discussion and conclusions to presents an alternative approach which could be used to satisfy the data infrastructure requirements of the IE community. An alternative perspective is presented: appropriate *formats* and *platforms* do exist, but they are not yet used extensively by the IE community. The *semantic catalogs of LCA data* (section 4.3) are one example of this alternative approach in practise. The platform is often described broadly as the *semantic web*, and the formats are the collaboratively developed standard *linked data* formats which are associated it. The datamodels for this platform are stored as *schemas* and *ontologies*.

The semantic web technology stack or *paradigm* can be framed as analogous to a relational database [14]. Figure 8.1 provides an overview of the concepts and technologies which are generally included under the term *semantic web*. This section will not describe these technologies[15]. Rather, it highlights the existing research in IE and related fields that are using this alternative approach, alongside argumentation for why this is desirable in light of the open-data ambitions of the field.

---

[14] Berners-Lee (1998) states that "the mapping is very direct – a *record* is an RDF **node**; the *field (column) name* is RDF **propertyType**; and the *record field (table cell)* is a **value**."
[15] See Antoniou & Van Harmelen (2008) for a technology-focused primer

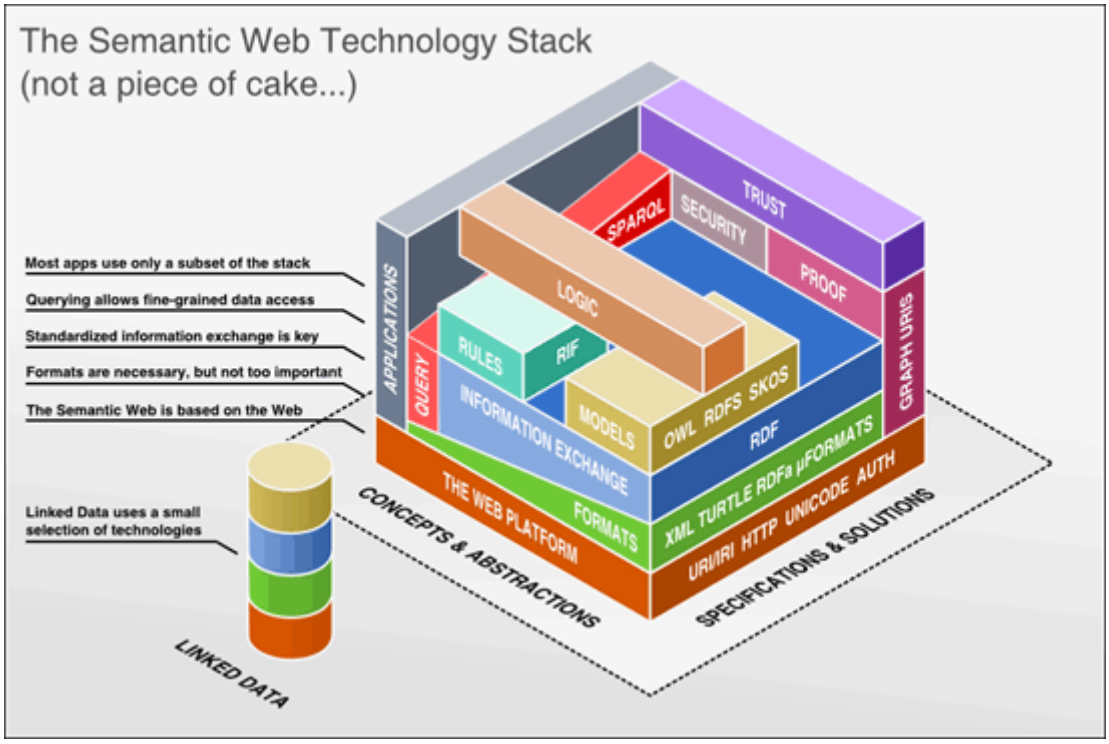

Figure 8.1: isometric diagram of the semantic web technology stack, emphasising concepts, specifications, and linked data. From Nowack (2009, creative commons license). json-ld is more recent, and hence missing from the formats.

This outlook section consists of three subsections. Various publications across industrial ecology and socioeconomic metabolism research have previously identified this infrastructure platform, including demonstrations and proofs-of-concept which begin to demonstrate the potential. These are summarised in the first subsection. The focus then sharpens onto the LCA community, which is increasingly embracing the semantic web and related technologies. Appendix N additionally offers an assessment which contrasts this proposed alternative with the IEDC approach, in terms of the "FAIR Guiding Principles for scientific data management and stewardship" from section 1.2.2. The Outlook concludes with a short summary.

8.1 IE Literature calling for a semantic web approach

The earliest publications arguing in favour of this direction for IE were by Kraines et al. (2005). They highlighted ways that new internet technologies could be effectively used to share and integrate the knowledge and models of researchers. Davis et al. (2010) then described their vision for "Industrial Ecology 2.0", with updated information and argumentation. They described the potential of linked open data on the web, and why this was particularly important for Industrial Ecology research, due to its interdisciplinary nature. Complementary work also described the semantic web toolset and example projects, such as the Enipedia collaborative Wiki which is focused on energy and industry (Davis et al., 2012; Davis, 2012). Within the *urban metabolism* research line, Ravalde & Keirstead (2017) argued for the development of an information ecosystem to enable holistic sociometabolic assessments, using the semantic web to "link together structured and unstructured information in a vast network, through which researchers can navigate, finding innovative and as yet unknown applications for the data". Toward this goal, they developed and released a free and open process-oriented database. Whilst this subset of examples from across Industrial Ecology are somewhat dispersed, a more sustained institutional approach appears to be ongoing in Lifecycle Assessment.

8.2 LCA on the semantic web

The semantic web data architecture for LCA was argued for in detail by Ingwersen et al. (2015). This paper also uses the term "data commons" to describe their vision. This vision is focused on improving interoperability and automation for LCA case studies, and the associated data management. Various ontologies and schemas were also published around this time (Ciroth & Srocka, 2015; Janowicz et al., 2015; Kuczenski et al., 2016; Yan et al., 2015). Case studies using these ontologies have followed, demonstrating the approaches in applications to the environmental performance of buildings (Schwartz et al., 2016) and chemical manufacturing (Cashman et al., 2016). The *Bonsai* project is also using the Resource Description Format (RDF) and other semantic web standards as the building blocks

of their planned “Big Open Network for Sustainability Assessment Information” (Bonsai.uno, 2019).

An LCA ‘capability roadmap’ has also recently been published by a working group as part of the UNEP-SETAC Life Cycle Initiative (Kuczenski et al., 2018). They describe the technical advances required to improve LCA transparency and replicability, focusing on the three areas of model *contents, structure*, and *collaboration*. Their proposed approach for LCA *product system models* involves a shift of focus “from text and tables in a report to the model as a digital object itself”. These digital objects would be referenced via web-based uniform resource identifiers (URIs), and interoperable with other semantic web-based linked data. The lead author of this paper is actively working on a Python-based open-source toolset to enable the capabilities detailed in this vision (GitHub, 2019a).

8.3 Outlook summary

The IE and SEM communities are rightly enthusiastic for the potential of open science. Open data is a fundamental component of this approach which is best supported by appropriate decisions regarding the technical infrastructure, platforms and data formats. This Outlook has highlighted a related collection of potential solutions to this, grouped under the label *semantic web*. This infrastructural approach will ensure that the essential data resources of the IE community are managed in-line with best-practises.

If others agree with this vision: in order to benefit from a *network effect*, it is essential that researchers begin to engage with this approach. We can begin by using the data already hosted on these platforms, and citing their work where appropriate. New datasets should reuse existing schemas and ontologies wherever possible. When pre-existing work is insufficient, those existing datamodels should be amended and extended, alongside transparent annotations and reasoning for why the existing options were not already sufficient. An updated Industrial Ecology Data Commons which embraces this approach would involve mapping existing data sources to common web-based standards, formats and models. This would fully satisfy the FAIR requirements, and provide great value to the research community..

References

Anex, R., & Lifset, R. (2014). Life Cycle Assessment. *Journal of Industrial Ecology*, *18*(3), 321–323. https://doi.org/10.1111/jiec.12157

Antoniou, G., & Van Harmelen, F. (2008). *A semantic web primer, second edition*. MIT press.

Archive.org. (2019). Internet Archive: Wayback Machine. Retrieved from https://archive.org/web/

Benyus, J. M. (1997). Biomimicry: Innovation inspired by nature. Morrow New York.

Berners-Lee, T. (1998). Relational Databases on the Semantic Web (1998). *URL Http://Www. W3. Org/DesignIssues/RDB-RDF. Html*.

Bollinger, L. A., Nikoli??, I., Davis, C. B., & Dijkema, G. P. J. (2015). Multimodel Ecologies: Cultivating Model Ecosystems in Industrial Ecology. *Journal of Industrial Ecology*, *19*(2), 252–263. https://doi.org/10.1111/jiec.12253

Cashman, S. A., Meyer, D. E., Edelen, A. N., Ingwersen, W. W., Abraham, J. P., Barrett, W. M., … Smith, R. L. (2016). Mining available data from the United States environmental protection agency to support rapid life cycle inventory modeling of chemical manufacturing. *Environmental Science and Technology*, *50*(17), 9013–9025. https://doi.org/10.1021/acs.est.6b02160

Castells, M. (1996). *The Rise of the Network Society. The Information Age: Economy, Society, and Culture Volume I (Information Age Series)*. London: Blackwell.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, *1*(1), 9–36.

Chmieliauskas, A., Chappin, E. J. L., Davis, C. B., Nikolić, I., & Dijkema, G. P. J. (2012). New Methods for Analysis of Systems-of-Systems and Policy: The Power of Systems Theory, Crowd Sourcing and Data Management. *System of Systems*, 77–98. https://doi.org/10.5772/1406

Ciroth, A. (2007). ICT for environment in life cycle applications openLCA - A new open source software for Life Cycle Assessment. *International Journal of Life Cycle Assessment*, *12*(4), 209–210. https://doi.org/10.1065/lca2007.06.337

Ciroth, A., & Srocka, M. (2015). JSON-LD : A smarter format for LCA data interchange. *Ilcm 2015*, (September).

Codd, E. F. (1990). *The Relational Model for Database Management : Version 2*. *Database*. Retrieved from http://books.google.co.kr/books?q=9780201141924

Date, C. J. (2006). *An introduction to database systems*. Pearson Education India.

Davis, C. B. (2012). Making Sense of Open Data: From Raw Data to Actionable Insight, 249. https://doi.org/10.4233/uuid:88c3c6f9-d6a2-4a82-9353-884a3b77b6ed

Davis, C., Nikolic, I., & Dijkema, G. (2012). *Harnessing semantic web technologies for the environmental sustainability of production systems*.

Davis, C., Nikolic, I., & Dijkema, G. P. J. (2010). Industrial ecology 2.0. *Journal of Industrial Ecology*, *14*(5), 707–726. https://doi.org/10.1111/j.1530-9290.2010.00281.x

Dittmar, P. G., Stobaugh, R. E., & Watson, C. E. (1976). The chemical abstracts service chemical registry system. I. General design. *Journal of Chemical Information and Computer Sciences*, *16*(2), 111–121.

Docs.ocelot.space. (2017). Ocelot — Ocelot 0.1 documentation. Retrieved from https://docs.ocelot.space/index.html

Docs, Python (2019). 5. Data Structures — Python 3.7.2 documentation. Retrieved from https://docs.python.org/3/tutorial/datastructures.html#dictionaries

ecoinvent.com. (2019a). ecoinvent. Retrieved from https://www.ecoinvent.org/

ecoinvent.com. (2019b). History – ecoinvent. Retrieved from https://www.ecoinvent.org/about/history/history.html

ecoinvent.com. (2019c). How to Submit Data – ecoinvent. Retrieved from https://www.ecoinvent.org/data-provider/how-to-submit-data/how-to-submit-data.html

ecoinvent.com. (2019d). System Models in ecoinvent 3. Retrieved from https://www.ecoinvent.org/database/system-models-in-ecoinvent-3/system-models-in-ecoinvent-3.html

Ehrenfeld, J. (2003). Putting a Spotlight on Metaphors and Analogies in Industrial Ecology. *Journal of Industrial Ecology*, *7*(1), 1–4. https://doi.org/10.1162/108819803766729131

Fischer-Kowalski, M., & Hüttler, W. (1999). Society's metabolism: the intellectual history of material flow analysis, Part II, 1970-1998. *Journal of Industrial Ecology*, *2*(4), 107–136. https://doi.org/10.1162/jiec.1998.2.1.61

Fischer-kowalski, M., & Weisz, H. (1999). Society as hybrid between material and symbolic realms : Toward a theoretical framework of society-nature interaction, (January 2017).

Freiburg. (2019). IEDC Database. Retrieved from http://www.database.industrialecology.uni-freiburg.de/

Frischknecht, R. (2006). Special Issue Honouring Helias A . Udo de Haes : LCA Methodology Notions on the Design and Use of an Ideal Regional or Global LCA Database, *1*(1), 40–48.

GitHub. (2019a). LCA data tools repository. Retrieved from https://github.com/bkuczenski/lca-tools

GitHub. (2019b). Semantic catalog download repository. Retrieved from https://github.com/bkuczenski/lca-tools-datafiles

GitHub. (2019c). Semantic catalog example workbooks. Retrieved from https://github.com/bkuczenski/lca-tools-datafiles/tree/gh-pages/doc

Golubiewski, N. (2012). Is There a Metabolism of an Urban Ecosystem ? An Ecological Critique, 751–764. https://doi.org/10.1007/s13280-011-0232-7

Guinée, J. B., Heijungs, R., Huppes, G., Kleijn, R., de Koning, A., van Oers, L., … Gorrée, M. (2002). life cycle assessment. Operational guide to the ISO standards. I: LCA in perspective. IIa: Guide. IIb: Operational annex. III: Scientific background. *The Netherlands: Ministry of ...*, 692. https://doi.org/10.1007/BF02978784

Guinee, J., Heijungs, R., Huppes, G., Zamagni, A., & Masoni, P. (2011). Life Cycle

Assessment : Past, Present, and Future. *Environmental Science & Technology*, *45*(1), 90–96. https://doi.org/10.1021/es101316v

Haberl, H., Fischer-kowalski, M., & Krausmann, F. (2004). Progress towards sustainability ? What the conceptual framework of material and energy flow accounting (MEFA) can offer, *21*, 199–213. https://doi.org/10.1016/j.landusepol.2003.10.013

Haq, Q. S. U. (2016). *Data Mapping for Data Warehouse Design*. https://doi.org/10.1016/B978-0-12-805185-6.00018-6

Harth, A., Umbrich, J., & Decker, S. (2006). Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *International Semantic Web Conference* (pp. 258–271).

Heijungs, R. (1997). *Economic drama and the environmental stage : formal derivation of algorithmic tools for environmental analysis and decision-support from a unified epistemological principle*. *PhD Thesis*. https://doi.org/10.1007/BF02978414

Henriksson, P. J. G., Guinée, J. B., Heijungs, R., De Koning, A., & Green, D. M. (2014). A protocol for horizontal averaging of unit process data - Including estimates for uncertainty. *International Journal of Life Cycle Assessment*, *19*(2), 429–436. https://doi.org/10.1007/s11367-013-0647-4

Herrmann, I. T., & Moltesen, A. (2015). Does it matter which Life Cycle Assessment (LCA) tool you choose? - A comparative assessment of SimaPro and GaBi. *Journal of Cleaner Production*, *86*, 163–169. https://doi.org/10.1016/j.jclepro.2014.08.004

Hertwich, E., Heeren, N., Kuczenski, B., Majeau-Bettez, G., Myers, R. J., Pauliuk, S., … Lifset, R. (2018). Nullius in Verba1. *Journal of Industrial Ecology*, *00*(0). https://doi.org/10.1111/jiec.12738

Hougen, O. A., Watson, K. M., & Ragatz, R. A. (1954). *Chemical Process Principles: Material and Energy Balances* (Vol. 1). John Wiley & Sons.

Imbeault-Tétreault, H., Jolliet, O., Deschênes, L., & Rosenbaum, R. K. (2013). Analytical propagation of uncertainty in life cycle assessment using matrix formulation. *Journal of Industrial Ecology*, *17*(4), 485–492. https://doi.org/10.1111/jiec.12001

Ingwersen, W. W., Hawkins, T. R., Transue, T. R., Meyer, D. E., Moore, G., Kahn, E., … Norris, G. A. (2015). A new data architecture for advancing life cycle assessment. *International Journal of Life Cycle Assessment*, *20*(4), 520–526. https://doi.org/10.1007/s11367-015-0850-6

International Organization for Standardization. (2006). ISO 14040-Environmental management - Life Cycle Assessment - Principles and Framework. *International Organization for Standardization*, *3*, 20. https://doi.org/10.1016/j.ecolind.2011.01.007

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit, Third Edition*.

Kraines, S., Batres, R., Kemper, B., Koyama, M., & Wolowski, V. (2006). Internet-based integrated environmental assessment, part II: Semantic searching based on ontologies and agent systems for knowledge discovery. *Journal of Industrial Ecology*, *10*(4), 37–60.

Kraines, S., Batres, R., Koyama, M., Wallace, D., & Komiyama, H. (2005). Internet-based integrated environmental assessment using ontologies to share computational models. *Journal of Industrial Ecology*, *9*(3), 31–50.

Kuczenski, B. (2015). Partial ordering of life cycle inventory databases. *International Journal of Life Cycle Assessment*, *20*(12), 1673–1683. https://doi.org/10.1007/s11367-015-0972-x

Kuczenski, B., Davis, C. B., Rivela, B., & Janowicz, K. (2016). Semantic catalogs for life cycle assessment data. *Journal of Cleaner Production*, *137*, 1109–1117. https://doi.org/10.1016/j.jclepro.2016.07.216

Kuczenski, B., Marvuglia, A., Astudillo, M. F., Ingwersen, W. W., Satterfield, M. B., Evers, D. P., … Laurin, L. (2018). LCA capability roadmap: product system model description and revision. *The International Journal of Life Cycle Assessment*, 1685–1692. https://doi.org/10.1007/s11367-018-1446-8

Lloyd, S. M., & Ries, R. (2007). Characterizing, propagating, and analyzing uncertainty in life-cycle assessment. *Journal of Industrial Ecology*, *11*(1), 161–181. https://doi.org/10.1162/jiec.2007.1136

Luján-Mora, S., Vassiliadis, P., Trujillo, J., Lujan-Mora, S., Vassiliadis, P., & Trujillo, J. (2004). Data mapping diagrams for data warehouse design with UML. *Conceptual Modeling - ER 2004, 23rd International Conference on Conceptual Modeling*, 191–204. https://doi.org/10.1007/978-3-540-30464-7_16

Meinshausen, I., Müller-Beilschmidt, P., & Viere, T. (2016). The EcoSpold 2 format—why a new format? *International Journal of Life Cycle Assessment*, *21*(9), 1231–1235. https://doi.org/10.1007/s11367-014-0789-z

Mendoza Beltran, A., Prado, V., Font Vivanco, D., Henriksson, P. J. G., Guinée, J. B., & Heijungs, R. (2018). Quantified Uncertainties in Comparative Life Cycle Assessment: What Can Be Concluded? *Environmental Science and Technology*, *52*(4), 2152–2161. https://doi.org/10.1021/acs.est.7b06365

MerriamWebster. (2019a). “Effective”: definition. Retrieved from https://www.merriam-webster.com/dictionary/effective

MerriamWebster. (2019b). “Represent”: definition. Retrieved from https://www.merriam-webster.com/dictionary/represent

Müller, D. B., Lundhaug, M., & Simoni, M. (2018). MinFuture: D2.2 Challenges, systems and data.

Mutel, C. (2017). Brightway: An open source framework for Life Cycle Assessment. *The Journal of Open Source Software*, *2*(12), 11–12. https://doi.org/10.21105/joss.00236

National Institute of Science and Technology. (2004). data structure. Retrieved from https://xlinux.nist.gov/dads/HTML/datastructur.html

Nexus.openlca.org. (2019). openLCA Nexus: The source for LCA data sets. Retrieved from https://nexus.openlca.org/

Oppel, A. J. (2010). *Data Modeling: A Beginner’s Guide*. McGraw-Hill.

Pauliuk, S., Heeren, N., Hasan, M. M., & Daniel B. Muller. (n.d.). A General Data Model for Socioeconomic Metabolism and its Implementation in an Industrial Ecology Data Commons Prototype. *In Review for the Journal of Industrial Ecology*.

Pauliuk, S., & Hertwich, E. G. (2015). Socioeconomic metabolism as paradigm for studying the biophysical basis of human societies. *Ecological Economics*, *119*, 83–93. https://doi.org/10.1016/j.ecolecon.2015.08.012

Pauliuk, S., Majeau-Bettez, G., Müller, D. B., & Hertwich, E. G. (2016). Toward a Practical Ontology for Socioeconomic Metabolism. *Journal of Industrial Ecology*, *20*(6), 1260–1272. https://doi.org/10.1111/jiec.12386

Pauliuk, S., Majeau-Bettez, G., Mutel, C. L., Steubing, B., & Stadler, K. (2015). Lifting Industrial Ecology Modeling to a New Level of Quality and Transparency: A Call for More Transparent Publications and a Collaborative Open Source Software Framework. *Journal of Industrial Ecology*, *19*(6), 937–949. https://doi.org/10.1111/jiec.12316

Ravalde, T., & Keirstead, J. (2017). A Database to Facilitate a Process-Oriented Approach to Urban Metabolism. *Journal of Industrial Ecology*, *21*(2), 282–293. https://doi.org/10.1111/jiec.12429

Regulations, E. (2018). *Faculty of Science , Leiden University and Faculty of Technology , Policy and Management , Delft University of Technology 1 September 2017 to 31 August 2018 Course and Examination Regulations Master ' s Programme Industrial Ecology*.

Schandl, H., Müller, D. B., & Moriguchi, Y. (2015). Socioeconomic metabolism takes the stage in the international environmental policy debate: A special issue to review research progress and policy impacts. *Journal of Industrial Ecology*, *19*(5), 689–694. https://doi.org/10.1111/jiec.12357

Schmachtenberg, M., Bizer, C., & Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8796*, 245–260. https://doi.org/10.1007/978-3-319-11964-9

Schwartz, Y., Eleftheriadis, S., Raslan, R., & Mumovic, D. (2016). Semantically Enriched BIM Life Cycle Assessment to Enhance Buildings' Environmental Performance. *CIBSE Technical Symposium, Edinburgh, UK*, (April), 14 pages.

Seto, K. E., Panesar, D. K., & Churchill, C. J. (2017). Criteria for the evaluation of life cycle assessment software packages and life cycle inventory data with application to concrete. *The International Journal of Life Cycle Assessment*, *22*(5), 694–706.

Silverston, L., & Agnew, P. (2009). *The data model resource book: Universal patterns for data modeling*. John Wiley & Sons.

Simsion, G., & Witt, G. (2004). *Data modeling essentials*. Elsevier.

Skoutas, D., & Simitsis, A. (2006). Designing ETL processes using semantic web technologies. *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP - DOLAP '06*, 67. https://doi.org/10.1145/1183512.1183526

Stefan PAULIUK, Niko HEEREN, Mohammad Mahadi HASAN, D. B. M. (2019). A General Data Model for Socioeconomic Metabolism and its Implementation in an Industrial Ecology Data Commons Prototype, 1–30.

Steffen, W., Richardson, K., Rockström, J., Cornell, S. E., Fetzer, I., Bennett, E. M., … others. (2015). Planetary boundaries: Guiding human development on a changing planet. *Science*, *347*(6223), 1259855.

Straalen, V. van, Huisman, J., Habib, H., Chancerel, P., Maehlitz, P., Rotter, S., … Cassard, D. (2015). ProSUM Project - Review and harmonisation of data - Deliverable 5.3, 1–99.

Theodoulidis, C., Wangler, B., & Loucopoulos, P. (1976). The entity relationship time model.

Conceptual Modelling, Databases, and CASE: An Integrated View of Information Systems Development, 87–115.

Tsichritzis, D. C., & Lochovsky, F. H. (1982). *Data models*. Prentice Hall Professional Technical Reference.

Tukker, A., Wood, R., & Giljum, S. (2018). Relevance of Global Multi Regional Input Output Databases for Global Environmental Policy: Experiences with EXIOBASE 3. *Journal of Industrial Ecology*, *22*(3), 482–484. https://doi.org/10.1111/jiec.12767

Tupper, C. (2011). *Data architecture: from zen to reality*. Elsevier.

Varela, F. G., Maturana, H. R., & Uribe, R. (1974). Autopoiesis: The organization of living systems, its characterization and a model. *BioSystems*, *5*(4), 187–196. https://doi.org/10.1016/0303-2647(74)90031-8

Verelst, J. (2004). Variability in conceptual modeling. *University of Antwerp*.

W3C-Prov. (2019). PROV-Overview. Retrieved from https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

W3C. (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). Retrieved from https://www.w3.org/TR/owl2-overview/

W3C. (2019a). JSON-LD 1.0. Retrieved from https://www.w3.org/TR/json-ld/

W3C. (2019b). W3C JSON-LD Working Group. Retrieved from https://www.w3.org/2018/json-ld-wg/

Wassenaar, T. (2015). Reconsidering industrial metabolism: From analogy to denoting actuality. *Journal of Industrial Ecology*, *19*(5), 715–727. https://doi.org/10.1111/jiec.12349

Weidema, B. P. (2009). Avoiding or ignoring uncertainty. *Journal of Industrial Ecology*, *13*(3), 354–356. https://doi.org/10.1111/j.1530-9290.2009.00132.x

Weidema, B. P., Frees, N., & Nielsen, A. M. (1999). Marginal production technologies for life cycle inventories. *International Journal of Life Cycle Assessment*, *4*(1), 48–56. https://doi.org/10.1007/BF02979395

Wernet, G., Bauer, C., Steubing, B., Reinhard, J., Moreno-Ruiz, E., & Weidema, B. (2016). The ecoinvent database version 3 (part I): overview and methodology. *International Journal of Life Cycle Assessment*, *21*(9), 1218–1230. https://doi.org/10.1007/s11367-016-1087-8

Wilkinson, M. D. (2016). Comment : The FAIR Guiding Principles for scienti fi c data management and stewardship, 1–9.

Wolf, M.-A., Düpmeier, C., & Kusche, O. (2011). The International Reference Life Cycle Data System (ILCD) Format - Basic Concepts and Implementation of Life Cycle Impact Assessment (LCIA) Method Data Sets. *EnviroInfo 2011: Innovations in Sharing Environmental Observations and Information*, (Ilcd). Retrieved from http://enviroinfo.eu/sites/default/files/pdfs/vol7233/0809.pdf

Yan, B., Hu, Y., Kuczenski, B., Janowicz, K., Ballatore, A., Krisnadhi, A. A., … Ingwersen, W. (2015). An ontology for specifying spatiotemporal scopes in life cycle assessment. *CEUR Workshop Proceedings*, *1501*(Lci), 25–30.

Zuiderwijk, A., Janssen, M., & Davis, C. (2014). Innovation with open data: Essential elements of open data ecosystems. *Information Polity*, *19*(1–2), 17–33.

Appendix A Data-mapping document

A.1 Auxiliary Mappings

Mapping ID	Order Level	Target table name	Target column name	Data Type	Key	Nullable	Use existing IEDC records?	Value	Comment
0	0	categories	id	int(11)	PRI	NO	YES	1	None of the 6 categories are a particularly accurate description for the
	0	categories	name	varchar(255)	UNI	NO	YES	Flow	necessary 'unit process' data-type (see section on categories in the report
	0	categories	description	varchar(255)		NO	YES	Objects flowing between processes	for more detail). However in the existing IEDC records for Type='Process
1	0	dimensions	id	int(11)	PRI	NO	YES	6, 7	Existing dimension values in IEDC are sufficient to describe the source
	0	dimensions	name	varchar(255)	UNI	NO	YES	Commodity, process	dataset. Different dimensions will be referenced by specific Aspects and
	0	dimensions	description	varchar(255)		YES	YES	[product, good, commodity] / [process or activ	Classification Definitions: see those entries for detail.
2	0	layers	id	int(11)	PRI	NO	YES	1	The catalog (unit process) datasets describe a number of layers depending on the unit used for each characterization. Any **non-reference** characterization entry which has a mass-based reference flow, must be a mass-ratio of some sort. However the IEDC data model forces requires
	0	layers	name	varchar(255)	UNI	NO	YES	Mass	that a dataset describes only 1 layer, due to a foreign key constraint on
	0	layers	description	varchar(255)		NO	YES		this column. Hence we choose the simplest: mass.
3	0	licences	id	int(11)	PRI	NO	YES	4	Semantic Catalog publication states: "This is an open access article under
	0	licences	name	varchar(255)	UNI	NO	YES	CC BY 4.0	the CC BY license".
	0	licences	description	varchar(255)		NO	YES		
4	0	provenance	id	int(11)	PRI	NO	YES	8	The provenance table entries are not mutually exclusive, and many could
	0	provenance	name	varchar(255)	UNI	NO	YES	Result of academic LCA work	feasibly apply to our dataset. A new entry could be used to be more
	0	provenance	description	varchar(255)		NO	YES	data derived from life cycle assessment	specific. But ideally data provenance would be modelled in a more
5	0	source_type	id	int(11)	PRI	NO	YES	5	
	0	source_type	name	varchar(255)	UNI	NO	YES	Publicly available report or research article	
	0	source_type	description	varchar(255)		NO	YES		
6	0	stats_array	id	int(11)	PRI	NO	YES	14	The semantic catalog data unfortunately does not include uncertainty
	0	stats_array	name	varchar(255)		NO	YES	undefined	information with the numerical values. Although the source LCA datasets
	0	stats_array	description	varchar(255)		NO	YES	moved from 0 as 0 is not allowed in mySQL db.	on which it is based often will include this.
	0	stats_array	loc	varchar(255)		YES	YES	static value	
7	0	users	id	int(11)	PRI	NO	NO	8	
	0	users	username	varchar(255)	UNI	NO	NO	[YourUsername]	
	0	users	name	varchar(255)		NO	NO	[YourName]	
	0	users	institution	varchar(255)		YES	NO	[YourInstitution]	
	0	users	start_date	datetime		NO	NO	2018-10-29	
	0	users	end_date	datetime		YES	NO	2050-06-09	
8	1	types	id	int(11)	PRI	NO	YES	18	
	1	types	name	varchar(255)	UNI	NO	YES	Process inventory	
	1	types	description	varchar(255)		NO	YES	flows entering and leaving process, for LCI datasets	
	1	types	reference_data_ca	int(11)	MUL	NO	YES	1	
	1	types	symbol	varchar(5)	UNI	NO	YES	PI	
9	2	projects	id	int(11)	PRI	NO	NO	3	3 = new project entry
	2	projects	project_name	varchar(255)	UNI	NO	NO	LCA Semantic_Catalog representations	
	2	projects	data_categories	varchar(255)		YES	NO	Various	These 3 VARCHAR columns without FK constraints allow any values, and are not constrainted to the [categories, types and layers] tables which they refer to. To allow multiple entries (as implied by the pluralised column names), this should be modelled with a junction-table for many-to-many relationships between projects and these tables.
	2	projects	data_types	varchar(255)		YES	NO	Various	
	2	projects	data_layers	varchar(255)		YES	NO	Various	
	2	projects	process_scope	varchar(255)		YES	NO	Various	
	2	projects	process_resolution	varchar(255)		YES	NO	Various	
	2	projects	product_scope	varchar(255)		YES	NO	Various	
	2	projects	product_resolution	varchar(255)		YES	NO	Various	
	2	projects	material_scope	varchar(255)		YES	NO	Various	
	2	projects	material_resolutio	varchar(255)		YES	NO	Various	
	2	projects	regional_scope	varchar(255)		YES	NO	Various	
	2	projects	regional_resolution	varchar(255)		YES	NO	Various	
	2	projects	temporal_scope	varchar(255)		YES	NO	Various	
	2	projects	temporal_resolutic	varchar(255)		YES	NO	Various	
	2	projects	description	text		YES	NO	The project which provided this data is described in Kuczenski et al. 2016. This was then modified and imported as part of	
	2	projects	keywords	varchar(255)		NO	NO	LCA, Semantic catalogs, knowledge model, ETL	
	2	projects	type_of_source	int(11)	MUL	YES	NO	5	
	2	projects	project_license	int(11)	MUL	YES	NO	4	
	2	projects	project_link	text		YES	NO	GitHub	
	2	projects	submission_date	datetime		NO	NO	NOW()	
	2	projects	submitting_user	int(11)	MUL	NO	NO	8	

10	3	datagroups	id	int(11)	PRI	NO	NO	11	11 = new data group entry
	3	datagroups	datagroup_name	varchar(255)	UNI	NO	NO	EI32_Und_SemanticCatalog	Different for each LCA semantic catalog which may be imported (future work)
	3	datagroups	datagroup_version	varchar(30)		YES	NO	0.1	
	3	datagroups	project_id	int(11)	MUL	YES	NO	3	
	3	datagroups	data_categories	varchar(255)		YES	NO	Various	Duplication of information contained in project table described above. It is unclear which entries for [categories, types and layers] should take precedence in the event that data conflicts arise between the various tables that can store this information.
	3	datagroups	data_types	varchar(255)		YES	NO	Various	
	3	datagroups	data_layers	varchar(255)		YES	NO	Various	
	3	datagroups	process_scope	varchar(255)		YES	NO	Various	
	3	datagroups	process_resolution	varchar(255)		YES	NO	Various	
	3	datagroups	product_scope	varchar(255)		YES	NO	Various	
	3	datagroups	product_resolution	varchar(255)		YES	NO	Various	
	3	datagroups	material_scope	varchar(255)		YES	NO	Various	
	3	datagroups	material_resolutio	varchar(255)		YES	NO	Various	
	3	datagroups	regional_scope	varchar(255)		YES	NO	Various	
	3	datagroups	regional_resolutio	varchar(255)		YES	NO	Various	
	3	datagroups	temporal_scope	varchar(255)		YES	NO	Various	
	3	datagroups	temporal_resolutic	varchar(255)		YES	NO	Various	
	3	datagroups	description	text		YES	NO	EcoInvent 3.2 undefined as represented in the Semantic Catalogs	
	3	datagroups	keywords	varchar(255)		NO	NO	EcoInvent	
	3	datagroups	type_of_source	int(11)	MUL	YES	NO	5	As above - potential for data integrity conflicts with Projects table.
	3	datagroups	project_license	int(11)	MUL	YES	NO	4	
	3	datagroups	submission_date	datetime		NO	NO	now()	
	3	datagroups	submitting_user	int(11)	MUL	NO	NO	8	

A.2 Core Mappings

Mappin g	Order Level	Target table name	Target column name	Data Type	Key	Nullabl	Entity name	alue / attribute nam	Transformation category	Transformation rule / Comment
11	0	units	id	int(11)	PRI	NO				
	0	units	unitcode	varchar(30	UNI	NO	Flow Quantity	referenceUnit	Transformation	Select all distinct units in the source dataset and insert into destination table. If value already exists, do not perform the insert.
	0	units	unit_name	varchar(10	UNI	NO	Flow Quantity	Name	Transformation	A unit_name is not available in the source data but this is a required field. There are three options: simply duplicate the unit_code as listed here; manually enter the names of the units; or write a look-up function to automatically enter the unit name. The latter 2 options require additional data/knowledge, possibly from the original source datasets on which the catalogs are based.
	0	units	alt_unit_name	varchar(100)		YES				
	0	units	factor	double		YES				
12	1	aspects	id	int(11)	PRI	NO				This is the first of 3 separate entries for Aspects, Classification_definitions, and the items of these classifications.
	1	aspects	aspect	varchar(25	UNI	NO		"Commodity_Flow"	Hardcoded	The aspects are referenced by all exchanges entered into the Data table.
	1	aspects	description	varchar(255)		NO		"The commidity whi	Hardcoded	
	1	aspects	dimension	int(11)	MUL	NO		5	Hardcoded	5=material
	1	aspects	index_letter	varchar(1)	UNI	NO		"0" (zero)	Hardcoded	
	1	aspects	index_letter_crib	varchar(255)		NO		"c0mm0dity fl0w2	Hardcoded	
13	1	classification_definition	id	int(11)	PRI	NO				The flattened source data entered into the Data table have 'Aspect1' reference this definition
	1	classification_definition	classification_name	varchar(25	UNI	NO		"EcoInvent_Commo	Hardcoded	
	1	classification_definition	dimension	int(11)	MUL	NO		6	Hardcoded	6=commodity
	1	classification_definition	description	text		YES		"All distinct flows defined in the Ecoinvent LCA semantic catalog are listed under under th		
	1	classification_definition	mutually_exclusive	tinyint(1)		NO		0	Hardcoded	
	1	classification_definition	collectively_exhaustive	tinyint(1)		NO		0	Hardcoded	
	1	classification_definition	general	tinyint(1)		NO		0	Hardcoded	
	1	classification_definition	created_from_dataset	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	meaning_attribute1	varchar(255)		NO		"EntityId"	Hardcoded	Note these match with mapping 14 below
	1	classification_definition	meaning_attribute2	varchar(255)		YES		"Flow Name"	Hardcoded	
	1	classification_definition	meaning_attribute3	varchar(255)		YES		"CAS Number"	Hardcoded	
	1	classification_definition	meaning_attribute4	varchar(255)		YES		"Compartment"	Hardcoded	
14	2	classification_items	id	int(11)	PRI	NO				These items relate to the classification system defined in mapping 13
	2	classification_items	classification_id	int(11)	MUL	NO	classification_d	"EcoInvent_Commo	Lookup	
	2	classification_items	parent_id	int(11)	MUL	YES		NULL	Hardcoded	No hierarchies defined on this classification
	2	classification_items	attribute1_oto	varchar(255)		NO	Flow	EntityId	Direct	
	2	classification_items	attribute2_oto	text		YES	Flow	Name	Direct	
	2	classification_items	attribute3_oto	varchar(255)		YES	Flow	CasNumber	Direct	
	2	classification_items	attribute4_oto	varchar(255)		YES	Flow	Compartment	Direct	

15	1	aspects	id	int(11)	PRI	NO				
	1	aspects	aspect	varchar(25	UNI	NO		"Boolean"	Hardcoded	
	1	aspects	description	varchar(255)		NO		"Any aspect which ca	Hardcoded	
	1	aspects	dimension	int(11)	MUL	NO		13	Hardcoded	13=custom
	1	aspects	index_letter	varchar(1)	UNI	NO		"A"	Hardcoded	
	1	aspects	index_letter_crib	varchar(255)		NO		"booleAn"	Hardcoded	
16	1	classification_definition	id	int(11)	PRI	NO				The flattened source data entered into the Data table have their 'Aspect3' and 'Aspect4' specified as this classification, which specifies if the exchanges and characterizations are the reference values
	1	classification_definition	classification_name	varchar(25	UNI	NO		"Boolean"	Hardcoded	
	1	classification_definition	dimension	int(11)	MUL	NO		13	Hardcoded	13=custom
	1	classification_definition	description	text		YES		Boolean: true or false		
	1	classification_definition	mutually_exclusive	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	collectively_exhaustive	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	general	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	created_from_dataset	tinyint(1)		NO		0	Hardcoded	
	1	classification_definition	meaning_attribute1	varchar(255)		NO		"Boolean value"	Hardcoded	
17	2	classification_items	id	int(11)	PRI	NO				
	2	classification_items	classification_id	int(11)	MUL	NO	classification_d	"Boolean"	Lookup	
	2	classification_items	attribute1_oto	varchar(255)		NO		TRUE	Hardcoded	
	2	classification_items	id	int(11)	PRI	NO				
	2	classification_items	classification_id	int(11)	MUL	NO		"Boolean"	Lookup	
	2	classification_items	attribute1_oto	varchar(255)		NO		FALSE	Hardcoded	
18	1	aspects	id	int(11)	PRI	NO				
	1	aspects	aspect	varchar(25	UNI	NO		"Direction"	Hardcoded	
	1	aspects	description	varchar(255)		NO		"The direction with v	Hardcoded	
	1	aspects	dimension	int(11)	MUL	NO		5	Hardcoded	
	1	aspects	index_letter	varchar(1)	UNI	NO		"D"	Hardcoded	
	1	aspects	index_letter_crib	varchar(255)		NO		"Direction"	Hardcoded	
19	1	classification_definition	id	int(11)	PRI	NO				The flattened source data entered into the Data table have their 'Aspect2' specified as this classification, which specifies whether the exchange is an input or output of the process dataset.
	1	classification_definition	classification_name	varchar(25	UNI	NO		"Direction"	Hardcoded	
	1	classification_definition	dimension	int(11)	MUL	NO		13	Hardcoded	
	1	classification_definition	description	text		YES		"The direction with which a flow relates to a process. Options are 'Input' and 'Output'"		
	1	classification_definition	mutually_exclusive	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	collectively_exhaustive	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	general	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	created_from_dataset	tinyint(1)		NO		1	Hardcoded	
	1	classification_definition	meaning_attribute1	varchar(255)		NO		"Direction value"	Hardcoded	
20	2	classification_items	id	int(11)	PRI	NO				
	2	classification_items	classification_id	int(11)	MUL	NO	classification_d	"Direction"	Lookup	
	2	classification_items	attribute1_oto	varchar(255)		NO		"Input"	Hardcoded	
	2	classification_items	id	int(11)	PRI	NO				
	2	classification_items	classification_id	int(11)	MUL	NO	classification_d	"Direction"	Lookup	
	2	classification_items	attribute1_oto	varchar(255)		NO		"Output"	Hardcoded	
21	4	datasets	id	int(11)	PRI	NO				
	4	datasets	dataset_name	varchar(25	MUL	NO	Process	Name	Direct	
	4	datasets	dataset_version	varchar(30)		YES		0.1	Hardcoded	Option to increment version number as multiple imports are attempted, if required
	4	datasets	datagroup_id	int(11)	MUL	YES	datagroups	"EI32_Und_Semantic	Lookup	Change this if importing different semantic catalog datasets
	4	datasets	data_category	int(11)	MUL	NO	categories	1	Lookup	See comment in categories entry
	4	datasets	data_type	int(11)	MUL	NO	types	Process inventory	Lookup	
	4	datasets	data_layer	int(11)	MUL	NO	layers	Mass	Lookup	See comment in layers entry
	4	datasets	process_scope	varchar(255)		YES	Process	ISIC Number	Direct	These varchar fields for scope and resolution data have limited documentation and a flexible string datatype. This provides limited guidance on the data they should contain and it's format.
	4	datasets	process_resolution	varchar(255)		YES				
	4	datasets	product_scope	varchar(255)		YES	Process	Classifications	Direct	Not the ideal place for process classifications from the catalogs. But they must go somewhere.
	4	datasets	product_resolution	varchar(255)		YES				
	4	datasets	material_scope	varchar(255)		YES				
	4	datasets	material_resolution	varchar(255)		YES				
	4	datasets	regional_scope	varchar(255)		YES	Process	SpatialScope	Direct	
	4	datasets	regional_resolution	varchar(255)		YES				
	4	datasets	temporal_scope	varchar(255)		YES	Process	TemporalScope	Direct	
	4	datasets	temporal_resolution	varchar(255)		YES				
	4	datasets	description	text		YES	Process	"Technology Level:",	Direct	
	4	datasets	keywords	varchar(255)		NO	Process	externalId	Direct	Use to store ID String
	4	datasets	data_provenance	int(11)	MUL	NO	provenance	8	Lookup	8=Result of academic lca work
	4	datasets	dataset_size	int(11)		NO				
	4	datasets	comment	text		YES	Process	Comment	Direct	
	4	datasets	aspect_1	int(11)	MUL	NO	aspects	"Commodity_Flow"	Lookup	Although in Datasets, these aspect ID entries and classifications are actually for the entries in the *data* table.
	4	datasets	aspect_1_classification	int(11)	MUL	NO	classification_d	"EcoInvent_Commo	Lookup	
	4	datasets	aspect_2	int(11)	MUL	YES	aspects	"Direction"	Lookup	
	4	datasets	aspect_2_classification	int(11)	MUL	YES	classification_d	"Direction"	Lookup	
	4	datasets	aspect_3	int(11)	MUL	YES	aspects	"isReferenceExchang	Lookup	
	4	datasets	aspect_3_classification	int(11)	MUL	YES	classification_d	"Boolean value"	Lookup	
	4	datasets	aspect_4	int(11)	MUL	YES	aspects	"isReferenceCharact	Lookup	
	4	datasets	aspect_4_classification	int(11)	MUL	YES	classification_d	"Boolean value"	Lookup	
	4	datasets	semantic_string_exam	text		NO				No semantic strings created in this mapping
	4	datasets	semantic_string_gener	text		NO				due to critique of this approach in Chapter 3
	4	datasets	type_of_source	int(11)	MUL	NO	source_types	"Publicly available da	Lookup	Source type is defined at three levels: Datagroup, dataset and project. These values may clash and no rules are defined for establishing precedence in such a case.

21 (continued)		4	datasets	project_license	int(11)	MUL	NO	licences	"unknown/not speci	Lookup	Although the semantic catalogs have a CC4.0 licence, it may be the case that the source datasets on which those catalogs were based (e.g. the publicly available EcoInvent data) has a different licence which should take precedence
		4	datasets	main_author	varchar(255)		NO		"Brandon Kuczenski"	Hardcoded	Using the main author of the semantic catalogs for this field
		4	datasets	visible	tinyint(1)		NO		1	Hardcoded	Yes
		4	datasets	submission_date	datetime		NO		Now()	Hardcoded	Current time function in SQL
		4	datasets	submitting_user	int(11)	MUL	NO	users	[YourUsername]	Lookup	
22		5	data	id	int(11)	PRI	NO				
		5	data	dataset_id	int(11)	MUL	NO	dataset		Lookup	Select the process dataset ID, for which this data entry is an exchange
		5	data	aspect1	int(11)	MUL	NO	Flow	Name	Lookup	Lookup the ID of the classification_item with the name equal to the flow name of the source.
		5	data	aspect2	int(11)	MUL	YES	Exchange	Direction	Lookup	Lookup the ID of the classification_item with the direction equal to the direction of the source exchange (in or out)
		5	data	aspect3	int(11)	MUL	YES	Exchange	isReference	Lookup	Lookup the ID of the classification_item for the Boolean value matching the True/False of the source
		5	data	aspect4	int(11)	MUL	YES	Characterizatior	isReference	Lookup	as above
		5	data	value	double		YES			Transformation	exchange value * characterization value
		5	data	unit_nominator	int(11)	MUL	NO	units		Lookup	Select the ID of the unit entered in the referenceUnit field of the Flow Quantity in the source dataset
		5	data	unit_denominator	int(11)	MUL	YES		1	Hardcoded	ID 1 = "Unity" - means that the numerator is divided by 1 and undergoes no transformation
		5	data	stats_array_1	int(11)	MUL	YES		14	Hardcoded	See comment in stats_array entry
		5	data	comment	text		YES			Transformation	Include both Comment strings from the source data. Format: "Flow Comment: ", [Flow > Comment], "\nFlow Quantity Comment: ", [Flow Quantity > Comment]

Appendix B Semantic catalog glossary of terms

Many of these definitions are taken from the original paper (Kuczenski et al. 2016) describing the catalogs and minimally edited for clarity.

B.1 Classes

from a knowledge modelling perspective, the notions of “flow”, “activity”, and “flow quantity” are classes – or abstract concepts. The data structure used for instances or entities are derived from the class on which they are based

1. Activity

a thing that happens to Flows, to transform inputs into outputs. This base class is instantiated as a process. Activities can be classified by their reference *exchange*.

2. Flow

a class of "thing in the world" that exists because of some instance of an Activity. Observable phenomena that are the accounting elements of industrial ecology. (A *commodity* in Heijungs 1997) A Flow has a direction with respect to an Activity: it is an output of one Activity and an input to another. Flows can be classified by their reference quantity (often "mass")

3. (Flow) quantity

a distinct quantitative characteristic that can be ascribed to a Flow. The fore-term *flow* is often omitted, leading to simply: *Quantities. Q*uantities be classified by their reference unit of measure (often "kg").

B.2 Instances

a particular flow, activity, or quantity is called an entity or an instance of a class. An LCA practitioner or data set developer creates a model by making observations of specific instances. Exchanges and characterizations describe relationships among instances, not classes

4. Process (instance)

a particular instance of an activity that occurred at a particular place and time, and whose characteristics were observed and recorded. Only a process has an "inventory"; and only a process can have quantified exchange values. Based upon the ISO 14040 & ISO 9000 definition. This is the only instance whose name differs significantly from the class on which it is based.

5. Flow instance

See *Flow*, but for *Process instances*, rather than their *Activity* base class

6. (flow) Quantity instance

See *Flow quantities*, but for *Flow instances rather than their* Flow *base class*

B.3 Relationships between instances[16]

In the semantic catalogs data model, there are no base classes defined for these entities which describe the relationships between the *process, flow* and *quantity* instances listed above.

7. Exchange

an established directional relationship between a *process* and a *flow* instance. One of the exchanges is labelled as the reference exchange.

8. Characterization

an established relationship between a flow and a flow quantity. Flows typically have many characterizations. For instance, the flow of "gasoline" has mass, volume, economic value, toxicity potential, energy content, and others. A characterization has an implicit or explicit spatiotemporal scope, which corresponds to the activity that generates or consumes the flow

B.4 Attributes

9. Spatiotemporal Scope

[16] Writing at https://github.com/bkuczenski/lca-tools, the author of the catalogs states: "These should probably be entities, too, because of serialization + linked data more than anything. I mean, they are entities in the LD schema. But they don't have stand-alone identities (in LD terms they are 'blank nodes'); their origins are defined implicitly by the entities they belong to and so they don't have external references; and they contain numerical data, as opposed to other entities that strictly contain qualitative data. So I think there is a sound conceptual reason for them to be their own object types rather than subclasses of LcEntity."

the contexts in which inventory information or impact estimates are valid. SpatiotemporalScope is associated with the classes Place and Time through the relations occursAtPlace and occursAtTime

10. Space

some, typically named, extent in geographic space, e.g., a country or region (spatial dimension).

11. Time

The time, usually a congruent series of calendar years (temporal dimension). Time should be an interval because an LCA activity generally represents the performance of a typical facility or a set of facilities over a time period, rather than at a specific moment

12. Exchange value

the magnitude of one flow exchanged through a process, in comparison to a unit amount of a reference exchange

13. Exchange Direction

Direction which specifies the relationship between a process and flow instance: Input or Output

14. Compartment

medium that contains a flow. E.g. Air, water, soil. Compartment is not described in detail in the paper.

15. Characterization factor/value

One of the *characterization* entries may be marked as the reference quantity for the *flow* it characterises ("isReference" = True). The characterization factor or value quantifies the magnitude of one *flow quantity* in comparison to a unit amount of the reference quantity (hence the reference flow has factor=1). These characterizations include any life cycle impact characterization factors which are defined for the flow

16. Unit of measure

An extensive magnitude of a quantity which can be used to define or describe a flow. Usually adopted by social convention, E.g. kg (for mass), m^3 (volume), or CO_2eq (global warming potential)

Appendix C Relationship between IE and SEM

This thesis frequently references *Industrial Ecology* and *Socioeconomic Metabolism*. It is worthwhile to discuss briefly the relationship between these two related domains. Firstly, both of the terms are biological metaphors (Ehrenfeld, 2003). The *ecology* of IE and the *metabolism* of SEM are direct references to concepts from biology which have been studied extensively. The general motivation behind the use of these metaphors is that nature has evolved the ability to renew and sustain itself as an autopoietic system (Varela et al., 1974), and humans can learn from this in order to redesign the biophysical basis of society, such that it can also be sustained. This could broadly be referred to as biomimicry (Benyus, 1997). The ecological metaphor evokes notions like populations, community, resilience, evolution, food webs and trophic cycles. Whereas the metabolic metaphor encourages thinking in terms of food, energy, chemical reactions, pathways and regulation. Concepts such as dynamic equilibrium and autopoiesis are central to both domains, and all of these concepts can provide at least some theoretical starting points and inspiration for the analysis of industry and society. These metaphors are limited however, and they should not be mistaken for *analogies*, which may lead to logical mistakes and the derivation of invalid conclusions (Ehrenfeld, 2003).

Further conceptual challenges stems from the fact that neither IE or SEM are limited by the metaphors on which their names were derived. Clarity on their relationship can be attained by

considering the actual research practises of the communities that adopt these names. Research in Industrial Ecology often consists of case studies that are conducted through the application of one or many *algorithmic tools and/or methodologies*, such as Lifecyle Assessment (LCA), Environmental Input-Output Analysis (EIOA), Material/Substance Flow Analysis (MFA/SFA). These tools may be referred to as the tools of IE, and their object of study is SEM (Pauliuk & Hertwich, 2015). The official regulations of the IE MSc. program at Leiden & Delft states that "graduates will… have a thorough knowledge of **the field of Industrial Ecology and its object society's metabolism**" (Leiden University & TU Delft, 2018). Thus, for this work, the relationship between IE and SEM is defined as: i*ndustrial ecology studies socioeconomic metabolism.*

Appendix D Instructions for connecting to the Freiburg IEDC in MySQL

This appendix describes how a new user uses the MySQL Workbench GUI software to connect to the Freiburg instance of the IEDC database. Users require a login and password to connect via this route.

1. Download MySQL installer https://dev.mysql.com/downloads/installer/
2. Install MySQL Server (for hosting local instance of database)
3. Install MySQL Workbench (as GUI)
4. Create saved database connection for Freiburg instance

   Hostname: www.industrialecology.uni-freiburg.de

   Port: 3306

   Username: iedc_guest*** [If you require a new login, Mahadi or Stefan at Freiburg IndEcol group may be able to assist]

Password: *****"

5. Check Freiburg connection works and that you're able to query data etc. (If not, troubleshoot)

Appendix E Cloning the IEDC to create your own instance

These instructions follow on from the above, and assume you have a working connection to the Freiburg instance. Note that MySQL uses the terms *database* & *schema* interchangeably, unlike many other vendors.

6. Ensure MySQL80 local database service is running. You may need to do this each time you want to use the db. In Windows:
    a. run services.msc
    b. scroll down to MySQL80 (80 is version number, yours may be different)
    c. Right click > Start
7. Create new saved database connection for local instance

    Hostname: localhost
    Port: 3306
    Username: YourRootOrAdminUser
    Password: *****

Migrate (clone schema and data) Freiburg database to local instance

Option 1: MySQL Migration Wizard (Workbench Menu > Database > Migration wizard)

- For me, most tables copied without incident.
- However, for an unknown reason, the Units and Aspects tables gave errors on the automatic data transfer when I did this (Tom)
    - I exported data from those tables to CSV and imported to local instance
- After this, I had a complete instance of the IEDC on to my local machine, ready for safe testing and experimentation etc.

Option 2: Mysqldump (data export) to file on local disk, then import data

- The second time I required an update of the source database, I used an alternative approach which did not run into the above errors. First export all data as a logical SQL dump file. Then import that data into your installed local instance.

2.1 Data Export/dump using MySQLWorkbench wizard

1. Connect to Freiburg db instance in workbench
2. Workbench menu > Server > Data Export
3. Select all (table) objects in IEDC schema (there are no Views, Functions, SPs, Events or Triggers at time of writing)
4. Select "Dump structure and data"
5. Export to self-contained file (choose a location on your pc)
6. Tick "...single transaction" and "Include create schema"
7. To start: Select other tab: "Export progress". The start button is in the bottom right of the interface.

The wizard generated this script, which could be run again through the command line after editing for your environment.

*mysqldump.exe --defaults-file="c:\users***\appdata\local\temp\tmpapaxcr.cnf" --user=iedc_guest_*** --host=www.industrialecology.uni-freiburg.de --protocol=tcp --port=3306 --default-character-set=utf8 --single-transaction=TRUE --skip-triggers "iedc"*

2.2 Create local db with full data import

- Note that this import took surprisingly long at maximum disk and CPU. The dump file was only 80MB, yet the import took 26 mins! It eventually completed without errors however.

8. Connect to local db server instance through GUI
9. Workbench menu > Server > Data Import
10. Select “Import from self-contained file”
11. Enter file location from the export of 2.1 above
12. To start: Select other tab: "Export progress". The start button is in the bottom right of the interface.

command line script:

*mysqldump.exe --defaults-file="c:\users***\appdata\local\temp\tmpapaxcr.cnf" --user=iedc_guest_*** --host=www.industrialecology.uni-freiburg.de --protocol=tcp --port=3306 --default-character-set=utf8 --single-transaction=TRUE --skip-triggers "iedc"*

If successful, you now have a complete and up-to-date copy of the IEDC running on your local machine.

The general syntax of the script required for extracting the schema of a MySQL database is simply

mysqldump -d -u <username> -p<password> -h <hostname> <dbname>

Option 3 (without a Freiburg IEDC login account): contact or check the GitHub repo of the author of this thesis, to gain access to a MySQL Dump file as described above.

Appendix F Query: column definitions

```
SELECT table_name, column_name, column_type, column_key, is_nullable
FROM INFORMATION_SCHEMA.COLUMNS
WHERE  table_schema = 'iedc'
AND NOT column_name like 'reserve%';
```

Appendix G Query: table-pairs with multiple foreign-key relationships

```
SELECT TABLE_NAME,group_concat(COLUMN_NAME) as "FK_Columns",
REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME, count(constraint_name)
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE REFERENCED_TABLE_SCHEMA = 'iedc'
GROUP BY TABLE_NAME, REFERENCED_TABLE_NAME
HAVING count(constraint_name) > 2
ORDER BY count(constraint_name) ASC;
```

Appendix H Query: columns for storing numerical values

/*select all column data types within the IEDC*/

```
SELECT distinct column_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'iedc';
```

/*select all columns which store numerical data, based on the total list of numerical types in the above… A manual check of column names where column_type = "INT" confirms that these are storing keys, not quantitative data*/

```
SELECT table_name, column_name, column_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'iedc'
AND column_type in ('double', 'float') #,'int';
```

This query returns the five numerical columns shown in Table 3.3 in the main text. Three of these five results are for storage of statistical uncertainty data. Robust and precise modelling efforts have the requirement that numerical data should be associated with parameter uncertainty wherever possible (Lloyd & Ries, 2007). Uncertainty representation is handled in the IEDC through description of the types of statistical distributions (such as uniform, lognormal and Bernoulli) in the *stats_array* table. This table is based upon the Python package of the same name which was originally developed for the Brightway2 LCA software (Mutel, 2017). A foreign-key relationship from the *Data* table column *stats_array1* defines which of these distribution types apply. The *data* columns *stats_array*[*2,3* or *4*] are used as parameters to quantify the uncertainty under each of the distributions. The names, meanings or definitions of these parameters are different, depending on the distribution selected for each *data* item.

The possible units-of-measure for each quantified data item are stored in the table *units*. Names, codes and descriptions for the many types of units which are used to consistently measure dimensions in IE (and other sciences) are included in this table. Units which describe the same physical dimension can be converted between one another. For instance, kilograms and pounds are both measures of mass. A numerical conversion factor can be stored in this table for this purpose, alongside the reference unit which this factor scales to. The *data* records can specify two units: the numerator and denominator. This allows for the construction of further compound units beyond those which are stored in the *Units* table. However, in most cases, the *denominator_unit* is "1" – such that the *numerator_unit* is simply used.

Appendix I Query: dataset-category and dataset-type-category conflict

```
SELECT Dataset_name, t.name as '"Types" Name' ,c1.name as 'Dataset "Category"',
c2.name as 'Type "Category"'
FROM iedc.datasets d
INNER JOIN types t ON d.data_Type = t.id
INNER JOIN categories c1 ON d.data_category = c1.id
INNER JOIN categories c2 ON t.reference_data_category = c2.id
WHERE d.data_category != t.reference_data_category ;
```

Query result:

Dataset_name	"Types" Name	Dataset "Category"	Type "Category"
2_IUS_Global_Materials_2050 Scenario_Wiedenhofer_2019	Flow	Stock	Flow
1_F_Global_Materials_2050 Scenario_Wiedenhofer_2019	Flow	Stock	Flow
1_F_Global_End_of_life_waste _Wiedenhofer_2019	Flow	Stock	Flow

Note that the dataset category is different to the flow category in each of these cases. Therefore, there is no way to determine which category is correct for these datasets.

Appendix J Aspects and dimensions data in the IEDC

The table below shows all the aspects and dimensions stored in the pro-populated IEDC data. It is intended to demonstrate that it is somewhat unclear how aspects and dimensions are related. Sometimes the Aspect name is the same as the Dimension name, but often not. The colour highlights show common dimensions for different aspects, and an anomalous row: a single dimension that has no aspects associated with it. The dimension is “layer”, which is also a table name and hence a concept modelled elsewhere in the IEDC.

The table was generated by outer-joining the aspect and dimension tables. As in the query:

```
SELECT  `a`.`aspect` AS `asp_name`,  `a`.`description` AS `asp_desc`,  `d`.`name` AS `dim_name`,  `d`.`description` AS `dim_desc`
FROM  (`dimensions` `d`  LEFT JOIN `aspects` `a` ON ((`d`.`id` = `a`.`dimension`)))
```

ORDER BY `d`.`name`;

Aspect name	Aspect description	Dimension name	Dimension description
commodity	Goods and products considered	commodity	product, good, commodity
EoL_good	End-of-life products, buildings, and infrastructure	commodity	product, good, commodity
product_type	Types of products	commodity	product, good, commodity
element	chemical elements	element	chemical elements
energy_carrier	Energy carrier	energy	energy type or carrier
extension	Costs, emissions factors, social impacts	extensions	units other than mass or energy
NULL	NULL	layer	layer of quantification
engineering_material	Engineering materials considered, subset of generic materials M	material	resource, material, engineering material
waste_scrap	waste and scrap types considered	material	resource, material, engineering material
input_material	Input of material to process	material	resource, material, engineering material
output_material	Output of material to process	material	resource, material, engineering material
process	Process where stock is located	process	process or activity
region	Region of process or stock	region	country, region, or place
origin_region	Region of origin (flow)	region	country, region, or place
destination_region	region of destination (flow)	region	country, region, or place
scenario	Scenerios considered (e.g., SSP)	scenario	scenario, alternative, version
service	Service categories: shelter, transport, etc.	services	service flows
time	Model time	time	physical time
age-cohort	age-cohorts	time	physical time
unity	trivial classification, 1 entry only	unity	one, 1

Appendix K Storage and interchange formats for LCA: why json-LD?

This section focuses rather on the storage and transmission of the data, in *files* which come in a variety of *formats*, as described in section 2.1.1. In a publication aimed at providing guidance to background database developers, Frischknecht (2006) states that the ideal LCA database "should communicate in a common data (exchange) format, offer centralised naming lists, be based on unit-processes". Ciroth & Srocka (2015) agree, arguing that "For LCA, data exchange is important, and therefore agreement on LCA data exchange formats is important". However the reality does not match this ideal state, as described in *Kuczenski et al. (2016*), LCA datasets "are presented with highly heterogeneous formats, interfaces, and distribution mechanisms. The lack of agreement among data providers for descriptions of processes and flows creates substantial barriers for information sharing and reuse of practitioners' models".

Numerous file format standards (in either the XML or JSON languages) have entered widespread usage for unit-process data. For instance:

XML-Based:

- **EcoSpold2** (Meinshausen et al., 2016), as developed initially for the ecoinvent database (Wernet et al., 2016)
- **ILCD** (Wolf et al., 2011)

JSON-Based:

- The native data structures used in the LCA software **Brightway2** (Mutel, 2017)
- **json-ld** (Ciroth & Srocka, 2015)

This final item on the list is used in this research. A brief description of the background and functionality of the format is therefore provided here. The json-ld LCA data format is intended to improve interoperability in LCA. It is developed by Ciroth & Srocka (2015), who also use this format now for their OpenLCA software (Ciroth, 2007) and the associated

online LCA dataset searching tool (nexus.openlca.org, 2019). Under the realisation that due to modern collaborative working practises “LCA data formats… are always *data exchange* formats”, they adopted modern *semantic-web resource description* technologies to minimise the friction in data exchange. They argue that "semantic technologies and ontologies are promising methods to support interoperability in LCA. They foster semantic interoperability without the need to enforce a *single* domain schema". The schema for their format is available at http://greendelta.github.io/olca-schema/.

Json-ld is short for JavaScript Object Notation–Linked Data. It is a World Wide Web Consortium (W3C) open standard, which extends conventional JSON to serialize linked data for web-based environments (W3C, 2019a). Linked data or semantic-web relationships consist of logical “triples” with the structure: subject-predicate-object. For an explanation of how these triples can be derived from the JSON format (made up of embedded key-value pairs), the supporting information of Kuczenski et al., (2016) provides an explanation. The context of the data includes a reference to the URI of an ontology which describes the data, and also pointers/references to other objects or resources on the semantic web. The current version of the format is stable at 1.0, with a working group currently working on an update, to add new features and improve usability (W3C, 2019b).

Appendix L Python code for evaluation of ecoinvent 3.2 undefined semantic catalog

Improved versions of this code may be available in the GitHub repository of the author.

```
# -*- coding: utf-8 -*-
# Created on Mon Oct  8 21:55:16 2018
# @author: Tom
```

```
# See https://github.com/bkuczenski/lca-tools-datafiles/tree/gh-pages/doc for other code examples, some lines of which (e.g. for file-reading were used in this script)
# @title: Evaluation of the Ecoinvent 3.2 Undefined Semantic Catalog

import json
from collections import defaultdict, OrderedDict, Counter
from itertools import islice
import pandas as pd

# This code uses the unzipped version of the catalog in the json subfolder
fn_ei32_und_unzipped = '../catalogs/json/ecoinvent_3.2_undefined_xlsx.json'
# fn_ei32_und = '.../catalogs/ecoinvent_3.2_undefined_xlsx.json.gz'
with open(fn_ei32_und_unzipped) as fp:
    catalog = json.load(fp)

def get_entity_helper_objects(_list):
    """ expects a list of dicts which represents top level entities:
    processes, flows, or quantities
    returns 2 dictionaries: lists_of_values and distinct_value_counts
    both have keys: the original key names of the top level of received dicts
    distinct_value_counts returns counts of all values found inside the dicts,
    up to 2 levels deep (based on manual exploration of data types)
    """
    lists_of_values = defaultdict(list)
    # a defaultdict of defaultdicts
    distinct_value_counts = defaultdict(lambda: defaultdict(int))

    for _dict in _list:
        if isinstance(_dict, dict): #  should be True for all values received
            for tag,v1 in _dict.items():
                if tag not in ['exchanges', 'characterizations']:
                    lists_of_values[tag].append(v1)
                    if isinstance(v1, dict):
                        for k, v2 in v1.items():
                            distinct_value_counts[tag][(k,v2)] += 1 # to improve
                    elif isinstance(v1, list):
                        for item in v1:
                            if isinstance(item, dict):
                                # this is the deepest level in the objects
                                for k3, v3 in item.items():
                                    distinct_value_counts[tag][(k3,v3)] += 1
                            else:
                                distinct_value_counts[tag][item] += 1
                    else:  # strings and ints at top level handled here
                        distinct_value_counts[tag][v1] += 1
        else:
            return("unexpected value received, not a list:", _list)
    return lists_of_values, distinct_value_counts

# create pointers/aliases for the 3 primary entity types, for easy reference
processes = catalog["processes"]
flows = catalog["flows"]
quantities = catalog["quantities"]

# for all entities (processes, flows and quantities), create helper objects...
# structures that to enable simple exploration and summary of their contents
```

```
p_lists_of_values, p_distinct_value_counts = get_entity_helper_objects(processes)
f_lists_of_values, f_distinct_value_counts = get_entity_helper_objects(flows)
q_lists_of_values, q_distinct_value_counts = get_entity_helper_objects(quantities)

# Exchanges and Characterisations define the relationships between entities
# Exchanges are stored within the processes which they are inputs/outputs for
# Characterizations are stored within the flows which they characterize
exchanges = []
characterizations = []
for p in processes:
    for e in p['exchanges']:
        # assumption is that if isReference is not present, then it is not the reference exch
        exchanges.append((e.get('direction'), e.get('flow'), e.get('isReference', False), e.get('value'), p['entityId']))

for f in flows:
    for c in f['characterizations']:
        characterizations.append((c.get('direction', None), c.get('quantity', None), c.get('isReference', False),
c.get('value', None), f['entityId']))


### Explorations of the data in the catalog

# Which process names occur frequently in the catalog?
process_name_counts = Counter(p['Name'] for p in processes)
# [ print(value,"of Name:", key) for key, value in process_name_counts.most_common(5) ]

# reference exchanges
exch_references = Counter(e[2] for e in exchanges)
# print("\n Exchange directions:", exch_references)
# False: 15938, True: 14158... both reference and non-reference exchanges included here

# reference characterizations
char_references = Counter(c[2] for c in characterizations)
# print("\n Flow reference characterizations:", char_references)
# True: 6767... only reference characterizations in this database

# What is the distribution of flow-quantities referenced by characterizations
characterization_quantities = Counter(c[1] for c in characterizations)

# Edit to get names of these Q_IDs
# [ print(value,"of QuantityId:", key) for key, value in characterization_quantities.most_common(5) ]


# how many exchanges are inputs and outputs?
exch_directions = Counter(e[0] for e in exchanges)
print("\nExchange directions:", exch_directions)

# all 30096 exchanges are Outputs and none are inputs: not as expected
 # Could be due to sign convention of values in EcoInvent and bad import?
# ... e.g. outputs are positive/negative and vice-versa?
# But no numerical data included in this dataset, so can't easily confirm this idea

# EcoInvent has only one classification listed per process.
print("\n" + "{} distinct classifications across {}
processes\n".format(len(p_distinct_value_counts['Classifications']), len(processes)))
```

```
## which are most common? Order the dictionary and show top 5
ordered_classifications = OrderedDict(sorted(p_distinct_value_counts['Classifications'].items(),
                        key=lambda t: t[1], reverse=True))

def take(n, iterable):
   "Return first n items of the iterable as a list"
   return list(islice(iterable, n))
[print(value,"of", key) for key, value in take(5,ordered_classifications.items()) ]

# which ISIC codes are commonly found in the catalog
# https://siccode.com/en/pages/isic - ISIC Codes!
ordered_isics = OrderedDict(sorted(p_distinct_value_counts['IsicNumber'].items(),
                        key=lambda t: t[1], reverse=True))

[print(value,"of IsicNumber:", key) for key, value in take(5,ordered_isics.items()) ]
# We can look up these numbers. For instance the most common:
# https://pseps.com/isic/3510-electric-power-generation-transmission-and-distribution


import random
def GetFlattenedProcessData(process_ID):
   # returns three objects: 1 dictionary and 2 lists of dictionaries
   flat_data = []
   flow_data = []
   #print("\nRandomly selected ProcessID:",process_ID)
   for p in processes:
      # should match just one process
      if p['entityId'] == process_ID:
         process_data = p
         #(p['entityId'],p['Name'],p['TemporalScope'], p['SpatialScope']
         #, p['TechnologyLevel'], p['Classifications'], p['Comment'], p['exchanges'])
         # 1 to n exchanges
         for e in p['exchanges']:
            # get exchange value or use random
            e_value = round(e.get('value', random.random()), 3)
            #always points to 1 flow
            flow = e['flow']
            for f in flows:
               if f['entityId'] == flow:
                  flow_data.append(f)
                  # 1 to m characterizations
                  for c in f['characterizations']:
                     # get char value or use random
                     if c['isReference']:
                        #reference characterizations are always 1
                        c_value = 1
                     else: # randomize
                        # todo: ensure the same value is used each time the same characterization is used
                        c_value = round(c.get('value', random.random()), 3)
                     #always points to 1 quantity
                     quantity = c['quantity']
                     for q in quantities:
                        if q['entityId'] == quantity:
                           #fill up the object with data
                           flat = {}
                           flat["e_direction"] = e['direction']
```

```
                    flat["e_isref"] = e.get('isReference', False)
                    flat["e_value"] = e_value

                    flat["f_id"] = flow
                    flat["f_name"] = f['Name']
                    flat["f_comp"] = f['Compartment']
                    flat["f_cas"] = f['CasNumber']
                    flat["f_syn"] = f['Synonyms']

                    flat["c_isRef"] = c['isReference']
                    flat["c_value"] = c_value

                    flat["q_id"] = quantity
                    flat["q_name"] = q['Name']
                    flat["q_unit"] = q['referenceUnit']

                    flat["value_combined"] = round(flat["e_value"] * flat["c_value"], 4)
                    # save row and begin again
                    flat_data.append(flat)

    return process_data, flat_data, flow_data

column_order = ["e_value","e_isref","e_direction","f_id","f_name"
            ,"f_comp","f_cas","f_syn","c_isRef","c_value","q_id","q_unit"
            ,"value_combined"]

# choose whether to use a pre-determined ID (matching the report), or a random one
process_ID = 'cd6b43df-8a42-42c4-84d7-0b38b5e14e06'
#process_ID = random.choice(p_lists_of_values['entityId'])

# process data is mapped to the *datasets* table
# flat data is mapped to the *data* table
# flow data is mapped to the *classification_items* table
process_data, flat_data, flow_data = GetFlattenedProcessData(process_ID)
flat_data_table = pd.DataFrame(flat_data, columns=column_order)
```

Appendix MOverview of semantic catalog background databases

After cloning the semantic catalog repository to a local machine, the catalogs download directory shows that there are 13 zipped catalog files. Their filenames clearly indicate that

some of the catalogs include impact assessment data rather than unit-process data (section 4.2). Omitting these 3 files leaves 10 unit-process catalogs. The remaining 10 files are each given helpful short-hand names, indicating their database type and system model (where relevant). Of the 10 files, there are 4 fully original databases. The others are different versions of the same databases. This includes 5 ecoinvent versions, 2 of GaBi, 2 of USLCI, and only 1 version of ELCD. To simplify further analysis, the 'undefined' system model versions are used where a selection is required. These have less processing applied, and are more fundamental than the datasets created after application of system models (section 4.2.3). This leaves four remaining catalogs, one for each source database type.

Json-ld catalog files can be treated as the *dictionary* data structure in Python. In the dictionary of one semantic catalog, the top-level keys are: *@context, catalogNames, dataSourceType, flows, processes* and *quantities* are included in all 10 catalogs. The main data of interest for this research is included in the final 3 keys of that list: *flows*, *processes* and *quantities*. Each of these keys has a Python *list* as the associated value.

Processing the contents of these lists can provide many insights into the data quality of the remaining four catalogs. The table below shows a summary of the quantity of the main entities and relationships, as described in previous sections. It can be seen that 'GaBi-Pro' includes no characterization data due to its proprietary nature. This means this catalog is unsuitable as a source. The distribution of data types between the remaining catalogs shows some interesting patterns, which may be of interest to further research but are beyond scope here.

	EI (und)	**ELCD**	**GaBi-Pro**	**USLCI**
process	13,307	503	3,319	701
exchange	30,096	211,652	3,309	29,719
flow	6,767	2,343	762	4,176
characterization	6,767	2,342	-	5,497
quantity	718	94	10	20

The remaining source code performs a number of checks to ensure the remaining catalogs each adhere to the theoretical description of the data as presented in section 4.2.4. Each *exchange* must create a unique one-to-one relationship between a *process* and *flow,* for each

possible direction (the same exchange flow may be both an input and output of a process). And each characterization must create a unique one-to-one relationship between a *flow* and *quantity*.

Appendix N Contrasting infrastructure alternatives in terms of the FAIR principles

This section provides comment on the extent to which the FAIR principles of scientific data stewardship (Wilkinson, 2016) are satisfied by the IEDC and the proposed *semantic web* approach. To what extent do the alternative infrastructure designs enable the *open data* to be *findable, accessible, interoperable* and *reusable?* The quotations are from the IE-focused publication by Hertwich et al. (2018).

1. "findable: indexing or archiving (meta)data with unique identifiers (e.g., digital object identifiers [DOIs]) at a searchable resource;"

The data stored in the IEDC do not have unique identifiers or DOIs, whereas all top-level entities on the semantic web require this. The IEDC data can be queried directly from any instance of the database, using custom-designed SQL queries. Whereas data stored on the semantic web can be queried using SPARQL queries, or by using generic *data crawlers* (e.g. Harth et al., 2006) . IEDC data is not automatically indexed by search engines but can be accessed through a custom-built web-interface (Freiburg, 2019). Whereas semantic data on the web is indexed automatically by a variety of systems.

2. "accessible: (meta)data use an open standard for machine readability and are made permanently available."

The metadata describing data in the IEDC is stored as records within the IEDC itself. Annotations for submitted data are not automatically accessible outside of the IEDC database server environment. Semantic web metadata is stored in schemas and also alongside data in documents. Open standards are not used in the IEDC, whereas the main standards institute of the web are the same people that are strongly advocating for the approach (W3C and Tim Berners-Lee). Permanent availability of the IEDC is dependent upon the continued hosting of database servers, which are currently the responsibility of specific research institutes (presently only Freiburg for the prototype). Projects such as the *Wayback Machine* of the *Internet Archive* are thus far ensuring that the web is permanently stored and accessible (Archive.org, 2019).

3. "interoperable: (meta)data use standard data vocabularies, in a formal, open, and broadly applicable language, and include references to connected data."

Pre-existing vocabularies or ontologies are not used in the IEDC. On the semantic web, this is strongly encouraged and "there is a trend towards the adoption of well-known vocabularies by more datasets" (Schmachtenberg et al., 2014). There is presently no mechanism for following connections to external data sources from within the IEDC. Whereas this is a fundamental aspect of the design specification for the semantic web technology stack.

4. "reusable: (meta)data are defined with relevant attributes for reuse such as a clearly defined license statement."

The IEDC provides some bespoke design choices to enable tracking of data licensing and provenance metadata. No extra layer of licensing is applied when a user loads data into the IEDC and so there are no *additional* legal barriers raised when data is loaded into the system. This is also generally true on the semantic web. Here, extensive effort has also been applied into developing standards for tracking provenance, such as the *PROV family of documents* which enables the "inter-operable interchange of provenance information in heterogeneous environments" (W3C-Prov, 2019).

To summarise, the four FAIR principles are not broadly satisfied by the current design of the IEDC prototype. Whereas they are generally inherently satisfied by using the semantic web technology stack. Transforming the main datasets used in socioeconomic metabolism into

web-based open data standards would help to create rich potential for productive cross-disciplinary collaborative research efforts.

Appendix O Mapping document structure and guidance

The mappings are separated into two parts, labelled the *auxiliary* and *core* data mappings. *Core* refers to the part of the IEDC schema shown in Figure 5.2, and *auxiliary* is everything else that was systematically excluded beforehand. The auxiliary mappings are comparatively simple, requiring less information to describe them with low ambiguity. The core mappings are often more complex and include *transformation rules* to describe them, plus additional references to the source data model. The contents of both the auxiliary and core mapping tables are explained here, via reference to the column headers.

- **MappingID** is a number used for referencing specific mappings in the subsequent section. Each MappingID describes one or more rows being inserted into a single IEDC destination table.
- **Order Level** refers to the necessary load order for the IEDC tables
- **Target table and column names** are self-explanatory.
- **Data Type** refers to the basic data type and precision of the IEDC column.
- **Key** includes an entry when the column is indexed. Explanations for entries available at https://dev.mysql.com/doc/refman/5.7/en/show-columns.html
- **Nullable** is “Yes” if the IEDC column is set to permit NULL values
- **Entity Name** depends upon the *transformation category* entry. When the Category = "lookup" for a row, a lower-case IEDC table name is entered. Else, it refers to the

Semantic Catalog entities, and Capital Starting Letters are used. *Hardcoded* values do not require specification of a source entity.

- **Value** is the column named used in the auxiliary data mappings. Integer values are keys of foreign tables. Text entries are entered as strings. If Nullable = True, empty cells mean NULL, else they imply an empty string ("").
- **Value / attribute name** when the core mapping is *hardcoded*, this is simply the value inserted into the destination. When a *lookup*, this specifies the ID or name of the value to be looked up in a related table. When the mapping is *direct*, the attribute name of the source is entered. For transformations, see the Comment column.
- **Use existing IEDC records?** Refers to the pre-populated data of the IEDC. Where possible, this data is used, rather than adding all new values from the source dataset. This minimises data redundancy, storage space, and is generally standard practise for filling databases. SQL insert statements must only be written for the entries which state "NO" for this column.
- **Transformation category** categorizes columns based on how they are handled in the mapping. Possible values are: Direct, Transformation, Hardcoded, Lookup.
- **Transformation rule** includes an entry when the category= “Transformation”. In this case, they are natural-language or pseudo-code descriptions of the transformation required.
- **Comment** is used to explain potentially unclear elements of any other column. They also highlight specific problems encountered in each mapping, which contribute to the final answer to the research question. A comment in the 'ID' row of a particular mapping always refers to the whole mapping, rather than just that row.