L95/196

12/95

SPINE

A Relation Database Structure for Life Cycle Assessments

Bengt Steen*, Raul Carlson and Göran Löfgren***,**

Göteborg
september 1995

*Swedish Environmental Research Institute, Gothenburg **Chalmers University of Technology, ***Chalmers Industriteknik

SPINE -A Relation Database Structure for Life Cycle Assessments

Contents

page

Preface 5

Summary 6

Introduction 7

2. Relational databases and database design 9

    2.1 What is a database? 9

    2.2 Relational databases 10

    2.3 The design process 10

    2.4 Data model terminology and notation. 11

3. LCA Data model 13

    3.1 General aspects 13

        3.1.1 Three systems involved

        3.1.2 Statistical aspects

        3.1.3 Numeric data

        3.1.4 Metadata

    3.2 The technological system 15

        3.2.1 Activities and Flows

            Flow connections

            Calculating sizes of inputs and outputs

            System hierarchy

            Base flow

            Activity parameters

            Allocations

        3.2.2 Substances

            Nomenclature

            Compounds and properties

        3.2.3 Object of study and the inventory process

    3.3 The environmental system 24

        3.3.1 Environment and geography

            Environment

            Geography

        3.3.2 Classification and characterization

            Numeric data

            Metadata

        3.3.3 Impact assessments

    3.4 The value system 27

4. The database 28

4.1 Terminology and symbols
28
4.1.1 Table Columns
4.1.2 Optional Tables and Attributes
4.1.3 Statistical representation
4.1.4 Symbols used
4.2 The tables of the database 30
4.2.1 Qualitative and Quantitative Meta Data
The table QMetadatatype
The table QMetaData
4.2.2 The table of Units
The table Unit
The table ActivityParameterType
4.2.3 The tables representing the database of substances
The table PropertyType
The table SubstanceProperty
The table FlowProperty
The table AlternateName
The table Substance
The table SubstanceCategory
The table Composition
4.2.4 The table of addresses and names
The table JuridicalPerson
4.2.5 The tables representing the Object of Study
The table Sector
The table ProcessType
The table ProcessName
The table ObjectOfStudy
4.2.6 The tables representing Environment and Geography
The table Environment
The table Geography
4.2.7. The tables representing the Activity and Flow concept
The table Activity
The table Componentship
The table Flow
4.2.8 Linking Activity, Flow and Internal Activity
The table FlowConnection
The table ActivityParameter
The table Allocation
The table BaseFlow
4.2.9 The table for documenting the Inventory
The table Inventory
4.2.10 The tables related to Impact Assessment
The table ImpactAssessment
The table AssessmentConnection
4.2.10 (cont.) The table ImpactIndex

The table Event
The table UnitEffect
The table UnitEffectValue
The table ValuationMethod
The table ActivityImpactIndex

5. Applications 57
5.1 User interfaces and tools 57
5.1.1 Simple database update-programs
5.1.2 Flowchart constructors and viewers
5.1.3 Analysis tools
5.2 Algorithms 58
5.2.1 Vertical structures and error propagation
5.2.2 Statistics and sensitivity analysis
5.3. Precautions 60
5.3.1 Deep structures and error propagation
5.3.2 Simplified work with no quality checks
5.3.3 Updating of data referred to

6. Future research needs 63

7. Acknowledgments 64

8. Literature 64

Appendix 1: The data model and SQL-code.

Appendix 2: Globally unique artificial keys.

Appendix 3: The NEP project.

Preface

This report has been prepared as a part of the Nordic NEP-project and is the final report from its first subproject, - “Datatools”. A closer presentation of the NEP project is made in appendix 3.

The report includes achievements made on Chalmers University of Technology, Departments of Technical and Environmental Planning and of Computing Science, where Raul Carlson made a datamodel of the life cycle inventory as a diploma work, supervised by Annmari Tillman.

Göran Lövgren has been responsible for the data modeling and Bengt Steen for the environmental aspects. Bengt Steen has also been the project leader.

We like to express our thanks to everyone who has contributed to this work. The relational database reflects very much the language that is used by the LCA community and it is through listening and participating in the discussion that the language is learned.

Summary

A database structure given the name SPINE (Sustainable Product Information Network for the Environment) has been developed to allow communication between different software tools, in particular the “EPS system” and Chalmers Industritekniks “ LCA Inventory Tool”. SPINE is a relational database structure and allow further expansion and development. SPINE is designed to be able to serve several types of software programs with data sets and is described in a standardised way in a SQL-code. SPINE may also be used as a common language for LCA data structures. The structure has been derived through a thorough analysis of what information that is of value in LCA, and how it relates to each other. A summary of the result is shown in appendix 1. However it is necessary to be somewhat familiar with datamodel language in order to be able to read it. An explanation of the signs used in datamodeling is given in section 2.4.

SPINE -A Relation Database Structure for Life Cycle Assessments

1. Introduction

During the first years of the 1990-ies two LCA systems/programs was developed independently at Chalmers (LCA Inventory Tool, LCAiT) and at IVL (The EPS evaluation system), (Steen and Ryding, 1992). By chance they were complementary in that they were focusing on two different parts of the environmental impact assessment. The LCAiT analyzed and documented technical systems while EPS used this information for a valuation of the environmental impact and support decisions in product development.

In an international perspective, there was two or possibly three "trends" that was important for the development of these software’s.

One was the development of the LCA-concept. LCA was becoming a discipline and changing from pure inventories towards more complete decision support tools through complementary impact assessments. SETAC (Society for EcoToxicology And Chemistry) became a forum for scientific international exchange, ISO and CEN standardization organizations and ICC (international chamber of Commerce) started to show interest in the LCA concept. SETAC wrote a ‘Code of Practice’ to harmonize the LCA thinking and terminology. (1993)

The second was the integration of the sustainability concept in governments and the scientific society. In the UNEP magazine "Industry and Environment" a special issue was made on the subject "Sustainable Industrial Development" during 1989 The UNEP director Mustafa Tolba, said that "The 1990s will witness enormous changes in most sectors of society, but in almost none will they be as evident as in industry and environment and the new relationship that has formed between them" (1989)

A third trend was the integration of computers into everyday office work. Experts and decisionmakers became used to express and exchange information using different programs and databases as tools. Such tools are more or less necessary to have in order to handle and process the huge amount of data involved in a life cycle assessment.

The Nordic NEP project was started 1993 in order to integrate the EPS system and the LCAiT into the product development process in companies. A special sub-project was created to connect the LCAiT and the EPS system so that it could be used together in the development process. After testing the software’s at companies and after specifying the performance needs it was found that the information used by the EPS and LCAiT software’s separately was not enough for the practical use of the software’s together. Information telling when the data was produced, who did it, which region was concerned etc. was needed to be able to combine data belonging to the same systems or at least know when combining data from two systems.

Such data can be stored in many ways, but in order to facilitate data exchange, minimize the work involved and to allow future changes of the database it was decided to develop a relational database allowing easy access to most types of data relevant for life cycle assessments.

The relational database should be able to communicate with the EPS-system and the LCAiT as well as with other software tools.

Separately a relational database for LCA inventories was designed as a diploma work by Raul Carlson at Chalmers (Carlson, 1994). At the end of the NEP project part of his results were used within the NEP project and a common structure was developed.

The new database was given the name SPINE, which stands for Sustainable Product Information Network for the Environment.

2. Relational databases and database design

2.1 What is a database ?

For most people a database is a collection of interesting data. The simple character is more or less true for the end user, but for people involved in the process of setting up or maintaining a database, it is far more complicated. The database in itself is, on a conceptuel level, made up by three different parts:

- the actual **data**
- a **structure** in which the data must be organized,
- a software system, called **DataBaseManagementSystem** (DBMS), that manages both data and structure.

.

There are different database technologies available: hierarchical databases, network databases, relational databases and object oriented databases. Relational database technology is, by being the subject of both international standards and a solid theoretical platform, by far the most widely spread database technology of today. It is commercially available from a great number of vendors and most programmers are skilled in the area. Since relational database technology must be the obvious choice in this project, the subsequent discussion is based on this technology.

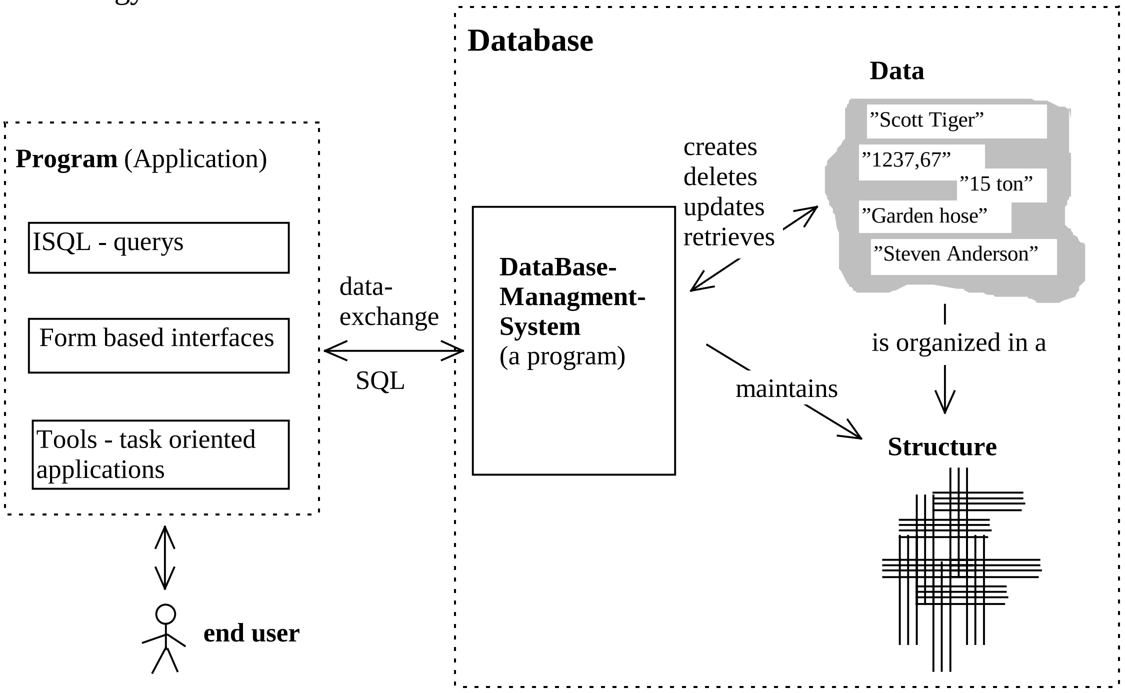


Figure 1 The database concept

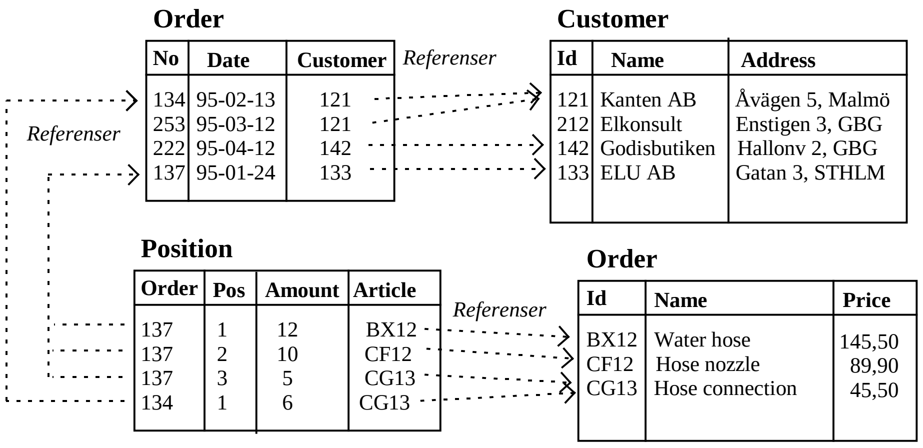
Buying a relational database from a vendor like Oracle, Informix, Sybase, etc, is buying the DBMS - data and structure must be obtained elsewhere. Since an adequate structure is a prerequisite for putting any data into a database, the conceivement of an adequate structure for LCA will be the main subject of this report and further discussed in subsequent chapters.

As the figure above implies, there must also be some kind of application which communicates with the database and makes data available to the end user. In most cases this concerns not only viewing the data, but also applying different calculations and search conditions in order to establish new informaton on aggregate levels.

The most simple kind of application is one that connects the user to the database through interactive SQL. SQL is a standardized computerlanguage for relational databases, adressing functionality such as searching for, updating, creating and deleting data. Interactive SQL makes it possible for the user to query and manipulate data through simply typing commands using the SQL syntax. It is usually, at no extra expense, shipped together with the DBMS directly from the vendor. Interactive SQL can occaisionally be useful for the expert mastering SQL, but for the common user other types of task-oriented and user-friendly applications are needed.

2.2 Relational databases

A relational database is organized in tables. A table typically represents a real world entity or concept, for instance customers,orders, invoices, and articles. Each table is made up by a number of columns, of which some are used for datastorage and others for keeping references to other tables. Together they build a more or less complex structure, which will be referred to as the database structure.



Order

No	Date	Customer
134	95-02-13	121
253	95-03-12	121
222	95-04-12	142
137	95-01-24	133

Customer

Id	Name	Address
121	Kanten AB	Åvägen 5, Malmö
212	Elkonsult	Enstigen 3, GBG
142	Godisbutiken	Hallonv 2, GBG
133	ELU AB	Gatan 3, STHLM

Position

Order	Pos	Amount	Article
137	1	12	BX12
137	2	10	CF12
137	3	5	CG13
134	1	6	CG13

Order

Id	Name	Price
BX12	Water hose	145,50
CF12	Hose nozzle	89,90
CG13	Hose connection	45,50

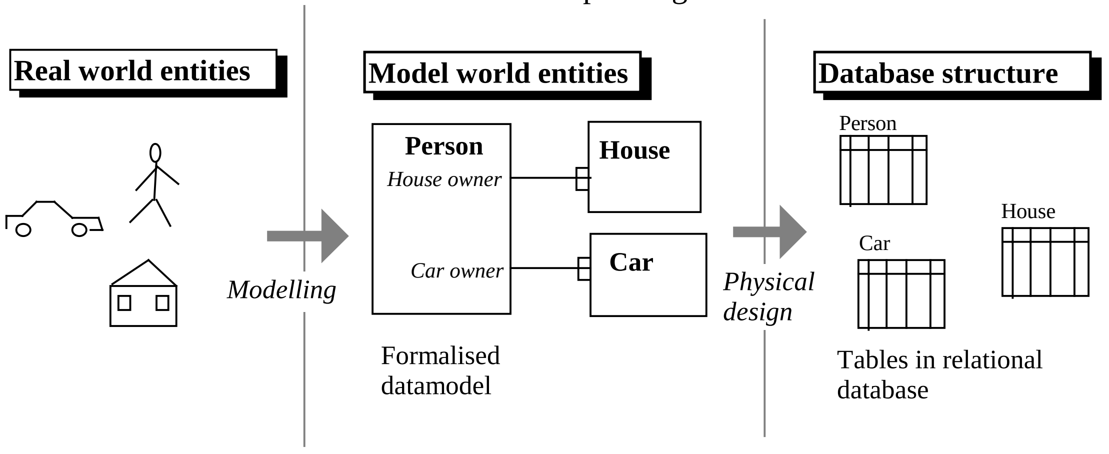
Example of a database structure with four related tables. From these tables we can, among other things, conclude that a company with the name ”ELU AB” put an order at Jan 5th 1995, which contains 12 water hoses, 10 hose nozzles and 5 hose connections.

Figure 2

2.3 The design process

How well a database structure will serve it’s purpose depends to a great extent on how well it corresponds with the real world concept it is supposed to represent. In fact, this is a critical success factor in the task of designing databases. The first step in a design process is therefore to identify and understand the real world concepts of the actual problem domain. A common practice is to establish a conceptuell model which reflects the concepts of the problem domain

at a high level of abstraction. Such a model allows for concentrating on general topics and principles without having to engage in technical matters more related to database technology. The models used in this field are commonly referred to as datamodels and deals with the definition of entities and their relations. A datamodel entity is a model representation of a real world entity or concept. Some entities are of a concrete nature and are therefore easily identified, but others can be far more abstract and sometimes harder to identify. Customers, articles and invoices are examples of the former and an event such as a doctors appointment can serve as an example of the latter.
A data model is, despite of it’s high level of abstraction, strongly formalized, and is easily transformed into a database structure. This transformation is quite straightforward, and most of the entities of the model will have a corresponding table in the database.



The database design process can be divided into two steps:
1) Modelling. This concerns the mapping of ”real world” entities of the problem domain into a conceptuell model (datamodel).
2) Physical design. The datamodel is mapped into databasestructure consisting of tables. Decisions made in this step are of a technical nature adressing topics such as performance and maintainability.

Figure 3

The establishment of a datamodel is, as already stated, an important part of the design process, but it also serves as a way of understanding the database stucture and how it can be exploited in different applications. Accordingly the description of the database is divided into two parts where the first one mainly adresses the datamodel and the second one adresses the databasestructure (actual tables and their contents).

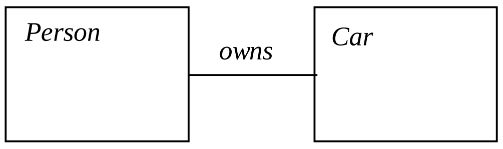
2.4 Data model terminology and notation

Datamodels are based upon three main components:

- **entities**, which are model representations of real world entities
- the **attributes** by which an entity is described. For instance is the entity *person* described by the attributes (among others) *name, age, weight* and *eyecolor*.
- **relations** between entities. Relations shows how entities can be associated with each other. For instance can the two entities *person* and *car* be be associated to each other by *ownership*.

Datamodels are usually graphs using rectangles as symbols for entities and lines for relations.

Example:



If the meaning of the relation is not apparent, an explanatory piece of text can, as in the example above, be supplied in the graph.

Figure 4

A ”fork” at the end of a relation-line indicates that a single occurence of an entity can have the same type of relation to a number of ocurrences of the other entity.

Attributes are usually put in a special list, sorted by their entities. Since attributes are transformed into into tablecolumns with one column for each attribute, we will not supply a special list of attributes in the datamodel but instead refer to the tables description.

3. LCA data model

3.1 General aspects

3.1.1 Three systems involved

In order to make an impact assessment of a product life cycle, information from three systems have to be combined.

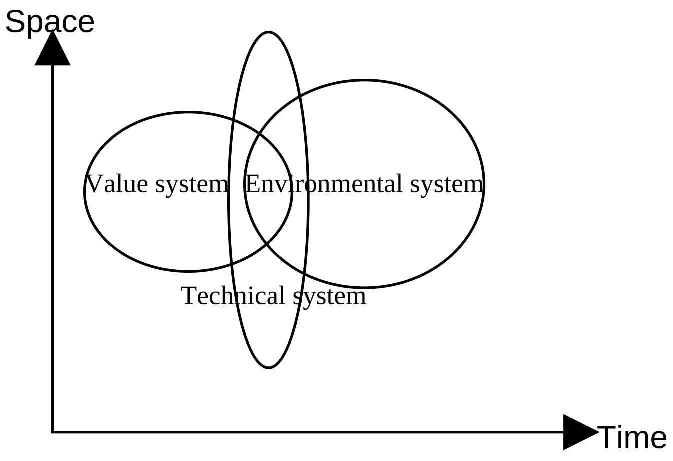


Figure 5 The value system, the technological system and the environmental system are all involved in an environmental life cycle assessment although they have different extension in time and space.

By the value system is understood a part of the social system. The values or rather attitudes against various changes in our environments are based on experiences in the past on an individual and cultural level. Information from the value system is either used for selecting data from the technological and environmental systems that will be compiled and analyzed or to weigh various impacts into common figures.

The technological system contains the human activities impacting on the environment. The technological system is normally well defined close to "the own" activity, but less defined further away. For example the steel used in a product of a company has a well known deliverer, while the coal used for producing electricity on the net may only be known as coming from the "world market". The technological system may have a local or global extension in space, but is rather limited regarding time extension to "now".

The environmental system is partly linked with the technological system, as in agriculture and fishing. The interest of the environmental system is normally focused on future changes.

The three systems have no clear borders as to their extension in time, space and quality. An event in the technical system may lead to one or several never ending chains of consequences. The database structure must allow keeping track on such infinite chains. The practitioner himself must be able to decide when it is meaningful to start and stop the recording of data from the chains.

A main objective of the database is to select and store present knowledge from the three systems. Another objective is to get the most out of the present knowledge rather than limiting the analysis to those areas, where the knowledge and data quality is sufficient. "Anything" may thus be stored in the database. It is up to the application program to decide which type of data it want to use.

As a consequence, the uncertainty is of great interest. It is realized and accepted as a fact that the knowledge is poor in some cases. This is partly because it has not been possible to collect the information and partly because there is a real variation in the population of data.

When evaluating an environmental impact from a human activity, the impacts on the environmental system in terms of specific emissions and specific resource depletion's from a known activity has to be represented by general emissions and general resource depletion's for which there is general knowledge of its consequences. Each specific emission and resource use may be represented by several general emissions. In other words it may be part of several populations of emissions or resource depletion's. As an example an emission in the city of Gothenburg may be analyzed on a local scale as a Gothenburg emission or on a national level as a Swedish emission or even as a sample of global emissions.

A similar reasoning may be applied on the value systems.

When actually carrying out an impact assessment the choices of general values, emissions and resources must be recorded in the database. Otherwise it will not be possible to repeat an impact assessment.

Each emission, resource depletion or other characteristic human activity starts a chain of events in the environment. Chains may be interacting, branching and looping. The end point effect of a chain is the one that is connected to the value system. The end point effect may be described in terms of number of "unit effects". A unit effect is a well defined entity, like a man-year of morbidity of a certain intensity.

3.1.2 Statistical aspects

The system description has to be made in such a way that the database user can tell if the activity he or she studies belong to any population of activities represented in the base and therefore has the same relations to other data as the existing database indicate.

In order to be able to assess the significance of the results we have to record the distribution of data belonging to a population of activities.

In environmental statistics log-normal distributions are common. Small variations in technical systems may be of ordinary normal distributions. Standard deviations are seldom available, but sometimes it is possible to get an estimate of maximum and minimum values. If possible, max. and min. values are set equal to 98 and 2 percentiles (2 standard deviations in a normal distribution). However in practice the max. and min. values are recorded or noticed or estimated without any consideration of how probable there are.

3.1.3 Numeric data

Several of the entities in the model contains a numeric value representing the quantitative aspect of some physical event. One of the most central attribute in the model is the quantity of a certain substance that is carried in a flow. Without such data emissions and resources cannot be calculated and, in the end, no effects can be estimated. In best case such quantitative flowdata can be based upon actual measurements made with high precision methods and instruments and in worst case an educated guess will have to do. Giving public access to such data, calls for supplying potential users with additional data from which quality, accuracy and applicability of the data can be concluded. This is actually data about data which is referred to as metadata.

3.1.4 Metadata

The metadata, concerning numeric data, comprises information such as age, means of aquisition, litterature references, geographic and other representation, etc. This data is contained in an entity called **QMetaData**. Any other entity containing a numeric value can be related to a QMetaData entity, thus establishing a set of metadata for the value.

3.2 The technological system

3.2.1 Activities and flows

The central concept of the datamodel is activity. Activities are, for example, manufacturing, incineration, transports, rawmaterial extraction, fuel combustion, use of a product and waste deposition . An activity has inputs and outputs of rawmaterials, products, resources, energy, wastes, emissions etc. Inputs and outputs are thought of as flows of matter and energy in and out of the activity. Some flows seems more immaterial in their nature, as for instance the use of land as a resource in forestry. This presents no problem as long the as a flow is quantifiable in some defined unit. **Activity** and **Flow** are two central entities in the datamodel.

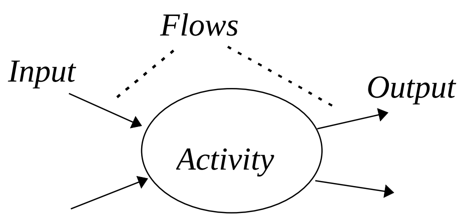


Figure 6

Flow connections

Activities can be connected to each other by their flows. An output of an activity is **connected** to an input of another activity. In this way we can model a network of activities with products, materials, energy, etc, flowing between them.

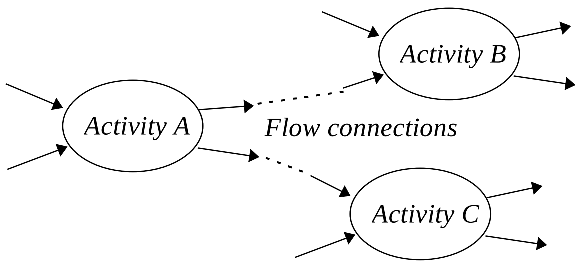
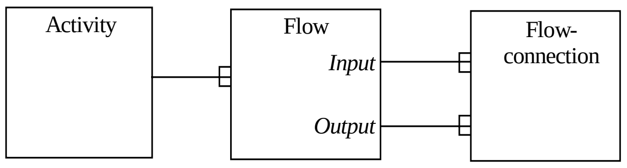


Figure 7

Connections are represented by the entity **Flowconnection** in the datamodel.



The central entities of the ER-model.

Figure 8

Calculating sizes of inputs and outputs

A flow can be assigned a numeric value representing the size of the flow in some unit. The values of all the flows belonging to the same activity constitutes a coherent set of values, representing the activity at a given throughput. A decrease or increase of the throughput values will result in a change of those values. To calculate actual values after such a change, one must determine or make assumptions about how different flows relate to each other, and express those relations in terms of mathematical functions. The most simple model, which also can be used as default if nothing else is stated, is to assume a direct proportional relation between all flows. This means that if any flow is increased or decreased by a certain factor, all other flows are changed in the same manner by the same factor. A more sophisticated model must allow for expressing relations between flows in mahematical terms, which is beyond the scope of this project.

System hierarchy

The activities participating in a network will together constitute a system where all non-connected flows passes the systemboundaries.

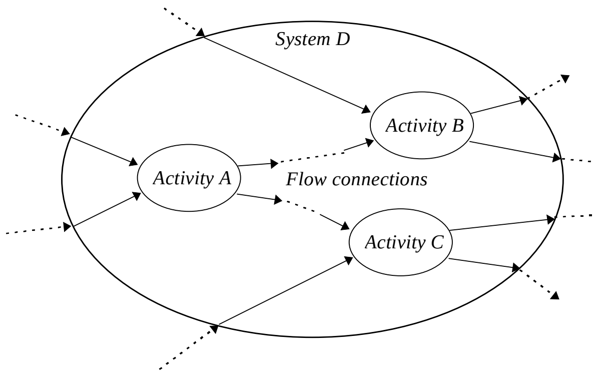


Figure 9

If we disregard the inner structure of the system and only pay attention to what happends at the systemboundaries, the system itself can be regarded as an activity.

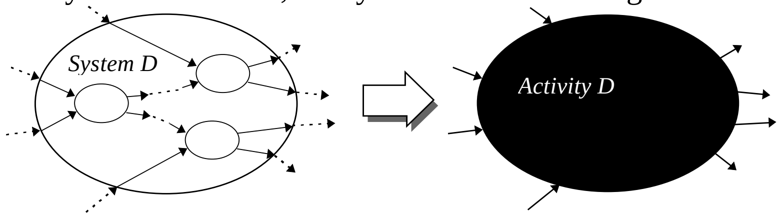


Figure 10

Any activity can both have an inner structure and participate in an inner structure. This means that activities can have hierachical structures of arbitrary depths.

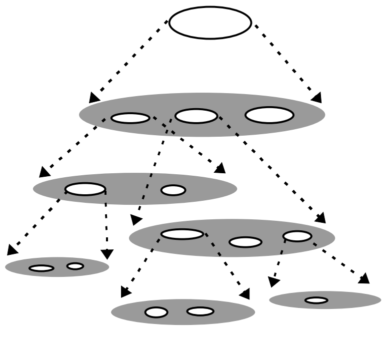


Figure 11

Many of the activities we model are likely to be used in several LCA:s, for instance transports, production of energy and waste treatment. An activity must therefor be able to participate in more than one hierarchy.

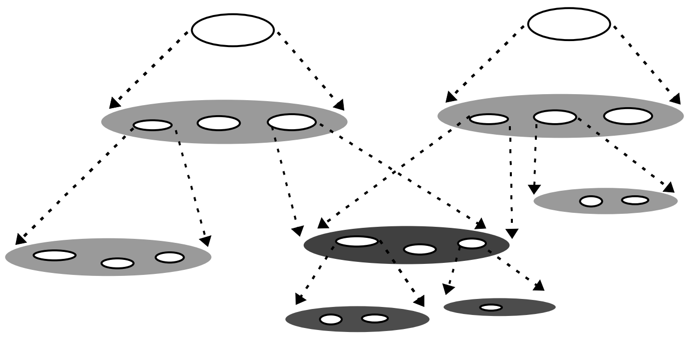


Figure 12

In the datamodel this relationship between activities is represented by an entity called **Componentship**. This entity carries structural information and shows how activities are related in a hierarchical structure. The activity on the higher level has the role of a system and the other the role of a subsystem.

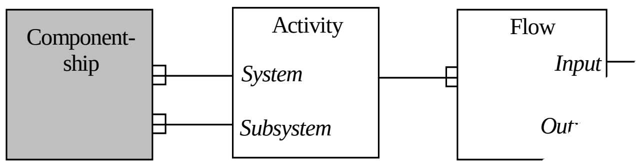


Figure 13

The fact that an activity can participate in several systems has an important implication for the entitiy flowconnection. This is no longer related only to the flows involved, but also to the fact that a connection is valid for certain system. In other words, an activity can connect differently to other activities depending on which system it is participating in, which is why flowconnections also must be related to componentships. Since both activities involved must be a part of the same system, there must be a relation for each activity.

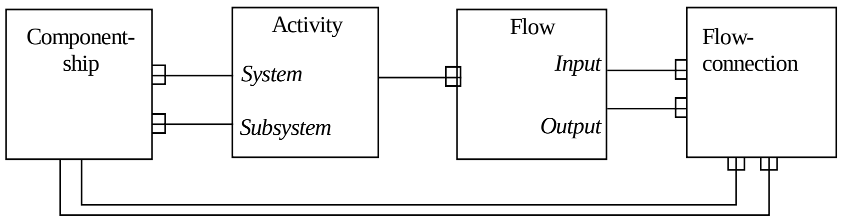


Figure 14

This structure-building ability of the datamodel faciliates several different approaches in LCA modelling:

- large systems can be arranged into an easy to grasp structure consisting of a few larger blocks, each block representing a subsystem.
- a system can be modelled in a top-down fashion where high level activities can, at later stages in modelling process, be made into subsystems with inner structures.
- any activity or system can be reused as a component in the making of new LCA:s.

Baseflow

In order to calculate a LCA model, the throughput of the entire system must be determined. All calculations must emanate from a given flow, which is considered to be the baseflow of the system.

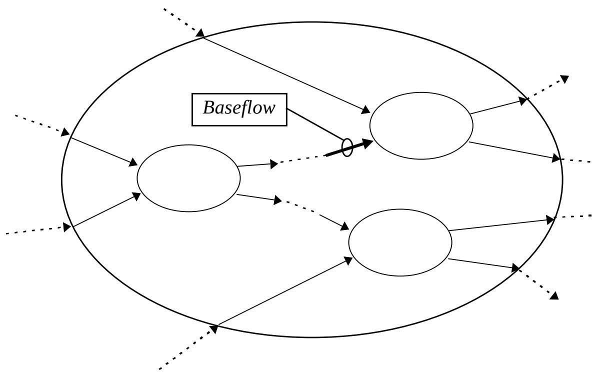


Figure 15

Since baseflow is not an entity but rather a role that a flow has in relation to its system, its represented as a relation in the datamodel.

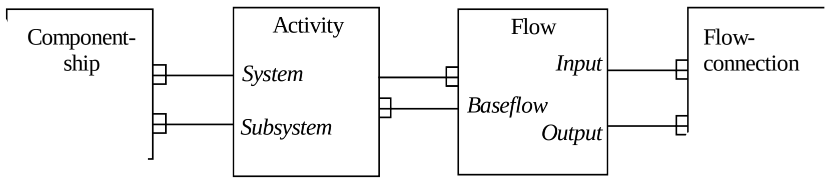


Figure 16

Activity parameters

So far the datamodel is conceived under the assumption that the quantitative aspect of a flow depends on nothing but other flows. This is not always true, which is demonstrated in a transport activity where the flows depends on the distance of the transport and some other **parameters** as well (drivers behaviour etc). Since an activity can belong to a number of systems, those parameters must be individually set in each system. This why such parameters are related to componentship rather then to the activity itself.

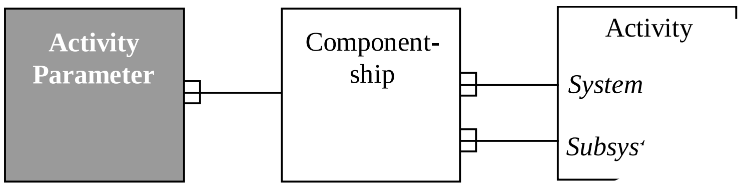


Figure 17

Allocations

Whenever an activity have more than one product output, decisions about the allocations of emissions and resources must be made. Such decisions are guided by the allocation principles that are accounted for in the inventory description (entity inventory). Regardless of the principles used (economic, by weight, etc) the outcome of the decision can be represented by a ratio telling how much of a flow that is to be allocated for a certain product output.

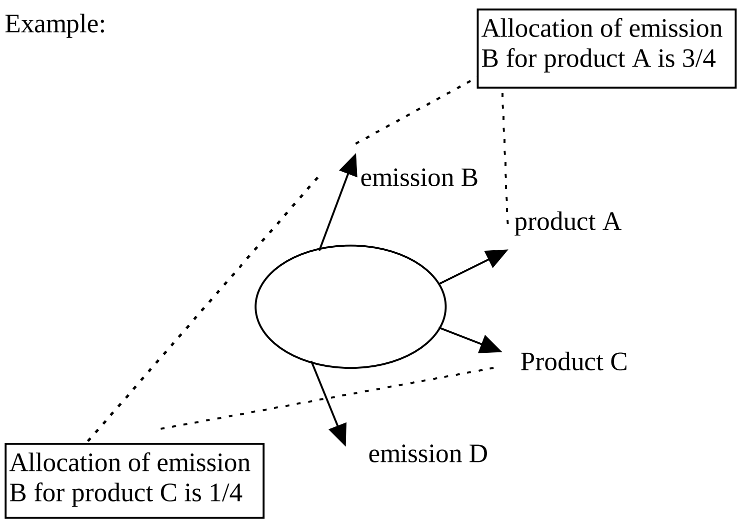

Example:
Allocation of emission B for product A is 3/4
emission B
product A
Product C
emission D
Allocation of emission B for product C is 1/4

Figure 18

The **allocation** is represented by an entity of its own in the datamodel. Since different decisions can be made for the same flows depending on which system they participate in, the allocation must, like flowconnections, be related to both the flows involved and a componentship.

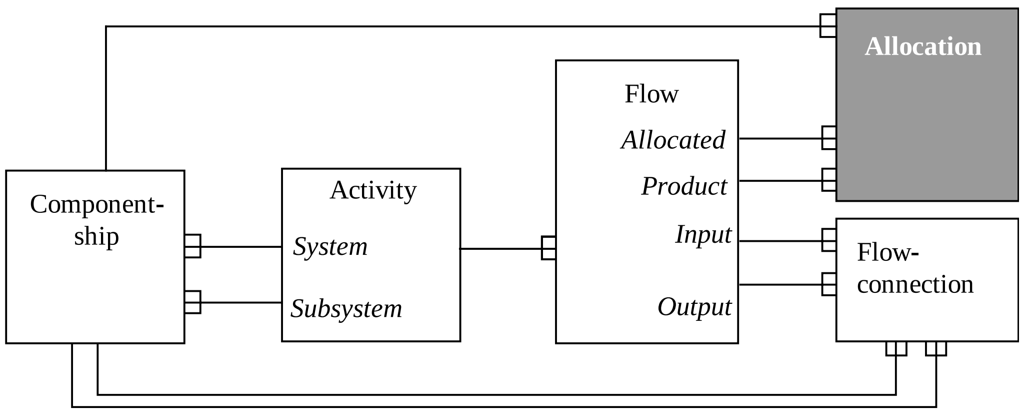


Figure 19

3.2.2 Substances

The things that makes up the contents of a flow can exist whether it is carried in a flow or not, and is therefor modelled as an entity of its own, called **Substance**. The word substance is used in a broad sense including all kinds of substances, materials, products, energyforms, resources etc. A flow is related to the actual substance that it contains.

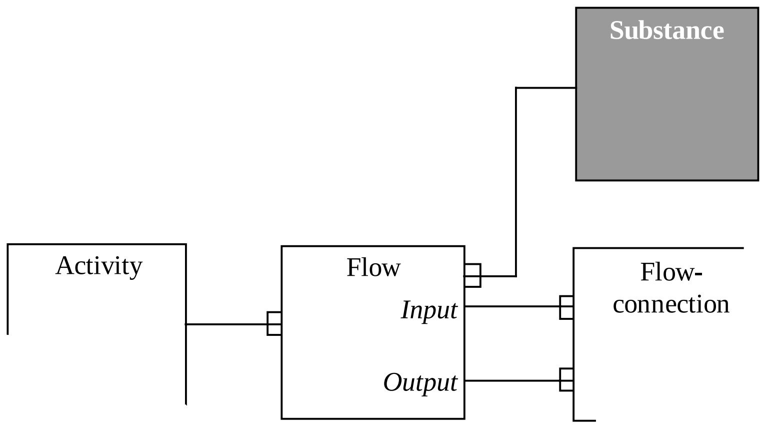


Figure 20

Nomenclature

In the long run there is a need get the substances in the database organized in a nomenclature. A nomenclature is basically an hierarchical structure, but we must also allow for a susbstance to belong to more than one category.

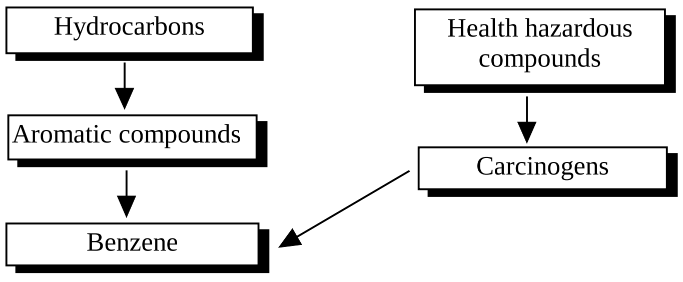


Figure 21

This actually calls for a network structure, which in turn calls for an entity dedicated to hold such structural information. This entity is called **Substance category** and has two relations: one to the substance which is superior in the nomenclature and one to the substance that is subordinate. Note that this entity keeps nothing but structural information, which means that also categories are seen as substance entities.

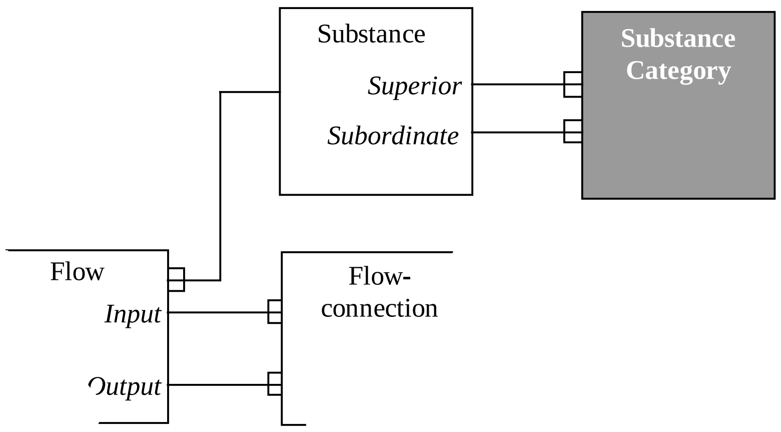


Figure 22

Compounds and properties

A substance can be a compound of other substances, which we must be able to account for in the database. It can be argued that a substance which is a component in a compound can be a compound itself and so on. This adds up to a hierachical structure similar to the systemhierachy of activities, and is modelled in the same way with an entity holding such structural information. The entity, which is called **Composition**, has two relations: one to the substance that has the role of the compound and one to the substance that has the role of the component.
Substances can also have properties that must be accounted for, for instance dry weight, water contents and density. A **Property** is an entity of its own, but must be related to the substance it bears upon. Some properties are not inherent in a substance, but arises when a substance is a subject of a flow, for instance temperature. In such a case the property can be related to the actual flow.

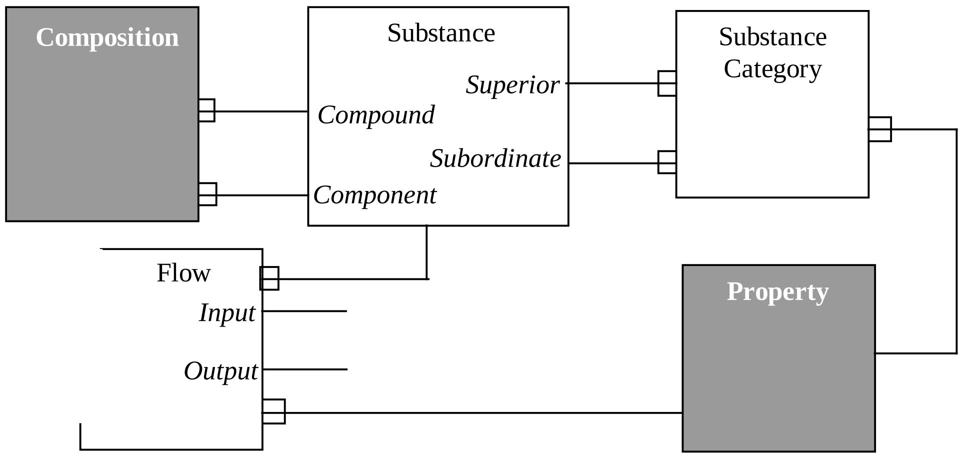


Figure 23

3.2.3 Object of study and the inventory process

We use model entities such as activities, flows, flowconnections, etc, to build a model of some real world activity or process. This real world entity will in this concept be referred to as the **Object of study**. Since the same object can be studied and modelled numerous times, it is convenient to have a separate model entity accounting for common information about the object. This can be information about the site of the process, which company that is responsible, etc. Any time a new model activity is conceived, it can be related to the object it concerns.
In order to judge the quality and applicability of the model one needs to know under which assumptions the model was conceived, and how the inventory process itself was conducted. This concerns information such as goal definition, purpose, functional unit, systemboundaries, practitioner, etc. This set of data is contained in an entity called **Inventory**, which will be related to the activity it concerns.

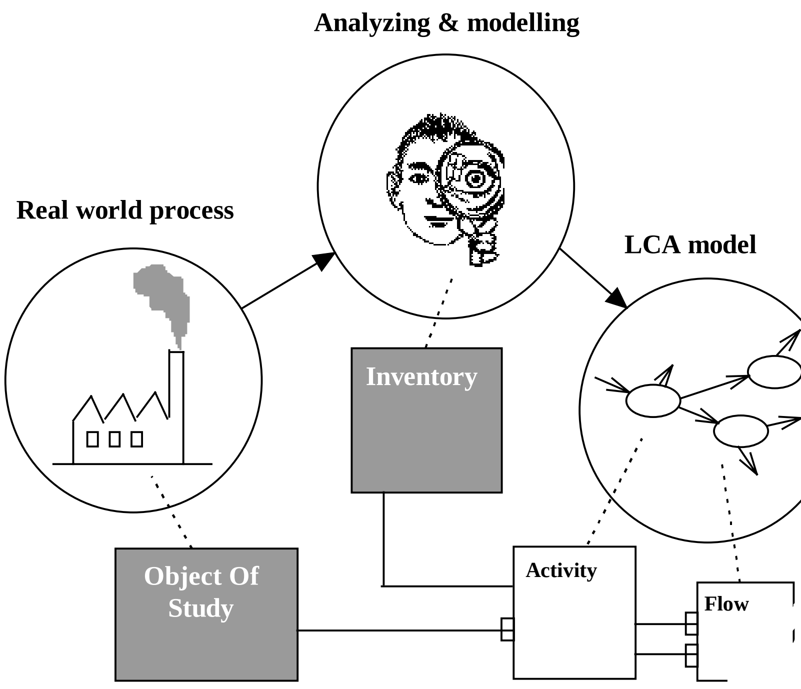


Figure 24

The making of inventories involves people and organizations in different roles. Since it is very likely that a person or organisation will appear in many occasions, it is convenient to handle such information in a separate entity that can be related to an actual inventory. Since hhis entity takes care of both people and organizations it is called **Juridical person**.

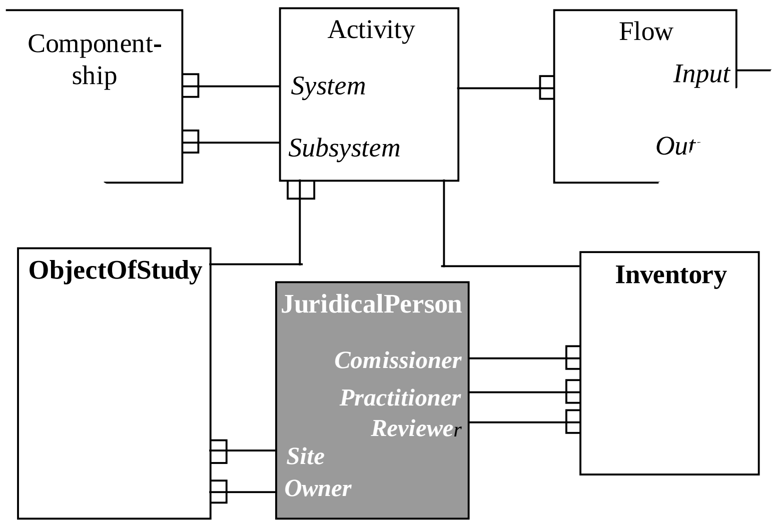


Figure 25

3.3 The environmental system

3.3.1 Environment and geography

In order to evaluate the environmental impacts of a system, one not only needs to know which emissions and resources that are released and extracted, but also the properties of the environment where those events take place.

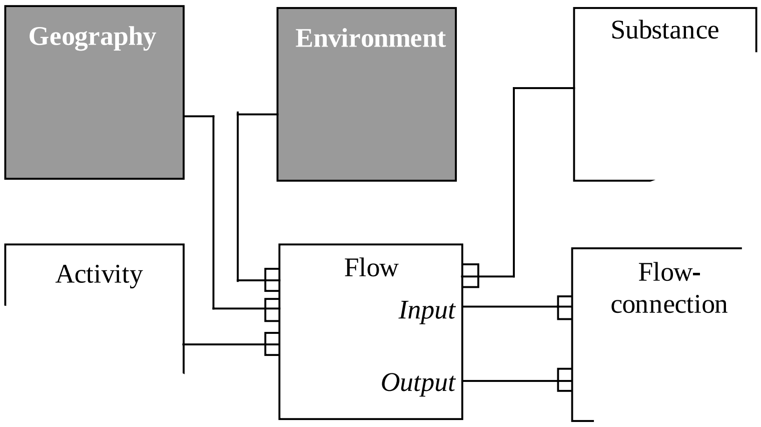


Figure 26

Environment

Whenever a flow passes the boundaries of the technical system into or out of a natural system, there is a potential environmental impact. The nature and effect of this impact is dependent upon different properties of the environment. We can roughly define those properties by using descriptions such as air, water, limnic systems, eutrophic lake, ground etc. However, every set of properties that together define a certain environment, form a model entity called **Environment**. Any flow that passes the system boundaries into or out of such an environment, is to be related to that environment.

Geography

It can be argued that the environment is inherent to a specific geographic area and consequently it would suffice to state only the location of a flow. This is not always true since many of the systems modelled in LCA are not sitespecific but represent average processes in larger areas. In these cases there may be a lack of coherency between environment and location. Accordingly we have chosen to treat environment and geography as to separate and independent entities that supply complementary information about a flow. A flow is separately related to its geographic area and its environment.

3.3.2 Classification and characterization

It has become common in life cycle analysis to classify and characterize inventory parameters in order to indicate what type of hazards they exhibit to the environment and to facilitate comparisons. The classification and characterization is normally regarded as a independent of where and how the flow occurred. Therefore the classification and characterization factors allowing quantitative comparison between inventory parameters within a class may be stored in tables that are solely related to substances and the type of environment (air, water soil) they are emitted to or extracted from (resources).

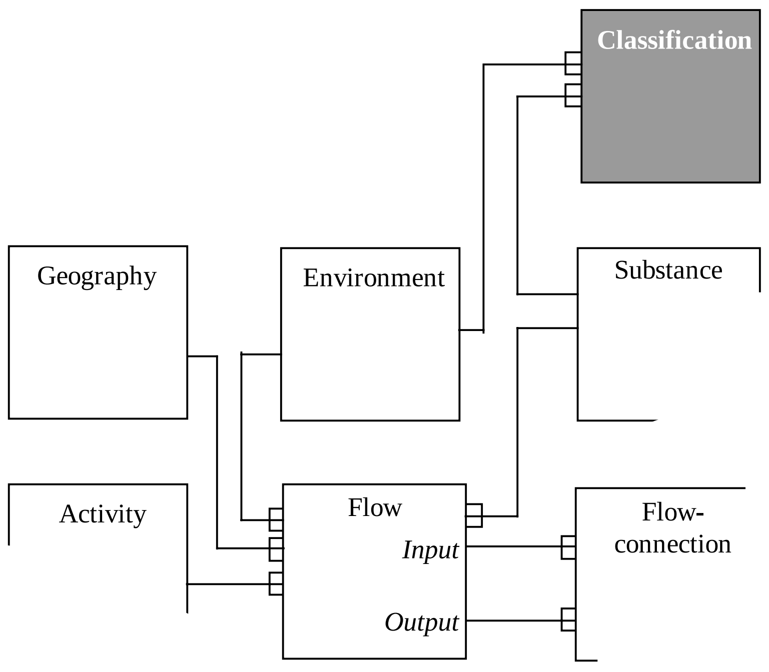


Figure 27

3.3.3 Impact assessment

In the database impact assessments may be made for defined emission populations, resource depletion's or other human activities practical to use as building blocks in an impact assessment from a more complex human activity. Such an assessment results in an ImpactIndex or an ActivityImpactIndex. The objects in the impact assessment procedure is shown below:

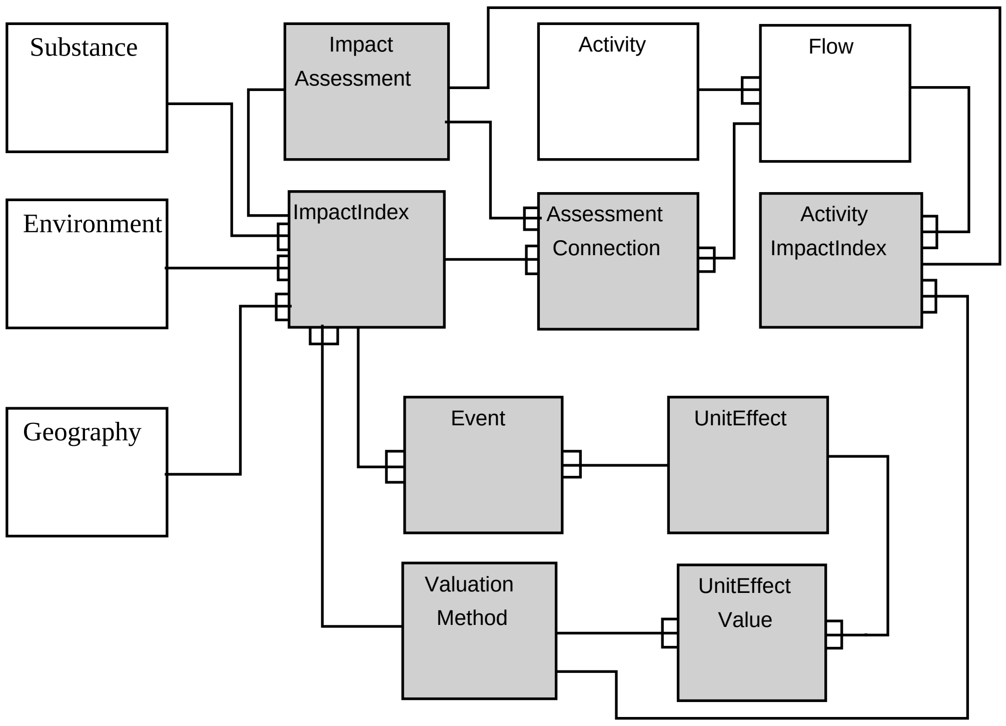


Figure 28

Consider for example an impact assessment that was made by Lars Larsen 1994 of an emission of sulfur dioxide. The emission was a very specific one, the 1992 January emission to air of 100 ton from a 100 m stack in the Swedish west coast. We could then note in the ImpactAssessment table data about the assessment procedure, for instance that it was made by Lars Larsen and in the ImpactIndex table the result and when the result is valid.

The calculation of the impact index for the emission of sulfur dioxide requires information on the fate of the sulfur dioxide in the environment and on its impact chain leading to end point effects (=unit effects). Information about the impact chain is recorded in the table Event i.e. which endpoint effects there are, how many unit effects that are caused by the total emission of sulfur dioxide and what contribution to the total unit effects from a unit amount (1 kg) of sulfur dioxide there is.

A list of unit effects and a description of the unit effects are given in the table **UnitEffect**. Various values estimated for the unit effects are given in the table **UnitEffectValue**. Each end point effect can be given several values depending on the valuation method and how it was used. Data about the Valuation methods are listed in the table **ValuationMethod**.

Impact indices are practical to use for decision-makers. Sometimes information are available on amounts of emissions and use of resources. Sometimes, as for designers, information is only available on amounts of material and processes used during the life cycle. For designers it is practical to use impact indices for complex activities, like the production of one kg of some material. For example if a life cycle inventory was made on the production of 1 kg of polyethylene on the Swedish west coast, and the practitioner wanted to make an impact assessment of this, he or she would probably like to use already made impact assessments for emissions and use of resources, available from the impact assessment part of the database. As there are several optional impact assessments that he or she can use, the choice must be recorded somewhere. A table named **AssessmentConnection** is created for this purpose. When the choice has been made an application program may find and present any impact characteristics which is connected to the production of 1 kg of polyethylene.

There are two separate tables used for impact indices. The reason for this is that different information is used for making activity indices and emission/resource indices. An activity impact index is defined through an assessment connection to specific flows and impact indices for emissions/resources. An impact index for an emission/resource is defined by the procedure of impact assessment and the fate in the environment described in the tables '**Event**', '**UnitEffect**' and the valuation of the end point effects described in the tables '**UnitEffectValue**' and '**ValuationMethod**'.

3.4 The value system

The value system is not explicitly modeled in the database model. Decisions based on valuations made during inventories and impact assessments are registered in tables describing the respective procedures. Various valuation methods are registered in the table '**ValuationMethod**'.

4. The database

In general, the data model described in the last chapter can be directly mapped into a similar set of relational database tables; each object in the model will be represented by one table in the database. However, a couple of extra tables will be needed to handle 'one to many' and 'many to one' relationships. Also, a few minor changes will be introduced to avoid redundancy in the database and some will be introduced to support control of database application programs.

4.1 Terminology and Symbols

4.1.1 Table Columns

The columns of the database tables will hold information of three different categories:

1. *Attribute*; data on the object that the table represents. (For example, in a table 'Car', designed to store information on cars, 'Colour' and 'NumberOfDoors' could be relevant attributes.)

2. *Primary key*; a set of one or more columns in a table, holding data which uniquely distinguishes one row from the other. (In the table 'Car', a column 'LicenseNumber' could be the primary key. This would assure that no two cars could have the same license number.)

   Many primary keys will be artificial, that is, they will not be one of the attributes of the table. To make possible data communication between different database implemetations these artificial primary keys should be chosen so that they are globally unique (See proposed method in appendix 2).

3. *Foreign key*; a set of one or more columns in a table, holding data which references the primary key of another table. (For example, in a table 'CarOwner', designed to store information on car owners, a column 'theCar' can keep license numbers of cars stored in the 'Car' table. The result would be that the foreign key 'theCar' relates one car owner with one car.)

4.1.2 Optional Tables and Attributes

For the reason that it is impossible to predict all future applications of the database, it has been considered a good idea to allow changes to it. If done unrestrictedly, such changes could make different database implementations incompatible with each other, meaning that it would be both difficult and expensive to exchange data between the implementations. Instead, the general idea is, that the tables and their attributes described in this documentation should be considered as the rigid database *kernel*. This kernel ought not to be changed, unless there are very good reasons for it. However, a specific implementation could have both additional attributes of the tables described here, as well as they could have additional tables.

Note: There are some tables and some attributes described here, which can not be considered as being part of the rigid kernel. Those tables and those attributes are referred to as *optional*, thrughout the text.

4.1.3 Statistical Representation

Handling of statistical representation of numeric data was described in the last chapter. In the database tables the interval of a numeric entity is named as *qName*Upper and *qName*Lower, or as *qName*Min and *qName*Max, where *qName* is the name of the column containing the quantity. Standard deviation, when such a column exists for a numerical value, is named StandardDev. For denoting whether a value should be interpreted as an arithmetic or as an geometric mean the attributes *qName*Type is used. The types are denoted "a" or "g". respectively.

4.1.4 Symbols used

As a support, when describing the tables, a number of figures are drawn to show relations between the tables. (See example in figure R2, below.) In these figures, tables are represented as rectangles, with the table name at the top. Tables at the scope of the explanation are symbolised by filled gray rectangles, while related tables are drawn as incomplete, transparent rectangles.

Primary keys are placed at the upper section of the rectangles (symbolised by a a key in a keyhole ( )), and foreign keys at the lower section (symbolised by a grey key on white ( ) or a white key on grey ( )). The representations of the two key types are separated by a horizontal line in the rectangle.

Foreign keys' references are drawn as arrows (→) from the foreign key of the referring table, to the primary key of the table to which the reference is directed.

Note: A column may well be part of more than one foreign key, and, in such cases, it is represented more than once in the foreign key section of the rectangle. This is purely notation; in a table there are no more than one column with the same name. Also, a column can be part of both the primary key and a foreign key, but still, in a table it does not exist more than one column with the same name.

Attribute columns are not included in the figures, unless they are part of a primary or foreign key.

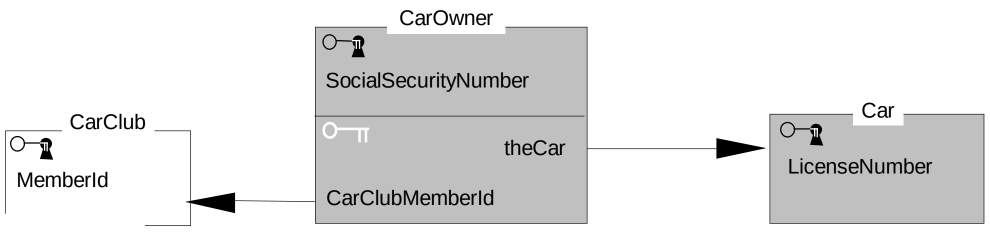


Figure 29. Example of tables, primary keys (○‑🔑), foreign keys (○‑╥) and foreign key references(➞). Tables at focus of explanation are complete and greyed, while related tables outside of the focus are incomplete and transparent.

4.2 The tables of the database

The tables are presented groupwise; conceptually related tables are held together in groups of one or more tables. Each group is presented with a figure, as described in the terminology section above, together with a declarative text in which the attribute columns and the two kinds of keys will be explained.

4.2.1 Qualitative and Quantitative Meta Data

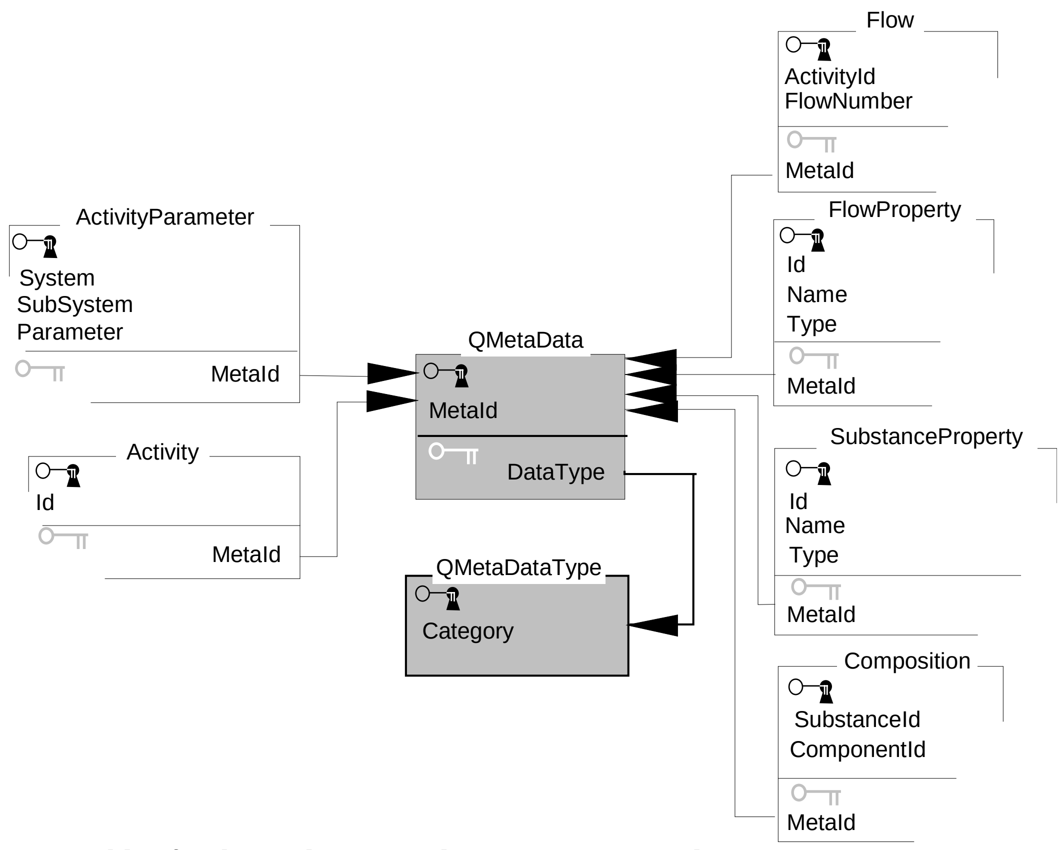


Figure 30. Tables for the qualitative and quantitative meta data.

The table QMetaDataType

The table 'QMetaDataType' is a supporting library table, designed to store the different qualitative types of quantitative data. Each numeric data can be subscribed to one such type.

Attributes:	Category	A category to which numerical data can belong. The table must contain all categories that are used in table QMetaData Example: 'Site specific', 'Average over different similar sites'
	Notes	A definition or description of the category or other relevant information.
Primary key:	Category	
Foreign key:	-	

The table QMetaData

The table 'QMetaData' stores meta data for all types of quantitative data, as described in section Numeric Data in the last chapter.

Attributes:	DataType	A data category to which the numeric data belongs. Example: 'Site specific', 'Average over different similar sites'
	Method	Identifiable description of the method for fabricating the numerical value. Example: 'Continuous DOAS measurements', 'Averaging over values found in literature'.
	DateConceived	Date referring to time of data origin.

		Example: ‘1988-12-01’, ‘1988-12’, ‘1988’
	LiteratureRef	If the data refers to published material, a reference can be given here. Example: ISBN or title+author+year published
	Represents	It is customary to substitute needed data with 'approximatively good' data. Example: Data on a plant A is approximated with data from plant B.
	Notes	If the attributes of the table is not enough to describe the data, it is possible to give an additional note. Example: ‘The date could not be given with any precision, since it was calculated from different measurements over a number of years’.
Primary key:	Id	There is no attribute of this table which can be unique, therefore an artificial primary key must be introduced. Example: ‘xerxesinc-19951201-000000012’
Foreign key:	DataType	References table ‘QMetaDataType’.

4.2.2 The table of Units

Every quantity is related to a unit. To promote a standardised use of units, the database is supplied with a unit library table. Figure 31 shows the tables which uses the units from the 'Unit' library table.

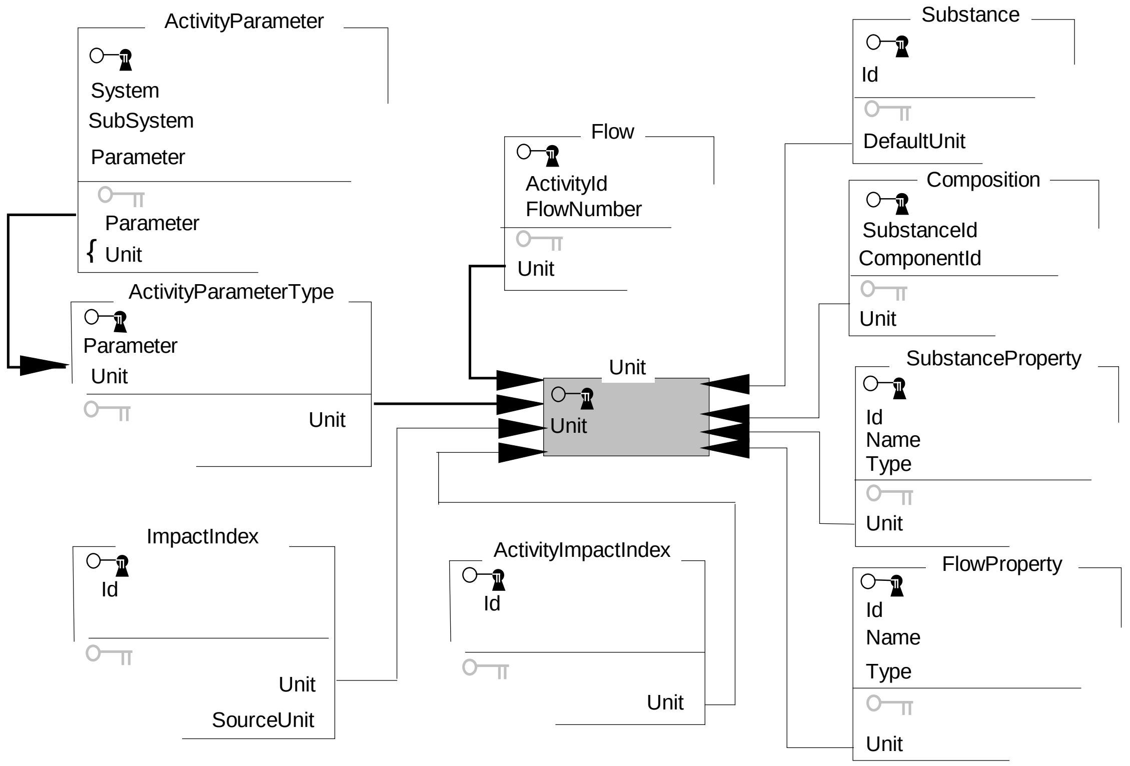


Figure 31. The library table Unit.

The table Unit

Attributes:	Name	The name of the unit is a string of characters, a text string. Example: 'kg/m*m', 'km', 'g'. The use of SI-units, and SI-derived units, is recommended.
Primary key:	Name	
Foreign keys:	-	

The table ActivityParameterType

The table ActivityParameterType stores Activity parameters.

Attributes:	Parameter	A parameterised Activity's parameter name. Example: 'Distance', 'Number of'.
	Unit	The unit of the parameter. Example: See description of table' Unit'.
Primary key:	Parameter Unit	
Foreign keys:	Unit	References table 'Unit'.

4.2.3 The tables representing the database of substances

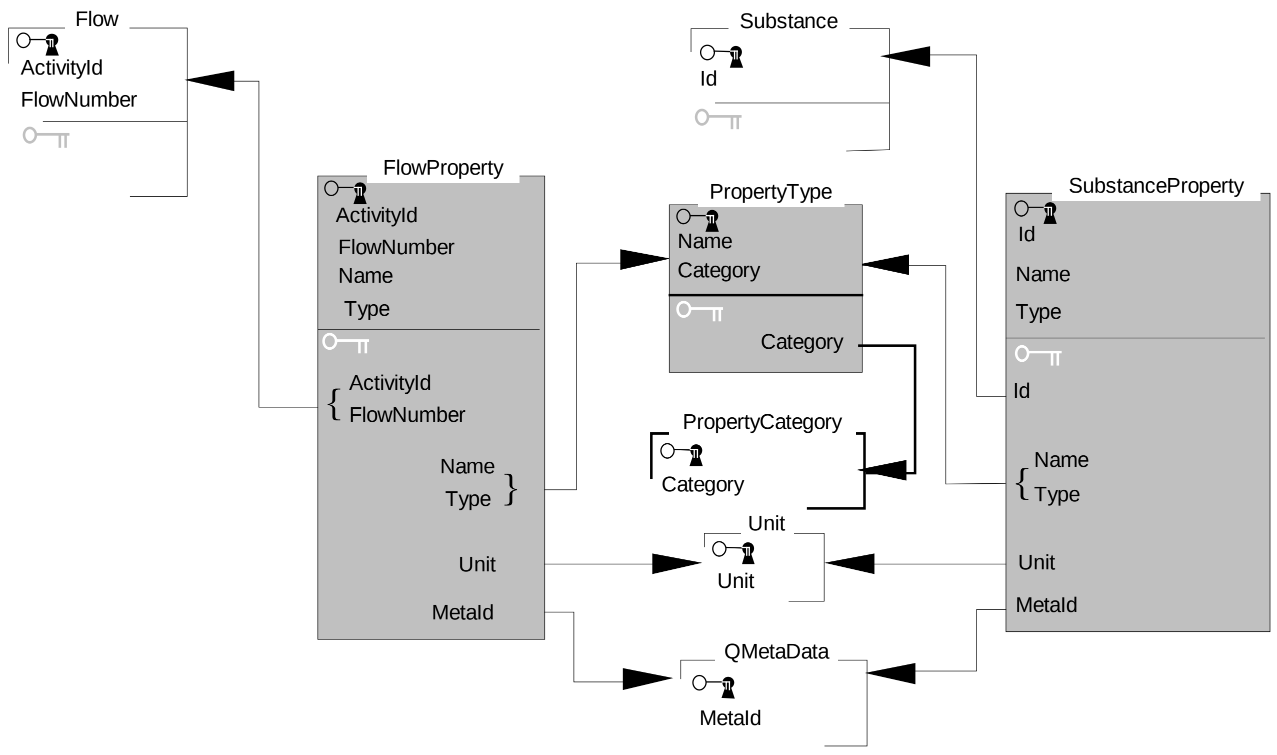


Figure 32. Focusing on flow and substance properties

The table PropertyType

’PropertyType’ is a supporting library table, storing different possible properties for substances and flows.

Attributes:	Name	Name of property. Example: ’Relative value’, ’Mass density’.
	Category	Type of property. Example: ’Economical’, ’Physical’.
	Notes	Supplied if the given attributes is not enough to describe the property type.
Primary key:	Name	
	Category	
Foreign keys:	Category	Optional: References optional table ’PropertyCategory’.

The table SubstanceProperty

’SubstanceProperty’ stores information on properties of a specific substance. Observe that a substance can have an unlimited number of different properties.

Attributes:	Type	Name of property. Example: Same as attribute ’Name’ of table ’PropertyType’.
	Category	Type of property. Example: Same as attribute ’Category’ of ’PropertyType’.
	Quantity	Meaningful properties are quantified. Example: Numerical entity, such as 12.3 .
	QuantityType	See separate discussion on statistical data.
	QuantityMin	See separate discussion on statistical data.
	QuantityMax	See separate discussion on statistical data.
	StandardDev	See separate discussion on statistical data.
	Unit	The quantity’s unit. Example: See table ‘Unit’.
	MetaId	Meta data stored in the general meta data table 'QMetaData'.
Primary key:	SubstanceId	Substance's artificial key. Example: ‘xerxesinc-19951201-000000012’
	Type	
	Category	
Foreign keys:	Type, Category	References table 'PropertyType', guarantees that property is supported in the property library.
	SubstanceId	References table 'Substance', relates property to its substance.
	MetaId	References table 'QMetaData', relates property data to its corresponding meta data.
	Unit	References table 'Unit', guarantees that the unit is supported in the unit library.

The table FlowProperty

'FlowProperty' is similar to the table 'SubstanceProperty' but stores information on properties of flows.

Attributes:	Type	Name of property. Example: Same as attribute 'Name' of table 'PropertyType'.
	Category	Type of property. Example: Same as attribute 'Category' of 'PropertyType'.
	Quantity	Meaningful properties are quantified. Example: Numerical entity, such as 12.3 .
	QuantityType	See separate discussion on statistical data.
	QuantityMin	See separate discussion on statistical data.
	QuantityMax	See separate discussion on statistical data.
	DeviationType	See separate discussion on statistical data
	Unit	The quantity's unit. Example: See table 'Unit'.
	MetaId	Meta data stored in the general meta data table 'QMetaData'.
Primary key:	ActivityId	Activity's artificial key. Example: 'xerxesinc-19951201-000000012'
	FlowNumber	Flow's enumerated key. Example: 1, 2.
	Type	
	Category	
Foreign keys:	Type, Category	References table 'PropertyType', guarantees that property is supported in the property library.
	ActivityId, FlowNumber	References table 'Flow', relates property to its flow.
	MetaId	References table 'QMetaData', relates property data to its corresponding meta data.
	Unit	References table 'Unit', guarantees that the unit is supported in the unit library.

In figure 33 the relations between substances and other close objects are shown.

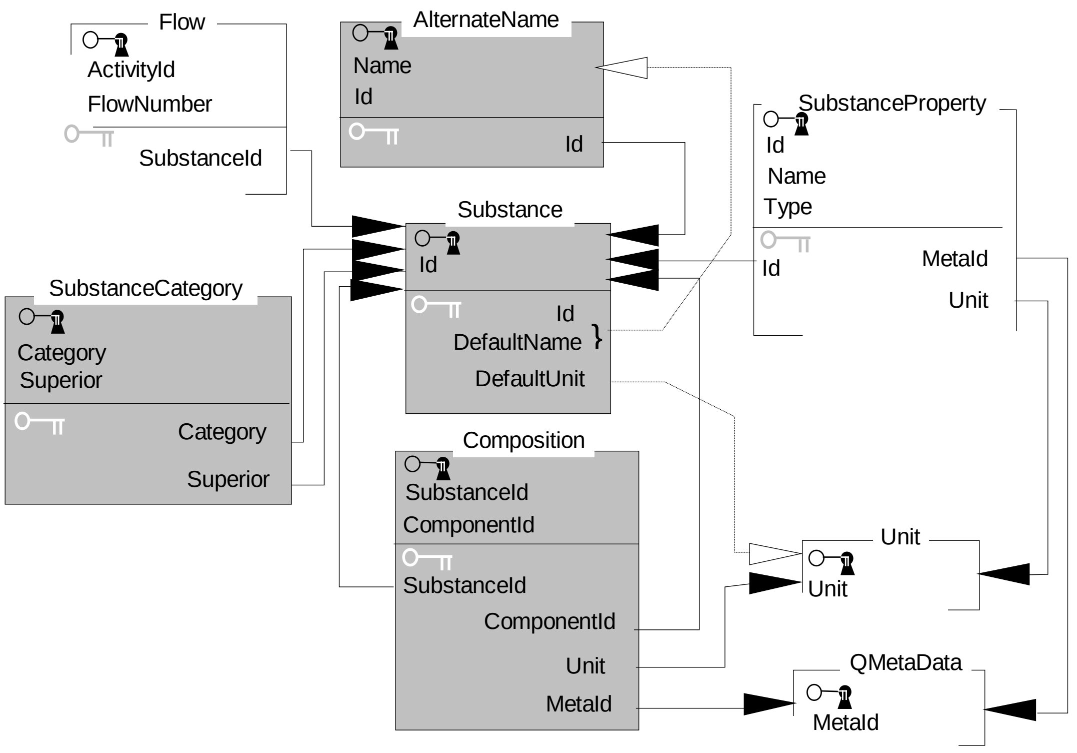


Figure 33. Focusing on the substance

The table AlternateName

A substance can be known by many synonymical names. The number of synonymical names cannot be known in advance, which suggests that synonyms are stored in a separat table.

Attributes:	Name	One of the synonymical names of which a substance is known. Example: ‘PolyVinylChloride’, ‘PVC’
Primary key:	SubstanceId	Substance's artificial key. Example: ‘xerxesinc-19951201-000000012’
	Name	
Foreign keys:	SubstanceId	References table 'Substance'
	Name	Optional: References optional table 'SubstanceDictionary'

The table Substance

The ’Substance’ table is the central table of the substance database, connecting all properties, composites, names etceteras, of a substance together to one specific substance .

Attributes:	DefaultName	The name considered to be the most general name, or the most ordinary name, of the substance. Example: ‘PVC’ is more used than ‘PolyVinylChloride’.
	DefaultUnit	The most ordinary unit by which the substance is measured. Example: ‘ton’ is more used than ‘g’ for steel.
	Notes	A description or definition of the substance.
Primary key:	Id	Since no attributes are unique, an artificial key is needed. Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	DefaultUnit	Optional: References table 'Unit' (Default unit should be a

		supported unit.)
	DefaultName	Optional: References table ’AlternateName’ (Default name should be one of the synonymical names.)

The table SubstanceCategory

The substances can be ordered in a hierarchy. The table ’SubstanceCategory’ stores the structure in this hierarchy by relating a substance to a superior substance.

Attributes:	-	
Primary key:	Superior	Substance’s artificial key. Example: ‘xerxesinc-19951201-000000011’
	Subordinate	Substance's artificial key. Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	Superior	References table 'Substance'
	Subordinate	References table 'Substance' Both 'Superior' and 'Subordinate' are references to the table 'Substance', but they must refer to different substances in the 'Substance' table. This must be checked by either a database routine or by an application program.

The table Composition

A substance can be described as being composed of other substances. The table ’Composition’ relates a substance to its composites, and vice versa.

Attributes:	Quantity	The quantity of the composite. Example: 5.6
	QuantityType	See separate discussion on statistical data.
	QuantityMin	See separate discussion on statistical data.
	QuantityMax	See separate discussion on statistical data.
	StandardDev	See separate discussion on statistical data.
	Unit	The unit by which the quantity is measured. Example: Generally '%' or 'pieces'.
	MetaId	Meta data stored in the general meta data table 'QMetaData'.
Primary key:	SubstanceId	Substance's artificial key. Example: ‘xerxesinc-19951201-000000011’
	ComponentId	Substance's artificial key. Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	SubstanceId	References table 'Substance', the substance which the composite is part of.
	ComponentId	References table 'Substance', the substance which the composite is (all composites are themsleves substances.)
	MetaId	References table 'QMetaData'

4.2.4 The table of addresses and names; JuridicalPerson

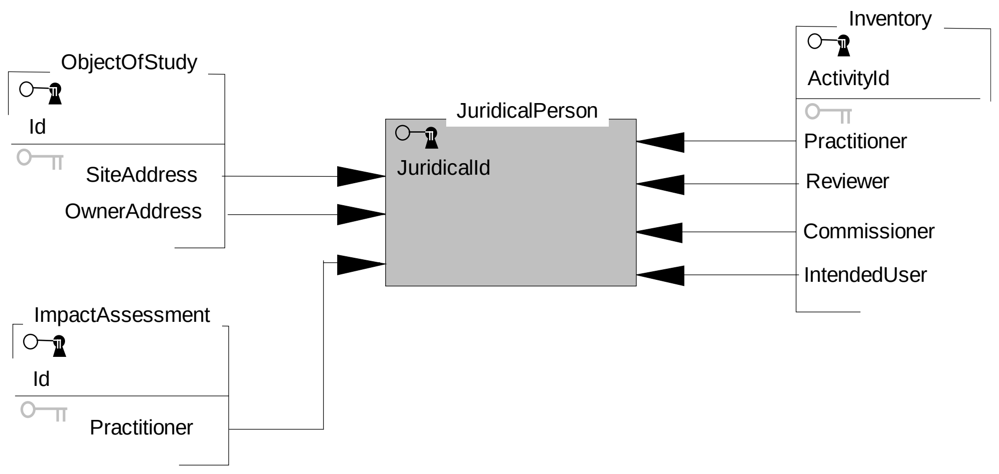


Figure 34.

The table JuridicalPerson

’JuridicalPerson’ is a general table for storing addresses and names. Information on companies, sites, persons and institutions are stored in the same table.

Attributes:	Name	Name of person, plant, institiution, or what ever is appropriate. Example: ‘Sven Svensson’, ‘STORA’
	MailAddress	Address to person, plant, institiution, or what ever is appropriate. Example: ‘Storgatan 3, 123 45 Storköping, Sweden’, ‘Sweden’
	Telephone	Telephone number to person, plant, institiution, or what ever is appropriate. Example: ‘+46-31-912 34 56’
	Fax	Fax-number to person, plant, institiution, or what ever is appropriate. Example: ‘+46-31-912 34 65’
	EmailAddress	E-mail address to person, plant, institiution, or what ever is appropriate. Example: ‘sven@server.net.se’
Primary key:	Id	Since names and addresses are not unique, an artifical primary key is needed. Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	-	

4.2.5 The tables representing the Object of Study

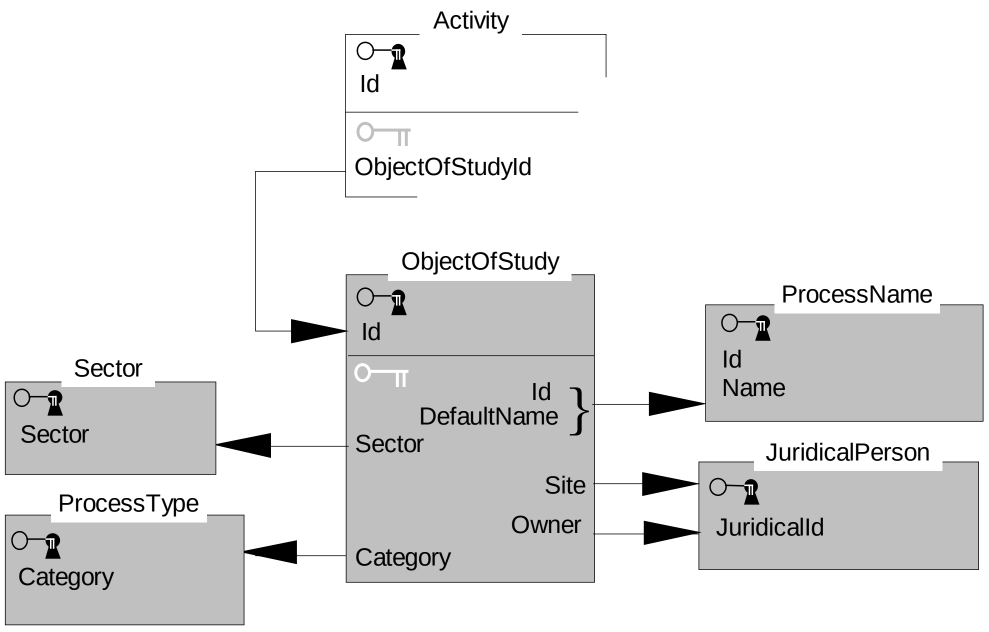


Figure 35.

The table Sector

To allow queries based on market sectors of objects of studies and Activities, a hierarchical structure of such sectors can be built. In a huge database, objects of studies related to a specific market sector can be more easily found.

Attributes:	Name	Name of market sector. Example: ‘Iron and ore’, ‘Forestral’
	Notes	If needed, a note on the sector can be given. Example: ‘I am not too sure that this sector name is a good choice, but I found no better in hierarchy list.’
Primary key:	Name	
Foreign keys:	Superior	The hierarchical structure is built by relating one sector name to another sector name in the same table. The second sector being superior to the former. (A foreign key of this kind is not a true foreign key. To maintain the hierarchical chain, the constraint must be checked explicitly.)

The table ProcessType

Objects of studies can be structured in a hierarchy in order of their processes. As for market sectors, this hierarchy can make database querying more efficient.

Attributes:	Category	Category of process. Example: 'Pulp production'

	Notes	If 'Category' is not descriptive enough, 'Notes' can be used to supply further explanation. Example: As pulp is understood chemical pulp, thermomechanical pulp and mixtures thereof.
Primary key:	Category	
Foreign keys:	Superior	This key works as does 'Superior' of the table 'Sector', that is, it builds a hierarchical structure by relating a category to a superior level category. (A foreign key of this kind is not a true foreign key. To maintain the hierarchical chain, the constraint must be checked explicitly.)

The table ProcessName

As a substance can be known by alternative synonymical names, also processes can be known by more than one name.

Attributes:	Name	The alternate name. Example: 'Incineration', 'Combustion'
Primary key:	Id	Artificial key of table 'ObjectOfStudy'. Example: 'xerxesinc-19951201-000000012'
	Name	
Foreign keys:	Id	References the table 'ObjectOfStudy'.
	Name	Optional: References an optional table 'ProcessDictionary'.

The table ObjectOfStudy

Attributes:	Name	The most general, or the most ordinary name of the object of study. Example: 'Combustion'.
	Sector	Reference to market sector of table 'Sector'. Example: 'Iron and ore', 'Forestral'
	Site	Reference to address stored in table 'JuridicalPerson'. Example: 'xerxesinc-19951201-000000012'
	Owner	Reference to address stored in table 'JuridicalPerson'. Example: 'xerxesinc-19951201-000000012'
	Category	Reference to activity type in table 'ProcessType'.
	Function	Description of the process or function of the object of study.
Primary key:	Id	Since no attribute is unique, an artificial key is needed. Example: 'xerxesinc-19951201-000000012'
Foreign keys:	Sector	References table 'Sector'.
	Category	References table 'ProcessType'.
	Site	References table 'JuridicalPerson'.
	Owner	References table 'JuridicalPerson'.

4.2.6 The tables representing Environment and Geography

For evaluation of environmental impact at local and/or regional levels, flows into or out of Activities must be supplied with information on geographical location and/or environmental type of the site where, for example, resources are extracted or emissions and waste are released.

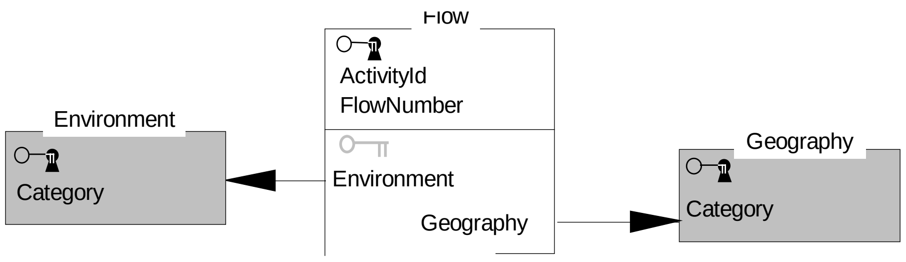


Figure 36

The table Environment

Environment types can be categorised in hierarchical structures.

Attributes:	Name	Name of environment type. Example: 'Sweet water', which is superior to types 'River' and 'Lake', the latter in turn being superior to 'Eutrophic lake'.
	Description	A description that allows the user to decide whether a certain environment belongs to the group of environments in the database or not.
	Notes	If needed, a note on environment type.
Primary key:	Name	
Foreign keys:	Superior	The hierarchical structure is built by relating one environment type name to another type name in the same table. The second type being superior to the former. (A foreign key of this kind is not a true foreign key. To maintain the hierarchical chain, the constraint must be checked explicitly.)

The table Geography

Geography can be structured in hierarchical levels, where the structural depth goes from global at the highest level, through regional and local levels, down to precise coordinates. Typically a certain database technology, Geographical Information System (GIS), is used for this kind of information. To prepare for such technology to be joined with to this database, an artificial key has been introduced. In the future this key can be mapped to correspond to a GIS-system key.

Attributes:	AreaName	The name of the geographical area. Example: 'Europe', 'Skåne'
	AreaType	The macro level of the area. Example: 'Nation', 'Region', 'Coordinate'
	Notes	If needed, a note on the geographical information.
Primary key:	Id	Artificial key, to support for artificial GIS key mapping. Example: 'xerxesinc-19951201-000000012'
Foreign keys:	SuperiorArea	The hierarchical structure is built by relating one geographical Id to another Id in the same table. The second being superior to the former. (A foreign key of this kind is not a true foreign key. To maintain the hierarchical chain, the constraint must be checked explicitly.)

4.2.7 The tables representing the Activity and Flow concept:

In this section the tables of the central concepts will be presented: the 'Activity' and 'Flow' tables, together with structure preserving tables, designed store flowcharts structures.

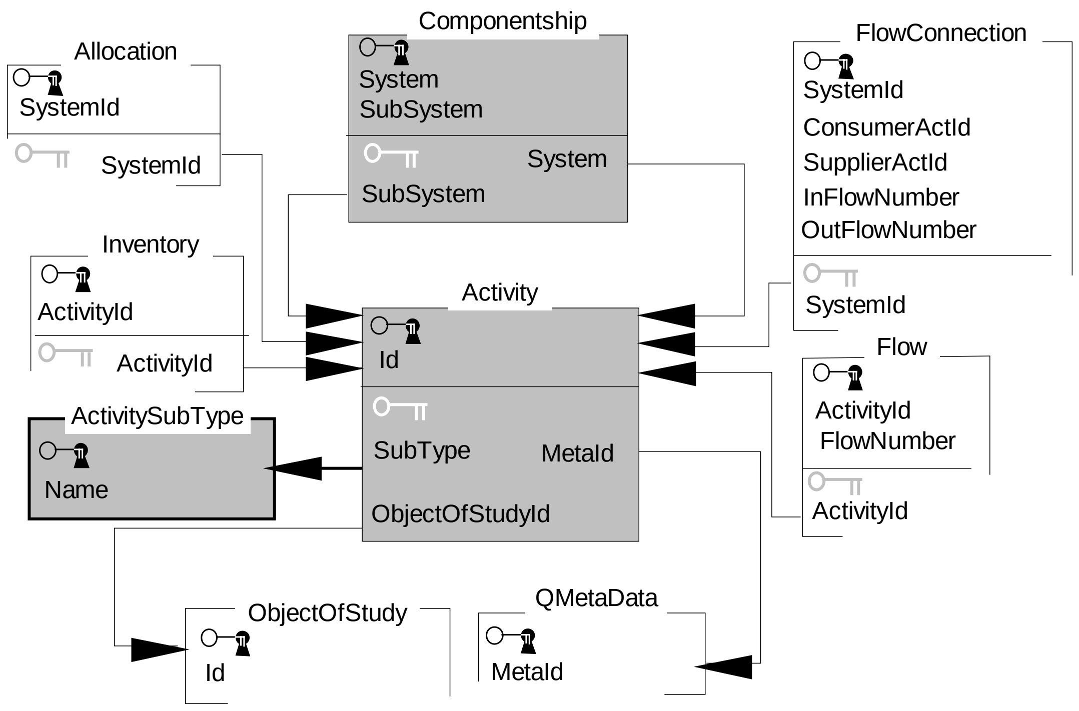


Figure 37

The table Activity

Since most data on an Activity is stored in the table 'ObjectOfStudy' and in other tables, the 'Activity' table includes mostly flags to support application programs.

Attributes:	ObjectId	The description of the Activity lies in the 'ObjectOfStudy' table.
	SubType	Distinguishes different types of Activities from each other. The subtyping should be used to control application programs and therefore the types should not be changed by application program users. Also, new types should not be introduced unless there are good reasons for it. The set of different allowed types are stored in the table 'ActivitySubtype', so SubType is a reference to that table. Example: 'Transport' or 'Process' (By the time, there might come a few more Activity subtypes.)
	Finished	A flag which tells if the Activity is considered to be finished by the 'owner'. Example: Flag, 'Y', for yes and 'N' for no.
	Aggregated	Optional: An application support-flag which tells whether the internal flows has been calculated and aggregated into external flows. Used to control application program-flow[1]. Example: Flag, 'Y', for yes and 'N' for no.

[1] Several other both attributes and tables might be introduced to support database applications.

	MetaId	The meta data lies in the 'QMetaData' table, related from the 'Activity' table or from the 'Flow' table. The reason for having this reference is that an activity could be associated with several data of the same quality, and that it may sometimes be more convenient to describe the metadata associated with an activity than with each of the flows associated with the activity.
Primary key:	Id	Artificial primary key. Example: 'xerxesinc-19951201-000000012'
Foreign keys:	ObjectId	References table 'ObjectOfStudy'.
	MetaId	References table 'QMetaData'.
	SubType	References table 'ActivitySubType'.

The table ActivitySubType

Attributes:	Name	The name of the subtype of the Activity. Example: 'Transport', 'Process'.
Primary key:	Name	
Foreign key:	-	

The table Componentship

The table 'Componentship' is a pure relational table, referencing internal Activities to the Acivity which hosts them as internal. An internal Activity can exist only once in a host Activity.

Attributes:	-	
Primary key:	SystemId	The host Activity. Example: 'xerxesinc-19951201-000000012'
	SubSystemId	The internal Activity. Example: 'xerxesinc-19951201-000000012'
Foreign keys:	SystemId	References table 'Activity'.
	SubSystemId	References table 'Activity'.

The Flow tables

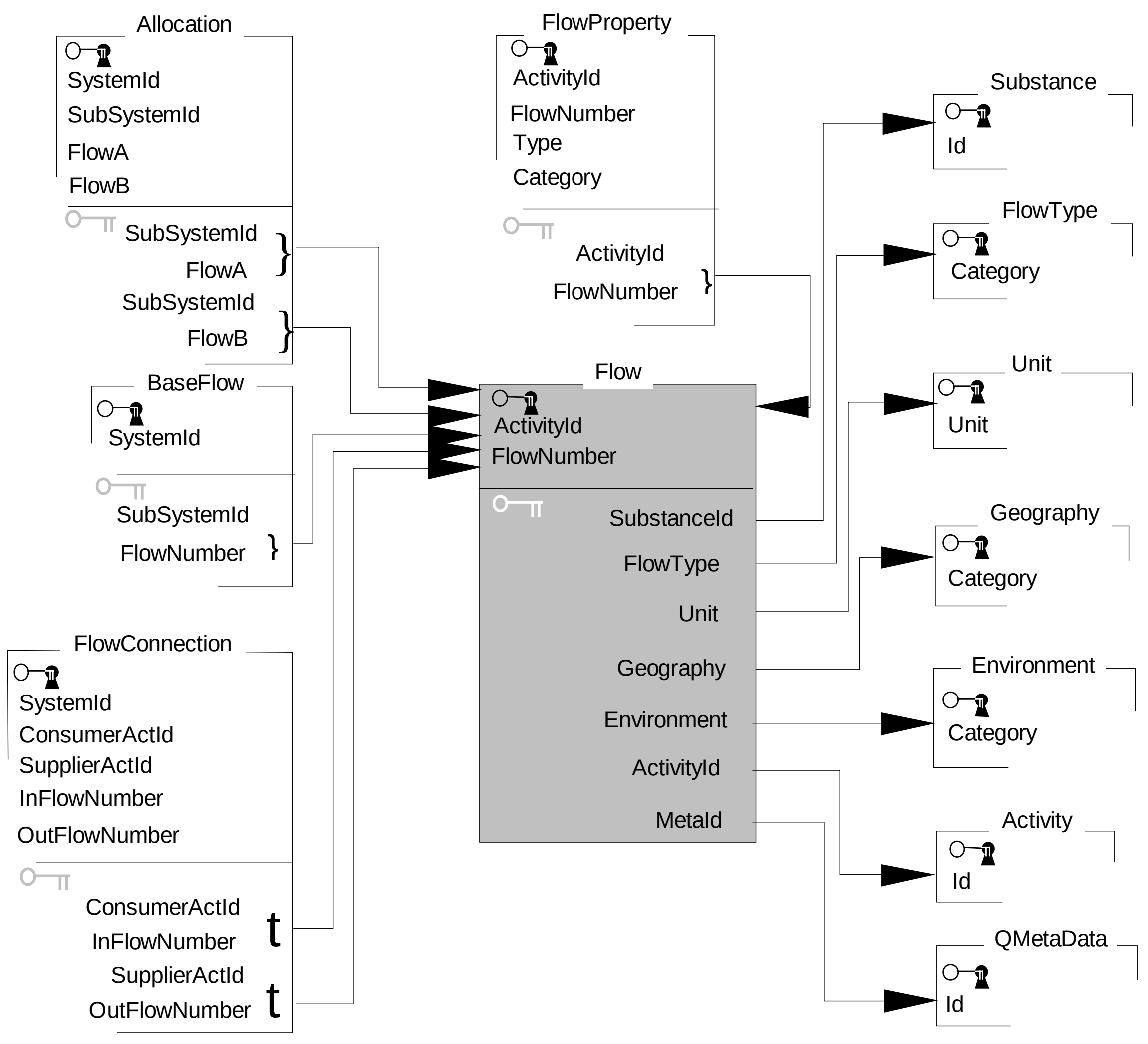


Figure 38

The table Flow

Attributes:	SubType	An important attribute which makes it possible to distinguish between inflows and outflows. Example: Either the flow is an 'Input' or is it an 'Output'.
	SubstanceId	Relates flow to what kind of a substance it is. (Observe that most descriptive information on the flow is stored in the 'Substance' table.)
	Quantity	The amount of the flow. Example: 5.6
	QuantityType	See separate discussion on statistical data.
	QuantityMin	See separate discussion on statistical data.
	QuantityMax	See separate discussion on statistical data.
	StandardDev	See separate discussion on statistical data.

	Unit	The unit of the flow. Example: See table ‘Unit’.
	Category	Information on to which category a substance belongs. Example: 'Raw material', 'Energy', 'Product', 'Emission', 'Waste’
	MetaID	Meta data is stored in separate table 'QMetaData'.
	ImpactMedia	Relation to table 'Environment'. Example: ‘Sweet water’, ‘spot market’ (if product)
	ImpactRegion	Relation to table 'Geography'. Can also be used to store information on products. Example: ‘xerxesinc-19951201-000000012’
Primary key:	ActivityId	Artificial key of table 'Activity'. Example: ‘xerxesinc-19951201-000000012’
	FlowNumber	Enumerated key, increases from 1 for each Activity. Example: 1, 2, ...
Foreign keys:	SubstanceId	References table 'Substance'.
	Unit	References table 'Unit'.
	MetaId	References table 'QMetaData'.
	ActivityId	References table 'Activity'.
	Category	References table 'FlowType'.
	ImpactMedia	References table 'Environment'.
	ImpactRegion	References table 'Geography'.

4.2.8 Linking Activity, Flow and Internal Activity

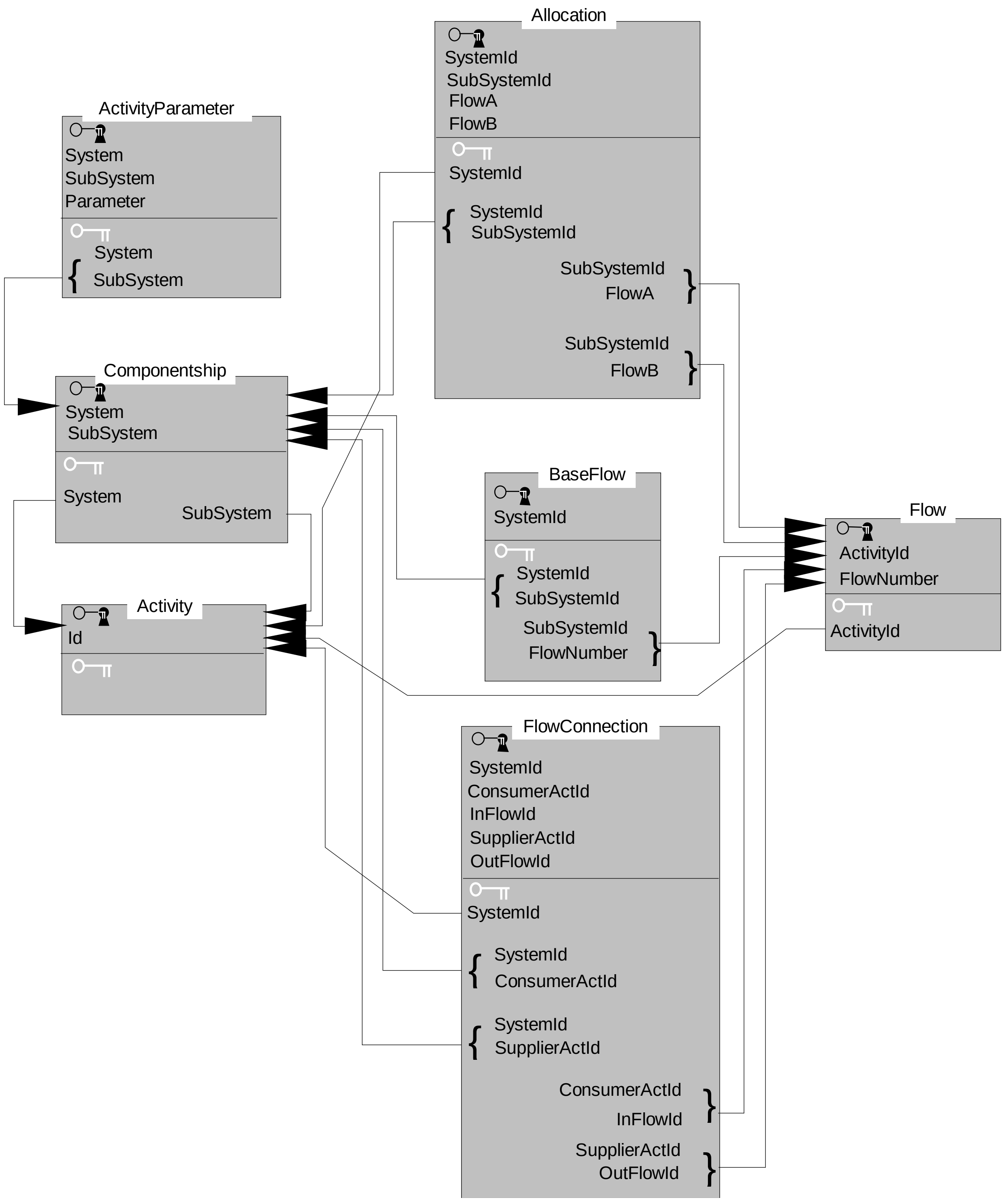


Figure 39

As a refresher, the primary key and the foreign keys of the tables 'Activity', 'Componentship' and 'Flow' are presented again here.

The table Flow:

Primary key:	ActivityId	
	FlowNumber	
Foreign keys:	SubstanceId	References table 'Substance'.
	Unit	References table 'Unit'.
	MetaId	References table 'QMetaData'.
	ActivityId	References table 'Activity'.
	Category	References table 'FlowType'.
	ImpactMedia	References table 'Environment'.
	ImpactRegion	References table 'Geography'.

The table Activity:

Primary key:	Id	
Foreign keys:	ObjectId	References table 'ObjectOfStudy'.
	MetaId	References table 'QMetaData'.

The table Componentship:

Primary key:	SystemId	
	SubSystemId	
Foreign keys:	SystemId	References table 'Activity'.
	SubSystemId	References table 'Activity'.

The table FlowConnection

In a flow chart, a flow connection connects two flows, one inflow and one outflow (to the connection). The inflow to the connection could come exclusively from one Activity or be one of several flows coming from other activities. The outflow can in the same way be the only flow to an Activity or it may be one of several flows to different Activities. Sometimes the ratio between these flows are constant and may then be stored in the table. The database restricts connections to occur only between Activities internal of a host Activity, and only once in each host.

Attributes:	QuantityType	See separate discussion on statistical data.
	InRatio	The inflow divided by the sum of all flows which have connections to the same outflow. Observe that it should be checked that the sum of the portions of the inflows to one connection is equal to 1. Example: 0.65
	InRatioUpper	See separate discussion on statistical data.
	InRatioLower	See separate discussion on statistical data.
	InRatioStandardDev	See separate discussion on statistical data.
	OutRatio	The outflow divided by the sum of all flows which have connections to the same inflow. Observe that it should be checked that the sum of the

		portions of the inflows to one connection is equal to 1. Example: 1
	OutRatioUpper	See separate discussion on statistical data.
	OutRatioLower	See separate discussion on statistical data.
	OutRatioStandardDev	See separate discussion on statistical data.
Primary key:	InFlowNumber	Reference to the inflow. Example: 1, 2, ...
	ConsumerActId	Reference to the Activity ‘having’ the inflow. Example: ‘xerxesinc-19951201-000000012’
	SystemId	Reference to the Activity which is hosting the two Activities which has the two connected flows. Example: ‘xerxesinc-19951201-000000012’
	SupplierActId	
	OutFlowNumber	
Foreign keys:	SystemId	References table 'Activity'.
	ConsumerActId, InFlowNumber	References table 'Flow'.
	SupplierActId, OutFlowNumber	References table 'Flow'.
	SystemId, ConsumerActId	References table 'Componentship'.
	SystemId, SupplierActId	References table 'Componentship'.

The table ActivityParameter

Figure 31 shows that the ’ActivityParameter’ relates to ’ActivityParameterType’, and that the ’ActivityParameterType’ table is related to the ’Unit’ table.

Attributes:	Type	The name of the parameter. Example: 'Distance', ‘Number of’
	Value	The amount or quantity of the parameter. Example: 12.0
	ValueType	See separate discussion on statistical data.
	ValueMin	See separate discussion on statistical data.
	ValueMax	See separate discussion on statistical data.
	ValueStandardDev	See separate discussion on statistical data.
	MetaID	Meta data is stored in general table 'QMetaData'.
	Unit	The unit of the parameter. Example: 'km', ‘pieces’
Primary key:	SystemId	Artificial key of host Activity. Example: ‘xerxesinc-19951201-000000012’
	SubSystemId	Artificial key of internal Activity. Example: ‘xerxesinc-19951201-000000012’
	Type	Since it is possible that one parameterised Activity has more than one parameter, the parameter type must be part of the primary key. It cannot, however, have more than one parameter of one type, and therefore the unit of the parameter cannot be part of the primary key.
Foreign keys:	SystemId, SubSystemId	References table 'Componentship'.
	MetaId	References table 'QMetaData'.
	Type, Unit	References table 'ActivityParameterType.'

The table Allocation

Allocations have no meaning for an Activity, unless it is is part of a study. Therefore, the table which stores the allocational rules is related only to internal Activities and their flows.

Note: It should be stressed that the allocational rules handled by this table are strictly mathematical. And it should also be stressed that this table is not the final solution to the handling of mathematical allocations in an Activity[2].

Attributes:	AOverB	Relates two flows, A and B, to each other. Example: If no allocation considerations are taken, AoverB need not contain any value. All flows of all Activities will instead be recalculated on basis of the baseflow of the system. If flow A is ‘cut out’ from the system, the ratio AOverB contains 0 (zero). If flow B does not fully contribute to the full amount of flow A, the allocated contribution should be stored as the ratio AOverB.
	QuantityType	See separate discussion on statistical data.
	UpperAOverB	See separate discussion on statistical data.
	LowerAOverB	See separate discussion on statistical data.
	StandardevAOverB	See separate discussion on statistical data.
	Notes	If a note is needed regarding the allocation decision, it can be supplied here. This is useful if a certain decision deviates from the overall allocation principle of the study.
Primary key:	SystemId	The Activity hosting the internal Activity. Example: ‘xerxesinc-19951201-000000012’
	SubSystemId	The internal Activity. Example: ‘xerxesinc-19951201-000000012’
	FlowA	The flow A of the internal Activity. Example: 1, 2, ...
	FlowB	The flow B of the internal Activity. Example: 1, 2, ...
Foreign keys:	SystemId, SubSystemId	References table 'Componentship'.
	SubSystemId, FlowA	References table 'Flow'.
	SubSystemId, FlowB	References table 'Flow'.

The table BaseFlow

To allow for good transparancy and reproduction of the results of the calculation on a system it is necessary to identify the flow relative to which other flows are calculated . The base flow also serves as a reference to the flow corresponding to the functional unit of the study.

Attributes:	-	
Primary key:	SystemId	
Foreign keys:	SystemId, SubSystemId	References table 'Componentship'.
	SubSystemId, FlowNumber	References table 'Flow'.

[2] It could be argued that an Activity should be supplied with default settings on the allocational rules, and that there should be a method for aggregating similar rules to an Activity, so that all relations between the flows should not be needed. The table presented here should be considered a design sketch, which allows for storage of the most necessary data.

4.2.9 The table for documenting the Inventory

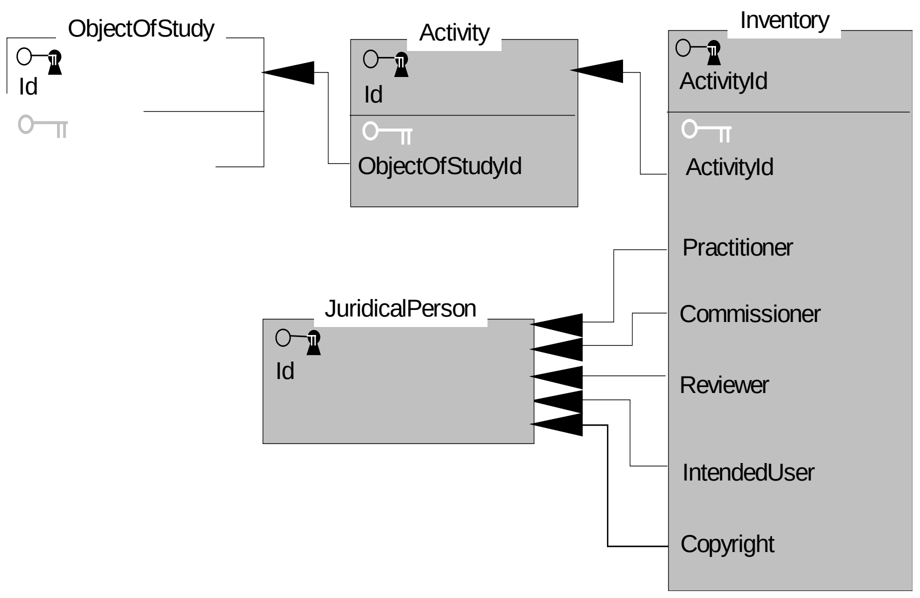


Figure 40

As a refresher, the primary keys of the tables ’Activity’ and ’JuridicalPerson’ are supplied here.

The table Activity

Attributes:	See separate description of table ’Activity’.	
Primary key:	Id	
Foreign keys:	ObjectId	References table ’ObjectOfStudy’.
	MetaId	References table ’QMetaData’.

The table JuridicalPerson

Attributes:	See separate description of table ’JuridicalPerson’.
Primary key:	Id
Foreign keys:	-

The table Inventory

Attributes:	Practitioner	The practitioner is the juridical person responsible for the study. Example: ‘xerxesinc-19951201-000000012’
	Reviewer	The reviewer is a juridical person representative to the review panel during the study. Example: ‘xerxesinc-19951201-000000012’
	Commissioner	The commissioner is the juridical person initiating the study. Example: ‘xerxesinc-19951201-000000012’
	IntendedUser	The intended user is the person, organisation or group of users, whom the study was made to serve.
	GeneralPurpose	The general purpose is the abstract idea which initiatet the study. Example: Typically the idea is expressed in terms such as "We must

		do something about all the waste produced by package material."
	DetailedPurpose	The detailed purpose is the answer to a formulated question. Example:"Does it pay, environmentally, to recycle packages?", or, "Which package type is the, environmentally, optimal one?".
	FunctionalUnit	The functional unit is a short form of the purpose of a whole technical system studied. Example: ‘Packages for 33cl of beverage for end user’
	FUExplanation	(FU=FunctionalUnit) An explanation of the meaning of the functional unit. Example: If there might be any ambiguities with the understanding of the calculational base of a study, this should be explained here.
	Copyright	A reference to a juridical person who has the publishing rights to the data set of the study.
	Availability	Information on how the set of data can be used and presented. Example: ‘Data must not be presented to the public without the copyright holder’s control over its use.’, ‘The result of this study may not be published unless together with the the complete study, with data sources and all.’ ‘Confidential’ , ‘Open’.
	Publication	An effective identity to a printed publication of the study, if such exists. Example: ISBN or title+author+ year published.
	DateCompleted	The date when the study was completed.
	Applicability	Under what circumstances can the study be applicable? Example: Assumptions which must hold. Technological requirements.
	Data	General summary of the data use, data quality and other data aspects of the study. Example: Overall descriptions that can be made regarding the data, average age, typical precision, typical source.
	LateralExpansion	If any external systems has been included, in calculational form, or in any other form that can not be seen from the technical subsystem's point of view. Such expansions might be done if the data inserted into the database is inherited in the form of complete or partial study performed somewhere else.
	Allocations	General principles used for allocations. Example: ‘In general the allocations are based on relative mass, since...’
	NatureBoundary	Where has the boundaries been drawn towards the nature system? Example: ‘We have not considered leakage out from waste depositions. Emissions to air has been considered as such, without any considerations to what happens when they fall to ground...’
	TimeBoundary	What time horizons has been applied? Example: ‘It has been assumed that the product is used for seven years, and that it exists at the same time as do the mines where the raw material are mined...’
	GeographyBoundary	What geographical horisons has been applied? Example: ‘For reasons of practicability, all material flows has been followed only to the borders of Europe...’
	OtherBoundaries	Other system boundaries, such as not included subsystems.
	Notes	If the attributes above is not enough or appropriate for the need.
Primary key:	ActivityId	
Foreign keys:	Practitioner	References table 'JuridicalPerson'.
	Reviewer	References table 'JuridicalPerson'.
	Commissioner	References table 'JuridicalPerson'.
	IntendedUser	References table 'JuridicalPerson'.

4.2.10 The tables related to Impact Assessment

The table ImpactAssessment

The impact assessment is an activity for which some records may be of interest.

Attributes:	Notes	Any considerations or literature references that may be of importance for the impact assessment
Primary key	Id	Identifies the impact assessment. Example: 'xerxesinc-19951201-000000012'
Foreign key	Practitioner	References table 'JuridicalPerson' Example: 'xerxesinc-19951201-000000012'

The table ImpactIndex

The table impact index contains information on impact indices for inventory parameters such as emissions and use of resources. It should not to be confused with the table ActivityImpactIndex which contains indices for technical activities such as production of 1 kg of steel and which data is based on the impact indices for emissions etc. plus inventory data.

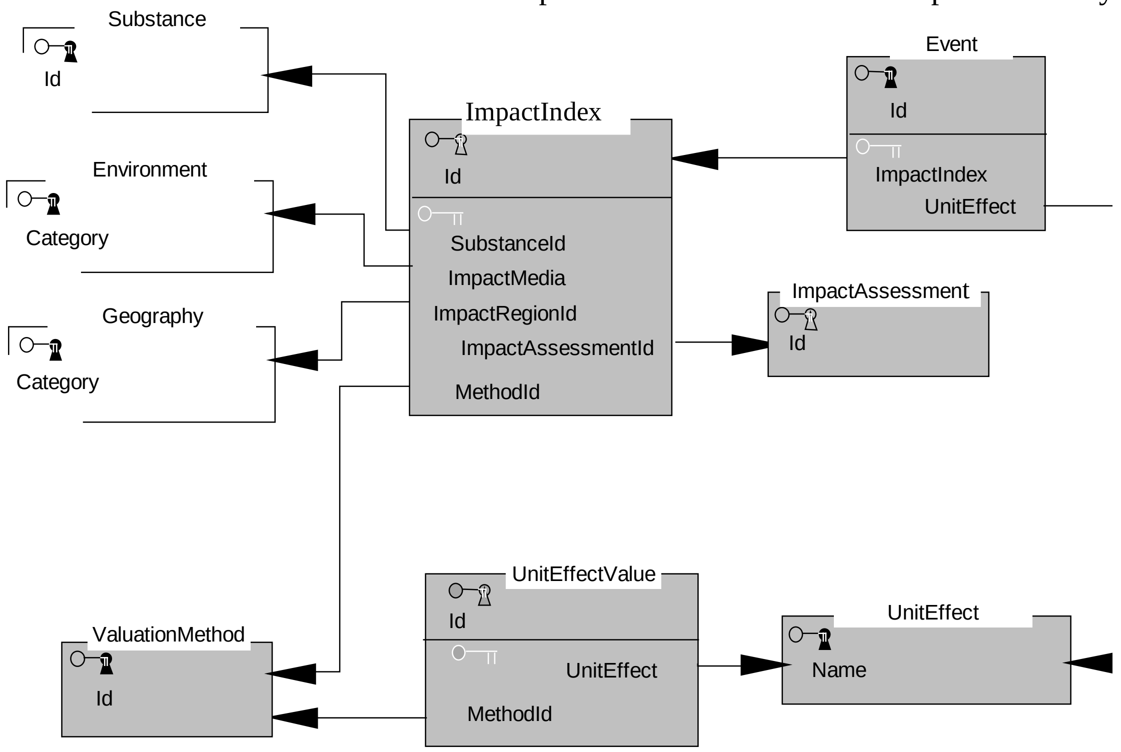


Figure 41

Attributes:	Time period	The time period it represents. Example: 1 hour, 100 years
	IndexValue	Calculated value Example: 3.2

	BaseUnit	Example: kg
	SourceStrengthMin	The minimum size of the emission or resource use for the index to be valid Example: 4
	SourceStrengthMax	Example: 50
	SourceUnit	Example g/s
	TimeFrameStart	The first moment from which the index value is valid Example 1990-02-13
	TimeFrameEnd	Example: 1995-12-31
	Notes	Any other information
Primary key:	Id	Example:'xerxesinc-199512202--00000014'
Foreign key:	ImpactMedia	References to table 'Environment'
	ImpactRegion	References table 'Geography'
	SubstanceId	References table 'Substance'
	ImpactAssessmentId	References table 'ImpactAssessment
	MethodId	References table 'ValuationMethod'
	BaseUnit	References table 'Unit'
	SourceUnit	References table 'Unit'

The table Event

The table Event belongs to the environmental system and is a parallel to the table activity in the technological system. It represents changes in the environment caused by human activities. The table describes the nature of the event and the extension of and contribution to the event caused by an activity described with an impact index.

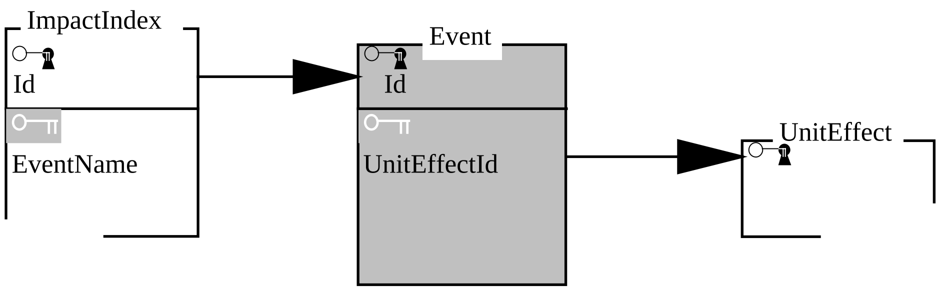


Figure 42

Attributes:	Name	Name of event. Example: increased lung cancer
	UnitEffect	End point effect caused by the event. Example: Human health
	Region	Environmental region, where the effect occur. Example: boreal region
	LocalEnvironment	Local environment, where the effect occur. Example: 'all', 'urban', 'urban ground level'
	ReportingArea	Administrative area for which data are applicable. Example: 'Portugal'
	TimeFrameStart	The event or population of events described may occur during a certain time period. TimeFrameStart is the starting date of this period. Example 1993-01-01
	TimeFrameStop	End of event time frame. Example 1993-12-31

	ExtensionAverage	Total number of unit effects there are due to activities and impact of the kind specified elsewhere. Best estimate of average.
	ExtensionSD	Standard deviation of extension representing uncertainty in estimation and real variations amongst the population of data on extension.
	ExtensionMin	Minimum extension
	ExtensionMax	Maximal extension
	ExtensionQType	Arithmetic or geometric average and SD. a or g.
	ContributionAverage	Average of fraction of ExtensionAverage that is caused by the activity given an impact index.
	ContributionSD	Standard deviation of contribution representing uncertainty in estimation and real variations amongst the population of data on contribution.
	ContributionMin	Minimum contribution
	ContributionMax	Maximum contribution
	ContributionQType;	Arithmetic or geometric standard deviation. a or g
	Notes	Any other relevant information
Primary key:	Id	Artificial primary key
Foreign key:	UnitEffect	References table 'UnitEffect'
	Region	References table 'Environment'
	LocalEnvironment	References table 'Environment'
	ReportingArea	References table 'Geography'
	ImpactIndexId	References table 'ImpactIndex'

The table UnitEffect

The table UnitEffect identifies the types of Unit effects used.

Attributes:	SafeGuardSubject	Safe guard subject to which the unit effect is a defined change. Example: Human health
	Unit	Unit used Example: man-year of morbidity
	Notes	Any other relevant information
Primary key:	Name	Name of unit effect Example: "excess mortality".

The UnitEffectValue

The end point in a cause effect chain is described in terms of unit effects. Unit effects may be given value. Various valuations may be made on the same unit effects.

Attributes:	UnitEffect	End point effect caused by the event. Example: Human health
	Value	Average value given to the unit effect
	StandardDeviation	Standard deviation of unit effect values.
	ValueType	Arithmetic or geometric average and standard deviation. a or g
	ValueMax	Highest value in a sample of values.
	ValueMin	Lowest value in a sample of values.
	Notes	Any other relevant information
Primary key	Id	Identification of value given.
Foreign key:	UnitEffect	References table 'UnitEffect'
	MethodId	References table 'ValuationMethod'

The table ValuationMethod

The valuation method and some characteristics of the valuation activity is described in the table ValuationMethod.

Attributes:	Unit	Unit in which value is given. Example: ELU
	Name	Name of valuation method Example: EPS enviroaccounting method, Ecoscarcity method
	Version	Version of the valuation method
	References	Literature reference
	Notes	Any other relevant information
Primary key:	ValuationMethodId	Id for method used for valuation

The table AssessmentConnection

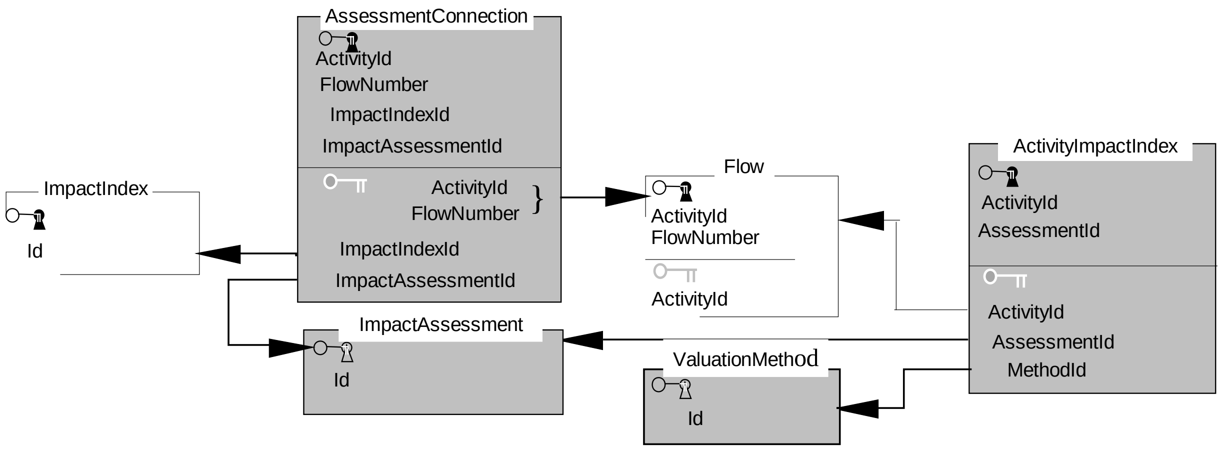


Figure 43

At the assessment connection it is decided which flows from a human activity that should be connected to which impact indices for emissions and use of resources in order to create an ActivityImpactIndex

Attributes:	Notes	Any considerations or litterature references that may be of importance for the assessment connection
Primary keys:	ActivityId	The activity to be assessed by an index Example: ‘xerxesinc-19951201-000000012’
	FlowNumber	Flows involved in the activity Example: 1, 2, 3 or 4....
	ImpactIndexId	The impact index used together with the flow to make an assessment or build an index. Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	ActivityId,FlowNumber;	References table 'Flow'
	ImpactIndexId	References table 'ImpactIndex'
	ImpactAssessmentId	References table ‘ImpactAssessment’

The table ActivityImpactIndex

Specific impact values from human activities are stored as activity impact indices.

Attributes:	IndexValue	Index value for an activity with a complex emission and resource use profile Example: 2.4
	BaseUnit	Index base unit. Example: ‘kg’, ‘m^2’ (the index may thus be 2.4 ELU/kg or ELU/m^2)
	Notes	Any other relevant information
Primary key:	Id	Example: ‘xerxesinc-19951201-000000012’
Foreign keys:	SubstanceId	References table 'Substance'
	MethodId	References table 'ValuationMethod'
	ActivityId, FlowNumber	References table 'Flow'
	AssessmentId	References table 'ImpactAssessment'.
	BaseUnit	References table ‘Unit’

5. Applications

5.1 User interfaces and tools

To manage the data in the database it will be necessary to develop a number of user interfaces. These interfaces can be designed at different levels of ambitions, with different functionality.

5.1.1 Simple database update-programs

The lowest level of user interface ambition is an interface which supports only data input and update functionality, together with some data browser functions. Typically, such a user interface is based on a set of forms and grids where the user either can fill in data or view data. An example of one such program is LCA Inventory Database Datamanager (LCAIDBDM), developed at department of Technical environmental Planning, Chalmers University of technology. Figure 42 shows a picture of how this program appears to the user.

Figure 44 A form-based, simple database interface

An interface of this kind could be compared with a library-database, giving support for data access, but without any support for calculations, data analysis and graphical representation of structures and numerals. This kind of interface is well fit for inserting data into the database.

5.1.2 Flowchart constructors and viewers

Several different ways of constructing and viewing an Activity's internal structure can be thought of. The typical LCI-practitioner draws flowcharts of the system being analysed, while the typical constructor or designer adds functions and materials to each other to construct an product. There might be a need for applications combining these two data representations, as well as there might be a need for completely new ways of representation, not yet known of.

5.1.3 Analysis tools

Analysing a functional unit or a product involves regarding a wide range of different aspects of the system being analysed. Both numerical and qualitative aspects are taken into account. A fully functionate database interface tool for LCIs should help the practitioner in these steps of analysis. Grouping of qualitative parameters, statistical analysis, as well as sensitivity analysis and calaculations of change should be supplied. Also, a designer or constructor should be supplied with a set of analysis tools, which, at least, helps the decision maker to see the statistical width of the numerical values presented.

5.2 Algorithms

5.2.1 General calculations

Each Activity is stored in a form so that its external flows are normalised to each other. Every time a stored Activity is used in a study (that is; the Activity is internal of a host Activity) the amounts of its flows needs to be calculated to correspond to the flows of the environment. Consider the two situations described in figure 45; To the left there are two free, not connected, internally normalised Activities. To the right the same two Activities are used in a study, which means that they are internal of the Activity represented by the larger rectangle surrounding the two. As pointed at in the figure, the connected flows (D and X) of the study must be equal on both sides of the connection, which makes the whole study normalised and corresponding to one single unit flow (Z).

Since one and the same Activity can be used many times in different studies, the result of the recalculations of the normalisations of the internal Activities cannot be stored as such in the database. Instead, the result is stored in the form of the external flows of the host Activity, while the internal Activities are left untouched in the database. To view the internal Activities and their flows as normalised to a flow of the host Activity, the whole system of connected flows must again be recalculated.

Figure 45.

To produce the same result each time a system is viewed, the algorithms for calculating these systems should be rigid and unambiguous. This is not trivial, however, since looped systems (such as recycling loops), allocational decisions and uncertainty affects and is part of the numerical calculations. There are different methods suggested for solving looped systems, as well as there are different principles used for allocational decisions. Data uncertainty is both quantitative as well as qualitative. Methods and algorithms designed to calculate a system of flowconnected Activities must take into regard all these different aspects.

5.2.2 Vertical structures

There are no limits on the depth of internal Activities inside Activties, and so on. This implies that a calculated result on an Activity can be based on an arbitrarily deep vertical structure. There are two problems associated with this: numerical error propagation and propagation of data quality.

Numerical errors, such as statistical width and bias, propagates through a calculational process, yielding a total statistical width and bias of the result. Of course this error cannot be less than the error of the included numericals. Therefore, when calculating an Activity, all its numericals has an at least as wide a statistical width as did its included parameters. Only under very specific circumstances is the width conserved, and in general it is relatively wider. Hereby the total numerical error of an Activity increases with the depth of its internal structure. This should be considered when designing a calculational tool.

Intuitively, data quality propagation should behave quite like numerical error propagation. The problem is to define calculational rules for data quality. It might be possible, but it requires an effort not yet made. Until such calculational rules has been defined, a user of data on an Activity with internal structure must be aware of this problem.

5.2.3 Statistics and sensitivity analysis

To allow for applying statistical methods to data, the data itself must be statistically defined. A quantity must not be given as a single value, but rather as a mean, supplied with information on its width. With this information, a rule of the thumb esimate can be made on the total statistical property of a system. If better analyses are required, all means and widths should be assessed, so that they truly can be described by the same statistical function, the normal distribution, for example. Since today, data seldom are described even with an indefinite statistical width, there might not yet be any options for a rigorous statistical data analysis.

Except handling statistical information on data given in the database, analysis tools could supply users with sensitivity analysis functions and tools for simulating the system’s behaviour under different circumstances.

5.3 Precautions

A computerised database, managed by automated calculating and analysing tools can be a very good help for LCI and LCA practitioners. However, there are risks associated with almost all kinds of automated tools. This section will try to highligt some of the risks associated with the LCI database and tools.

5.3.1 Deep structures and error propagation

It is quite straightforward to construct a computer program which calculates a numerical total amount of resource and energy use, emissions, products and waste on an Activity, regardless of the Activity's structural depth. If such programs are constructed without the ability to handle statistical error propagation, their result might be misleading or useless. The reason is that numerical errors, such as statistical widths, tends to get wider for each aggregational level. This implies that the uncertainty of the numerical results increases as the structural depth increases. This fact suggests a statistical limit to the vertical depth of structures: the structure cannot be so deep that the uncertainty of the numerical result makes it impossible to base decisions on the results.

A program not taking error propagation into account will never warn users when the uncertainty increases, which might mislead users to believe that all numerical results has a the same relevance in decision making.

5.3.2 Simplified work with no qualitative checks

In addition to numerical uncertainties, LCI and LCA handles many different qualitative uncertainties, such as lack of knowledge, uncertain data sources, uncertainty about data type compatibility, uncertainty about compatibility between boundaries of different systems, etceteras.

In not-computerised LCIs and LCAs, considerations regarding qualitative uncertainties are made continuously, at different steps during the assessment and in different ways; While a practitioner searches for suitable data, he or she uses some criterias to sort out bad data quality from good data quality. These criterias are based on knowledge and experience and they are also further developed during the data gathering process. This comes as a result from the acquaintance with different data sources. A not-computerised assessment also is an interactive process, involving not only the practitioner but also other persons; such as persons responsible for delivering data, other practitioners, one or more reference groups etceteras. All these are often involved in one assessment processes. During this process, discussions between the practitioner and these other persons works as to improve the quality of the assessment.

There is no way to guarantee that qualitative aspects are taken into account during any assessment. Of course, this is true also when using computer based databases and computerised assessment tools. There is an additional risk, however, when using computers. A computer based database makes it simple to find data (if it exists in the database), and database applications can make it simple to construct technical systems (flowcharts), to perform numerical analyses and to perform impact calculations from those technical systems.

The simplicity in itself might mislead inexperienced practitioners to believe that it is sufficient to consult the database when searching for data, or that the numerical capabilities of a supplied analysis tool is interchangable with qualitative considerations.

It is not likely that it will be possible to program an analysis tool so that it automatically checks if qualitative considerations has been made during an assessment. Such checks need to be made by the practitioner. Users of the database and its tools should therefore be educated or experienced in the field of LCA.

5.3.3 Updating of data referred to

By making it possible to store the internal structure of Activities constructed from other Activities in the database, data transparency can be fully supported. Not only is there a reference to the data source, but the data source itself resides in the same database, and it can be immediately accessed and viewed. However, the conceptual model, and implicitly also the database structure, does not fully support this data transparency potential.

The database structure allows changes to be made to data refered to. The reason for this, is that there are many different applications where a disallowance of changes is not wanted. To support data transparency, however, it is important that the data source, from which an Activity is constructed, does *not* change. If it does, a reference to it has no meaning. The same is true if a published report references data: the data must not change, or the reference will become useless.

To enable consistent references to data in the database, additional constraints must be added to the database.

6. Future research needs

So far the experience of the use of the database is limited. Following the principle of learning by doing, much of the research needs are yet to be formulated.

However a few items have appeared already at this stage, and is shortly indicated below:

1. A general application tool for SME companies is requested.

2. Development of a quality assurance program. In order to enter data into the database, such a program should be used to check the completeness and reasonableness of the results.

3. Further development of the part of the database describing the impact on environment. A type of "inventory" in the environment is necessary to assess the magnitude and significance of the impact. This inventory is at present documented in reports and literature. Such a documentation decreases the traceability of the rational for the values used in impact assessment.

4. The table “classification” containing information of classification and characterisation ought to be merged with the tables around impact assessments describing unit effects. etc. This may be done by letting the table **Event** store information on characterisation and the table **UnitEffect** list relevant classes. However today the classification and characterisation is used as a substance specific property, and does not need to be associated with any particular area or environment where the emission occur.

7. Acknowledgments

The support from the Nordic Industrial Research Fund, (NIF), the Norwegian Research Fund (NFR), NUTEK in Sweden and the Swedish Waste Research Council is greately acknowledged.

Representatives from the following companies have contributed to the design of the database:

ABB, Berol Nobel, Elkem, Mölnlycke, Norsk Hydro, Stora, Tetrapak and Volvo.

A special thanks to Thomas Rydberg for the suggestion of the database name, and to Sven-Olof Ryding, Ole Jörgen Hanssen, Siv Jorunn Förde and Göran Svensson for many fruitful inputs.

8.Literature

(1989) Tolba, M. Industry and Environment, UNEP.

(1992) Steen, B., and Ryding, S.O., The EPS enviro-accounting method, IVL-report nr B 1080, Swedish Environmental. Research Institute, 1992.

(1993) SETAC, Guidelines for Life-Cycle Assessment, A Code of practice, Proceedings from the SETAC workshop held in Sesimbra, Portugal, march 31-April 3, 1993.

(1994) Carlson, R. , Design and Implementation of a Database for use in the Life Cycle Inventory Stage of Environmental Life Cycle Assessment. Diploma work in Computing Science, Department of Computing Science, Chalmers University of Technology and Gothenburg University, 1994.

Appendix 2

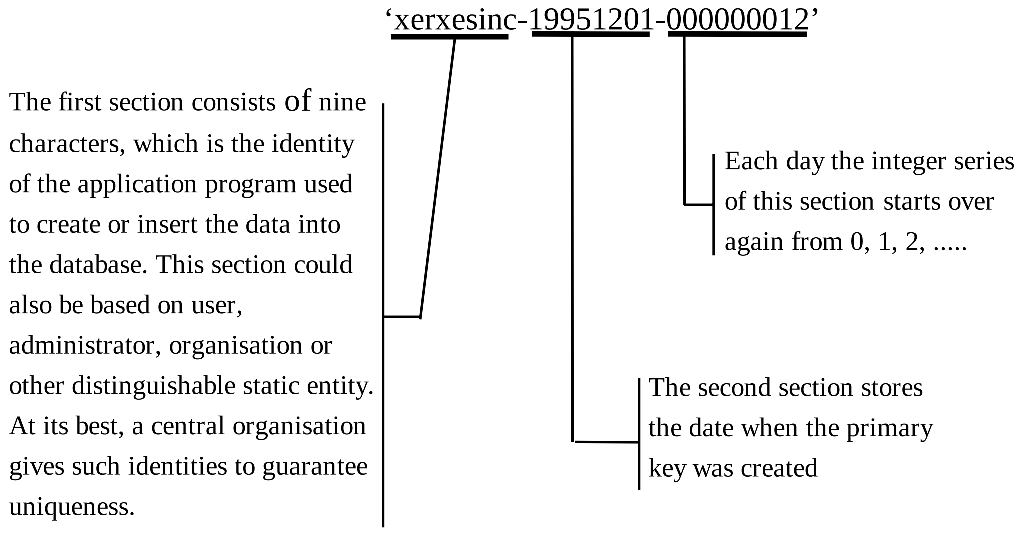
Globally unique artificial keys.

Many entities are not unique and cannot be distinguished from each other, unless they are supplied with an artificial unique identity. This is very common, and is known from very different fields; peoples names are not unique, and therefore many organisations, such as nations, has introduced different kinds of organisational numbers. Cars are supplied with license plates, products are supplied with product numbers, telephones with telephone numbers, etceteras. All this is done to make possible a distinguishment between otherwise indistinguishable entities.

For introducing distinguishment, artificial primary keys has been introduced in the database. Generally, if there is only one database, and if the the primary keys have no meaning outside of the database, the anatomy of the primary keys are absolutely arbitrary. An automatic generating rule is designed for the keys, and few human ever looks at them (only programmers and database administrators). However, in this case, where it is likely that there will be several different database implementations, and where these different database implementations will come to exchange data with each other, a second consideration must be taken into account: It no longer is enough only to generate unique primary keys within the database, but the generation rule must also be designed so that it guarantees uniqueness within the whole set of different database implementations. Such a generating rule, desigend to generate unique identities in disjunct sets, are referred to as a *global naming convention*.

The global naming convention suggested to be used with this database is based on 30 characters, divided into three different sections of nine characters each, the three are separated by two occurnces of the character ‘-’. A typical identity of this form would look as:

‘xerxesinc-19951201-000000012’



The first section consists of nine characters, which is the identity of the application program used to create or insert the data into the database. This section could also be based on user, administrator, organisation or other distinguishable static entity. At its best, a central organisation gives such identities to guarantee uniqueness.

Each day the integer series of this section starts over again from 0, 1, 2, ...

The second section stores the date when the primary key was created

By this arrangement there can be 26^9 different database users/implementations, each day creating 26^9new entities of each of the types with artificial primary keys.

Of course there are a multitude of other global naming conventions doing equally well as this simple one. The important thing is to do one that is simple enough and that is accepted. This generating rule might be one such convention.