



PROJECT N. 037033

EXIOPOL

A NEW ENVIRONMENTAL ACCOUNTING
FRAMEWORK USING EXTERNALITY
DATA AND INPUT-OUTPUT TOOLS
FOR POLICY ANALYSIS

THE EXIOPOL DATABASE MANAGEMENT SYSTEM — MAIN DESIGN

Report of the EXIOPOL project

Title	The EXIOPOL database management system – main design
Purpose	A report that provides an extensive list of design criteria of the database, 4-5 strategic design options, and proposes a design option
Filename	EXIOPOL_DIII 4 b-1b_FINAL.doc
Authors	Arjan de Koning, Reinout Heijungs, Gjalt Huppes
Document history	not relevant
Current version	3
Changes to previous version.	not relevant
Date	15 October 2008
Status	Draft
Target readership	Cluster III partners
General readership	not relevant
Dissemination level	Cluster III partners

Arjan de Koning, Reinout Heijungs, Gjalt Huppes
Institute of Environmental Sciences (CML) - Universiteit Leiden

October 2008

Prepared under contract from the European Commission

Contract no 037033-2

Integrated Project in

PRIORITY 6.3 Global Change and Ecosystems
in the 6th EU framework programme

Deliverable title:	The EXIOPOL database management system – main design
Deliverable no. :	D III.4.b-1b
Due date of deliverable:	Month 18
Period covered:	from 1 st March 2007 to 1 st September 2008
Actual submission date:	10.10.2008
Start of the project:	01.03.2007
Duration:	4 years
Start date of project:	01.03.2007
Project coordinator:	Anil Markandya
Project coordinator organisation:	FEEM

Executive Summary

This report offers a description of the overall design of the EXIOPOL database management system for environmentally extended input-output analysis, along with the considerations that have lead to this design. It discusses the main elements of the database, some general principles of databases and their design, and surveys the main choices and issues in the EXIOPOL context. The design allows for the usual application in economic models while the environmental extensions not only cover relations with the environment system, as resource extractions, land use and emissions, but also material flows within the economic system. The high-level architecture is shown in the figure below (where SUT stands for supply-use table, IOT for input-output table, EE for environmental extension, and the prefix MR for multi-region).

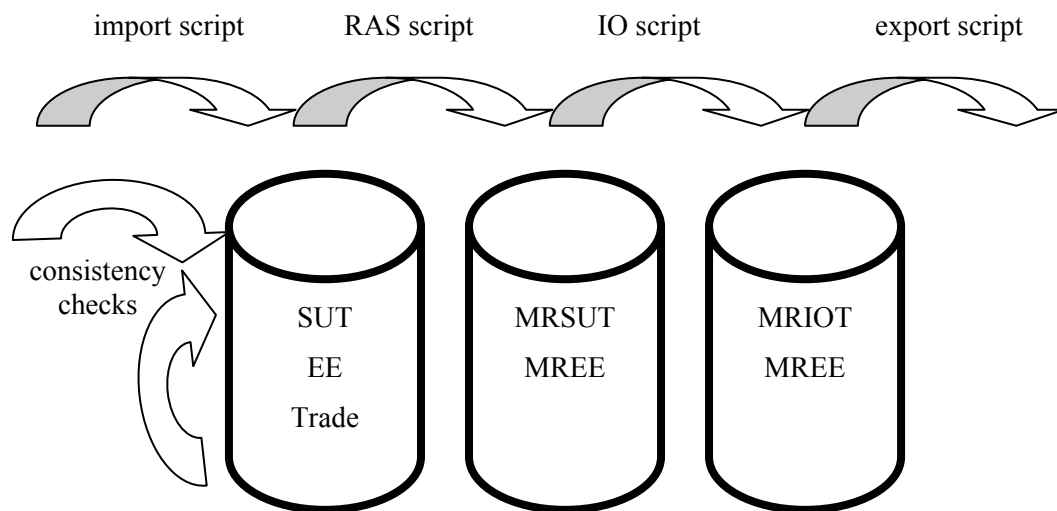


Table of contents

Executive Summary	iii
1 Introduction	1
1.1 Overview of WS III.4	1
1.2 Problem definition	1
1.3 Design methods and structure of the report	2
2 A database for environmentally extended input output analysis (EEIOA).....	4
2.1 Data ingredients	4
2.2 Data operations	4
2.3 Data purpose	5
3 General principles of database theory.....	8
3.1 What is a database?.....	8
3.2 The database management system	8
3.3 Data processing and multiple databases.....	9
3.4 Overview of the EXIOPOL database structure	10
4 Design requirements and considerations	13
4.1 Choice of hardware platform and operating system	13
4.2 Choice of programming language.....	14
4.3 Choice of RDBMS design tool	14
4.4 Choice of RDBMS software	14
4.5 Issues of performance and resource management.....	15
4.6 Issues of security	17
4.7 Issues of dissemination	17
4.8 Scalability	19
4.9 The user interface.....	19
4.10 Issues of internationalisation and localisation.....	19
4.11 Missing data	19
4.12 Error processing	20
4.13 Database testing	20
4.14 Requirements changes and design changes	22
4.15 Revision control	22
List of references	23
Annex I: Contributors to the report	23

1 Introduction

1.1 Overview of WS III.4

This report is one of a series of reports that describe the EXIOPOL database management system that is developed in Workstream III.4 of the EXIOPOL project. The full series consists of the following reports and other deliverables.

Number	Title or description
DIII.4.a-2	Trade Linking Method Method for trade linking described
DIII.4.a-3	Four Country Test Algorithm available and applied on a test of 4 countries
DIII.4.b-1a	Data Input Protocol The protocols for data input are described after development in discussion with the leaders of WP.III.2.a/b and WP.III.3.a/b. The data input protocol shall link easily with the form in which data are made available after the transformations in Block 1 (i.e usually matrices in Excel, or Access)
DIII.4.b-1b	EXIOBASE Main Design Provision of a report that provides an extensive list of design criteria of the database, 4-5 strategic design options, and proposes a design option. This report shall reflect the holistic perspective on the database development work as discussed under 'Objectives'
DIII.4.b-1	Nomenclature and Transformations Description of the basic nomenclature used for all output data, with transformation tables to classifications as used in all data sources
DIII.4.b-2	Computational Structure Description of all steps and intermediate results
DIII.4.b-3	EXIOBASE Software Full Database Manager
DIII.4.b-4	EXIOBASE Data Database filled with all data
PDIII.4.c-1	Set-Up of World Trade Model Interface Preliminary report on interface to World Trade Model and its extensions
PDIII.4.c-2	Outline on interface to economy-wide and bottom-up sectoral models
DIII.4.c-1	World Trade Model Interface Interface and report on World Trade Model and its extensions
DIII.4.c-2	Interface and report on economy-wide and bottom-up sectoral models

The present report is DIII.4.b-1b. It offers a description of the overall design of the database management system, along with the considerations that have lead to this design. The result of the implementation itself is referred to as EXIOBASE, and is incorporated in DIII.4.b-3.

1.2 Problem definition

Work package III.4.b of the EXIOPOL project is devoted to the transformation of the input data into environmentally extended input output tables, to be used independently or as part of further modelling endeavours. Given the number of input data to be transformed, the requirements as to easy updatability and the computational intensive nature of the transformations this will have to be done with software and cannot be carried out by hand.

Such software to store the input data and to carry out the transformations was not available. Therefore this software has been developed to be able to meet the EXIOPOL project objectives.

The overall objectives of the software have been specified as:

- “to develop an integrated computational structure which transforms input data into outputs in the form of environmentally extended input output tables and exemplary PIOTS, in a number of forms.”
- “make this computational structure externally available for modelling purposes, and, by its open and transparent nature, to allow for its transparent improvement and updating.”

This document describes the overall design of the EXIOPOL database management system that is used to meet the objectives specified above. The other documents in this series describe in more detail the subparts of the system.

For this rather small software system the design specifications are quite detailed. The following considerations were taken into account when it was decided to make a rather detailed design before construction/coding of the software began.

- Software construction is not the prime speciality of the team that builds the system; rather they have an expertise in economic-environmental systems and their modelling.
- The software is critical for the accomplishment of the EXIOPOL project. If it the software doesn't work or cannot fulfil the demands of the EXIOPOL project, then the EXIOPOL project is seriously impaired.
- The software has to be maintained for a longer period and likely has to be maintained by people who did not develop the software and who are not a member of the EXIOPOL consortium.
- The software contains all bits and pieces of information (algorithms interfaces to other software) that are specified by other EXIOPOL cluster III work packages.

Two design approaches are surely to fail: no design at all and directly start programming or designing every last detail. Therefore the critical task is to make a database design that is well-balanced in this respect, giving enough guidance but not too much complexity to the software design process. This report discusses the considerations of the design process and its results in main lines. Details concerning nomenclature are to be found in DIII.4.b-1, and concerning the computational structure in DIII.4.b-2. The database manager itself is in DIII.4.b-3.

1.3 Design methods and structure of the report

No single formal design method exists that guarantees the making of the “best” design. Design is a dirty, heuristic and iterative process. For instance, some software design issues may only be solved after implementation of the software has started and failed, the so-called wicked problems (McConnel, 2004).

The software system design was started in a top down fashion and this report describes the final result of the heuristic, dirty and iterative design process. To be able to get a feeling for the quality of the design we used checklists to check if we covered all the aspects of the software design.

The design of the software system was started with a further specification of the software requirements. Partly these requirements had already been specified in the Description of Work (DoW) and were further worked out during the scoping phase of the EXIOPOL project.

Based on the requirements the high level architecture of the system was drawn-up. High level system architecture includes the specification of the components/modules of the system, rules for internationalisation/localisation, error processing, graphical user interface design, etc. All requirements find its way in the system architecture. At that point also some key decisions with respect to the programming method needs to be discussed i.e. which programming tools, for which platforms, etc.

The overall system design results in the identification of the entities that should be stored in the database management system. We already define these entities in this overall design report as it determines – at high level – the data that the subparts/components can work on. Classifications used in the specification of the entities are discussed in the DIII.4.b-1 report. Detailing of the entities and relationships between entities is done in the respective module reports.

The report starts, however, by a discussion of the elements of the EXIOPOL project that are supposed to find a place in the database and of the database concept in general.

2 A database for environmentally extended input output analysis (EEIOA)

This chapter describes the most central features that the database should cover or allow to be covered by additional add-on tools.

2.1 Data ingredients

According to the Cluster III scope description (Tukker & Heijungs, 2008), there are a number of elements that have to be accommodated by the database:

- country level supply and use tables (commodity-by-industry);
- trade data, sometimes in vector form, but also sometimes in matrix form;
- final demand, split in a number of categories;
- factor inputs, split in a number of categories;
- valuation matrices, accounting for trade and transport margins, subsidies and taxes, etc.;
- data on consumers and waste that is not part of the usual supply and use system;
- environmental extensions (split into many emissions and resource extractions);
- physical data on products used or supplied (e.g. on electricity in MJ, or on cars in kg copper content).

All this information is available for a number of countries, and at least for one year, but in future for a number of years. In SUTs and IO tables, core information is available in monetary units. For this, a single monetary unit (the euro) has been chosen. The currency transformation for non-Euro countries is done when bringing the country data into the DBMS format. Different forms of price systems (purchaser price, basic price, etc.) can be used in the system.

Other information is available in physical units as well, referring to the same flows as quantified in monetary units, like MJ of electricity, number of cars, copper content in cars, and hours of labour. Some flows are in physical units only, as with resource extractions and emissions, like kg of copper ore, kg copper in coal mined, kg of CO₂, Becquerel of radiation and square meters of land used.

2.2 Data operations

The system contains data, but it also allows several operations with or on these data:

- data import, by means of an import template;
- data consistency checks, in several forms, e.g., checking for industries with environmental extensions but without final demand;
- data views, e.g., displaying a country level SUT;
- trade linking operations, transforming the country level SUTs and trade data into a multi-region SUT;

- conversion of the SUT into an IOT (commodity-by-commodity or industry-by-industry) by means of different industry and technology assumptions;
- exporting data and/or tables to serve as an input to subsequent models, such as the World Trade Model;
- establishing a connection between the environmental extensions and impact assessment and valuation systems, such as those in Cluster II of EXIOPOL or LCA impact assessment.

Notice that this not necessarily mean that all these functions are part of the database system itself. It even does not mean that all these ingredients are necessarily produced within the EXIOPOL project. For instance, the link with Cluster II type impact assessment and valuation was not originally addressed as a task but has been included. Some further elements will surely become an indispensable part of the EXIOPOL project. The choice what exactly is and what is not incorporated ultimately is not yet final. The database structure should be sufficiently transparent and open that new and missing elements can be added in a later stage, also by non-EXIOPOL persons and beyond the project.

2.3 Data purpose

Cluster III has as main goal to develop an operational and detailed EU-25 input-output table with environmental extensions (further abbreviated as: EEIO). This is basically a monetary input-output table to which information about emissions and resource extraction is added per sector, together with data on the composition of product flows. The database that will be developed will include external costs per sector as calculated in Cluster II as an extension as well. The EEIO table for the EU25 will be embedded in a global context. This is essential to be able to take pollution and externalities embedded in imports to the EU25 into account (Peters et al., 2006; Nijdam and Wilting, 2003), but also to be able to analyse the effects of sustainability measures taken in Europe on the economic competitiveness of the EU25. The EEIO table will also be organised in such a way that it can:

- support analyses at micro-level, such as cost-benefit analysis (CBA) and LCA, via so-called hybrid LCA or EEIO-LCA; thus both improving this micro analysis and linking it to the meso and macro level;
- support scenario-analysis for analysis of impacts of sustainability options at meso- and macro level. Such scenario analyses can be done in a variety of ways. One option is to 'force' exogenously a technology-, emission- or demand scenario on the EEIO table. Another option is to endogenize economic developments. For this purpose, the EEIO table developed will be linked to a number of existing CGE models in use with the EU and to some macroeconomic models. Furthermore, an alternative model (the World Trade Model; Duchin, 2004) will be expanded in this project to create a third approach to scenario analysis.

The basic elements of functional demands are therefore:

- monetary input-output tables for EU-25;
- environmental extensions including emissions and resource extractions;
- connection to external costs (Cluster II);
- global context, covering main trade partners of EU-25;
- connections to existing models and tools, such as CBA, LCA, hybrid LCA, EEIO-LCA, CGE models, World Trade Model.

In addition, the Description of Work mentions several other connections, such as

- environmental themes, such as global warming, ozone depletion, etc.;
- total material requirement (TMR), and other MFA indicators (copper example);
- ecological footprint.

The EEIO-tables will provide IO data with several satellites that enable the connection to such existing models and tools and to the external costs part of Cluster II.

One special consideration in this set-up is on how to deal with material flows as in MFA, which covers both flows from and to the environment and also covers flows between activities within the economy. For both types of flows, a representation is possible in the system, allowing for a full MFA/SFA. The terms in which such system flows are defined is open, as long as it is in terms of attributes of the flows involved. Therefore, they need not coincide for examples with the terms used for resource extractions. Specifications may range from total mass to specific elements. This subject of detailing attributes is left open, with at least one exception. The copper flows through the system will be specified, requiring an attribute 'copper' for all flows concerned. These copper flows not only refer to resource extractions and product flows but also to emissions, where they may be part of the compounds involved. Based on the experience in the case application, adaptations to EXIOBASE may be considered.

Linked to considerations as to what it should do, there are the conditions that are related to the content. In particular, there is the issue of the exact type of data made available for the intended functionality. Apart from the details as to sectoral resolution and physical flows covered, a principal point is the form of the accounting format. Main options are:

- rectangular supply and use table
- square supply and use table
- square input-output table of the industry-by-industry type;
- square input-output table of the commodity-by-commodity type;

Supply and use tables form the basis for making input-output tables. Moreover, there are several ways of deriving IO tables from S&U tables (Kop Jansen & Ten Raa, 1990). When using input-output tables with environmental extensions, the rectangular SUTs are most apt in covering material flows, as they allow for a more detailed specification of products than rectangular ones. The multi-region tables then also can incorporate at least some of the detail which is available in trade statistics. However, in most basic data sets available at country level, each product has been specified as "the" product of one industry, possibly produced as co-product by other industries as well (electricity by sugar refineries). For this practical reason we now use square supply and use tables. The data model is designed however so as to be easily adapted to rectangular SUTs.

As to the IO tables, the two main variants are industry-by-industry tables and commodity-by-commodity tables, both square. Conceptually, it is not products which emit but activities as represented by industries. This is most clear when there are several industries producing the same products. The link to emissions and resource extractions therefore is most clear in the industry-by-industry type. We provide both versions, however, leaving matters of interpretation to users.

Thus S&U tables and IO tables organized in several ways will be delivered as a final result.

The basic architecture of the EEIO accommodates data of the supply and use type, as well as data of the input-output type under several construction variants.

In simultaneously designing and implementing a database, there is a tension between desirability, how the system would ideally be) and feasibility, how the system can be filled with the presently available data and presently available transformation routines. In this respect, it is important to quote the Description of Work (p.71-77): “Even for different sectors within a country historical reasons have led to incomparable methods for data gathering, classification and transformation. A clear view to future requirements can align activities within statistical bureaus, with a view on contributing to European tables.” The emphasis of our work should thus clearly be future-oriented, making a sound structure that can be used by the member states’ statistical bureaus within a few years time. The issue of the data format for industries, that is the changeover from NACE 1.2 to NACE 2, around the year 2011, is discussed elsewhere.

To be able to offer not only a sound structure for the future, but also an operational database that can be used during and after the EXIOPOL project, temporary solutions are needed. These relate to data formats, transformations, aggregation and estimation steps.

The EXIOBASE puts emphasis on being adaptable to a future harmonized data structure, but now it incorporates elements based on presently available imperfect and unharmonized data.

3 General principles of database theory

3.1 What is a database?

In ordinary language, the term database is often loosely used for a collection of data. However, computer scientists use a much more strict definition. Wikipedia (accessed 3 July 2008) states that “A database is a structured collection of records or data. A computer database relies upon software to organize the storage of data. The software models the database structure in what are known as database models. The model in most common use today is the relational model.” As the EXIOPOL project is concerned with large amounts of data, that are moreover to be managed and used by different persons, a well-structured way of organizing the data is a critical aspect for achieving a successful result.

The most used type of database structure is the relational database. It consists of a collection of tables, each of which contains information on a number of entities. The connections between the different sets of entities is established by means of the relations. Both entities and relationships can have attributes. As such, so-called entity-relationship models (ERM) form the core of the structure of the database. A simplified example of an entity-relationship diagram (ERD), is given in **Figure 3.1**.

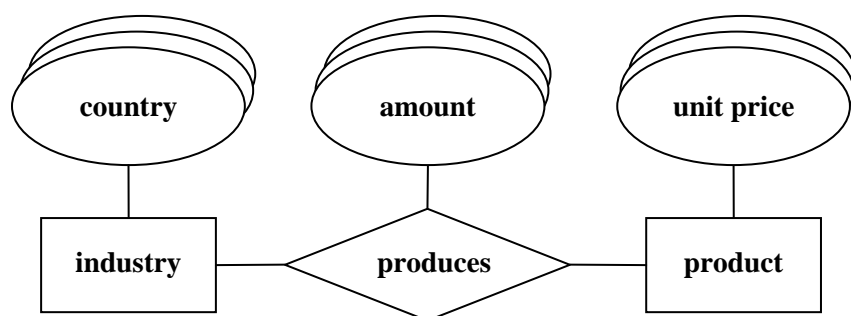


Figure 3.1: Example of a simple entity-relationship diagram with two *entities*, (industry and product), having a *relationship* (produces), with exemplary *attributes* to the entities (country, unit price) and to the relationship (amount).

It is a critical task to design the ERM before the actual process of filling the database starts. Thus, we see a clear distinction between the structure of the database and the content of the database. Especially important to mention in the context of EXIOPOL is the role of industry and product classifications. In the scope document, reasons have been mentioned for basing the EXIOPOL database on NACE 1.1 classification, not on NACE 2. The database structure, however, can be constructed in such a way that it is independent of the details of specific classifications. A revision of the classification during or after the project, or a switch to NACE 2 will give rise to extra data collection and transformation efforts, but there is no need for changes in the database structure.

3.2 The database management system

An ERM designed according to the principles of database theory is only a concept; it does not work yet. It needs to be implemented in software. A database management system

(DBMS) is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMSs can be programmed by a professional programmer in any programming language, but it is efficient and convenient to rely on a well-tested DBMS environment. Within the context of relational databases, the choice of software system for developing the relational database management system (RDBMS) is important. Examples of such software systems are Oracle, Firebird, MySQL and Microsoft Access. Such programs help the database designer and/or user with several tasks:

- to construct the ERM;
- to enter data in the different tables;
- to analyze the data
- to create views in which the data are presented in a way that suits the user;
- to enable access to the data (reading and/or writing) by external programs.

The different RDBMSs are quite different with respect to user-friendliness, capacity and license costs. But they are all similar in one important aspect: they all operate under the structured query language (SQL). In principle, a relational database that has been set-up under, say Microsoft Access, can easily be transferred to, say, Firebird. This aspect can be important when there are reasons to migrate to another software system for RDBMSs during or after EXIOPOL, for instance when the original implementation becomes too limited, or when it is too expensive for a broader user group.

3.3 Data processing and multiple databases

A RDBMS is principally a tool for organizing and retrieving data. SQL statements, of which we see an example in **Figure 3.2**, allow the user to manipulate the data. Although the SQL contains mathematical expressions, a RDBMS is not particularly powerful in doing lots of computations or in doing complicated computations. In the context of EXIOPOL, such operations are needed at many places, e.g., in the trade linking process, in the conversion of SUT to IOT, and in the calculation of the Leontief inverse. The use of SQL can become problematic for such tasks: algorithms are difficult to implement, they work slowly or not at all at PCs, etc. Therefore, it is critical to think of solutions to this problem in connection with the database design.

```
SELECT      *
FROM products
WHERE       price>50
ORDER BY    cpa;
```

Figure 3.2: Example of a simple set of SQL statements.

As mentioned before, many programs can access a relational database. So, it should be possible to develop other programs that first read the data from a database, then do the complicated mathematics, and finally write the results back to the database. Such procedures may require as substantial amount of careful manual labour, as is normally the case when a researcher combines, say Microsoft Excel and Matlab. However, by an appropriate design of the exchange formats and scripts, such steps can also be fully automated. For instance, an

interface can be built that contains a “Calculate” button, that first exports data, then evokes a secondary program, and finally imports the processed data. In fact, a user might be unaware of the fact that an external program is being used or that different databases are employed.

This principle can even be extended so as to have multiple databases, that are connected by SQL scripts or external computational routines. The first database might contain, say the country level supply and use tables and the trade data, the second database might contain, say, the trade linked multi-region supply-use table. Again, a user may be unaware of the fact that these are two databases, not one. See **Figure 3.3**.

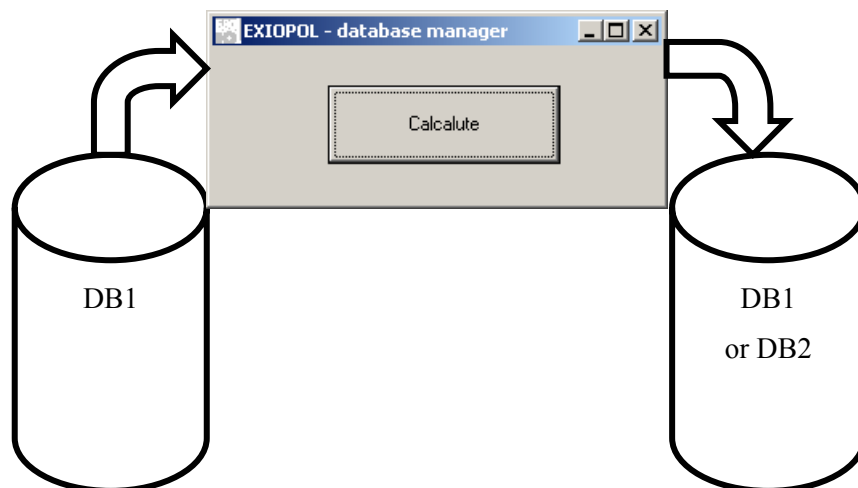


Figure 3.3: Illustration of how an external program may import data from a database, process it, and export it to the same or another database.

3.4 Overview of the EXIOPOL database structure

The EXIOPOL project is a project with a highly modular structure. In the scope document (Tukker & Heijungs, 2008, p.55) the following levels were distinguished:

- the raw datasets (SUTs, NAMEAs, etc.);
- the primary country level SUT and extensions;
- the trade-linked multi-region SUT (MRSUT) and multi-region IOT (MRIOT);
- the derived country level SUTs;
- the derived country level IOTs;
- the specific applications (multipliers, footprint, etc.).

The first level has been decided not to be part of the database, as it is fully inconsistent between many countries. Likewise, the last level does not provide results that go to the database, but go to other applications (such as the World Trade Model), as the result of export scripts.

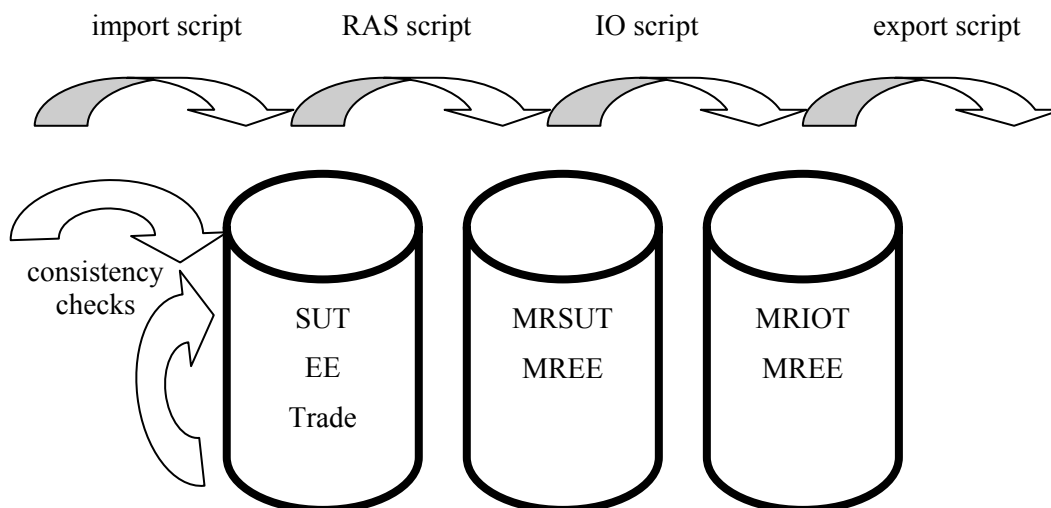


Figure 3.4: Databases structure of the EXIOPOL database manager. Legend: cylinders represent databases; grey arrows represent data flows and transformations; white arrows represent consistency checking procedures.

The high level architecture for the EXIOPOL database is shown in **Figure 3.4**. In this figure, the following elements show up:

- The leftmost cylinder is the database in which the supply and use tables for all individual countries are stored in the harmonized EXIOPOL classification (De Koning et al., 2008a) and structure (De Koning et al., 2008b). It also contains additional data items, such as the environmental extensions, the trade data, the factor inputs and the final demand data.
- In other workstreams (WS III.2 and WS III.3), raw data are collected for individual countries, transformed into the harmonized EXIOPOL classification, and stored in Excel tables with a predefined structure. These tables are imported in the leftmost database with a piece of software. No matrix structures are imported, but only vertically oriented tables. This has two advantages: the number of columns is often restricted across different software and platform specifications, and a vertically-oriented import allows for a piecewise entry and update of data, for instance agricultural data on day 1 and data on chemical industries on day 2.
- During this process of data import, several consistency checks are carried out. For instance, it is checked if all industry and product names are in conformity with the list of D III.4.b-1 (De Koning & Heijungs, 2008a). Numerical data can also be subject to checks. For instance, certain data types should be non-negative, or between 0 and 1.
- The data inside the leftmost database is entered on a record-by-record basis through the import routines. This does not allow for consistency checks for the data tables as a whole. Hence additional consistency checks can be carried out the level of the first database. For instance, it can be checked if there are industries with supply, but without use data, or with emission data but without supply-use data.
- The country level SUTs in the leftmost database contain data about import and export, but they are not yet connected by trade flows. The RAS script performs this trade linking, according to the specifications of D III.4.a-2 (Bouwmeester & Oosterhaven, 2008). This may involve various options for trade linking, e.g. bottom-

up vs. top-down, an interregional vs. a multi-regional model, etc. At this moment, a partially automated RAS script is available (in Matlab/Gauss): some manual readjustments may be needed. During the next few months, however, it will become a fully automated procedure, which moreover will be able to run without the Matlab or Gauss environment.

- The RAS script transforms the first database of separate SUTs into one large trade-linked multi-regional SUT. This will be stored in a second database, which is represented by the middle cylinder. It could be argued that there is no need for this second database, because it could be calculated “on the fly” from the first database with the fully automated RAS script. However, it is expected that the time needed for this computation is too long for “on the fly”. Moreover, we expect that the stand-alone value of the second database is high enough to justify its storage in a separate database. This second database contains besides the trade-linked SUT other information types like environmental extensions, final demand, etc. Of course, the data on the individual countries as specified in the first database bin are here as well, but in an adapted form.
- The second database contains a multi-region SUT (MRSUT). An IO script will transform this into a multi-region IOT (MRIOT). In principle, there are various options for constructing an IOT from a SUT, and this applies equally for a multi-region table. For instance, choices can be made with respect to commodity-by-commodity vs. industry-by-industry, to the treatment of secondary products, etc.
- The result of the IO script will be a MRIOT, several of them, according to the number of options specified for making the transformation. These MRIOTs are stored in a third database. Again, this will be a separate database, for reasons of stand-alone utility and for memory considerations as well.
- Finally, the MRIOT should be fed into the “existing models” (WP III.4.c). Depending on the exact choices to be made, different export formats will be developed (e.g., tab-delimited, Excel, XML).

Figure 3.5 shows the prototype of the database manager, a piece of software that connects the different databases with the different procedures and computational steps.

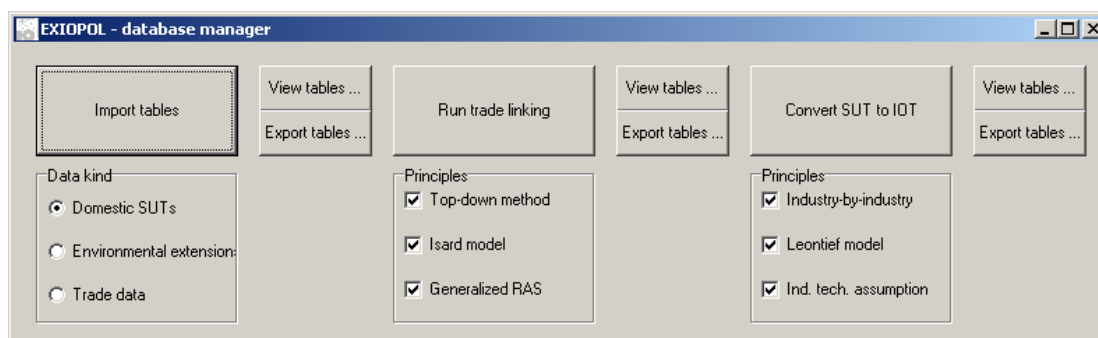


Figure 3.5: Prototype of the database manager. There are import and conversion steps (the large buttons) which process data to and from databases. In addition, there are buttons (the smaller ones) to view and export the data tables in a variety of formats. Notice that this interface is an illustration only. Appearance and available options will be different in the actual implementation.

4 Design requirements and considerations

Requirements describe in detail what a software system is supposed to do, and are the first step towards a solution. There are several reasons why we explicitly describe and discuss the requirements in detail:

- We want to ensure that the intended applications drive the functionality of the system.
- We define the requirements explicitly to avoid arguments about the functionality of the system at a later stage when it becomes very complex and costly to make changes to the software.
- We want to minimise changes to the system after development (coding) begins.

Of course we must accept that during the project – when the level of understanding of the problems we try to solve increases – requirements may change. We must therefore also have a procedure for accepting changes to the requirements (Section 4.13).

Parts of the desirable features of the system were already formulated in the DoW of the project proposal (Tukker et al., 2006). System requirements were further discussed during the scoping phase of the EXIOPOL project and reported in the scoping documents (Tukker & Heijungs, 2008). However many important aspects of the software system have not yet been discussed, as such discussions first have to have firm ground in practical experience. For these aspects the design team has formulated its own requirements based on experience and expectations.

This chapter discusses a large number of aspects that have to be considered when choosing a certain design. Some of these issues have already been solved or have already led to a certain design. For some other issues, no such decisions have been made yet, or no such decision will even be made within the EXIOPOL project. The system design is as open as possible to accommodate additions. However, basic changes will remain complex and costly. Additions in terms of added attributes do not require fundamental adaptations in RDBMS and software. Adding entities and relations will usually have fundamental consequences in almost all scripts and software.

4.1 Choice of hardware platform and operating system

A hardware platform is a hardware architecture that allows software to run. Two well-known examples are the personal computer and the mainframe.

The operating system is the core software that allows other applications to run on the platform, and to communicate with the devices, such as the monitor and the keyboard. Well-known operating systems are Microsoft Windows and Linux.

Choice of hardware platform and operating system are often connected, i.e. most PCs run on Windows while most mainframes run on Unix, and Linux can run on both.

An alternative that might be considered is a web-based server-client system, where the clients are Windows-based PCs, and the server is a more powerful mainframe.

For the choice within EXIOPOL, an important trade-off is to be made between the general applicability on the Windows-based PC and the larger capacity and better performance of a mainframe. For now, a choice for a Windows-based PC has been made. However, the design of software and database will be made in such a way that a transition to a mainframe will be

relatively easy. This means that the use of dedicated Windows-code is restricted to the man-computer interface, and that the code for the computation and transformation steps will be as much as possible universal.

4.2 Choice of programming language

Programmers are more productive when they use a familiar programming language and use a high level language (McConnell, 2004). Two of the members of the CML's project team have proven experience using Borland Delphi for the construction of stand-alone programs on multiple platforms. Delphi is a high level language and it seems therefore logical to use this programming language for reasons of productivity (Cantù, 2005).

Delphi and the Delphi RAD (Rapid Application Development) have all the capabilities and tools expected of a modern high level language. Components are available for the integration of the Delphi programs with multiple database products. XML components and web based services are also available in Delphi.

Borland Delphi runs on Windows 2000, XP and Vista (32 bits). The executables generated with Borland Delphi run on every 32 bits Windows platform.

4.3 Choice of RDBMS design tool

A separate RDBMS design tool has been chosen to make a detailed design of the database tables and relationships procedures.

DB Designer 4 is a database design system, distributed under the GPL2 license. It is written both for the MS-Windows and the Linux platform. The program is specially tailored for MySQL but can be used for every database that supports SQL. It can export the database model as a SQL script file which in turn can be read by any SQL capable database product to create the actual database.

The choice for the use of this tool was inspired by the following aspects:

- The design can be implemented in any database management system, which is very convenient if at any point of time another RDBMS is chosen.
- The program works both on Linux and MS-Windows and is open source.
- It is freely available. Many RDBMS are extremely expensive and often work on a particular DBMS.
- The program is relatively simple which is convenient for the design of EXIOBASE.

The design of the RDBMS could have been carried out directly in MS-Access. However the design would then be very much tied to MS-Access which would hamper a possible migration to another RDBMS product.

4.4 Choice of RDBMS software

Only relational database management systems have been considered for the storage (not dissemination) of data in the EXIOPOL project. Relational database management systems (or object relational DBMS) are the only tools that can store the amount of data in the EXIOPOL project in a flexible, robust and secure manner. See Chapter 3 for the basic concepts of databases.

However the choice for a particular RDBMS product is a difficult one. There are some conflicting requirements that touch upon this aspect:

- The requirements call for an open and transparent system.
- The RDBMS must be maintained for years to come after the EXIOPOL project ends. Therefore the RDBMS product must be available outside the EXIOPOL project without high associated costs.
- It must be easily updateable and maintainable.
- No resources are available to purchase a costly RDBMS product.
- The system must allow for easy storage and computation not only now, but also when data for multiple years are entered.

In the present set-up, Microsoft Access has been selected as a RDBMS, as it is widely available, relatively user-friendly, many people have experience using it. At the same time it allows for the construction of a database architecture that can be migrated to another RDBMS (such as MySQL) when reasons show up to do so.

4.5 Issues of performance and resource management

Database connections, threads, memory, disk size etc. are resources that might be constrained and need to be managed. Resource management may be an issue for the transformation modules. When inverting the technology coefficients matrix to create the Leontief inverse the complete matrix has to be kept in memory.

The EXIOPOL database will deal with quite some data. Some numbers, from DIII.4.b-1:

- there are 43 countries;
- there are 131 industries;
- there are 131 products.

This already creates a 43 use tables with 131 times 131 cells, and 43 supply tables of the same size. Of course, not all cells of these tables will be filled, and especially the supply table may be quite sparse. Nonetheless, we are speaking of a maximum of almost 1.5 million data items, for which floating point values may be needed, each one occupying 4, 8 or 10 bytes, depending on the accuracy of the representation. For data storage, single precision will likely suffice; for computation (see below) probably not.

This is the numerical data for only a part of the system. On top of that, we have numerical data for environmental extensions, trade data, factor inputs, final demand, valuation matrices, etc.

Besides numerical data, we have tables with strings (alphanumeric data), containing industry names and codes, product names and codes, emissions names and codes.

Altogether the amount of data that is to be stored in each of the three databases is quite large. This may create different types of problems. For instance, data may not fit in the MDBS (e.g., the maximum file size for Microsoft Access is 2 GB). And even when it fits, the entry and retrieval of data may become very slow. Various solutions are available to prevent these problems:

- As shown in **Figure 3.4**, it is very logical to split the database and to redistribute the data among three databases.

- As discussed in Section 3.2, a careful design and use of native SQL can ensure an easy migration to a more powerful system (such as MySQL, which has a maximum size for an individual table of 8 million TB).

But even then, we have to acknowledge that matrix operations must be made as well, for RAS, for the conversion from SUT to IOT, for certain applications of IOT that involve Leontief multipliers, etc. The size of this matrix in memory is quite large as shows this example calculation. Suppose we have a matrix with size 5633×5633 (131 industries with 131 products in 43 countries). The technical coefficients might be of type single, double or extended. A single takes 4 bytes, a double 8 bytes and type extended takes 10 bytes. This makes the size of the matrix in memory of a size 13, 26 and 32 MB for the single, double and extended type respectively. For matrix computations, especially for large matrices, single precision may induce severe round-off errors (Heijungs & Sumner, in prep). Therefore, we will have to go for the double type or extended representation of floating point numbers. But still, we think that this falls within what today's desktops can handle.

Perhaps sparse matrix routines might be needed to save memory. This also helps to speed-up some types of calculation. But as a downside sparse matrix come with extra computational overhead. Also sparse matrix routines are most effective in reducing the size of a matrix when the matrix is of a particular type (block diagonal or tridiagonal). The need and benefits of using sparse matrix routines will be investigated using a real-size test when the data is there.

Resource management doesn't seem to be an issue for the RDBMS in the Exiopool system. The amount of data to be stored in the RDBMS and the amount of memory used by the RDBMS are likely to be far within the limits of today's desktop computers. This has been observed in a database test which has been performed as part of the present workpackage.

As specified in the requirements, the EXIOPOL system has to run as a stand alone program on regular desktop hardware. Performance, measured as the time for completion of an action may be a big issue for some parts of the EXIOPOL system. Many of the transformations between the different data bases in the system will likely require a lot of time on regular desktop hardware. However, as the transformation from a non-EXIOPOL classification to an EXIOPOL classification is done outside the system, the computational efforts will be restricted to three steps (cf. **Figure 3.4**):

- data import, using the import scripts;
- trade linking, using the RAS scripts;
- SUT-to-IOT conversion, using the IO script;
- data export, using the export scripts;

The import module – reading dozens of files, checking the format and contents of these files and writing to the first RDBMS – will likely take quite some computing time. The transformation of the data in the first RDBMS to the second and third RDBMSs will also take quite a while because of the need to invert matrices for the RAS procedure and – depending on the approach taken – for the treatment of secondary products in the SUT.

The whole EXIOPOL system design is created with the knowledge that the transformation steps are computationally intensive and therefore the intermediate calculations have to be stored. If the calculations would have been extremely quickly (< 1 s) there was less need to store calculation results. They could have been created on-the-fly.

On the other hand, querying the RDBMS needs to be fast. A first test was performed for the first RDBMS, using MS-Access. It showed that this is possible and performs well on regular desktop hardware.

4.6 Issues of security

Security of software and data touches several issues:

- the restriction of access to view data to a selected group of users;
- the restriction of access to enter, delete and/or modify data to a selected group;
- the prevention of unintentional entry, deletion and/or modification of data;
- the design of a backup policy to safeguard software and data in case of calamities.

In the EXIOPOL project, the present phase is one of development. No publicly available software and/or data will be there during the project. After the end of the project, this may well change. During the last phase of the EXIOPOL project, the target groups for using and entering data will have to be specified (see also Section 4.7). This may lead to the specification and implementation of access rules.

At present, the software codes reside at CML. Raw data is converted into a standardized format by the different partners (like NTNU, RuG, etc.) and delivered to CML in Excel format, for entry and further processing in the database. As a professional organization, CML has standard security protocols for restricting access to the data systems as well as for making backups of user data. At a later stage, the software and data, including the underlying raw data and software codes, will be disseminated, first to the project team members, and later to user groups and/or an administrator that will continue the efforts after 2011. This is to be decided by the project board and/or commissioner.

As the requirements specify that the Exiopol system will be used as standalone system no special security mechanisms will be incorporated in the EXIOPOL system. This needs to be done by each user according to his safety policy. It also means that we assume that only 1 user at a time will have access to the EXIOPOL system because everybody will have one's own working copy. Moreover this user will have full control over everything in the EXIOBASE system.

If at any point the databases in the EXIOPOL system are going to be directly assessable through a web-interface, security will become an important issue. As security has not been build into the RDBMS and EXIOPOL system from the ground up it is likely that the whole design and implementation of the EXIOPOL databases needs to be reconsidered as to critical access rules.

4.7 Issues of dissemination

In this section, we discuss various functional demands that relate to the way in which the database should be made available. Note that we restrict the discussion here to users of the database, and that the persons that build and maintain the database, including the EXIOPOL team, may have a quite different way of working.

A first requirement is that the data should be made available to a large group of users in a convenient way with minimal resources needs. This already restricts the way of publication to a digital one, although supporting information may be made available in hard copy form. Dissemination of the database can proceed in different ways:

- through an internet page with links to downloadable datasets, like Eurostat does;
- through an internet access for subscribers only, like the Swiss ecoinvent database;
- through CD-ROMS that can be obtained for free or for a certain fee at a central agency.

In this cyber era, the first two options appear to be preferable. Our proposal is thus that the EEIO database result will be made available through an internet page. Whether or not subscription is needed, and if so, what the conditions and fee should be remains to be decided. Also, continuation of the internet access after the EXIOPOL project has been finished is a point of concern. This will have to be resolved in the same manner as error repair and updates will be handled.

A second issue concerns the form in which the database is offered. Again restricting the discussion to digital dissemination, we envisage a number of obvious possibilities:

- Microsoft Excel format (or compatible with that, e.g., for use in OpenOffice);
- Microsoft Access format (partly compatible with OpenOffice);
- professional database (e.g., Oracle or Paradox);
- Matlab format;
- plain ASCII format;
- XML format.

The use of Matlab and professional databases has the clear disadvantage that they are not available for many users. They require the licensing of expensive programs (several thousands of euros is not uncommon), and they require the user to learn to operate a new program. The use of ASCII has the disadvantage that they offer for large datasets too little structure. In that respect, the use of XML would be a good compromise between simplicity and structure. The formats of Microsoft Excel and Access have the great advantage that most potential users possess these programs and have experience in working with them. Excel has in this respect an important limitation: its capacity is restricted to 255 columns, so that large IO or S&U tables must be broken into smaller blocks. We think that Microsoft Access offers the best compromise for the dissemination of results, perhaps with the addition of a few macros to facilitate the conversion into Microsoft Excel and XML.

A third question is related to the difference between database results and database management system. The EXIOPOL project comprises many sorts of information: IOTs, SUTs, NAMEAs, data from EPER, auxiliary data, etc. These data will be transformed, aggregated, combined, processed, etc. into the final EEIO. A question that we will have to answer is what we will make available to the outside world: only the final EEIO, or also the much larger and more complicated database management system including all primary data and auxiliary data and transformation steps. An answer to this question should involve many aspects: whether we think non-EXIOPOL people can understand the complex database structure, whether we want users to be able to update primary data sources or to adapt classifications, etc. Our tentative answer is a partial Yes: the database management system is an EXIOPOL internal thing, and all raw data sets will not be made available. What will be public is data at three levels:

- the different country level SUTs with environmental extensions;
- the trade-linked MRSUT with environmental extensions;
- several derived trade-linked MRIOTs with environmental extensions.

See also **Figure 3.5**, where “View” and “Export” buttons suggest which data may be retrieved by external users.

4.8 Scalability

Scalability is an issue when developing RDBMS that might become very big and the processing of all request has to be distributed among more servers. Given that the EXIOPOL system is to operate as a standalone program, serving one user at a time from a single desktop computer, scalability is not an issue. In other words, scalability is not required, unless a central decision is made for the development of a multi-user version.

4.9 The user interface

The focus of the EXIOPOL system will be on robust and flexible storage of environmentally extended input-output data that can be used for a wide range of applications. A graphical user interface which provides an entry to the RDBMS which are part of the EXIOPOL system, will not be made. The RDBMS itself is implemented in MS-Access. This program provides in itself by default many graphical tools that can be used by the user to get an overview of the data or for data manipulation. Therefore a custom GUI for the RDBMS is not necessary.

If at any time a GUI is deemed necessary it can easily be created within MS-Access or build as a separate layer on top of MS-Access.

The data transformation routines will work as command line programs only. This has the advantage that they can be called from external programs which makes further automation possible.

4.10 Issues of internationalisation and localisation

One of the issues in a MRSUT/MRIOT is the use of multiple languages, alphabets and other conventions. As all deliverables of EXIOPOL will be done in English, the menus and dialogues of software and data will only be developed in English as well.

There are, however, other issues where this issue shows up. For instance, Carbon dioxide is in Greek Διοξείδιο του άνθρακα and in Bulgarian Въглероден диоксид. Even the Latin-based alphabets use diacritical signs (ü, ñ, ç, etc.) in their names for products, emissions, etc. The EXIOPOL nomenclature (DIII.4.b-1) has been designed in a way that the use of foreign alphabets and diacritical signs are not needed, only base ASCII is used. This facilitates the design of the EXIOPOL system with respect to data entry and export.

Another complication that must be mentioned is the notation of numbers. The number pi can be written as 3.14 or as 3,14. Many software systems allow for a global variable, the decimal separator, to be set at custom. However, this sometimes still yields conflicts between programs. In EXIOPOL we adopt the convention of pocket calculators and English speaking countries: the point is the decimal separator. No thousands separator is used (so not 3.141,592,6), but scientific notation (3.1415926E+00) is allowed.

4.11 Missing data

The issue of missing data requires special attention. SUTs and IOT are quite sparsely filled. This can be due to two reasons: data may be missing, or data may have the value zero.

We propose that at import, data tables specify only non-zero elements, all other cells of a SUT can be assumed to have a value zero. Viewing or exporting data in matrix format will

the insert explicit zeros at those places. These data cells will also be treated as zeros in all manipulations (matrix multiplication, matrix inversion, etc.) in the diverse scripts.

Real zeros that are known to be zero may be entered in the system. They will not be stored in the tables of the RDBMS, however, for reasons of maintaining storage capacity and processing speed.

4.12 Error processing

No special requirements have been formulated with respect to the way errors have to be processed by the EXIOPOL system. Nonetheless for a consistent approach error processing is discussed in this high level design document. First we must distinguish between error processing during the development phase of the EXIOPOL system and the production version of the EXIOPOL system.

During development both assertions and error checking routines will be incorporated in the program. In the production version only error checking routines will be left in the program as lots of error checking code including assertions, makes the program slow and bloated. Compiler directives will be used to assign the error checking routines to the development version and the production version.

Assertions are snippets of code that are used during development that allows the program to check itself as it runs. When an assertion is true, that means everything is operating as expected. When it's false, that means it has detected an unexpected error in the code. Assertions are primarily for use during development and maintenance. Assertions are used to catch conditions that should never occur. Error-handling code is used for conditions that are expected to occur. Error handling typically checks for bad input data, assertions check for bugs in the code (McConnell, 2004).

In the production version, error-handling routines must handle the exceptions gracefully, providing the user with information about the exceptional situation encountered. On the other hand, errors in the development version may make the program fail hard.

The error-handling and error logging routines will be centrally managed (maybe in a single class). This avoids having all kinds of different error message routines throughout the code.

For the detection of errors, the import routines are of prime importance. The import routines will read some kind of file containing data and will write these data to the RDBMS. The data in the files may not be in the expected format and it essential that the data are sanitised before they are entered in the RDBMS. The import routines likely will contain a lot of error checking routines. On the other hand, routines that get their data from the RDBMS can be sure that the data will be in the correct format and less error checking of the data will be necessary because all the data in the database are fully typed. Concluding data errors can only occur in the import module and any error somewhere else in the system is surely a program error.

Error messages will not only be shown on the screen but also a file will be maintained that logs the error messages. It will be very useful for debugging but also for correction of errors in the data files that need to be imported into the EXIOPOL system.

4.13 Database testing

Testing of the database involves the validation of the functionality implemented by, and data contained within, the database (Ambler S.W., 2007). Testing the operation of the database can be done from two points of view. One point of view is from the outside as experienced

by applications using the database and another viewpoint is from within the database. Testing from the first viewpoint can be called black-box testing the second viewpoint can be called clear-box testing (Ambler S.W., 2007).

Black-box testing involves answering question like:

- an SQL statement returns an expected result;
- SQL calculation returns the expected result;
- A query returns a result within a reasonable time.

Clear-box testing involves checks on:

- stored procedures/functions;
- triggers;
- constraints;
- existing data quality;
- referential integrity/data consistency.

Clear-box tests could be in terms of “shouldHaveUniqueAccountID” or “shouldHavePositiveAccountBalance”.

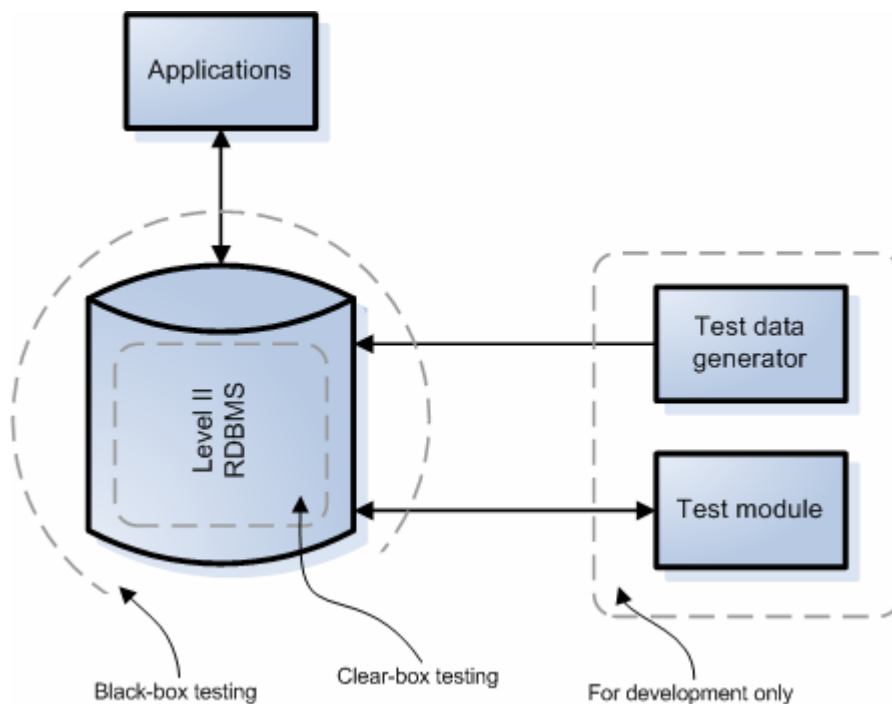


Figure 4.1: Testing the RDBMS. The test includes the creation of a test data generator and a test module. These two modules are available in the development version only.

During development, both black-box and clear-box test routines will be incorporated in a test module. Sample data will be needed to be able to test the data. A test data generator will also be created as a separate module. These test modules will only be compiled into the development version of the EXIOPOL system. See **Figure 4.1**.

4.14 Requirements changes and design changes

EXIOPOL is a project which aims to achieve novel things. Thereby, the project necessarily involves regular readjustments of plans and designs. As has been observed in DIII.4.b-1, classifications may change during the project, or afterwards. The same holds true for the database architecture. Insights may grow and experience may guide us. **Figure 3.4** shows a structure with 3 RDBMSs, which we think is fairly universal. In other words, we think that this overall design is likely to be stable. Details within each of the RDBMSs are much more likely to change: new entities may show up to be needed, entities may be merged, relationships may turn out to be different. The use of Access as a development facilitates the structured working on a system in progress. Changes of names of tables can be incorporated into the SQL-queries quite easily. And it is the stored procedures which are called from the Delphi program(s). So, it is our conviction that the modular structure in combination with the rigidity of a relational database set-up and the use of SQL is a flexible one with respect to changes in the requirements and/or design.

4.15 Revision control

Software that's worth creating is worth putting under revision control (Ambler, 2007).

Although we have considered the use of revision control software like CVS or Subversion, no revision control software is used in the project. There are only few people working on the construction/coding of EXIOBASE and they are working in the same institute. If after the project revision control is required in maintenance and updating, such routines can be added then as easily as now.

List of references

- Ambler, S.W. (2007) Test-driven development of relational database. *IEEE Software* May/June, 37-43.
- Bouwmeester, M.C., & J. Oosterhaven, (2008) Technical report: The EXIOPOL database management system – Inventory of trade data and options for creating linkages. EXIOPOL DIII.1.a-3.
- Kop Jansen, P. & T. ten Raa (1990) The choice of model in the construction of input-output coefficients matrices. *International Economic Review* 31:1, 213-227
- McConnell, S. (2004) *Code Complete* 2nd ed., Microsoft Press, Redmond, isbn 0-7356-1967-0
- Cantù, M. (2005) *Mastering Borland Delphi 2005*. Sybex, San Fransisco, London, isbn 0-7821-4342-3
- De Koning, A., R. Heijungs & G. Huppes (2008a) Technical report: The EXIOPOL database management system – nomenclature and transformations. EXIOPOL DIII.4.b-1.
- De Koning, A., R. Heijungs & G. Huppes (2008b) Technical report: The EXIOPOL database management system – Protocols for data input. EXIOPOL DIII.4.b-1a.
- Nijdam, D.S., H.C. Wilting, M.J. Goedkoop, J. Madsen (2005) Environmental load from Dutch private consumption. How much damage takes place abroad? *Journal of Industrial Ecology* 9:1/2, 147-168
- Peters, G.P. & E.G. Hertwich, E.G. (2006) The importance of imports for household environmental impacts. *Journal of Industrial Ecology* 10:3, 89-109
- Tukker, A. et al. (2006) *EXIOPOL; a new environmental accounting framework using externality data and input-output tools for policy analysis*. Annex I - Description of Work.
- Tukker, A. & R. Heijungs (2008) Technical report: Definition study for the EE IO database. EXIOPOL DIII.1.a-5.

Annex I: Contributors to the report

This report is the result of discussions between all partners in the EXIOPOL consortium. It has been edited and written by the following persons:

Arjan de Koning, Institute of Environmental Sciences (CML) - Universiteit Leiden

Reinout Heijungs, Institute of Environmental Sciences (CML) - Universiteit Leiden

Gjal Huppes, Institute of Environmental Sciences (CML) - Universiteit Leiden