An evaluation framework for database design transformation in CASE-tools

(HS-IDA-MD-99-103)

Claes Johansson (claesj@ida.his.se)

Supervisor Björn Lundell
Submitted by Claes Johansson to the University of Skövde as a dissertation toward
the degree of M.Sc. by examination and dissertation in the Department of Computer
Science.

1999-09-21

I certify that all material in this dissertation which is not my own work has been
identified and that no material is included for which a degree has already been
conferred upon me.

Signed: _______________________________________________
An evaluation framework for database design transformation in CASE-tools

Claes Johnsson (claesj@ida.his.se)

Keywords: Evaluation, evaluation framework, design transformation, CASE-tools

Abstract

In this report a method is applied that was originally developed for the creation of evaluation frameworks for CASE-tools. Here the method is taken out of its original context and applied for the creation of an evaluation framework for database design transformation in CASE-tools. The focus of this report is on the process of creating the framework and not on the resulting framework.

The creation of the framework is divided into two main processes. During phase 1 a framework is developed from existing literature. This framework is then refined in phase 2 when the framework is tested in an in-depth exploratory study of support for design transformation in two commercial CASE-tools.
# Table of contents

1 Introduction............................................................................................................ 4

2 Background ............................................................................................................ 6
   2.1 Evaluation and database design ......................................................................... 6
      2.1.1 Evaluation and evaluation frameworks .................................................. 6
      2.1.2 Database design ....................................................................................... 6
   2.2 Design transformations, models and CASE-tools ............................................... 8
      2.2.1 Design transformations ............................................................................ 8
      2.2.2 Models ................................................................................................... 10
      2.2.3 SQL-92 .................................................................................................. 11
      2.2.4 CASE-tools ............................................................................................ 11

3 Method support for development of evaluation framework ..................................... 13
   3.1 Method background ......................................................................................... 13
      3.1.1 Origin and motivation for the method ..................................................... 13
      3.1.2 Method process ...................................................................................... 13
   3.2 Using the method for reconciliation of reported work in the literature............. 16
      3.2.1 Grounded theory and Literature ............................................................ 16
      3.2.2 Laboratory experiment .......................................................................... 17
   3.3 Applying the method to design transformation ............................................... 18
      3.3.1 Context for the study .............................................................................. 18
      3.3.2 Focus of the study .................................................................................. 18

4 Phase 1 a literature based framework ................. 20
   4.1 The process of creating a literature based framework ...................................... 20
      4.1.1 Data collection ....................................................................................... 20
      4.1.2 The initial concepts ............................................................................... 20
      4.1.3 Coding and analysis .............................................................................. 25
   4.2 Results from Phase1 a literature based framework ......................................... 29
      4.2.1 Information preservation ....................................................................... 29
      4.2.2 Verification of the forward engineering framework ................................. 29
      4.2.3 Validation of the forward engineering transformation ............................ 30
      4.2.4 Traceability ............................................................................................ 30
5 Phase 2 A study of support for design transformation in two commercial CASE-tools......31

5.1 The tools .......................................................................................................31
  5.1.1 ER/Studio 3.0 ........................................................................................31
  5.1.2 Modelator 4.0 ........................................................................................32

5.2 Applying the method.....................................................................................33
  5.2.1 Analysing the tools support for the concepts that were created in phase 133
  5.2.2 Testing the framework against a schoolbook example .........................36
  5.2.3 Finding totally new concepts by examining the tools ............................37
  5.2.4 Analysis of the study ............................................................................37
  5.2.5 Refining the framework .......................................................................39

6 Analysis...........................................................................................................41
  6.1 Criteria’s for the analysis ...........................................................................41
  6.2 Analysing the process of creating an evaluation framework ......................41
  6.3 The final framework ...............................................................................43

7 Conclusion and discussion..............................................................................45

References: .......................................................................................................46

Appendix A Example of transformation between IE and SQL (after Teorey, 1991)................................................................................................................49

Appendix B Neutral transformations to eliminate IS-A relations (after Hainaut et al., 1996) ........................................................................................................56

Appendix C Repository and versioning control..............................................58

Appendix D Initial transformation in Modelator ............................................59

Appendix E Test of the transformation example presented by Theory(1991) in Modelator........................................................................................................61

Appendix F Test of the transformation example presented by Theory(1991) in ER/studio........................................................................................................71

Appendix G Customisation of the forward engineering process in ER/Studio...81

Appendix H Small Airport database (after Elmasri et al., 1994) ....................84

Appendix I Small Airport database SQL code..............................................85
Appendix J  Report generation in ER/Studio................................................................. 88
Appendix K  Repository support in Modelator ......................................................... 89
Appendix L  Repository support in ER/Studio......................................................... 90
Appendix M  Relations between concepts and sub concepts in the refined framework .................................................................................................................. 91
Appendix N  List of quotation in this report (in chronological order)............. 92
1 Introduction

Evaluation is an inherently complex process (Malmsjö, 1998). To address an important aspect of this processes Lundell and Lings propose a method for the development of evaluation frameworks for CASE-tool evaluation (Lundell and Lings, 1998, 1999; Lundell et al., 1998, 1999; Lundell, 1999). It is the author’s intention to use this method for the development of an evaluation framework for database design transformation in CASE-tools.

The aim for this project is to give some understanding of how the method for development of evaluation frameworks proposed by Lundell and Lings (1998 1999); Lundell et al., (1998 1999); Lundell (1999) works when taken out of its original context, and used for the creation of an evaluation framework for database design transformation in CASE-tools.

For this project a number of objectives were established:


2) Analyse different techniques for database design transformation using relevant literature.

3) Develop an evaluation framework for database design transformation in CASE-tools.

4) Refine the framework developed in objective 3 by conducting an in-depth exploratory study of support for design transformation in two commercial CASE-tools that supports SQL and the Enhanced Entity-Relationship model (EER).

5) Analyse and compare the result of the test.
1 Introduction

Objectives 2 and 3 relate to an application of the first iterative phase of the method developed by Lundell and Lings. Likewise, objective 4 relates to the second phase in the method.

The main contribution in this project is a test of transferability of the method proposed by Lundell and Lings (1998 1999); Lundell et al., (1998 1999); Lundell (1999) to a new context and applied in a different setting.

The secondary contribution is a framework for database design transformation as realised in CASE-tools. It is important to point out that the focus of this project will be an investigation of the process for establishing an evaluation framework, and not on the resulting product from applying the method (i.e. the evaluation framework).

This report is organised as follows: In chapter 2 a brief introduction is given to the concepts of database design and evaluation. The concepts design transformation and models (with a focus on SQL and EER) can also be found in chapter 2. In chapter 3 the original method developed by Lundell and Lings is briefly described. The chapter also describes the differences between the original settings for the method and those in this report. Chapter 4 shows the process of creating the literature based framework (phase 1). In chapter 5 phase 2 (an in-depth exploratory study of support for design transformation in two commercial CASE-tools) is described. In the final chapters, 5 and 6, the process of creating the evaluation framework and the framework itself are analysed and results and discussion presented.
2 Background

2.1 Evaluation and database design

2.1.1 Evaluation and evaluation frameworks

Malmsjö (1998) defines the concept evaluation as a process where:

(a) performance of a system (activity) is specified, and

(b) the outcome of this elucidation is compared with the ends (goals, objectives) of the system, and

(c) the difference between these two parameters can constitute an essential basis for corrective actions.

Malmsjö (1998) writes that the idea is that the evaluation is done continually.

Lundell et al., (1999) writes that a necessary prerequisite for the validity of any evaluation process is that there is a common understanding among involved stakeholders with respect to what factors should be considered when assessing the product to be evaluated. The term evaluation framework has been used to refer to a set of such factors.

2.1.2 Database design

Database design is concerned with representing a ‘real world’ situation by means of a database system (Benynon-Davis and Lloyd, 1992). Elmasri and Navathe (1994) describe their version of database design to be conducted in a number of steps (see figure 1). According to Elmasri and Navathe (1994) the first step in database design is
2 Background

requirements collection and analysis. In this step the database designer collects the different requirements that the users of the forthcoming database have. The result of this phase is a written set of user requirements.

When the database designer has collected user requirements, the next step is to create a conceptual database design. A conceptual schema is developed in a high-level data model, for example the Entity-Relationship (ER) model. The conceptual schema is a detailed description of the data that emerged during requirements collection and analysis. The conceptual schema also shows the relationship between the data, data types and constraints.

When the conceptual model is complete and correct it is time for the implementation of the database. According to Elmasri and Navathe (1994) a commercial DBMS (Database Management System) will be used for this. The DBMS uses an implementation data model so the conceptual schema has to be translated (see design transformation) into the implementation data model. Elmasri and Navathe (1994) call this step the Logical design or data model mapping.

The final step in database design is physical database design. In this phase the internal storage structures and file organisation will be defined (Elmasri and Navathe, 1994). Parallel with the database design are the development of the application programs (Elmasri and Navathe, 1994).
2 Background

2.2 Design transformations, models and CASE-tools

2.2.1 Design transformations

There are four types of transformations that can be characterised as design transformations in a database context (Fahrner and Vossen, 1995). The first one is the
2 Background

forward engineering transformation. Forward engineering is the transformation of a high-level conceptual data model into a logical data mode (Fahrner and Vossen, 1995). A model, according to FRISCO (1998), is defined as “a purposely abstracted, clear, precise and unambiguous conception.” (FRISCO, 1998 p. 55).

The second type of design transformation is the reverse engineering transformation. Reverse engineering is the transformation of a logical model into a high-level conceptual data model, or as Fahrner and Vossen (1995) describes it, “transformation of some logical model into the formalism of the ER model.” (p. 214). Souto (1998) defines the purpose of Reverse engineering as, “to produce a conceptual description of a given database where the input may consist of any combinations of source code description, a data dictionary, a database instance, and application programs.” (Souto, 1998)

The concept of design transformation also encompasses the transformation of one type of high-level conceptual data model into another. Missaoui (1998) uses the term schema transformation when he refers to this type of transformation.

The final transformation considered under information preservation is the transformation of one type of logical data model into another. Fahrner and Vossen (1995) write that “Transformations of this kind [(between two logical models)] seems to be of increasing practical importance,” (Fahrner and Vossen, 1995 p. 239).

In this project I will focus on the forward engineering part of design transformation.
2 Background

2.2.2 Models

The ER model developed by Chen (1976) and the different variations of the model are frequently used for conceptual design of databases (Elmasri and Navathe, 1994). There is currently no standard ER model and many books and papers have introduced various semantics to be included in the original approach Teorey et al., (1986).

The version of the EER model proposed by Elmasri and Navathe (1994) includes all the modelling concepts of the ER model. This version of the EER model also includes the concepts of subclass and superclass, with the related concepts of specialisation and generalisation.
2 Background

The specific EER notation that I will be using for the development of the evaluation framework in this project will be Martin’s Information Engineering (IE). An introduction to the notation can be found in Martin (1990).

2.2.3 SQL-92

SQL is a data sublanguage that is used for accessing a relational database that is managed by a relational database management system (RDBMS) (Melton and Simon 1993). There are different versions of SQL. In this project SQL-92 will be used as the target language for transformations. The reason for this is that the SQL-92 notation is a standard. For an introduction to SQL-92 see Melton and Simon (1993) and Date and Darwen (1997).

2.2.4 CASE-tools

According to Lundell and Lings (1998b), there are different interpretations of the concept CASE. Bubenko and Wangler (1992) define a CASE-tool as “a software environment, that assists a systems analyst and designer in the process of designing, specifying, analyzing, and maintaining a software product (an information system)”. (Bubenko and Wangler, 1992).

Kelly et al., (1996) writes that “CASE technologies are expected to provide task related support for software developers in analysing, designing and implementing a set of information systems (IS) or their components according to a method.” (Kelly et al., 1996 p.1)
2 Background

According to Loucopoulos et al., (1995) the overall aim of CASE technology is to improve the productivity and quality of the resulting systems by assisting the developer throughout the different stages of the development process. CASE provides the software tools that support methodologies used in modelling all levels of an organisation, so it may be described as software tools for enterprise support consisting of enterprise and information systems strategic planning, project planning systems development, documentation and maintenance.

Bubenko and Wangler (1992) partition the functional features of CASE-tools as:

- User interaction: Most of today’s CASE-tools permits the designer to work with graphical forms and textual input and output in some kind of windows mode.

- Design support: is support for transformation of specifications from one “level” to another. Examples are view integration, restructuring of a specification and transformation of a non-executable requirement specification into an executable specification for prototyping.

- Development work and project management: includes planning and control. Support should be included for recording and maintaining design history and design decisions, communication facilities, authorisation management, change management and tracking and decision tracing support.
3 Method support for development of evaluation framework

3.1 Method background

3.1.1 Origin and motivation for the method

The method that will be used for the development of the evaluation framework is originally proposed for the development of evaluation frameworks for CASE-tool evaluation and is proposed by Lundell and Lings (1998 1999); Lundell et al., (1998 1999); Lundell (1999). The method is based on a qualitative research approach that is strongly influenced by Grounded theory as developed by Glaser and Strauss (1967). The goal for the method is to establish an evaluation framework for an adoptive environment (Lundell et al., 1999).

The idea that an evaluation framework should cover both the organisational needs and what the current technology can offer is one of the cornerstones in the method (Lundell et al., 1999). Another aspect that the method emphases is that involved stakeholders shall have a common understanding and definition of the different concepts that emerge during the development of a framework.

3.1.2 Method process

The method consists of two phases. The aim with phase one is first to reach a consensus among the stakeholders about the meaning of the different concepts in the framework and to create a ‘alpha version of the evaluation framework’.

The first phase consists of three main process. These are:
3 Method support for development of evaluation framework

- Data collecting
- Coding
- Analysis

The processes are not performed in a sequence. The idea is instead that they are to be performed together as much as possible.

Planning of the process of collecting all the data that are to be used in the framework can not be done in advance according to grounded theory (Glaser, 1992); (Lundell and Lings, 1998). The data collection is instead dependent upon the emerging framework.

The data source used for the data collection can be either organisational (from the organisation itself) or public (Lundell and Lings, 1998).

The coding process is divided into two sections: Open coding and selective coding (Lundell and Lings, 1998). During open coding the substantive codes are developed. These substantive codes represent the different concepts that are to be used in the framework. During selective coding the theoretical codes are developed. The theoretical codes show the relationships between the different substantive codes i.e. the relationships between the different concepts in the framework. The analyse is done continually during the entire framework development process.

In the second phase the emerging framework is used in a pilot evaluation of CASE-tools. The purpose of phase 2 according to Lundell (1999), is to:

- Improve precision
- Increase realism
- Expand the emerging framework
3 Method support for development of evaluation framework

After phase 2 the framework is cycled back to phase 1 and interpreted within the organisational context, which in this project implies the investigator's reflections and re-interpretation of the original phase 1 framework.

The overall process stops when the framework is considered stable and effective (Lundell et al., 1999). An overview of the method can be seen in figure 3.

Figure 3 The original method developed by Lundell and Lings (after Lundell et al., 1999)
3 Method support for development of evaluation framework

3.2 Using the method for reconciliation of reported work in the literature

3.2.1 Grounded theory and Literature

Grounded theory has its origins in social science (Glaser and Strauss, 1967). For a motivation of the use and possible gain from the use of grounded theory in a CASE-tool context read Lundell and Lings (1997).

According to Glaser (1992) there are three types of literature with regards to grounded theory methodology. These are:

- non professional literature
- professional literature related to the area under research.
- professional literature that is unrelated to the substantive area.

Non professional literature, according to Glaser (1992) consist of “pure descriptions of various sorts with virtually no or minimal conceptualizations”. Glaser (1992) further assets that non professional literature can be read at any stage of the research as data, and helps the generation of concepts and hypotheses. The main thing of interest regarding grounded theory and non professional literature is that it can be considered, which means that it can be valid to use non professional literature in the first phase of this project. Examples of relevant non professional literature include non research data press and user manuals for CASE-tools.

Glaser (1992) writes that the user of grounded theory shall avoid professional literature related to the area under research in the beginning of the study. When the theory seems sufficiently grounded in a core variable and in an emerging integration of core variables, then the researcher can start to used the professional literature.
3 Method support for development of evaluation framework

related to the area under research. This will obviously be impossible for a study that uses this literature for the development of the initial evaluation framework. However, this does not invalidate the application of the method of Lundell and Lings which is specifically only influenced by grounded theory.

Glaser (1992) also writes that it is important to read and study literature, but from unrelated fields. In my literature study I will try to find influences for my evaluation framework not only from literature about transformation between EER and the relational model, but also from literature that treats other sorts of transformations i.e. transformations between ER and object oriented models.

3.2.2 Laboratory experiment

In phase 2 of this project I use a form of laboratory experiment for the refinement of the evaluation framework that was created in phase 1. According to Wynekoop (1991) it is possible to characterize a laboratory experiment as a researcher–created setting with experimenter control over the different variables that the experiment is conducted on.

CASE-tool context can, in a laboratory experiment, entail systematic variations of tool or method. One big disadvantage with laboratory experiment is that it is an assumption that the real-world context can be ignored (Wynekoop, 1991).
3 Method support for development of evaluation framework

3.3 Applying the method to design transformation

3.3.1 Context for the study

In the method described by Lundell and Lings (1998 1999); Lundell et al., (1998 1999); Lundell (1999) the framework is developed with consideration given to the organisational setting. In this project there will not be an explicit organisation to develop the evaluation framework for. It would be possible to say that the organisation that this framework will be developed for is the entire database community. The database community is a very big organisation and will in this project be a ‘silent organisation’ i.e. there will not be any two way communication between the author of this report and the community. For the same reasons it will be impossible to reach a consensus understanding among the different ‘stakeholders’. The only thing that I can do instead is to be very precise when I define what I mean with the different concepts that emerge during this project (something that is important in all research).

In the method proposed by Lundell and Lings the interviews with the different stakeholders is a very important process. This project will instead use literature sources.

3.3.2 Focus of the study

The literature that has been used for this framework is mostly professional literature related to the area under research. There is not a wealth of literature directly related to qualitative issues of the transformation between EER and SQL-92. There are many papers written about the transformation between different conceptual models and other types of transformation, and some of these papers were useful for the development of
the literature based framework. There is also a lot of material written about database
design in general. In the beginning I read and gathered information quite freely, but
when the different concepts started to emerge I shifted the focus of my literature study
towards these concepts. The data collection, the coding and analysis where done in
parallel. I thereby claim that I have followed the method proposed by Lundell and
Lings (1998 1999); Lundell et al., (1998 1999); Lundell (1999), with regard to data
collection, coding and analysis.

The CASE-tools chosen for phase 2 were required to have some basic features if they
were to be used to improve the framework. The tools have to be able to produce some
kind of SQL code. The CASE-tools also have to have an EER notation as their
conceptual model.
4 Phase 1 a literature based framework

4.1 The process of creating a literature based framework

4.1.1 Data collection

During initial data collection it was my intention to find concepts for the open coding part of my framework with a very wide focus. My goal was to find as many concepts as possible. During this stage of the development of the framework I kept my mind open for all types of concepts. Concepts that were not directly relevant to my problem were also added, the reason being that they could be useful during selective coding. I also added concepts to my ‘list’ that had similar meaning.

In the following part of this chapter I will present the process of establishing the framework.

4.1.2 The initial concepts

During the initial part of my investigation of the database literature I looked for concepts that could be of interest for my framework. If a parallel is to be made with the method proposed by Lundell and Lings this process is related to the initial data collection and open coding in the original method. The first concept that became a candidate for the framework was Information preservation.

According to Fahrner and Vossen (1995) the goal of any schema transformation is to convert the source data model into a schema with the same “information capability as the target model.” (Fahrner and Vossen, 1995 p. 219). If a transformation fulfils this
goal it is Information preserving according to Fahrner and Vossen (1995). Fahrner and Vossen’s (1995) definition of information preservation made it clear to me that some kind of concept is required that indicates that the information in the source schema and in the target schema is the same after the forward engineering transformation. When I continued consulting the literature I found the concept ‘transformation correctness’ that seemed to have interesting similarities with the information preservation concept described by Fahrner and Vossen (1995). I added the transformation correctness concept to my list of candidates for the framework. Gogolla (1997) uses the term transformation correctness when he addresses the transformation between different conceptual models. Gogolla (1997) writes that a transformation is correct if “the next conceptual model has the relevant properties the previous model required” (Gogolla, 1997). Gogolla’s (1997) ideas of transformation correctness indicate that the target schema must have the relevant properties of the source schema and that this is a factor that must be evaluated during the evaluation process of the forward engineering transformation. The paper presented by Gogolla (1997) addresses transformations between different conceptual models. In this framework the source model is a conceptual model (EER) and the target model is a logical model (SQL-92) this makes it necessary to make some kind of translation of the concept to this context. If a transformation is to be correct in this framework no features shall be lost in the transformation, for example all the attributes in an entity are to be translated to SQL code during the translation.

Another concept that seemed to be related to the concepts earlier described was semantic preservation (Hainaut et al., 1996). Hainaut et al., (1996) uses the term semantic preservation, a concept that says that not only is the information to be the same in the target schema as in the source schema, but also the semantics (Hainaut et
4 Phase 1 a literature based framework

al., 1996). According to Hainaut et al., (1996) a transformation between two schemas is semantic preserving if the two schemas describe the same real world portion.

The papers presented by Fahrner and Vossen (1995), Gogolla (1997) and Hainaut et al., (1996) indicate that an important property of any transformation is that the two schemas have the same semantics, information and properties. In this framework the concept information preservation will address all these factors, and is a very important concept to study during the evaluation of the forward engineering transformation.

The report presented by Lundell et al., (1999) shows that an organisation might need to customise the forward engineering transformation, for example for efficiency reasons. The capability to customise the forward engineering process does obviously conflict with the information preservation goal as it makes it possible to add or delete information during the transformation process. Lundell et al., (1999) shows that customisation of the forward engineering process is a concept that it is important to study during the evaluation of the forward engineering process.

During the analysis of the paper by Bubenko and Wangler (1992) I realised that the concepts of validation and verification of the forward engineering transformation would be interesting to evaluate.

Bubenko and Wangler (1992) write that “The Verification and validation step has the objective of checking whether the formal conceptual specification is consistent, and whether it correctly expresses the functional requirements, by the users” (Bubenko and Wangler, 1992 p. 3). Pressman (1997) uses the concepts validation and verification in a software testing context. “Verification refers to the set of activities that ensure that software correctly implements a specific function. Validation refers to a different set
of activities that ensure that the software that has been built is traceable to customer requirements.” (Pressman, 1997).

Bubenko and Wangler (1992) writes that the “verification problem” can be viewed at the syntactic and at the semantic level. This indicates that if it is going to be possible to verify the transformation process we do have to analyse the diagram for inconsistency before the transformation. The (ISO/IEC, 1995) standard has one atomic sub characteristic that is called diagram analysis and defines the concept as “attributes relating to its ability to support the analysis of graphical figures input to the CASE-tool and extracting and storing requirements and/or design information.” (ISO/IEC, 1995 p. 27).

During the search for literature I did find rules for how the transformation between SQL and EER can be conducted. The rules are created by Teorey (1991) and can be found in appendix A.

SQL-92 has a number of foreign key constrains (Melton and Simon, 1993). The rules presented by Teorey (1991) show how these foreign key constrains are to be used; this can be seen in appendix A, where different transformation between IE and SQL are presented (Teorey, 1991).

The author knows from earlier experience of evaluation of a CASE-tool (Bringle and Johansson, 1998) that the transformation of the IS-A relation can be problematic. The focus on the literature was therefore changed toward the problem of transforming IS-A relations into SQL. Hainaut et al., (1996) write that most CASE-tools do support some form of IS-A relation. Hainaut et al., (1996) also claims that most current database systems do not have the necessary logical constructs for the representation of the IS-A relation, something that Griebel et al., (1996) confirm when they write that “Some
EER IC [(Integrity Constrain)] properties (e.g. disjoint relationships) cannot be expressed in the abstract relational layer, but can be modeled by trigger sets of a target database system” (Griebel et al., 1996 p. 3). Hainaut et al., (1996) have proposed techniques for transformation of IS-A relations into standard constructs. Appendix B shows the three basic techniques for neutralisation of IS-A relations. The techniques can be used to understand the transformation result of the IS-A relation.

Validation was the concept that I then started to explore. Bubenko and Wangler (1992) write that “Validation can be done by paraphrasing a conceptual schema in natural language and giving that paraphrase to a user for examination.” (Griebel et al., 1996) writes that these descriptions could be in textual form. This indicates that automated report generation can be a good tool for supporting the validation process of the forward engineering transformation in CASE-tools. Report generation features in the ISO standard as ”attributes relating to its ability to automate the development of reports to be produced by the system under development” (ISO/IEC, 1995 p. 29).

A paper by Griebel et al., (1996) presented a concept that became a candidate for the framework, namely traceability. According to Griebel et al., (1996) traceability in information system design is considered an absolute requirement for advanced system development. Griebel et al., (1996) write that “Within the broader context of IS development it requires correlation of design decisions across different sub-models, and between data sets and meta data, on an ongoing basis.” (Griebel et al., 1996 p. 1). The ISO/IEC (1995) standard has one atomic subcharacteristic that is the specification traceability analysis. Specification traceability analysis is “attributes relating to its ability to perform traceability analysis.” ISO/IEC, 1995 p. 33). The support for tractability in CASE-tools during the forward engineering process is a factor that it is important to evaluate.
4 Phase 1 a literature based framework

The paper written by Griebel et al., (1996) shows that a repository can support the traceability aspects of database development. A definition of the concept repository and its relevant property, versioning management, are to be found in appendix 3.

4.1.3 Coding and analysis

The open coding and the selective coding were not performed in a sequence. In the beginning of the literature study (the initial data collection) open coding was the dominant process. Open coding is in this context the process of developing an understanding for the meaning of the concepts that are to be found in the framework. As the concepts started to emerge I did start to analyse the relations between the concepts. This process is related to the process of selective coding described in the original method. Relations between concepts are illustrated in figure 4. One example of a relation in this framework is that a concept consists of sub concept (for example information preservation, figure 5). Another type of relation is that a concept may be a goal that may be achieved with its sub concepts (an example of this is the traceability concept with its sub concepts, figure 6). The result of the open coding in this framework is the set of concepts that are candidates for the final framework.

During selective coding the three concepts information preservation (Fahrner and Vossen, 1995), transformation correctness (Gogolla, 1997) and semantic preservation (Hainaut et al., 1996) merged into one because of there similarity. I also placed the concept customisation of the forward engineering process under the information preservation part in the framework, the reason for this being that the concepts information preservation and customisation of the forward engineering process may
4 Phase 1 a literature based framework

cflict with each other. Figure 5 shows the concepts that information preservation encompasses in this frame work.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Sub concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation of the forward engineering process</td>
<td>Automated report generation (ISO/IEC, 1995)</td>
</tr>
<tr>
<td>Tractability</td>
<td>Repository support in CASE tools (Griebel et al., 1996), Version management (Bernstein et al., 1994).</td>
</tr>
<tr>
<td>Information preservation</td>
<td>Transformation correctness, (Gogolla, 1997), Semantic preservation (Hainaut et al., 1996), Information preservation (Fahrner and Vossen, 1995), Customisation of the forward engineering process (Lundell et al., 1999).</td>
</tr>
</tbody>
</table>

Figure 4 Relationship between the concepts in the framework.

Hainaut et al., (1996) rules for neutralisation of the IS-A relation shows that it is not possible to transform an IS-A relation that is information preserving to SQL-92. The transformation has to be performed in two steps, one step from an IS-A relation to an EER diagram with a lesser semantic expressiveness (the neutralisation) and thereafter the transformation to SQL-92. It is obvious that the original IS-A relation is not represented in the SQL-92 schema. The fact that it is possible to do the neutralisation in different ways is also something that indicates that the transformation of an IS-A relation into SQL-92 is not information preserving.
The diagram correctness concept was considered a part of the verification of the forward engineering process. The reason for this is that, according to Bubenko and Wangler (1992), the diagram must be correct if it is going to be possible to verify the transformation. I also added a set of rules for the transformation between EER and SQL developed by Teorey (1991) to the verification part of the framework. The reason for this is that the rules and the transformation examples can be used to compare the results of the transformation in a CASE-tool.

The two papers presented by Bubenko and Wangler (1992) and Griebel et al., (1996) showed that the ability to generate reports in the CASE-tool is an important concept for the validation of the transformation, and were therefore added to the validation part of the framework.

The concept of two way mapping was taken out of the framework. The reason for this was that I found it impossible to connect the concept to forward engineering transformation. The concept two way mapping connected to both the forward and reverse engineering. In this framework only the forward engineering transformations are evaluated.
During my study of the concept traceability I did realise that an important characteristic for traceability is its support for repository features and especially support for version control. The three concepts are not independent of each other; they are more to be seen as non-mandatory parts of each other. Figure 6 shows the relation between the three concepts and which parts are to be evaluated in this framework.

The dark grey part of figure 6 represents traceability reached with repository support and versioning management. This is the most important aspect to evaluate in this framework. The light grey areas in figure 6 represent support for traceability through either versioning management or repository features and are also important to evaluate. There are also ways to reach traceability that don't involve versioning management or repository support. These aspects are represented by the uncoloured parts of the ‘traceability circle’, and are also important to evaluate.

Figure 6 Relations between concept Traceability and its sub concepts
4.2 Results from Phase 1 a literature based framework

4.2.1 Information preservation

Applying the method developed by Lundell and Lings on design transformation, using relevant literature, shows that forward engineering is to be information preserving i.e. that the source schema and the target schema have the same semantics, information and properties. It is therefore important to evaluate whether the transformation is information preserving. The ability to customise forward engineering is a concept that may conflict with the information preservation concept and is therefore an important concept to evaluate. My analysis shows that the transformation of an IS-A relation to SQL-92 is not information preserving.

4.2.2 Verification of the forward engineering framework

The study shows that it is important to verify that the transformation has been performed correctly. One criterion for the correctness of a transformation is that there are no errors and inconsistencies in the source schema. It is therefore important that there are features in the CASE-tool to analyse the target diagram for any inconsistency before the transformation.

I have also presented rules for how the transformation between EER and SQL can be conducted. The rules can be used for the verification of the transformation. My literature study also shows that the transformation of the IS-A relation can be problematic. I have therefore added rules for how it is possible to eliminate the IS-A relation.
4 Phase 1 a literature based framework

A CASE-tool’s ability to support diagram analysis and that the transformation between EER and SQL are performed correctly are aspects that are important to evaluate.

### 4.2.3 Validation of the forward engineering transformation

My literature study shows that it is important that it is possible to validate the framework. To do this it is important that there are features in the CASE-tool that generates reports. The reports can then be used for the validation of the framework.

### 4.2.4 Traceability

According to my literature study it is important to evaluate a CASE-tool’s ability to support tractability during the forward engineering process. The literature study shows that a repository can be a good tool for the support of traceability and especially mechanisms that handle versioning control. It is therefore important to evaluate the CASE-tool’s support for repository features and especially the support for versioning control.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

5.1 The tools

The two tools that were selected for phase 2 are ER/Studio 3.0 (Embarcadero Technologies, 1999) and Modelator 4.0 (Metodedata a.s., 1999). The reason that these tools were chosen is that they are quite different. Something that the author thinks can help enriching the framework. The author has some experience from the ER/Studio tool (Bringle and Johansson, 1998); this was the main reason for choosing this tool. The Modelator tool was chosen because it performs the transformation between the conceptual model and SQL in a different way from ER/Studio. Both tools support the IE notation as a conceptual model, something that can make the comparison of the two tools easier. Neither of the tools supports SQL-92. This is of course something that will make the comparison of the transformation result from the two tools more difficult.

5.1.1 ER/Studio 3.0

ER/Studio is a CASE-tool developed by Embarcadero Technologies (Embarcadero Technologies, 1999). The tool does support both forward and reverse engineering transformations. Forward engineering is done in two steps. The conceptual model is first transformed into a physical model with lesser semantics than the original conceptual model. The physical model is then transformed into a target database or a SQL-script.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

The modelling notations that ER/Studio supports are IDEF1X, IE and Filtered IE, which is a simplified version of IE with the difference that foreign keys are not displayed in the entity boxes.

ER/Studio supports a number of target databases among others Oracle 8 and 7, DB/2 and DB/2 Universal Database and Informix OnLine and SE. It is possible to generate either a target database or a SQL script during the forward engineering transformation. When the SQL code has been generated there is a built-in program in ER/Studio named ISQL that can be used for viewing, editing and executing SQL code. ER/Studio does not support SQL-92. The target language for the transformations in this project will therefore be Oracle 8. The reason for this is that the notation is well documented.

5.1.2 Modelator 4.0

Modelator is a database design tool, made by MetodeData (Metodedata a.s., 1999). It is based on the Entity-Relationship modelling principle. Like ER/studio, Modelator supports both forward and reverse engineering transformations. The forward engineering process is not performed in two steps as in ER/Studio. The transformation is instead done directly from the conceptual model. The tool supports a number of notations e.g. crow's foot and NIAM. Also like ER/Studio Modelator supports a number of databases among others MS SQL server and Oracle 8. The tool doesn’t support SQL-92 but supports a SQL version that is called “standard SQL”. This is the language that will be used as a target language during this phase. Modelator does not support the IS-A relation.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

5.2 Applying the method

Phase 2 will be conducted in three main stages. In the first stage the two tools will be investigated on the different concepts that were created in phase 1. In the second stage one of the tools will be tested against a textbook example.

In the third stage the tools will be examined for features that were not found in the literature.

Phase 2 will therefore test the framework that was created in phase 1. It is also possible that phase 2 will add totally new concepts to those generated during the creation of the literature based framework.

5.2.1 Analysing the tools support for the concepts that were created in phase 1

Information preservation

To find out if the tools could perform transformations that were information preserving I started the investigation of the tools by creating SQL scripts of the models that Teorey (1991) presented as examples of transformations between EER and SQL (appendix A). The examples were to be used in the verification part of the literature based framework The initial transformations (appendix D) showed that the tools did not use foreign key constrain triggers as in the script developed by Teorey (1991). If the user of the CASE-tool wants to add these constrains they must be added manually during the customisation of the forward engineering transformation. The transformation result of the models developed by Teorey (1991) applied to the Modelator tool and with the foreign key constrains customised can be found in appendix E.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

It was not possible to specify what the foreign key constraint triggers were to look like when a relationship were optional in ER/Studio. Because of this I did not add any foreign key constrains triggers to the SQL script that I generated in ER/Studio. The transformation result of the models developed by Teorey (1991) applied to ER/Studio can be found in appendix F.

Both tools had problems representing unary relations. Modelator transferred the foreign key to the table in one to one relations using the primary key’s name on the foreign key (appendix E.8 and E.9). This means that there will be attributes with the same name in the same table.

ER/Studio solved this problem by letting the user add a ‘role name’ to the foreign key when creating a unary relation (appendix F.8 and F.9). Neither of the tools could perform a correct transformation of a unary relation that is many to many optional (appendix E.10 and F.10) Modelator did not make any thing of the relation. ER/Studio did make a new table of the relation but the tool just added the primary key of the entity to the new table. A correct transformation should make a new table of the relation with two ‘versions’ of the primary key from the entity.

The transformation of an IS-A relation was not tested during this part of phase 2. The reason for this was that the Modelator tool did not support the IS-A relation and ER/Studio did not support SQL-92. The literature based framework does show that the transformation of an IS-A relation to SQL-92 cannot be information preserving.

During the transformation of the models presented by Teorey (1991) the author did realise that it is not possible to represent the cardinality of a relation to an entity that doesn’t get a foreign key in SQL 92. This was something that was missed during the development of the literature based framework.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

The customisation of the forward engineering transformation is not done in one single place in Modelator. This makes it quite hard to perform the customisation. ER/Studio has a series of windows where the user is asked a number of questions about how the transformation is to be conducted (appendix F). During the customisation of the forward engineering process it is possible to exclude entities.

Verification of the forward engineering transformation

The two tools have support for diagram analysis. Modelator checks the model for errors before the generation of the SQL code is conducted. The result of the check is shown in an auto-generated report. In ER/Studio it is possible to check the model when ever the user wishes. The user can also let the tool check the model before the physical model is generated. When the final transformation between the physical model and the database/SQL script is done the tool performs a check of the model. The result of a model check is, as in Modelator, shown in a report.

Validation of the forward engineering transformation

There is no single external customer for the transformations performed during this project. It will therefore be impossible to truly evaluate the validation concept. But the tool’s support for report generation can be evaluated.

ER/Studio has the ability to generate a great number of reports (appendix G). It is also possible to generate these reports in HTML format. This is something that makes it possible to quickly publish and share the reports with the help of the Internet.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

The report in ER/Studio that is most interesting for this project is the one that is automatically generated when a forward engineering transformation is conducted. The report shows what the tool did and in which order during the transformation.

The Modelator tool does not have any support for automated report generation apart from the report that is generated during the checking of the models before forward engineering.

Traceability and repository support

Neither of the tools has a repository that is as advanced as that described in appendix C. Modelator does have a feature that is called model explorer (appendix G). The feature creates a tree structure of the model and can be used to make a quick overview of the model. ER/Studio has a more interesting feature from a traceability point of view. The tool creates a tree structure of all the different models (sub models and physical/logical) that are in a database. The tree structures are shown to the left in appendix L. This feature does to some extent support versioning management.

5.2.2 Testing the framework against a textbook example

The transformations of the model examples presented by Teorey (1991) only shows how the tested tools handle the most basic transformations. To see how one of the tools handles the transformation of a more complex conceptual model a textbook example of a small airport database (after Elmasri et al., 1994 p. 659) was used together with ER/Studio. The author does not claim to have made a perfect transformation between the example presented by Elmasri et al., (1994) (the model is in Elmasri EER) and IE. The model only to serves as a source of inspiration. The
Phase 2 A study of support for design transformation in two commercial CASE-tools

model represented in ER/Studio can be found in Appendix H. The transformation of the model into Oracle 8 shows that the transformation of the IS-A relation was not done as Hainaut et al., (1996) proposes with neutralisation of IS-A relations. The IS-A relation was instead replaced with a one to many dependent relation from the super type to the subtype (this was done during the transformation between the logical model and the physical model). The CASE-tool did not have any problem (except the problem with the transformation of the cardinality described in 5.2.1) with the transformation of entities with multiple relations added to them (as an example see the airplane table appendix I). ER/Studio also managed to transform the dependent relation as described in the transformation rules presented by Teorey (1991).

5.2.3 Finding totally new concepts by examining the tools

During this part of phase 2 it was my intention to provide data which could facilitate new concepts that where not established during the creation of the literature based framework (phase 1). During my study of the tools I did find a lot of interesting features but unfortunately it was impossible to connect any of the new features to the forward engineering process.

5.2.4 Analysis of the study

The two tools did not automatically handle the foreign key constraint triggers in the same way as Teorey (1991) (appendix A). Griebel et al., (1996) writes that “Update propagation properties (cascade, restrict, update, set null and set default...) which define the behaviour of the database in case of violations of IC [(Integrity Constrains)] are not included in common EER notations” ( p.3). If the user wants to uses these
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

features he has to add them himself during the customisation of the forward engineering process.

During the study it became obvious that it is impossible to fully represent the cardinality of the EER model in SQL-92. The semantics of the cardinality is not preserved in the SQL schema.

The test of the framework against a textbook example shows that the IS-A relation can be neutralised in other ways than those presented by Hainaut et al., (1996). The transformation of the IS-A relation is still not information preserving, it is replaced with another type of relation.

The test of the framework also shows that the CASE-tool can transform entities with multiple relations correctly and that dependent entities can be transformed.

Neither of the tools could perform the transformation of the unary relation that was many to many, both optional. Modelator did not create a new table of the relation and ER/Studio did not add two versions of the primary key from the entity as primary keys in the table that represents the relation. The transformation of the unary relation that was many to many, both optional, did not fulfil the concept of transformation correctness. The relation was lost in the transformation. This means that how CASE-tools transform unary relations and especially many to many unary relations can vary and that the transformation is not always performed correctly.

In one of the tools it is possible to exclude some entities during the customisation of the forward engineering transformation. This indicates that the customisation of the forward engineering transformation can conflict with the information preservation concept.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

The study of verification of the forward engineering transformation part of the framework showed that both the tools have support for diagram analysis. There are some minor differences in when the analysis is done.

Modelator has no support for report generation apart from the report that is generated when the graphical model is checked. In ER/Studio it is possible to generate a great number of reports. The most interesting report for this framework is the report that is generated during forward engineering.

The two tools have some minor features that can be characterised as repository features. Neither tool has any real support for versioning management as described in appendix C, but ER/Studio has a feature that can be used for manually managing different versions.

The study of the tools did not give any totally new concepts for the framework.

5.2.5 Refining the framework

Teorey (1991) uses foreign key constraint triggers to improve the semantics of SQL-92 code. The two CASE-tools did not automatically add the foreign key constraint triggers. This indicates that how the foreign key constrain triggers are handled in CASE-tools is important to evaluate. My analysis shows that the user of the CASE-tool has to know what the foreign key constraint triggers should look like in the transformation result. This indicates that it is important to consider the user’s knowledge in SQL-92. These two features are closely related to the sub concept customisation of the forward engineering process and therefore should be added to that part of the framework.
5 Phase 2 A study of support for design transformation in two commercial CASE-tools

The analysis shows that both tools had problems transforming unary relations and especially unary relations that are many to many. This shows that it is important to evaluate how the CASE-tool handles unary relations. This refinement is to be seen as a special case of the sub concept transformation correctness because there were missing features after the transformation of the unary relation, something that should not be there if the transformation were correct according to Gogolla (1997).

The transformation rules that were to be used in the verification part of the framework were used during the information preservation test of the transformations. The information preservation concept was therefore moved to the verification of the forward engineering transformation part of the framework.

One of the tools generate a report during the forward engineering transformation. This report shows what happened and in which order during the transformation. Such a report can be used to verify that the transformation was done correctly. This shows that a CASE-tool’s ability to generate reports during the forward engineering transformation is something that is important to evaluate not only during the validation of the transformation but also during the verification of the transformation. Appendix M shows the refined framework.
6 Analysis

6.1 Criteria for the analysis

For this analysis the four Criteria presented by Marshall and Rossman (1999) will be used. The criteria are:

1. **credibility**: in which the goal is to demonstrate that the inquiry was conducted in such a manner as to ensure that the subject was accurately identified and described.

2. **transferability**: In which the researcher must argue that his findings will be useful to others in similar situations, with similar research questions or questions of practise.

3. **dependability**: In which the researcher attempts to account for changing conditions in the phenomenon chosen for the study.

4. **confirmability**: captures the traditional concept of objectivity. Could the findings of the study be confirmed by another?

6.2 Analysing the process of creating an evaluation framework

To show the process of establishing an evaluation framework for database design transformations in CASE-tools using the method presented by Lundell and Lings is the main purpose of this project.

The author of this report claim to have used the method presented by Lundell and Lings during the development of the literature based framework by applying an iterative interaction between the open and selective coding processes. The author did,
6 Analysis

however, feel that it was sometimes difficult not to use a more traditional literature study.

The transferability aspect is maybe the most important aspect of this project. This report shows one way of taking the method developed by Lundell and Lings and using it in a different context. There were some problems with the change of context, between the original purpose for the method (CASE-tool evaluation) and the context of this project (database design transformation). In the original method a number of different sources are to be used for phase one. In this project only one type of source was used in phase one, namely relevant literature. The author of this report tried to handle this transferability problem by trying to clarify the original method with respect to a literature based grounded theory study.

The use of a pilot study to refine the framework was transferred to this context without any major changes. It is possible to say that phase 2 did add to the framework with respect to the criteria:

- Improve precision
- Increase realism
- Expansion of the emerging framework

(Lundell, 1999)

An example of improved precision was that the investigated CASE-tool’s problems with the transformation of unary relations were found during phase 2, something that the literature never indicated. An other example of improved precision was that one of the CASE-tool didn’t translate the IS-A relation in any of the ways that is described in the literature.
6 Analysis

An example of increased realism from phase 2 was that none of the tools evaluated in phase two could automatically generate foreign key constraint triggers as described in the literature.

Totally new concepts were not found during phase 2. However it is possible to say that phase 2 did expand the framework. An example of this was the new sub concept ‘assessment of the users, knowledge in SQL 92’.

The dependability criteria cannot be applied to the process of creating a framework. The question of whether the product of the process (the framework) is dependable is discussed in section 6.3.

The confirmability of the framework is also discussed in section 6.3.

6.3 The final framework

In answer to the question whether the framework can be useful to others in similar situations the author must point out that the focus of this project was on the process of establishing a framework and not on the framework itself. The different concepts that it are to be found in the framework are therefore to be seen as examples of aspects that can be interesting to examine when the forward engineering process is to be evaluated. The refinement that emerged during phase 2, and especially the problems that the CASE-tools had with some transformations, can be of some interest as examples of limitations in two commercial CASE-tools.

The framework’s dependability became higher than it would have been if only literature was used for the creation of the framework, because it has been tested on two commercial CASE-tools. It is important to point out that CASE-tools evolve and that
6 Analysis

the limitations in the tested tools may not be found in future versions of the tools. It is also important to point out that the refinement of this framework was done on ‘merely’ two CASE-tools and the limitations of the tools could be specific to these tools. One important aspect of the method proposed by Lundell and Lings is that the process of establishing the framework can be done in more than one ‘cycle’ through the method. If the user of the framework suspects that the framework has a low dependability it is possible to cycle it through the method again and use, for example, newer CASE-tools for phase 2.

Whether the framework is confirmable is a question that is very hard to answer. It is very important to point out that the framework would be different if it was created by some one other than the author. Whether the framework can be used in a real evaluation of forward engineering is a question that could only be answered if the framework were used by a person that hasn’t been involved in the process of creating the framework. This was beyond the scope of this project.
7 Conclusion and discussion

This report shows how the method proposed by Lundell and Lings (Lundell and Lings, 1998, 1999; Lundell et al., 1998, 1999; Lundell, 1999) can be successfully applied when taken out of its original context and applied to database design transformation.

One important aspect to investigate in this project is whether phase 2 refined the framework, and it is possible to say that phase 2 improved precision, increased realism and expanded the framework. The final framework can be found in appendix M.

One of the main ideas with the method proposed by Lundell and Lings is that the evaluation framework should cover both the organisational needs and what the current technology can offer. This was partly lost because this project was conducted with a silent organisation as a target for the framework.

The author of this report experienced that, taking the method out of its context and applying it in a context where the only source of information for phase 1 was relevant literature, was difficult, because the development of the framework had a tendency towards a more traditional way of using literature, something that may have been avoided if other types of information sources were used. Such sources could include interviews with CASE-tool users and model experts.
References:

Bernstein, Philip A. and Dayal, Umeshwar (1994) An Overview of Repository Technology, In Bocca, Jorge B., Jarke, Matthias and Zaniolo, Carlo (Eds.) Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile, 12-15 September 1994, pp 705-713.


Bringle, Per and Johansson, Claes (1998). CASE-tool evaluation, University of Skövde, Sweden


Date, C. J. and Darwen, H. (1997) A guide to the SQL standard Addison-Wesley Longman, INC, USA


Gogolla, Martin. On behavioural model quality and transformation Proceedings of the ER ’97 Workshop on Behavioral Models and Design Transformations: Issues and
References:

*Opportunities in Conceptual Modelling*, UCLA, Los Angeles, California, 6-7 November 1997


References:


Wynekoop J. and Congner S (1991) A review of computer aided software engineering research methods Information systems research Elsevir Science Publisher B.V(Holland)
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

Table of contents

A.1 Transformation rules .....................................................................................49
A.2 Cardinality representation in IE (After Martin, 1990) ....................................51
A.3 One-to-one, both mandatory..........................................................................51
A.4 One-to-one, one entity optional .................................................................52
A.5 One-to-one, both optional..............................................................................52
A.6 One-to-many, both sides mandatory .............................................................53
A.7 One-to-many, one entity is optional.............................................................53
A.8 Many-to-many, both optional ....................................................................54
A.9 One-to-one, with both optional....................................................................54
A.10 One to many, with the many side optional.................................................55
A.11 Many-to-many with both sides optional....................................................55

A.1 Transformation rules

Teorey (1991) writes that the basic transformation can be described in terms of the three types of tables they produce. These are:

1. An entity relational table with the same information content as the original entity.
   This transformation always occurs for entities with binary relationships that are many to many, one to many on the parent side, or one to one on one side. This sort of table is also used for an entity with any ternary or higher degree relationship, a generalisation hierarchy, or a subset hierarchy.

2. An entity relationship table with the embedded foreign key of the parent entity.
   This transformation always occurs for entities with binary relationships that are one to many for the entity on the many side, for one to one relationships for one of the entities, and for each entity with a unary relationship that is one to one or one to many.

3. A relationship table with the foreign keys of all the entities in the relationship. This transformation always occurs for relationships that are binary and many to many,
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

relationships that are unary and many to many, and all relationships that are of
ternary or higher degree (It is not possible to represent ternary or higher relations
in the IE notation (Martin, 1990).

Weak entities defers form ordinary entities only in their need for the keys of other
entities to establish their uniqueness (Teorey, 1991).

According to Teorey (1991) are the following rules to be applied on the usage of null
values in these transformations:

- Nulls are allowed only for foreign keys of optional entities in an entity table.
- Nulls are not allowed for foreign keys of mandatory entities in an entity table.
- Nulls are not allowed for any keys in a relationship table because only complete
  rows entries are meaningful in the table.

Teorey (1991) summarise tree basic transformation steps for this process. These are:

1) Transform each entity into a table containing the key and nonkey attributes of the
   entity

2) Transform every many-to-many binary ore unary relationship into a relationship
   table with the keys of the entities and the attributes of the relationship.

Transform every ternary or higher-level n-ary relationship into a relationship into a
relationship table.
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

A.2 Cardinality representation in IE (After Martin, 1990)

Each X is associated with one and only one Y:

```
X  Y
```

Each X is associated with zero or one Y:

```
X  Y
```

Each X is associated with one or more Y's:

```
X  Y
```

Each X is associated with Zero one or more Y's:

```
X  Y
```

Each X is associated with more than one Y:

```
X  Y
```

A.3 One-to-one, both mandatory

```
create table X (X_id char(12) not null unique, Y_id char(12) not null unique, primary key (X_id), foreign key (Y_id) references Y on delete set default on update cascade)
create table Y (Y_id char(12) not null unique, primary key (Y_id))
```
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

A.4 One-to-one, one entity optional

create table X (X_id char(12) not null unique,
Y_id char(12) not null unique,
primary key (X_id),
foreign key (Y_id) references Y
on delete cascade
on update cascade)

create table Y (Y_id char(12) not null unique,
primary key (Y_id))

A.5 One-to-one, both optional

create table X (X_id char(12) not null unique,
Y_id char(12) unique,
primary key (X_id),
foreign key (Y_id) references Y
on delete set null
on update cascade)

create table Y (Y_id char(12) not null unique,
X_id char(12) unique,
primary key (Y_id),
foreign key (X_id) references X
on delete set null
on update cascade)
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

A.6 One-to-many, both sides mandatory

create table X (X_id char(12) not null unique, primary key (X_id))

create table Y (Y_id char(12) not null unique, X_id char(12) not null, primary key (Y_id), foreign key (X_id) references X on delete set cascade on update cascade)

A.7 One-to-many, one entity is optional

create table X (X_id char(12) not null unique, primary key (X_id))

create table Y (Y_id char(12) not null unique, X_id char(12) not null, primary key (Y_id), foreign key (X_id) references X on delete set default on update cascade)
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

A.8 Many-to-many, both optional

A.9 One-to-one, with both optional
Appendix A Example of transformation between IE and SQL (after Teorey, 1991)

A.10 One to many, with the many side optional

create table X (X_id char(12) not null unique, X_a char(12) not null, primary key (X_id), foreign key (X_a) references X on delete set default on update cascade)

A.11 Many-to-many with both sides optional

create table X (X_id char(12) not null unique, primary key (X_id))
create table X2 (X_id char(12) not null, X_a char(12) not null, primary key (X_id, X_a), foreign key (X_id) references X on delete set cascade on update cascade, foreign key (X_a) references X on delete set cascade on update cascade)
Appendix B Neutral transformations to eliminate IS-A relations (after Hainaut et al., 1996)

In this appendix will the different basic techniques for the elimination of the IS-A relation be presented after Hainaut et al., (1996).

Table of contents

B.1 Techniques for is-a neutralisation ..................................................................56
B.2 IS-A materialisation ......................................................................................56
2.3 Upward inheritance........................................................................................57
B.4 Downward inheritance...................................................................................57

B.1 Techniques for is-a neutralisation

The three techniques for IS-A neutralisation are:

- **IS-A materialisation**, each subtype is represented by an independent entity type related to the supertype by a one-to-one relationship type.

- **upward inheritance**, the IS-A relation is represented by the supertype only.

- **downward inheritance**, that represent the subtypes only the supertype collects the entities that do not belong to subtypes B and C, if any:

B.2 IS-A materialisation

![Diagram of IS-A materialisation](image)
Appendix B Neutral transformations to eliminate IS-A relations (after Hainaut et al., 1996)

2.3 Upward inheritance

B.4 Downward inheritance
Appendix C Repository and versioning control

In Appendix C will the concepts repository and versioning control be described (after Bernstein et al., 1994).

According to Bernstein et al (1994) is “A repository is a shared database of information about engineered artefacts produced or used by an enterprise. Examples of the artefacts include software, documents, maps, discrete manufactured components and systems and information systems.” Bernstein et al (1994). One attribute that Bernstein et al (1994) claims that an repository most have is version control. “A version is a semantically meaningful snapshot of an object at some point in its lifecycle.” Bernstein et al (1994). The repository maintains each object’s version history, which is a directed graph with one node per version and an edge from version A to version B is B was derived from A. Features of a version model should include:

- A mechanism for representing versions as objects in the repository.
- A version naming mechanism, preferably both with default and user-supplied names.
- An operation for deriving a new version from an old version.
- Constraints on the version history.
- A mechanism for identifying a particular version to be used in a given context.
- The semantics of checkout and check-in of versioned objects.

An operation for declaring that two or more independently developed versions merge into one version.
Appendix D Initial transformation in Modelator

Table of contents

D.1 One-to-one, both mandatory ................................................................. 59
D.2 One-to-one, both optional ................................................................. 60

D.1 One-to-one, both mandatory

// SCRIPT FOR Standard SQL
// Generated from Modelator 1999-06-24 11:29:49 Model :
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (  
    X_id INTEGER NOT NULL,
    X1 VARCHAR(12),
    PRIMARY KEY (  
        X_id
    )
);
CREATE TABLE Entity_Y (  
    Y_id INTEGER NOT NULL,
    Y1 VARCHAR(12),
    X_id INTEGER NOT NULL,
    PRIMARY KEY (  
        Y_id
    )
);
// FOREIGN KEYS
ALTER TABLE Entity_Y
ADD
FOREIGN KEY (  
    X_id
)
REFERENCES Entity_X (  
    X_id
);

// INDEXES
// INDEXES ON FOREIGN KEYS
ALTER TABLE Entity_Y
ADD UNIQUE (  
    X_id
);
D.2 One-to-one, both optional

```
// SCRIPT FOR STANDARD SQL
// Generated from Modelator 1999-06-24 11:28:41 Model :
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (  
  X_id INTEGER NOT NULL,
  X1 VARCHAR(12),
  PRIMARY KEY (  
    X_id
  )
);
CREATE TABLE Entity_Y (  
  Y_id INTEGER NOT NULL,
  Y1 VARCHAR(12),
  X_id INTEGER,
  PRIMARY KEY (  
    Y_id
  )
);
// FOREIGN KEYS
ALTER TABLE Entity_Y  
ADD  
FOREIGN KEY (  
  X_id
)  
REFERENCES Entity_X (  
  X_id
);
// INDEXES
// INDEXES ON FOREIGN KEYS
ALTER TABLE Entity_Y  
ADD UNIQUE (  
  X_id
);
```
Table of contents

E.1 One Entity ............................................................................................................. 61
E.2 One-to-one, both mandatory .............................................................................. 62
E.3 One-to-one, one entity optional......................................................................... 63
E.4 One-to-one, both optional .................................................................................. 64
E.5 One-to-many, both sides mandatory .................................................................... 65
E.6 One-to-many, one entity is optional .................................................................... 66
E.7 Many-to-many, both optional ............................................................................ 67
E.8 One-to-one, with both optional .......................................................................... 68
E.9 One to many, with the many side optional ......................................................... 69
E.10 Many-to-many with both sides optional ........................................................... 70

E.1 One Entity

// SCRIPT FOR Standard SQL
// Generated from Modelator 1999-06-22 09:14:44 Model:
// DROP
DROP TABLE Entity_X;
// TABLES
CREATE TABLE Entity_X (X_id INTEGER NOT NULL,
X1 VARCHAR(12),
PRIMARY KEY (X_id)
);

SELECT * FROM Entity_X;
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.2 One-to-one, both mandatory

```
// S C R I P T F O R S t a n d a r d S Q L
// Generated from Modelator 1999-06-22 09:29:12 Model :
// D R O P
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// T A B L E S
CREATE TABLE Entity_X (  
  X_id INTEGER NOT NULL,  
  X1 VARCHAR(12),  
  PRIMARY KEY (  
    X_id  
  )
);
CREATE TABLE Entity_Y (  
  Y_id INTEGER NOT NULL,  
  Y1 VARCHAR(12),  
  X_id INTEGER NOT NULL,  
  PRIMARY KEY (  
    Y_id  
  )
);
// F O R E I G N K E Y S
ALTER TABLE Entity_Y
  ADD
  FOREIGN KEY (  
    X_id  
  )
  REFERENCES Entity_X (  
    X_id  
  )
  ON DELETE SET NULL
  ON UPDATE CASCADE;
// I N D E X E S
// I N D E X E S O N F O R E I G N K E Y S
ALTER TABLE Entity_Y
  ADD UNIQUE (  
    X_id  
  )
);
```
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.3 One-to-one, one entity optional

```
// SCRIPT FOR STANDARD SQL
// Generated from Modelator 1999-06-22 09:47:03 Model:
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (
    X_id INTEGER NOT NULL,
    X1 VARCHAR(12),
    Y_id INTEGER,
    PRIMARY KEY (X_id)
);
CREATE TABLE Entity_Y (
    Y_id INTEGER NOT NULL,
    Y1 VARCHAR(12),
    PRIMARY KEY (Y_id)
);
// FOREIGN KEYS
ALTER TABLE Entity_X
    ADD FOREIGN KEY (Y_id)
    REFERENCES Entity_Y (Y_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
// INDEXES
// INDEXES ON FOREIGN KEYS
ALTER TABLE Entity_X
    ADD UNIQUE (Y_id);
```
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.4 One-to-one, both optional

![Diagram of Entity_X and Entity_Y with a one-to-one relationship]

// SCRIPT FOR Standard SQL
// Generated from Modelator 1999-06-22 09:56:26 Model :
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (  
    X_id INTEGER NOT NULL,  
    X1 VARCHAR(12),  
    Y_id INTEGER,  
    PRIMARY KEY (  
        X_id  
    )  
);

CREATE TABLE Entity_Y (  
    Y_id INTEGER NOT NULL,  
    Y1 VARCHAR(12),  
    PRIMARY KEY (  
        Y_id  
    )  
);

// FOREIGN KEYS
ALTER TABLE Entity_X
ADD
FOREIGN KEY (  
    Y_id  
)  
REFERENCES Entity_Y (  
    Y_id  
)  
ON DELETE SET NULL  
ON UPDATE CASCADE;

// INDEXES
// INDEXES ON FOREIGN KEYS
ALTER TABLE Entity_X
ADD UNIQUE (  
    Y_id  
) ;
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.5 One-to-many, both sides mandatory

```sql
// SCRIPT FOR STANDARD SQL
// Generated from Modelator 1999-06-22 09:59:54 Model:
// DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (
    X_id INTEGER NOT NULL,
    X1 VARCHAR(12),
    PRIMARY KEY (X_id)
);
CREATE TABLE Entity_Y (
    Y_id INTEGER NOT NULL,
    Y1 VARCHAR(12),
    X_id INTEGER,
    PRIMARY KEY (Y_id)
);
// FOREIGN KEYS
ALTER TABLE Entity_Y
    ADD FOREIGN KEY (X_id)
    REFERENCES Entity_X (X_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
// INDEXES
CREATE INDEX FK1_Entity_X_Entity_Y ON Entity_Y (X_id);
```
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.6 One-to-many, one entity is optional

```
// Generated from Modelator 1999-06-22 10:01:39 Model :
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (  
    X_id INTEGER NOT NULL,  
    X1 VARCHAR(12),  
    PRIMARY KEY (    
        X_id    
    )  
);
CREATE TABLE Entity_Y (  
    Y_id INTEGER NOT NULL,  
    Y1 VARCHAR(12),  
    X_id INTEGER,  
    PRIMARY KEY (    
        Y_id    
    )  
);
// FOREIGN KEYS
ALTER TABLE Entity_Y
ADD
FOREIGN KEY (    
    X_id    
)
REFERENCES Entity_X (    
    X_id    
)
ON DELETE SET NULL
ON UPDATE CASCADE;
// INDEXES
// INDEXES ON FOREIGN KEYS
CREATE INDEX FK1_Entity_X_Entity_Y ON Entity_Y (    
    X_id    
);
```
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.7 Many-to-many, both optional

// Script for Standard SQL
// Generated from Modelator 1999-06-22 10:05:00 Model:
// DROP
DROP TABLE Entity_X;
DROP TABLE Entity_Y;
// TABLES
CREATE TABLE Entity_X (  
    X_id INTEGER NOT NULL,  
    X1 VARCHAR(12),  
    PRIMARY KEY (  
        X_id  
    )  
);
CREATE TABLE Entity_Y (  
    Y_id INTEGER NOT NULL,  
    Y1 VARCHAR(12),  
    PRIMARY KEY (  
        Y_id  
    )  
);
// FOREIGN KEYS
// INDEXES
// INDEXES ON FOREIGN KEYS
CREATE INDEX FK1_Entity_X_Entity_Y ON Entity_Y (  
);
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.8 One-to-one, with both optional

![Diagram of Entity_X with X_id (Pk) and X1]

```
// SCRIPT FOR Standard SQL
// Generated from Modelator 1999-06-23 13:29:34 Model :
// DROP
DROP TABLE Entity_X;
// TABLES
CREATE TABLE Entity_X
    (X_id INTEGER NOT NULL,
     X1 VARCHAR(12),
     PRIMARY KEY (X_id)
    );
// FOREIGN KEYS
ALTER TABLE Entity_X
    ADD
    FOREIGN KEY (X_id)
    REFERENCES Entity_X (X_id)
    ON DELETE SET NULL
    ON UPDATE CASCADE;
// INDEXES
// INDEXES ON FOREIGN KEYS
ALTER TABLE Entity_X
    ADD UNIQUE (X_id);
```
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.9 One to many, with the many side optional

```sql
DROP TABLE Entity_X;
CREATE TABLE Entity_X (
    X_id INTEGER NOT NULL,
    X1 VARCHAR(12),
    X_id INTEGER,
    PRIMARY KEY (X_id)
);
ALTER TABLE Entity_X
    ADD
    FOREIGN KEY (X_id)
    REFERENCES Entity_X (X_id)
    ON DELETE SET NULL
    ON UPDATE CASCADE;
ALTER TABLE Entity_X
    ADD UNIQUE (X_id);
```

---

X_id (Pk)
X1
Entity_X
Appendix E Test of the transformation example presented by Theory(1991) in Modelator

E.10 Many-to-many with both sides optional

// SCRIPT FOR Standard SQL
// Generated from Modelator 1999-06-22 10:19:36 Model:
// DROP
DROP TABLE Entity_X;
// TABLES
CREATE TABLE Entity_X (  
    X_id INTEGER,  
    X1 CHAR(12)  
);
// FOREIGN KEYS
// INDEXES
// INDEXES ON FOREIGN KEYS
CREATE INDEX FK1_Entity_X_Entity_X ON Entity_X (  
) ;
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

Table of contents

F.1 One Entity .....................................................................................................71
F.2 One-to-one, both mandatory .................................................................72
F.3 One-to-one, one entity optional.............................................................73
F.4 One-to-one, both optional .................................................................74
F.5 One-to-many, both sides mandatory .......................................................75
F.6 One-to-many, one entity is optional ......................................................76
F.7 Many-to-many, both optional ...............................................................77
F.8 One-to-one, with both optional ............................................................78
F.9 One to many, with the many side optional ............................................79
F.10 Many-to-many with both sides optional 80

F.1 One Entity

<table>
<thead>
<tr>
<th>Entity X</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_id (Pk)</td>
</tr>
<tr>
<td>X1</td>
</tr>
</tbody>
</table>

--
-- ER/Studio 3.0 SQL Code Generation
-- Company: His
-- Project: DATA MODEL
-- Author: Claes Johansson
-- Date Created: Saturday, June 26 1999 at 12:59 PM
-- Target DBMS : Oracle 8
--
-- TABLE: "Entity X"
--CREATE TABLE "Entity X"
  X_id   NUMBER(10, 0) NOT NULL,
  X1     CHAR(12),
CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
);
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.2 One-to-one, both mandatory

```
-- ER/Studio 3.0 SQL Code Generation
-- Company: His
-- Project: DATA MODEL
-- Author: Claes Johansson
-- Date Created: Saturday, June 26 1999 at 01:03 PM
-- Target DBMS : Oracle 8

-- TABLE: "Entity Y"
CREATE TABLE "Entity Y"
    (Y_id NUMBER(10, 0) NOT NULL,
     X_id NUMBER(10, 0) NOT NULL,
     Y1 CHAR(12),
     CONSTRAINT PKEntity_Y PRIMARY KEY (Y_id),
     CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
       REFERENCES "Entity X"(X_id)
    )
;
-- TABLE: "Entity X"
--CREATE TABLE "Entity X"
    (X_id NUMBER(10, 0) NOT NULL,
     X1 CHAR(12),
     CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
    )
;
```

```
Entity X
| X_id (Pk) | X1    |
Entity Y
| Y_id (Pk) | Y1    |
```
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.3 One-to-one, one entity optional

```
CREATE TABLE "Entity Y"
(
    Y_id    NUMBER(10, 0)    NOT NULL,
    X_id    NUMBER(10, 0),
    Y1      CHAR(12),
    CONSTRAINT PKEntitty_Y PRIMARY KEY (Y_id),
    CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
            REFERENCES "Entity X"(X_id)
) ;

CREATE TABLE "Entity X"
(
    X_id    NUMBER(10, 0)    NOT NULL,
    X1      CHAR(12),
    CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
) ;
```

```sql
-- ER/Studio 3.0 SQL Code Generation
-- Company:      His
-- Project:      DATA MODEL
-- Author:       Claes Johansson-- Date Created: Saturday, June 26 1999 at 01:06 PM
-- Target DBMS : Oracle 8

-- TABLE: "Entity Y"
CREATE TABLE "Entity Y"(
    Y_id    NUMBER(10, 0)    NOT NULL,
    X_id    NUMBER(10, 0),
    Y1      CHAR(12),
    CONSTRAINT PKEntitty_Y PRIMARY KEY (Y_id),
    CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
            REFERENCES "Entity X"(X_id)
) ;

-- TABLE: "Entity X"
CREATE TABLE "Entity X"(
    X_id    NUMBER(10, 0)    NOT NULL,
    X1      CHAR(12),
    CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
) ;
```
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.4 One-to-one, both optional

---
-- ER/Studio 3.0 SQL Code Generation
-- Company: His
-- Project: DATA MODEL
-- Author: Claes Johansson
-- Date Created: Saturday, June 26 1999 at 01:09 PM
-- Target DBMS: Oracle 8
---
-- TABLE: "Entity Y"
CREATE TABLE "Entity Y"
  (Y_id    NUMBER(10, 0) NOT NULL,
   X_id    NUMBER(10, 0),
   Y1      CHAR(12),
   CONSTRAINT PKEntity_Y PRIMARY KEY (Y_id),
   CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
   REFERENCES "Entity X"(X_id)
  );

-- TABLE: "Entity X"
CREATE TABLE "Entity X"
  (X_id    NUMBER(10, 0) NOT NULL,
   X1      CHAR(12),
   CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
  );
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.5 One-to-many, both sides mandatory

CREATE TABLE "Entity Y"(
    Y_id    NUMBER(10, 0)    NOT NULL,
    X_id    NUMBER(10, 0)    NOT NULL,
    Y1      CHAR(12),
    CONSTRAINT PKEntity_Y PRIMARY KEY (Y_id),
    CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
        REFERENCES "Entity X"(X_id)
)
);

CREATE TABLE "Entity X"(
    X_id    NUMBER(10, 0)    NOT NULL,
    X1      CHAR(12),
    CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
)
;
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.6 One-to-many, one entity is optional

CREATE TABLE "Entity Y"
(  Y_id    NUMBER(10, 0)    NOT NULL,
  X_id    NUMBER(10, 0)    NOT NULL,
  Y1      CHAR(12),
  CONSTRAINT PKEntity_Y PRIMARY KEY (Y_id),
  CONSTRAINT RefEntity_X2 FOREIGN KEY (X_id)
    REFERENCES "Entity X"(X_id)
);

CREATE TABLE "Entity X"
(  X_id    NUMBER(10, 0)    NOT NULL,
  X1      CHAR(12),
  CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
);
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.7 Many-to-many, both optional

```
-- TABLE: "Entity Y"
CREATE TABLE "Entity Y"
  (Y_id  NUMBER(10, 0)  NOT NULL,
   Y1    CHAR(12),
   CONSTRAINT PKEntity_Y PRIMARY KEY (Y_id)
) ;

-- TABLE: "Entity X"
CREATE TABLE "Entity X"
  (X_id  NUMBER(10, 0)  NOT NULL,
   X1    CHAR(12),
   CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
) ;

-- TABLE: "Entity X Entity Y"
CREATE TABLE "Entity X Entity Y"
  (X_id  NUMBER(10, 0)  NOT NULL,
   Y_id  NUMBER(10, 0)  NOT NULL,
   CONSTRAINT PKEntity_X_Entity_Y PRIMARY KEY (X_id,Y_id),
   CONSTRAINT RefEntity_X7 FOREIGN KEY (X_id)
    REFERENCES "Entity X"(X_id),
   CONSTRAINT RefEntity_Y8 FOREIGN KEY (Y_id)
    REFERENCES "Entity Y"(Y_id)
  ) ;
```
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.8 One-to-one, with both optional

```
CREATE TABLE "Entity X"
(
  X_id     NUMBER(10, 0)    NOT NULL,
  X_id2    NUMBER(10, 0),
  X1       CHAR(12),
  CONSTRAINT PKEntity_X PRIMARY KEY (X_id),
  CONSTRAINT RefEntity_X9 FOREIGN KEY (X_id2)
    REFERENCES "Entity X"(X_id)
) ;
```
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.9 One to many, with the many side optional

CREATE TABLE "Entity X"
    (X_id     NUMBER(10, 0)    NOT NULL,
     X_id2    NUMBER(10, 0)    NOT NULL,
     X1       CHAR(12),
     CONSTRAINT PKEntity_X PRIMARY KEY (X_id),
     CONSTRAINT RefEntity_X9 FOREIGN KEY (X_id2)
                      REFERENCES "Entity X"(X_id)
    );
Appendix F Test of the transformation example presented by Theory(1991) in ER/studio

F.10 Many-to-many with both sides optional

```
--
-- ER/Studio 3.0 SQL Code Generation
-- Company:     His
-- Project:     DATA MODEL
-- Author:      Claes Johansson
-- Date Created: Saturday, June 26 1999 at 01:35 PM
-- Target DBMS : Oracle 8
--
-- TABLE: "Entity X"
CREATE TABLE "Entity X"
(  X_id    NUMBER(10, 0)    NOT NULL,
   X1      CHAR(12),
   CONSTRAINT PKEntity_X PRIMARY KEY (X_id)
)
;

-- TABLE: "Entity X Entity X"
CREATE TABLE "Entity X Entity X"
(  X_id    NUMBER(10, 0)    NOT NULL,
   CONSTRAINT PKEntity_X_Entity_X PRIMARY KEY (X_id),
   CONSTRAINT RefEntity_X18 FOREIGN KEY (X_id)
      REFERENCES "Entity X"(X_id),
   CONSTRAINT RefEntity_X19 FOREIGN KEY (X_id)
      REFERENCES "Entity X"(X_id)
)
;
```
Appendix G Customisation of the forward engineering process in ER/Studio

Appendix G shows the series of windows that are shown to the user during the customisation of the forward engineering process.
Appendix G Customisation of the forward engineering process in ER/Studio
Appendix G Customisation of the forward engineering process in ER/Studio
Appendix H Small Airport database (after Elmasri et al., 1994)
Appendix I Small Airport database SQL code

-- ER/Studio 3.0 SQL Code Generation
-- Company: HIS
-- Project: DATA MODEL
-- Author: Claes Johansson
-- Date Created: Wednesday, July 07 1999 at 01:42 PM
-- Target DBMS: Oracle 8

-- TABLE: Airplane
CREATE TABLE Airplane(
    REG#      CHAR(10)         NOT NULL,
    Model     CHAR(12)         NOT NULL,
    Number    NUMBER(10, 0),
    CONSTRAINT PK2 PRIMARY KEY (REG#),
    CONSTRAINT RefHangar15 FOREIGN KEY (Number) REFERENCES Hangar(Number),
    CONSTRAINT RefPlane_Type32 FOREIGN KEY (Model) REFERENCES Plane_Type(Model)
);

-- TABLE: "Airplane Corporation"
CREATE TABLE "Airplane Corporation"(
    REG#    CHAR(10)    NOT NULL,
    Name    CHAR(10)    NOT NULL,
    CONSTRAINT PK16 PRIMARY KEY (REG#, Name),
    CONSTRAINT RefAirplane27 FOREIGN KEY (REG#) REFERENCES Airplane(REG#),
    CONSTRAINT RefCorporation28 FOREIGN KEY (Name) REFERENCES Corporation(Name)
);

-- TABLE: "Airplane Person"
CREATE TABLE "Airplane Person"(
    REG#    CHAR(10)    NOT NULL,
    Ssn     CHAR(10)    NOT NULL,
    CONSTRAINT PK17 PRIMARY KEY (REG#, Ssn),
    CONSTRAINT RefAirplane30 FOREIGN KEY (REG#) REFERENCES Airplane(REG#),
    CONSTRAINT RefPerson31 FOREIGN KEY (Ssn) REFERENCES Person(Ssn)
);

-- TABLE: Corporation
CREATE TABLE Corporation(
    Name       CHAR(10)    NOT NULL,
    Phone      CHAR(10),
    Address    CHAR(10),
    CONSTRAINT PK8 PRIMARY KEY (Name)
);

-- TABLE: Employee
CREATE TABLE Employee(
    Ssn       CHAR(10)            NOT NULL,
    Salary    DOUBLE PRECISION,
    Shift     CHAR(10),
    CONSTRAINT PK4 PRIMARY KEY (Ssn),
    CONSTRAINT RefPerson35 FOREIGN KEY (Ssn) REFERENCES Person(Ssn)
REFERENCES Person(Ssn)
);
;
-- TABLE: Hangar
CREATE TABLE Hangar(
  Number      NUMBER(10, 0)    NOT NULL,
  Capacity    NUMBER(10, 0),
  Location    CHAR(10),
  CONSTRAINT PK3 PRIMARY KEY (Number)
);
;
-- TABLE: Person
CREATE TABLE Person(
  Ssn       CHAR(10)    NOT NULL,
  Adress    CHAR(10),
  Phone     CHAR(10),
  Name      CHAR(10),
  CONSTRAINT PK6 PRIMARY KEY (Ssn)
);
;
-- TABLE: Pilot
CREATE TABLE Pilot(
  Ssn       CHAR(10)    NOT NULL,
  LicNum    CHAR(10),
  Restr     CHAR(10),
  CONSTRAINT PK7 PRIMARY KEY (Ssn),
  CONSTRAINT RefPerson36 FOREIGN KEY (Ssn)
REFERENCES Person(Ssn)
);
;
-- TABLE: Plane_Type
CREATE TABLE Plane_Type(
  Model       CHAR(12)         NOT NULL,
  Weight      NUMBER(10, 0),
  Capacity    NUMBER(10, 0),
  CONSTRAINT PK1 PRIMARY KEY (Model)
);
;
-- TABLE: "Plane_Type Employee"
CREATE TABLE "Plane_Type Employee"(
  Model    CHAR(12)    NOT NULL,
  Ssn      CHAR(10)    NOT NULL,
  CONSTRAINT PK9 PRIMARY KEY (Model,Ssn),
  CONSTRAINT RefPlane_Type2 FOREIGN KEY (Model)
REFERENCES Plane_Type(Model),
  CONSTRAINT RefEmployee3 FOREIGN KEY (Ssn)
REFERENCES Employee(Ssn)
);
;
-- TABLE: "Plane_Type Pilot"
CREATE TABLE "Plane_Type Pilot"(
  Model    CHAR(12)    NOT NULL,
  Ssn      CHAR(10)    NOT NULL,
  CONSTRAINT PK10 PRIMARY KEY (Model,Ssn),
  CONSTRAINT RefPlane_Type5 FOREIGN KEY (Model)
REFERENCES Plane_Type(Model),
  CONSTRAINT RefPilot6 FOREIGN KEY (Ssn)
REFERENCES Pilot(Ssn)
);
-- TABLE: Service
CREATE TABLE Service(
    Date DATE    NOT NULL,
    REG# CHAR(10) NOT NULL,
    Hours NUMBER(10, 0),
    CONSTRAINT PK5 PRIMARY KEY (Date,REG#),
    CONSTRAINT RefAirplane14 FOREIGN KEY (REG#)
        REFERENCES Airplane(REG#)
)
;

-- TABLE: "Service Employee"
CREATE TABLE "Service Employee"(
    Date DATE    NOT NULL,
    REG# CHAR(10) NOT NULL,
    Ssn  CHAR(10) NOT NULL,
    CONSTRAINT PK11 PRIMARY KEY (Date,REG#,Ssn),
    CONSTRAINT RefService8 FOREIGN KEY (Date,REG#)
        REFERENCES Service(Date,REG#),
    CONSTRAINT RefEmployee9 FOREIGN KEY (Ssn)
        REFERENCES Employee(Ssn)
)
;
Appendix J Report generation in ER/Studio

Appendix J Report generation in ER/Studio
Appendix H shows the different reports that are possible to generate in ER/Studio.
Appendix K Repository support in Modelator

[Diagram of a database model showing entities and relationships]
Appendix L Repository support in ER/Studio
Appendix M Relations between concepts and sub concepts in the refined framework

Refined framework

**Concepts:** Verification of the forward engineering process

**Sub concepts:** Automated report generation, Diagram correctness, Information preservation, Transformation rules

**Traceability**

Repository support in CASE tools

**Validation of the forward engineering process**

Version management, Automated report generation

**Information preservation**

Semantic preservation, Information preservation, Transformation correctness, Customisation of the forward engineering process.

The semantics of the IS-A relation and the cardinality cannot be fully preserved in a transformation

The transformation of the unary relation

The users knowledge in SQL-92

The ability to ad foreign key constrains triggers
Appendix N List of quotation in this report (in chronological order)


N2 “[T]ransformation of some logical model into the formalism of the ER model.” (Fahrner and Vossen, 1995 p. 214)

N3 “[T]o produce a conceptual description of a given database where the input may consist of any combinations of source code description, a data dictionary, a database instance, and application programs.” (Souto, 1998)

N4 “Transformations of this kind [between two logical models] seems to be of increasing practise importance,” (Fahrner and Vossen, 1995 p. 239)

N5 “a software environment, that assists a systems analyst and designer in the process of designing, specifying, analyzing, and maintaining a software product (an information system)” (Bubenko and Wangler, 1992).

N6 “CASE technologies are expected to provide task related support for software developers in analysing, designing and implementing a set of information systems (IS) or their components according to a method.” (Kelly et al., 1996 p. 1)

N7 “[P]ure descriptions of various sorts with virtually no or minimal conceptualizations,”. Glaser (1992)

N8 “same information capability as the target model. ” (Fahrner and Vossen, 1995 p. 219)

N9 “[T]he next conceptual model has the relevant properties the privies model required” (Gogolla, 1997).

N10 “The Verification and validation step has the objective of checking weather the formal conceptual specification is consistent, and weather it correctly express the functional requirements, by the users.” (Bubenko and Wangler, 1992 p. 3)

N11 “Verification referees to the set of activities that ensure that software correctly implements a specific function.”. (Pressman, 1997)
Appendix N List of quotation in this report (in chronological order)

N12 “Validation referees to a different set of activities that ensure that that the software that has been built is traceable to customer requirements.” (Pressman, 1997).

N13 "attributes relating to its ability to support the analysis of graphical figures input to the CASE-tool and extracting and storing requirements and/or design information.” (ISO/IEC,1995 p. 27).

N14 “Some EER IC [(Integrity Constrain)] properties (e.g. disjoint relationships) cannot be expressed in the abstract relational layer, but can be modeled by trigger sets of a target database system.” (Griebel et al., 1996 p. 3)

N15 “Validation can be done by paraphrasing a conceptual schema in natural language and giving that paraphrase to a user for examination.” (Griebel et al., 1996)

N16 ”attributes relating to its ability to automate the development of reports to be produced by the system under development” (ISO/IEC, 1995 p. 29).

N17 “Within the border context of IS development it requires correlation of design decision across different sub-models, and between data sets and meta data, on a ongoing basis.” (Griebel et al., 1996 p. 1).

N18 “[A]ttributes relating to its ability to perform traceability analysis.” ISO/IEC, 1995 p. 33)

N19 “Update propagation properties (cascade, restrict, update, set null and set default...) which define the behaviour of the database in case of violations of IC [(Integrity Constrains)] are not included in common EER notations.” (Griebel et al., 1996 p. 3)