RIDL*

de Troyer, O.M.F.

Publication date:
1989

Document Version
Publisher's PDF, also known as Version of record

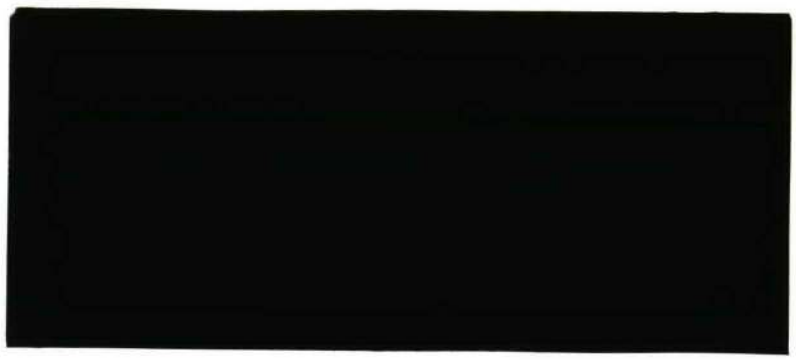
[redacted]

ITK

RESEARCH
REPORT

INSTITUTE FOR LANGUAGE TECHNOLOGY AND ARTIFICIAL INTELLIGENCE

INSTITUUT VOOR TAAL- EN KENNISTECHNOLOGIE



ITK Research Report No. 3

January 1989

RIDL*: A Tool for the Computer-Assisted Engineering of Large Databases in the Presence of Integrity Constraints.

O.M.F. De Troyer

Institute for Language Technology and Artificial Intelligence ITK

Tilburg University, Tilburg, the Netherlands

RIDL*: A Tool for the Computer-Assisted Engineering of Large Databases in the Presence of Integrity Constraints.

ABSTRACT

When designing large databases, tools and methods that transform higher level formalisms into logical database designs become very important. Rarely if ever do these transformations take into account integrity constraints existing in the "conceptual" model. Yet these become essential if one is forced to introduce redundancies for reasons of e.g. query efficiency: we then need constraint specifications to describe these redundancies. We therefore adopted a model (Binary Relationship Model or "NIAM") that is rich in constraints and built a fexible tool, RIDL*, that graphically captures NIAM semantic networks, analyzes them and then transforms them into relational designs (normalized or not) in SQL, under the control of a database engineer assisted by a rule base. This is made possible by a rule-driven implementation of a new, stepwise synthesis process, and its benefits are illustrated by its treatment of e.g. subtypes. RIDL* is operational at several industrial sites in Europe and the U.S. on sizeable database projects.

To be published in the proceedings of the ACM-SIGMOD "International Conference on Management of Data" , Portland, Oregon 1989.

CONTENTS.

1. Introduction

Developing a large relational-based system is a non-trivial and complex project; it has many phases and requires the participation of many different people: users, analysts, database administrators, programmers. Since most large systems have very long lifetimes, the decisions that have to be taken may have long-lasting effects and far-reaching consequences, and must be well-documented.

Methodologies, or combinations of them, well-supported by software tools thus become rather a necessity when dealing with large systems. Moreover, in these cases it is essential that such a methodology be based on a formal specification mechanism that allows extensive ***integrity rule definition*** and ***partial design prototyping***, so that at early stages (partial) specifications of the system can already be checked for correctness and consistency in the large. Also, the abstraction ability of the formalism is paramount in determining its usability as specification vehicle. In particular, object oriented formalisms serve this purpose very well. They allow for several types of abstractions like abstraction from instance level, aggregation and most importantly generalization through an inheritance and override mechanism.

When concentrating on database design (there is also the issue of application design), we therefore need a language which is more conceptual in nature, has more abstraction capabilities than the relational model, and is richer in the area of integrity constraints. The relational model in its present implementations nearly completely ignores the semantics of the data, and consequently this semantics must then in principle be hand-crafted into the database's application programs. Usually this happens in the form of ad-hoc constraint

checks, transaction procedures and other hacker's delights. Database design often starts out too early from "rough", application-inspired aggregations of data which then must be made into more correct ones using decomposition (normalization) rules, a technique that still leaves a lot to the intuition of the database engineer.

To be able to concentrate more on the semantics of the information rather than merely on its aggregation structure we need a formalism which should at least be free from any fore-ordened data organisation, allow to abstract from any representation of individuals and allow to express a semantics which is much richer than the trivial semantics expressable in the relational model.

As specification formalism we have adopted here the Binary Relationship Model (BRM) (1974). It is somewhat related to the Functional Data Model [12] with which it shares the ability to support powerful and elegant functional query and data manipulation languages [6],[4],[25], [10]. In the BRM all relationships are expressed between exactly two object types either as "fact types" or as "sublink types". The meta-concept object type allows to make the abstraction from individuals or occurrences while the sublink type supports the inheritance mechanism also known from object oriented languages such as e.g. Smalltalk. Fact types are used to express aggregation, general relationships etc.. Additionally, the BRM explicitly addresses the issue of constraints. For a comprehensive comparison of the BRM (or NIAM, see below) to Entity-Relationship modelling from the viewpoint of semantics, we refer to [23]. The name "NIAM" (Nijssen Information Analysis Method [25],[18],[26]) refers to the notation most often associated with the BRM; we give a condensed description of it in section 2.

In this paper we describe a software workbench, named RIDL* and in particularly its engineering method. The workbench is based on the BRM and supports the activities related to the specification, design and generation of large and complex (relational) databases. The relational designs are generated from the conceptual database design (expressed in the BRM) using ***synthesis***.
The synthesis method for constructing a relational database from a (binary) conceptual design becomes quite complex, at least if one wants to provide the flexibility needed for the design of large database system. In this paper we concentrate on this important aspect of the RIDL* system. A short overview of the RIDL* system is given in section 3. (A much more extensive description, however not including the aspects covered in this paper, can be found in [7]). Snapshots of RIDL*'s operation are given, taken from the screen of an Apollo workstation on which the software is currently available. The overall example is a well-known hypothetical database system supporting the organization of conferences taken from the literature and known as the "CRIS-case" [20].

2. The Binary Relationship Model (BRM)

Descriptions of the BRM (under different names) appear in several forms in the literature ([25], [5], [10], [15], [18], [26], ...).

Its main characteristics are:
a. objects are classified into object types, there is an explicit distinction between "non-lexical" and "lexical" objects, the latter standing for strings or numbers in the universe of discourse;

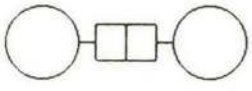
b. all information is stored as link, called fact instance involving two object types - hence the name "binary";
c. (non-lexical) object types may be organized into subtypes (e.g. because of additional fact properties) using sublink types;
d. it supports the specification of constraints, rules and other forms of "semantics" using e.g., some functional language ([25], [16], [10],...).

We adopt the "NIAM" graphical notation for these concepts :

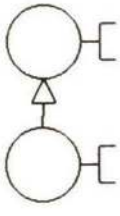
a NOLOT -- (NOn-Lexical Object Type)

a LOT -- (Lexical Object Type). A LOT may be involved in one fact type only, with a NOLOT.

a LOT-NOLOT. Sometimes for notational convenience, for a particular object type we might not care to represent ***explicitly*** the distinction between its non-lexical entities and their lexical representation.

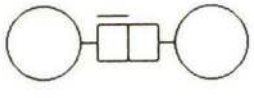


a fact type. The "boxes" are called roles. Each fact type involves exactly two object types (which may be the same).

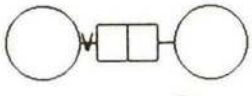


sublink types - the subtype occurrences implicitly inherit all properties of the supertype. Subtypes need not be disjoint; not all of a NOLOT's occurrences need be in one of its subtypes.

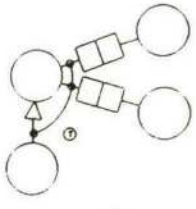
Constraints are named n-place predicates with variables ranging over the chosen object types and some generic lexical object types such as string, integer, real, etc.. Certain constraint types occur so frequently and are so fundamental that they have a graphical representation as well. We only introduce the constraint types used in the example schemas given in this paper:



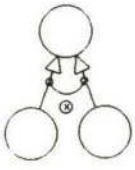
The identifier constraint (simple functional dependency) is drawn as a line over the key role.



A total role constraint stating that each instance of an object type should participate in a given role is represented by a "V" sign.



A total union constraint is a generalization of a total role constraint. Each instance of the object type should participate in at least one of the indicated roles or subtypes.



The exclusion constraint expresses the mutually exclusion of a number of subtypes.

The last three types of constraints are examples of "set-algebraic constraints" on role- and object populations (see below where we briefly discuss BRM analyzer, RIDL-A).

The above is not a formal description of the BRM. For such we refer to the literature e.g. see [5].

A major effect of this modeling technique is the uniform representation of entity types, domains, attributes, relations etc. as object types. Facts may be considered as objects; the actual classical distinction among "entity", "relation" or "attribute" is made by applying a synthesis algorithm to the resulting object/link type structure, rather than being a modelling choice.

3. The RIDL system - Architecture and Functionalities.*

The RIDL* approach to capturing information system specification starts with the information analysis phase. Actual knowledge acquisition about the application domain typically precedes this. Although a module RIDL-F assisting this activity is currently under development as part of RIDL*, we shall not discuss this here.

The architecture of the RIDL* system is presented in figure 1. This displays the three major modules of the system, which are :
1. the RIDL-G module, the conceptual design interface,
2. the RIDL-A module, the validation module,
3. the RIDL-M module, the (relational) database generation module.
We introduce each of these modules successively.

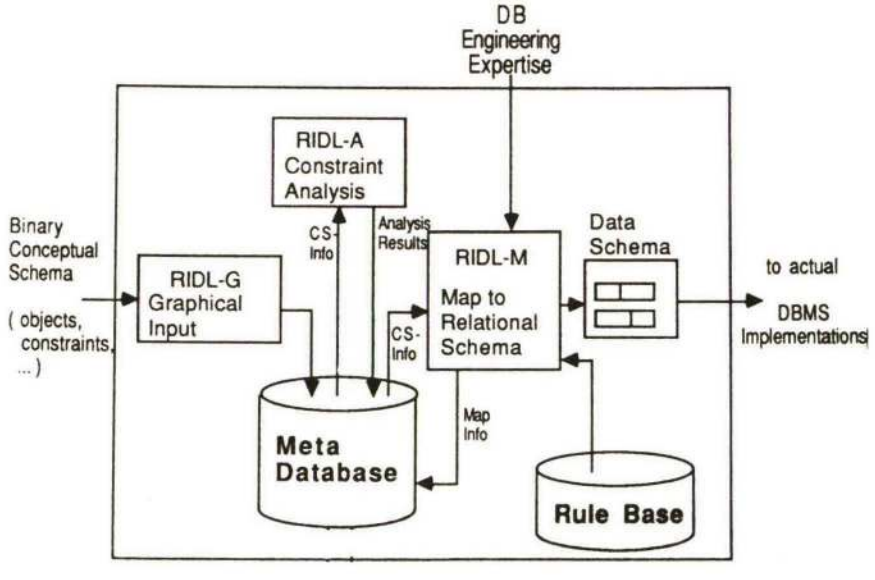


figure 1.

3.1. RIDL-G.

RIDL-G is an interactive graphical module which assists the development of ***binary conceptual schemas*** and provides supporting documentation features. It is a typical graphical tool with windows, (pop up-)menus, icons, mouse and full graphical editing. Figure 2 shows a picture of RIDL-G in action.

The binary conceptual schemas developed with RIDL-G are stored in

RIDL*'s own meta-database. It may contain several independent conceptual schemas. Its implementation is a relational (ORACLE) database, and its design is partly "open", meaning that a comprehensive set of views is available to the RIDL* user to allow him to prepare his own style of data-dictionary and query meta-information for use in his particular project environment.

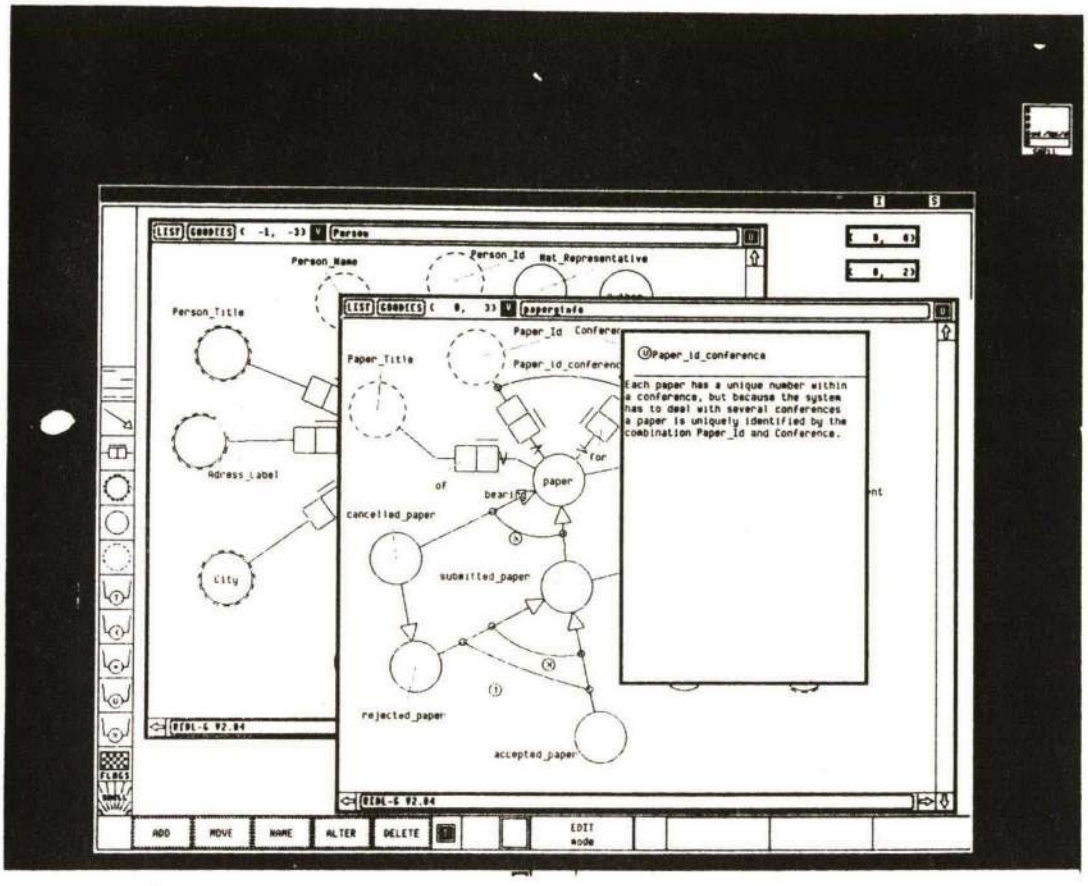


figure 2.

3.2. RIDL-A.

At each stage of the database engineering project the binary schemas may be checked for validity, completeness and consistency using RIDL-A, the analyzer module. RIDL-A performs 4 specific functions:

1. It verifies the correctness of the schema according to the rules of the BRM. Certain rules of the BRM are enforced by RIDL-G as the schema is constructed, the others are checked on demand.
2. It determines whether the binary schema contains all necessary concepts to be a complete description.
3. It verifies the consistency of the set-algebraic constraints defined in the binary schema on the populations of roles and object types.
4. It detects non-referable object types in the conceptual schema, i.e. object types for which it is not possible to refer uniquely and unambiguously (one-to-one) to all of their instances. This one-to-one property should be inferable from constraints in the binary schema. This condition is needed because ultimately we want to obtain a relational database for the conceptual schema. Obviously, the values that will be stored in the database will be "lexical" and thus we need to be guaranteed of a lexical representation-

(type) for each non-lexical object(-type).

3.3. RIDL-M.

The kernel of the RIDL* system and subject of this paper is the RIDL-M module. The "M" stands for Mapper. It takes all or part of the binary schema and generates a relational data schema, ***with additional constraint specifications*** for the semantics given in the binary conceptual schema. Since most RDBMSs at this moment support constraints poorly, if at all except for unique indexes (keys) and/or NOT-NULL conditions, these generated formal constraint specifications may have to find their way into the eventual application designs "by hand".

RIDL-M also maintains extensive and precise maps (in both directions) allowing the application programmer to go back and forth between the conceptual schema and the relational schema generated from it. As such they are nearly indispensable tools for the application designers: they describe how the modeling concepts from the application domain translate into the implementation language of the RDBMS, and vice versa.

Figure 3 gives an illustration of RIDL-M. The leftmost window at the top of the screen shows RIDL-M's user interface. Windows, buttons and pop-up menus are used to control the generation process. A more elaborate description of RIDL-M and its underlying principles are given in the next section.

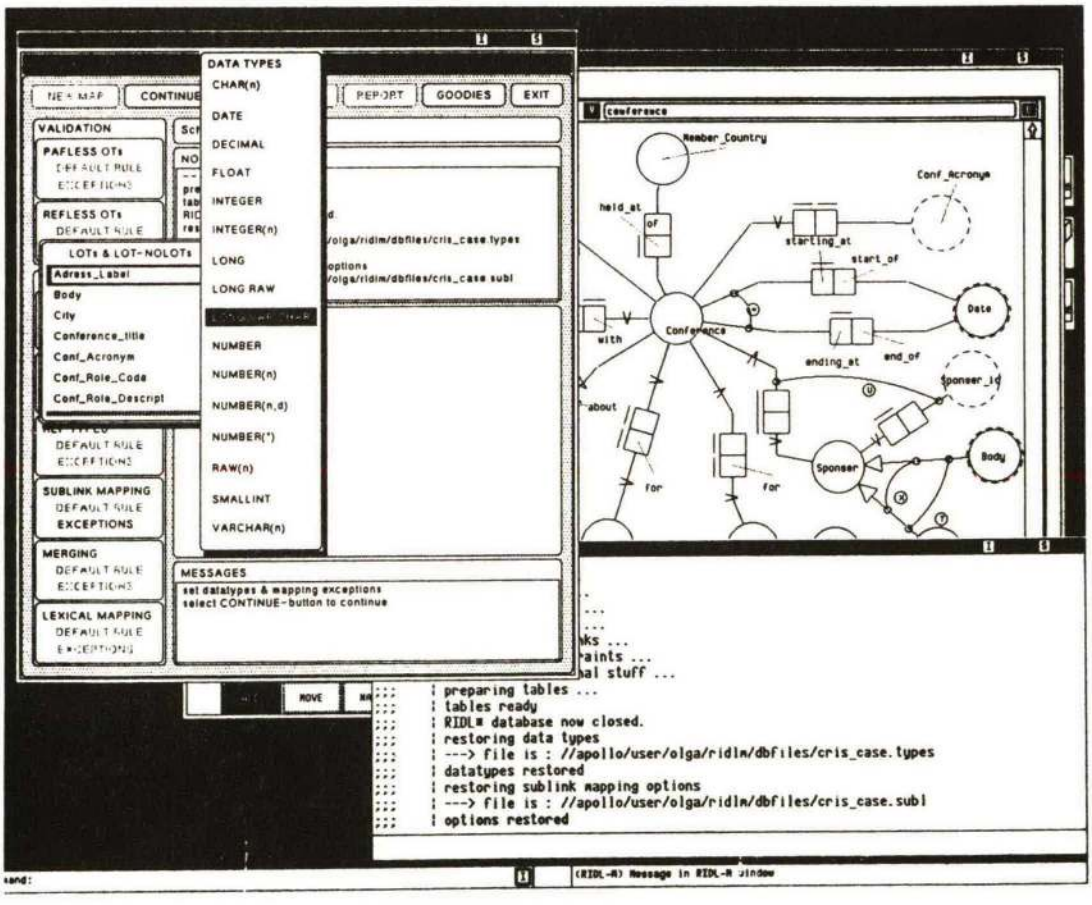


figure 3.

4. The RIDL mapper*

RIDL-M builds a (by default fully normalized) relational data schema from a binary conceptual schema.

Several algorithms to construct a relational schema from a conceptual schema or semantical network are already proposed in the literature (e.g. for the BRM [5] [22], entity-relationship model [2] [23], functional model [12]). These algorithms mostly have common underlying principles which are basically simple and straightforward. As an example we sketch the naive algorithm to transform a binary schema into a relational schema. We presume the binary schema to be correct and complete according to the rules of the BRM (as ascertained by RIDL-A).

step 1: Construct a relation for each NOLOT by grouping all functionally dependent roles for the NOLOT as attributes in one relation.

step 2: For each subtype NOLOT add an extra attribute referring to a supertype of this subtype to the constructed relation. This is needed to later express referential integrity for this subtype.

step 3: For each many-to-many fact type, create a separate relation only consisting of two attributes, one for each role.

step 4: Replace non-lexical attributes (attributes derived from NOLOT-roles) by one of the lexical representation types of the NOLOT from which they are derived. Care has to be taken that the necessary foreign key constraints still can be expressed (i.e. relate to compatible domains).

step 5: Add additional constraints according to the constraints of the binary schema (this is not as easy as it sounds).

It can be shown that in the absence of additional constraints which express functional or multivalued dependencies in a procedural fashion, this algorithm always yields a relational schema in fifth normal form. For most "mapping" algorithms in the literature indeed the main concern is with a proof that the generated relational schema satisfies some normal form and that the algorithms preserves some of the essential properties (like the functional dependency and the referentially integrity property) of the conceptual schema.

However, past experience learns us that for real-life database engineering these algorithms and the tools implementing them are poorly suited. For example, the many smaller tables derived by normalization have to be joined dynamically which may result in an unacceptable increase of I/O consumption [9]. Other problems are due to undocumented decisions or a choice of grouping strategy which may prove inconvenient in the (often already existing) application environment. Furthermore, constraints often considered as first class citizens in the conceptual modelling seem to become pariahs during the transformation process. Only constraint types with a corresponding constraint type in the relational model (e.g. functional dependency, foreign keys) are conserved. In addition, the cross-link between the conceptual and the relational schema is left to the intuition of the application programmer. Name conventions used for relations and

attributes are often the only guiding principle. This may result in a misunderstanding of the structure of the generated schema and a consequent misuse of it, thus losing a great deal of the benefits gained by the conceptual modelling.

We have taken an essentially different starting point for the design of the RIDL-M module. Instead of being merely concerned about the activity of normalization we have developed a method and a tool that with the assistance of a rule base and the expertise of the database engineer generates a more optimal data schema from the viewpoint of the application environment, and therefore not even necessarily in third normal form. To guarantee control over this (much more) complex process and the redundancies it may introduces in the data schema we needed to develop a set of formal methods and foundations.

4.1. Principles Underlying RIDL-M.

In RIDL-M the generation of a relational database schema from a conceptual schema is based on database schema transformation theory [27].
We have adopt a model-theoretic view of databases. We represent a database schema as a logical theory and view the models of the theory as representing possible states of the universe of discourse.
We use the notation STATE(S) to denote the set of all possible models of a database schema (or theory) S and we use the term database state for such a model.

Definition 1.
Given two database schemas S_1 and S_2,
a schema transformation from S_1 into S_2 is defined as a mapping

$$g : STATES(S_1) \rightarrow STATES(S_2)$$

such that, given a database state of S_1 one and only one database state for S_2 is obtained.
S_1 and S_2 need not be schemas using the same type of formalism. E.g. for our purposes S_1 is a database schema expressed in the BRM while S_2 is the "corresponding" data schema expressed in the Relational Model (RM).
In most cases, schema transformations are used to provide different views (e.g. conceptual view, different relational views) on a database. In that case only one of the two databases schemas has a physical representation and a schema transformation is used to generate the "virtual" database state (the view) from the "base" database state. However, there is no reason to restrict this to transformations from "base" to "virtual" databases. When dealing with update specifications on virtual databases or with data translations between different databases we also have to consider the inverse mapping to assure to be able to go back and forth between the two databases.

Definition 2.
A schema transformation g from S_1 into S_2 is lossless if g is one-to-one (bijection).
In this case S_1 and S_2 are said to be state equivalent.

Requiring a transformation g : S1 -> S2 to be one-to-one implies that it should not be possible to have a database state for S2 without a unique corresponding database state for S1, hence the definition state equivalence.
It is important to note that state equivalence does not imply that

both schemas have the same semantical power. As an example a binary schema containing sublinks can be transformed into a state-equivalent binary schema without sublinks (see figure 4). It can be shown easily that this last schema expresses less semantics than the original one.

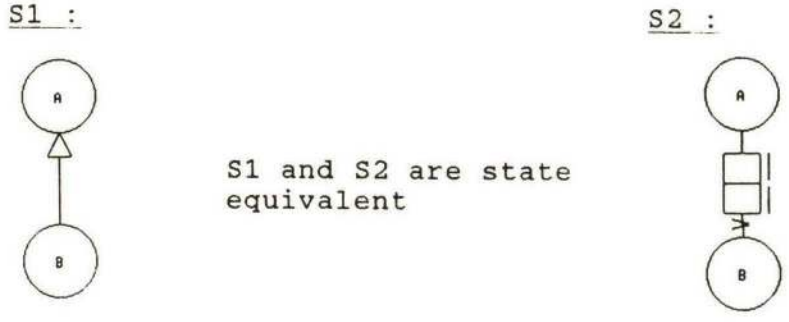


figure 4.

In general, schema transformations from the BRM to the relational model are not one-to-one. This is because the two models do not have the same expressive power. Either we need to restrict the class of binary schemas which can be transformed into a state equivalent relational schema, or we need to extend the relational model with additional constraint types. These constraint types are needed firstly to express in the relational schema the constraints defined in the BRM schema and secondly to state the losslessness of the transformation. The constraints which assure the losslessness of the transformation are called the lossless rules of the transformation. Naturally, we have chosen to extend the relational model, it being the target of our efforts. We generate the necessary constraints in a pseudo-SQL or using SQL2 [11] such that an application programmer may be able to incorporate them into the database applications in order to achieve state equivalency of the relational schema with its BRM definition.

Defining the transformation from a BRM schema to a RM schema as a "monolithic" transformation algorithm is not very useful: it would not achieve the flexibility required for a database engineering tool and it would be very hard to prove the losslessness of this transformation. Therefore we have defined this schema transformation as the composition of a number of very basic schema transformations. These basic schema transformations are quite elementary and therefore it is easier to prove their losslessness.

The basic schema transformations used can be divided into three kinds: binary to binary, binary to relational, and relational to relational. The transformations of the first kind are used to convert a binary schema into its most canonical form. They eliminate superfluous definitions, reduce constraints to their canonical form and replace non-elementary concepts by their definitions. The transformations of the second kind transform such a canonical binary schema into a "binary" relational schema and the transformations of the third kind are used to "sculpt" this relational schema. An example transformation of this last kind is the well known projection/join transformation used to obtain relations in third normal form or conversely to combine relations into one relation [24]. The lossless rules of this transformation include a multivalue dependency for the projection transformation and an equality constraint for the inverse join transformation.

There is an important advantage to this transformation composition technique. We are now able to "drive" the composition of these basic transformations by rules specified externally to the algorithm. In this way external control may ultimately influence the transformation process nearly without limitations. Currently a limited number of these rules are built in and externalized as options or choices available to the database engineer e.g. the treatment of null-values. These options are explained in more detail in section 4.2.
In a later implementation these rule specifications may in part be extracted from functional requirements and process specifications obtained through suitable tools (RIDL-F, RIDL-P) which are currently under development. For example, query information can be used to steer the mapping towards limited de-normalization whereas right now the database engineer has to infer the correct RIDL-M controls from his own knowledge.

This RIDL-M architecture is illustrated in figure 5. The transformation base contains the basic schema transformations, the rule base contains the rules which together with the user mapping options drive the transformation engine. The meta database contains the schema to be transformed.

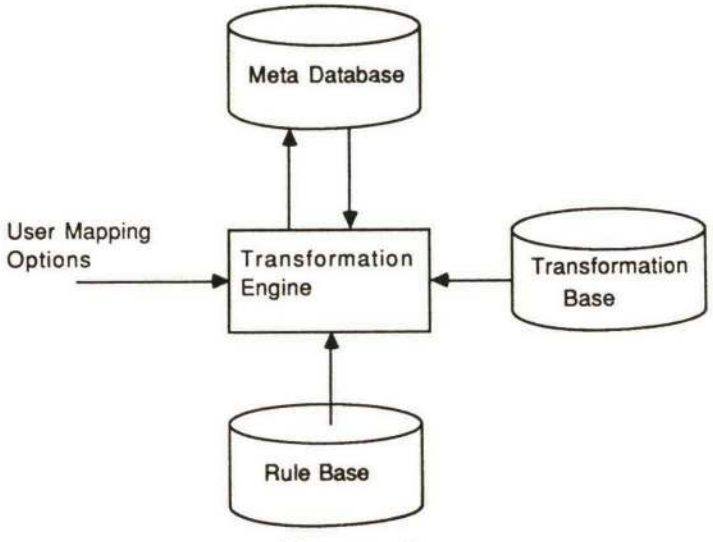


figure 5.

The basic schema transformations and the proof of their correctness will be given in forthcoming work.

4.2. The mapping options.

As explained in section 4.1 the transformation process can be influenced by the database engineer. This can be done by exercising a number of "mapping options" that trigger the rules which influence the mapping process. These mapping options include :
1. control on the admissibility of null values in attributes,
2. the mapping of sublink types,
3. the choice of different lexical representations for a NOLOT,
4. the decision whether to combine tables ,
5. when and how to omit certain tables.

Mapping options to control denormalization are currently under development.
Because of space limitations we only explain the first three mapping options.

4.2.1. The null value options.

The null value option controls the admissibility of null values in attributes. By default, null values are inadmissible in attributes which are part of the primary key of a relation (as stated in the "Entity Integrity Rule" of the relational model, see [4]). In the remaining attributes, null values may be admissible depending on constraints specified in the binary schema.

Next to this default option there are three alternative options : "NULL NOT ALLOWED", "NULL NOT ALLOWED IN KEYS" and "NULL ALLOWED".

The first alternative, "NULL NOT ALLOWED" is a very restrictive one; none of the attributes should allow null values. This implies that according to the constraints of the binary schema (mainly total role and role-equality constraints) the fact types of the binary schema will be grouped into relations in such a way that null values in the data schema are not needed. As a consequence, a large number of small tables will in general be generated.

The second alternative, "NULL NOT IN KEYS", restricts the admissibility of null values to attributes not part of a primary or a candidate key.

The third alternative, "NULL ALLOWED", as a matter of fact allows the database engineer to violate the earlier mentioned "Entity Integrity Rule" of the RM. The reason why we have introduced the possibility to obviate this integrity rule is the following. Some NOLOTS may only have a non-homogenous lexical representation type. The entities of such a NOLOT are distinguishable but there is no overall unique identification function that applies to all of them. This means in turn that there is no "primary key" concept for these entities. However, there will be two or more candidate keys and for each given entity of the NOLOT at least one of them will act as primary key. To keep information on such a non-homogenously referencible NOLOT into one relation (rather than possibly introducing redundancy by duplicating this information into two or more relations), we have to allow null values in the "primary keys". Furthermore some relational database systems allow null values also in primary key attributes (ORACLE is an example).

4.2.2. The sublink mapping options

A sublink mapping option controls the transformation of the sublink types of the binary schema. By default the (identified) fact types defined on the subtype of a sublink type are grouped into one relation, called the sub-relation in RIDL-M terminology, and those defined on the supertype of the sublink are grouped into another relation, called the super-relation. The sublink type is expressed by means of a *foreign key*, linking the primary key or a candidate key of this sub-relation to a primary key in the super-relation. This option is the default and is announced as "SUBOT & SUPOT SEPARATE".

The sublink mapping option "SUBOT & SUPOT TOGETHER" groups all (identified) fact types defined on the subtype as well as those on the supertype of a sublink type into one relation. This sublink mapping option permits to perform stronger typing on the conceptual schema without needlessly compromising the efficiency requirements. Indeed the default sublink mapping option (strong typing) in general results

in a larger number of relations with only a few attributes. Therefore more dynamic joins might be needed.

The third sublink mapping alternative is called "SUBOT INDICATOR FOR SUPOT". This option groups fact types like for the default option, but causes an extra attribute (called the indicator attribute) to be added to the relation derived for the supertype; the value in a given tuple (row) of this attribute is supposed to indicate whether the tuple corresponds to a tuple in the sub-relation or not. By adding this indicator attribute, redundancy of a "procedural" kind is introduced, presumably for the benefit of query efficiency. To control this redundancy RIDL-M generates extra constraints (a "conditional" equality constraint). If the target DBMS does not support this type of constraint then an SQL-like statement is generated which to an application programmer acts as a formal specification for a program segment to enforce this constraint. (For an example, see below.)

The sublink mapping option is a global option with exceptions; the selected option holds for all the sublink types of the binary schema, but may be overridden for chosen individual sublink types.

4.2.3. The lexical mapping options.

It is explained in section 2 how NIAM makes an explicit distinction between non-lexical objects and lexical objects. Data Base Management Systems typically deal only with lexically represented information. It is of course possible to introduce surrogates [4] as a representation for non-lexical objects, but this representation is an artifact. Consequently, this means that the non-lexical information has to be represented by lexical data in the relational schema. See also section 3.2, there we also introduced for a NOLOT the concept of lexical representation type or naming convention; a way to refer to a NOLOT by a (combination) of LOT(s). It is quite usual to have several, even a great many, lexical representation types for the same NOLOT.

Often therefore, we have to choose the lexical representation type that will be used to represent the NOLOT instances in the database. RIDL-M selects for each NOLOT the "smallest" lexical representation type, this is the one which involves the least number of LOTs and NOLOTs and will have the smallest physical representation as derived from the data types of the LOTs involved. Since this limits the freedom of the database engineer, flexibility needs to be added to allow selection for each NOLOT of the preferred lexical representation and even to use more than one representation type for a NOLOT. Even within the same relation two different naming conventions for the same NOLOT might be useful, e.g. if that would allow easier expression of a particular constraint or query.

Below are four state equivalent (normalized) relational schemas for the binary given given in figure 6. They are generated using different mapping option combinations. Other alternatives are possible as well. We use a straightforward graphical representation for a relational schema. Attributes allowing null values have their name between brackets. A primary key is indicated by a full line, a candidated key by a dotted line and a foreign key by an arrow. Additional constraints are given using SQL2 syntax [11].

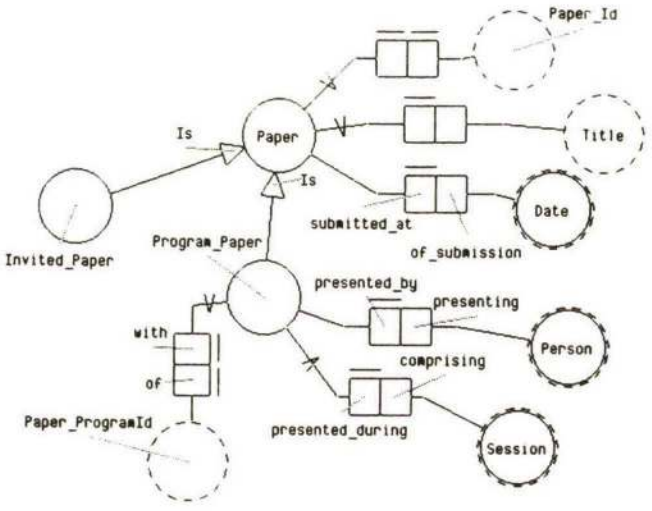
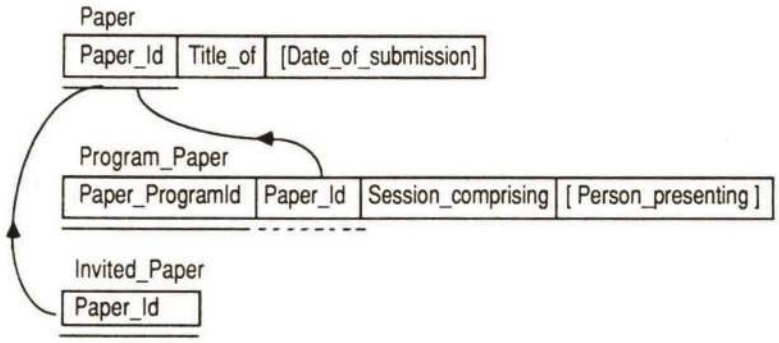
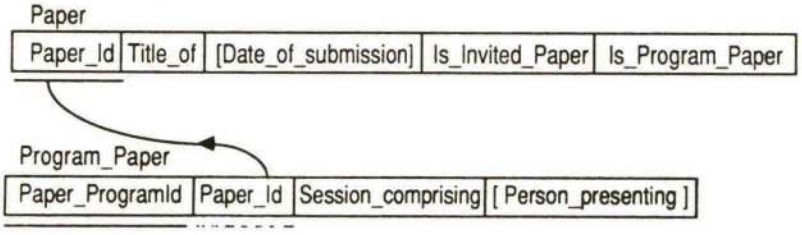


figure 6.

Alternative 1.



Alternative 2.




```
EQUALITY VIEW CONSTRAINT :
        ( SELECT Paper_Id
          FROM Program_Paper
        )
        IS EQUAL TO
        ( SELECT Paper_Id
          FROM Paper
          WHERE ( Is_Program_Paper = 1 )
        )
 CONSTRAINT C_EQ$_3
```

Alternative 3.

Paper

Paper_Id	Title_of	[Date_of_submission]	Is_Invited_Paper	[Paper_ProgramId_Is]

Program_Paper

Paper_ProgramId	Session_comprising	[Person_presenting]

```
EQUALITY VIEW CONSTRAINT :
        ( SELECT Paper_ProgramId
          FROM Program_Paper
        )
        IS EQUAL TO
        ( SELECT Paper_ProgramId_Is
          FROM Paper
          WHERE ( Paper_ProgramId_Is IS NOT NULL )
        )
  CONSTRAINT C_EQ$_3
```

Alternative 4.

Paper

Paper_Id	Title_of	[Date_of_submission]	Is_Invited_Paper	[Paper_ProgramId _with]

[Session_comprising]	[Person_presenting]

```
CHECK( --  Dependent Existence
       (   ( Person_presenting IS NOT NULL )
       AND ( Paper_ProgramId_with IS NOT NULL )
       )
    OR ( Person_presenting IS NULL )
     )
CONSTRAINT C_DE$_8

CHECK( --  Equal Existence
       (   ( Paper_ProgramId_with IS NULL )
       AND ( Session_comprising IS NULL )
       )
    OR (   ( Paper_ProgramId_with IS NOT NULL )
       AND ( Session_comprising IS NOT NULL )
       )
     )
CONSTRAINT C_EE$_6
```

4.3. The RIDL-M output

The relational schema built by RIDL-M is independent of any target DBMS, it is called a generic relational schema. From this generic relational schema a schema definition for any relational (or relation-like) DBMS can be derived using the specific database definition language of such a DBMS.
At the time of writing, RIDL-M generates fully operational ORACLE, INGRES and DB2 schema definitions, and a "neutral" schema definition in the SQL2 (draft) standard [11]. Syntaxes for SYBASE and other RDBMSs are in the works.
Below is a fragment of a generated SQL2 schema definition for the binary schema of figure 6. Notice the additional semantics, given in an SQL-like fashion, which express the constraints specified in the conceptual schema. They are added as comment lines because (even) the SQL2 standard does not currently support these type of constraints.

```
-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--                              TABLE Program_Paper
-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CREATE TABLE  Program_Paper
  ( Paper_ProgramId
       D_Paper_ProgramId  -- DATA TYPE CHAR(2)
       NOT NULL
       PRIMARY KEY
         CONSTRAINT C_KEY$_11
       REFERENCES Paper ( Paper_ProgramId_Is )
         CONSTRAINT C_FKEY$_8
  , Person_presenting
       D_Person           -- DATA TYPE CHAR(30)
       -- NULL
  , Session_comprising
       D_Session          -- DATA TYPE NUMERIC(3)
       NOT NULL
  );

-- ------------------------------------------------------------------------------
--                    View Constraints For Table Program_Paper
-- ------------------------------------------------------------------------------

-- EQUALITY VIEW CONSTRAINT :
--          ( SELECT Paper_ProgramId
--            FROM Program_Paper
--          )
--          IS EQUAL TO
--          ( SELECT Paper_ProgramId_Is
--            FROM Paper
--            WHERE ( Paper_ProgramId_Is IS NOT NULL )
--          )
--    CONSTRAINT C_EQ$_3
-- ------------------------------------------------------------------------------
```

generated relational schema fragment

In addition to the generated schema definition files, RIDL-M provides a detailed so-called map report. This report describes the complete cross-reference link (in both directions) between the conceptual binary schema and the generated relational schema (in its RDBMS syntax). The map report is divided into two parts, the forwards map and the backwards map. The forwards map describes how each of the binary schema concepts (LOTS, NOLOTS, facts, roles, sublinks *and constraints*) are expressed in the relational schema. The backwards map tells how the relational schema concepts are derived from the binary schema concepts. More specifically, for each generated relational schema concept (domain, relation, attribute, constraint) the binary schema concepts from which it is derived (or which have participated in its derivation) are given. Below are two fragments of the map report associated with the SQL2 schema definition given higher up. The first fragment is a excerpt of the forward map report, the second fragment is a part of the backwards map report.

The map report is essential for application programmers; they need to know how to translate (high level) process specifications on the conceptual schema into application programs onto the generated data schema and how to interpret the results of such application programs (output data, error messages, etc.) into conceptual schema terms. And this forwards map will also play a key role in ultimately *compiling* such high-level process specifications into relational application programs. An early production-quality prototype of such a compiler for query processes on the BRM, known as the RIDL compiler (built in 1983, [6]) has already proven the effectiveness of that approach.

```
-------------------------------------------------------------------------------
FACT WITH ROLE presented_by ON NOLOT Program_Paper AND ROLE presenting ON
 LOT-NOLOT Person
      MAPPED TO
      SELECT Paper_ProgramId , Person_presenting
      FROM Program_Paper
      WHERE ( Person_presenting IS NOT NULL )
-------------------------------------------------------------------------------
FACT WITH ROLE presented_during ON NOLOT Program_Paper AND ROLE comprising ON
 LOT-NOLOT Session
      MAPPED TO
      SELECT Paper_ProgramId , Session_comprising
      FROM Program_Paper
-------------------------------------------------------------------------------
                              ...
-------------------------------------------------------------------------------
SUBLINK Is FROM NOLOT Program_Paper TO NOLOT Paper
      MAPPED TO
      SELECT Paper_ProgramId_Is , Paper_Id
      FROM Paper
      WHERE ( Paper_ProgramId_Is IS NOT NULL )
-------------------------------------------------------------------------------
                              ...
-------------------------------------------------------------------------------
IDENTIFIER : ROLE ON NOLOT Paper AND LOT Paper_Id
      MAPPED TO
      UNIQUE( Paper_Id )
          ON Paper
        CONSTRAINT C_KEY$_5
-------------------------------------------------------------------------------
```

fragment 1

```
TABLE Paper
      DERIVED FROM
      FACT WITH ROLE ON NOLOT Paper AND ROLE ON LOT Title ,
      FACT WITH ROLE ON NOLOT Paper AND ROLE ON LOT Paper_Id ,
      SUBLINK Is FROM NOLOT Program_Paper TO NOLOT Paper ,
      SUBLINK Is FROM NOLOT Invited_Paper TO NOLOT Paper ,
      FACT WITH ROLE submitted_at ON NOLOT Paper AND ROLE of_submission ON
 LOT-NOLOT Date
-------------------------------------------------------------------------------
                              ...
-------------------------------------------------------------------------------
COLUMN Paper_ProgramId IN TABLE Program_Paper
      DERIVED FROM
      ROLE of ON LOT Paper_ProgramId - ROLE presented_by ON NOLOT Program_Paper
      ,
      ROLE of ON LOT Paper_ProgramId - ROLE presented_during ON NOLOT
      Program_Paper ,
      ROLE of ON LOT Paper_ProgramId - ROLE with ON NOLOT Program_Paper ,
      ROLE of ON LOT Paper_ProgramId
-------------------------------------------------------------------------------
                              ...
-------------------------------------------------------------------------------
EQUALITY VIEW CONSTRAINT :
    ( SELECT Paper_ProgramId
      FROM Program_Paper
    )
    IS EQUAL TO
    ( SELECT Paper_ProgramId_Is
      FROM Paper
      WHERE ( Paper_ProgramId_Is IS NOT NULL )
    )
  CONSTRAINT C_EQ$_3
      DERIVED FROM
      NOLOT Program_Paper ,
      SUBLINK Is FROM NOLOT Program_Paper TO NOLOT Paper ,
      TOTAL : ROLE presented_during ON NOLOT Program_Paper AND LOT-NOLOT
      Session ,
      TOTAL : ROLE with ON NOLOT Program_Paper AND LOT Paper_ProgramId
-------------------------------------------------------------------------------
FOREIGN KEY Program_Paper ( Paper_ProgramId )
        REFERENCES Paper ( Paper_ProgramId_Is )
  CONSTRAINT C_FKEY$_8
      DERIVED FROM
      SUBLINK Is FROM NOLOT Program_Paper TO NOLOT Paper
-------------------------------------------------------------------------------
```

fragment 2

5. Concluding Remarks

We have described the database design process used in RIDL*, a database engineering workbench based essentially on the NIAM method. In this method, information system design starts at the conceptual level, resulting in a (binary) conceptual schema independent of any implementation considerations. Afterwards this binary schema is transformed into a relational schema. During this transformation, called the mapping process, implementation and efficiency aspects may be taken into consideration in order to generate a (relational) data schema that will give the best performance in the given application environment. To do this the database engineer disposes of a number of so-called mapping options. Current research is concentrated on how to expand RIDL-M into a rule driven system, that also has the capability to automatically generate the database schema that best fits a particular application environment. To achieve this, we are currently defining such a set of "expert" rules to drive the transformation process. Next, we shall extract the triggers for these expert rules from requirements and functional specifications supplied by the RIDL* user. Note that it is precisely the way the mapping "algorithm" is implemented as a programmable sequence of elementary transformations that enables us to do this.

A note on the implementation:

RIDL* as described above has been completely implemented [7]. It runs currently on an Apollo workstation, and is mostly written in Common Lisp (RIDL-M) and Objective C (RIDL-A). It is being used at the time of this writing at a few industrial locations where it routinely generates databases of up to 120-150 ORACLE tables (this is not a limit). More interestingly perhaps, the generated (pseudo-)SQL constraints cause the output design to reach approx. 1 to 1.2 pages per table on the average, not counting forwards or backwards maps.

Acknowledgements

I would like to thank Robert Meersman for suggesting improvements to an earlier draft of this paper.

BIBLIOGRAPHY and REFERENCES

[1] Abrial J.R., "Data Semantics". In : Database Management Systems. Eds. J.W. Klimbie, K.L. Koffeman. Elsevier North Holland, New York (1974).

[2] Casanova M.A., Amaral de Sa J.E., "Mapping Uninterpreted Schemes into Entity-Relationship Diagrams: two Applications to Conceptual Schema Design". In: IBM Journal of Research and Development 28(1) pp. 82-94 (1984).

[3] Chen P.P., "The Entity-Relationship Model - towards a unified view of data". In : ACM Trans. on Database Systems 1(1) pp. 9-36 (1976).

[4] Date C.J., "An Introduction to Database Systems, Volume II". Addison-Wesley Publishing Company (1980).

[5] De Troyer O., Meersman R., "Transforming Conceptual Schema Semantics to Relational Data Applications". In: Information Modelling and Database Management. Ed. H. Kangassallo. Springer

Verlag (1987).

[6] De Troyer O., Meersman R., Ponsaert F., "RIDL User Guide", Control Data DMRL Research Memorandum (1983) [available from the authors].

[7] De Troyer O., Meersman R., Verlinden P., "RIDL* on the Cris Case: A Workbench for NIAM". In [19].

[8] Falkenberg E., "Concepts for Modelling Information". In: Modelling in Data Base Management Systems, Proceedings of IFIP TC-2 Conference. Ed. G.M. Nijssen. North Holland Publishing Company (1976).

[9] Inmon W.H., "Optimizing Performance with Denormalization". In: Database Programming and Design 1(1) (1987).

[10] International Standards Organisation, "Concepts and Terminology for the Conceptual Schema and the Information Base". ISO TR #9007 (also as: N695; Ed. J.J. van Griethuysen) (1982).

[11] International Standards Organization - ANSI, "SQL-2 Standard" (working draft addition, ANSI X3H2-88-72, ISO DBL CPH-2)". Ed. J. Melton (1988).

[12] Kerschberg L., Pacheco J., "A Functional Data Base Model". In Series: Monographs in Computer Science and Computer Applications. No.2/76 Pontificia Universidade Catolica, Rio de Janeiro, Brasil.

[13] Lee R.M., "Logic Semantics and Data Modeling: An Ontology". In: "Data and Knowledge", Proceedings of IFIP Working Conference DS-2, Eds: R. Meersman, A. Sernadas, North-Holland Publ. Co. (1988)

[14] Mark L., Roussopoulos N., "Integration of Data, Schema and Meta-Schema in the context of Self-documenting Data Models". In: Entity-Relationship Approach to Software Engineering. Elsevier Science Publishers (North Holland) (1983).

[15] Mark L., "The Binary Relationship Model - 10th Anniversary". Technical Report CS-TR-1933, U. of Maryland, College Park, MD, USA (1987).

[16] Meersman R., "The RIDL Conceptual Language", Control Data DMRL research memorandum (1982) [available from the author].

[17] Meersman R., "Towards formal models for reasoning about conceptual database design". In: "Data and Knowledge", Proceedings of IFIP Working Conference DS-2, Eds: R. Meersman, A. Sernadas, North-Holland Publ. Co (1988)

[18] Nijssen G.M., "A Gross Architecture for the next generation Database Management Systems". In : Modelling in Database Management Systems; proceedings of IFIP TC-2 Conference. Ed. G.M. Nijssen. North Holland Publishing Company (1976).

[19] Olle, T.W., Verrijn-Stuart A.A., and Bhabuta L.: "Computerized Assistance During the Information Systems Life Cycle". Proceedings of the IFIP CRIS-88 Conference, North-Holland Publishing Company, Amsterdam (1988-9)[to appear].

[20] Olle, T.W.: "Design Specifications for Conference Organization". In Appendix B of [19].

[21] ORACLE Corp., "Oracle V.5 SQL User Manual", One Oracle Plaza, Belmont, CA, USA (1987).

[22] Shoval P., Even-Chaime M.: "ADDS: A System for Automatic Database Schema Design Based on the Binary-Relationship Model". In: Data & Knowledge Engineering 2(2), North Holland Publishing Company (1987).

[23] Teorey T.J., Yang D., Fry J.P.: "A Logical Design Methodology for Relational Databases using the Extended ER-model". In: Comp. Surveys 18(2), (1986).

[24] Ullman J.D.: "Principles of Database and Knowledge Base Systems", Pitman (1988).

[25] Verheijen G., van Bekkum J., "NIAM : An Information Analysis Method". In : Proceedings of IFIP TC-8 Conference on Comparative Review of Information Systems Methodologies (CRIS-1). Eds. A. Verrijn-Stuart, T. W. Olle, H. Sol. North Holland (1982).

[26] Wintraecken J.J., "NIAM in Theorie en Praktijk", Academic Service, 1986 [Book in Dutch, to appear in English, Reidel Publ. Co, 1988-9].

[27] Kobayashi I., "Losslessness and Semantic Correctness of Database Schema Transformation: Another Look of Schema Equivalence". In: Information Systems Vol. 11, No. 1, pp 41-59, 1986.

ITK: P.O. BOX 90153 5000 LE TILBURG THE NETHERLANDS