COMIC:

A SYSTEM FOR CONCEPTUAL MODELLING AND INFORMATION CONSTRUCTION*

Hannu Kangassalo

University of Tampere, Department of Computer Science,
P.O.Box 607, SF-33101 Tampere 10, Finland

Abstract: The designer of a data base or an information system has to develop conceptual models describing the Universe of Discourse (UoD) for the specification of meaning and semantic structure of the data. He has also to handle descriptions of data structures and perform technical design of the system. Finally the system must be implemented. Often the amount of work required is so large that it would be useful to have a computerized tool to support the design process. In this paper such an integrated system is briefly described. The system, called COMIC, supports first the conceptual modelling phase, then the logical design phase, and finally the implementation of the data base, and moreover it can be used as a query and update system of the data base. The graphical language, CONCEPT D, used to describe a conceptul schema is new. CONCEPT D is a visual language that supports conceptual modelling of the UoD, the development of a conceptual schema, and the use of the application data base corresponding to the conceptual schema. The language is based on the intensional approach to conceptual modelling, i.e. the description of knowledge in concepts is emphasized instead of extensions of these concepts. The language contains three sub-languages: one, called CONCEPT D/D, for knowledge acquisition about concepts used in the UoD and developing graphic descriptions of concept definition hierarchies, one, called CONCEPT D/CS, for describing conceptual schemata of the UoD, and one, called Conceptual Query Language CQL, for interacting with the application data base. A conceptual schema is used as a basis for making queries from the application data base corresponding to the conceptual schema. A graphic conceptual schema supports the user in recalling and understanding the conceptual structure of the UoD. It hides the relational data base schema from the user so that he works only with concepts of the UoD. He has not to know e.g. the relations or foreign keys between relations. In this work some of the main features of CQL are presented. A query is formulated by pointing at concepts required, and possibly by giving some selection criteria and instructions for formatting the output. CQL is unique in its capability in manipulating concept definition hierarchies. The core component of the COMIC system is the concept data base on which other components, such as a conceptual schema editor, a schema translator and a conceptual query language have been built.

1. INTRODUCTION

In the design of data bases and information systems there has been a continuous trend to increase the amount and complexity of data processed by these systems. Accordingly, the utilization of semantic information describing the Universe of Discourse (UoD) and the meaning of data has been increasing [1,7,8]. There are several reasons for this, arising from different backgrounds:

1. The requirements from users to improve the quality of data produced by information systems.
2. In order to benefit from formal design methods and advanced design tools it is necessary to describe the object of design, i.e. data, in greater detail than before [13,28].
3. Attempts to apply and integrate methods from artificial intelligence to data management, software engineering and decision support systems [3,4,5,28,29].
4. Attempts to develop methods and tools for improving users' knowledge of the semantics of available data [1,6].

Conceptual modelling itself, by which we mean the study and development of concepts concerning the UoD, and the construction of models by using these concepts, has been studied very little. The best introductions seem to be found in the literature concerning the construction of scientific theories and the analysis of properties of these theories [eg. 9,10]. More work has been done on modelling languages, such as the Entity-Relationship approach [13].

Because of the requirements described above, the design and management of the information resources of a company is becoming more and more important as well as more and more difficult. The effective management of the structure and meaning of data requires that there are efficient and flexible design and management tools available, which must be easy to use.

The goal of the COMIC system is to support the entire design process of a data base and an information system, as well as their implementation and use. The main emphasis is on the methods and tools for conceptual modelling. However, attention has also been given to the entire data base design methodology, because unintegrated methods and tools have often shown themselves difficult to use and they have been abandoned after short initial trials. The COMIC system releases a user from the burden of knowing data base technology in detail, so that he can concentrate on solving problems that relate to the domain of application.

In section 2 the phases of the data base design process are briefly described using concepts and terminology of the approach used in this work. The architecture of the COMIC system in its current form is described in section 3. Section 4 contains a short summary.

2. THE DATA BASE DESIGN PROCESS

Data base design ranges from the design and definition of the conceptual content of the UoD to the technical specification of a data base. In its first phases it concentrates strongly on the recognition, analysis and definition of concepts used in the UoD and later applied in the definition of meaning of data and relationships between components of data. The definition of meaning of data requires that the concepts are well known and defined. From this it follows that conceptual modelling of the UoD has become an inseparable part of all design methodologies (see, e.g., [33]). Data base design can no longer be seen as a single, purely technical task. It pervades the whole information system design and implementation process. The conceptual structure of data can not be implicitly built into the system or programs, but it must be explicitely described and it must be possible to study it separately when needed. The knowledge about concepts of the UoD and their relationships has to be moved from programs into the conceptual

description of the UoD and made available to all users or designers who may need it. (see [36]).

The goal of the data base design is thus, in addition to the technical design of a data base, to design the information content of data so that it is well defined and easily adaptable according to changes of the UoD, and is independent of changes in the technical implementation of the system. To reach this goal requires that the technical structure of data be designed on the basis of a carefully analyzed and defined conceptual content of the UoD.

In figure 1 a model of the data base design process is shown. For the sake of clarity, the describtions of possible iterations have been left out. Similar models have been presented elsewhere, e.g., in [12]. The main difference to other models is in the concepts and methodology applied in the first three phases of this model. The phases of work are symbolized by boxes and the results of each phase by ovals. The arrows show the main direction of the design process. The first three phases are completely independent of any data base management system. Their goal is to develop a conceptual schema describing the UoD of the user community. It forms the conceptual basis for the whole design process.

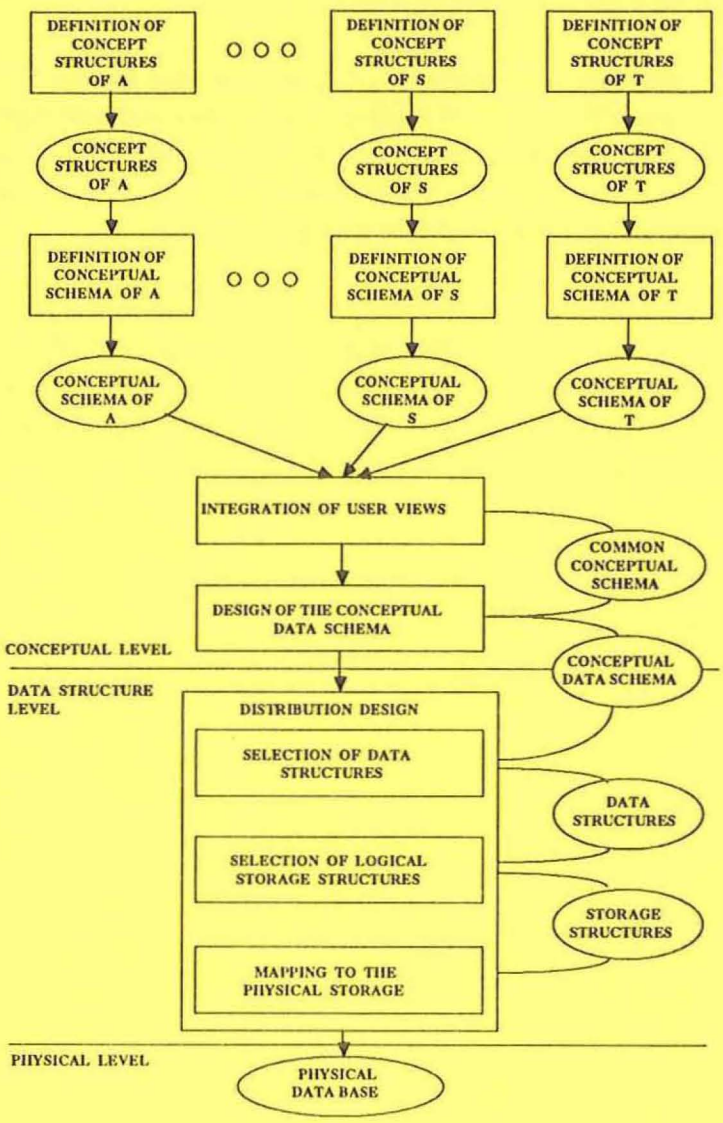

Figure 1. A model of the data base design process.

A conceptual schema contains a conceptual description of the UoD and data collected from the UoD. It is developed by using only concepts used in the UoD and it consists of concepts and rules of the UoD.

It is important to note that it contains (or it should contain) all the concepts and integrity rules recognized in the UoD (100%-rule) [36].

The design is started by collecting, analyzing and defining the concepts of the UoD needed by users. In figure 1 the users or groups of users are symbolized by capital letters A, B,...,T. The goal of this phase is to establish a well defined conceptual foundation for further work. The results are called concept structures. A concept structure is a hierarchical description (definition) of a concept. It gives a detailed analysis of the knowledge content of a concept.

From experiments we have learned that, at least as far as abstract and complex concepts are concerned, the construction of concept structures is a very useful or even necessary step, instead of trying to build a conceptual schema directly [23]. Concept structures are building blocks from which the conceptual schema can be constructed.

Concept structures are regarded as separate objects. In the second phase the UoD of a user or of a group of users is described by using concepts defined by them. The result is a conceptual schema or view of the UoD. There is at least one view for each user or a group of users.

It should be noted that a view is not a description of data alone, but is a local 'theory' of the UoD. It contains concepts and rules on different levels of abstraction. Only some of them will be implemented as data and programs in the information system. The rest are necessary for correct function of the UoD and for understanding the data. Thus the view is also a local knowledge base, which can be used e.g. for training new employees, for design of new rules or concepts, etc.

The number of user views may be large. Because each of them is a subjectively biased description of the users' UoD, they must be integrated into a common conceptual schema from which all contradictions have been removed and relationships between different views, as well as common components of views, are described in an unified way. Integration is done in the third phase. It is a complex task, which needs good support for checking the completeness, integrity and consistency of the common conceptual schema [2,16,30].

Often a conceptual schema contains structures and rules for which there are no suitable means for implementation in the underlaying data base management system. It must then be restricted to fit to the rules of the data model and the DBMS. This is done in the fourth phase. Structures and rules which cannot be implemented by using the DBMS must be taken out of the conceptual schema and implemented in application programs.

3. THE COMIC SYSTEM

3.1. An overview of the system architecture

In the following a short description of the architecture and functions of the COMIC system is given. The COMIC system is more than just a collection of design tools. It is an integrated design support system for eliciting and acquisition knowledge from users, for creating, storing and manipulating a conceptual schema, and for using the conceptual schema and the corresponding application data base. The conceptual schema is the core of the architecture. It is connected with an underlying data base management system in such a way that it can be used directly as a component of the conceptual query language. The COMIC system consists of several integrated components (see figure 2):

1. **Concept editor** for eliciting and acquiring knowledge from users and for creating, manipulating and storing graphical concept structures.
2. **Conceptual schema editor** for creating, storing, browsing and manipulating a conceptual schema based on concept structures.
3. **Integrator** for pairwise integration of conceptual schemata.
4. **Schema translator** for transforming a conceptual schema into a corresponding relational data base schema which can be directly given to a relational data base management system.
5. **Conceptual query language CQL** for making queries and updates to the relational data base by pointing at components in the conceptual schema.
6. **Concept base management system** for storing semantic information contained in concept structures and conceptual schemata.
7. **Application data base** which contains data corresponding to the conceptual schema.

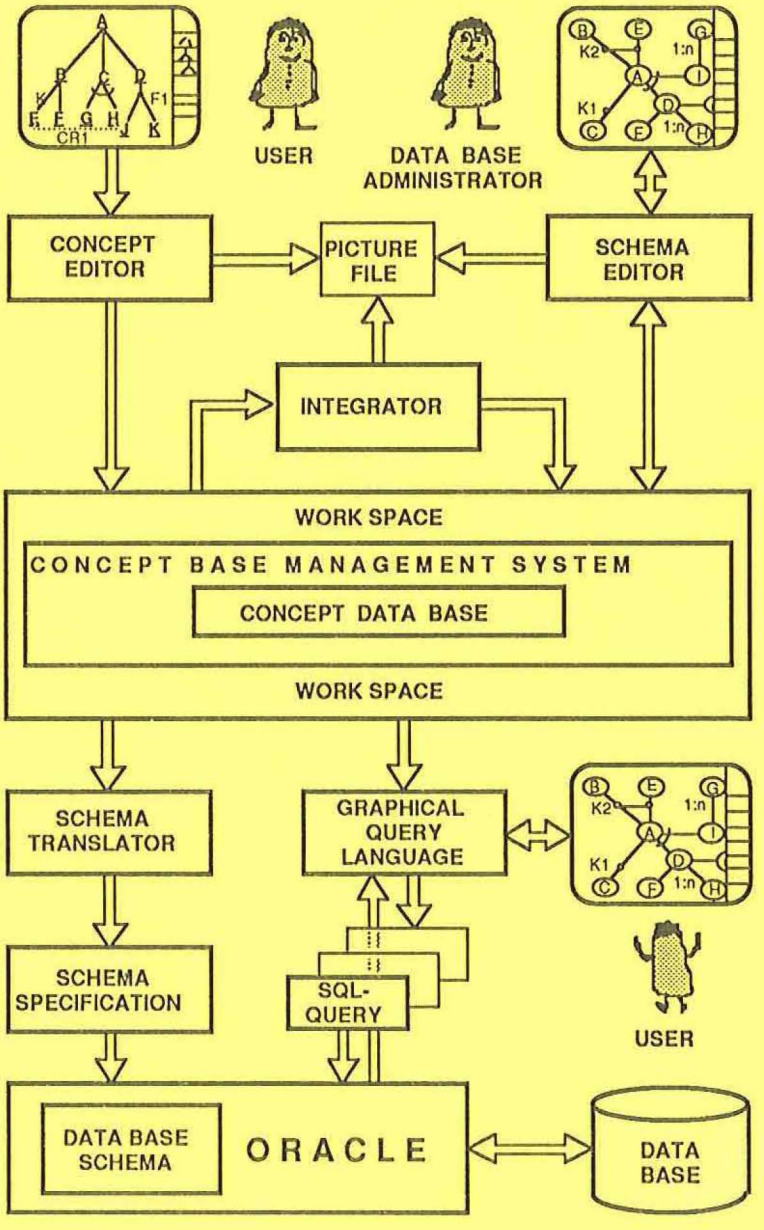

Figure 2. The architecture of the COMIC system

The concept data base is used by components 1 - 5 as a dynamic design data base, as a concept definition library, as a store for the conceptual schema and as a dictionary needed by the system for the evaluation of CQL- queries.

3.2. Development of the conceptual schema

3.2.1. Motivation for using the conceptual schema

The goal of conceptual modelling is to clearly recognize and describe the set of concepts of the UoD regarded as important by users, and to construct a conceptual description of the UoD as the users want to see it by using the set of well defined concepts. On the other hand, the goal is to establish a firm foundation for the specification of the meaning of the data to be stored in the data base, and for the specification of relationships between these data.

3.2.2. Concepts and concept structures

In the following a concept is defined to be an independently identifiable formal construct with an internal structure, and consisting of structured semantic information, i.e. knowledge [17,18,22]. Concept structure diagrams describe important concepts of the working environment of users. They may describe data requirements, reports, data structures, objects, events, processes or relationships of the UoD. They can be as large and complex as needed to explain the essential content of concepts. Concepts are not classified into entities, attributes or relationships - we just have concepts which are regarded as basic epistemological components of human knowledge.

The working hypothesis used in this work is that the basic epistemological relation between concepts is the relation of intensional containment, and the methodology and notations used in conceptual modelling should be based on this relation. The relation of intensional containment is a binary relation defined within the set of concepts [26]. An intuitive explication can be given by saying that concept **a** contains intensionally concept **b** if the knowledge that forms concept **a** contains the knowledge that forms concept **b**. Note that we are talking about the knowledge required to recognize phenomena **a** and **b** in the UoD, not the way how the definitions of these concepts are constructed. The relation of intensional containment gives a possibility to develop all generally known modelling constructs in a systematic and consistent way, together with some novel modelling constructs [18].

A concept structure is a construct which consists of a defined concept (definiendum) and of its definition hierarchy, and in which the properties of the definiendum derive from the properties of basic concepts [18]. The graphical layout is meaningful in a concept structure diagram. The definiendum is on top of the hierarchy and concepts defining it are on the next or lower levels of the hierarchy. Structurally a defined concept is always a directed acyclic graph based on the relation of intensional containment. The definiendum contains intensionally other concepts which are some of its characteristics. A characteristic of a concept can be another concept contained in the defined concept, a relationship of intensional containment, or an additional inscription contained in the definition hierarchy. An inscription can be a condition, a constraint, a conditional constraint, or the definition of an identifying property or the limitation of its scope. The defining concepts are in turn defined by other concepts until the level of undefined, basic concepts is reached. For an occurrence of a concept to exist, occurrences of all its defining concepts must exist, but not vice versa.

The definition of a concept is represented as a diagram that associates the definiendum with the set of defining concepts. The type of the diagram together with some attached inscriptions specify how the properties of the definiendum are to be derived from the properties of defining concepts. The most commonly used types of definition are:

1. Aggregation, in which a concept is defined as a collection of its characteristics. There are several different types of aggregation.

2. Generalization, in which a concept is defined as a collection of common characteristics of its defining concepts. A generalization can be either an unconstrained generalization or a constrained generalization. Constraining can be done either implicitly or explicitly.
3. Value transformation, in which a concept is 'defined' by specifying how the value representing it can be derived from values representing defining concepts.

It is important to note that a definition and the corresponding concept are different things. Structurally a concept is always an aggregation, i.e., a collection of its characteristics. A definition is a rule or instruction which specifies how the knowledge forming the defined concept is to be constructed from the knowledge given in the definition itself and in the defining concepts. Concept structures in COMIC are described by using a graphical language CONCEPT D/D [18,19,22]. Some extremely simple examples of concept structures are shown in the following.

In **aggregation** a definiendum is constructed by composing two or more characteristics together and by assigning a name to the resulting construct. At least one of the characteristics must be a concept. In the definition all the relations between contained concepts must be specified, as well as all inscriptions attached to them. The general pattern of aggregation is in Figure 3. For simplicity the symbols for exclusive-or (classification) and for identifiers have been omitted.

A qualifier can be a condition list, a constraint list or a conditional constraint. A CR-qualifier can be a constraint list or a conditional constraint. The details of conditions and constraints are not described. To each concept name a list of attributes can be attached. An attribute can be e.g. a value set, occurrence constraint or a semantic rule.

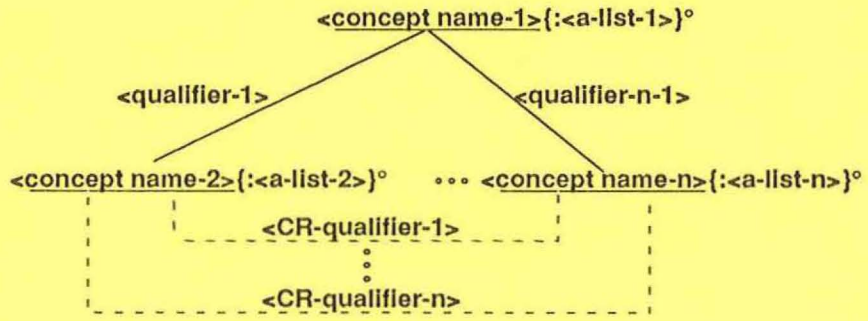

Figure 3. Pattern of aggregations.

Figure 3 indicates how conditions, constraints, and conditional constraints can be attached to a definition. They may concern values of occurrences, value sets, extensions of concepts, equality or differences between occurrences and occurrence time or occurrence conditions of concepts. They can be written either using a natural language or a formal language. Examples of simple aggregations are in Figure 4.

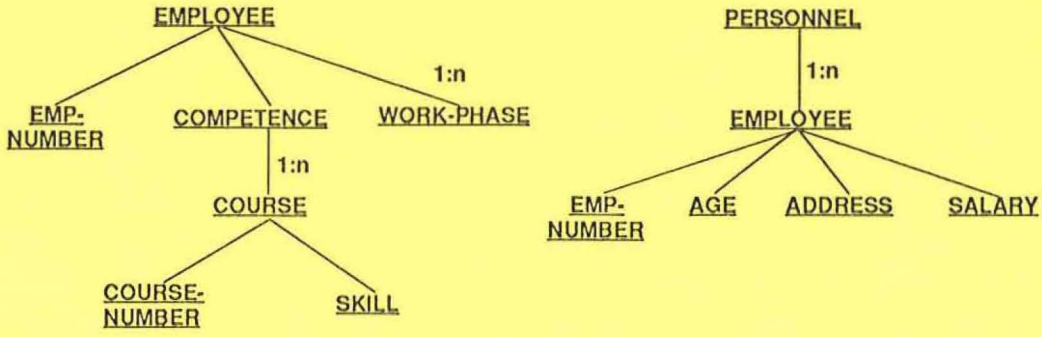

Figure 4. Examples of definitions by aggregation.

The concept EMPLOYEE has been defined with three concepts (EMPNUMBER, COMPETENCE, WORK-PHASE) and one cardinality constraint (1:n). Concept COMPETENCE contains concept COURSE and one cardinality constraint. The definition of PERSONNEL indicates that personnel of a company consists of a set of employees. This kind of structure, sometimes called an association in the literature is only a special case of aggregation in CONCEPT D.

An occurrence of the concept defined by aggregation exists if occurrences of all defining concepts exist and all constraints in the definition are true. This implements in a natural way different kinds of subclass and specialization abstractions used in some other modelling techniques. They are represented by ordinary concept definitions in CONCEPT D.

For a certain occurrence of the definiendum, the occurrence of a defining concept can be missing if the condition attached to the corresponding intensional containment line is false. By using this mechanism different versions of the same concept can be defined easily.

In generalization a definiendum is constructed by assigning to it either all common characteristics of defining concepts, or some subset of all common characteristics of defining concepts. The general graphic pattern of generalization is in Figure 5.

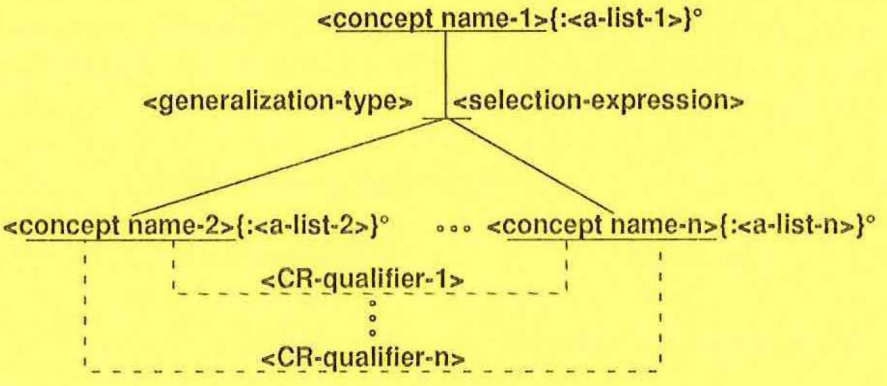

Figure 5. Pattern of generalizations.

Definition by generalization differs from the definition by aggregation in one essential aspect. The definition must be evaluated before the intension of the definiendum is revealed. The resulting concept is structurally just an ordinary concept. This fact explains why it is important to differentiate between a concept and its definition.

In the evaluation three alternatives exist. These alternatives are expressed by generalization type (**G** | **GE** | **GI**) in the general pattern. In unconstrained generalization (**G**) the definiendum will contain all common characteristics of its defining concepts. In explicit generalization (**GE**) the selection expression is applied to the result of unconstrained generalization and all characteristics for which it evaluates **true** are accepted as characteristics of the definiendum. During the evaluation of the definition first all common characteristics are recognized, then the chatacteristics not selected are removed and finally the implications of removals are worked out. The last step has to be done because the removal of a characteristic may cause some structural inconcistencies in the definiendum, which must then be eliminated. In implicit generalization (**GI**) the selection expression is applied to the result of unconstrained generalization and all characteristics for which it evaluates **false** are accepted as characteristics of the definiendum. In this case the definiendum contains those characteristics which are common to all defining concepts and are not mentioned in the selection expression. Also in this case some implications must be taken into account. A selection expression can also be a list of concept names.

In **value transformation** a definiendum is constructed by specifying how the value representing the definiendum is derived from the values representing the defining concepts. In the definition also some conditions, constraints and conditional constraints can be used. The general graphic pattern of value transformation is in Figure 6.

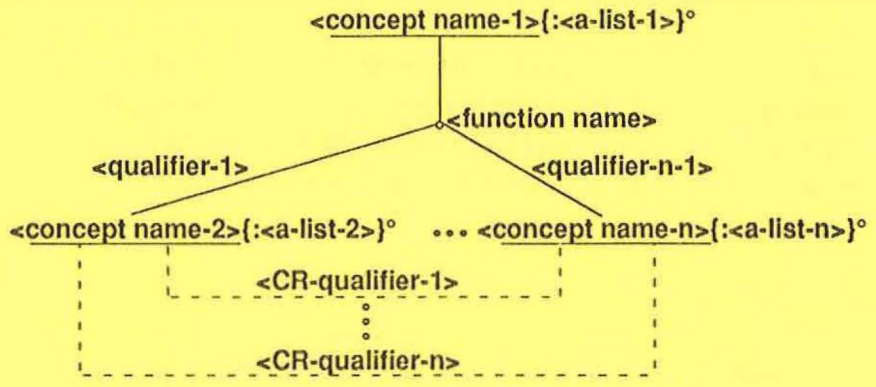

Figure 6. Pattern of value transformations.

For example, the value representing concept SALARY is derived from the values representing concepts HOURS and WAGE according to salary function specification. In value transformation cardinality specifications can be used to indicate the number of occurrences of each defining concept required by the function. A condition specification can be used to indicate that a defining concept is a optional argument for the function.

Concept structures are drafted on the screen of a graphical workstation under the control of the concept editor. The user employs a mouse and menus for constructing concept structures. There is a menu option for each different type of concept definition.

Concepts stored in the concept base can be retrieved and edited. As a result of the concept development phase there will be in the concept data base a library of concept definitions. From this fact an idea of standard concept definitions is developed [15]. All core concepts of an enterprise can be standardized and users can then use restricted versions of standard definitions. Standard concepts can be developed to completely follow the legislation. They contain all knowledge which must be observed in different systems. By using them the conceptual analysis of information systems can be simplified and the number of errors can be reduced.

3.2.3. Conceptual schema and the schema editor

A conceptual schema (CS) is developed from concept structures. A conceptual schema is a formal construct composed of concepts, intensional relationships between concepts, and of constraints and conditions between concepts. A CS not only describes the entities and their attributes and relationships in the UoD, but it defines the whole system of concepts that pertain to the UoD.

A conceptual schema can be seen as a three dimensional construct which consists of concept definition hierarchies. Structurally it is a directed acyclic graph based on the relation of intensional containment. In fact, it has exactly the same formal structure as a concept structure. They differ only in their graphical representation.

A conceptual schema may contain several hundred concept structures. In the graphical representation a concept structure is viewed 'from the side'. This representation supports strongly the idea of hierarchical

definition and helps the user to analyze the structure of the concept being defined. For a conceptual schema (and even for an exceptionally large concept structure) this representation is impossible because the diagram would contain a lot of lines that are almost horizontal and cross many other lines. The representation space must be used more effectively. The solution used in COMIC is to apply a three dimensional representation, which is believed to correspond closely to the natural way of organizing human knowledge. The evidence collected so far seems to support this hypothesis at least with some people.

The graphical representation of a CS consists of nodes representing concepts and of arcs representing relationships between concepts [20,22]. There are two types of arcs: arrows representing the relation of intensional containment, and dotted arcs representing various types of constraints. The head of an arrow points to the defined concept. To each arc additional inscriptions can be attached. An example is in figure 7.

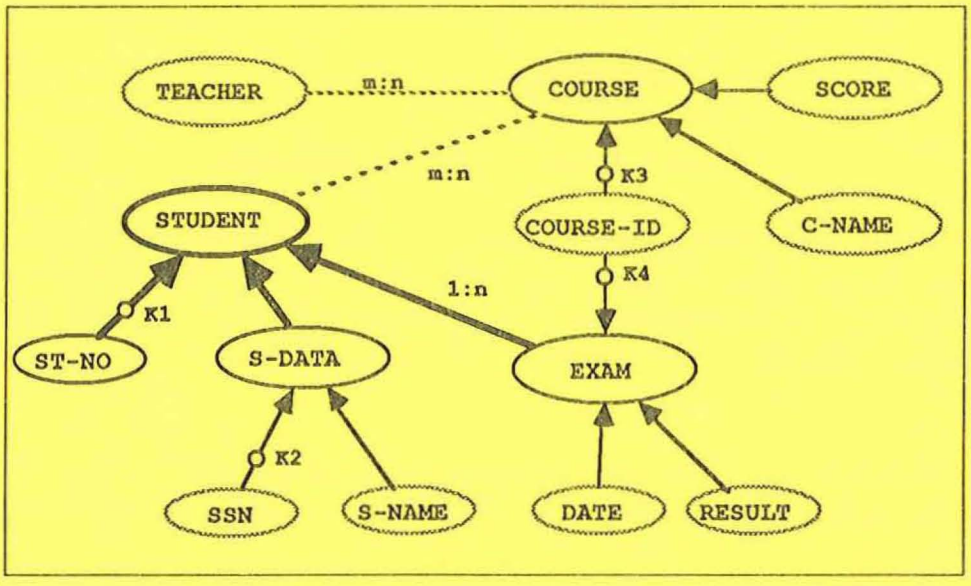

Figure 7. A sample conceptual schema.

The conceptual schema editor is a tool for inspection and modification of the CS [32]. The CS or a part of it is shown on the screen from 'the top down through the levels'. The levels are stacked on each other in such a way that the highest level of the CS is on top of the stack and the lower levels under it sequenced according to the number of the level. The levels are semitransparent in such a way that several levels can be viewed at a time without being confused by the whole complexity of the CS. This feature creates an illusion of three-dimensionality which supports the understanding of the structure of the CS.

The user has a possibility to 'navigate' in the CS. The schema can be scrolled up and down, left and right, and new levels can be added on the top or on the bottom of the screen stack, or they can be removed. In this way the information on the screen is focused very effectively and the amount of information visible to the user is considerably increased. Because some parts of the schema can be off the screen it must be possible to project a miniaturized view of the schema on the screen in order to make the whole schema visible. A concept structure can be transformed to the corresponding conceptual schema by using the conceptual schema editor. The resulting schema will be stored in the concept base from which it can be retrieved for inspection or for integration with some other conceptual schema.

3.2.4. Integration of local conceptual schemata

The set of local conceptual schemata or user views produced from separate concept structures must be integrated to make up a global conceptual schema. In the COMIC system the integration is quite a difficult

task because of the rich modelling language. The integration is an interactive process performed by using the integration module, which works on two conceptual schemata at a time. The designer sees both schemata on the screen and gives commands on how the integrated conceptual schema is to be constructed. The design of the integration module is not as yet completed, but the integration strategy will be a binary ladder strategy, in which the set of conceptual schemata is ordered according to their relative importance or weight and in every integration cycle the heaviest local schema is integrated into the partially integrated global conceptual schema until the whole set is exhausted. Detailed integration rules are given in [27].

3.3. Use of the conceptual schema in COMIC

3.3.1. Transformation of CONCEPT D conceptual schema into a relational database schema

A conceptual schema must be transformed to a relational data base schema before the data base is implemented. The transformation process takes advantage of structural information inherent in the conceptual schema. While representing an application with a set of concepts, the CONCEPT D conceptual schema also includes structural definitions of these concepts, i.e., concept structures. Concept structures and conceptual schemata contain implicitly both functional and multivalued dependencies [35], as well as inclusion dependencies, too [11]. The transformation produces the relational data base schema in 4NF with lossless join property. In certain cases the designer has a possibility to decide that 3NF is sufficient.

The transformation consists of two phases [34]. In the first phase a conceptual schema is decomposed into several partial conceptual schemata on the basis on those conditions, constraints and conditional constraints that deal with cardinalities of occurrences of concepts. The decomposition algorithm goes recursively through the conceptual schema. For every concept encountered it checks whether it contains any defining concepts, or is just a basic concept. In the case of derived concept, its definition type determines how the decomposition is performed on that level. An identifier hierarchy consisting of a sequence of identifiers on consecutive levels must be associated with every partial conceptual schema generated.

In the second phase partial conceptual schemata are transformed to relation schemes, which together constitute a relational database schema. In this schema every partial conceptual schema has a corresponding relation scheme. Basic concepts and possibly concepts defined by a transformation function are taken as attributes to relation schemes. Identifiers and identifier hierarchies are represented as key attributes. Finally it is checked that the resulting schema is nonredundant, that is, that any relation scheme is not already contained in some other relation schemes. The transformation process cannot be made completely automatic, because in some situations a decision of a user is needed.

The resulting relational database schema and a corresponding set of data description language statements for the relational DBMS is generated. In the present version of the COMIC system ORACLE and SQL are used. In the concept data base the mapping between the conceptual schema and the relational schema is constructed.

3.3.2. Conceptual query language CQL

The use of a graphical conceptual schema opens a possibility to develop a user-friendly query interface [21,24,25]. The formulation of a query should be as easy as possible, i.e. require no knowledge of the structure of the data base, or mental mappings between the knowledge of the user and the conceptual schema. The conceptual query language CQL attempts to meet this requirement. In the implementation design only those features of the language are taken into account which can be implemented by using SQL.

The goal in the design of CONCEPT D and CQL is that the structure of user's own internal conceptual model would be isomorfic with the conceptual schema on the screen as much as possible. Then the conceptual schema would reflect the mental structure of knowledge the user has in his mind. That would eliminate a complex structural transformation from the user using the conceptual schema and CQL. A conceptual schema supports the user in recalling and comprehending conceptual structure of the UoD and the data base.

The user of the CQL query facility will see a graphical conceptual schema on the screen of the workstation. The formulation of a query consists of selecting an object of interest (OI) from the schema and pointing at it with a cursor that can be moved with the mouse. Constraints can be imposed on the OI by use of the cursor, menus and the keyboard. There are several benefits to using a conceptual level graphical query facility [21]:

1. The user has to have no knowledge of the structure or technical features of the data base.
2. The query language has only very few syntactic rules.
3. The use of a conceptual schema for formulating queries helps the user come to a better understanding of the subject matter. The user can see the three-dimensional conceptual schema around the OI and study it in detail easily.

An example of the screen is in figure 8. The menu is on the right, the conceptual schema is on the left, and the text line for condition specifications is below the schema. The schema contains three hierarchical levels. STUDENT is on the highest level and SCORE and some other concepts on the lowest level.

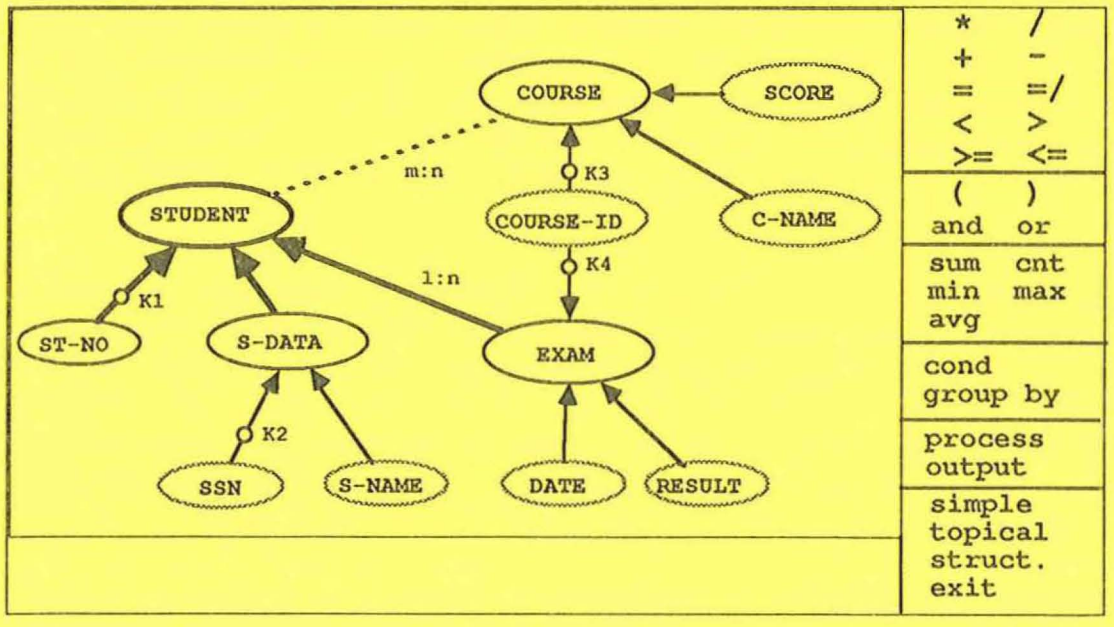

Figure 8. User interface for making CQL queries [24].

The class of queries can be divided in simple queries and queries containing complex relationships. In simple queries the OI is just one concept and the data representing its occurrences.

In simple queries the user may want a list of all occurrences of the OI, or just a subset defined by some selection criteria. A selection criterion can be e.g. a given value of an identifier of the selected concept, some value or a range of values representing a concept contained in the selected concept, or a relationship the selected concept has with some other concept. Boolean expressions can be used in constraints.

For example, if a list of all students is wanted, one points at STUDENT and selects the process command from the menu. The system finds out that STUDENT contains other concepts (ST-NO, S-DATA, EXAM) which in turn contain other concepts. The output list has six columns as follows:

STUDENT(ST-NO,SSN,S-NAME,(COURSE-ID,DATE,RESULT))

ST-NO and SSN both identify occurrences of students and COURSE-ID identifies occurrences of courses. Because of the 1:n relationship between STUDENT and EXAM, the occurrences of examinations are identified with a pair ST-NO,COURSE-ID (or with SSN,COURSE-ID).

If some subset of students is to be selected, then the user has to give qualifications. For example, to select a student whose ST-NO is 30504, touch STUDENT, select **cond**, ST-NO, and write '= 30504' and give a process command. The result will be a list like the following.

STUDENT	(ST-NO,	SSN,	S-NAME,	COURSE-ID,	DATE,	RESULT)
	30504	230961X	MIKKO LAHTI	A2.1	6.5.88	2
	30504	230961X	MIKKO LAHTI	A2.3	4.9.88	1
	30504	230961X	MIKKO LAHTI	A2.4	7.9.88	3

If several students are to be to selected, then a list of their student numbers is to be supplied.

To make several qualifications on different concepts contained in the same superordinated concept, the user has first to touch the superordinated concept, and then repeat the qualification sequence described above for each contained concept. Accordingly, to select only some contained concepts for the output, one has to touch them all after the superordinated concept.

In some cases a user is interested in two or more concepts which belong to different, but interrelated concept structures. The relationship between them can be of three different types, or it can be a combination of these types.

1. The related concepts share at least one concept which is contained in all of them.
2. The related concepts are all contained in the same concept, i.e., they share a common identifier from the higher level.
3. The related concepts are connected through a chain of constraints of various types.

The evaluation of a query containing complex relationships can result in two basically different results. One alternative is that there is exactly one path of relationships between the related concepts. The system has to find the path, analyze it, generate the corresponding set of SQL data manipulation commands, and finally process the data base in order to find out whether there is data corresponding to that path.

The other alternative is that there are several relationships between the related concepts. The system can find them all and the user has to choose the action in the next step, by proceeding in one of the following ways:

1. The user can tell the system that the data in all relationships should be listed (i.e. full search is required).
2. The user can tell the system that the automatic mode should be used. This means that the system must try to find the 'most plausible' of the relationships the user is interested in.
3. The user can tell the system that the manual mode is to be used. The user must himself select the relationship between the selected concepts and indicate this to the system by

touching the corresponding arcs in the schema.

A query can be qualified also in the case of complex relationships. The qualification is done similarly to that of a simple query. This alternative has not been implemented yet.

Similar approach is used for updates of the application data base [31]. A user selects the IO by pointing at it on the screen and possibly gives qualifications. The system analyses the OI, opens a small window under the concepts which must be updated and waits for the users response. User writes the data on each window and gives the process command. The system produces SQL commands and updates the data base.

4. SUMMARY

A new approach to conceptual modelling and a tool supporting the associated methodology are introduced. The approach is based on the idea that in conceptual modelling the concepts used to describe the UoD must first be constructed and a conceptual schema of the application shall be developed by using them. The construction of concepts requires that the internal structure of them be analyzed and described.

Concepts are defined graphically by using concept structure diagrams. They describe the internal structure of concepts. They are based on the use of the relationship of intensional containment. In a concept structure all knowledge about the concept can be given in an easily understandable form. Concept structures are manipulated by means of the concept editor.

By using the set of defined concepts a conceptual schema of the UoD is developed. The conceptual schema is a multilevel construct which is viewed from the top down on the screen of the workstation. The conceptual schema is manipulated by means of the schema editor. The user can navigate in the schema to all directions.

The set of local conceptual schemata must be integrated before the development of the data base schema. Integration is done with the schema integrator, which is based on the use of the ladder strategy. In the present version of the system the integrator has not been implemented, yet.

The schema translator is used to transform a conceptual schema into a corresponding relational data base schema. The schema specification can be given to a relational data base management system. In the present version of the system ORACLE is the DBMS used.

A special feature of the COMIC system is its conceptual query facility based on the conceptual query language CQL. A user can make queries by pointing at the conceptual schema visible on the screen of the workstation, with no need to know the structure of the data base. Updates of the application data base can also be made from the conceptual schema level.

The system is being implemented on APOLLO DN3000 workstations at the University of Tampere. The work started in January 1984. The last phase of the development ended at the end of 1988. Continuation of the work is being planned. Several possibilities for extensions to previous work have been recognized.

ACKNOWLEDGEMENT: I would like to express my thanks to many people who have contributed to the COMIC project either by giving comments to my own work or by developing parts of the implementation. Much of the work has been done in close cooperation and many solutions have evolved and been improved quite a lot in long discussions. Especially I would like to mention Ossi Numminen and Jyrki Nummenmaa who have been developing the schema editor. Sami Kari implemented the CQL, Raimo Mansikkaoja designed the rules for the integrator, and Arto Viitanen studied the possibility to use

COMIC in a distributed environment and implemented the current version of the concept data base. Jari Pösö developed an original method for schema translation. Jyrki Nummenmaa and Arto Viitanen developed the update function for CQL.

REFERENCES

[1]. G.Barber, P.DeJong and C.Hewitt, Semantic Support for Work in Organization. In: R.E.A.Mason (Ed.), Information Processing 83. North-Holland 1983.

[2]. C.Batini, M.Lenzerini and M.Moscarini, Views Integration. In: [25].

[3]. M.Brodie and J.Mylopoulos (Eds.), On Knowledge Base Management Systems. Springer-Verlag, 1986.

[4]. M.Brodie, J.Mylopoulos and J.W.Schmidt (Eds.), On Conceptual Modelling. Springer-Verlag,1984.

[5]. M.L.Brodie and S.N.Zilles (Eds.), Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. ACM SIGMOD Record, Vol.11, No.2, February 1981.

[6]. J.Bubenko,jr. On the Role of 'Understanding Models' in Conceptual Schema Design. In: Proc. of the 5th Int. Conf. on Very Large Data Bases. Rio de Janeiro, October 3-5, 1979.

[7]. J.Bubenko,jr. Information Modeling in the Context of System Development. In: S.H.Lavington (Ed.), Information Processing 80. North-Holland 1980.

[8]. J.Bubenko,jr. Information and Data Modelling: State of the Art and Research Directions. In: H.Kangassalo (Ed.), Second Scandinavian Research Seminar on Information Modelling and Data Base Management. Acta Universitatis Tamperensis, Ser B. Vol 19, Tampere, 1983.

[9]. M.Bunge, Scientific Research I, The Search for System. Springer-Verlag, Berlin, 1967.

[10]. M.Bunge, Treatise on Basic Philosophy, Vol. I, Semantics I: Sense and Reference. D.Reidel Publishing Company. Dordrecht, 1974.

[11]. M.Casanova,R.Fagin,C.H.Papadimitriou, Inclusion Dependencies and Their Interaction with Functional Dependencies. Journal of Computer and System Sciences 28, 29-59 (1984).

[12]. S.Ceri (Ed.), Methodology and Tools for Data Base Design. North-Holland, , 1983.

[13]. P.Chen (Ed.), Entity-Relationship Approach to Systems Analysis and Design. North-Holland 1980.

[14]. E.F.Codd, Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, Vol.4, No.4, December 1979.

[15]. G.Di Battista, H.Kangassalo and R.Tamassia, Definition Libraries for Conceptual Modelling. In C.Batini, (Ed) 7th International Conference on Entity-Relationship Approach. (Rome, Italy, November 16-18), North-Holland 1988.

[16]. R.Elmasri and S.Navathe, Object Integration in Logical Database Design. In: International Conference on Data Engineering. Los Angeles, April 24-27, 1984.

[17]. H.Kangassalo, On the Concept of Concept in a Conceptual Schema. In: H.Kangassalo (Ed.), First Scandinavian Research Seminar on Information Modelling and Data Base Management. Acta Universitatis Tamperensis, Ser.B, Vol.17, Tampere 1982.

[18]. H.Kangassalo, CONCEPT D - A Graphical Formalism for Representing Concept Structures. In: H.Kangassalo, (Ed.), Second Scandinavian Research Seminar on Information Modelling and Data Base Management. Acta Universitatis Tamperensis. Ser.B, Vol.19, Tampere 1983.

[19]. H.Kangassalo, CONCEPT D/D - A Technique for Graphical Description of Concept Structures. University of Tampere, Department of Mathematical Sciences, October 1984, (Unpublished working paper) (In Finnish, English version is in preparation). 73 pages.

[20]. H.Kangassalo, CONCEPT D/CS - A Technique for Graphical Description of Conceptual Schemata. University of Tampere, Department of Mathematical Sciences, October 1984. (Unpublished working paper), (In Finnish, English version is in preparation).92 pages.

[21]. H.Kangassalo, A Definitional Conceptual Schema as a Query Interface of the Information System.

(Third Scandinavian Research Seminar on Information Modelling and Data Base Management, Finland, June 1985), In: H.Kangassalo (Ed.), Information Modelling and Data Base Management. Lecture Notes in Computer Science. Springer-Verlag, Berlin, (to appear).

[22]. H.Kangassalo, CONCEPT D: A Graphical Language for Conceptual Modelling and Data Base Use. Invited paper, IEEE 1988 Workshop on Visual Languages. (Pittsburgh, USA,October 10 - 12), IEEE, New York, 1988.

[23]. H.Kangassalo and P.Aalto, Experiences on User Parti- cipation in the Development of a Conceptual Schema by Using a Concept Structure Interface. In: B.Shackel (Ed.), Human-Computer Interaction INTERACT-'84. North-Holland, 1985.

[24]. S.Kari, A Plan for the Conceptual Query Language. MS Thesis. University of Tampere, Department Of Mathematical Sciences. November 1986. (In Finnish).

[25]. S.Kari, H.Kangassalo and J.Pösö, CQL - Conceptual Query Language: A Visual User Interface to Application Data Bases. Proceedings of the Joint Scandinavian-Japanese Seminar on Information Modelling and Knowledge Bases. Ellivuori, Finland, June 6 - 10, 1988. Acta Universitatis Tamperensis, Ser.B, (to appear).

[26]. R.Kauppi, Einführung in die Theorie der Begriffssysteme. Acta Universitatis Tamperensis, Ser.A, Vol.15, Tampereen Yliopisto, Tampere, 1967.

[27]. R.Mansikkaoja, Integration of CONCEPT D Conceptual Schemata. MS Thesis, University of Tampere, Department of Computer Science, April 1987, (In Finnish).

[28]. R.Meersman and G.M.Nijssen, From Data Bases to Knowledge Bases. In: Infotech State of the Art Review, London, November 1983.

[29]. T.Moto-Oka (Ed.), Fifth Generation Computer Systems. North-Holland, , 1982.

[30]. S.Navathe, R.Elmasri and J.Larson, Integrating User View in Database Design. Computer, January, 1986.

[31]. J.Nummenmaa and A.Viitanen, Data Base Updates from Conceptual Level. Proceedings of the Joint Scandinavian- Japanese Seminar on Information Modelling and Knowledge Bases. Ellivuori, Finland, June 6 - 10, 1988. Acta Universitatis Tamperensis, Ser.B,(to appear).

[32]. O.Numminen and J.Nummenmaa, Graphic Editors for Knowledge Acquisition and Conceptual Schema Design. Proceedings of the Joint Scandinavian-Japanese Seminar on Information Modelling and Knowledge Bases. Ellivuori, Finland, June 6 - 10, 1988. Acta Universitatis Tamperensis, Ser.B,(to appear).

[33]. T.W.Olle, H.G.Sol and A.A.Verrijn-Stuart (Eds.), Information Systems Design Methodologies: A Comparative Review. North-Holland, , 1982.

[34]. J.Pösö, Translation of COMIC Conceptual Schema into the Relational Database Schema. University of Tampere. Department of Mathematical Sciences. Report C44, October 1986.

[35]. J.D.Ullman, Principles of Database Systems. Pitman, London 1980.

[36]. J.van Griethuysen, Concepts and Terminology for the Conceptual Schema and the Information Base. ISO/TC97/SC5/WG3 -report. Available from ANSI under publication number ISO/TC97/SC5-N695.

* The work described in this paper has been supported by the Academy of Finland, the Center for Technological Development (TEKES) and the University of Tampere.