An Expert System for Semantic and Relational Database Design

Mokrane Bouzeghoub

Laboratoire MASI, Université P. et M. Curie, Centre de Versailles

45, avenue des Etats-Unis 78000 Versailles, France

Email: mob@litp.inria.fr

Abstact :

Database design has become an art that requires a high level of competence. The database design process is characterized by a certain indetermination in the way of choosing data structures and constraints. Several different schemas may describe the same reality. The design process is also characterized by an intuitive and empirical methodology; consequently, the quality of the schema obtained is heavily dependent on the database administrator's experience and insight in the database design. The development of increasingly sophisticated and efficient relational Data Base Management Systems has emphasized the need for increasingly complex information systems. It is therefore no longer possible to envision database design carried out without a computer aid. The expert system approach, described in this paper, is more adapted to the difficult problem of database modeling. It integrates algorithms, heuristics and practical know-how, and proposes an interactive methodology which is based on abstraction levels, user friendly interfaces and a modular knowledge base. This approach was first implemented by a prototype which was followed by a commercial product named SECSI® (Système Expert en Conception de Systèmes d'information).

Keywords :

Information Systems, Database Design, Semantic Data Models, Relational Model, Expert Systems, Computer Aided Software Engineering.

® SECSI is a registrated mark of INFOSYS.

1. Introduction

Relational data bases are nowadays the preferential tools for memorization and restitution of large amounts of data. This is especially true because of the great performance of the systems (DBMS). However, using the relational model as a conceptual design tool was somewhat controversial [KENT 79]. Indeed relational concepts are neither simple to use nor sufficient to capture the semantics of the user's application. At the conceptual level, new models are necessary to capture the semantics of the real world with preciseness and naturalness. Semantic data models seem to be more suitable for this objective [SMIT 77], [HAMM 81], [MYLO 80], [HULL 87], [PECK 88].

Even with these new models, database design remains a lengthy and tedious process without computer aids. Many computer tools have already been built, but they are far from solving all the data base design problems.

We have proposed a new approach based on techniques used in artificial intelligence [BOUZ 83]. This approach aims to combine various categories of knowledge coming both from relational theory, semantic data models, artificial intelligence and software engineering. An expert system product for conceptual and logical design has been built to apply the main ideas of this approach and to point to specific problems [BOUZ 85].

The SECSI system is developed in Prolog language, under PC-MS/DOS environment and UNIX workstation environment (particularly SUN workstations). Several other environments are envisioned in the nearest future. The product was commercially available since june 1988, and more than fifthy installations exist today. An english version will be available too.

The product SECSI can be considered as a central part of a software engineering environment. It was one of the first tools of the CASE technology. SECSI can be used as well as by expert designers for data modeling or schema validation, and by novice users and students in order to learn database design and relational technology.

After a brief outline, in section 2, of the data base design problems in both traditional and new databases, section 3 presents an overview of the existing tools. Section 4 characterizes the expert system approach for database design. Sections 5 and 6 detail the expert system approach, its objectives, its structure and the knowledge involved. Section 7 highlights how a deductive approach can efficiently contribute to the database conceptual design. Finally, section 8 concludes by pointing to the future developments.

2. Database Design Problems

Data base design is a difficult task. Since the begining of the database era, many design problems have been pointed. In addition to these problems, the introduction of knowledge bases, deductive databases and object oriented databases, impose new complex problems. The following sub-sections outline the different problems as well as in traditional databases and in future databases. In the following, we are not concerned by the physical organisation of data, nor by its optimization. We point only to those problems implied by the conceptual level and the logical level (i.e. semantic level).

2.1. Design problems in traditional databases.

In this section, we outline the main problems which must be solved in traditional databases.

*(1) **Size of the application :*** the difficulty of the design increases with the size of the application. Designing small data bases is quite easy, but structuring several hundred objects, relationships and constraints without computer aids is always a hard task, especially for people whose profession is not to be a "specialist" in database design.

*(2) **Relative perception of the real world and modeling choices :*** the real world may be reported in different ways with respect to the designers' perceptions: several database schemas may describe the same reality. The problem is to characterize the best schema with respect to formal rules, then to find a methodology to build and to choose this schema.

*(3) **Availability of information and the problem of restructuring :*** the designer cannot often get a detailed description of objects because of the lack of knowledge about the application. However the design must start with incomplete and imperfect knowledge. As the design is based on a fuzzy universe of discourse, new information may arise and may entail changes in the definition of classes, relationships and constraints.These changes must be integrated without reconsidering all thc modeling phases which were already done.

2.2. Design problems in future databases

Besides the traditional database problems, knowledge bases, deductive databases and object oriented databases have introduced the following problems [BROD 84], [GALL 84], [BROD 86].

(1) Representation of more complex objects: future databases will be able to represent more complex objects than flat data structures. The new objects may be rules, abstract data types, texts, graphics or images. Hence the corresponding representation models could be very difficult to learn and to use.

(2) Representation of more complex constraints : models have to express more complex constraints as general integrity constraints (state constraints) and dynamic constraints (transition constraints). For example, some logic based models allow to express most constraints as any first order formulas. Besides this problem of specification, the designer has to decide wether a set of integrity constraints is consistent, easy to check and to maintain.

(3) Representation of the dynamics of objects : traditional database design makes a strict separation between data and the behavioural of this data. The trends in the new models are (as well as in the object oriented languages) to integrate the dynamic aspect with the static aspect of the database [OODB 86]. This leads to a double reflection effort during the database design process.

This list of problems shows how complex and difficult the database design was and will be. Then, nowday, no useful methodology could be envisioned without computer design aid. That is why CASE technology becomes more and more necessary and urgent.

3. Overview of Existing Database Design Tools

To solve some of the traditional database problems, many tools have been proposed and commercialized during the last decade [DBEN 84], [BOUZ 86b], [BROD 87]. We distinguish three categories of tools. They are generally characterized by the methodology and the models which are supported.

The first category of tools consists of graphical tools which are generally based on semantic data models like the Entity-Relationship Model [CHEN 76], or the Semantic Hierarchy Model [SMIT 77]. The different graphical tools which are provided act as a word process; they permit to design, to store and to layout aesthetic diagrams, but they rarely help in the modeling choices nor in the consistency checking of the designed schemas. All the design choices are left to the database administrator (DBA). In the best case, the graphical tool offers a few facilities for syntactic control or, if integrated with a data dictionary, it insures the correspondance between the components of the schema and the terms of the dictionary. These tools are very interesting for their user friendly aspect and their help to a clean documentation, but they are far from being effective modeling tools as they do not make any design choice.

The second category of tools provides a set of algorithms which are generally built upon the Relational Model [CODD 70]. Such tools provide programs deriving a normalized relational schema from a set of attributes and dependencies [BERN 76] [BEER 79] [FAGIN 77]. This approach is one of the best formalized, it permits to characterize and to automaticaly design the best relational schema, with respect to redundancy and to update anomalies. Unfortunately, this approach ignores natural objects such as entities, relationships, generalization hierarchies and semantic integrity constraints. The only constraints which are handled are generaly functional dependencies and multivalued dependencies (all of them considered as semantic links). But the acquisition of these dependencies, in the context of the universel relation assunption, remains a combinatory problem which requires a detailed study of the semantics of all elementary relationships between attributes. However, if these tools are not suitable for the conceptual level, they could be strong components at the logical level.

The third category of tools consists of a set of interactive programs which can be considered as a combination of manual tools and algorithmic tools. They often call for CAD techniques [BROD 87]. Interactions between the users and the system are question-answering oriented or graphics language oriented. This approach is based on the idea that database design is partly an automated process and partly a human art. Because of the last reason, this category of tools is more realistic than the previous category of tools, and thus more interesting from the practical point of view. However if this approach does not always succeed, it is because computer aids are programmed once and for all; hence it is difficult to modify or to add design rules. Tools appear as black boxes in which you must believe and accept their results, or you don't believe, then you remake manually the design. But this remains an interesting approach if we combine it with the new

developments in knowledge engineering and deductive systems. That is what we have done with SECSI.

The computer design tool we have built, combines the advantages of these three catogories of tools and aims to do more by providing a unified methodology which combines theoretical knowledge (expressed by algorithms and rules) and practical know-how (expressed by heuristics and cooking recipes). The deductive approach fulfils the lacks of the first two categories of tools, and a modular knowledge base increases the evolution of the design techniques and allows the CASE system to evolve better than the third category of tools.

4. SECSI: a New Generation of Tools for the Database Designers

This section summarizes the different objectives of SECSI, highlights its different characteristics as an expert system and reports the different levels of expertise of its knowledge base.

4.1. Main Objectives of SECSI

SECSI has been conceived for answering many problems related to the development of database applications. Generally, design tools are devoted to a given specific task within the design process. SECSI aims to cover all the life cycle of this process, from the conceptual level to the physical level [BOUZ 86a]. These objectives are declared hereafter as a set of questions for which the desired system must give answers.

*(1) **Conceptual modeling** :* how to transform incomplete specifications of a problem, expressed in an ambiguous natural language, into a formal, complete and consistent conceptual schema ? How objects can be classified as entities, relationships, attributes or constraints ? Can the human designer be liberated from these different choices ?

*(2) **Physical design** :* how to generate an efficient data storing and retrieving structure, starting from the conceptual schema of a database ? How to take into account the different parameters that characterize : (i) the given application and its cost requirements, (ii) the software and hardware environment which will be used for development and implementation ?

*(3) **Schema restructuring** :* how to make sure that the database structure will evolve as the information system changes ? How to add new attributes, new relationships and new constraints without redesigning the whole database ? How to integrate several data base schemas that have been designed independently ?

*(4) **Database administration** :* how to develop and maintain a detailed and precise documentation on the complete life cycle of the database (from the design phase through the operational phase) ? What are the repercussions on the data and the already existing programs when the characteristics of the software or hardware environment are modified ?

Some of these objectives have already been reached (e.g. conceptual and logical design, some schema restructuring and some documentation). Others are in the prototyping phase (e.g. physical design, view integration).

More than just a tool for the design of database schemas, SECSI can also be used in many other ways as for example, a validation tool of manual design or as an educational tool in data modeling. Combined with a DBMS, SECSI can be considered as a powerfull prototyping environment of relational databases. SECSI is already used to design buiseness applications as well as technical applications like geographical databases.

4.1. Main Characteristics of an Expert System for Database Design

The general definition of an expert system states that it is a specialized software tool which solves a problem as well as a human expert can do [HAYE 84] . Practically, in our area, this definition may be interpreted by the following four concrete characteristics:

The first characteristic is ***to constitute a complete knowledge base*** including theoretical algorithms and rules, and experimental knowledge in the database design process.

The second characteristic is ***to provide an interactive methodological environment*** which accepts incomplete specifications, provides the same reasoning as a human expert, permits backtracking to any design step, uses a question-answering system and can infer over examples.

The third characteristic is ***to be an open system*** which can learn and integrate new theoretical and experimental rules. The system must also transfer its expertise through its use (via explanation of its design rules) and through justification of its results.

The fourth characteristic is ***to provide end-user friendly interfaces*** by offering a workstation which reproduces the main features of the manual design (i.e. a graphical interface to design the first rough sketch of the database schema, a natural language to facilitate the communication of the specifications, and a declarative language to specify high level assertions that could have ambiguous interpretation in the natural language or a complex representation in the graphical interface). To facilitate the interaction between the system and the user, other features like icons, menues and mouse must be used too.

4.2. The Different Levels of Expertise of a Case Tool in Database Design

The expertise of the system is divided into three categories:

*(1) **Theoretical knowledge*** : this knowledge coincides with the design concepts (models, rules) and the design methodology (advices, reasoning principles). Theoretical knowledge is composed of algorithms, rules and heuristics :

- Algorithms : some parts of the design process are well isolated and formalized, and have already been expressed by many efficient algorithms, such as normalization algorithms, cost evaluation of transactions and access paths optimization.
- Rules : correspond to some known or admitted expertise as in normalization process (Armstrong's inference rules), view integration rules, mappings rules between different models, and consistency enforcement rules.
- Heuristics : may be particular interpretations of the real world, assumptions in some value distributions, or simplification of the correlations between attributes or between constraints.

*(2) **Specific domain knowledge*** : for each application domain (e.g. insurance, banking, medicine, travels), there is some common terminology, managerial rules and skills which represent the specific know-how of the domain. This know-how can be represented, as well as we can fomalize it, either by general behavioural rules or by general predefined structures. This knowledge is stored in the system data dictionary and reused, when appropriate, during the design process.

*(3) **Specific application knowledge*** : within a specific application domain, there are some properties which characterize each application (e.g. reservation in a given travel agency). These properties are described by facts and rules which constitute the detailed description of a given application.

All the knowledge referred to above constitute the knowledge base of the expert system SECSI.

5. An Open System Architecture

SECSI architecture is characterised by its modularity which permits, thanks to the atomicity of its knowledge base, to add new design rules and new various interfaces. This section describes the components of this architecture.

5.1. The knowledge base

The knowledge base is composed of two parts: a rule base and a fact base.

The rule base is created and updated by the expert designer. This base contains general rules such as normalization rules, mapping rules; but also specific rules which can be system dependent or even application dependent. General design rules are grouped into the DESIGN module which is used by the methodological regulator and the explanator. This part of the knowledge constitutes the system shell, hence automatically delivered in the basic version of the system.

The fact base is designed by the database administrator. It contains compiled specifications describing a given application. This part is created and updated by the DBA. It corresponds to the problem which is submitted to the system.

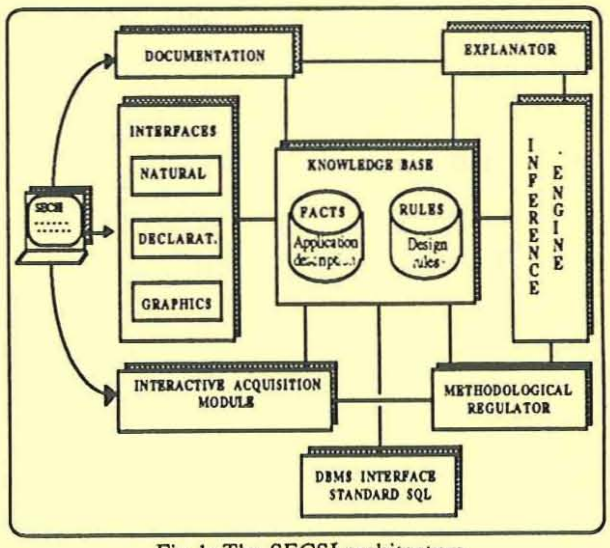


Fig.1: The SECSI architecture

To represent these two types of knowledge, we utilize two different representation models: a semantic network to represent facts and production rules to represent behavioural constraints and to specify the design rules. The semantic network is defined by a set of typed nodes (attributes, values, srtructured objects and possibly their instances) and a set of typed arcs representing relationships between nodes (aggregation, generalization, association and some constraint arcs) [BOUZ 84]. To enhance the semantics of this model, additional constraints are defined: domains, roles, keys, intersection of classes, cardinalities and functional dependencies. The figure 2 gives an illustration of the different concepts of this semantic data model.

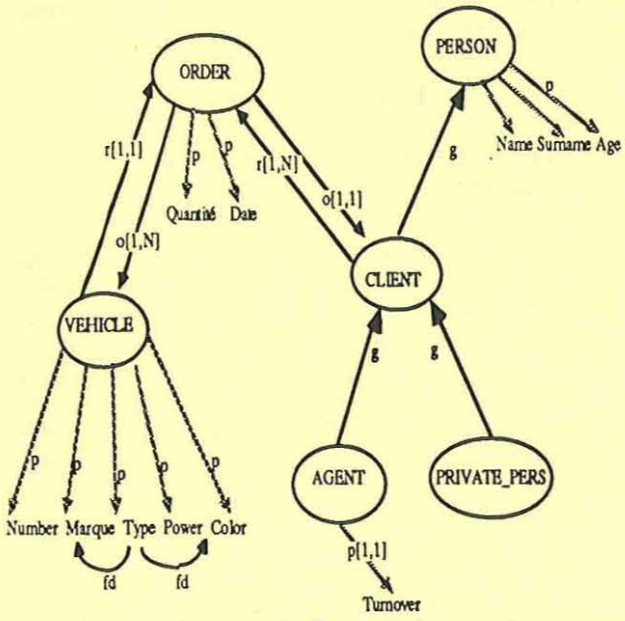


Fig.2: An example of a semantic network

Each semantic relationship is represented by a couple of binary arcs. Arcs **a** and **p** are called atomic aggregation arcs, arcs **r** and **o** are called molecular aggregation arcs, and arcs **g** and **s** are called generalization-specialization arcs. Cardinality constraints are expressed both over **a/p** arcs and **r/o** arcs. Other constraints like functional dependencies are portrayed by **fd** arcs too.

6.2. The inference engine

The inference engine is a program composed of the basic mechanisms which permit to manage the rules into the rule base and to apply them onto the fact base. This inference engine functions by an alternate use of backward-chaining and forward-chaining. The combination of these two principles is made necessary because each step of the used methodology consists of:

(1) Proving an hypothesis : for example, is a constraint derivable from a set of other given constraints ? This leads to the use of a theorem prover principle, based on a backward-chaining.

(2) Transforming specifications from one given form to another : for example, the successive mapping of the natural language specification to the relational model schema. This leads to the use of an inference engine, based on a forward chaining.

Besides this basic program, the inference engine provides two important modules: the methodology regulator and the results justificator. The first module allows the user to backtrack to any design step to redesign his database or to modify his specifications. The second module makes the system able to explain and justify its results.

6.3. The external interfaces

Describing the application in a comprehensible manner is an important problem in data base design. SECSI offers three types of interfaces : the specification interfaces, the acquisition interfaces and the interaction interfaces.

*(1) **The specification interfaces*** consists of three languages :

i. A natural language which accepts simple sentences, possibly composed of a conjunction of subjects,a verb and conjunction of complements. Its role is particuraly important for novice designers and for the easy communication of specifications. It also allows a rapid handling of the system without learning any formal language.

ii. A high level declarative language which helps to specify what the natural language cannot easily express. This language permits also to describe any database schema specified into any given data model

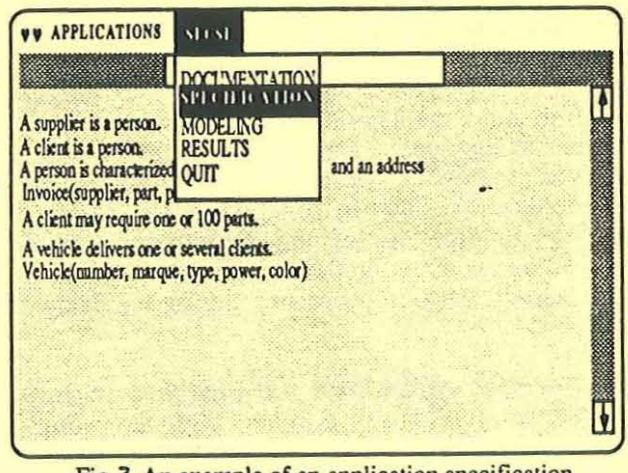


Fig.3 An example of an application specification

iii. A graphical interface which enhances the user friendly interaction. This interface is used either as an input facility or as a layout feature. As there is no common standard representation of graphics, SECSI provides graphics generator which permits to each user to have his own diagrams.

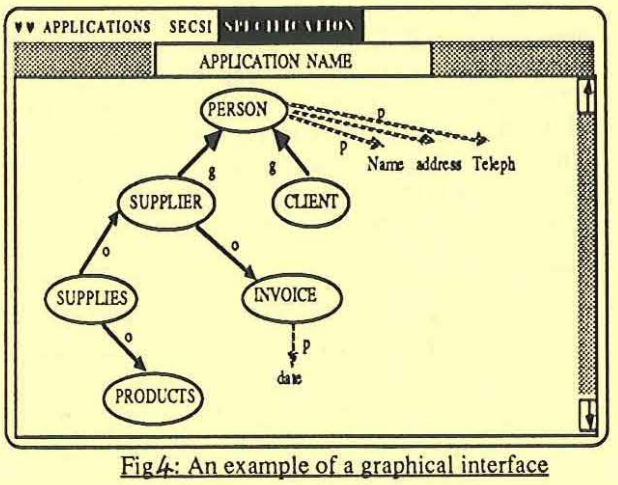


Fig 4: An example of a graphical interface

To be more flexible, the designer of a data base has the choice to describe his application in one or more of these languages, and then combining them in the same specification. An interactive parser generates a fact base from the description, this base being progressively enriched by the interactive acquisition module and transformed into a canonical semantic representation.

*(2) **The acquisition interface*** : the interactive acquisition assistance helps in completing the description of a problem through a question-answering system that automatically reminds the user what he might have forgotten to specify or what he has ignored in his description. This interactive acquisition process is based on deduction rules, heuristics, and the analysis of examples fed in by the user.

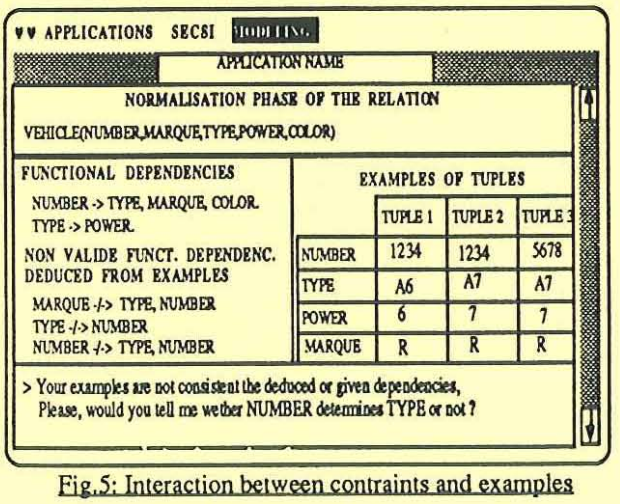


Fig.5: Interaction between contraints and examples

*(3) **The interaction interfaces*** distinguishe icons, menus, forms and documentation :

i. The menus visualize the authorized opera- tions on a given active window. According to the stages of design, only the permitted operations are visualized or accessible.

ii. Forms facilitate the capture of some constraints (e.g. functional depedencies, cardinalities), and permit to hihglight the default options generated by the system.

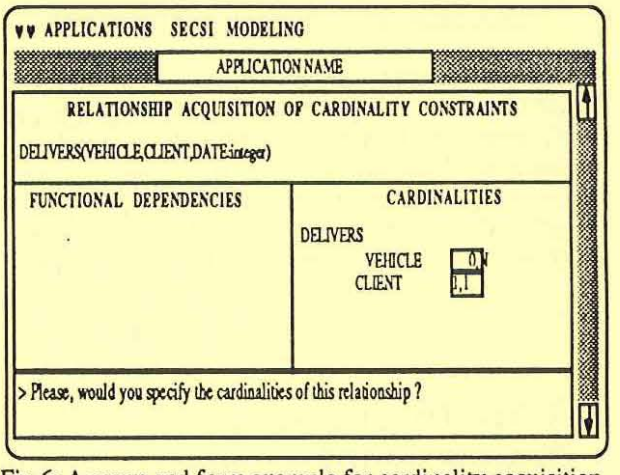


Fig.6: A menu and form example for cardinality acquisition

ii. The documentation is accessible at all levels whenever the user of the system requests it, during a session or independently. It consists of a concise user's manual, a synthesis of the methodology implemented by the system, and the definition of the main notions of data bases to which the system refers.

6.4. The Expected Results

When the design process is terminated, the system produces the following results:

(1) A set of basic relations in 4NF and the various keys of these relations (primary keys and candidate keys).

(2) A set of virtual relations (or views) and the definition of the corresponding relational queries which permit to derive them from the implemented database relations.

(3) A set of constraints like domain, referential and other general semantic constraints which have been generated by the mapping rules and the normalization process.

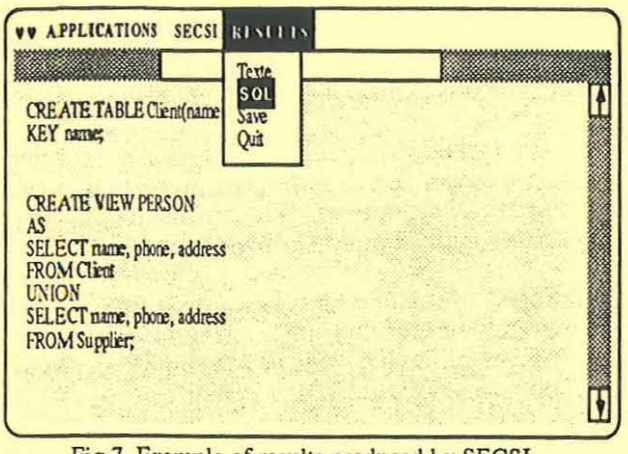


Fig.7. Example of results produced by SECSI

All the results can be obtained, as needed, either in standard SQL, as far as this language can do it, or in a more readable ad hoc language (i.e. a declarative language). Fot those results that cannot be generated in SQL (especially semantic integrity constraints), the declarative language is used, and it is of the responsibility of the database administrator to program these results in the corresponding language used for his application.

6.A Modular and Progressive Design Approach

The design approach adopted by SECSI and portrayed in figure 8 is based on three abstraction levels : the conceptual level, the logical level, and the physical level. To each level there correspond a specific model and a specific design process. The methodology proceeds by stepwise refinement, going from informal description of a given problem down to physical representation of this problem in terms of records and files. Starting from informal knowledge, three main phases successively produce: a formal specification, a conceptual schema, a logical relational schema and a physical schema.

The conceptual design phase generates from the external description of an application a sound conceptual schema stored as a semantic network with its associated constraints. This generation is performed while conversing with the end-user. The different views of the application given by the end-user(s) are mixed into one description after elimination of redundancy and resolution of conflicts. A verification step performs the validation of the application description in order to generate a consistent conceptual schema. In addition to the syntactic controls, this phase detects generalization cycles, recursive associations and objects which play several roles in the same relationship. The system tries to eliminate the possible inconsistencies using predifined inference rules or with the end-user's help.

The logical design phase transforms the conceptual schema into a fourth normal form relational schema with its associated integrity constraints and views. It performs the interactive acquisition of constraints and the choice of first normal form relations. Constraints such as intersection and union of classes, cardinalities of relationships and functional dependencies between attributes are captured. The first normal form relations are constructed by suppressing generalization hierarchies and removing multivalued attributes. Normalization is carried out using dependencies between attributes.

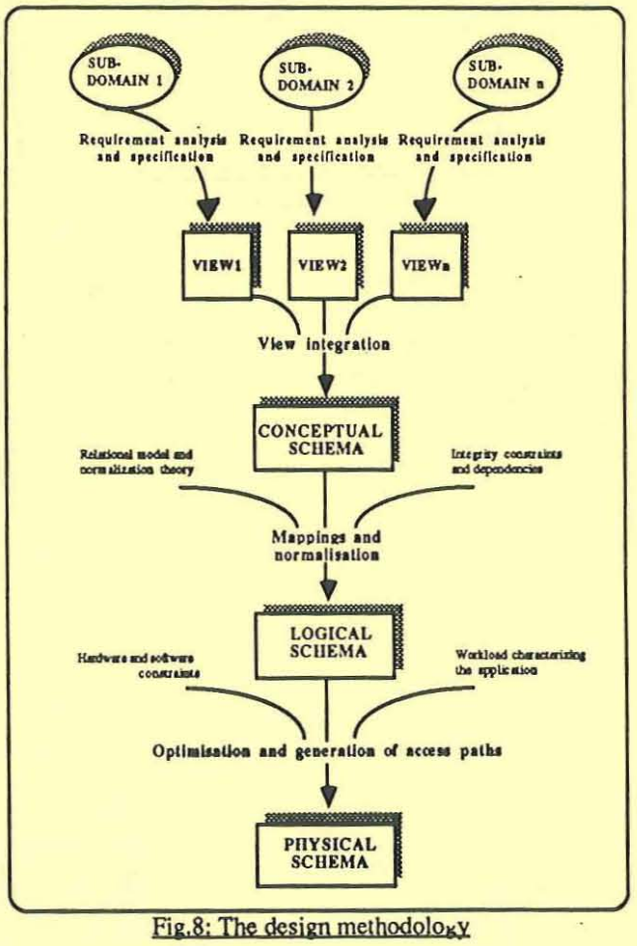


Fig.8: The design methodology

The physical design phase gives an optimized physical schema of the data base. This schema includes both a set of initial implemented relations and a set of indexes and formats. The choice of implemented relations and plausible attributes for indexing depends on the most important or most frequent queries which will be performed on the database. This choice needs some estimations depending on the DBMS used.

The design of a database is an iterative process. It implies numerous comings-and-goings between the universe of discourse and the expert system. SECSI provides this iteration at two levels :

(i) by permitting the interuption of a working session without losing the achieved task. The recover of the session can be done without any redesign of the last schema.

(ii) by authorizing the interuption of a dialogue to modify an assertion or to return to a preceding question.

Some resumptions are automatic at each of this levels without having to manually return to the current stage of modeling.

The design of a data base is not a fully automatic process : a permanent interaction with the designer allows the combining of algorithmic tasks with human decisions.

7. How a Deductive Approach Can Contribute to the Conceptual Design

This section highlights the contribution of expert system approach in the database design process. We particularly focus on that parts where the system has enough knowledge to infer from rules, heuristics and examples.

7.1. The Interpretation and the Control of Specifications

Besides the syntactical checks of sentences, one of the hard problem, when compiling user specifications, is to decide wether a term in a given sentence must be considered as an attribute, an entity, a relationship or an integrity constraint.

Sentences are not only interpreted as independent units, but also as a whole consistent specification whose interpretation is stronger than that of each sentence. So when a sentence is followed by another one, its interpretation could be modified because we get more information by reading two sentences than by reading only one. For example, from the following sentence: "*A product has a number, a unit-price, and its supplier*", we understand that there is an object named "product" and characterized by three attributes: "number", "unit-price", "supplier". But if we add a new sentence like: "*Each product supplier, described by his name and address, supplies one to ten parts*", we modify the previous interpretation by removing the attribute "supplier" and generating another object ("supplier") described by two attributes ("name" and "address") and a relationship ("supplies") between "product" and "supplier". In fact we got more information in the second sentence as we know the minimum and the maximum number of products supllied by a given supplier (i.e. cardinality constraints). But the second sentence introduces an additional complexity, related to the usage of synonyms ("product" and "parts"), that can be solved if a data dictionary is provided.

Another problem is related to objects that play several roles in the same relationship. For example, in the sentence: "*A person could be married to another person*", we do not know who is the wife and who is the husband. The interpretation must complete this sentence by acquiring the different roles from the user and modifying the previous sentence as follows: "*A person as a husband could be married to another person as a wife*". Every body who reads this sentence can infer more than a relationship between a person and a person; he can deduce that a person cannot be married with himself (because of the term "another" in the sentence). One can also deduce that there exist some persons who are not married.

Redundancy is a frequent problem in the specification phase. Some new sentences, altough they are true, do not augment the semantics of the application, as the new described facts can be deduced from the previous ones. For example, in the following description, the third sentence is redundant to the first two: "*A person has a name and and age. An employee is a person. An employee has a name and an age.*" Again, in the following example, there is a redundancy, but it is an underhand one: "*Employees and secretaries are persons. A secretary is an employee*". Indeed the second sentence makes a part of the first one redundant; as a secretary is an employee, it is not necessary to say that he or she is a person, this fact can automatically be deduced.

As the previous paragraphs show, the interpretation of a given specification is not only a syntactic process, but a very high level semantic process based on expert knowledge :

- a lexicon of terms which correspond to the usual abstrations (like aggregation and generalization), and to the different terms used in the vocabulary of the application,

- a set of semantic rules which permit to distinguish between atomic objects (attributes) and molecular objects (entities and relationships),

- a set of inference rules which capture the integrity constraints involved by the different sentences.

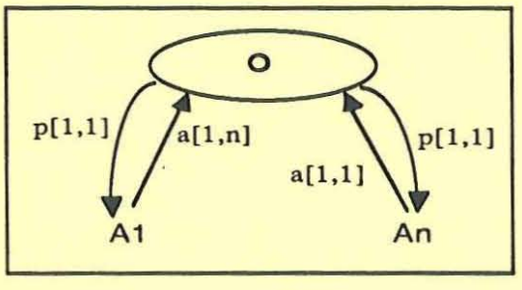
- a set of reinterpretation rules which are able to modify the interpretation of an objet to another object with respect to a given set of related sentences.

- a set of redundancy checking rules which avoid the assertions that can be deduced from other given facts.

7.2. The Acquisition of Constraints

One the most problem related to the constraints acquisition is the combinatory explosion (e.g. the acquisition of functional dependencies in the relational model). In this case, we cannot envision to ask the user all the possible questions. So we must use various means in order : (i) to limit the different combinations to consider and (ii) to avoid all questions for which an answer can be produced by using a set of inference rules.

As stated above, the problem of combinatory explosion arises for all constraints involving attributes (e.g. functional dependencies, cardinalities of attributes, keys). To illustrate this problem, we will consider the case of the acquisition of functional dependencies. Suppose we have an object type O(A1,...,An) where for each instance, we allow the attributes to be monovalued or multivalued. This can be represented by the following diagram:



As one can see, cardinalities specify in one hand the monovalued or multivalued attributes (p cardinalities), and in the other hand wether, for a given attribute value, we can associate one or many instances of the object (a cardinalities). There is a particular correlation between cardinalities and functional dependencies. Indeed, from a set of attributes with their cardinalities, we can deduce some functional dependencies. For example:
If Card(a(A1,O))=[1,1] and Card(p(O,A2))=[1,1] Then A1—>A2.
In the same way, from a combination of cardinalities and functional dependencies, one can derive new cardinalities and son on. For example:
If Card(p(O,A1))=[1,1] and A1—>A2 Then Card(p(O,A2))=[1,1].
Finally, as it is well-known, one can derive, using Armstrong's axioms, new functional dependencies from a given set of dependencies. For example:
If A1—>A2 and A2—>A3 Then A1—>A3.

This process can be done as far as necessary to generate the maximum facts. All generated facts are so much gained questions for the user. This deduction process is based on two assumptions:

(i) the number of combinations necessary to get cardinalities is lower that that of getting functional dependencies,

(ii) the acquisition of cardinalities is more natural than the acquisition of functional dependencies.

However, all functional dependencies are not implied by cardinalities. This derivation process can then be regarded as a heuristic to reduce the combinatory explosion in the search of functional dependencies. For all other dependencies which are not implied by cardinalities, one must use Armstrong's inference rules or, in the last, questions to the user.

Before requiring questions to get dependencies, one can go far, when possible, in the usage of heuristics. For example, we can make the assumption that, in most applications, there is no need to search for functional dependencies with more than four or five attributes in their left handside. This can contribute to considerably reduce the combinatory explosion.

Fianally, one can also use examples to reduce the combinatory explosion of functional dependencies. More precisely, examples could be used to generate some non valide functional dependencies; hence a set of unnecessary questions to ask to the designer. For example, from the following small relational extension,

NAME	AGE	ADDRESS
Dupond	24	Paris
Durand	27	Lyon
Durand	27	Dijon
Martin	24	Paris

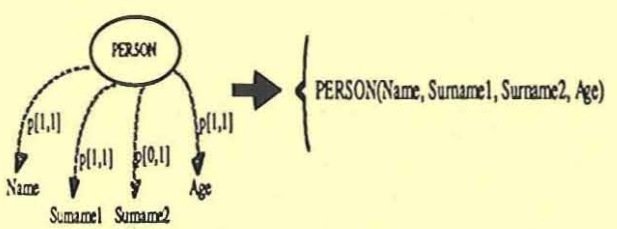
one can generate the following non valide dependencies:
NAME -/-> ADDRESS
AGE -/->ADDRESS
(AGE, ADDRESS) -/->NOM
(NAME,AGE) -/-> ADDRESS
We can notice that we cannot say anything about the NAME and AGE.

To summarize, the problems related to constraint acquisition could be solved by using different techniques as : (i) interaction rules between constraints, (ii) heuristics, (iii) examples and finally, when necessary, (iv) questions for the user.

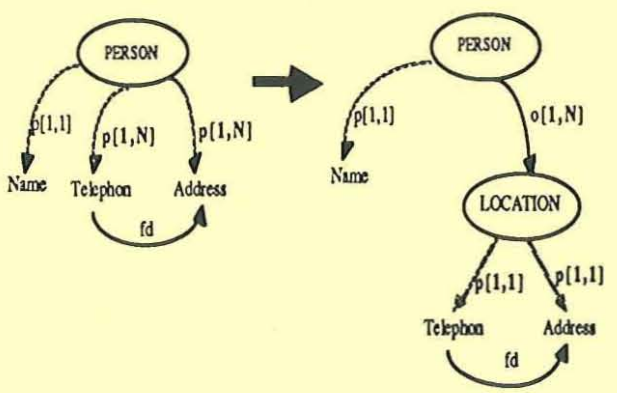
7.3.Transformation of the Semantic Model to the Relational Model

One of the main tasks of the system SECSI is to transform a conceptual schema expressed in a semantic model into a logical schema expressed in a relational model. As it is known, the problem in this case is to not loose the semantics between the two levels. For this objective, the system provides a set of transformation rules which conserves the semantics of the conceptual level at the logical level, by generating new objects, semantic integrity constraints and views (queries).

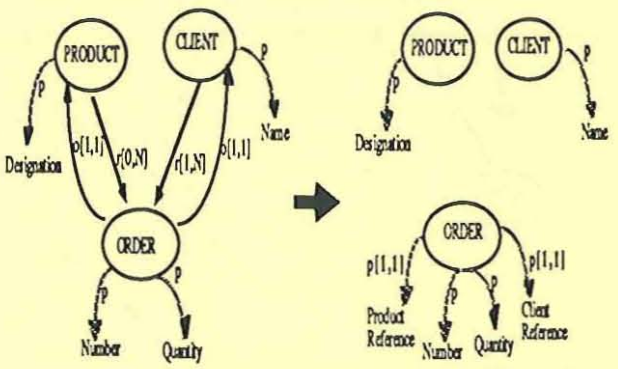
One simple transformation rule is to represent any molecular objet with monovalued attributes (cardinalities of **p** arcs equal [1,1] or [0,1]) of the semantic model by first normal form relation in the relational model.



When some attributes are multivalued attributes (cardinalities of **p** arcs equal [1,N] or [0,N]) they must be transformed into a molecular object related to the first one, to satisfy the atomicity of values which characterizes the first normal form relations. Depending on the attributes wether they are related by functional dependencies or not, this transformation may generate one or several molecular objects.

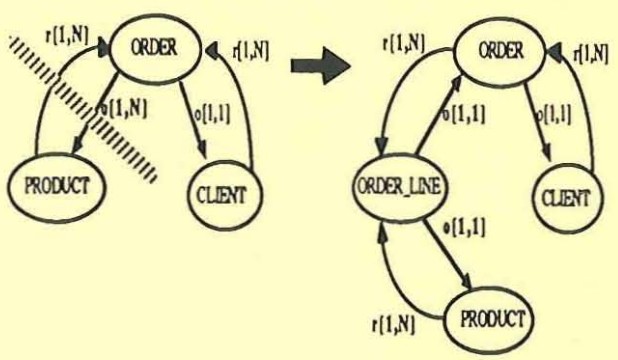


To be more relational, we must remove molecular aggregation arcs (r/o) and replace them by references and referential constraints, as portrayed in the following schema:

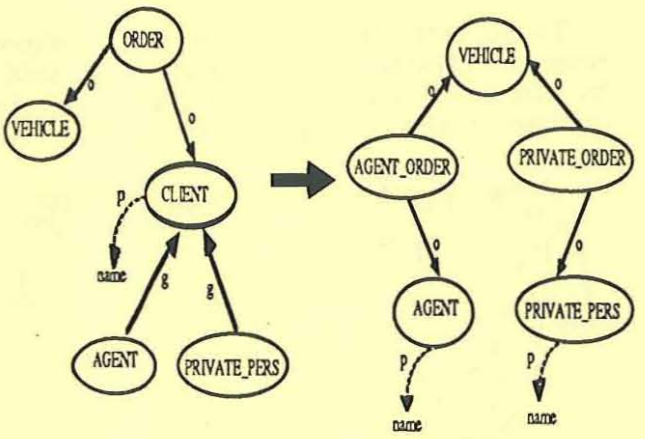


In fact, this transformation depends on the cardinality values. To satisfy the first normal form definition of relations, the cardinality of one of the two arcs (either **r** or **o**) must be equal to [1,1] or [0,1]. References designate the foreign keys of the related objects.

If the two cardinalities are different of [1,1] or [0,1], we must use another transformation which implies the definition of an intermediate object in such a way that one of its cardinalities equals [1,1]. That permits to come back to the previous transformation. This case is portrayed by the following schema.

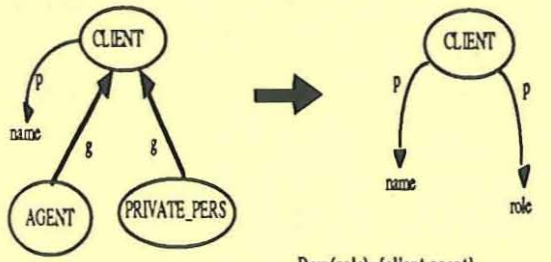


One of the semantic concepts which is not supported by the relational model is the generalization hierarchy. Thanks to the inheritance property, this concept can be represented by basic relations (implemented relations) and virtual relations (calculated relations or views). For example, in the left handside of the following schema, one can replace the CLIENT object by a virtual relation calculated from the union of AGENT and PRIVATE_PERS. The suppression of this generic class is immediately followed by the inheritance of its properties (related objects) to its sub-classes (schema of the right handside).



CLIENT = AGENT U PRIVATE_PERS

But this transformation is not the unique one allowed. Indeed, instead of removing the generic class, one can remove the sub-classes by replacing them with a specific attribute (say "role") which captures the role played by a given client in the specialization hierarchy. The sub-classes AGENT and PRIVATE_PERS should be calculated by restriction of the CLIENT class using the new attribute role. This case is illustrated by the following schema transformation:



Dom(role)={client,agent}
PRIVATE_PERS=[PERSON/Role="client"]
AGENT=[PERSON / Role="agent"]

In a given schema, the two preceding transformations are not both possible. Their application depends on wether the sub-classes have specific properties or not. If they have, the first transformation is better, otherwise the second one is better. In practice, this strategy is not so simple; we can also move up some specific attributes if we introduce a specific integrity constraint the check the null values on this attribute. To decide for each case which transformation to apply, the expert system may use heuristics based on the number of sub-classes, the number of specific properties of each class and the complexity of the possible integrity constraints to be generated.

In general, the mapping process from the conceptual schema to the logical relational schema is based on a specific strategy which is built in such a way that :

- no semantic information is lost during each transformation,
- no duplicate relation schemas are generated,
- the number and the complexity of the generated integrity constraints would not be too high.

After the mapping process, he normalization process is carried out, as in usual, by generating for each relation its minimal cover of functional dependencies.

7.4. The Justification of the Results

Justifications and explanations are especially emphasized int the expert system area. Such aspect is devoted in one hand to increase the user's knowledge in database design, and in other hand to enhance the credibility of the results obtained by the system. Explanations and justifications are little different. In the first case, the system has to explain the concepts, the methodology and the design rules which constitute the knowledge base. In the second case, the system has to justify different representation choices for a given database application. SECSI supports these two aspects at different levels.

At the first level, the system provides explanations for every ununderstood concept or question during the interactive design process. If a given concept is not understood by the user (e.g. a cardinality of a functional dependency), SECSI elaborates a text composed by a definition of the concept and an illustrative example). If a given question asked by the system is not understood because of its complexity (i.e. contains concepts which are not understood) or of its fuzzy form, the system decomposes the given question into easy sub-questions for which the user has only to answer "yes" or "no". This gives the system the ability to be used by both expert users and naive users. The documentation about the system use and the syntactic form of the different languages is also integrated at this level.

At the second level, the system justifies why an object of the application is represented as a normalized relation or as a virtual relation, why a given fact is represented by an integrity constraint whereas the user waits for a relationship, but also why a given fact does not explicitly appear in the results, or why some artificial objects are in the results whereas the user description did not contain them. To answer to a given question, the system elaborates a synthesis of the different design rules applied to the concerned object, from the analysis of the external description to the normalized relational schema. This synthesis is

based on the different states of the knowledge base, which are saved all along the design steps.

Explanation and justification is a very complex process which needs to remember a large amount of data an design rules. Its feasability is demonstrated in the first prototype of SECSI [BOUZ 86c], but its current industrial application is very limited.

7.5. The Incremental Design

As often claimed, the database design task is an iterative and long process which cannot be done definitely in a short time. Many refinements are necessary during a long period of time (several weeks or several months depending on the complexity of the application). The contribution of an expert system to this problem is to provide some methodological and some recovering features which permit the user to backtrack to any design step and to interrupt his design with the possibility to recover his session a few days later, without redesigning his application.

During each recovering session, the user would modify his first specification by adding new facts or modifying old ones. Hence, two different problems arise:
- how to make sure that the specification is consistent,
- how to integrate these new facts without reconsidering with the user all the pevious design (especially the set of previous asked questions).

To reach this double objective, SECSI stores in an extended fact base all the deduced facts from its rule base and all the captured fact from the user's answers. When a given session is recovered with some possible specification updates, the system generates a new fact base and procceeds through its following design steps. Whenever the system needs to ask a question to the user, the extended fact base is used first to derive a possible answer. Then only questions concerning new facts and modified facts are effectively asked to the user.

8. Conclusion:

In this paper, we have described an expert system approach for database design. As an answer to the various modeling problems, SECSI relies on the most elaborated concepts of databases and the most recent techniques of artificial intelligence. Compared to the existing tools, this approach seems more suitable for data base design in the sense that it takes advantage of both theoretical development and practical experience. Even if practically limited, the ability to justify several design choices and to explain reasoning alternatives is one of the important feature of expert systems; it makes them attractive for complex problems.

More than just a tool for the design of database schemas, SECSI can also be used in many other ways, as for example, a validation tool of manual design or as an educational tool in data modeling. Combined with a DBMS, SECSI can be considered as a powerfull prototyping environment of relational databases. SECSI is already used to design buiseness applications and technical applications like geographical databases.

Three main versions are dedicated to the design of databases : Version 1 concerns the production of a normalized relational schema from the given specifications in natural or declarative language. Version 2 is dedicated to the optimization of structures and to the generation of access paths. Version 3 is an extension of the conceptual modeling to the integration of views or to an incremental design of the data base. Each version evolves horizontally through a permanent research in improving the interfaces, the reasoning explanation and the documentation, offering among others the graphic edition and layout possibilities adapted to each type of model used. Currently, only the version 1 has reached the industrial level and then commercialized. The other version are in the prototyping level.

With future development of deductive databases and knowledge bases, this approach is more adequate to integrate new concepts and new design rules. We think that only powerful expert systems will efficiently handle the complexity introduced by these new research developments.

References

[BATI 85] BATINI C. and CERI S. "Database Design: Methodologies, tools and environments" panel sesion, ACM SIGMOD 1985.

[BEER 79] BEERI C. and BERNSTEIN P.A. "Computational problems related to the design of normal form relation schemas" ACM Transactions On Database Systems, marcH 1979.

[BERN 76] BERNSTEIN Ph. "Synthesizing Third Normal Form Relations from Functional Dependencies" ACM TODS, Vol1,N°4, 1976.

[BORG 85] BORGIDA A. and WILLIAMSON K. "Accomoding Exeptions in DB and Refining the schema by learning from them." VLDB Conf, Stockholm august 85.

[BOUZ 83] BOUZEGHOUB M. and GARDARIN G. "The design of an expert system for database design." Intl. Workshop on New Applications of Databases, Cambridge (UK), sept. 83. Published in New

Applications of Databases, Academic Press, Gardarin & Gelenbe eds. 1984.

[BOUZ 84] BOUZEGHOUB M. "MORSE : A Functional Query Language and its Semantic Data Model" INRIA RR270 and Proceed of 84 Trends and Application conf on Databases, IEEE-NBS Gaithersburg (USA), 1984.

[BOUZ 85] BOUZEGHOUB M., GARDARIN G., METAIS E. "Database Design Tools: an Expert System Approach" VLDB Conf, Stockholm august 85.

[BOUZ 86a] BOUZEGHOUB M. "SECSI: Un Système Expert en Conception de Systèmes d'Informations" Thèse de Doctorat de l'Universite Pierre et Marie Curie (Paris VI), mars 1986.

[BOUZ 86b] BOUZEGHOUB M., COMYN I. & RICHARD D. "Conception de Bases de Données: Etat de l'art sur la modélisation conceptuelle, l'intégration de vues et la conception physique" Rapport MASI-Université Paris VI N° 180, 1986.

[BOUZ 86c] BOUZEGHOUB M. & METAIS E. "L'explication du raisonnement et la justification des resultats dans le système expert SECSI", 2iem Colloque International en Intelligence Artificielle de Marseille, 1986].

[BROD 84] BRODIE M., MYLOPOULOS J., SCHMIDT Y. "On Conceptual Modelling: Perspectives from Artificial Intelligence, Data Bases and Programming languages. Springer-Verlag, NY 1984.

[BROD 86] BRODIE M. & MYLOPOULOS J. (editors) "On Knowledge Base Management Systems" Springer Verlag, 1986.

[BROD 87] BRODIE M. "Automating Database Design and Development" A Tutorial of the SIGMOD Conf., San Francisco 1987.

[BROW 83] BROWN and STOTT-PARKER "LAURA: A formal Database Model and her Logical Design Methodology." Proceed. VLDB Conf, Florence 1983.

[BUBE 82] GUSTAFSSON M., KARLSSON T. & BUBENKO J. "A Declarative Approach to Conceptual Information Modeling" in Information Systems Design Methodologies Olle, Verijn-Stuart editors, North Holland /Publ. Co, 1982.

[CERI 83] CERI S. (editor) "Methodology and Tools for Database Design" North Holland Publ. Co, 1983.

[CHEN 76] CHEN P.P. "The Entity Relationship Model - Toward a Unified View of Data" ACM trans. On Database Systems V1, N1, March 1976.

[CODD 70] CODD E.F. "A Relational Model of Data for Large Shared Data Banks", Comm ACM,Vol13, N°6, 1970.

[CODD 79] CODD E.F. "Extending The Database Relational Model to capture more Meaning." ACM trans. On Database Systems, 4, Dec 79.

[DBEN 84] Database Engineering Revue, "Special Issue on Database Design Aids" Vol7, N°4, 1984.

[FAGI 77] FAGIN R. "Multivalued Dependencies and a New Normal Form for Relational Databases" ACM TODS, Vol2, N°3, Sept 1977.

[GALL 84] GALLAIRE H., MINKER J., NICOLAS J.M. "Logic and databases: a deductive approach" ACM Computing Surveys vol 16, N° 2, Juin 84.

[GARD 89] GARDARIN G; & VALDURIEZ P. "Relational Databases and Knowledge Bases" Addison Wesley Publ. Co. 1989.

[HAMM 81] HAMMER N. and McLEOD D. "Data Base Description with SDM: A Semantic Data Model" (ACM TODS V6, N3, Sep 81).

[HAYE 84] HAYES-ROTH F. "The knowledge based Expert Systems: A Tutorial" Computing Revue, Vol 17, N°9, 1984.

[HULL 87] HULL R. & KING R. "Semantic Database Modeling: Survey, Applications and Research Issues" ACM Computing Surveys, Vol 19, N°3, 1987.

[KENT 79] KENT W. "Limitations of Record-Based Information Models."ACM Trans.on Database Systems 4,1,1979.

[MYLO 80] MYLOPOULOS J. BERNSTEIN P.A. WONG H.K.T." A language facility for designing database intensive applications" ACM trans. On Database Systems vol15, nb2, 1980.

[NILS 82] NILSSON N.J. "Principales of Artificial Intelligence", Springer_Verlag Berlin Heidelberg New York, 1982.

[OODB 86] First Workshop on Object Oriented Database Systems, IEEE Computer Society Press, 1986.

[PECK 88] PECKHAM J. & MARYANSKI F. "Semantic Data Models" ACM Computing Surveys, Vol 20, N°3, 1988.

[SMIT 77] SMITH J.M. and SMITH D.C.P. "Data Bases Abstractions Aggregation and Generalisation " ACM trans. On Database Systems, june 77.

[TUCH 85] TUCHERMAN L., FURTADO A. and CASANOVA M. "A Tool for Modular Database Design." VLDB Conf, Stockholm 1985.

[ULLM 82] ULLMAN J.D. "Principles of Database Systems" computer science press 1982.

[ZANI 81] ZANIOLO C., MELKANOFF M.M: "On the design of relational data base schemata", ACM trans. On Database Systems, V6, N1; march 1981.