Contributions to Logical Database Design

Vitalie COTELEA
Academy of Economic Studies of Moldova, Chisinau, Moldova
vitalie.cotelea@gmail.com

This paper treats the problems arising at the stage of logical database design. It comprises a synthesis of the most common inference models of functional dependencies, deals with the problems of building covers for sets of functional dependencies, makes a synthesizes of normal forms, presents trends regarding normalization algorithms and provides a temporal complexity of those. In addition, it presents a summary of the most known keys' search algorithms, deals with issues of analysis and testing of relational schemes. It also summarizes and compares the different features of recognition of acyclic database schemas.
Keywords: *Logical Database Design, Functional Dependencies, Normal Forms, Acyclic Database Schema*

1 Introduction

A properly designed database provides access to accurate and updated data. Because a correct design is essential for achieving the goals of using a database, the ability to design databases and associated applications is critical to the success of any modern enterprise. Automatic drafting and the development of database and of the information system as a whole consists of a series of steps that require methodological research, efficient models and algorithms and software design tools.

Within the logical design phase, the conceptual schema, expressed in a high-level data model is transformed into a global logical schema, described in a logical data model, for example, the relational model, without taking into account a specific DBMS, in this case, is obtained a system independent logical design, but dependent on the data model.

The global logical schema is normalized [16], [17], all keys [40] and the links between relations are identified. Then, the global logical schema and the information about access to data serve as input for the next step which is distribution design [47]. The objective of this phase is to design local logical schemes, which are distributed to all the stations of the distributed system. The paper reviews the current issues that pertain to logical database design for the purpose of automation of this process.

2 Visions on functional dependencies inference models

It is being considered [52] that the problem of logical design of relational database consists in laying the theoretical and practical basis for taking decisions about:

- What should the relations of the database;
- What attributes should each relation consist of.

Here a design aspect needs to be mentioned - determination of integrity constraints [49]. Constraints, however, are withdrawn at the conceptual design phase. Therefore, the success of DBMS utilization with up-to-date mechanisms to maintain integrity constraints cannot depend on a common treatment process for obtaining these constraints [58].

The simplest integrity constraint is functional dependency and Codd introduced it into the database theory [15]. Deriving from semantic rules, which translate the restrictions of the domain of interest, the designer has to define, among other types of dependencies, and functional dependencies and to introduce them in the definition of the database schema. The properties of dependencies are the ownership of the relational scheme (intensions) of the database, and not of any extension of the database, that is these dependencies are invariant and need to be satisfied by all legal extensions that correspond to the scheme [18]. The only way to find the valid functional dependencies for a schema consists in a careful analysis of

each attribute's significance and of the way values are assigned to the attributes [51].

Starting from a set of functional dependencies, attached to a relational scheme, other valid functional dependencies can be deducted. There are many rules of inference and in order to be able to make a formal presentation of these, three of them were chosen by Armstrong [1], and the rest of them are derived from them [35].

Armstrong's axioms represent a set of sound and complete inference rules. They are sound because they generate only functional dependencies and are complete, as they generate all the possible functional dependencies based on a given set of dependencies.

But the utilization of these axioms in automatic inference is complicated. The inference has an exponential nature and for certain dependence it is not unique. In addition, it is difficult to say unequivocally that using Armstrong axioms, a dependency can be inferred or not, due to that the dependency cannot be deducted because of an incorrect order of applied axioms.

These disadvantages do not allow the direct use of rules in the software of relational schemas design [65]. For these reasons, another set of rules, also sound and complete, is defined in [41], but which guides inference and directs it towards more promising goals. These sequences are called RAP-derivations, after the first letters of the rules used.

However, the RAP-derivation sequences are not unique for a given functional dependency and they have not eliminated the drawbacks mentioned above. To exclude these disadvantages, Beeri and Bernstein [6] propose a tree derivation model for functional dependencies. With the help of this model, the complexity of calculating the closure of a set of attributes under a set of functional dependencies is linear. But, to demonstrate various assertions related to functional dependency structures is not appropriate.

The derivation tree was the precursor of an inference model, called directed derivation acyclic graph (DDAG), introduced by Maier [41]. This model can be designed as a graphical representation of RAP-derivation. But this model too can generate a range of sequences of inference for a given dependency.

Thus, a model is required that would derive in a linear fashion, the set of attributes that are functionally dependent (under a set of functional dependencies) on a given set of attributes. This model must have the property of uniqueness and be a very easy tool to use in demonstrating assertions about structures of functional dependencies. In general, this model will not represent anything else than a sequence of sets of attributes, which are built in an iterative way, involving, in their building, groups of functional dependencies with left sides included in the previous set.

The algorithm based on this model would be effective, if it would not check repeatedly each functional dependency, even if it has already been verified. It must be based on the fact that each dependency is used only once, and namely, only when its left side is already included in the calculated result up to that point. This can be achieved, for example, by a counter being associated to each dependency that stores the number of attributes from its left side that are not yet included in the result. When this counter becomes 0, the respective functional dependency must be taken into account so that the right side is added to the result. Of course, whenever a new attribute is added to the result, the algorithm has to decrement the counter value of each dependency containing this attribute in the left side. To execute this mandatory step effectively, there must be maintained for each attribute, a list of dependencies that contain this attribute in their left side.

If this strategy is applied, then the complexity of calculating the closure of attributes under a given set of functional dependencies with this model is linear.

Calculation of the closure of a set of attributes [36] essentially simplifies the problem of determining whether a functional dependency belongs to the closure of a set of

functional dependencies without having actually built this closure.
As a sequence of sets of attributes, it can be constructed and a reduced version of the model, which can be applied effectively to the inference of functional dependencies. It is obvious that this model should be equivalent to the application of dependencies inference with Armstrong axioms, that is, must possess the property of soundness and completeness [57].

3 Current status of covers for functional dependencies

Most algorithms of relational database theory use a set of functional dependencies as input. Algorithms' efficiency depends on the cardinality of the set.
Below are presented a few problems referring to diverse covers for sets of functional dependencies. A set of functional dependencies is a cover for another set, if the sets are equivalent.
Sets of functional dependencies can be nonredundant when they do not contain redundant dependencies, can be left-reduced or right-reduced, when they do not contain extraneous attributes in the respective side. Can be canonical, when dependencies are left-reduced, non-redundant and the right side is formed of only one single attribute.
Gottlob in [33] investigates the relative size of equivalent covers for functional dependencies.
Taouil and Bastide [53] propose the so-called proper cover, which is defined as a set of left-reduced functional dependencies, but every functional dependency has a single attribute on the right side.
It may be noted that this type of cover is in a close relationship with canonical cover and, consequently, with reduced cover. In fact, a canonical cover is a proper nonredundant cover. The relationship with reduced cover is also more direct. In this case, a proper cover can be obtained directly from a reduced cover, using the projectivity axiom, which decomposes the dependencies in the right side.
It should be noted that there is no condition imposed for the obtained set to be nonredundant, thus, the obtained set will not be necessarily a canonical cover.
The important result in the domain of covers for dependencies is described in [42] and consists in demonstrating the existence of a minimum cover for functional dependencies and the presentation of a polynomial algorithm for the computation of such a cover.
Finding the minimum covers for a certain set of functional dependencies is useful because of at least two points of view:

- To reduce the necessary time required of their imposition over appropriate database contents;
- To reduce the time required to execute the algorithm for calculating the closure (because it is proportional to the size of the set of dependencies).

As defined by Maier [42], minimality is defined in relation with the cardinality of the sets of dependencies. The notions of minimum set (with as few dependencies as possible) and of optimal set (which is as concise as possible), were introduced by Maier [41]. He noted that although a minimum cover for a particular set of functional dependencies can be found in polynomial time, the obtainment of an optimal cover is a NP-complete problem.
Since the problem of finding optimal cover proved to be complex, it makes sense to investigate whether the minimum covers are indeed the best covers that can be obtained. Mannila and Räihä in [44] explore further the correlation of these two covers. It has been proven that the length of a minimum set has the following properties:

- Cannot be bounded by a linear function on the length of the optimal cover;
- Is bordered by the square of the optimal cover length.

It is also proven the fact that NP-completeness of the optimization problem is somehow surprisingly caused exclusively by the difficulty of optimizing only one single class of dependencies having equivalent left sides [44].

This result reveals a few practical significances, since the equivalence classes that occur in practice, are short. Optimization problem of an equivalence class is studied by several researchers, but it should be noted that the left side and the right side present different behavior.

Thus, the task of obtaining an optimal cover is part of the NP-complete problems category. The only way to obtain a solution in an acceptable time would consist in proposing certain methods of decomposition of the initial problem in sub problems which can be solved and after that, the particular solutions to be combined in order to build the solution to the given problem.

Various types of covers assign specific properties to the database schema. This is the main reason why so much attention is given to the algorithms of creating and testing covers of functional dependencies.

4 Normal forms approaches

Database relations contain both structural and semantic data. The structure is described by the relation scheme and semantics is expressed by the functional relationships between attributes. A good project of the database assumes that the grouping of attributes is rational and satisfies the following conditions:

- All the keys and their features to be found and specified for all database relations;
- The content of the relational schemes to be characterized by a minimal redundancy which needs to be controlled by the DBMS;
- Among attributes, there shouldn't be any unwanted functional dependencies;
- It is necessary to exclude insert, update and delete anomalies of the data operations;
- The restructuring of relational schemas has to be minimal in the case of database development.

For obtaining a performing database, an important role pertains to the normalizing technique of relations. This technique allows obtaining the logical scheme through a process of gradual improvement of an originally designed scheme by using normal forms. After each stage of improvement, the relations in database reach a special degree of perfection by eliminating a certain type of unwanted dependencies (partially and transitively dependent and multivalued dependencies), so they are in a particular normal form.

The improvement process must meet the following requirements:

- Ensure lossless-join decomposition, i.e., the final logical scheme must contain all data from the initial scheme;
- Ensure preservation of data dependencies, i.e., in the final scheme, each dependency must have the left and right sides in the scheme of the same relations;
- Represent a minimal decomposition of the original relations. None of the relations that make up the final schema should be contained in another relation of this schema.

The quality of a relation (or the ability to represent the real world without generating updating problems) is measured by the degree of standardization. Codd proposed three normal forms that he called First Normal Form (1NF) [15], Second Normal Form (2NF) and Third Normal Form (3NF) [16]. A stricter definition than 3FN was proposed by Codd and Boyce [17] and is known under the name Boyce-Codd Normal Form (BCNF). All these normal forms, except for 1FN are based on the functional dependencies between the attributes of a relational scheme [11].

The 1NF refers to the structure of the relation. It is required that each attribute of a relation is based on an atomic domain. Database designers have no problem to recognize whether a relationship is not in the first normal form [65]. They can bring the relation in 1FN algorithmically by replacing the composed domains by atomic constitutive domains. For 2NF, 3NF and BCNF, it is necessary that designers of the database know the meaning and the real application of keys, such as the candidate keys, primary ones, super-keys, etc.

The 2NF is based on the concept of full dependency. A relation is considered in 2NF with respect to a set of functional dependencies, when it is in 1NF one and each attribute, which is not a part of any key, is fully dependent upon every key of the relation. In other words, there is no attribute of that type that would partially dependent at least on one key. It is clear that 2NF is relevant, in the case when at least one key of the relation is compound, that is consists of at least two attributes.

A relation is in 3NF with respect to a set of functional dependencies, if it is in 1NF and none of the attributes that are not part of any key don't transitively dependent upon a key.

BCNF is an extension of 3NF, when two or more composite keys overlap (which have at least one common attribute). If these conditions are not met, 3NF and BCNF are equivalent. A relation is in BCNF if and only if the left side of every non-trivial dependency is a super-key.

Although the appearance of papers on normal forms for relational schemes seemed to come to an end, works in design theory continue to occur, such as [48, 45, 55, 43]. It was noted that relations which are in 3NF and are not in BCNF represent interesting properties. These properties can be found in the well-known text- books [59, 24] and in the research works [21, 60]. In [21], it is proved that if a relation is in 3NF, but is not found in BCNF, then the relation must contain at least one compound key. In another work, Vincent has deduced a stricter result for this case, and namely that the relation must possess at least two keys that dispose common attributes [60]. However, the brought demonstration does not cover explicitly the fact that the two candidate keys are compound.

In specialty literature, it was also observed that the decomposition of one scheme into others, which meets the highest normal forms, is not a sufficient condition for a good project. Decomposition should be strengthened further by additional properties such as lossless join and preservation of constraints [24]. This requirement becomes questionable when we see relations, in 3NF, converted into BCNF, with loss of certain functional dependencies. Although Makowsky proposed a division technique of the attribute set [43], that preserves dependencies, the separation of attributes is not always possible.

Logical schema design of a database implies the determination of the normal form, in which the relations within the database should be. In the majority of cases, the relational databases are constituted of relations which are in 1NF or 2NF. This is explained by the fact that the superior normal forms, though reduce the difficulty of accomplishing the update tasks they are also reducing at the same time the performances of the data retrieval operations [20].

Relations in higher normal forms contain a small number of attributes and this issue favors the operations of data actualization, but burdens their retrieval process, because data satisfaction require simultaneous interrogation of multiple relations, so performing certain join operations, which are costly in terms of required computing resources.

It is clear that this form conflicts with the ANSI/SPARC database architecture, fact which destroys the independence between applications and the logical structure, but also the physical one of the databases.

Another problem is to establish relations to be part of the database, in the normal form specified in the previous step and involves defining the relation schemes and integrity constraints. The way of establishing the set of relations in the database is called relations normalization technique.

Normalization can be achieved by two methods: through decomposition and through synthesis.

Normalization through decomposition utilizes the top-down division method of a table into two or more tables, keeping connection information (attributes). Decomposition is a reversible process, step by step, of progressive replacement of a given set of relations with successive sets, in which the relations are simpler and with more regular structures [50]. Reversibility

ensures that the initial set of relations can be recovered and therefore no information is lost [18].

It is well known the fact that a satisfactory decomposition (with preservation of dependencies) of the relational database schema in BCNF is not always possible. This issue depends on the given set of functional dependencies, and the corresponding decisional problem is an NP-hard one. The only algorithm which guarantees the preservation of dependencies as well as the existence of the BCNF was proposed in [46] which is a raw approach and always requires exponential time. To be useful in practice, for example in automated design tools, more effective means are required.

The paper [37], presents a very efficient algorithm that always finds a decomposition, in BCNF, with preservation of dependencies, if any, and usually an effective one, and which is exponential only in well-known cases.

Normalization through synthesis is a method which starts from an attributes set of global relation and from a set of functional dependencies, highlighted by the analysis process and builds the basic relations of a decomposition, selecting its attributes in a certain way. Under certain conditions, the synthesis can provide a valid decomposition that preserves functional dependencies. The synthesis of the schema in 3NF was proposed by Bernstein [10].

Today, it is required a modification and an improvement of this algorithm to meet the demands of the day, taking into account the dynamic modification of the database structure imposed by the emergence of new applications, views which also should enjoy the properties of the lossless join. Moreover, it should consider the fact that some attributes of potential views may not be involved in the functional dependencies.

5 Aspects for development of keys searching algorithms

In databases the keys play an important role. Tuples can be identified, saved and searched in an unique way. In general, the key is an attribute or a set of attributes that uniquely identifies a particular tuple.

The keys are generalized by a type of functional dependencies. They specify the relationship between two sets of attributes. Functional dependencies are used for database normalization. Therefore, the sizes of the set of functional dependencies and of the set of keys present high interest.

Majority of authors in the database domain provide definitions for the key, but not also a calculation algorithm for this. David Maier [41] and Jeffrey Ullman [59] provide algorithms for calculation of closure for a set of attributes or a set of functional dependencies, but the calculation of a key is left for readers, with the suggestion to use the algorithm of calculation the closure. The determination of the keys of a small relational scheme with a small set of functional dependencies can be simple, but if a scheme has a relatively large number of attributes and/or functional dependencies, then the finding of keys cannot be a trivial process [24].

Algorithms for finding the set of all keys for a relational schema were constructed in [9], [25], [23], [29], [40]. It should be noted that the methods proposed by Lucchesi, Osborn and Fernandez are the only of polynomial complexity just in some particular cases. Thus, often, especially when the number of minimal keys is relatively small, these algorithms are better than those in [9], [25], [23].

The prime attributes and the minimal keys play an important role in the process of relations normalization. Lucchesi and Osborn [40] proved that the following two problems are NP-complete:

1. The prime attribute problem. Being given a relational scheme and an attribute A, to determine if A belongs to a key.

2. The key cardinality problem. Being given a scheme and an integer $m>1$, to determine if there is a key with a cardinality smaller than m.

Specifically, based on NP-completeness of statement (2), Maier in [42] noted that there is probably no polynomial time algorithm for

finding an optimal cover for a set of functional dependencies. This problem belongs to the class of NP-complete problems, for which no one has yet found any polynomial time algorithms.

Beer and others [4] have shown that problem (2) remains NP-complete, even if the entry of the algorithm consists of a fixed relations (matrix) instead of a relational schema. In [40] Lucchesi and Osborn have shown that the problem of the cardinality of the key is polynomial convertible towards the prime attribute problem. So, if NP≠P, then the transformation is not possible for the relations.

The keys represent a class of constraints, which is of great importance for maintaining the database in a consistent state. Currently, there are two competing approaches in defining keys. These are the natural keys and the surrogate keys.

A natural key is a candidate key, which represents a logical link of a subset of attributes of a relational scheme. The existence of a natural key is known to the users and to business. This may consist of several attributes, although there are numerous examples of natural keys formed of a single attribute.

The main disadvantage is the susceptibility of natural keys to modifications in both the value and structure. Changing the structure of a natural key usually involves serious problems in the database and in maintaining applications, because the modification has to be applied in several places. Consequently, the use of natural keys as primary keys is often challenged in the industry.

A surrogate key is a single attribute whose values are (1) numeric, (2) generated by the system and (3) utilized to uniquely identify tuples in a relation. Their existence and their values are invisible to users.

A perceived advantage in the use of surrogate keys as primary keys consists in their immutability, which is a consequence of their separation from the businesses logic. E.F. Codd in [14] has defined the surrogate keys as an architecture of the relational database. Each relational scheme has a surrogate attribute as primary key. Primary surrogate keys are propagated to other relations as foreign keys.

Wastl in [62] introduces an inference system to obtain the keys of a relational scheme. Entities derived with this system are functional dependencies. The system is closed, meaning that all functional dependencies that are obtained with it, are in logical sequence. The system is complete, meaning that for each key of the scheme it represents a functional dependency that can be derived. The completeness of the system was used for bounding the cardinality of the set of keys of the scheme, with the value $\left\lfloor e^{|F|/e} \right\rfloor$, where $|F|$ represents the number of functional dependencies defined on the scheme.

In [56] it is shown that the number of keys of a relational schema is bounded by $\left(\left\lfloor \frac{n}{\frac{n}{2}} \right\rfloor \right)$, where n is the number of attributes of the scheme. It should be noted that the estimation depends on the number of attributes, and in [62] – on the number of functional dependencies defined over the schema. This is an essential difference.

The idea proposed in [39] is to build, based on canonical or minimum covers for a set of dependencies, of a matrix, called MAC, which facilitates the calculation of closures of all sets of attributes. Using this matrix, it is proposed a fast calculation algorithm of the closure of a set of attributes, fact which improves the search of keys. The improved algorithm of finding keys does not need to calculate the closure of the subsets which are tightly relevant for the keys, and namely, which are keys or a part of keys. Based on this algorithm and on the theorem of reference from [25], the efficient algorithm for finding all the keys has the complexity $O(n*|R|*|F|^2)$, where $|R|$ represents the number of attributes of the scheme, and $|F|$ - the number of functional dependencies defined over the scheme.

6 Issues and benchmarks for analysis and testing of relational schemes

One category of problems that may arise in the development of certain applications using a database, is that of an incorrect design of relational schemas. Testing the correctness of a scheme can be made using functional dependencies (or of other type of constraints) attached to that scheme.

More decision making problems related to relational schemes with functional dependencies are difficult to calculate. Such problems include the prime attribute problem as well as the testing whether a scheme is in a certain normal form. The algorithms for these problems are necessary for the tools of databases' design. But these problems, for the time being, can be solved only by algorithms of exponential complexity.

Although the size of the instances is usually given by a set of attributes and, therefore, is quite small, such algorithms cannot be utilized for all design tasks.

Testing if a relational schema is in BCNF is an easy problem. In fact, it should be tested for all dependencies defined over the scheme, if their left parties are super-keys [46]. This is clearly made in polynomial time, in contrast to testing if the schema is in 3NF, which is NP-complete task, because the testing of prime attributes is NP-complete [40].

It should be noted that the first statement is correct, only if the given set of dependencies is an exhaustive one. In case it does not represent the closure of the set of dependencies, the problem is of an exponential character.

Although the testing of the BCNF, if the above condition is satisfied, is executable in polynomial time, to detect whether a sub-scheme of the scheme is in this form, it is an NP-complete problem [6]. The reason of complexity increase lies in the following. For the Boyce-Codd problem is given the relational schema. In other words, the set of functional dependencies is part of the input. And it just remains to be tested if the left side of each dependency is super-key. But in the case of a subschema the set is not explicitly known.

Worland [63] presents an algorithm that determines whether a scheme is in 3NF. The algorithm operates, classifying the attributes of the schema in the so-called dependent sets, which are based on the set of functional dependencies defined over the given scheme. To view the dependencies, a new type of graph of dependencies is introduced. The algorithm works faster than the algorithms designed for finding all candidate keys of the scheme, especially if there is more than a dependent set.

Obviously the question arises if, for recognition of the normal form of a scheme, it is essential to determine the prime and nonprime attributes or the determination of all keys. A solution would be to determine the equivalent features of these entities, but which could be calculated in polynomial time.

7 Criteria for recognition of acyclic database schemas

Hypergraphs generalize the notion of graph, introducing the notion of hyperedge of the graphs, by extending the notion of the edge of the graphs, non-imposing restrictions on the number of nodes belonging to an edge. This relaxation of the constraint on the number of nodes allows generalization of results of the graphs theory and the study of specific classes of hyper-graphs. Hyper-graphs are therefore preferred to graphs due to the higher generality. However, this flexibility is not only one facet of a medal, the other side can be the increase of algorithms' complexity defined over the hyper-graphs, in comparison with those which are applied to the graphs.

A special interest for the theory and practice of database design is represented by acyclic hyper-graphs.

Acyclic hyper-graphs were introduced in hyper-graphs same as the trees are a special case of graphs [7]. Besides representing an interesting mathematical structure, the acyclic hyper-graphs are a fundamental element in the study of database theory and

of constraints' satisfaction. However, unlike regular non-oriented graphs, there exist a range of nonequivalent notions of acyclicity of hyper-graphs.

The database schemas can be viewed as hyper-graphs with individual relational schemes corresponding to the edges of a hyper-graph. There is a natural bijection between database schemas and hyper-graphs, where each attribute of a database schema corresponds to a node in a hyper-graph and each relation scheme corresponds to an edge in the hyper-graph.

Under this setting, a new class of database schemas, called acyclic, was introduced and proved that it claims a number of desirable properties. Of particular interest among these are the α, β and γ acyclicity levels each characterizing an equivalence class of properties for the database schemas, represented by hyper-graphs.

Acyclic database schemas (corresponding to acyclic hyper-graphs) were first studied by Beer and others [3]. This natural class of database schemas has proved to possess important and desirable properties [3, 30, 27, 5, 26]. Acyclic hyper-graphs have become a subject of many researches.

One of the main reasons regarding the opportunity of using the acyclic database schemas is that there are important problems that are NP-hard regarding the general database schemas, but which become solvable in polynomial time when they are limited to acyclic instances. Examples of such problems include:

- Determining the global consistency [5];
- Evaluation of conjunctive queries [64];
- Calculation of joins or projections of joins [64].

In addition, the acyclic schemes of databases can be recognized in linear time [54]. D'Atri and Moscarini [22] have offered a recursive algorithm of pruning to determine the acyclicity level of the hyper-graph.

When such problems of difficult computation occur that have to be solved on a general schema of the database, it is natural to decompose it into α-acyclic instances, on which efficient algorithms can be applied. This has motivated some recent studies on α-arborescence of hyper-graphs, the minimum number of α-acyclic hyper-graphs in which the edges of the given hyper-graph can be partitioned [61, 13]. The main contributions are brought to asymptotic determination of the arborescence of a completely uniform hyper-graph with a large size of the edge [8].

Grohe and others [34] have shown that the evaluation of conjunctive queries with the width of the bounded tree is treatable. Decision making issues such as evaluation of Boolean conjunctive queries and query content are solvable for acyclic queries [32, 31].

A number of other properties have been identified that have been studied by a few researchers in quite different contexts, and every of these properties are equivalent with the acyclicity. Thus, the class of acyclic database schemas is a natural class, an important one, as it can be characterized in a number of ways, each corresponding to a desirable property of the database schema or to a natural property of the graphs theory.

As known, there exist various undesirable and pathological phenomena that may occur in the general schemes of the database, but not in the acyclic databases schemas. So, by directing attention to the case of acyclic schemas, the theory is much more elegant. In addition, this restriction has the property [27] that acyclic database schemas are sufficiently general to comprise the biggest number of situations of the "real world".

At least the database designers should be aware of existence of acyclicity and strive to utilize it. Given the circumstances, it results that, by focusing on acyclic cases, researchers can develop a strong and stylish theory which often are applied to schemas that represent the domain of interest for which the database is built.

For acyclic schemes the algorithms are efficient, polynomial time, while for nonrestrictive cases the problems are part of NP-complete class. One example is to determine the global consistency. Other examples are presented by Yannakakis [64].

In addition, the determination of acyclicity

degree may be achieved by a simple algorithm.

There exist various interesting problems related to relational databases, in the case when a certain type of object can be viewed as a collection of sets, and a property of the object depends on the structure of this set. Now, a collection of sets can be viewed as a hyper-graph. It seems that, for various properties, the acyclicity of a hyper-graph is equivalent to the validity of these properties. Such properties appear in (at least) three distinct areas.

The first field appears when a database schema is treated as a collection of sets of attributes. Certain properties of relational databases are discussed [5] that depend on the structure of the schema. For example, such a property refers to the fact that, if the database on a schema is pair-wise consistent, then it is totally consistent.

A second area is the theory of dependencies. One of the most important types of dependencies is the join dependency, which can be viewed as a collection of sets. The desirable property here consists in the fact that the join dependency is logically equivalent with a set of binary join dependencies, i.e. with a set of multi-valued dependencies.

A third area pertains to queries processing. Here, join expressions are of great importance, and these, again, [28, 2] are collections of sets. Interesting problems relate to the existence of a few access paths in efficient time and/or efficient space. All these problems of distinct areas are linked by the conditions of acyclicity over the structure of the hyper-graph.

Obviously, in order to describe all these problems, there are presented a series of definitions and new constructions and a number of conditions which are proved to be equivalent to the acyclicity. Here needs to be emphasized that there are various types of acyclicity [26] for hyper-graphs, and consequently for the database schemas.

In this way, are described the features for a set of multi-valued dependencies to be a consequence of a join dependency. Also, the conflict-free sets of multi-valued dependencies can be characterized [12]. For example, it is shown that an acyclic join dependency with n relation schemes is equivalent to a set of at least $n-1$ conflict-free multi-valued dependencies. This fact strengthens the result of [27], which says that any acyclic join dependency is equivalent to a set of multi-valued dependencies the size of which is polynomial in relation with the size of the join dependency.

In the paper [38] were presented complementary approaches for the design of $\beta-$acyclic databases schemas. Lakshmanan has introduced a new concept called "independent cycle." Based on this is developed a criterion for $\beta-$acyclicity, which is equivalent to the existing definitions of $\beta-$acyclicity. From this, as well as from the concept of the dual hyper-graph, there can be developed an efficient algorithm for testing the $\beta-$acyclicity.

Because of the fact that various degrees of acyclicity of the schemas has various properties of the databases [19] the development of certain equivalent conditions for the existence of acyclic database schemas are welcomed, in general, and the determination of the most efficient algorithm for testing the $\beta-$acyclicity, in particular. This is motivated by the fact that for a $\beta-$acyclicity schema, any subschema is also $\beta-$acyclic, thus allowing every user to benefit from the qualities of the database schema.

Apart from this, it is useful to study Berge-acyclic schemas for the case when the hyper-graph consists of several strongly connected components.

8 Conclusions

It can be concluded that designing a good database schema is more an art than a science. In the last two decades, there have been achieved many scientific advances in the logical database design, while this cannot be affirmed about the other phases. However, it cannot be stated that the logical design process can be automated. It is only the

beginning of the way. There exists a range of issues not yet resolved. Many of the known algorithms are of exponential complexity and cannot be applied in practice. The process of analysis of the existent database features is extremely modestly studied.

In addition, most methods used in database design are, in fact, empirical solutions, often, unsupported by any scientific basis or any engineering discipline. The ad hoc design methods, often, lead to inflexible solutions that do not meet the business requirements. Costly remedial measures often cause more delays in operation, without any tangible improvement. Many design tools are presented in the form of individual analyses. Although these analyses provide valuable information, they, however, can hardly be an adequate substitute to a discipline or a systematic design tool.

References

[1] W.W. Armstrong, "Dependency Structures of Data Base Relationships", *IFIP Gong.*, Geneva, Switzerland, pp. 580-583, 1974.

[2] A. Giorgio, A. D'Atri and M. Moscarini, "On the Existence of Acyclic Views in a Database Scheme", *Theor. Comput. Sci*., V.35, pp.165-177, 1985.

[3] C. Beeri, R. Fagin, D. Maier, A.O. Mendelzon, J.D. Ullman and M. Yannakakis, "Properties of acyclic database schemes", in *Proc. ACM STOC* (Milwaukee, Wisc., May 11-13, 1981), ACM, New York, pp.355-362, 1981.

[4] C. Beeri, M. Dowd, R. Fagin and R. Statman, "On the Structure of Armstrong Relations for Functional Dependencies", *Journal of the ACM*, V.31, N.1, pp.30-46, Jan. 1984.

[5] C. Beeri, R. Fagin, D. Maier and M. Yannakakis, "On the Desirability of Acyclic Database Schemes", *J. Assoc.Comput.*, V.30 N.3, pp. 479-513, *Mach*., 1983.

[6] C. Beeri and A. P. Bernstein, "Computational problems related to the design of normal form relational schemas", *ACM Trans. Database Syst*., V.4, N 1, pp.30-59, 1979.

[7] C. Berge, "Graphs and hypergraphs", 2., rev. Ed, North-Holland Publ. Co., Amsterdam, xiv,528 p, 1976.

[8] J.-C. Bermond, Y. M. Chee, N. Cohen and X. Zhang, "The α-arboricity of complete uniform hypergraphs", *SIAM J. Discrete Math*. V.25, N.2, pp. 600–610, 2011.

[9] P. A. Bernstein, "Normalization and Functional Dependencies in the Relational Data Base Model", Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, Toronto, 119p, 1975.

[10] P. A. Bernstein, "Synthesizing Third Normal Form Relations from Functional Dependencies", *ACM Trans. Database Syst*., V.1, N 4, pp.277-298, 1976.

[11] P. Beynon-Davies, "Database systems", 3rd Edition, Palgrave-Macmillan, 616 p, 2004.

[12] K. Chase, "Join graphs and acyclic data base schemas", in Proc. of the International Conf. on Very Large Databases (VLDB '81) (Cannes, France), pp.95-100, 1981.

[13] Y. M. Chee, J. Lijun, A. Lim and K. H. Tung Anthony, "Arboricity: An Acyclic Hypergraph Decomposition Problem Motivated by Database Theory", Preprint submitted to Discrete Applied Mathematics, 11p, 2011.

[14] E.F. Codd, "Extending the Relational Data Model to Capture More Meaning", *ACM TODS*, V.4, N.4, pp.397-434, 1979.

[15] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Comm, of ACM*, V.13, N6, pp. 77-387, 1970.

[16] E.F. Codd, "Further Normalization of the Database Relational Model", in *Data Base Systems, Courent Comp.Sci. Symposia Series*, 6, Englewood Cliffs, NJ: Prentice-Hall, Rustin, pp.33-64, 1972.

[17] E.F. Codd, "Recent Investigation in Relation Data Base Systems", IFIP Congress, pp.1017-1021, 1974.

[18] M. T. Connolly and C. E. Begg, "Database systems: a practical approach

to [17] design, implementation, and management", Fourth Edition, Addison-Wesley, 1374p, 2005.

[19] V. Cotelea, "Modele şi algoritmi de proiectare logică a bazelor de date", Editura ASEM, Chişinău, 266 p, 2009.

[20] C. J. Date and H. Darwen, "Databases, Types, and the Relational Model. The Third Manifesto", Addison Wesley; 3th edition, 572 p, 2006.

[21] C. Date and R. Fagin, "Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases Design", *ACM Transactions on Database Systems*, V.17, N.3, pp.465-476, 1992.

[22] A. D'Atri and M. Moscarini, "Acyclic hypergraphs: their recognition and top-down vs bottom-up generation", *Technical Report R.29*, Consiglio Nazionale Delle Ricerche, Instituto di Analisi dei Sistemi ed Informatica. pp. 82–94, 1982.

[23] C. Delobel and R.G. Casey, "Decomposition of a Data Base and the Theory of Boolean Switching functions", *IBM Jour. of Res. and Develop.*, V.17, N5, pp.374-386, 1973.

[24] R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems", Addison-Wesley Publishing Company, 5th edition, 1168 p, 2006.

[25] R. Fadous and J. Forsyth, "Finding candidate keys for relational data bases", in *Proc. of the ACM SIGMOD International Conference Management of Data*, ACM Press, pp.203-210, 1975.

[26] R. Fagin, "Degrees of acyclicity for hypergraphs and relational database schemes", *J. Assoc. Comput. Mach.*, V.30, N.3, pp.514-550, 1983.

[27] R. Fagin, A. Mendelzon and J.D. Ullman, "A Simplified Universal Relation. Assumption and Its Properties", *ACM Trans. on Database Systems*, V.7, N.3, pp.343-360, 1982.

[28] R. Fagin, "Acyclic database schemes (of various degrees): a painless introduction", *Lect. Notes Comput. Sci*., Nr. 159, pp. 65-89, 1983.

[29] M.C. Fernandes, "Determining the normalization level of a relation on the basis of Armstrong's axiom", *Computer an Artificial Intelligence*, V.3, N.6, pp.495-504, 1984.

[30] N. Goodman and O. Shmueli, "Syntactic Characterization of Tree Database Schemas", *J. ACM*, V.30, N.4, pp.767-786, 1983.

[31] G. Gottlob, N. Leone and F. Scarcello, "Hypertree decompositions and tractable queries", *Journal of Computer and System Sciences*. V.64, N.3, pp.579-627, 2002.

[32] G. Gottlob, N. Leone and F. Scarcello, "The Complexity of Acyclic Conjunctive Queries", *Journal of the ACM*, V.43, N.3, , pp. 431–498, 2001.

[33] G. Gottlob, "On the size of nonredundant FD-covers", *Information Processing Letters*, V.24, N.6, pp.355-360, 1987.

[34] M. Grohe, T. Schwentick and L. Segoufin, "When is the evaluation of conjunctive queries tractable?", in *Proc.ACM Symposium on Theory of Computing*. pp.657–666, 2001.

[35] S. Hartmann and S. Link, "The implication problem of functional dependencies in complex-value databases", *Electronic Notes in Theoretical Computer Science (ENTCS)*, V.123, pp.125-137, 2005.

[36] M. Kifer, A. Bernstein and P. M. Lewis, "Database Systems: An Application-Oriented Approach", Addison-Wesley, Boston, MA, Second edition, 1272 p, 2006.

[37] H. Koehler, "Finding faithful Boyce-Codd Normal Form decompositions", *Algorithmic Aspects in Information and Management* (AAIM), Lecture Notes in Computer Science, Springer, V.4041, pp.102-113, 2006.

[38] V.S. Lakshmanan, N. Chandrasekaran, C.E. Veni Madhavan, "Recognition and top-down generation of β-ciclic acyclic database schemes", *Lect. Notes. Comput. Sci*., V.181, pp.344-366, 1984.

[39] H. Li and L. Zhou., "Use closure of relevant sets of attributes to efficiently find candidate keys", in *Proc. International Conference on Computer Science and Software Engineering*, CSSE 2008, V.1: Artificial Intelligence, December 12-14, IEEE Computer Society, Wuhan, China. pp.237-242, 2008.

[40] C.L. Lucchesi, S.L. Osborn , "Candidate keys for relations", *Jour. Of Comput. And Syst. Sci.*, N.17, pp.270-279, 1978.

[41] D. Maier, "The theory of relational database". Computer Science Press, 637p, 1983.

[42] D. Maier, "Minimum cover in the relational database model", *Jour. Of ACM*, V.27, N.4, pp.664-674, 1980.

[43] J. Makowsky, E. Ravve, "Dependency Preserving Refinements and the Fundamental Problem of Database Design", *Data and Knowledge Engineering*, V.24, N.3, pp.277-312, 1998.

[44] H. Mannila and K.J. Räihä, "On the relationship of minimum and optimum covers for a set of functional dependencies", *Acta Informatica*, V.20, pp.143-158, 1983.

[45] H. Mannila, K.J. Raiha, "The Design of Relational Databases", Addison-Wesley, xii, 318 p, 1992.

[46] S.L. Osborn, "Testing for Existance of a Covering Boyce-Codd Normal Form", *Information Processing Letters*, V.8, N.1, pp.11-14, 1979.

[47] M.T. Özsu and P. Valduriez, "Principles of Distributed Database Systems", ed. Dorling Kindersley (India) Pvt Ltd, 720p, 2006.

[48] J. Paredaens, P. De Bra, M. Gyssens and D. Van, "The Structure of the Relational Database Model", EATCS Monographs on Theoretical Computer Science. W. Brauer, G. Rozenberg, A. Salomaa, eds., SpringerVerlag, V.17, 231 p, 1989.

[49] R. Ramakrishnan and J. Gehrke, "Database Management Systems", McGraw Hill Higer Education, 906 p, 2000.

[50] A. Silberschatz, H.F. Korth and S. Sudarshan, "Database System Concepts", Sixth Edition, McGraw-Hill, 1376 p, 2010.

[51] A.M. Silva and M.A. Melkanoff, "A method for helping discover the dependencies of a relation", *Advances in Data Base Theory*, V.1, Gallaire H., Minker J.,. Nicolas J.M (eds.), Plenum Press, pp.115-133. 1981.

[52] S. Sumathi and S. Esakkirajan, "Fundamentals of Relational Database Management Systems", Springer-Verlag Berlin and Heidelberg GmbH & Co. K. 776 p, 2007.

[53] R. Taouil and Y. Bastide, "Computing proper implications", in ICCS-2001 International Workshop on Concept Lattice-Based Theory, Methods and Tools for Knowledge Discovery in Databases, Lecture Notes in Computer Science, Palo Alto, CA, USA, Springer, pp.49–61, 2001.

[54] R.E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs", *SIAM J. Comput.*, V.13, pp.566-579, 1984.

[55] B. Thalheim, "A survey on database constraints", *Reine Informatik*, I-8/1994, Fakultat fur Mathematik, Naturwissenschsften und Informatik, 23 p, 1994.

[56] B. Thalheim, "On the number of keys in relational and nested relational databases", *Discrete Applied Mathematics*, V.40, pp.265-282, 1992.

[57] J.D. Ullman, "Principles of Database Systems", Computer Science Press, New York, N. Y., Second edition, 485 p, 1982.

[58] J.D. Ullman, "Principles of Database and Knowledge-Base Systems", Vol. II: "The New Technologies", Spektrum Akademischer Verlag, 511 p, 1990.

[59] J.D. Ullman, "Principles of Database and Knowledge-Base Systems", Vol. I: "Classical Database Systems", Computer Science Press, Rockville, MD, 631 p, 1990.

[60] W. Vincent and B. Srinivasan, "A Note on Relation Schemes which are in 3NF but not in BCNF", *Information Processing Letters*, V.48, N.6, pp.281-283, 1993.
[61] J. Wang, "The Information Hypergraph Theory", Science Press, Beijing, 162p, 2008.
[62] R. Wastl, "On the Number of Keys of a Relational Database Schema", *Journal of Universal Computer Science*, V.4, N.5, pp.547-559, 1998.
[63] P. B. Worland, "An efficient algorithm for 3NF determination", *Information Science*, V.167, N.1-4, pp.177-192, 2004.
[64] M. Yannakakis, "Algorithms for acyclic database schemes", in Proc. of the International Conf. on Very Large Databases (VLDB '81) (Cannes, France), pp.82–94, 1981.
[65] С.Д. Кузнецов, "Основы баз данных", 2-е изд.,испр., Москва, БИНОМ, 484 с, 2007.

Vitalie COTELEA is Associate Professor at Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Moldova. He is the author and co-author over 90 scientific works, including one monograph and more than 10 books. His work focuses on Databases and Information Systems Design and Declarative Programming. He has graduated the Faculty of Mathematics and Cybernetics in 1974 of State University of Moldova, Chisinau. He holds a PhD diploma in Computer Science from 1988 of Kiev State University, Ukraine.