Algebraic Graph-Oriented = Category-Theory-Based

Manifesto of Categorizing DataBase Theory

Third version

Boris Cadish and **Zinovy Diskin** [1]

Laboratory for Database Design,
Frame Inform Systems,
Elizabetes Str. 23,
Riga, LV-1234, Latvia

E-mail: diskin@frame.riga.lv
Fax: +371 7 828036

December 20, 1995

Foreword. Category theory is well known as a high-level polymorphic framework suitable for specifying complex structures and formalisms. Quite unexpectedly, however, it turned out that even elementary category theory notions can be valuable in *practice* of software engineering (see [18, 9, 12]), and in the summer of 1994 we wrote a manifesto ([7], the first version of the present one) to claim our belief in the extreme fruitfulness of incorporating the category theory (CT) machinery into the database (DB) area.

That time we did not know about works by Johnson with coauthors (see [18] for references) which seem to be the first publications on the real influence of CT machinery in the DB field. So, a real mutual liking between CT and DB is about two years old. We cannot say that their relations were unclouded during these years, and now we recognize quite clearly that theoretical achievments do not directly lead to revolutions in the field of information (and any other) technologies: the normal way of changes in the latter is the way of more or less rapid evolution only more or less modulated by theoretical ideas. Nevertheless, we consider the last year as a confirmation of our previous predictions: the list of (CT+DB)-publications is actually growing (though not so rapidly as we hoped), and at a recent International Workshop ([6]) there was a special section on (CT+DB), and methodologies being developed in Sydney ([18]) and Riga ([9, 12]) seem to meet some (methodological, of course, but partly also technological) challenges in the DB area. As before, we forecast an intensive growth of CT and DB interaction and present below the third version of our manifesto in which the text is slightly revised and the bibliography list is updated.

[1] Supported by Grant 94.315 from the Latvian Council of Science

Foreword to the first version. Nowadays is the time when the distance between a computer science investigation and its practical realization is dramatically decreasing. Moreover, now we live in such an informationized world that even the distance between a declaration of an idea and its advanced elaboration is also decreasing. That is why it is the time of scientific declarations often called manifestos as, for instance, "The Object-Oriented Database System Manifesto" by M. Atkinson *et al* ([3]), "Third-Generation Data Base System Manifesto" by M. Stonebraker *et al*([23]), "A Categorical Manifesto" by J.Goguen ([15]).

So, we have also called the declaration presented below a Manifesto. It is written by a specialist in database design (who has more than a decade experience in semantic modeling and designing very large databases, and about three year experience in applying category theory concepts to his subject) and by a mathematician (who has more than a decade experience in categorial treating logical concepts and about three year experience in DB theoretical interpretations of categorial constructs). Our (three-year, as one could guess) collaboration was not easy due to obvious reasons, yet it has convinced us in unexpectedly high relevance of category theory "abstract nonsense" for such concrete things as semantic modeling, database design, management of multibase systems and database theory as a whole. At present, it seems for us that the current situation in relating category theory with DB theory and practice is very similar to the 16th century interaction of differential and integral calculi, on one hand, with mechanics and engineering on the other. Thus, being excited with our discovery, and having in mind the distinctive features of our time mentioned above, we have decided to begin propagating our observations with a declarative document in a manner of a brief manifesto. Correspondingly, theorems (obligatory for a mathematician) are replaced by slogans, and proofs - by justifications.

Acknowledgement. The ideas presented below were repeatedly discussed with Sergey Ageshin, Ilya Beylin and Vitaly Džitenov during our seminars on algebraic methods in data modeling and management. They provide a very important position of a practitioner and a theorist in Programing (and now in CT as well) without which the considerations would be much poorer. We greatly acknowledge them for their participation and patience in interpreting the "abstract nonsense" of category theory and translating it into beautifully concrete procedures.

Our special thanks go to Ami Motro, and to anonymous referees of the *NGITS'95* Workshop as well, for discussing a general picture of theory-practice interaction in the DB area.

The seventies were the time of relational data model (RDM) in the database area. RDM has formed a precise specification framework for DB theory, thus introducing "a new era ... because it provided a significant barrier between the model and its implementation, thus insulting users from a myriad of implementation details" (Hull [16]). In addition, anybody must agree with Serge Abiteboul that "a major factor in that success has been the existence of simple-to-use languages allowing the definition and manipulation of data" ([1]). On the other hand, these handy languages were based on the firm mathematical ground of relational first-order logic and model theory so that, for the seventies, RDM has appeared as a nice amalgamation of logical rigor and handiness.

Since that time new areas of applications have emerged. These applications need to manage large amounts of evolving and shared data structures connected with complex semantic relationships that scarcely can be packed in the rigid frame of flat relations; in addition, the focus has turned to modeling data as seen by the application and the user. To meet new requirements there was developed a diversity of languages and techniques for data definition and manipulation ranging from nested relational to object-oriented semantic models; in particular, there was proposed a plethora of semantic models based on handy graph-oriented structural schemas (see, *eg*, surveys [17, 16]).

An orthogonal dominant trend is connected with the paradigm of cooperative information system. Now it is evident that an information system (IS) of the next generation will appear as a large number of ISs distributed over complex computer/communication network. In addition, local ISs should be sufficiently autonomous and can significantly differ in their origin and development, hence, in their organization, architecture, data models *etc* which constitutes multilingual nature and heterogeneity of the next generation ISs as one of their chief characteristics. So, management of heterogeneity becomes a crucial issue, and the most important initial step is to develop a language for specifying heterogeneous systems (see, *eg*, [14]).

From an overall view point, a peculiarity of the situation consists in extremely strong experimental activity based on surprisingly weak formal semantic foundations. The lack of a precise specificational framework is commonly recognized (M.Atkinson *et al*[3], C.Beery [5], S.Navathe [20] *etc*), an illuminating example is the fact of invoking abstract Wittgenstein's philosophy at a conference organized by practitioners ([14]), in the conclusion report of which it was emphasized that there is a great need in languages for specifying data interoperation:

> ...It is clear that the problem of identifying and resolving semantic heterogeneity in a heterogeneous database environment is far from solved, and that *the problem itself is still at the stage of being understood.* ...Wittgenstein said long ago that if an idea was worth discussing, then it was worth having a language to discuss it. Thus, researchers need to develop languages for handling data interoperation. ...
>
> *Perhaps, one of the most difficult problems in developing integrated systems is in figuring out what schemas, data, and application code mean.*[2]

In this context, what is required first of all is a general language allowing to specify basic DB theory concepts like database schema, instance, query, view, update *etc* in a way independent of any specific data model (rather than a single universal data model capturing all other data models).

[2] Italic is ours

In fact, nowadays a DB specialist is forced to alter between *Scylla* of easy-to-use but scarcely formalized graph-based approaches, and *Charybdis* of logically clear and perfect but inflexible relational languages. Due to the evident tendency of any practitioner towards the former, what exists out in the field today is "cookbook approaches ... poorly formed and having too many subjective human considerations" (Navathe [20]). As a result, specifications are mixed with implementations and a procedural description is often the problem statement and its resolving algorithm simultaneously (so that any discussion of success and optimality of the algorithm in question is cancelled).

Thus, the field strongly needs a unified data and metadata definition/manipulation language combining evidence and user-friendliness of graph-oriented object semantic models together with formal rigor and clearness of relational models. In addition, this language should be suitable for specifying dynamic structures as well, and, finally, be easy for manipulating – the last feature is often closely connected with natural algebraizability of a language. These requirements can be summarized under the title of *an algebraic graph-based formalized language for specifying structure and dynamics of data and data manipulation.*

The main slogan. *We assert that in order to deal properly with elaboration and development of such a language, it is "necessary and sufficient" to incorporate methodology and machinery of the mathematical category theory into the DB area.*

Justification. While RDM amounts to a family of algebras, calculi and semantic structures based on sets, the language announced above can be thought of as a family of algebras, calculi and semantic structures based on graphs. The former was supported by the ordinary first-order logic methodology and machinery. The latter exists scarcely supported by an *ad hoc* mathematics created by DB theorists. The point is that algebra and logic over graphs require quite another approach than their ordinary (over sets) prototypes, and it is difficult to handle them properly without a corresponding preliminary framework and intuition. However, a mathematical framework for algebraic graph-based logic already exists and, moreover, is well elaborated: this is the framework offered by categorial logic and categorial model theory as they were developed in the mathematical category theory (see [24] for references).

Indeed, it was discovered in [18, 8] and developed in [9, 12] (see also [22] for an independent work in the same direction) that various semantic diagrams like *eg*, ER-diagrams which are widely used by practitioners and theorists can be naturally considered a special kind of structures being studied in categorial logic since the late sixties, and called there *sketches* (a standard reference is [4]). However, while category theorists prefer to deal with a few fixed diagram properties (commutativity, (co)limitness), it was shown in [12] that one can safely enjoy the possibility to set arbitrary signatures of diagram properties. In addition, it is reasonable to use anywhere is possible the algebraic treatment of diagram properties via introducing diagram operations: a certain part of such an algebraizable diagram is presented as the result of applying some operation to the other part (in a full similarity with, say, presenting predicate $\oplus(x, y, z)$ by $z = x + y$). In this respect the sketch framework we are speaking about is similar to FOL whose theories can be written down for arbitrary vocabularies of predicate and operation symbols that provides sufficient flexibility for practical using.

So, a conceptual database schema can be considered as a *generalized* sketch, that is, a graph in which some diagrams (of nodes and arrows) are labeled by special symbols taken from a predefined signature. In addition, queries against the schema are reduced to *diagram operations* which are denoted by labeling corresponding diagrams with special *operation symbols* taken from the signature. In other words, we have a unifying graphical mechanism for denoting both basic data and queries against them.

Moreover, if one extends (more precisely, closes) a given semantic diagram (considered as a DB schema) with all the new nodes and arrows denoting derived data which can be retrieved from the DB instances over the diagram, then one will come to a kind of another familiar structure of categorial logic called *topos* which is also being studied intensively since sixties. Thus, general graph-based logic can be developed as a logic of generalized sketches, and corresponding algebras are algebras of diagram operations over sketches ([12, 13, 11]).

As for metadata specification languages, we emphasize that what is required first of all is rather a possibility to speak and specify such concepts as database schema, database instance, query, view, update *etc* in a data-model-independent way than a single universal data model capturing all others data models. At first glance it seems impossible to speak about database schemas without any description of what kind of things they can be because it appears to be a kind of *substantial speaking about nothing*. However, actually a methodological framework and machinery for such a strange thing are already developed in the mathematical category theory (which was even entitled by *abstract nonsense* during its early days). So, there emerges an important task of categorial formulating the DB theory conceptual framework (an initial step was made in [10, 13].

Quite briefly, the essence of category theory consists in characterizing internal structures of objects through mappings between them, thus constituting some universe of discourse as a collection of objects - nodes, together with a collection of inter-object mappings - arrows - closed under composition, that is, as *a category* in the formal sense. Then it is normally turned out that surprisingly many derived objects of interest can be also characterized or identified via mappings and related machinery, and such a situation became paradigmatic in various applications of category theory to mathematics, logics and computer science. Thus, as soon as one has characterized the (internal) structure of objects in question externally via arrow diagrams, (s)he can immediately adopt the machinery of category theory. So, the latter offers a polymorphic framework for specifying and manipulating objects of arbitrary nature in a unified algebraic way. [3]

This way of regarding things can be called *arrow thinking* in contrast to usual thinking in terms of sets and elements. It is remarkable that both data and metadata specifications can be formalized in the same way via the arrow logic of category theory.

The reasons considered above can be regarded as demonstrating "the necessity" of the sketch machinery. To justify "sufficiency", we note that a chief categorial construction of *topos* can be specified by a sketch (is *sketchable* as a categorial logician would say). On the other hand, constructive set theory as well as higher-order logic (type theory) can be interpreted in toposes (see, *eg*, [19]).

Moreover, it is also known that similarly to nested-relational algebraization of higher-order logic, any topos can be presented as a graphical (sketch) algebra over the corresponding graph. So, the full power of higher-order logic including its algebraizability can be simulated by sketches (and conversely). Actually, this means that *everything that can be specified formally* [4] *can be specified formally by sketches* (yet the latter are graph-based and much more handy).

Corollary. We can distinguish two main streams of incorporating CT into the DB theory and DB area.

The first one consists in developing a categorial theory of metadata modeling, *ie*, in building an abstract formal framework providing the possibility to describe main data modeling (DM) constructs (such as schema, instance, view, query, update etc.) in a way independent on any concrete DM-paradigm. Actually it is a matter of approaching the problem formulated in the [14] and quoted on p.1. Initial steps of the category theory solution we suggest were presented in [8, 10, 13].

The second way of incorporating category theory is connected with a new data modeling paradigm based on a well-known in categorial logic notion of sketch. The goal is to build a

[3] For example, one can define the operation of integrating abstract entities without any specifying what these entities are, so that merging objects (in the OO sense), view integration, schema integration *etc* can be described via the same (or similar) arrow construction(s). The only precondition one should impose is to assume that besides these abstract entities themselves there are defined arrows between them, and for computer implementation of such "an abstract nonsense" these abstract arrows must be concretely specified and put into computer (see [21]).

[4] in the sense adopted in the modern mathematics

graph-based specification paradigm similar and quite parallel to the RDM paradigm in being

- *fully formalized* (on both syntax and semantics levels),
- *algebraizable* (like predicate calculi are algebraizable by relational algebras) and
- *universal* (in the sense that anything that can be specified formally could be specified via the paradigm languages as well).

The sketch approach satisfies all these criteria in a quite immediate way. [5] However, there is a lot of work to do in elaborating concrete specification techniques, optimal operational semantics of sketch operations and computer implementation of sketch specification languages. In particular, it was demonstrated in [9, 12, 13] that the sketch paradigm of data modeling is perfectly suited for view integration, since, in fact, it eliminates some kinds of structural conflicts at all, and explains how to handle others via diagram operations over sketches. Thus, sketches emerge as a powerful tool for resolving structural conflicts that always was a challenge for view integration methodologies. Moreover, in practice of (manual) data structuring sketches proved so easy to draw and to interpret, that a way of implementing a prototype DB-designer workbench capturing the whole diversity of semantic data models in a unified manner became clear very soon ([2], cf. also [21]).

Remark. There are already known several attempts of categorial modeling DB theory concepts (the list in Appendix is far from being complete). As a rule, they deal with one or another aspects of DB modeling and do not constitute a supporting framework for metadata modeling on the whole. As for the level of data modeling, it seems that in the mentioned works there are suggested some new kinds of semantic models which has their own advantages (maybe, strong) and disadvantages (maybe, small) but, for us, the focus of the problem is different. Namely, it consists in building a complete and consistent graph-based specification framework similar to the relational one with respect to points outlined above.

[5] In [5] Beeri asks: "Can we use the same query language paradigm, or framework, uniformly in any system, irrespective of the specific type system it uses for its data?" Sketches do provide a positive answer: the language of diagram operations over sketches is just a language of the kind Beeri described

Conclusion: The Manifesto

The essence of category theory consists in characterizing internal structures of objects through mappings between them, thus constituting some universe of discourse as a collection of objects together with inter-object mappings so that many derived objects of interest can be also characterized or identified via mappings and related machinery.

Such a situation became paradigmatic in various applications of category theory to mathematics, logics and category theory itself. It constitutes general intention, internal harmony and external justification of category theory, whose numerous achievements teach us that actually, to define properly the universe we are going to deal with, it is necessary to define mappings between objects of that universe. It seems that the value of this lesson is not yet appreciated in the DB area : while there are precise descriptions of what is a relational (database) schema, or a semantic schema of a definite kind, or data structured according to some schema, there has been told nothing about what is a mapping of relational schemas, a mapping of ER-schemas etc.

We firmly believe that

- A well elaborated precise, clear and flexible specification language provides a half of success in design and implementation of data management systems;
- Category theory language has the following unique bundle of advantages:
  - *evidence*, owing to its graph-based nature;
  - *easy of manipulating*, owing to its algebraic nature;
  - *rigor*, owing to its mathematical ground;
  - *unifying power*, owing to its abstract polymorphic nature;
  - capability to model *dynamic structures and concurrency*;
  - *universality*, owing to ability of simulating any formal specification language;
- The chief category theory paradigm of defining internal structure of objects by characterizing an external adjoint structure of arrow diagrams (*arrow or diagram thinking*) is just the specification methodology required in the DB area;
- Category theory machinery just offers those kinds of formalisms that DB theory needs, namely, those which are algebraic and graph-based;
- Categorical logic and model theory can and must play for semantic and OO data models the role quite similar to that one ordinary mathematical logic and model theory play for relational models.

Initial steps are made, the list of (CT+DB)-references is growing and we predict its exponential increase in the nearest future. We conjecture that in some years "the abstract nonsense" of category theory will become a basic mathematical discipline necessary for a DB designer very much like ordinary linear algebra and calculus are necessary for a engineer-designer in mechanics.

References

[1] S. Abiteboul. Towards a deductive object-oriented database language. *Data and Knowledge engineering*, 5:263–287, 1990.

[2] S. Ageshin, I. Beylin, B. Cadish, and Z. Diskin. Outline of AGO: Algebraic graph-oriented approach to specification and imlementation of data management systems. (in preparation), 1995.

[3] M. Atkinson, F. Bancilhon, D. DeWitt, K. Ditrich, D. Maier, and S. Zdonik. The object-oriented database system manifesto. In *The first Conference on Deductive and Object-Oriented Databases, Kyoto, Japan, 1989*, 1989.

[4] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice Hall International Series in Computer Science, 1990.

[5] C. Beery. New data models and languages – the challange. In *PODS'92*, pages 1–15, 1992.

[6] J. Bubenko, L. Kalinichenko, and Yu. Zhuravlev, editors. *Advances in DataBases and Information Systems, ADBIS'95*, Proc.2nd Int.Worlshop, Moscow ACM Chapter, Moscow(Russia), 1995.

[7] B. Cadis and Z. Diskin. Algebraic Graph-Oriented = Category Theory Based. Manifesto of categorizing database theory. Technical Report 9406, Frame Inform Systems, Riga,Latvia, 1994. (On ftp: //ftp.cs.chalmers.se/pub/users/diskin/manifest.*).

[8] Cadish and Z. Diskin. Algebraic graph-oriented approach to view integration. Part I: Specification framework and general strategy. Technical Report 9301, Frame Inform Systems/LDBD, Riga, Latvia, 1993.

[9] B. Cadish and Z. Diskin. Algebraic graph-based approach to management of multibase systems, I: Schema integration via sketches and equations. In *Next Generation of Information Technologies and Systems, NGITS'95*, 2nd Int.Workshop, pages 69–79, Naharia (Israel), 1995. Extended abstract is available by ftp: //ftp.cs.chalmers.se/pub/users/diskin/ngits95.*).

[10] Z. Diskin. Algebraic graph-based approach to management of multibase systems, II: Mathematical aspects of schema integration. Technical Report 9502, Frame Inform Systems/LDBD, Riga,Latvia, 1995. (On ftp: //ftp.cs.chalmers.se/pub/users/diskin/tr9502.*).

[11] Z. Diskin. Formalization of graphical schemas: General sketch-based logic vs. heuristic pictures. In *10th Int.Congress of Logic,Methodology and Philosophy of Science.* Volume of Abstracts, page 401, Florence (Italy), 1995. (On ftp://ftp.cs.chalmers.se/pub/users/diskin/lmps95.*).

[12] Z. Diskin. Formalizing graphical schemas for conceptual modeling: Sketch-based logic vs. heuristic pictures. Int.Symp.*"Knowledge Retrieval, Use and Storage for Efficiency, KRUSE'95"*, 1995

[13] Z. Diskin and B. Cadish. Databases as graphical algebras: Algebraic graph-based approach to data modeling and database design. Submitted for *Algebraic Methodology and Software Technology, AMAST'96* (On ftp: //ftp.cs.chalmers.se/pub/users/diskin/amast96.*), 1995.

[14] P. Drew, R. King, D. McLeod, M. Rusinkiewicz, and A. Silberschatz. Report on the workshop on semantic heterogeneity and interoperation in multidatabase systems. *SIGMOD Record*, 22(3):47–56, 1993.

[15] J.A. Goguen. A categorical manifesto. Technical report, SRI International, SRI-CSL-89-8, 1989.

[16] R. Hull. Four views of complex objects: A sophisticate's introduction. In *Nested relations and complex objects in Databases*, LNCS'361, pages 87–116, 1987.

[17] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[18] M. Johnson and C.N.G. Dampney. On the value of commutative diagrams in information modeling. In *Algebraic methodology and software technology, AMAST'93*, Proc.3rd Int.Conf., Twente (Holland), 1993.

[19] J. Lambek and P. Scott. *Introduction to higher order categorical logic.* Cambridge University Press, 1986.

[20] S. Navathe. The next ten years of modeling, methodologies and tools. In *ER'92*, number 645 in LNCS, (Karlsrue, Germany), 1992.

[21] D.A. Nelson and B.N. Rossiter. Prototyping a categorical database in P/FDM. In *Proc.of the Moscow ACM Chapter 2nd Int.Worlshop* Advances in DataBases and Information Systems, ADBIS'95 *(Moscow,Russia)*, pages 247–258, 1995.

[22] F. Piessens and E. Steegmans. Canonical forms for data specifications. In *Proc. Computer Science Logic'94*, Springer LNCS No.933, pages 397–411, 1994.

[23] M. Stonebraker, L. Rowe, B. Lindsay, J. Gray, M. Carey, M. Brodie, P. Bernstein, D. Beech, and (The Committee for Advanced DBMS Function). Third-generation data base system manifesto. In *Proc.IFIP WG 2.6 Conf. on Object-Oriented Databases (DS-4)*, 1990.

[24] C. Wells. Sketches: Outline with references. On ftp //ftp.cwru.edu/math/wells.*.