DATA MODELS IN VIEW OF COOPERATIVE DIALOGUE SYSTEMS

Melita Hajdinjak

University of Ljubljana, Faculty of Electrical Engineering
1000 Ljubljana, Tržaška 25, Slovenia

melita.hajdinjak@fe.uni-lj.si (Melita Hajdinjak)

Abstract

Databases and information systems that are based on the widespread relational data model are often hard to use because they do not explicitly attempt to cooperate with the user. In dialogue systems or natural-language interfaces, for instance, the most vital cooperative-answering technique is *query relaxation*, which expands the scope of a query by relaxing the constraints implicit in the query in order to capture neighbouring or possibly relevant information. Thus, dialogue systems, particularly information-providing dialogue systems, require a cooperative data model. Since none of the most common data models, e.g., relational data models, deductive data models, and object-oriented data models, enable query relaxation, we refer to a very promising data model, which results from a natural generalization of the relational data model from set-theoretic algebra of relations to a category-theoretic setting. This data model includes the fundamental relational data model, and it is able to represent all the basic relational operations in many richer-structured environments. Apart from the standard domain orderings such as numerical and alphabetical orderings, it comprises heterogeneous semantic orderings, and thereby supports query relaxation.

Keywords: Data models, Dialogue systems, Cooperative behaviour, Query relaxation, Relational databases.

Presenting Author's Biography

Melita Hajdinjak is a Fellow Researcher and Teaching Assistant at Faculty of Electrical Engineering, University of Ljubljana, Slovenia. She has a B.Sc. in Mathematics from Faculty of Mathematics and Physics, an M.Sc. and Ph.D. in Electrical Engineering from Faculty of Electrical Engineering, and is currently working toward her Ph.D. in Mathematics. Her area of interest includes knowledge representation, natural-language dialogue systems, human-machine communication, and theoretical computer science. For the bachelor work she has attained the Franc Močnik Award, and for the doctoral disseratation she has attained the Pomurje Research Award 2006. She is fluent in German and English and has elementary knowledge of French.

ISBN 978-3-901608-32-2

1 Introduction

As already emphasized by Codd [1], theoretical database studies form a fundamental basis for the development of homogeneous and sound database management systems (DBMS), and consequently of information-providing natural-language dialogue systems. The formal database studies use either a specially developed database theory or other formal theories, such as mathematical logic, as their framework. The general paradigm of a database system (within an information-providing natural-language dialogue system) is as follows:

1. The database system accepts a query written in a formal query language (e.g., SQL or Datalog),
2. The database system evaluates the query against the database (in the relational case, a collection of relational tables).
3. The database system returns the complete set of tuples satisfying the query, i.e., the answer set.

In the early history of database science, it was accepted that there existed a unifying data model that could be shared by most, if not all, data, and that a key task of database scientists was to identify this model. At some point between the entrenchment of the relational model [2] and the advent of object-oriented databases database scientists and database users stopped looking for a unifying model and became interested in domain-specific modeling. Some of them have considered domain-specific modifications of the relational data models [3, 4, 5, 6, 7, 8], others have embraced new data models [9, 10, 11, 12].

The relational model, whose theoretical basis is in set theory and first-order predicate logic, has grown slowly in importance since its exposition by Codd [2] in 1970, to the point where it has been generally the model of choice for the implementation of new databases. As understood by Ullman [13], the perhaps most important reason for the model's popularity is the way it supports powerful, yet simple and declarative languages with which operations on data are expressed. However, while powerful in theory, the relational model - with its data tables of rows and columns - is beginning to show its age and limitations in practice. Basic relational systems are not always suitable for one or more of the following reasons [12]:

- More complex data structures are needed for modeling non-traditional applications than the simple relational representation.
- New data types are needed in addition to the basic numeric and character string types.
- New operations and query language constructs are necessary to manipulate the new data types.
- New storage and indexing structures are needed.

Prior to the rise of the relational model, there were powerful databases with more flexible data structures than two-dimensional tables. These pre-relational databases, however, became frozen in time. Although theoretically capable of handling a wide range of data types, multi-valued databases lack modern features. Some pre-relational databases, however, have evolved into post-relational databases. The flexible physical structures they use store data simultaneously in accordance with both the relational and object-oriented data models. Tables and objects become interchangeable at the database layer, enabling support for SQL (i.e., Standard Query Language) as a universal mechanism for sharing data between different systems and applications. Notwithstanding, Tebbutt [14] assumed that it is unlikely that post-relational databases would displace relational databases. There are simply too many applications deployed on relational platforms, and many of these applications will never hit the limitations of the relational data model.

2 Relational data model

The relational data model represents the database as a collection of *relations*. When a relation is thought of as a table of values, each row in the table represents a collection of related data values, i.e., a fact that typically corresponds to a real-world entity or relationship. The data type describing the types of values that can appear in each column is represented by a *domain* of possible values. In the formal relational-model terminology, a row is called a *tuple*, a column header is called an *attribute*, and the table is called a *relation*. In a relational database, there will typically be many relations, and the tuples in those relations are usually related in various ways.

There are two formal languages of for the relational data model: the *relational algebra* and the *relational calculus* [13, 12]. The relational algebra subsumes the basic set of operations to manipulate the relational database, i.e.,

- five basic operations on relations: *Cartesian product* $\times$, *projection* π, *selection* σ, *union* $\cup$, and *set difference* $-$,
- several additional operations, such as *natural join* $\bowtie$ or *intersection* $\cap$.

The algebra operations produce new relations, which can be further manipulated using operations of the same algebra. A sequence of relational algebra operations forms a *relational algebra expression*, whose result will also be a relation that represents the result of a database query (or retrieval request).

The relational algebra has three main advantages over non-relational data models [8]:

- From the point of view of *usability*, the model has a simple interpretation in terms of real-world concepts, i.e., the essential data structure of the model

ISBN 978-3-901608-32-2

is a relation, which can be visualized in a tabular format.

- From the point of view of *applicability*, the model is flexible and general, and can be easily adapted to many applications.
- From the point of view of *formalism*, the model is elegant enough to support extensive research and analysis.

Hence, the relational data models have gained acceptance from a broad range of users, they have gained popularity and credibility in a variety of application areas, and they facilitate better theoretical research in many fundamental issues arising from database query languages and dependency theory.

Whereas the relational algebra defines a set of operations for the relational data model, the relational calculus provides a higher-level declarative notation for specifying relational queries. The relational calculus is important because it has a firm basis in mathematical logic and because the SQL for RDBMs (i.e., Relational Database Management Systems) has some of its foundations in the tuple relational calculus. However, SQL also incorporates some of the operations from the relational algebra and its extensions [12].

Note, relational dialogue systems are often hard to use, even in traditional applications, because they do not explicitly attempt to cooperate with the user. This is to say, a user may need more information, or might even need different information, than the query requests. *Cooperative behaviour* or *cooperative answering* [15, 16] plays an important part in natural-language interfaces and dialogue systems built for the purpose of studying natural-language exchanges between users and computers. It has been shown by Hajdinjak and Mihelič [17] that here the most vital cooperative-answering technique is *query generalization* or *query relaxation*, which expands the scope of a query by relaxing the constraints implicit in the query (e.g., types of constants and variables, predicate relations, and join dependencies across literals in the original query) in order to capture possibly relevant information [18]. See section 4.

3 Alternative data models

Relational databases may be considered a forerunner of logic in databases. The use of logic for knowledge representation and manipulation is primarily due to the work of Green [19], i.e., his work was the basis of various studies that led to question-answering systems, which are concerned mainly with a highly deductive manipulation of a small set of facts, and thus require an inferential mechanism provided by logic. The integration of logic programming and relational database techniques has led to the active research area of deductive databases [20, 21], i.e., databases whose query language and (usually) storage structure are designed around a logical model of data. This combines the benefit of the two approaches, such as representational and operational uniformity, reasoning capabilities, recursion, declarative querying, efficient secondary storage access.

The function symbols of Prolog [22], which are typically used for building recursive functions and complex data structures, have not been found useful for operating over relational databases made up of flat relations. As a result, a restricted form of Prolog without function symbols is called Datalog (with negation), with a well-defined declarative semantics based on the work in logic programming, has been widely accepted as the standard deductive database language [13, 9]. The deductive database management systems do, however, share with the relational systems the important property of being declarative, i.e., of allowing the user to query or update by saying what he or she wants, rather than how to perform the operation.

An important distinction between the relational data model and Datalog is that in Datalog, there are two ways relations can be defined. A predicate whose relation is stored in the database is called an *extensional database* (EDB) relation, while one defined by logical rules is called an *intensional database* (IDB) relation. In the relational model, all relations are EDB relations.

In the last two decades, a number of deductive database systems or prototypes based on Datalog have been reported. For a survey see [21]. Deductive database systems have been used in a variety of application domains including scientific modeling, financial analysis, decision support, language analysis, parsing, and various applications of transitive closure such as bill-of-materials and path problems. They are best suited for applications in which a large amount of data must be accessed and complex queries must be supported.

Ceri et al. [9] stated in 1989 that one of the major challenges that Datalog research had still to meet is to convince the knowledge base community of its practical merits. The weaknesses of Datalog have been, however, indicated as follows:

- Very few applications have been shown which can take full advantage of Datalog's expressive power.
- Datalog is not considered as a programming language, but rather as a "pure" computational paradigm.
- Datalog does not compromise its clean declarative style in any way. Sometimes it is even required that the programmer takes control on inference processing, by stating the order and method of execution of rules.
- Datalog systems have been considered as close worlds, which do not talk to other systems.

However, Datalog, in contrast to relational algebra, enables some cooperative behaviour, i.e., it supports the following cooperative-answering techniques (see section 4):

- evaluation of presuppositions in queries,
- detection and correction of misconceptions in queries,
- formulation of intensional answers.

Nevertheless, Datalog does not support *query relaxation* (i.e., generalization of queries and of responses), which is the most vital cooperative-answering technique in information-providing dialogue systems.

Furthermore, the emergence of object-oriented programming languages in the 1980s and the need to store and share complex-structured objects led to the development of object-oriented databases [10, 12]. Initially, they were considered a competitor to relational databases, since they provided more general data structures. They also incorporated many of the useful object-oriented paradigms, such as abstract data types, encapsulation of operations, inheritance, and object identity. However, the complexity of the model and the lack of an early standard contributed to their limited use.

Object-oriented databases are now mainly used in specialized applications, such as engineering design, multimedia publishing, design and manufacturing systems, financial portfolio risk analysis systems, telecommunications service applications, world wide web document structures, and hospital patient record systems. These newer applications have requirements and characteristics that differ from those of traditional business applications, such as more complex structures for objects, longer-duration transactions, new data types for storing images or large textual items, and the need to define nonstandard application-specific operations. The object-oriented approach offers the flexibility to handle some of these requirements without being limited by the data types and query languages of the relational database systems. A key feature of object-oriented databases is, however, the power they give the designer to specify both the structure of complex objects and the operations that can be applied to these objects.

4 Achieving cooperativity

The cooperative-answering techniques that have grown out of research in three areas in which question-and-answer discourses arise, i.e., natural-language interfaces and dialogue systems built for the purpose of studying natural-language exchanges between users and computers, databases, and logic programming in deductive databases, can be separated into five categories. The techniques in each category are differentiated by the following capabilities [16]:

1. consideration of specific information about a user's state of mind, i.e., considering user beliefs in order to anticipate user expectations,
2. evaluation of presuppositions in a query, e.g., if the query fails (the answer is no), a cooperative answer should be an explanation of why the query fails, exposing any false presuppositions,
3. detection and correction of misconceptions in a query, i.e., whenever the user asks a query that cannot have an answer, the system should infer the probable mismatches between the user's view of the world and the knowledge in the knowledge base, and a cooperative answer should contain a correction to rectify this mismatch,
4. formulation of intensional answers, i.e., rewrites of the concrete answer that teach the user about the structure of the database and of the domain, that help to clean up misconceptions, or that are, eventually, more succinct than concrete answers (important when databases contain huge stores of data),
5. generalization of queries and of responses, i.e., the scope of the query is extended so that more information can be gathered in the answer.

In the relational database community, Chu, Chen, and Lee [23] have explored an abstraction/refinement method, a cooperative-answering technique of the last category, for providing related answers to the original query. A query is abstracted into a more general query that is then refined into a set of new queries to be evaluated against the database. The abstraction and refinement rely on the database having explicit hierarchies of the relations and of the terms in the domain, called the *type abstraction hierarchy*. A query rewrite is accomplished by replacing relations and terms from the query with corresponding relations and terms from higher in the hierarchy. This resulting query is considered more general than the original query. This cooperative method is called *query relaxation*.

Query relaxation is a general approach to seek additional answers to a query that may or may not be of direct interest to the user. However, it has been shown by Hajdinjak and Mihelič [17] that in information-providing dialogue systems relaxation is the crucial cooperative-answering technique leading to user satisfaction. This was the results of an evaluation of dialogue data from two Wizard-of-Oz (WOZ) experiments [24]. The data was evaluated with the PARADISE evaluation framework [25], which was recently proposed as a potential general methodology for evaluating and comparing different versions of spoken-language dialogue systems. The study introduced the dialogue costs *database parameters*, which reflected the structure of the database and the cooperativity of the dialogue system, and it confirmed the significance of relaxed answers directing the user to select relevant, available data.

Note, none of the above mentioned data models, i.e., relational data models, deductive data models, and object-oriented data models, enables query relaxation. Persuaded that many applications will never reach the limitations of the widespread relational data model, which is based on a well-established set-theoretic formalism, we claim that the only desirable data model enabling query relaxation is a natural generalization of the relational data model. Such a generalization should, in

ISBN 978-3-901608-32-2

addition to the introduction of cooperative behaviour into relational databases, be fundamental enough to unify significant classes of different specialized applications and it should have a strong and effective theoretical formalism. Hence, it should maintain the abovementioned desirabilities (i.e., usability, applicability, formalism).

There have been several attempts to introduce cooperativity into relational databases [3, 4, 5, 6, 7, 8]. However, none of them do, in our opinion, fulfill these requests. The first mentionable attempt to produce a proper generalization of the relational data model is due to Hajdinjak [26]. The strong theoretical formalism of the generalized relational data model in question arises from *category theory* [27], and it provides a sound basis for the investigation of new, possible applications. Category theory is a relatively new and powerful field of mathematics with already-recognized capabilities for providing an effective and natural formalism for relational [28, 29] and object-based databases [30]. However, because of the widespread tendency to produce new formalisms, the full potential of category theory in database modeling has not been realized to any great extent.

The proposed categorical generalization of the relational data model [26, 31] comprises the fundamental relational data model, and it is able to represent all the basic relational operations in many richer-structured environments. Apart from the standard domain orderings such as numerical and alphabetical orderings, this model comprises heterogeneous semantic orderings, and thereby supports query relaxation.

5 Conclusion

We have exposed the advantages and the disadvantages of the relational data model, which has gained popularity, credibility, and acceptance from a broad range of users. We have stated that the relational data model is inappropriate for cooperative information-providing dialogue systems since it does not support query relaxation.

While this article has dealt with cooperative behaviour in databases and information systems, it, at least partly, returned to the idea of a unifying data model. We have claimed that the first mentionable attempt to produce such a model was due to Hajdinjak [26]. This data model, gained as a generalization of the relational data model, introduces cooperative behaviour into relational databases, it is fundamental enough to unify significant classes of different specialized applications and it has a strong and effective theoretical formalism.

6 References

[1] E. F. Codd. Relational database: A practical foundation for productivity. *Communications ACM*, 25(2):109–117, 1982.

[2] E. F. Codd. A relational model of data for large shared data banks. *Communications ACM*, 13(6):377–387, 1970.

[3] H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[4] S. J. Thomas and P. C. Fischer. Nested relational structures. In P. C. Kanellakis, editor, *Advances in Computing Research III, The Theory of Databases*, pages 269–307, Greenwich, UK, 1986. JAI Press.

[5] A. Motro. Vague: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, 1988.

[6] A. Motro. Flex: A tolerant and cooperative user interface to databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):231–246, 1990.

[7] P. Buneman, P. Jung, and A. Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science*, 9(1):23–55, 1991.

[8] W. Ng. An extension of the relational data model to incorporate ordered domains. *ACM Transactions on Database Systems*, 26(3):344–383, 2001.

[9] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.

[10] W. Kim. *Modern Database Systems: The Object Model, Interoperability, and Beyond*. ACM Press and Addison-Wesley, New York, 1995.

[11] D. Raymond. *Partial Order Databases, Ph.D. thesis*. Department of Computer Science, University of Waterloo, Waterloo, Canada, 1996.

[12] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, Fourth Edition*. Pearson Education, Inc., Boston, 2004.

[13] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, Inc., Rockville, Maryland, 1988.

[14] D. Tebbutt. Driven to consider: database alternatives in a post-relational world. *Australian developer*, pages 50–51, 2003.

[15] F. Cuppens and R. Demolombe. Cooperative answering: A methodology to provide intelligent access to databases. In L. Kerschberg, editor, *Proceedings of the 2nd International Conference on Expert Database Systems*, pages 621–643, Virginia, USA, April 25-27 1988. George Mason University, Fairfax, Benjamin/Cummings.

[16] T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1(2):123–157, 1992.

[17] M. Hajdinjak and F. Mihelič. The paradise evaluation framework: Issues and findings. *Computational Linguistics*, 32(2):263–272, 2006.

[18] T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a platform of cooperative answering. *Journal of Intelligent Information Systems*, 1(3/4):293–321, 1992.

[19] C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer

and D. Michie, editors, *Machine Intelligence 4*, pages 183–205, New York, 1969. Elsevier-North Holland.

[20] H. Gallaire and J. Minker. *Logic and Databases*. Plenum Press, New York, 1978.

[21] R. Ramakrishnan and J. D. Ullman. A survey of research on deductive database systems. *Journal of Logic Programming*, 23(2):125–149, 1995.

[22] I. Bratko. *Prolog Programming for Artificial Intelligence, Third Edition*. Addison Wesley, Harlow (England), 2000.

[23] W. W. Chu and Q. Chen. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):738–749, 1994.

[24] M. Hajdinjak and F. Mihelič. Conducting the wizard-of-oz experiment. *Informatica*, 28(4):425–430, 2004.

[25] M. A. Walker, D. Litman, C. A. Kamm, and A. Abella. Paradise: A general framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, pages 271–280, Madrid, Spain, July 7-12 1997. Association of Computational Linguistics.

[26] M. Hajdinjak. *Knowledge Representation and Evaluation of Cooperative Spoken Dialogue Systems, Ph.D. thesis*. Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, 2006.

[27] S. Mac Lane. *Categories for the Working Mathematician, Second Edition*. Springer Verlag, New York, 1998.

[28] A. Islam and W. Phoa. Categorical models of relational databases i: Fibrational formulation, schema integration. In M. Hagiya and J. C. Mitchell, editors, *Lecture Notes in Computer Science 789, Theoretical Aspects of Computer Software*, pages 618–641, Berlin, 1994. Springer Verlag.

[29] M. Johnson, R. Rosebrugh, and R. J. Wood. Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories*, 10:94–112, 2002.

[30] B. N. Rossiter, D. A. Nelson, and M. A. Heather. *The Categorical Product Data Model as a Formalism for Object-Relational Databases, Computing Science Technical Report no. 505*. University of Newcastle upon Tyne, Newcastle, UK, 1994.

[31] M. Hajdinjak, A. Bauer, and F. Mihelič. A cooperative categorical generalization of relational algebra. *Sent to Transactions on Knowledge and Data Engineering*.

ISBN 978-3-901608-32-2