New Hope on Data Models and Types: Report of an NSF-INRIA worksop

Serge Abiteboul[1], Peter Buneman[2], Claude Delobel[3],
Richard Hull[4], Paris Kanellakis[5], Victor Vianu[6]

Abstract: In May, 1990, a small workshop was held in New Hope, Pennsylvania, to discuss the fundamental issues raised by continuing work on the interface between databases and programming languages. Four topics were addressed: new directions stemming from object-oriented data models, contributions of type theory to database programming languages (DBPLs), applications of logic to DBPL issues, and DBPL implementations.

This workshop was organized under the auspices of the INRIA-NSF program, *Languages for Databases and Knowledge Bases.*

1 Introduction

The boundary between databases and programming languages is continuing to erode. The need to combine database and programming language functionalities stems primarily from the increased complexity of database applications, and the desirability of providing persistence for data structures in programming languages. Indeed, recent object-oriented database (OODB) systems provide important examples of incorporating into databases a number of behavioural constructs taken largely from programming languages, and research into persistent programming languages has experimented with incorporating some database capabilities into programming languages.

The research efforts to date have been largely successful in their stated goals. Equally important, they have highlighted some of the more profound questions raised by the attempt to combine programming and database functionalities. One such question concerns the tension between the programming language notion of *type* and the database (and object-oriented programming language) notion of *class*: types describe the structure of programming values, while classes have both a structural component and an associated *extent* or set of currently active members. Two fundamental problems are: (1) it is unclear which extent(s) should be associated with a class or type; and (2) static type-checking paradigms do not apply to systems which allow objects to "migrate" between classes, and hence to change type.

A second fundamental issue concerns the development of formal foundations for database programming languages (DBPLs). For example, the relational model enjoys a close connection with mathematical logic: a number of research contributions have resulted from this connection, including, for example, the development of declarative query languages (which in turn influenced the development of SQL), and the current activity in deductive databases. The connection between logic and OODB models is not as immediate. For example, many OODB models incorporate the set structure and thus have second-order characteristics (in the database sense of the term). Also, a central component of databases concerns a *state* which changes over time, and a key contribution of the OODB models is the incorporation of behavior (including update) at a fundamental level in the schema. In contrast, the logic tools used to date on the relational model have been largely independent of updates. Indeed, although the community has not agreed on what object identity

1: I.N.R.I.A.; 2: U. Pennsylvania; 3: LRI, Altaïr; 4: USC; 5: Brown U.; 6: UCSD.

is, its interaction with persistence and updating is a central aspect. A second approach to providing foundations for the OODB models stems from functional programming languages. These languages have constructs matching or at least close to many OODB constructs, and offer substantial typing disciplines. Again, however, functional languages – even though some of them have imperative features – do not appear to have given sufficient prominence to persistent changes of state.

These and related issues formed the focus of the workshop. Although not exhaustive in its coverage, the workshop raised a number of fundamental issues in the areas of data models and types, and discussed some promising directions for resolving them. The workshop was organized around four themes:

(a) Issues raised by current OODB models and systems;

(b) Contributions of type theory to DBPLs;

(c) The use of logic to provide foundations and specific results; and

(d) Practical issues raised in the area.

In this summary we briefly review each of these topics. No attempt is made here to survey the relevant literature, and the references are restricted primarily to recent work relevant to the topics discussed. (The reader may consult these papers for more bibliographic information.) A complete report with a more detailed presentation of the lectures and discussions can be obtained from the authors.

2 New Directions for Object-Oriented Data Models

Object-oriented database systems were introduced to provide more flexibility in designing database applications. The available systems usually offer (i) richer data models than relational systems and (ii) better software environments for developing database applications. However, in many applications, one cannot take full advantage of the richer data model because of shortcomings of the systems currently available and under development. Indeed, many fundamental issues are not well understood, including, for example:

(1) The relationship between programming language types and database schemas.

(2) The concept of view as used in relational systems.

(3) The common OODB practice of viewing a *class* to be a type with an associated *extent* (or "current population"). This is exacerbated by the common practice of blurring classes and types, in the sense that in some OODB models it is permitted to use class names to restrict coordinate/attribute ranges in type declarations.

(4) The relationship between object identifiers in OODBs and references in programming languages.

(5) Providing mechanisms to permit the "migration" of objects from one class to another.

(6) The problems associated with typing paradigms for heterogeneous collections.

To a large extent this is because the field of object-oriented DBMSs has developed very quickly, primarily by borrowing fragments of a number of technologies (relational, object-oriented, semantic data models). In many cases the concepts used in OODBs are not faithful copies of the originals, and the field has not had a chance to address in any real depth the issues arising from their combination.

Several speakers, including S. Abiteboul, A. Borgida, P. Buneman, R. Hull, D. Jacobs, D. Maier, and P. Richard, focused on these problems and suggested proposals for resolving them.

Buneman's presentation, set the stage for much of the ensuing discussion in this area by considering the extent to which conventional programming language techniques can capture DBPL functionalities. Buneman claimed that OODBs didn't provide anything new in terms of data models. However, object-oriented databases are *a priori* languages, and so, have forced us to address problems such as what programming language interfaces should semantic data models have – an issue so far largely ignored. With this in mind, Buneman examined some of the problems that arise when one tries to incorporate data models (or more properly instances thereof) as data structures in a programming language.

Richard emphasized the distinction between the notion of type systems as it is currently encountered in conventional programming languages and many DBPLs, and of schemas as in relational databases [ALR90, LR90a, LR90b]. S. Abiteboul proposed a powerful view mechanism [Abi90] as a means of providing more flexibility, and showed how such mechanisms correct poor features of currently used OODB models.

During the talks, four more focused problems arising in DBPLs were raised: choosing extents for types; differences between object identifiers and references (if any) (see [AK89, WHW90]); the blurring of types and classes; approaches for supporting object migration [BANLB*87, Sig89, ACO85, SS89]; and type checking for heterogeneous collections [OBB89, Bor90].

Interestingly, modelling issues (perhaps by their religious nature) raised very lively discussions. In particular, the four questions mentioned just above were highly controversial. This shows that although semantic modelling concepts and database languages have been maturing for quite some time, these notions are not yet settled and require further research.

3 Programming Language Types and Databases

A second theme of the workshop concerned *types and the problems associated with them in the context of databases.* One of the positive effects of this session was partly to dispel database community's phobia of "type terrorists", some of whom were rumored to have infiltrated the workshop. As noted earlier, there are considerable differences between types as arising in conventional programming languages and schemas as arising in databases. Nevertheless, there was a general consensus that:

1. database languages should be enhanced with polymorphic types and should employ convenient subtyping-inheritance hierarchies;
2. database languages should be statically typed, as much as possible, and the programmer should be aided by some form of type reconstruction;
3. several aspects of database languages require the development of new techniques for typing; e.g., the inclusion of a minimal set of run-time checks to ensure type correctness; and

4. most of the type information of the database is concentrated in the schema, so that schema modification gives rise to a particular set of interesting new typing problems.

In the first of five talks on these issues, Kanellakis focused on the results in [KMM90], and described recent developments on the algorithmic question of type reconstruction for languages with polymorphism, such as core-ML, the Milner-Mycroft calculus, and System F.

Moving away from pure lambda calculus languages to the Fun language of Cardelli and Wegner, Breazu-Tannen's talk explained how simultaneously to model parametric polymorphism, recursive types, and inheritance [BCGS89, BGS90]. The talk of Ohori discussed issues in designing a polymorphic type system that can serve as the "polymorphic core" of database programming languages [OBB89, OB88]. Such a core can be developed in two steps – first by defining a typed data algebra which is rich enough to represent various data models, and then by extending existing polymorphic type disciplines to the typed data algebra.

Moving away from functional types entirely, Waller examined a simple type system that is at the core of any Object-Oriented Database Management System. Using techniques reminiscent of data-flow analysis techniques for program schemes [AKW90, Wal90], he considered the problem of incremental method checking to handle schema updates. Finally, A. Borgida described a hybrid typing paradigm, in which the compiler inserts run-time tests into the code under certain conditions in order to ensure the ultimate integrity of the database.

It seems clear that the understanding of types has quite matured recently and that technical solutions to database needs seem already to exist. More efforts are still required to import the type technology to the database community. A positive aspect is that the community is more and more willing to accept such technology.

4 The Perspective from Logic

It is essential to extend the fruitful coexistence of logic and relational databases to the object-oriented database framework. A session devoted to logic aspects of object-oriented databases focused on several new developments. Beeri examined the connection of logic-oriented database languages to ideas from algebraic specification. Kifer and Warren, representing the "Stony Brook School of Thought", described several "object-oriented" logics with first-order semantics. Vianu examined issues of expressive power of languages for complex objects. Kuper discussed constraint logic programming.

Beeri started from the premise that, in contrast to relational systems, where formal foundations were given with the model, there is no commonly accepted formal description of OODB's. In considering the scope and requirements of such a foundation, he was led to reconsider the relationship of database models and languages to other areas. In particular, there seems to be a close connection between the basic ideas of relational databases and logic programming to those of algebraic specifications. Beeri claimed that relational db's, Horn-clause based deduction, ADT's, and also most structural aspects of OODB's can be so defined, and this provides a uniform framework for dealing with many db issue, in particular, declarative programming that unifies (to some extent) functional and relational programming, queries, and views. When considering the behavioral aspects of OODB's, a higher order flavor may be required.

This claim was also put forward very strongly by the Stony Brook School. Kifer gave an overview of the main ideas developed in the works on C-logic, O-logic, F-logic, and HiLog [Mai86, KJ89, KG89, CKW89b, CKW89a, CKW90]. The main theme of these works is that the higher-order concepts that are characteristic of object-oriented programming can and should be modeled by logical formalisms with a first-order semantics. Kifer argued that features such as object-identity, sets, classes, inheritance, typing, and encapsulation can be captured in this framework.

Various query languages for complex objects have been proposed. They are mostly extensions of relational algebra and calculus [AB88, Hul87], and of Datalog$^{\neg}$ [AG91, AK89, NS88], to complex objects. Typically, queries in these languages have the potential for unrealistically high complexity. Indeed, the design of tractable query languages for complex objects remains a challenging issue. Vianu [GV90b, GV90a] discussed various calculus-like languages for databases containing complex objects, and ways of evaluating their complexity.

A major recent development in logic programming is the interaction of logic and constraint paradigms (CLP, Prolog III, CHIP [JL87, D+88]). While declarative database query languages have been proposed, constraint programming has not really influenced query language design. The reason for this seems to be that constraint solving can be more easily combined with top-down evaluation strategies than with bottom-up ones. Kuper described a way to bridge the gap between bottom-up, efficient declarative database programming and efficient constraint solving (the presentation was based on joint work with Kanellakis and Revesz [KKR90]).

The need for formal techniques in object-oriented databases seems to be more and more felt by researchers in the field. The variety of approaches and their interest demonstrate that this is perhaps one of the most challenging issues facing the database research community. Like the data modelling session, the logic session was quite animated. Issues such as the problem of naming objects in an object-oriented logic seemed to inflame a substantial fraction of the audience (to the apparent surprise of another).

5 Practical Issues

Another theme of the workshop was concerned with the implementation of database programming languages or the use of type information in the context of databases. The first talk in the session was by Matthes. He focused on the design and implementation of data-intensive applications by presenting the experience of the DBPL/DAIDA project. Here, the conceptual design phases are divided into three different steps:

- the knowledge representation language TELOS for domain analysis;
- the semantic data model TDL (based on TAXIS) for conceptual design of system states and transitions; and
- the imperative database programming language DBPL for efficient management of typed data using sets and first-order predicates for data manipulation in a persistent multi-user environment [MS89, SM90].

Transformations from one step to another are made automatically as far as possible. The second talk dealt with the optimization of query languages in object-oriented database systems. The basic issue on query optimization is to find transformation rules such that the semantics is the same but the cost of evaluation against the database is less. The Revelation project, presented by D. Maier, explores such possibilities [GM90, GMDK90]. Finally, Connor showed how a protected viewing mechanism [C+90] could be programmed in a general-purpose language, Napier88 [M+89].

The reason for this session was that in the field, systems incorporating new ideas are being implemented as fast as new ideas are emerging. For instance, the work of Connor is fully implemented and deliverable. That much of the foundational work in other sessions was relevant to this session was evidence for importance of the field.

Acknowledgements: We wish to thank all the participants to the workshop. A number of them contributed to the complete report: C. Beeri, A. Borgida, V. Breazu-Tannen, D. Jacobs, M. Kifer, G. Kuper, D. Maier, A. Ohori, P. Richard, J. Schmidt, E. Waller, D. Warren. However, we are fully responsible for the errors or misrepresentations of point of views that may (intentionally or not) occur in the report.

We are also grateful to Karen Carter, Anthony Kosky, and LimSoon Wong for their help in the organization of the workshop.

The workshop received partial support from the National Science Foundation under grant # INT-8817874 and from the Institut National de Recherche en Informatique et Automatique, as part of a joint INRIA-NSF program.

References

[Abi90] S. Abiteboul. Virtuality in object-oriented databases. In *Journées Bases de Données*, Montpellier, France, 1990.

[AB88] S. Abiteboul and C. Beeri. On the power of languages for the manipulation of complex objects. In *INRIA Research Report* 846 (1988). To appear in TODS.

[ACO85] A. Albano, L. Cardelli, and R. Orsini. Galileo: A strongly-typed, interactive conceptual language. *ACM Transactions on Database Systems*, 10(2):230–260, June 1985.

[AG91] S. Abiteboul and S. Grumbach. A rule-based language with functions and sets. *ACM Transactions on Database Systems*, to appear, 1991.

[AK89] S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. In *ACM-SIGMOD Intl. Conf. on Management of Data*, 1989. To appear in *Journal of the ACM.*

[AKW90] S. Abiteboul, P. Kanellakis, and E. Waller. Method schemas. In *ACM Intl. Symp. on Principles of Database Systems*, 1990.

[ALR90] M. Adiba, C. Lecluse, and P. Richard. *Rationale and Design of Serendip, a Database Programming Language.* Technical Report, Altair, 1990. In preparation.

[BANLB*87] K.L. Chung, B.A. Nixon, D. Lauzon, A. Borgida, J. Mylopoulos, and M. Stanley. *Design of a Complier for a Semantic Data Model.* Technical Report CSRI-44, Department of Computer Science, University of Toronto, May 1987.

[BCGS89] V. Breazu-Tannen, T. Coquand, C. Gunter, and A. Scedrov. Inheritance and explicit coercion. In *Proc. 4th IEEE LICS*, 1989.

[BGS90] V. Breazu-Tannen, C. Gunter, and A. Scedrov. Computing with coercions. In *Proc. 5th Conf. on Lisp and Functional Programming*, 1990.

[Bor90] A. Borgida. On type checking and heterogeneity. 1990. in preparation.

[CKW89a] W. Chen, M. Kifer, and D.S. Warren. Hilog: a first-order semantics of higher-order logic programming constructs. In *Proceedings of North American Conf. on Logic Programming*, 1989.

[CKW89b] W. Chen, M. Kifer, and D.S. Warren. Hilog as a platform for database languages (or why predicate calculus is not enough). In Morgan Kaufmann, editor, *The 2nd Intl. Workshop on Database Programming Languages*, 1989.

[CKW90] W. Chen, M. Kifer, and D.S. Warren. Foundations of higher-order logic programming. 1990. submitted for publication.

[C+90] R. Connor, a protected viewing mechanism in Napier88, *in proc. EDBT-90*

[D+88] M. Dincbas et. al. The constraint logic programming language chip. In *Proc. Fifth Generation Computer Systems*, Tokyo, Japan, 1988.

[GM90] G. Graefe, D. Maier. Query Optimization in Object-Oriented Database Systems: a Prospectus. *In Advances in Object-Oriented Database Systems K. Dittrich, editor Lecture Notes in Computer Science 334, Springer-Verlag*, 1988.

[GMDK90] G. Graefe, D. Maier, S. Daniels, T. Keller. A Software Architecture for Efficient Query Processing in Object-Oriented Database Systems with Encapsulated Behavior. *Unpl. Manuscript*, April, 1990.

[GV90a] S. Grumbach and V. Vianu. Tractable query languages for complex objects. in preparation.

[GV90b] S. Grumbach and V. Vianu. Playing games with objects. In *Proc. Int'l. Conf. on Database Theory*, 1990. to appear.

[Hul87] R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256, Academic Press, 1987.

[JL87] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proc. ACM Symp. on Principles of Programming Languages*, pages 111–119, 1987.

[KG89] M. Kifer and G. Laussen. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD Int'l. Conf. on Management of Data*, pages 134–146, 1989.

[KJ89] M. Kifer and J. Wu. A logic for object-oriented logic programming (maier's o-logic revisited). In *Proc. ACM Symp. on Principles of Database Systems*, pages 379–393, 1989.

[KKR90] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. In *Proc. ACM Symp. on Principles of Database Systems*, pages 299–313, 1990.

[KMM90] P.C. Kanellakis, H. Mairson, and J.C. Mitchell. Unification and ML type reconstruction. 1990. to appear as book chapter in MIT Press volume on unification, dedicated to J.A. Robinson (full version manuscript available); see also *ACM POPL* 89 and *ACM POPL* 90.

[LR90a] C. Lecluse and P. Richard. *Data Abstraction, Bulk Data and relations in Database Programming Languages.* Technical Report 44, Altair, 1990. to appear.

[LR90b] C. Lecluse and P. Richard. *Schemas and Views in Database Programming Languages.* Technical Report, Altair, 1990. to appear.

[Mai86] D. Maier. A Logic for Objects, In *Proc. Workshop on Foundations of Deductive Databases and Logic Programming,* Washington D.C., pp 6-26, 1986.

[M+89] R. Morrison , A.L. Brown, R. Connor, A. Dearle. Napier88 Reference Manual, *Persistent Programming Research Report PPRR-77-89,* University of St Andrews

[MS89] F. Matthes and J.W. Schmidt. The Type System of DBPL. In *Proc. of the 2nd Workshop on Database Programming Languages,* Salishan Lodge, Oregon, pages 255–260, June 1989.

[NS88] S.A. Naqvi and S.Tsur. *A Logic Language for Data and Knowledge Bases.* Computer Science Press, Rockville, Md., 1988.

[OB88] A. Ohori and P. Buneman. Type inference in a database programming language. In *Proc. 4th Conf. on Lisp and Functional Programming,* 1988.

[OBB89] A. Ohori, P. Buneman, and V. Breazu-Tannen. Database programming in Machiavelli – a polymorphic language with static type inference. In *Proc. ACM-SIGMOD Intl. Conf. on Management of Data,* pages 46–57, Portland, Oregon, May – June 1989.

[Sig89] SIGMOD Record. Special Issue on Rule Management and Processing in Expert Database Systems 18(3). September 1989.

[SM90] J.W. Schmidt and F. Matthes. DBPL Language and System Manual. Esprit Project 892 MAP 2.3, Fachbereich Informatik, Universität Hamburg, West Germany, April 1990.

[SS89] Tim Sheard and David Stemple. Automatic verification of database transaction safety. *ACM Transactions on Database Systems,* 14(3):322–368, September 1989.

[Wal90] E. Waller. Updates in recursion-free method schemas. 1990. in preparation.

[WHW90] S. Widjojo, R. Hull, and D. Wile. A specificational approach to merging persistent object bases. In *Proc. of Fourth Intl. Workshop on Persistent Object Systems Design, Implementation and Use*, Morgan Kaufmann, Inc., Martha's Vineyard, Mass., September 1990. to appear.