# Semantics in Databases

Leopoldo Bertossi[1], Gyula O. H. Katona[2], Klaus-Dieter Schewe[3], and
Bernhard Thalheim[4]

[1] Carleton University, School of Computer Science, Herzberg Building,
1125 Colonel By Drive, Ottawa, Canada K1S 5B6
bertossi@scs.carleton.ca

[2] Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences,
PostBox 127, H-1364 Budapest, Hungary
ohkatona@renyi.hu

[3] Massey University, Department of Information Systems, Private Bag 11 222,
Palmerston North, New Zealand
k.d.schewe.@massey.ac.nz

[4] Computer Science Institute, Brandenburg University of Technology at Cottbus,
PostBox 101344, D-03013 Cottbus, Germany
thalheim@informatik.tu-cottbus.de

The term "Semantics" is one of the overloaded in computer science and used in various meaning. This variety can also be observed in database literature. In computer linguistics or web research, semantics is a component of the language which associates words or components of a grammar with their meaning (linguistic content). In modeling and specification, semantics assigns set-theoretic denotations to formulas in order to characterize truth. At the same time, semantics is used as the basis for certain methods of proof (truth and proof semantics in semiotics). In programming language technology, semantics is often used in the sense of operational semantics, i.e. consists in an interpretation of commands of a programming language by machine operations. This widespread usage of the term "semantics " has led to very different goals, methods, and applications. Semantics includes at the same time the interpretation of utterances, temporal, contextual, subjective and other aspects. Semantics is either considered operationally on the basis of applications or implementations, or logically associating a database state or a collection of database states to a truth value or pragmatically by relating utterances to the understanding of the user. These three understandings may be mapped to each other.

We require, however, that a formal mathematical theory is provided beside these substantial differences of the usage of "semantics". The same statement must have a stable interpretation and cannot be differently interpreted by different computer systems. The formal theory provides a way to prove correctness of specifications. A specification has the liveness property if a number of positive properties can be proven. A specification has the safety property if a number of negative properties cannot happen. Semantics is specified by means of formulas in a logic. Formulas reflect three different applications of semantics in databases. First, formulas are used for communicating the meaning between parties involved. Second, formulas support to verify whether the implementation carries the same meaning. Finally, formulas enable in the validation of the specification by proving liveness and safety properties. This expression of semantics of databases by formulas allows to consider semantics by means of mathematical logics, i.e. by theories in a certain logical language. Therefore, a database application may be considered as a collection of models of the given theory.

In the database research, semantics is used in a variety of meanings:

- Stable semantics is declared by static integrity constraints. A static integrity constraint is valid in each database state. Feasibility considerations have led to the introduction of a number of specific constraints called dependencies.
- Static integrity constraints are mapped to transition constraints. Transition constraints are Hoare triples (precondition, operation, postcondition) specifying that the postcondition must be valid after the operation has been applied to a database state that satisfies the precondition. The transaction approach is based on Hoare triples with equal pre- and postcondition.
- Dynamic semantics of applications is often left unspecified. Typical dynamic constraints are temporal formulas describing the life cycle of database objects.
- Database instances can be either elementary instances or deductively generated instances. Deductive database are used for generation of instances by an abstract generator. Different generators led to a variety of semantics such as stable semantics.
- Semantics can be state-dependent. In this case, phases can be distinguished. The database or parts of it satisfy a theory describing the phase.
- Semantics can be partially represented by structural properties. The research on semantic data models has led to a number of main structures such as types, subtypes, supertypes, and special types like attribute, entity and relationship types. Further, constructors are used for construction of complex types.
- Structural semantics concentrates on the meaning of singleton word fields. Research on database components and research on terminological ontologies follow these approaches.
- Conceptual modeling follows approaches developed by generative semantics. The meaning of a schema is constructed by the meaning of its constituents.
- Approaches used in the semantic web and ontology research follow those of practical semantics. The meaning of constructs is based on rules for practical application in interaction scenarios.
- The semantic aspect of utterances is expressed by the content. Therefore, content management targets too on semantics of data.

The state of practice does not reflect achievements of research in our area:

- A small set of the large variety of integrity constraint classes can be declaratively specified in database programming languages. These languages only have fragmental facilities for the expression of database semantics. Moreover, these facilities are redundant, lack orthogonality, are partially non-declarative and have side effects.
- DBMS have fragmental facilities for handling of database semantics. These facilities are incomplete and partially unsound.

- Specification languages used in database development tools vary in semantics of similar constructs. The same diagram may have another meaning in another tool.
- Integrity constraints can be specified in some database development tools. A translation of the integrity constraints to the programming language of the DBMS is, however, not provided.
- In most existing relational DBMS, integrity handling and checking is far from being efficient.
- The facilities for integrity constraint maintenance are slowly and reluctantly integrated into implementations of the SQL2 or SQL3 standard.
- Treatment of integrity constraints is not well founded in the SQL3 standard. Meaning and semantics is not well understood. There exists a large variety of interpretations for each group of integrity constraints.
- Enforcement policies used in SQL3 are rather confusing. Integrity constraints can be enforced in deferred or intermediate mode, may be enforced at row level or at statement level.

Semantics research has been neglected over the last years. In the early days of database research, issues related to database research played a prominent role, and papers discussing database models, conceptual design, integrity constraints and normalization often dominated major database conferences. This began to change at the end of the 80ies. Nowadays those issues do not appear to be part of the mainstream research. Dependencies and other constraints have been considered to be the "dying swan" in the database research. We observe a number of explanations for this development:

- The research has not found its way to major implementations of DBMS. The support for integrity constraint maintenance is still rudimentary in DBMS.
- The simple questions got solved. The remaining problems are far more difficult to tackle.
- The research has been conducted within a large variety of database models without a reference to research within other models. For instance, there is no systematic summarization of research on constraints in other models beyond those for the relational model and the entity-relationship model.
- The results on semantics are spread over a large number of conferences in the last decade. It is often difficult to find the proceedings or the references.
- The large variety of models has led to a large number of incompatible approaches and solutions. Their differences are not understood. A lot of research has been conducted for models which are not used or are not known to young researchers.

Semantics research has left a large number of open problems. We may distinguish a number of major research areas which are still open:

Satisfiability of specification should be provable for a collection of integrity constraints. If a schema is unsatisfiable the part of the specification causing this property must be identifiable.

**Integration of static and operational constraints** allows to consistently switch between different equivalent constraint sets. A coherent theory of integrity enforcement for different DBMS enables in switching applications between DBMS.

**Strong and soft interpretation** of constraints is not yet solved in a satisfiable form. Most constraints can be given in a strong logical form. Some constraints may use fuzzy or deontic approaches.

**Global normalization** is still not investigated in detail. Normalization is currently using only a local variant in a type-wise form.

**Continuous engineering** requires to cope with consistent extensions of structures, functionality and integrity support of a running database system.

**Integration of quality requirements** into specification will lead to provide a motivation for requirements such as normal forms and acyclicity.

**Complexity of constraint sets** is currently only known for simple classes of constraints such as sets of keys and functional dependencies. Average complexity results would lead to a better understanding of semantics in 'normal' applications.

**Enforcement of integrity constraint sets** is currently mainly supported by DBMS on the level of rule triggering systems. The alternative approaches to integrity enforcement should be integrated into these approaches.

**Implication problems** are tackled on the level of classes of constraints. A database application uses however a set of constraints from different classes. The interaction of constraints needs more research.

**Treatment of semantics by views** has not yet been satisfactorily solved. At the same time, most large database systems are heavily based on views.

**Distributed integrity management** allows to avoid the centralized integrity management in distributed applications. Distributed systems are still developed on the allocation of data sets to nodes without consideration of semantics.

**Integration of vertical, horizontal and deductive normalization** allow a more flexible optimization of database behavior. Therefore, a common framework for vertical, horizontal and deductive normalization is sought.

**Treatment of incomplete specifications** is necessary for practical reasons. Usually, specifications are incomplete. An incomplete specification of constraints leads to completely different normalized schemata.

Implementation of integrity constraints in database systems is still based on the waterfall model:

1. The initial aim in conceptual database design is the development of structural or semantical integrity constraints.
2. The operational optimization of integrity constraints is based on normalization approaches which restructure the database by decomposition or fragmentation and lead to equivalent sets of integrity constraints which integrity enforcement is simpler in the given DBMS.
3. During the mapping phase check constraints are derived which allow to control consistency of databases on the row level, by single sub-queries or aggregation or inter-table sub-queries.

4. Constraints which cannot be mapped to check constraints are mapped to assertions on the basis of independent and inter-table constraints if the DBMS intended to be used for the database implementation supports assertions.
5. Static integrity constraints can be mapped to triggers if the DBMS supports triggers. Trigger support may vary and uses additional semantics in the DBMS.
6. Stored procedures enable in encapsulation of operations and integrity constraints. Integrity maintenance is wrapped into the stored procedure code.
7. Integrity constraint checking at the application level is mainly performed by the transaction management or at the level of application programs.

At the same time we observe a number of pitfalls in the existing research literature:

- *Normalization* is still considered as the ultimate weapon for structural optimization. The approaches well developed so far cover vertical normalization at the local type level. Horizontal or deductive normalization is still not integrated into vertical normalization approaches. Global normalization is moreover in an embryonic state.
- *Tuning and operational optimization* lead to physical schemata which are structurally and semantically different from the conceptual schema. The later is used for programming and querying.
- The *class-centered approach to integrity specification* concentrates on the class-wise treatment of integrity constraints. Instead of that, we should concentrate on the treatment of a set of integrity constraints valid in a application.

The papers in this volume reflect a variety of approaches to semantics in databases:

P. Barceló, L. Bertossi and L. Bravo develop a theory of *consistent answers* from database that violate constraints, i.e. answers to queries that are generated in every minimal repair of the database.

J. Biskup and B. Sprick design a *unifying framework* for consistent and integrated reasoning on *semantic constraints and security constraints* in highly distributed systems. The formalization is based on a propositional multimodel logic for multi-agent systems with temporal and epistemic operators.

H. Decker discusses the potential of paraconsistent reasoning for datalog, computational logics and for foundations of *partially inconsistent databases.*

S. Hartmann introduces *soft cardinality constraints*. If cardinality constraints are conflicting then a resolution strategy may use a weakening approach and may consider only those which are not in conflict and which conflict is not causing other inconsistencies. A number of conflict resolution strategies is introduced.

S. Hegner develops a characterization of *type hierarchies under open specifications*. The algebraic characterization is based on weak partial lattices, the logical one on products of models of propositional-based specifications.

H.-J. Klein focuses on sure answers in the case of *null values* which are representing existing but unknown values or are indicating that there is no information. He uses the introduced theory to generate transformations of SQL queries that approximate sure information answers.

S. Link generalizes the integrity enforcement on the basis of greatest consistent specializations to *maximal consistent effect preservers*. He introduces a general definition and framework that formalizes effect preservation of operations while maintaining database consistency.

F. Neven and T. Schwentick survey results on *pattern languages for tree-structured data* such as XML data. It is shown that an intermediate logic between first-order logic and monadic sencond-order logic captures the expressive power of languages with regular path expressions.

D. Seipel and U. Geske investigate the semantics of *cardinality constraints* on the basis of *disjunctive deductive database programs* and consider various deductive calculi for cardinality constraints. There is no sound and complete calculus using only definite deductive rules. Instead, a mix of disjunctive rules and rules from the definite calculus can be used.

J.M. Turull-Torres considers the *expressive power of relational query languages* on the basis of an abstract notion of *similarity* or indistinguishability of databases. He develops a general framework for definition of semantic classifications of queries and shows that the hierarchy is orthogonal with the time-space hierarchy for Turing machines.