ASM Foundations of Database Management

Bernhard Thalheim¹ and Klaus-Dieter Schewe²

¹ Christian Albrechts University Kiel, Department of Computer Science
   Olshausenstr. 40, D-24098 Kiel, Germany
² Massey University, Information Science Research Centre
   Private Bag 11 222, Palmerston North, New Zealand
   thalheim@is.informatik.uni-kiel.de, k.d.schewe@massey.ac.nz

Abstract. Database structuring is well understood since decades. The operating of databases has been based in the past on temporal logics and did not yet get an easy to understand formal underpinning. Therefore, conceptions like transaction and recovery are mainly discussed at the logical or operational level. This paper shows that database structuring and functionality can be defined within a uniform language. We base database semantical on the operational semantics of abstract state machines (ASM). This uniform mechanism allows to define the structuring, the functionality, the distribution and the interactivity of a database system in a way that supports abstract consideration at various layers of abstraction, that supports refinement of specifications to more detailed ones and that support proof of properties.

1 Adequacy and Deficiencies of Database Technology

1.1 Strength and Weaknesses of Database Technology

Database systems are currently broadly used for support of data-intensive services. These broad usage is based on advantages such as the following:

Consistent storage of data: Data are uniquely stored in the most actual version. Each user gets the same data. Inconsistency can be avoided. Furthermore, redundancy can be reduced and standards can be enforced.

Multi-user support: Data can be consistently shared among different users. Also, conflicting requirements can be balanced. Security is enforced by restricting and managing access to data. Data can be consistently distributed within a network.

Integration into component-ware: Currently, database systems are turned into middle-ware components in information-intensive applications. Database operating is based on the transaction paradigm. A transaction is a logical unit of work. Database systems are designed to support transactions.

Nevertheless, database engines do not completely support complex applications such as internet services, real-time applications, stream information systems and web information systems.
Content and information instead of data: With the development of more complex application sophisticated support for various abstraction levels on data is required. Instead of raw or micro-data users want to retrieve condensed data or macro-data and to work on them.

Transaction semantics: The transaction approach is based on atomicity, consistency, isolation and durability requirements on transaction application. User action are sometimes not atomic and require sophisticated support for intermediate actions. The isolation level may vary over time. The are different degrees for durability.

View support: Each object in the database needs to be identifiable. Query languages allow to construct macro-data from the micro-data of the objects contained in the database. These macro-data may be composed into complex derived objects. Due to the query language some of them may be not identifiable. Further, their connection to the micro-data may be not injective. Therefore, data manipulation on macro-data cannot translated to data manipulation on micro-data.

Missing operational semantics: Semantical treatment of structuring is based on the predicate logic since the advent of the relational database model. Entity-relationship modelling can be based on a generalized predicate logic. Object-oriented models are often defined in a fuzzy manner without existing models. Moreover, database operation is still not well based.

In this section we show now that the weaknesses of database models and technology for internet application can be reduced by the ASM approach.

1.2 Well-Founded Structuring and Operating Transparency

Database systems are typically specified through a specification of database structuring and the assumption of canonical database operating. Structuring of databases is given by a signature of the database and a set of static integrity constraints. They form the schema of the database. The signature is used as an alphabet of a canonical first-order (or first-order hierarchical [13]) predicate logic. Static integrity constraints may be expressed as formulas of this logic. The functionality of a database system is defined on the basis of an algebra that is generically defined over the signature. This algebra is extended to a query algebra and to modification operations for the database system. Database management systems provide services that are assumed to be given whenever we are talking on database operating. Their behaviour and their operating is assumed to be canonically given.

This assumption might be appropriate for the predesign or business user layer [5,13] of specification. The implementation independence of the business user layer and the operating transparency supports concentration application specific aspects of a database system. It is not appropriate for the conceptual layer since we need to consider database behaviour as well. This inappropriateness causes many problems for database programming and operating. This paper shows how the ASM approach can be used to overcome this gap.