Extended Transaction Models and the ACTA Framework

Panos K. Chrysanthis[1] and Krithi Ramamritham[2]
[1]Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA
[2]Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai, India

Synonyms

Advanced transaction models; Generalization of ACID properties

Definition

Although powerful, the transaction model adopted in traditional database systems is found lacking in functionality and performance when used for applications that involve reactive (endless), open-ended (long-lived) and collaborative (interactive) activities. Hence, various extensions to the traditional model have been proposed, referred to as *extended transactions*. These models are characterized by the structure of their transactions, the commit and abort dependencies and the visibility rules among transactions. *ACTA* is a comprehensive transaction framework that facilitate the specification, analysis and synthesis of extended transaction models. The name *ACTA*, meaning *actions* in Latin, was chosen given the framework's appropriateness for expressing the properties of actions used to compose a transactional computation.

Key Points

By means of the notion of transactions, database systems offer reliability guarantees concerning the correctness of data in spite of failures and concurrent accesses by multiple users. However, the transaction model as well as the simple data model adopted in traditional database systems have been found lacking in *functionality* and *performance* in their support of the emerging advanced database applications such as design databases, computer publishing, network management, multidatabases and mobile databases. In order to deal with the inherent limitations of the traditional data and atomic transaction model, researchers have proposed semantic and object-oriented data models and extensions to the traditional transaction model. Nested transactions was the first such extension that added a hierarchical structure to the traditional flat atomic transactions. The hierarchical structure allows concurrency within a transaction and fine-grained failure and exception handling since sub transactions can abort independently without causing the abortion of the whole transaction.

The original nested transaction model was subsequently enhanced with new types of sub transactions, relaxed abort and commit dependencies and visibility rules for externalizing partial results among transactions. These extensions led to a variety of open-nested transactions models such as Sagas, Split Transactions, Flex Transactions, ConTracts and S-transactions, and of correctness criteria such as quasi serializability, epsilon-serializability, semantic atomicity, quasi failure-atomicity.

All the above extensions have been introduced with specific applications or with specific transaction properties in mind [2]. Their ad hoc character makes it difficult to identify the properties of transactions that adhere to a particular model and to ascertain in what respects an extended transaction model is similar or different from another. The need for a comprehensive transaction framework that would facilitate the precise specification of the properties of a model, vis a vis visibility, consistency, recovery and permanence, and allow the formal comparison of different models led to the development of *ACTA* [1]. ACTA is a first-order logic based formalism with a precedence relation that allows a transaction modeler to specify both the high level properties (requirements) of a model and the lower level behavioral aspects of the model in terms of axioms. Specifications include

the following four components: (i) the set of transaction management events associated with the transaction model, such as *begin, commit, abort, split*, and *join*; (ii) the semantics of these significant events, characterized in terms of their effect on objects (their value and synchronization state) and other transactions (different types of dependencies, such as commit dependency and abort dependency); (iii) the view of each transaction, specifying the state of objects visible to that transaction; and (iv) the conflict set of each transaction, containing those operations with respect to which conflicts need to be considered.

Besides supporting the specification and analysis of existing transaction models, ACTA has the power to specify the requirements of new transactional applications and synthesize models that satisfy these requirements. This was demonstrated by deriving new transaction definitions either by starting from first principles or by modifying and/or combining the specifications of existing transaction models. The exercise of analyzing and synthesizing different transaction models revealed the many advantages of using a simple formalism like ACTA to deal with advanced transactions and has influenced a lot of transaction processing work in industry and academia.

Although ACTA has been developed to characterize extended transaction models, it has been extended to express the various correctness criteria beyond serializability. The use of this formalism resulted in a consolidated notion of correctness in which the different serializability-based criteria, such as predicatewise serializability and cooperative serializability, can been seen as special cases [3].

Cross-References

- ▸ ACID Properties
- ▸ Compensating Transactions
- ▸ ConTract
- ▸ Correctness Criteria beyond Serializability
- ▸ e-Commerce Transactions
- ▸ Flex Transactions
- ▸ Generalization of ACID Properties
- ▸ Multilevel Transactions and Object-Model Transactions
- ▸ Nested Transaction Models
- ▸ Open Nested Transaction Models
- ▸ Polytransactions
- ▸ Sagas
- ▸ Semantic Atomicity
- ▸ Split Transactions
- ▸ Transaction
- ▸ Transaction Management
- ▸ Transactional Processes
- ▸ Web Transactions
- ▸ Workflow Transactions

Recommended Reading

1. Chrysanthis PK, Ramamritham K. Synthesis of extended transaction models using ACTA. ACM Trans Database Syst. 1994;19(3):450–91.
2. Elamagarmid AK. Database transaction models for advanced applications. Los Altos: Morgan Kaufmann; 1992.
3. Ramamritham K, Chrysanthis PK. A taxonomy of correctness criteria in database applications. VLDB J. 1996;4(1):181–293.

Extendible Hashing

Donghui Zhang[1], Yannis Manolopoulos[2], Yannis Theodoridis[3], and Vassilis J. Tsotras[4]
[1]Paradigm4, Inc., Waltham, MA, USA
[2]Aristotle University of Thessaloniki, Thessaloniki, Greece
[3]University of Piraeus, Piraeus, Greece
[4]University of California-Riverside, Riverside, CA, USA

Definition

Extendible hashing is a dynamically updateable disk-based index structure which implements a hashing scheme utilizing a directory. The index is used to support exact match queries, i.e., find the record with a given key. Compared with the B+-tree index which also supports exact match queries (in logarithmic number of I/Os), extendible hashing has better expected query