Letter from the Special Issue Editor

Offering transactional properties has been one of the cornerstones of the success of database management systems. As database technology has evolved, covering new application domains, handling new data models, and supporting new hardware and distributed infrastructure, transaction management had to be redefined, redesigned and rethought.

This starts by understanding what are actually the requirements of the applications in terms of "keeping data consistent" and providing users with a "consistent view of the data". Do they require the traditional transactional properties such as atomicity and strong isolation levels or can they accept weaker levels of consistency? Furthermore, the distributed nature of current data stores comes with great promises but also challenges. Replication is a common mechanism to provide better performance and availability but comes with its own cost of keeping replicas consistent. While scaling up by adding new server nodes every time demand increases sounds promising, transaction management tasks have shown to be very difficult to scale either relying on complex coordinating protocols such as two phase commit or on a central component that is difficult to scale. Distribution and replication across geo-distributed data centres adds the additional cost of wide-area communication.

Many different research communities are actively looking for solutions each bringing their own expertise and background. This becomes obvious when looking at the venues in which work on consistency for distributed data is published. Contributions can be found in conferences and journals related to database systems, distributed systems, operating systems, storage systems, and computer systems in general. This exchange of ideas broadens the scope and has led to a whole new range of possibilities.

This diversity becomes apparent in the first three articles of this issue that explore and analyze the wide range of consistency models and definitions that have been proposed in the recent past, how they relate to each other, what are there trade-offs, and how they can be implemented. In the first article, Agrawal *et al.* explore the design space in regard to distribution and replication, how replica management is intertwined with transaction management, and what are the options to apply changes to replicas. In the second article, Faleiro and Abadi discuss the trade-off between fairness, isolation and throughput in transactional systems, arguing that only two out of the three can be achieved at the same time. Along a set of recently developed data stores, they show how each of them only supports two of these properties. In the third article, Bravo *et al.* discuss the importance of logical and physical clocks to track the order of events in a distributed system, their costs in term of space and coordination overhead, and how they they are used in a wide range of systems to implement both weak as well as strong consistency.

The second set of papers presents concrete solutions to support large-scale transactional applications in distributed environments. The first two papers present novel concurrency and recovery components. Bernstein and Das present an optimistic concurrency approach based on parallel certifiers that supports transactions that access more than one partition while keeping global synchronization to a minimum. They rely on causal message ordering to guarantee the proper serialization order across all partitions. Levandoski *et al.* present the Deuteronomy architecture which separates transaction functionality from storage functionality. Their transaction manager, based on mulit-version concurrency control, exploits modern hardware and specialized data structures, and uses caching and batching to minimize data transfer. The last two papers present cloud platforms specifically designed to provide scalability and elasticity for transactional applications. Salomie and Alonso present the Vela platform that exploits both virtualization and replication. Virtualization enables dynamic and elastic deployment of database instances within and across heterogeneous servers while replication offers scalability. Finally, Jimenez-Peris *et al.* present CumuloNimbo, an elastic fault-tolerant platform for multi-tier applications providing a standard SQL interface and full transactional support across the entire data set.

I hope you enjoy reading these articles as much as I did!

Bettina Kemme
McGill University
Montreal, Canada