A Concurrency Model for Transaction Management

Marc Descollonges
INSOP / Borland
1 Blue Hill Court
Scotts Valley, CA 95066
mdescollonges@mcimail.com
408-438-2022

Abstract

Various mechanisms, such as locking, time-stamped based algorithms, optimistic methods, versioning, have been used to support data consistency in transactions. Depending on the application's environment, InterBase offers the choices of two models for transaction handling, the consistency model, and the concurrency model. This talk presents a comprehensive overview of how Interbase's unique concurrency model, implemented with a multi-generational record structure, can improve global transaction throughput, by allowing a transaction to read a stable view of the database, without having to wait for simultaneous updates to release access to the data. A discussion of how this concurrency model handles the six classic problems of transaction management is presented: Lost updates, Dirty reads, Cursor instability, Non-reproducible reads, Phantom records, Update side effects.

Using Multiversioning to Improve Performance Without Loss of Consistency

Roger Bamford
Oracle Corporation

Abstract

Oracle first implemented multiversioning, which was called "read consistency", in Version 4 (ca. 1983) of the kernel. It supported the old master/new master programming paradigm for both application builders and query-processor implementors while avoiding the reader/writer serialization that results from using automatic share locks. The resulting transaction consistency model was arcane, not exactly serializable, but pragmatic in actual use. In this session, Mr. Bamford will cover read consistency from a number of viewpoints. First, the model will be described and a little detail about Oracle's current implementation will be given. Next, the consequences of having read consistency as part of the database's core architecture will be discussed. In particular, the effects on the designs for row locking, query processing, distributed query, distributed cache coherence, referential integrity, and set updates will be described. This will be followed by a comparison of Oracle's read consistency model to other common concurrency models, looking at cost and throughput under a variety of workloads. Finally, Mr. Bamford will propose how Oracle intends to carry read consistency into the future as support is added for complex objects, distributed computations, and long transactions.