Research on the Principle and Application of Multiversion Data in SQL Server 2008

Aiwu Li¹ and Hongying Liu²

¹ Dept. of Computer Science
Guangdong Vocational College of Posts and Telecom
Guangzhou, China

² Dept. of Computer Science and Engineering, Guangzhou Vocational & Technical
Institute of Industry & Commerce
Guangzhou, China
law_mail@126.com, 327147616@qq.com

Abstract. In this paper, to solve the problem of read/write waiting in SQL Server, designed two additional database option parameters: read-committed-snapshot, allow-snapshot-isolation and introduced multiversion data technology, while performing modification operations, the old version data will be stored in tempdb database. Read operation can use the old version data in tempdb database, eliminate the need to use shared lock. Set a new snapshot isolation level, read operations no longer need to use shared lock. The experiment results show it can solve the read/write waiting problem in serializable isolation level.

Keywords: Multiversion data, read and write wait, transaction isolation level.

1 Introduction

In SQL Server 2000 and earlier releases, the read operation will use shared locks, the update will use exclusive locks, the two locks are not compatible with each other. Read waiting to write and vice versa will occur [1-2]. At read committed isolation level, the read operation will wait for modifications until the transaction commits. At serializable isolation level, modify operation will wait for the read operation until the transaction is complete. Since the occurrence of read/write waiting, SQL Server’s operating efficiency is greatly reduced [3-4].

In order to solve the problem of read/write waiting, from 2005 release, SQL Server added two additional database option parameters: read-committed-snapshot, allow-snapshot-isolation and introduced multiversion data technology. While performing modification operations, the old version data will be stored in tempdb database [4-5].

While open read-committed-snapshot parameter in read committed isolation level, if the transaction in which modification was conducted does not finish, the read operation will use the old version data in tempdb database, eliminating the need to use shared lock [6-7].
While open allow-snapshot-isolation parameter, the update will also produce multiversion data, and a new snapshot isolation level can be set, which achieved the same effect, with serializable isolation level, but read operations no longer need to use shared lock. If the data read are being modified by other connection, the read operation will read the old version data before the transaction began. By this way, can solve the read/write waiting problem in serializable isolation level [8-9].

2 Transaction Processing and Transaction Isolation Level in SQL Server 2008

2.1 Transaction Processing in Database System

Transaction processing is designed to maintain a computer system in a known, consistent state, by ensuring that any operations carried out on the system that are interdependent are either all completed successfully or all canceled successfully.

Databases which treat the integrity of data as paramount often include the ability to handle transactions to maintain the integrity of data.

A single transaction consists of one or more independent units of work, each reading and/or writing information to a database or other data store. When this happens it is often important to ensure that all such processing leaves the database or data store in a consistent state.

A database transaction, by definition, must be atomic, consistent, isolated and durable.

Transactions in a database environment have two main purposes:

1. To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.

2. To provide isolation between programs accessing a database concurrently. If this isolation is not provided the programs outcome are possibly erroneous.

Transaction processing allows multiple individual operations to be linked together automatically as a single, indivisible transaction. The transaction-processing system ensures that either all operations in a transaction are completed without error, or none of them are. If some of the operations are completed but errors occur when the others are attempted, the transaction-processing system "rolls back" all of the operations of the transaction (including the successful ones), thereby erasing all traces of the transaction and restoring the system to the consistent, known state that it was in before processing of the transaction began. If all operations of a transaction are completed successfully, the transaction is committed by the system, and all changes to the database are made permanent; the transaction cannot be rolled back once this is done.

Transaction processing guards against hardware and software errors that might leave a transaction partially completed, with the system left in an unknown, inconsistent state. If the computer system crashes in the middle of a transaction, the