**Optimistische Sperrverfahren**

http://de-gmtdmp.mookie1.com/t/v2/learn?tagid=V2_4153&src.rand=1487405130&src.visitorid=7kfDSLf%2FCHiHmEs6PF1dIqtE%2FQ8W5PoHXvUxKS4SrJc%3D&src.id=datenbanken-verstehen.de

Die validierenden bzw. **optimistischen Sperrverfahren**, auch bekannt als Optimistische-Lockverfahren, verwenden im Gegensatz zu den **pessimistischen Sperrverfahre**n keine verzögerte Transaktionsausführung.

Die **Gültigkeit einer Transaktion** wird erst am **Ende geprüft**, wenn es zu einer Festschreibung (Commit) auf der Datenbank kommt. Ist die Transaktion aufgrund einer nicht-serialisierbaren Ausführung ungültig, wird sie zurückgenommen (Rollback).

**Optimistische Sperrverfahren** kommen dann zum Einsatz, wenn die Wahrscheinlichkeit für Transaktionskonflikte bzw. nicht-serialisierbare Transaktionen gering ist. Dadurch wird der Aufwand für das Ermitteln von Objekten und Setzen von Sperren eingespart. Zurückgesetzte Transaktionen werden einfach erneut ausgeführt, bis die Transaktion erfolgreich ist.

**Beispiel für ein optimistisches Sperrverfahren**

Jede Transaktion durchläuft die drei Phasen: **Arbeitsphase, Validationsphase und Schreibphase**. Nur wenn die Transaktion die Validation besteht, werden Änderungen auf der Datenbank festgeschrieben (Commit). Durch diese Vorgehensweise ist das Zurücknehmen (Rollback) von Änderungen nicht notwendig.

[](http://www.datenbanken-verstehen.de/wordpress/dbv/uploads/optimistischen_sperrverfahren_beispiel.jpg)  
Die Besonderheit besteht darin, dass Änderungen zunächst nicht auf der Datenbank vorgenommen werden, da jede Transaktion ihre Operationen in einem separaten Speicherbereich (Workspace) durchführt. Dadurch werden Änderungen während der Arbeitsphase für andere Transaktionen nicht sichtbar. Erst mit dem Festschreiben in der Datenbank, können andere Transaktionen mit der Änderung arbeiten.

**Vorteile optimistischer Sperrverfahren**

**Synchronisationsverfahren** mit optimistischem Ansatz eignen sich nicht für Datenbanken, die häufig Änderungen auf einem Datenbestand durchführen. Dabei handelt es sich meist um **Transaktionssysteme**, die bspw. Bestellungen und Reservierungen abwickeln. Angenommen wir haben ein Hotelreservierungsportal bei dem eine Vielzahl an Benutzern gleichzeitig ein Hotel reservieren wollen.

Sobald ein Reservierungsprozess ausgelöst wird und der Benutzer beginnt, seine Daten anzugeben, muss mindestens eine Sperre auf Zimmerebene vorliegen. Sonst könnte der Fall eintreten, dass bei Ende der Reservierung kein Zimmer mehr verfügbar ist, weil andere Benutzer ihre Daten schneller erfasst haben.

Bei Datenbanken mit wenigen parallelen Änderungen an Datensätzen trägt das Verfahren zur **schnelleren Verarbeitung der Daten** bei. Eine Einsatzmöglichkeit wäre die Beladung von **Eingangstabellen** in einem **Data Warehouse**. In den dort verwendeten Schichten **Stage** und **Cleanse**, werden Daten i. d. R. eins zu eins übernommen, da hier noch **keine Normalisierung** vorgenommen wird.