Konsistenzsicherung durch Verwaltung von Inkonsistenzen

Author(s):
Leikauf, Peter

Publication Date:
1990

Permanent Link:

https://doi.org/10.3929/ethz-a-000578566 →

DISS. ETH Nr. 9208

Konsistenzsicherung durch Verwaltung von Inkonsistenzen

ABHANDLUNG

Zur Erlangung des Titels

DOKTOR DER TECHNISCHEN WISSENSCHAFTEN

der

EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE ZÜRICH

vorgelegt von

Peter Leikauf
Dipl. Informatik-Ing. ETH

geboren am 18. Dez. 1958
von Fislisbach AG

Angenommen auf Antrag von

Prof. Dr. C.A. Zehnder, Referent
Prof. Dr. R. Marti, Korreferent

1990

Zusammenfassung

Die Sicherstellung der Konsistenz in einem Datenbanksystem ist eine Forderung, deren Erfüllung für den Datenbestand von existentieller Wichtigkeit ist. Die dafür notwendigen Massnahmen seitens des Datenbankverwaltungssystems geraten jedoch in Konflikt mit zwei ebenfalls zentralen Forderungen: Sie können einerseits die Effizienz zu stark beeinträchtigen und andererseits dem Benutzer zu viel Flexibilität nehmen.

In der heutigen Praxis wird dieses Problem üblicherweise auf zwei verschiedene Arten zu lösen versucht: Entweder durch Verzicht auf Flexibilität oder durch Verzicht auf Konsistenz. Im ersten Fall wird die Konsistenzsicherung in die Anwenderprogramme verlagert (und damit dem Benutzer der direkte Schreib-Zugriff auf die Daten verwehrt), im zweiten Fall wird darauf überhaupt verzichtet. Zwar gibt es für beide Verfahren Klassen von geeigneten Anwendungen; aber oftmals wären statt dessen die volle Konsistenz *und* viel Flexibilität wünschenswert, vielleicht unter bewusstem Verzicht auf Effizienz.

In dieser Arbeit wird ein Verfahren vorgestellt, mit welchem für ein relationales Datenbankverwaltungssystem sowohl maximale Konsistenz als auch maximale Flexibilität erreicht werden können. Dies äussert sich darin, dass der Benutzer beliebig viele und komplexe Konsistenzbedingungen definieren kann und dennoch in Art und Reihenfolge seiner elementaren Datenoperationen völlig frei ist. Die zugrunde liegende Idee besteht darin, inkonsistente Operationen vorerst beliebig zuzulassen, aber zu registrieren und nachträglich sobald als möglich zu eliminieren.

Es wird gezeigt, dass der Preis für solch ein kompromisslos liberales System - nebst einer zu erwartenden Effizienzeinbusse - letztlich darin besteht, dass sich während einer gewissen Zeit unsichere Datenelemente in der Datenbasis befinden können. Bei relationalen Datenbanken handelt es sich dabei um provisorisch gebildete oder provisorisch gelöschte Tupel.

Die Implementation eines entsprechenden Datenbankverwaltungssystems "SoftRDS" gelang mit bemerkenswert einfachen Mitteln, aufbauend auf der bestehenden Ein-Tupel-Schnittstelle LIDAS/RDS. Grundlage für dieses System ist das Prinzip der Speicherung von Datenbeschreibung, Daten und Inkonsistenzen in einer gemeinsamen Datenbank.

Abstract

Ensuring the semantic integrity of stored data is of fundamental importance in a database. Providing the required functionality as an integrated service of a database management system has two significant drawbacks, however. First, the checking of integrity constraints may affect the efficiency of database applications. Moreover, the mechanism is not flexible enough in many cases.

In practice, the above problems are usually solved by either giving up flexibility or by foregoing database integrity. In the first case, the responsability for the checking of semantic integrity constraints is simply moved from the database management system to application programs. As a result of this, direct write-access to the database is ruled out. In the second case, constraint checking is simply dropped entirely. Of course, there are applications for which one of the above pragmatic solutions can be tolerated. More often, however, comprehensive integrity checking with a certain amount of flexibility is much more desirable, even if efficiency suffers as a result.

In this thesis, we propose a new approach for relational database systems which guarantees the highest degree of semantic integrity while allowing as much flexibility as possible. Although the user can define semantic integrity constraints of arbitrary complexity, there are absolutely no restrictions on the type or sequence of elementary database operations - even if these operations were to lead to inconsistent database states. The basic idea is to simply register each "illegal" operation so that its effects can be undone at a later stage.

Ultimately, the highest price to be paid for this kind of flexible constraint enforcement is the temporary existence of uncertain data in the database. For example, a relational system will have to deal with tuples which are flagged as either provisionally inserted or provisionally deleted.

The proposed ideas have been implemented in the system SoftRDS which is based on LIDAS/RDS, a simple relational database manager with a one-tuple-at-a-time interface. The extensions turn out to be surprisingly simple and elegant. They rely on the principle that data, data descriptions, and inconsistencies are all stored in a single integrated database.