Transaktionen in Datenbankprogrammiersprachen semantische Integration und prädikative Implementierungsstrategien

Author(s):
Reimer, Manuel

Publication Date:
1984

Permanent Link:
https://doi.org/10.3929/ethz-a-000336982 →

Diss. ETH Nr. 7553

Transaktionen in Datenbankprogrammiersprachen

Semantische Integration und prädikative Implementierungsstrategien

ABHANDLUNG
zur Erlangung des Titels eines
Doktors der Technischen Wissenschaften
der
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE
ZÜRICH

vorgelegt von
Dipl.-Inform. MANUEL REIMER
geboren am 20. April 1956
in Hamburg

Angenommen auf Antrag von
Prof. Dr. C. A. Zehnder, Referent
Prof. Dr. J. W. Schmidt, Korreferent

Zürich 1984

Zusammenfassung

Datenbankmodelle enthalten neben den Konzepten zur Datenstrukturierung und zur Datenmanipulation auch Konzepte zur Erhaltung der Datenintegrität. Zusammengesetzte Operationen auf den Datenstrukturen einer Datenbank, welche die Invarianz der für die Datenbank definierten Integritätsbedingungen gewährleisten, können als Transaktionen formuliert werden. Die vorliegende Arbeit behandelt alle Aspekte einer Integration des Datenbankkonzepts "Transaktion" in eine Datenbankprogrammiersprache, die das relationale Datenbankmodell und eine Pascal-artige Programmiersprache kombiniert.

Ausgangspunkt der programmiersprachlichen Integration ist das Prozedurkonzept, das sukzessive an die Bedürfnisse der Transaktionsprogrammierung angepasst wird. Da Transaktionen im allgemeinen nur auf bestimmten Ausschnitten einer Datenbank operieren, werden die Zugriffsrechte in der Transaktionsdefinition auf Teilrelationen, sogenannte selektierte Relationen, eingeschränkt und spezielle Mechanismen zur Zugriffskontrolle und zur Parameterübergabe eingeführt. Selektierte Relationen basieren dabei auf Prädikaten zur Datenselektion, wie sie im zugrundeliegenden relationalen Datenbankmodell formuliert werden können.

Bei der Semantikdefinition der Transaktionsdeklaration, des Transaktionsaufrufs und der Transaktionsausführung findet die Problematik der konkurrierenden Ausführung von Transaktionen besondere Beachtung. Die Definition einer korrekten, parallelen Ausführung basiert auf der axiomatischen Spezifikation der konkurrierenden Transaktionen mit Prä- und Postkonditionen, aus denen sich eine Postkondition der parallelen Ausführung ableiten lässt. Sind alle Operationen einer Transaktion und deren semantische Bedeutung a priori bekannt, kann ausgehend von der abgeleiteten Postkondition ein korrekter Scheduler zur parallelen Ausführung einer vorgegebenen Menge von Transaktionen erzeugt werden. Wird dagegen von den einzelnen Operationen einer Transaktion abstrahiert, müssen Strategien zur Parallelitätskontrolle verwendet werden, die ausschliesslich auf der Kenntnis der Datenobjekte beruhen, auf die von einer Transaktion prädikativ zugegriffen wurde.

Aufgrund des Zeitpunkts der Konflikterkennung lassen sich zwei Klassen von Implementierungsstrategien für Transaktionen unterscheiden. Die prädikative Validierung erkennt Konflikte mit konkurrierenden Transaktionen a posteriori am Transaktionsende und beseitigt sie durch Zurücksetzen von Transaktionen. Dagegen wird das Auftreten von Konflikten beim prädikativen Sperren a priori durch Blockieren konkurrierender Transaktionen verhindert. Da keine dieser Strategien in beliebigen Anwendungsumgebungen und bei beliebigen Transaktionslasten eine effiziente Transaktionsausführung gewährleistet, werden diese Strategien zu einer adaptiven Methode kombiniert. Bei dieser Implementierungsstrategie passt der Scheduler die Parallelitätskontrollmethode dynamisch an die Menge konkurrierender Transaktionen und an die von ihnen benötigten Daten an.

Abschliessend werden die konkreten Realisierungen des Transaktionskonzepts der Datenbankprogrammiersprache DBPL im Rahmen des DBPL- und des LIDAS-Projekts betrachtet. In diesem Zusammenhang werden auch Fragen der Transaktionsmodellierung und -generierung diskutiert.

Abstract

Database models encompass not only concepts for data structuring and data manipulation but also concepts for the preservation of data integrity. Compound operations on the data structures of a database that are guaranteeing the invariance of the database integrity constraints can be formulated as transactions. In this thesis all aspects of an integration of the database concept 'transaction' into a database programming language are investigated in the context of a relational database model and Pascal-like programming languages.

The basis for the programming language integration is the procedure concept that is adapted gradually to the requirements of transaction programming. Since transactions operate only on certain parts of a database in general, access rights are restricted to subrelations, so-called selected relations, in the transaction definition. Special mechanisms are introduced for access control and for parameter passing. Selected relations are based on relational predicates of first-order calculus for data selection.

The definition of the semantics of the transaction declaration, call, and execution considers particularly the concurrent execution of transactions. The definition of a correct parallel execution is based on the axiomatic specification of the concurrent transaction by means of preconditions and postconditions. From these conditions a postcondition of the concurrent execution can be derived. If all operations of a transaction and their semantic meaning are known in advance then from the derived postcondition a correct scheduler can be generated for the parallel execution of a given set of transactions. If, however, a transaction is abstracted from the individual operations then strategies for concurrency control must be applied that are scheduling transactions on the information which data objects are accessed predicatively.

By the time of conflict detection two classes of strategies for the implementation of transactions can be distinguished. Predicative validation detects conflicts to concurrent transactions a posteriori at transaction commit and resolves conflicts by aborting transactions. On the contrary the occurence of conflicts is prevented a priori with predicative locking through delaying concurrent transactions. Since these strategies do not guarantee an efficient execution of transactions in arbitrary application environments and by arbitrary transaction profiles, these strategies are combined to an adaptive method. By this implementation strategy the scheduler adapts the concurrency control policy dynamically to the set of concurrent transactions and to the required data.

The transaction concept of the database programming language DBPL is implemented in the context of the DBPL and LIDAS projects. Thereby, the modelling and generation of transactions is also discussed.