In: Proc. of the 7th Workshop on Transaction Concepts, Nieheim-Ostwestfalen, Germany, Jan. 1996. GI Datenbank Rundbrief 17, May 1996.

Eine WAL-basierte und objektorientierte *Recovery*-Strategie

Fernando de Ferreira Rezende[1] und Thomas Baier

Universität Kaiserslautern - Fachbereich Informatik
Postfach 3049 - 67653 Kaiserslautern - Deutschland
Telefon: (0631) 205 3274 - Telefax: (0631) 205 3558
E-Mail: rezende@informatik.uni-kl.de

Wir stellen eine **WAL**-basierte und **o**bjektorientierte ***R**ecovery*-**S**trategie (WALORS) vor. WALORS ist geeignet für den Einsatz in objektorientierten Datenbanksystemen, da der *Recovery*-Mechanismus objektbasiert arbeitet und ein Objekt als logische Einheit betrachtet wird. Zur effektiven Realisierung der *Recovery*-Komponente wird jedes Objekt mit einer LSN (*log sequence number*) ausgestattet. Die LSN beschreibt den logischen Zeitpunkt, zu dem das Objekt das letzte Mal modifiziert wurde. Jede Änderungsoperation wird in der Log-Datei in Form eines Log-Satzes mitprotokolliert. Bei der Gewinnung der Log-Informationen benutzt WALORS die Strategie des Eintrags-*Logging*, welches auf der Objekt-Ebene durchgeführt wird. Jeder Log-Satz enthält unter anderem auch eine LSN. Die LSN ist monoton wachsend, der jeweils letzte Log-Satz der Log-Datei erhält demnach die bis zu diesem Zeitpunkt größte LSN. Die LSN wird nicht explizit im Log-Satz vermerkt, sie wird durch die Adresse des Log-Satzes in der Log-Datei repräsentiert. So kann man sehr effizient auf einen bestimmten Log-Satz in der Log-Datei zugreifen, ohne daß mit Hilfe von komplizierten Hash-Verfahren aus einer gegebenen LSN die Adresse des entsprechenden Log-Satzes ermittelt werden muß. Durch die Zuordnung einer LSN zu jedem Log-Satz besitzt jede Änderungsoperation eine Art logischen Zeitstempel. Wenn man diesen Stempel auch dem Objekt aufdrückt, welches durch die entsprechende Änderungsoperation modifiziert wurde, kann in einem Fehlerfall sehr effizient entschieden werden, ob auf dieses Objekt *Recovery*-Maßnahmen anzuwenden sind oder nicht.

Es werden Transaktions-*Recovery*, *Crash-Recovery* und Platten-*Recovery* unterstützt. Aktive Transaktionen können sowohl teilweise (*partial rollback*) als auch vollständig (*total rollback*) zurückgesetzt werden. Bei einem Systemwiederanlauf wird die Log-Datei in drei Durchläufen abgearbeitet. Es erfolgen ein ANALYSE-, ein *UNDO*- und ein sich anschließender *REDO*-Lauf. WALORS arbeitet (im Gegensatz zu ARIES [MHLPS89, MoNa94, RoMo89]) mit einer selektiven *REDO*-Strategie. Es werden lediglich verlorene Änderungen von Gewinnertransaktionen wiederholt. Dieses Vorgehen ist vorteilhaft, da teuere E/A-Operationen eingespart werden und die Effizienz beim Systemwiederanlauf nach einem Fehler erhöht wird [BHG87]. Bei einer *REDO*-Strategie hingegen, die alle nicht materialisierten Änderungen wiederholt (*redo all*), wird ein Teil der Operationen nur durchgeführt, um sie im anschließenden *UNDO*-Lauf wieder zurückzusetzen. Der Vorteil des selektiven *REDO* in Verbindung mit selektivem *UNDO* ist dadurch gekennzeichnet, daß Änderungen von Verlierertransaktionen, die nicht materialisiert wurden, keinen physischen *UNDO*- bzw. *REDO*-Aufwand verursachen.

Um festzustellen, welche der vollzogenen Änderungen zurückgesetzt werden müssen, wird der LSN-Wert des materialisierten Objekts benutzt. Eine Änderung muß zurückgesetzt werden, wenn die Objekt-LSN größer oder gleich der LSN des zugehörigen *UNDO*-Log-Satzes ist. Ist die Objekt-LSN kleiner als die des Log-Satzes, dann wurde die Änderung noch nicht in die Datenbank eingebracht und deshalb ist ein *UNDO* dieser Änderung auch nicht notwendig. Andererseits muß eine Änderung wiederholt werden, wenn sie noch nicht materialisiert wurde. Dazu wird auch hier die LSN des entsprechenden Log-Satzes mit der Objekt-LSN verglichen. Ist die Objekt-LSN kleiner als die LSN des zugehörigen *REDO*-Log-Satzes, dann ist eine Wiederholung der Änderungsoperation notwendig.

Auf der Basis vergebener LSN-Werte wird auch das WAL-Protokoll (*write ahead logging* [Gray78]) realisiert. Im Log-Puffer werden temporär die Log-Sätze gehalten. Unter Beachtung des WAL-Protokolls werden erst dann die Log-Sätze in die Log-Datei geschrieben, wenn ein modifiziertes Objekt aus dem Datenpuffer verdrängt werden soll. In diesem Fall wird die Log-Datei mindestens bis zu der LSN geschrieben, die der Objekt-LSN entspricht. Das WAL-Protokoll garantiert, daß die zugehörigen Log-Sätze in der Log-Datei protokolliert werden (*UNDO*-Informationen),

1. Finanziell unterstützt von CNPq (Bundesrat für wissenschaftliche und technologische Entwicklung) des Bundesministeriums für Wissenschaft und Technologie von Brasilien.

bevor irgendwelche Änderungen in die Datenbank eingebracht werden dürfen. Andererseits, wenn eine Transaktion erfolgreich abgeschlossen wird, müssen spätestens bis zum *Commit* alle Log-Sätze dieser Transaktion in die Log-Datei geschrieben werden (*REDO*-Informationen).

WALORS unterstützt eine flexible Pufferverwaltung. Es werden sowohl die Ersetzungsstrategien *STEAL* und *NOSTEAL* als auch die Ausschreibstrategien *FORCE* und *NOFORCE* unterstützt [HaRe83]. Zur Verminderung des Wiederanlaufaufwandes nach dem Auftreten eines Systemfehlers arbeitet WALORS mit *Fuzzy-Checkpoints*, die das Herausschreiben von geänderten Objekten nicht erzwingen. Im wesentlichen werden nur Statusinformationen in die Log-Datei geschrieben. Dadurch ist der Aufwand für das Erzeugen eines Sicherungspunktes sehr gering.

Für die Protokollierung von Änderungen, die zurückgesetzt werden sollen, wird das Schreiben eines CLRs *(compensation log record)* veranlaßt. Der Aufbau eines CLRs unterscheidet sich nicht wesentlich vom Aufbau eines herkömmlichen Log-Satzes. Ein CLR enthält *REDO*-Informationen und einen Zeiger auf den Log-Satz derselben Transaktion, der in der Rückwärtsverkettung vor dem Log-Satz der zurückzusetzenden Änderung steht. Durch die Existenz eines CLRs für jedes Zurücksetzen einer Änderungsoperation kann festgestellt werden, ob die protokollierte Rücksetzoperation bereits in die materialisierte Datenbank eingebracht wurde.

Der CLR dient auch zur Durchsetzung der Idempotenz der *Recovery*-Maßnahmen. Anhand der CLRs kann man beim Wiederanlauf des Systems erkennen, ob die durch die CLRs beschriebenen *UNDO*-Operationen bereits materialisiert wurden. Aber im Gegensatz zu dem Vorgehen im Normalbetrieb kann beim Zurücksetzen einer Transaktion nicht den CLR-Zeigern gefolgt werden, weil das bedeuten würde, daß möglicherweise Log-Sätze von noch nicht materialisierten Änderungen übersprungen werden würden. (Der Grund dafür ist, daß WALORS selektives *REDO* macht und kein *REDO-ALL* Strategie folgt.) Die Konsequenz ist, daß man jeden Log-Satz einer Transaktion untersuchen muß. WALORS löst das Problem durch die Verwaltung von zwei Ketten für die Verlierertransaktionen, eine *NON-UNDO*-Kette und eine *CLR-EXIST*-Kette (beide besitzen LIFO-Eigenschaften (*last in first out*)). Bei der Untersuchung von CLRs wird ein Eintrag in einer der beiden Ketten erzeugt. Wenn die *UNDO*-Operation bereits durchgeführt wurde, gelangt ein Eintrag in die *NON-UNDO*-Kette. Wurde der CLR noch nicht materialisiert, d. h. die *UNDO*-Operation noch nicht durchgeführt, so wird ein Eintrag in der *CLR-EXIST*-Kette erzeugt. Beim Lesen eines *UPDATE*-Log-Satzes wird unter Verwendung der beiden Ketten entschieden, ob der Log-Satz zurückgesetzt werden muß. Existiert in der *NON-UNDO*-Kette für die Änderung ein Eintrag, so ist die Änderung bereits zurückgesetzt worden. Existiert für die Änderung ein Eintrag in der *CLR-EXIST*-Kette, dann wird in Abhängigkeit davon, ob die Änderung selbst bereits materialisiert wurde, die *UNDO*-Operation durchgeführt. Existiert weder in der *NON-UNDO*- noch in der *CLR-EXIST*-Kette ein Eintrag für die Änderungsoperation, so wird, wenn die Änderung bereits materialisiert wurde, ein CLR geschrieben und die Änderung zurückgesetzt. Mit diesem Verfahren gewährleistet WALORS, daß die *UNDO*-Operationen in der richtigen Reihenfolge zurückgesetzt und keine CLRs von CLRs geschrieben werden.

Schließlich arbeitet WALORS mit speziellen *UPDATE*-Log-Sätzen für Lösch- und Einfügeoperationen (*UPDATE_delete_obj* bzw. *UPDATE_create_obj*), für die spezielle CLR-Log-Sätze eingeführt wurden (*CLR_create_obj bzw. CLR_delete_obj*). Für detaillierte Informationen sei auf [ReBa96, Baier95] verwiesen, wo auch die Erweiterungen von WALORS für *Client/Server*-Systeme und für geschachtelte Transaktionen beschrieben sind.

Literaturverzeichnis

[Baier95] Baier, T.: *WALORS - Eine WAL-basierte und objektorientierte Recovery-Strategie*. Diplomarbeit, Univ. Kaiserslautern, Nov. 1995.

[BHG87] Bernstein, P.A., Hadzilacos, V., Goodman, N.: *Concurrency Control and Recovery in Database Systems*. Addison-Wesley , USA, 1987.

[Gray78] Gray, J.N.: Notes on Database Operating Systems. In: *Operating Systems: An Advanced Course*, Springer-Verlag, Berlin, Germany, 1978. (Lecture Notes in Computer Science No. 60).

[HaRe83] Härder, T., Reuter, A.: Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys* **15** (4), 1983.

[MHLPS89] Mohan, C., Haderle, D., Lindsay, B.G., Pirahesh, H., Schwarz, P.: *ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging*. IBM ARC Research Report RJ6649, San Jose, USA, Jan. 1989.

[MoNa94] Mohan, C., Narang, I.: ARIES/CSA: A Method for Data Base Recovery in Client-Server Architectures. In: *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Minneapolis, USA, Mai 1994.

[ReBa96] Rezende, F.F., Baier, T.: *WALORS - A WAL-Based and Object-Oriented Recovery Strategy*. Internal Report, Univ. Kaiserslautern, Jan. 1996.

[RoMo89] Rothermel, K., Mohan, C.: ARIES/NT: A Recovery Method Based on Write-Ahead Logging for Nested Transactions. In: *Proc. of the 15th Int. Conf. on Very Large Data Bases*, Amsterdam, Holland, Aug. 1989. pp. 337-346.