Wiederherstellung (Recovery)

Fragestellungen

Aufgaben der Komponenten für das Recovery:

- Sicherstellung der Dauerhaftigkeit der gespeicherten Daten, d.h. Daten, die in einer Transaktion einmal bestätigt wurden (*commit*), bleiben auch bei Systemausfällen erhalten.
- Sicherstellung der Konsistenz der gespeicherten Daten d.h. bei einem Systemausfall inmitten einer laufenden Transaktion muss nach dem Neustart des DBMS der konstistente Zustand vor Beginn dieser Transaktion wieder hergestellt werden (*rollback*).
- Sicherstellung der Konsistenz der Daten, wenn eine Anwendung eine Transaktion abbricht.

Arten von Fehlern/Ausfällen

- Transaktionsabbruch
  Eine Transaktion wird absichtlich vom Benutzer oder einem Programm abgebrochen. Es kann auch sein, dass das DBMS eine Transaktion abbricht, etwa im Falle einer Verklemmung.

  In diesem Fall ist ein Rollback (*undo*) notwendig.[1]
- Systemabsturz
  Das DBMS oder das Betriebssystem stürzt ab, aus welchem Grund auch immer. Alle zu diesem Zeitpunkt im Hauptspeicher befindlichen Informationen im Hauptspeicher gehen verloren.

  In diesem Fall müssen abgeschlossene Transaktionen, deren Daten noch nicht im Sekundärspeicher angelegt sind, ganz oder teilweise wiederholt werden (*redo*). Noch nicht abgeschlossene Transaktionen müssen rückgängig gemacht werden, sofern Teile der Veränderungen bereits im Sekundärspeicher durchgeführt wurden (*undo*).

[1] Da Transaktionen verschränkt durchgeführt werden, kann es sein, dass das Rollback einer Transaktion auch das Rollback einer anderen Transaktion erforderlich macht, um die Konsistenz der Datenbank zu garantieren. Dieses Phänomen nennt man *kaskadierendes* Rollback. Wenn man zur Synchronisation Sperrverfahren nach dem *strikten* 2-Phasen-Sperrprotokoll verwendet, entstehen nur so genannten *rücksetzbare* Abläufe und das Phänomen der kaskadierenden Rollbacks kann nicht auftreten.

- Medienfehler
  Plattencrash o.ä., d.h. nicht nur der Hauptspeicherinhalt geht im Fehlerfall verloren, sondern auch Daten auf dem Sekundärspeicher.

  In diesem Fall werden redundant gehaltene Daten benötigt: Backup (Online- oder Offline-Backup). Und alle ab dem letzten Backup-Zeitpunkt abgeschlossenen Transaktionen müssen reproduziert werden (*redo*), dazu werden Log-Dateien benötigt.

Im Beispiel von Abb. 1 sehen wir 5 Transaktionen, die noch nicht abgeschlossen sind, wie T_3 und T_4 oder deren Ergebnisse noch nicht vollständig auf dem Sekundärspeicher geschrieben sind.

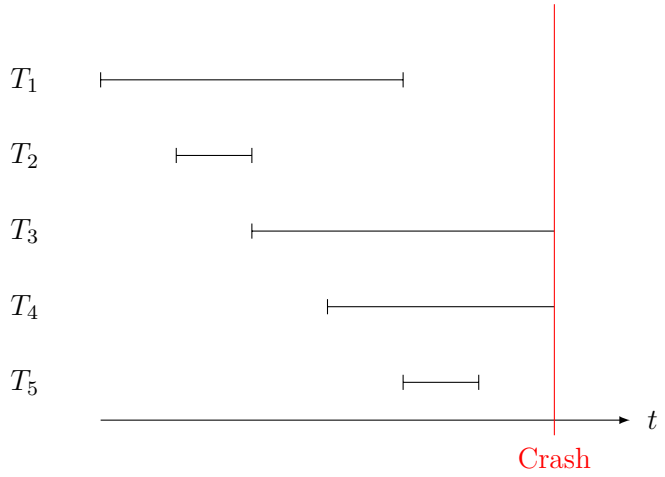

Abbildung 1: Beispiel eines Systemabsturzes

Im Falle eines Absturzes zu dem mit der roten Linie gekennzeichneten Zeitpunkt müssen also die Transaktionen T_1, T_2, T_5 wiederholt werden (*redo*), während die Transaktionen T_3, T_4 zurückgesetzt werden müssen (*undo*).

Arten von Speicher

- Volatiler Speicher
  Hauptspeicherbereich, den die Pufferverwaltung kontrolliert. In diesem Speicher gehaltene Daten sind bei einem Systemabsturz verloren.

- Sekundärspeicher
  Speicher, in der Regel Festplatten, in dem die eigentlichen Daten- und Indexdateien etc. gespeichert sind. Dieser Speicher ist nach

einem Systemabsturz unverändert intakt. Inhalte gehen aber bei einem Medienfehler verloren.

- Stabiler Speicher
  Sekundärer Speicher, der *alle* Arten von Abbrüchen überlebt. Für stabilen Speicher werden heute oft Systeme mit RAID o.ä. eingesetzt.

Man beobachte, wie die drei Arten von Speichern mit den drei Arten von Ausfällen korrespondieren.

Protokoll für das Recovery

Konzept des *Write-Ahead Logging Protocol*s (WAL)

Bei Änderung von Daten wird folgendermaßen verfahren:

- Änderungen an Daten werden in der entsprechenden Seite im Puffer durchgeführt.
- Ein Logeintrag mit der Seite vor der Änderung (*before image*) und nach der Änderung (*after image*) wird in eine Logdatei geschrieben, die sich auf stabilem Speicher befindet.
- Eine Seite ersetzt (später) einen Block im sekundären Speicher.
- Außerdem wird der Transaktionszustand in der Logdatei gespeichert.
- Eine Transaktion darf ein Commit erst dann abschließen und den Abschluss der Transaktion im Log vermerken, wenn alle Redo-Logeinträge im stabilen Speicher geschrieben sind.

Das Prinzip besteht also darin, dass zunächst die Logeinträge geschrieben werden und erst dann Änderungen an den Blöcken im sekundären Speicher gemacht werden – dashalb der Name *Write-ahead logging*.

DBMS verwenden in der Regel als Protokoll ARIES. ARIES ist eine Technik, die in einem Papier von C. Mohan et al.[2] von IBM beschrieben wird.

Wir wollen in der Vorlesung nur die grundlegenden Konzepte betrachten.

[2] C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, Peter Schwarz *ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging*, ACM Transactions on Database Systems, Vol 17, No. 1, March 1992, 94-162

Logeinträge

Konzeptionell ist ein Log eine sequenzielle Datei, in der jeder Eintrag eine Sequenznummer hat (LSN *log sequence number*). Die Logdatei wird auf stabilen Speicher geschrieben.

Logeinträge mit Informationen über den Zustand einer Transaktion[3] haben folgende Inhalte:

- [T_i, begin]
- [T_i, commit]
- [T_i, abort]

Logeinträge mit Informationen über die Änderung von Daten enthalten folgende Informationen:

- [T_i, b_id, op, before-image, after-image]

Dabei steht b_id für die Identifizierung des Blocks, der geändert wurde, op für die Art der Operation. Denkbar sind auch Implementierungen, bei denen eine RecordId an Stelle der b_id verwendet wird.

DBMS haben in der Regel den Mechanismus eines *Checkpoints*. Ein Checkpoint wird ins Log geschrieben, wenn alle gepufferten Seiten auf den Sekundärspeicher geschrieben wurden. Dies kann z.B. automatisch mit einer eingestellten Periode passieren. Der Logeintrag zu einem Checkpoint enthält dann die Id aller Transaktionen, die zum Zeitpunkt des Checkpoints noch aktiv waren:

- [checkpoint T_i, T_j, ...]

Der Vorteil von Checkpoints besteht darin, dass man bei einem Recovery nur die Einträge *nach* dem letzten Checkpoint erneut durchführen muss, wenn ein *redo* erforderlich ist. Und bei einem *undo* muss man nur noch die Transaktionen analysieren, die bei einem Checkpoint noch aktiv waren.

[3] Die Art der hier beschriebenen Logeinträge nennt man auch *physisches Log*, bei dem die geänderten Seiten ins Logbuch geschrieben werden. Stattdessen kann man auch die Anweisungen für die Datenbank-Operationen (eventuell mit ihren Kompensationen) ins Logbuch schreiben, nicht die wirklich veränderten Seiten. Dann spricht man von einem *logischen Log*.

Vorgehen beim Recovery

Ablauf bei Abbruch einer Transaktion

Lese das Log rückwärts und führe für alle Einträge zur abgebrochenen Transaktion ein *undo* durch.

Ablauf der Wiederherstellung bei Systemcrash

1. Ermittle den letzten korrekten Zustand (Checkpoint) und betrachte diesen als Anfang der Logdatei.
2. Stelle eine leere Undo-Liste und eine leere Redo-Liste bereit.
3. Lese das Log vorwärts:
   bei $[T_i, \text{begin}]$: schreibe T_i in die Undo-Liste
   bei $[T_i, \text{commit}]$: lösche T_i in der Undo-Liste, schreibe T_i in die Redo-Liste.
4. Lese nun das Log vom Ende rückwärts:
   für alle Änderungen von Transaktionen in der Undo-Liste: führe ein undo durch.
5. Lese nochmals das Log vorwärts:
   für alle Änderungen von Transaktionen in der Redo-Liste: führe ein redo durch.
6. Benachrichtige alle Transaktionen in der Undo-Liste über den Abbruch.

Ablauf bei Medienfehler

1. Spiele Backup ein und setze den Anfang der Logdatei auf den Zeitpunkt des Backups.
2. Führe nun mittels des Logs das Recovery wie oben seit diesem Zeitpunkt durch.

Burkhardt Renz
TH Mittelhessen
Fachbereich MNI
Wiesenstr. 14
D-35390 Gießen

Rev 3.1 – 11. Mai 2014