Übungsblatt 4

Adrian Schollmeyer

Aufgabe 1

Shadow Paging Beim Shadow Paging werden Änderungen nicht sofort in den globalen Datenrum geschrieben. Stattdessen werden sie bis zum Commit-Zeitpunkt in einem privaten Datenraum für jede Transaktion vorgehalten und erst im Fall eines Commits in den globalen Datenraum übernommen.

Um eine inkonsistente Sicht von bereits laufenden Transaktionen auf Daten zu verhindern, die von einer Transaktion committet werden, lesen deise solche Daten aus ihrem privaten Datenraum, sodass sie immer auf den Datenbestand zum Zeitpunkt des Transaktionsbeginns zugreifen.

Recovery Zeil von Recovery ist die Garantie der Datensicherung ohne Intervention, um im Fehlerfall automatisch die Datenbank auf einen konsistenten Zustand zurücksetzen zu können. Recovery-Komponenten sind zuständig für die Sicherung der Atomaritäts- und Dauerhaftigkeitseigenschaften einer Transaktion.

An der Recovery beteiligte Systemkomponenten An der Recovery beteiligt ist der Speicher-Manager (SM), der die Schnittstelle zwischen flüchtigem und stabilem Speicher bildet und nochmals in Recover-Manager (RM) und Puffer-Manager (PM) unterteilt ist.

Der RM ist dabei zuständig, dass, alle Änderungen einer „committed“ Transaktion auch tatsächlich im stabilen Speicher sind, keine Änderungen von aktiven oder abgebrochenen Transaktionen im stabilen Speicher sind und die Datenbank nach einem Fehler in einen konsistenten Zustand gebracht wird.

Der PM verwaltet den Puffer (DB- und Log-Puffer) und holt dabei Seiten vom stabilen Speicher in den Puffer bzw. schreibt Daten vom Puffer in den stabilen Speicher oder ersetzt Seiten im Falle eines Pufferüberlaufs.

Fehlerklassen Unterschieden wird zwischen Transaktionsfehlern, Systemfehlern, Mediafehlern und Kommunikationsfehlern. Je nach Fehlerart sind unterschiedliche Recoverymaßnahmen nötig.

Transaktionsfehler haben den Abbruch einer Transaktion zur Folge und beeinflussen den Rest des Systems nicht. Dies können explizite Transaktionsabbrüche des Benutzers, des Systems oder auch Softwarefehler im Anwendungsprogramm sein, die einen Abbruch der Transaktion auslösen. Behandelt werden diese durch isoliertes Zurücksetzen der Änderungen der abgebrochenen Transaktion.

Systemfehler haben die Zerstörung des Hauptspeichers zur Folge (z. B. der Absturz eines DBMS, OOM-Killer). Diese betreffen nicht den Hintergrundspeicher.

Behandelt werden diese, indem Änderungen von noch nicht beendeten Transaktionen, die bereits in die DB eingebracht wurden, zurückgesetzt werden. Zudem müssen die Änderungen von abgeschlossenen Transaktionen, die noch nicht in die DB eingebracht wurden, nachvollzogen werden.

Mediafehler haben einen Fehler im Speichermedium als Ursache, bspw. „Head-Crashes“, Controller-Fehler oder Naturgewalten. Als Folge tritt der Verlust stabiler Datenbankdaten auf. Behandelt werden diese durch Einspielen eines Backups/DB-Archivs von einem anderen Medium.

Kommunikationsfehler entstehen bei der Kommunikation zwischen Client und Server bzw. zwischen Knoten des Datenbankclusters. Auftretende Klassen sind dabei Nachrichtenfehler (verlorene Nachrichten), Netzwerkpartitionierung (Trennung des Netzwerks) und Performanceprobleme. Mit lokalen Protokollen können dabei entweder durch einen lokalen RM Atomarität und Isolation lokaler Transaktionen sichergestellt werden. Mit verteilten Protokollen werden verteilte Transaktionen behandelt. Dies umfasst auch Commit- und Recovery-Protokolle.

Recoveryklassen

Aufbau Logbuch Das Logbuch ist eine Folge von Änderungen, die durch Transaktionen durchgeführt werden. Jeder Eintrag besteht aus:

- LSN (Log Sequence Number), eindeutige steigende Durchnummerierung der Log-Einträge
- TA (Transaktionskennung), Verweis auf die Transaktion, welche die Änderung durchführt.
- PageID (Seitennummer)
- Redo (Redo-Information)
- Undo (Undo-Information)
- PrevLSN (Verweis auf Vorgänger-Logeintrag derselben Transaktion)

Im Logbuch werden dann alle Aktionen der Transaktionen festgehalten, die die Daten im stabilen Speicher im Fall beeinflussen könnten (Transaktionsbeginn, Schreiboperationen, Commits, Aborts). Leseoperationen werden nicht erfasst, da sie den Zustand des Speichers nicht beeinflussen. In Redo und Undo wird erfasst, welche Aktionen zum wiederholen bzw. rückgängig machen der im Logeintrag aufgeführten Operation nötig sind.

WAL Bei WAL werden vor jedem Commit zunächst alle zugehörigen Logeinträge einer Transaktion ausgeschrieben, um ein späteres Redo zu ermöglichen. Vor dem auslagern einer modifizierten Seite werden zunächst alle zugehörigen Log-Einträge in das Log-Archiv geschrieben, um Undo bei abgebrochenen Transaktionen zu ermöglichen. In der Grundidee loggt WAL also zunächst also alle geplanten Änderungen, bevor diese tatsächlich auf dem Speicher ausgeführt werden, sodass im Fehlerfall

anhand des Logs und der letzten Sicherung ein konsistenter Zustand wiederhergestellt werden kann.

Aufgabe 2

Aufgabe 3

NO-UNDO verbietet das Auslagern geänderter Seiten vor dem Commit. NO-REDO wiederum erzwingt das Auslagern geänderter Seiten zum Commit. Greifen zwei Transaktionen T_1, T_2 auf Seite P auf die Datenobjekte A und B zu, wird für beide Transaktionen im Puffer ein jeweils anderes After Image erzeugt, welches die Änderung der jeweils anderen Transaktion nicht beinhaltet, sofern beide Transaktionen starten, bevor die jeweils anderen Transaktion ihre Änderung committen könnte (d. h. sie haben beide das gleiche Before Image). Welche der beiden Transaktionen nun zuletzt committen würde, würde die geänderte Seite der Vorgängertransaktion überschreiben müssen.

Aufgabe 4

Vorteile:

- Die Arbeit auf Kopien macht es einfach möglich, Änderungen rückgängig zu machen. Dafür müssen die Kopien lediglich nicht gespeichert werden.
- Durch Veröffentlichung der Kopien ist eine atomare Sichtbarmachung des neuen (konsistenten) Datenbankzustands möglich.
- Bei der Herstellung der Isolationseigenschaft ist es hilfreich, dass Zwischenzustände bei Shadow Paging für Dritte nicht sichtbar sind.
- Sicherung und Abbruch sind relativ kostengünstig, da sie nur auf Metadaten arbeiten.

Nachteile:

- Prinzipiell entstehen durch die Anfertigung von Kopien hohe Kosten. Diese können gemindert werden, in dem tatsächliche physische Kopien erst dann angefertigt werden, wenn es zu Änderungen auf der Seite kommt (sprich: sobald sich die Kopie vom Original unterscheiden würde). Weiterhin können die Kopien in ihrer Größe beschränkt werden, sodass nur die Ausschnitte des Originals kopiert werden, die sich tatsächlich unterscheiden.
- Der Transaktionsmanager benötigt die Mitarbeit der Objektmanager, da diese Schattenkopien erzeugen sowie deren atomare Sicherung/Vernichtung realisieren müssen.

Aufgabe 5

Prinzipielles Log:

(1) T_1, begin

(2) T_2, begin

(3) T_1, A, redo: $a := 950$, undo: $a := 1000$

(4) T_2, C, redo: $c := 3100$, undo: $c := 3000$

(5) T_1, B, redo: $b := 2050$, undo: $b := 2000$

(6) T_1, commit, EOT

(7) T_2, A, redo: $a := 900$, undo: $a := 950$

(8) T_2, commit, EOT

Zu beachten: T_2 überschreibt $a := 900$, da zum Zeitpunkt BOT_2 noch $a = 1000$ war. Beim undo hingegen muss hier schon $a := 950$ gesetzt werden, da T_1 committet ist.

Aufgabe 6

(1) T_1, begin

(2) T_2, begin

(3) T_1, A, redo: $a := 950$, undo: $a := 1000$

(4) T_2, C, redo: $c := 3100$, undo: $c := 3000$

(5) T_1, B, redo: $b := 2050$, undo: $b := 2000$

(6) T_1, commit, EOT

(7) T_2, A, redo: $a := 900$, undo: $a := 950$

(8) (fehlend: T_2, commit, EOT)

(9) Kompensationseintrag (4) T_2: C, $c := 3000$

Aufgabe 7

(a)

- In Analysephase wird T_1 als abgeschlossen erkannt, T_2 als aktiv.
- A, B, C werden als Dirty Pages erkannt.
- `firstLSN` $= 3$
- `lastLSN`$(T_2) = 7$
- Redo wird ausgeführt
- CLR: Undo (4) T_2, `UndoNxtLSN` $= 7$

(b)

Sollten keine LSN-Einträge in die Datenbank geschrieben, kann es bei wiederholten Abstürzen dazu kommen, dass Redo mehrfach angewendet wird. Ohne Speicherung der LSN sind die Operationen nicht mehr idempotent und können zu einem inkonsistenten Datenbankzustand führen.