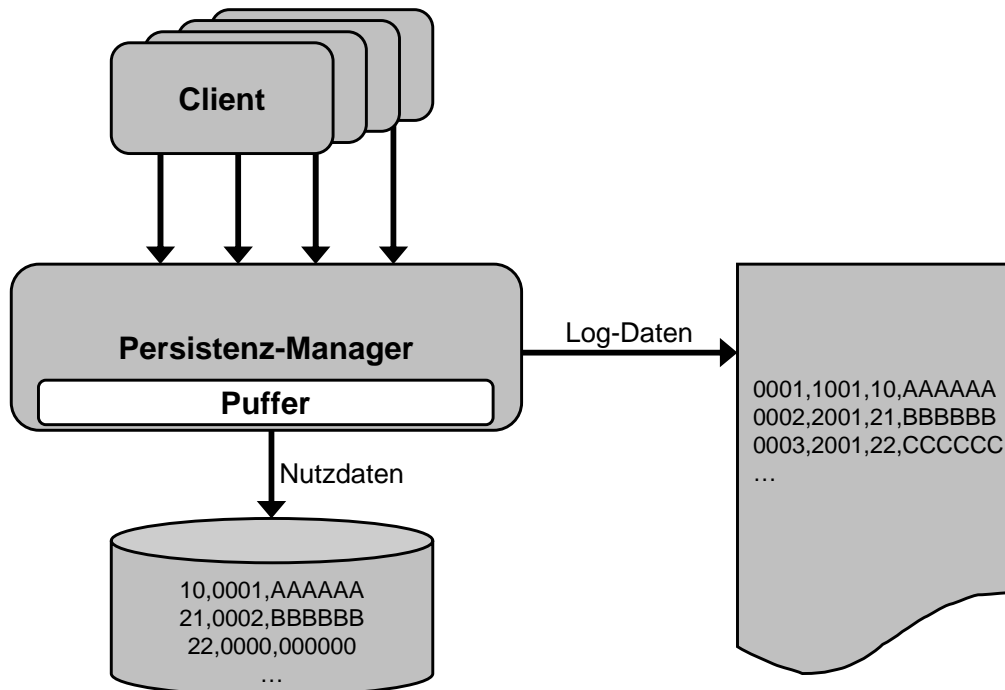
	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Logging und Recovery
	Bearbeitungsbeginn	KW21
	Bearbeitungsende	KW24

## Aufgabe 5: Logging und Recovery




### 1. Persistenzverwaltung

Realisieren Sie nach obiger Architektur einen stark vereinfachten Persistenz-Manager, der nebenläufige Zugriffe von Clients auf Nutzdaten gestattet, geänderte Daten verzögert auschreibt und Vorkehrungen für die Wiederherstellung eines konsistenten Datenbestands nach einem Systemabsturz trifft.

Der Persistenz-Manager verwaltet dazu neben dem Nutzdatenbestand auch Logdaten, um noch nicht ausgeschriebene Änderungsoperationen festzuhalten. Nutzdaten und Logdaten müssen auf persistenten Speichern abgelegt werden. Nutzdaten werden für die Speicherung um eine Seiten-ID und eine Log-Sequenznummer ergänzt, Logdatensätze um eine Sequenznummer, eine Transaktions-ID sowie eine Seiten-ID.

Durch Schreiboperationen geänderte Nutzdaten werden zunächst nur im internen Puffer des Persistenz-Managers abgelegt. Dabei können auch bestehende ältere Versionen direkt im Puffer überschrieben werden, ohne dass sie zuvor auf den persistenten Speicher ausgeschrieben wurden. Wenn der Puffer nach einer Schreiboperation mehr als fünf Datensätze enthält, werden die Datensätze abgeschlossener Transaktionen auf den persistenten Speicher ausgeschrieben (non-atomic). Auf diese Weise können veraltete Datensätze im persistenten Speicher sowie auszuschreibende Datensätze abgeschlossener Transaktionen im Puffer existieren, nicht jedoch „schmutzige“ Datensätze noch nicht abgeschlossener Transaktionen im persistenten Speicher (noforce, nosteal). Log-Informationen müssen jedoch unverzüglich und insbesondere vor Abschluss einer Transaktion auf den persistenten Speicher geschrieben werden, da sonst im Falle eines Systemabsturzes keinerlei Ansatzpunkt für eine Recovery existiert.

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Logging und Recovery
	Bearbeitungsbeginn	KW21
	Bearbeitungsende	KW24

Im System existiert nur ein Persistenz-Manager, auf den mehrere Clients nebenläufig zugreifen (denken Sie an das Singleton-Pattern und Thread-Sicherheit beim nebenläufigen Zugriff). Im Einzelnen soll der Persistenz-Manager mindestens folgende Operationen anbieten:

`beginTransaction()`

Beginnt eine neue Transaktion. Der Persistenz-Manager erzeugt eine eindeutige Transaktions-ID und liefert diese als Rückgabewerte an den Client.

`commit(int taId)`

Schließt die angegebene Transaktion ab.

`write(int taId, int pageId, String data);`

Schreibt im Rahmen der angegebenen Transaktion die angegebenen Nutzdaten mit der angegebenen Seiten-ID in den Puffer. Die Nutzdaten ersetzen dabei die evtl. vorhandenen Inhalte der angegebenen Seite vollständig.

## 2. Clients

Realisieren Sie eine Client-Klasse, die in mehreren Instanzen parallel gestartet werden kann, so dass die einzelnen Instanzen nebenläufig auf den Persistenz-Manager zugreifen.

Die Clients führen wiederholt einfache Transaktionen nach dem folgenden Schema auf dem Persistenz-Manager durch:


`beginTransaction() write() write() ... commit()`

Zwischen den einzelnen Operationen sollen die Clients kurze Pausen einlegen, um ein realistisches Verhalten anzunähern. Die Anzahl der Schreiboperationen kann variieren.

Um ohne Sperrverwaltung auszukommen, greifen die Clients paarweise disjunkt auf die Seiten zu, d.h. Client 1 auf die Sätze 10..19, Client 2 auf die Sätze 20..29 usw. Innerhalb einer Transaktion können durchaus mehrere Seiten verändert werden. Die eigentlichen Nutzdaten können einfache Strings sein.

## 3. Recovery-Werkzeug

Aufgrund der noforce, nosteal, non-atomic Implementierung des Persistenz-Managers ist keine Undo-Recovery nach Systemabstürzen erforderlich, wohl aber eine Redo-Recovery. Realisieren Sie ein Recovery-Werkzeug, das die Analyse- und Redo-Phase im Rahmen der Crash-Recovery wie in der Vorlesung besprochen durchführt. Aus den Logdaten müssen dazu zunächst die so genannten Gewinner-Transaktionen bestimmt und anschließend deren ausstehende Schreiboperationen auf dem Nutzdatenbestand durchgeführt werden. Denken Sie dabei auch an die Aktualisierung der LSNs, um die Idempotenz der Recovery sicherzustellen.

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Logging und Recovery
	Bearbeitungsbeginn	KW21
	Bearbeitungsende	KW24

## Hinweise

- Persistenz-Manager und Clients lassen sich als Bestandteile einer Java-Application implementieren. Auf diese Weise können die Clients als einzelne Threads gestartet werden, die auf eine einzige Instanz des Persistenz-Managers zugreifen (Singleton-Pattern).
- Für die Speicherung der Nutz- und Log-Daten verwenden Sie folgendes Vorgehen: Speichern Sie jede Seite in einer einzelnen (Text-)Datei, auf die der Persistenz-Manager per `FileReader` bzw. `FileWriter` zugreift. Die Dateinamen entsprechen dann PageID bzw. LSN und jede Datei enthält eine Zeile Text, in der die einzelnen Bestandteile z.B. durch Kommata getrennt sind.
- Der Puffer lässt sich z.B. mithilfe einer `Hashtable` realisieren. Neben den Nutzdaten muss natürlich auch die Zugehörigkeit von Datensätzen zu Transaktionen verwaltet sowie der Überblick über noch laufende und bereits abgeschlossene Transaktionen gewahrt werden.
- Das Anlegen von Sicherungspunkten ist nicht erforderlich.
- Die Implementierung einer graphischen Benutzeroberfläche ist weder erforderlich noch erwünscht.
- Eine Abgabe der Ergebnisse ist nicht erforderlich; der Erfolg der Aufgabenbearbeitung wird in der Präsenzübung in der KW 24 überprüft.
- Für die Entwicklung steht Ihnen eine Eclipse-Umgebung zur Verfügung, die Sie mit dem Befehl `eclipse340` starten.