

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Verteilte Transaktionen
	Bearbeitungsbeginn	KW23
	Bearbeitungsende	KW25

## Aufgaben 6: Verteilte Transaktionen

Implementieren Sie ein System zur Durchführung verteilter Transaktionen nach dem Modell von X/Open DTP (Abbildung 1). Im Rahmen einer Transaktion greift dabei eine Anwendung schreibend auf mehrere Ressourcen-Manager zu, die sich zum Abschluss der Transaktion auf einen atomaren Ausgang verständigen. Die Abstimmung unter den Ressourcen-Managern soll von einem dedizierten Koordinator geleitet und mithilfe des Two-Phase-Commit-Protokolls durchgeführt werden (Abbildung 2). Die Kommunikation über die TX- und XA-Schnittstellen kann dabei in einer gegenüber dem Original-Standard vereinfachten Form entsprechend den unten angegebenen Methodenbeschreibungen umgesetzt werden.

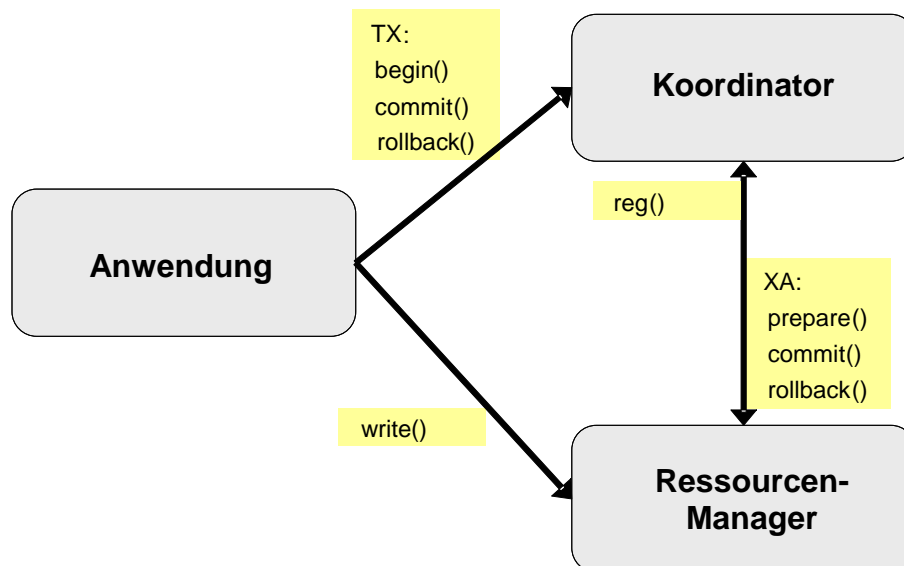



Abbildung 1: Transaktionsverwaltung nach vereinfachtem X/Open DTP

### Anwendung

Die Anwendung führt nacheinander mehrere Transaktionen durch. Im Rahmen einer Transaktion wird zunächst der Beginn der Transaktion beim Koordinator gemeldet, der eine eindeutige ID für diese Transaktion vergibt. Diese ID wird bei Zugriffen auf Ressourcen-Manager angegeben. In einer Transaktion sind mindestens zwei Ressourcen-Manager einzubinden und mit einigen Schreibzugriffen anzusprechen. Schließlich gibt die Anwendung das Kommando zum Festschreiben oder Verwerfen der Transaktion an den Koordinator (für eine erfolgreiche Bearbeitung der Aufgabe sind beide Varianten zu unterstützen), der den Ausgang der entsprechenden Operation mitteilt.

### Koordinator

Der Koordinator verwaltet zu jeder Transaktion die Liste der teilnehmenden Ressourcen-Manager. Nach Anweisung zum Festschreiben der Transaktion leitet der Koordinator ein Two-Phase-Commit unter den Teilnehmern ein. Bei erfolgreichen **prepared**-Meldungen aller Teilnehmer werden die Ergebnisse festgeschrieben; bei **aborted**-Meldung mindestens

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Verteilte Transaktionen
	Bearbeitungsbeginn	KW23
	Bearbeitungsende	KW25

eines Teilnehmers oder bei einem expliziten Abbruch durch die Anwendung werden die Ergebnisse verworfen (siehe auch Abb. 2). Die Schnittstelle des Koordinators soll folgende Funktionalität bieten:

**int begin()**

Beginnt eine neue Transaktion.

Rückgabewert: Transaktions-ID der neuen Transaktion.

**boolean commit(int taid)**

Leitet das Festschreiben der angegebenen laufenden Transaktion ein.

Rückgabewert: Meldung über Erfolg oder Misserfolg des Abschlusses.

**boolean rollback(int taid)**

Leitet den Abbruch der angegebenen laufenden Transaktion ein.

Rückgabewert: Meldung über Erfolg oder Misserfolg des Abbruchs.

**boolean reg(ResourceManager rm, int taid)**

Registriert den angegebenen Ressourcen-Manager als Teilnehmer der angegebenen Transaktion.

Rückgabewert: Meldung über Erfolg oder Misserfolg der Registrierung.

## Ressourcen-Manager

Ressourcen-Manager erlauben schreibende Zugriffe auf von ihnen gekapselte Datenspeicher. Beim ersten Zugriff durch die Anwendung mit einer neuen Transaktions-ID registriert sich ein Ressourcen-Manager für diese Transaktion beim Koordinator. Schreibzugriffe werden zunächst im (flüchtigen) Arbeitsspeicher gesammelt. Bei Eingang einer **prepare**-Anweisung muss der Ressourcen-Manager dafür sorgen, dass die Transaktion erfolgreich abgeschlossen werden kann. Bei anschließendem Eingang einer **commit**-Anweisung wird entsprechend die neue Version als endgültig übernommen und die alte Version verworfen; bei Eingang einer **rollback**-Anweisung ist die neue Version wieder zu entfernen. Trifft eine **rollback**-Anweisung ein, wenn der Ressourcen-Manager noch im *Active*-Zustand (siehe Abb. 2) ist, werden einfach die Daten im flüchtigen Speicher verworfen.

Ressourcen-Manager sollen im Rahmen der Aufgabe auf eine **prepare**-Anweisung gelegentlich mit **aborted** antworten, um das Funktionieren des Protokolls auch im Fehlerfall zu zeigen.

Die Schnittstelle des Ressourcen-Managers soll folgende Funktionalität bieten:

**boolean write(int taid, int pid, String data)**

Schreibt die angegebenen Daten (Seite, Nutzdaten) im Rahmen der angegebenen Transaktion.

Rückgabewert: Meldung über Erfolg oder Misserfolg des Schreibzugriffs.

**boolean prepare(int taid)**

Weist zur Vorbereitung des endgültigen Festschreibens der angegebenen Transaktion an.

Rückgabewert: Meldung über Erfolg oder Misserfolg der Vorbereitung, entsprechend **false** für *aborted* und **true** für *prepared*.

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Verteilte Transaktionen
	Bearbeitungsbeginn	KW23
	Bearbeitungsende	KW25

**boolean commit(int taid)**

Weist zum Festschreiben der angegebenen Transaktion an.

Rückgabewert: Meldung über Erfolg oder Misserfolg des Festschreibens, wobei ein Misserfolg gemäß dem 2PC-Protokoll in dieser Phase nicht zulässig ist (**true** für *ack*).

**boolean rollback(int taid)**

Weist zum Abbruch der angegebenen Transaktion an.

Rückgabewert: Meldung über Erfolg oder Misserfolg des Abbruchs, wobei ein Misserfolg gemäß dem 2PC-Protokoll in dieser Phase nicht zulässig ist (**true** für *ack*).

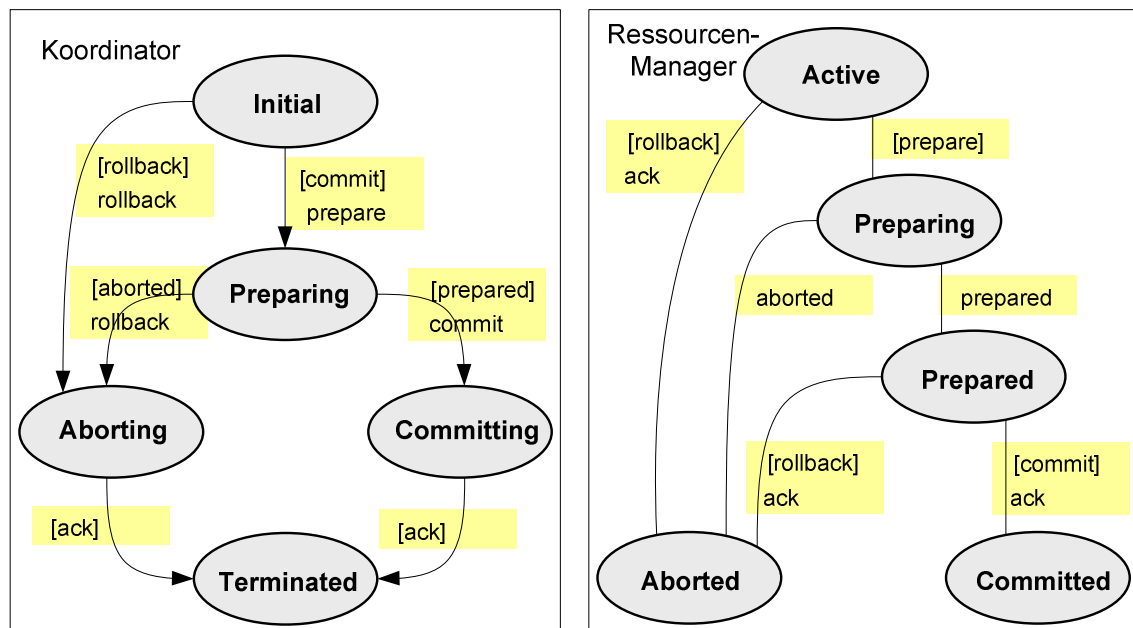


Abbildung 2: Zustände und Nachrichten beim Two-Phase-Commit-Protokoll


## Realisierungsalternativen

Für die Bearbeitung der Aufgabe sind zwei Herangehensweisen möglich:

1. Die Realisierung des Persistenz-Managers aus Aufgabe 5 kann entsprechend den Anforderungen angepasst werden. Hierzu ist u.A. eine Aufteilung der vorhandenen Funktionalität in Koordinator und Ressourcen-Manager erforderlich.

Der Empfang der **prepare**-Nachricht erfordert lediglich das Ausschreiben eines entsprechenden Log-Satzes (sowie von ggf. noch nicht persistenten Log-Sätzen der entsprechenden Transaktion). Beim Empfang einer **rollback**-Nachricht wird ebenfalls ein entsprechender Logsatz geschrieben. Ein Rücksetzen von Änderungen in der Datenbank ist auf Grund der No-Steal-Strategie nicht erforderlich.

Die Wiederanlaufprozedur des Ressourcen-Managers muss derart erweitert werden, dass für alle Transaktionen mit unklarem Ausgang (d.h. es wurde ein prepared-Logsatz gefunden, nicht jedoch ein commit-Logsatz oder rollback-Logsatz) eine Anfrage an den Koordinator über den Ausgang der Transaktion gestellt wird, um das globale Commit-Ergebnis zu ermitteln.

	Lehrveranstaltung	Datenbanken und Informationssysteme
	Aufgabe	Verteilte Transaktionen
	Bearbeitungsbeginn	KW23
	Bearbeitungsende	KW25

Neben dem Ressourcen-Manager muss auch der Koordinator eine Logdatei führen. In dieser werden die Entscheidungen (Commit/Rollback) über den globalen Ausgang der Transaktionen protokolliert (d.h. nach dem Eintreffen aller **prepared**-Nachrichten). Außerdem werden die Informationen über die vollständige Beendigung einer Transaktion (d.h. nach dem Eintreffen aller **ack**-Nachrichten). Bei einem Wiederanlauf müssen diese Log-Informationen verwendet werden, um potentielle Anfragen der Ressourcen-Manager nach dem globalen Ausgang von Transaktionen zu beantworten.

- Der Ressourcen-Manager kann neu implementiert werden. In diesem Fall kann von einem vereinfachten Szenario ausgegangen werden, bei dem Systemfehler, Datenverluste, Abstürze etc. nicht berücksichtigt werden. Logging und Recovery sind daher sowohl in den Ressourcen-Managern als auch im Koordinator nicht notwendig. Festzuschreibende Daten sind allerdings wie bei Aufgabe 5 im Dateisystem zu persistieren.

Bei Eingang einer **prepare**-Anweisung im Ressourcen-Manager werden die Daten ausgeschrieben, dabei aber vorhandene ältere Versionen nicht überschrieben! Die parallele Existenz zweier Versionen während der *Prepared*-Phase muss geeignet unterstützt werden; die neue Version ist in dieser Phase noch als „vorläufig“ zu betrachten. Bei anschließendem Eingang einer **commit**-Anweisung wird entsprechend die neue Version als endgültig übernommen und die alte Version verworfen; bei Eingang einer **rollback**-Anweisung ist die neue Version wieder zu entfernen.

## Hinweise

- Die Implementierung und Ausführung einer einzigen Anwendung ist ausreichend. Eine Sperrverwaltung im Ressourcen-Manager ist daher nicht erforderlich.
- Die Implementierung von Timeouts im 2PC-Protokoll ist nicht erforderlich.
- Asynchrone Kommunikation zwischen den einzelnen Komponenten wird nicht gefordert. Koordinator, Ressourcen-Manager und Anwendung lassen sich als Bestandteile einer Java-Anwendung implementieren. Für den Koordinator bietet sich das Singleton-Pattern an; die Exemplare der Ressourcen-Manager müssen unterscheidbar sein und geeignet verwaltet werden.
- Bedenken Sie, dass sich auch bei synchroner Kommunikation und ohne Multi-Threading einzelne Komponenten in verschiedenen Zuständen befinden können und damit Methodenaufrufe (entsprechend eingehender Nachrichten) erfolgen können, die im aktuellen Zustand der betroffenen Komponente „unerwartet“ sind. Die Komponenten müssen sich in jedem Fall robust und „sinnvoll“ verhalten. Die Zustandstabellen in den X/Open-Spezifikationen der TX- und XA-Schnittstelle können hier als Anhaltspunkte dienen (siehe Abb. 2).
- Auf der Homepage zum Modul sind die Original-Spezifikationen bereitgestellt.
- Versehen Sie Ihre Klassen mit geeigneten Ausgaben z.B. auf der Konsole, so dass sich die Abläufe im System nachvollziehen lassen.
- Eine Abgabe der Ergebnisse ist nicht erforderlich; der Erfolg der Aufgabenbearbeitung wird in den Präsenzübungen in der KW25 überprüft. Stellen Sie sicher, dass Ihre Bearbeitung der Aufgabe in KW25 auf jeden Fall abgeschlossen und flüssig präsentierbar ist.