

Prof. Dr.-Ing. Dr. h. c. Theo Härder
Universität Kaiserslautern
Fachbereich Informatik
haerder@informatik.uni-kl.de
<http://www.haerder.de/>

Vorlesung

Transaktionssysteme

SS 2009

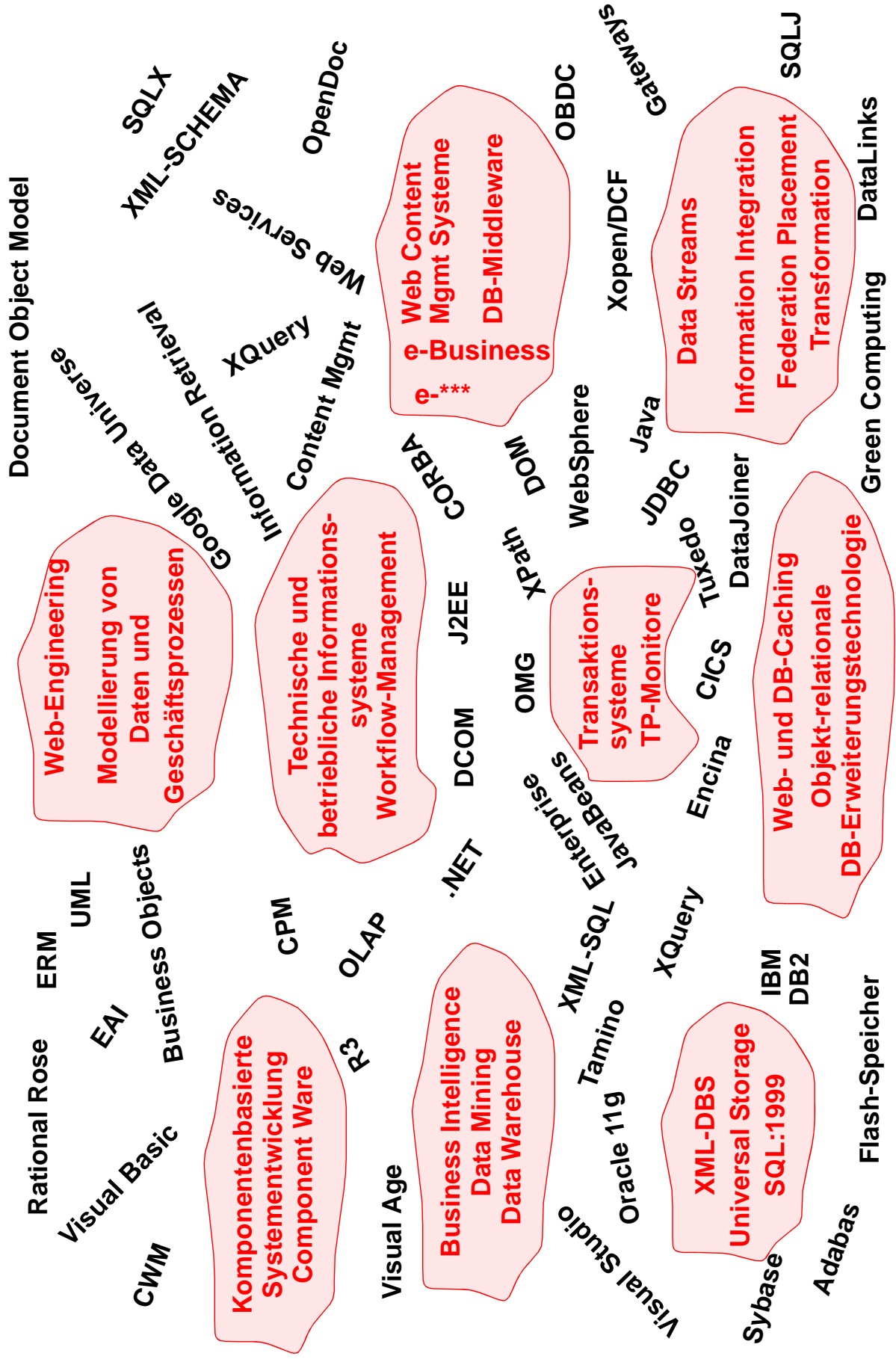
Theo Härder

Vorlesung:

Do., 10.00 - 11.30 Uhr, 36-265

Beginn: Do., 23. 04. 2009

Weltbild des Lehrgebietes Informationssysteme



Ziele

- **Vermittlung von vertieften Kenntnissen zu allen Entwicklungs- und Einsatzaspekten von Transaktionssystemen:**
 - Aufbau von Transaktions- und Client/Server-Systemen
 - Verständnis von Transaktionsmodellen und ihren Variationen sowie ihre
 - Nutzung in heterogenen Systemen, bei Workflows und in Web-Anwendungen
 - Bewertung der zugehörigen Realisierungskonzepte
- Die **Inhalte der Vorlesung** werden von **einigen Anforderungen**, die heute an einen **"Softworker"** gestellt werden, **bestimmt**. Das folgende **Kenntnisprofil** kann als **Ideal für einen modernen Softworker** dienen. Er
 - kennt die XY-Maschine
 - kann mindestens mit zwei Betriebssystemen umgehen und hat Grundkenntnisse bei Netzwerken
 - beherrscht eine Assembler- und eine höhere Programmiersprache
 - ist vertraut mit einem Datenmodell und der zugehörigen DB-Sprache (z. B. SQL:1999 als relationaler Standard)
 - *kann Datenbanken strukturieren und verwalten*
 - *kann einen TP-Monitor benutzen*
 - *kann Transaktionssystem-Anwendungen entwickeln*
 - beherrscht eine Entwurfsmethodik und kann die entsprechenden Entwurfswerkzeuge bedienen
 - beherrscht eine Testmethodik und kann die entsprechenden Testwerkzeuge anwenden
 - ist vertraut mit einer formalen Spezifikationssprache
 - kann die Anforderungen mit den Anwendern absprechen und
 - ist in der Lage, ein DV-Projekt zu leiten. (erweitert nach H. Sneed).

Übersicht (vorl.)

0. Grundlagen (Wiederholung)

- Transaktionskonzept (DBAW)
- Wichtigste Eigenschaften/Realisierungskonzepte von ACID

1. Architekturen von TA-Systemen

- Ziel: Ableitung von Schichtenmodellen
- Schichtenmodell eines zentralisierten TA-Systems
- Verteilte Transaktionen unter Kontrolle eines TP-Monitors
- Kontrolle verteilter Transaktionen durch VDBS oder durch Middleware
- Web als TA-System
- Standardisierung der Leistungsbewertung

2. Client/Server-Systeme

- Anwendungsentwicklung – allgemeine Probleme
- Sicht des Endbenutzers
- Entwurfsaufgaben
- Client/Server-Modell, zwei- und dreistufige C/S-Architektur
- Effiziente Realisierung von Servern
- TP-Monitor – Prinzipien und Einsatz
- Aufbau des DB-Servers (Drei-Schichten-Modell)
- TA-Verarbeitung in offenen Systemen
- Optimierte Protokolle für 2PC
- Zusammenfassung – Vielfalt an Bezeichnungen

Übersicht (2)

3. Transaktionsmodelle

- Kontrollbereiche
- Beschreibung von Transaktionsmodellen
- Gekettete Transaktionen
- Prinzipien der Schachtelung — offen, geschlossen
- Geschlossen geschachtelte Transaktionen — Eigenschaften
- Vertiefung – Sperrprotokolle
- Offen geschachtelte Transaktionen — Eigenschaften
- Mehrebenen-Transaktionen, Sagas
- Komplexe Transaktionsmodelle – Erkennung von Abhängigkeiten

4. TA-Modelle für heterogene Systeme, Workflows und Web

- Heterogene TA-Systeme
 - Autonomie vs. Transparenz
 - Probleme: Serialisierbarkeit, Atomarität, Deadlocks
- TA-Verwaltung in heterogenen Systemen
 - Korrektheitsbedingungen bei Mehrebenen-TA
 - Globale Serialisierbarkeit und globale Atomarität
- Daten- und Funktionsintegration
- Transaktionale Workflows und ConTracts
- Geschäftstransaktionen und Web Services
 - Elektronischer Handel, Geschäftsmodelle
 - Atomarität bei Geschäftstransakti

LITERATURLISTE

Aussagen zu wesentlichen Inhalten der Vorlesung in:

Gray, J., Reuter, A.:

Transaction Processing—Concepts and Techniques, Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1998 (5th printing).

Bernstein, P.A., Newcomer, E.:

Principles of Transaction Processing, Morgan Kaufmann, San Mateo, 1997.

Weikum, G., Vossen, G.:

Transactional Information Systems, Morgan Kaufmann, 2001.

An bestimmten Stellen zusätzlich hilfreich:

Dittrich, K.R., Geppert, A. (Eds):

Component Database Systems, Morgan Kaufmann, 2001.

Härder, T., Rahm, E.:

Datenbanksysteme — Konzepte und Techniken der Implementierung, 2. Auflage, Springer-Verlag, 2001.

Meyer-Wegener, K.:

Transaktionssysteme, B. G. Teubner, Stuttgart, 1988.

Rahm, E.:

Hochleistungs-Transaktionssysteme, Vieweg-Verlag, 1993.

Orfali, R., Harkey, D., Edwards, J.:

Client/Server Survival Guide, Third Edition, Wiley Computer Publishing Group (John Wiley & Sons, Inc.), New York, 1999.

Stallings, W.:

Betriebssysteme — Prinzipien und Umsetzung, 4. Auflage, Pearson Studium, 2003.

LITERATURLISTE (2)

ÜBERSICHTSAUFSÄTZE:

Bernstein, P.A.:

Transaction Processing Monitors. Communications of the ACM 33 (11), 1990, 75-86.

Härder, T., Meyer-Wegener, K.:

Transaktionssysteme und TP-Monitore. Eine Systematik ihrer Aufgabenstellung und Implementierung, Informatik — Forschung und Entwicklung (1986), 1: 3-25.

Härder, T., Meyer-Wegener, K.:

Die Zusammenarbeit von TP-Monitoren und Datenbanksystemen in DB/DC-Systemen. Existierende Systeme und zukünftige Entwicklungen, Informatik — Forschung und Entwicklung (1986), 1: 101-122.

Härder, T., Meyer-Wegener, K.:

Transaktionssysteme in Workstation-Server-Umgebungen, Informatik — Forschung und Entwicklung (1990), 5: 127-143.

Rahm, E.:

Der Database-Sharing-Ansatz zur Realisierung von Hochleistungs-Transaktionssystemen, Informatik-Spektrum 12 (2), 65-81 (1989).

ZEITSCHRIFTEN:

ACM TODS Transactions on Database Systems, ACM-Publikation (vierteljährlich.)

THE VLDB Journal VLDB Foundation (vierteljährlich)

Information Systems Pergamon Press (6-mal jährlich)

ACM Computing Surveys ACM-Publikation (vierteljährlich)

TAGUNGSBÄNDE:

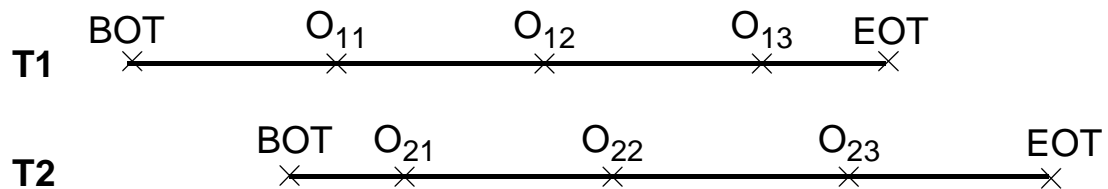
SIGMOD Proceedings, jährliche Konferenz der ACM Special Interest Group on Management of Data

VLDB Proceedings, jährl. Konferenz „Very Large Data Bases“

ICDE Proceedings, jährliche Konferenz „Data Engineering“

Transaktionen (Grundlagen/Wiederholung)

- **Ablaufkontrollstruktur: Transaktion**



- **Eigenschaften**

- **Atomicity (Atomarität)**

- TA ist kleinste, nicht mehr weiter zerlegbare Einheit
- „Alles-oder-nichts“-Prinzip

- **Consistency**

- TA hinterlässt einen konsistenten DB-Zustand
- Zwischenzustände dürfen inkonsistent sein
- Endzustand muß Integritätsbedingungen erfüllen

- **Isolation**

- Nebenläufige TA dürfen sich nicht gegenseitig beeinflussen

- **Durability (Dauerhaftigkeit)**

- Wirkung erfolgreich abgeschlossener TA bleibt dauerhaft in der DB
- TA-Verwaltung muß sicherstellen, dass dies auch nach einem Systemfehler gewährleistet ist
- Wirkung einer erfolgreich abgeschlossenen TA kann nur durch eine sog. kompensierende TA aufgehoben werden

Transaktionen (2)

- **DB-bezogene Definition der Transaktion:**

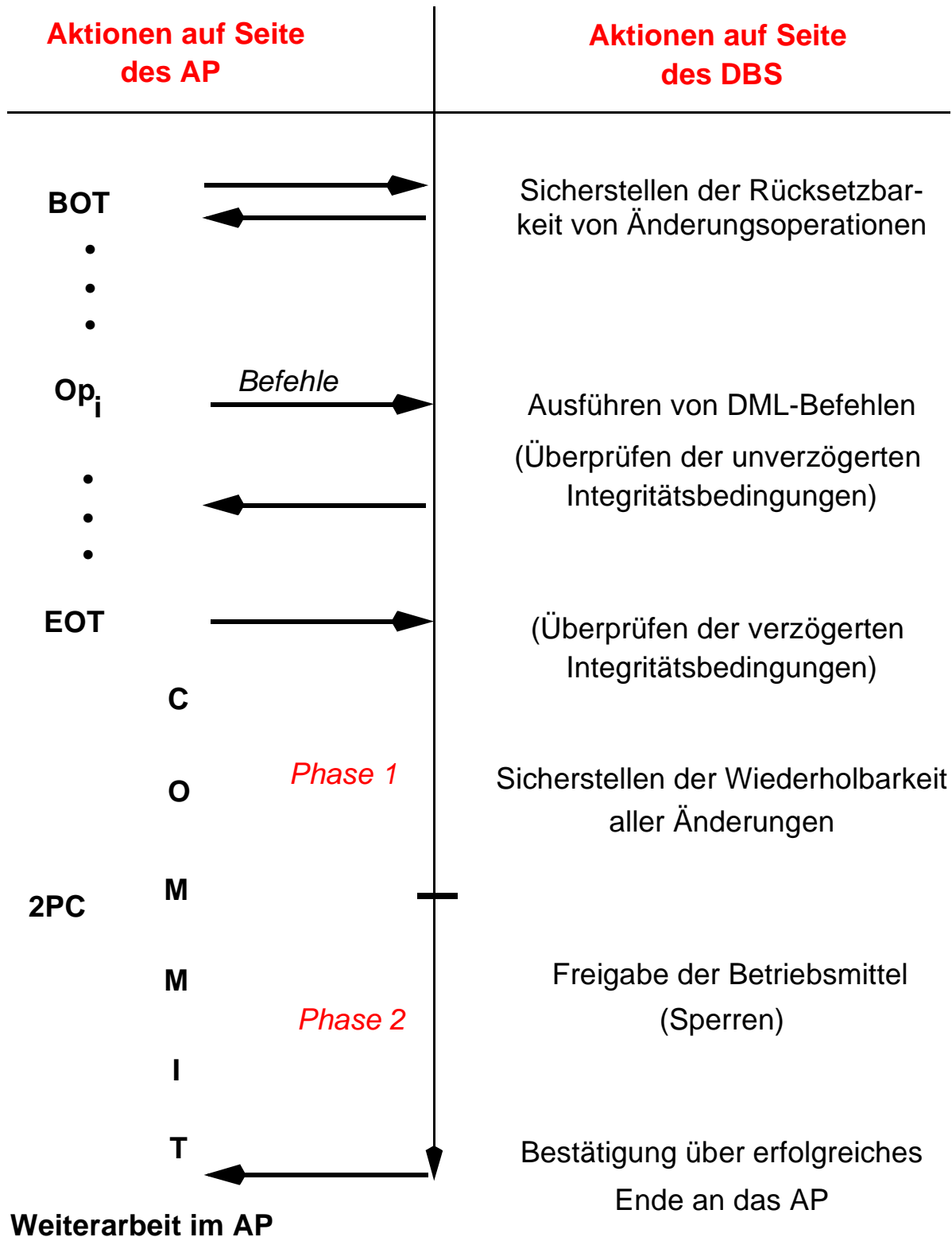
Eine TA ist eine ununterbrechbare Folge von DML-Befehlen, welche die Datenbank von einem logisch konsistenten Zustand in einen logisch konsistenten Zustand überführt.

- **Transaktionsverwaltung**

- koordiniert alle DBS-seitigen Maßnahmen, um ACID zu garantieren
- besitzt zwei wesentliche Komponenten
 - Synchronisation
 - Logging und Recovery
- ➔ Sie bieten **Transparenz der Nebenläufigkeit** (concurrency transparency) **und Fehlertransparenz** (failure transparency) für den AW-Programmierer und Endbenutzer!
- kann zentralisiert oder verteilt (z.B. bei VDBS) realisiert sein
- soll Transaktionsschutz für heterogene Komponenten bieten

Transaktionen (3)

- Schnittstelle zwischen AP und DBS



Transaktionen (4)

- **Integritätskontrolle**

- modellinhärente Integritätsbedingungen (relationale Invarianten)
- anwendungsspezifische Integritätsbedingungen
- Trigger
- ECA-Regeln

- **semantische Integritätsbedingungen**

- **Reichweite**

- Attribut
- Tupel
- Relation
- mehrere Relationen

- **Zeitpunkt der Überprüfung**

- immediate
- deferred

- **Art der Überprüfbarkeit**

- Zustand
- Übergang

- **Anlass für Überprüfung**

- Datenänderung
- Zeitpunkt

Transaktionen (5)

- **Synchronisation**

- Ziel: Vermeidung von Anomalien
 - Abhängigkeit von nicht freigegebenen Änderungen (*dirty read*)
 - Verlorengegangene Änderung (*lost update*)
 - Inkonsistente Analyse (*non-repeatable read*)
 - Phantom-Problem
- **Formales Korrektheitskriterium: *Serialisierbarkeit*:**

Die parallele Ausführung einer Menge von TA ist **serialisierbar**, wenn es eine serielle Ausführung derselben TA-Menge gibt, die den **gleichen DB-Zustand** und die **gleichen Ausgabewerte** wie die ursprüngliche Ausführung erzielt.

- **Verfahren**

- Sperrverfahren
- Optimistische Verfahren
- Zeitmarkenverfahren
- Mehrversionenverfahren
- Spezialverfahren

Transaktionen (6)

- Synchronisation (Forts.)

- Verfahren (Forts.)

- Beispiel: Sperrverfahren

- **2PL-Regeln:**

1. Vor jedem Objektzugriff muß Sperre mit ausreichendem Modus angefordert werden
2. Gesetzte Sperren anderer TA sind zu beachten
3. Eine TA darf nicht mehrere Sperren für ein Objekt anfordern
4. **Zweiphasigkeit:** Wachstumsphase vor Schrumpfungsphase (Beachte Varianten: *strikt* und *preclaiming*)
5. Spätestens bei Commit sind alle Sperren freizugeben

- Beispiel: Hierarchische Sperren

	IR	IX	R	RIX	U	X
IR	+	+	+	+	-	-
IX	+	+	-	-	-	-
R	+	-	+	-	-	-
RIX	+	-	-	-	-	-
U	-	-	+	-	-	-
X	-	-	-	-	-	-

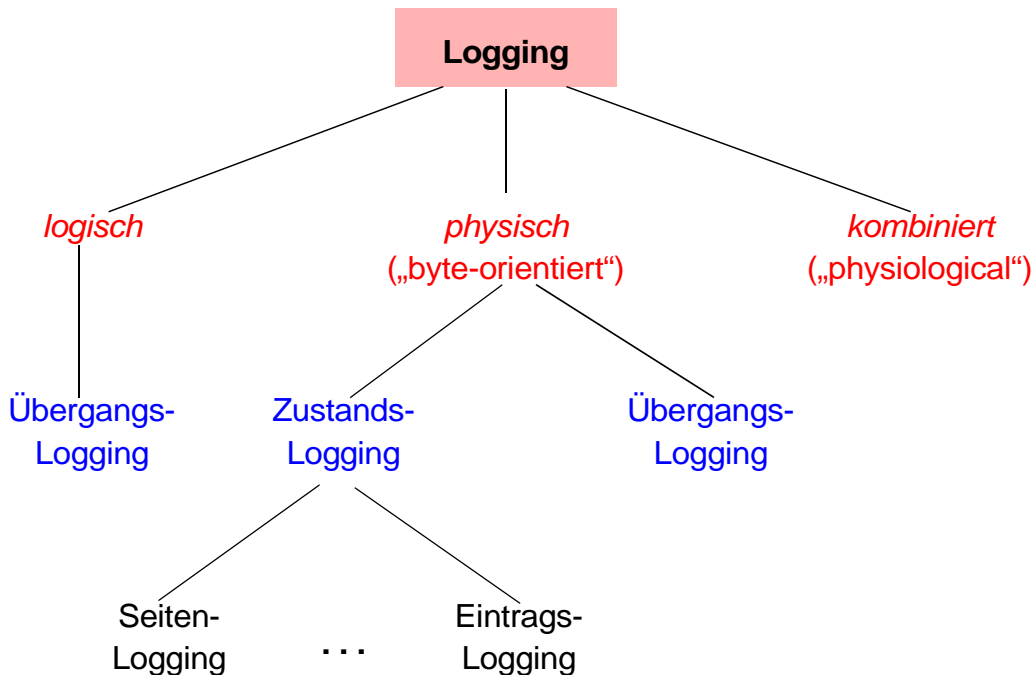
Transaktionen (7)

- **Logging und Recovery**
 - **Recovery-Arten**
 - **Transaktions-Recovery**
 - vollständiges Zurücksetzen (TA-UNDO)
 - partielles Zurücksetzen auf Rücksetzpunkt (Savepoint)
 - **Crash-Recovery nach Systemfehler**
 - (partielles) REDO für erfolgreiche Transaktionen (Wiederholung verlorengangener Änderungen)
 - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen der Änderungen aus der permanenten DB)
 - **Medien-Recovery nach Gerätefehler**
 - Spiegelplatten
 - vollständiges Wiederholen (REDO) aller Änderungen auf einer Archivkopie
 - **Katastrophen-Recovery**
 - Nutzung einer aktuellen DB-Kopie in einem “entfernten” System oder
 - stark verzögerte Fortsetzung der DB-Verarbeitung mit repariertem/neuem System auf der Basis gesicherter Archivkopien (Datenverlust!)

Transaktionen (8)

- **Logging und Recovery (Forts.)**

- **Logging-Verfahren**



- Abhängigkeiten zu anderen Systemkomponenten

- **Einbringstrategie**

- direkt (NON-ATOMIC, Update-in-Place)
- verzögert (ATOMIC, Bsp.: Schattenspeicherkonzept)

- **DB-Pufferverwaltung**

- Verdrängen ‚schmutziger‘ Seiten (STEAL vs. NOSTEAL)
- Ausschreibstrategie für geänderte Seiten (FORCE vs. NOFORCE)

- **Empfehlenswert:** NON-ATOMIC / STEAL / NOFORCE

- **Sperrverwaltung**

- Log-Granulat kleiner gleich Sperrgranulat!

Transaktionsverwaltung

- **Einsatz kooperierender Ressourcen-Manager (RM)**

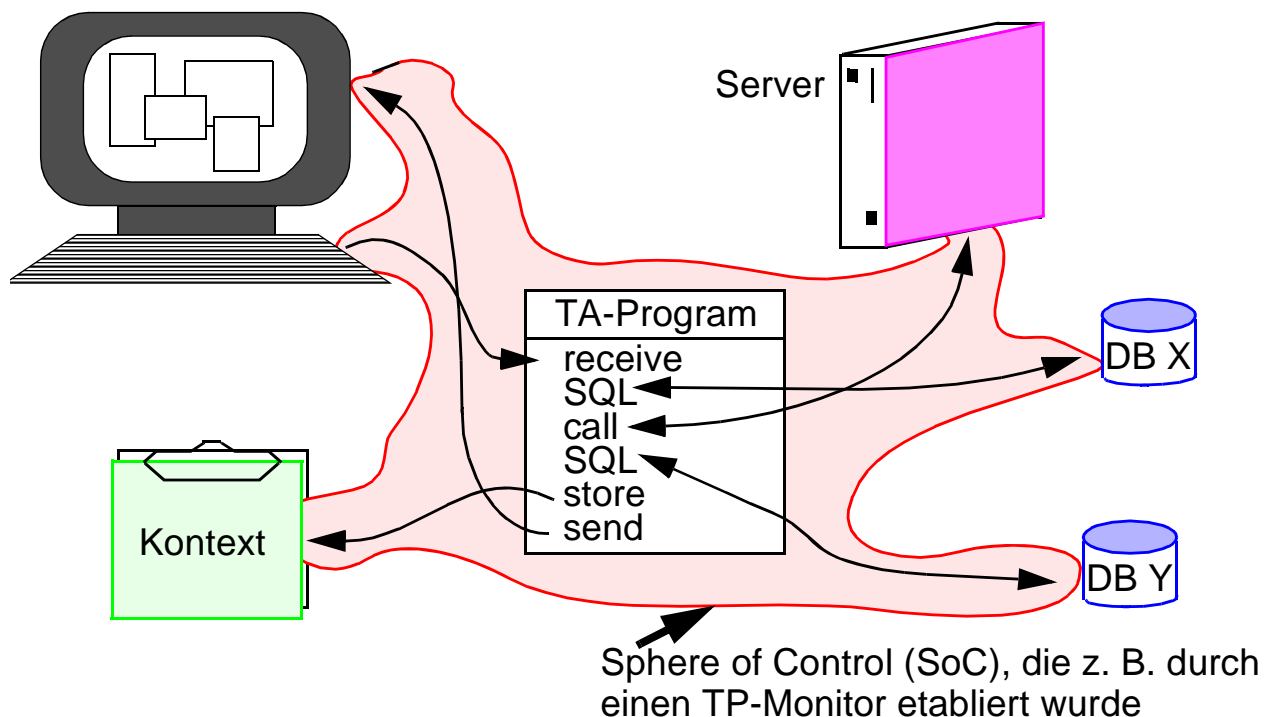
- RM sind Systemkomponenten, die **Transaktionsschutz für ihre gemeinsam nutzbaren Betriebsmittel (BM)** bieten
- RM gestatten die externe Koordination von BM-Aktualisierungen durch spezielle Commit-Protokolle

- ➔ **Gewährleistung von ACID für DB-Daten und auch für andere BM**

(persistente Warteschlangen, Nachrichten, Objekte von persistenten Programmiersprachen)

- **Ziel:**

TA-orientierte Verarbeitung in heterogenen Systemen



- ➔ **Die gesamte verteilte Verarbeitung in einer SoC ist eine ACID-TA**

- alle Komponenten werden durch die TA-Dienste integriert
- für die Kooperation ist eine Grundmenge von Protokollen erforderlich