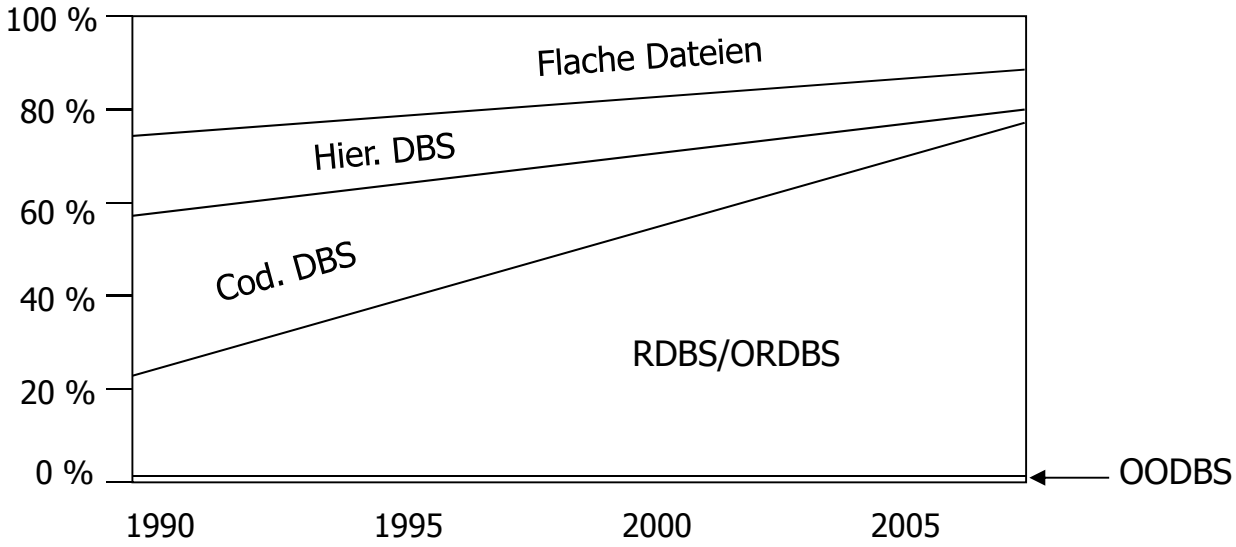
Kapitel 1.
Anforderungen an DBMS und Transaktionskonzept

Inhalt:
Wiederholung,
Anforderungen an DBS,
Transaktionsbegriff

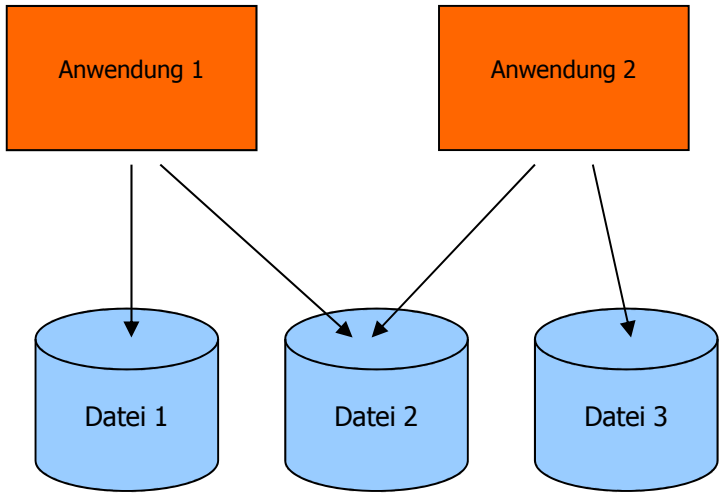
Dateien und DBS (1)

- **Es gibt verschiedene Datenmodelle und sie realisierende DBS**
  - Relational und objektrelational (RDBS/ORDBS)
  - Hierarchisch (DBS nach dem Hierarchiemodell)
  - Netzwerkartig (DBS nach dem CODASYL-Standard)
  - Objektorientiert (OODBS)
- **Aufstellung berücksichtigt nur strukturierte Daten.**
  - 85 % der weltweit verfügbaren Daten sind semi- oder unstrukturiert (Internet, wiss. Aufzeichnungen und Experiment, usw.)
  - SQL-XML-DBS, XML-SQL-DBS, native XML-DBS, Information-Retrieval



Nutzung von Dateisystemen

- **Dateisystem**
  - **Permanente Datenhaltung innerhalb von BS-Dateien**
  - **Betriebssystem/Dateisystem bietet Funktionen für**
    - Erzeugen / Löschen von Dateien
    - Zugriffsmöglichkeiten auf Blöcke/Sätze der Datei
    - einfache Operationen zum Lesen/Ändern/Einfügen/Löschen von Sätzen (dynamisches Wachstum)
  - **Probleme/Nachteile**
    - Datenredundanz und Inkonsistenz
    - Inflexibilität
    - Mehrbenutzerbetrieb, Fehlerfall
    - Integritätssicherung
    - Missbrauch der Daten
    - Verantwortlichkeit



Anforderungen an ein DBS (1)

1. Kontrolle über die operationalen Daten

- **Alle Daten können/müssen gemeinsam benutzt werden**
  - keine verstreuten privaten Dateien
  - Querauswertungen aufgrund inhaltlicher Zusammenhänge
  - symmetrische Organisationsformen
    (keine Bevorzugung einer Verarbeitungs- und Auswertungsrichtung)
  - Entwicklung neuer Anwendungen auf der existierenden DB
  - Erweiterung/Anpassung der DB (Änderung des Informationsbedarfs)
- **Redundanzfreiheit (aus Sicht der Anwendung)**
  - keine wiederholte Speicherung in unterschiedlicher Form für verschiedene Anwendungen
  - Vermeidung von Inkonsistenzen
  - zeitgerechter Änderungsdienst, keine unterschiedlichen Änderungsstände
- **Datenbankadministrator (DBA):**
  zentrale Verantwortung für die operationalen Daten

Anforderungen an ein DBS (2)

2. Leichte Handhabbarkeit der Daten

- **Einfache Datenmodelle**
  - Beschreibung der logischen Aspekte der Daten
  - Benutzung der Daten ohne Bezug auf systemtechnische Realisierung
- **Logische Sicht der Anwendung**
  - zugeschnitten auf ihren Bedarf
  - lokale Sicht auf die DB
- **Leicht erlernbare Sprachen**
  - deskriptive Problemformulierung
  - hohe Auswahlmächtigkeit
  - Unterstützung der Problemlösung des Anwenders im Dialog

Anforderungen an ein DBS (3)

2. Leichte Handhabbarkeit der Daten (Forts.)

- **Durchsetzung von Standards**
  - unterschiedliche DBS bieten einheitliche Schnittstelle
  - Portierbarkeit von Anwendungen
  - erleichterter Datenaustausch
- **Erweiterung der Benutzerklassen**
  - Systempersonal
  - Anwendungsprogrammierer
  - anspruchsvolle Laien
  - parametrische Benutzer/ gelegentliche Benutzer

Anforderungen an ein DBS (4)

2. Leichte Handhabbarkeit der Daten (Forts.)

- **Beispiel im Relationenmodell**

DB-Schema

FB

FBNR	FBNAME	DEKAN

PROF

PNR	PNAME	FBNR	FACHGEB

STUDENT

MATNR	SNAME	FBNR	STUDBEG

PRÜFUNG

PNR	MATNR	FACH	DATUM	NOTE

Anforderungen an ein DBS (5)

2. Leichte Handhabbarkeit der Daten (Forts.)

- **Beispiel im Relationenmodell (Forts.)**

FB

FBNR	FBNAME	DEKAN
FB 9	WIRTSCHAFTSWISS	4711
FB 5	INFORMATIK	2223

PROF

PNR	PNAME	FBNR	FACHGEB
1234	HÄRDER	FB 5	DATENBANKSYSTEME
5678	WEDEKIND	FB 9	INFORMATIONSSYSTEME
4711	MÜLLER	FB 9	OPERATIONS RESEARCH
6780	NEHMER	FB 5	BETRIEBSSYSTEME

STUDENT

MATNR	SNAME	FBNR	STUDBEG
123 766	COY	FB 9	1.10.95
225 332	MÜLLER	FB 5	15. 4.87
654 711	ABEL	FB 5	15.10.94
226 302	SCHULZE	FB 9	1.10.95
196 481	MAIER	FB 5	23.10.95
130 680	SCHMID	FB 9	1. 4.97

Ausprägungen

PRÜFUNG

PNR	MATNR	FACH	PDATUM	NOTE
5678	123 766	BWL	22.10.97	4
4711	123 766	OR	16. 1.98	3
1234	654 711	DV	17. 4.97	2
1234	123 766	DV	17. 4.97	4
6780	654 711	SP	19. 9.97	2
1234	196 481	DV	15.10.97	1
6780	196 481	BS	23.12.97	3

Anforderungen an ein DBS (6)

2. Leichte Handhabbarkeit der Daten (Forts.)

- **Beispiel im Relationenmodell (Forts.)**
  - **Deskriptive DB-Sprachen (wie SQL)**
    - hohes Auswahlvermögen und Mengenorientierung
    - leichte Erlernbarkeit auch für den DV-Laien
    - RM ist symmetrisches Datenmodell, d.h., es gibt keine bevorzugte Zugriffs- oder Auswertungsrichtung
  - Anfragebeispiele

I. Finde alle Studenten aus Fachbereich 5, die ihr Studium vor 2009 begonnen haben.

```
SELECT  *
FROM    STUDENT
WHERE   FBNR = 'FB5' AND STUDBEG < '01.01.09'
```

Anforderungen an ein DBS (7)

2. Leichte Handhabbarkeit der Daten (Forts.)

- **Beispiel im Relationenmodell (Forts.)**
  - Anfragebeispiele (Forts.)

II. Finde alle Studenten des Fachbereichs 5, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

```
SELECT   *
FROM     STUDENT
WHERE    FBNR = 'FB5' AND MATNR IN
         (SELECT   MATNR
         FROM      PRÜFUNG
         WHERE     FACH = 'DV' AND NOTE ≤ '2')
```

III. Finde die Durchschnittsnoten der DV-Prüfungen für alle Fachbereiche mit mehr als 1000 Studenten.

```
SELECT   S.FBNR, AVG (P.NOTE)
FROM     PRÜFUNG P, STUDENT S
WHERE    P.FACH = 'DV' AND P.MATNR = S.MATNR
GROUP BY S.FBNR
HAVING   (SELECT   COUNT(*)
          FROM     STUDENT T
          WHERE    T.FBNR = S.FBNR) > 1000
```

Anforderungen an ein DBS (8)

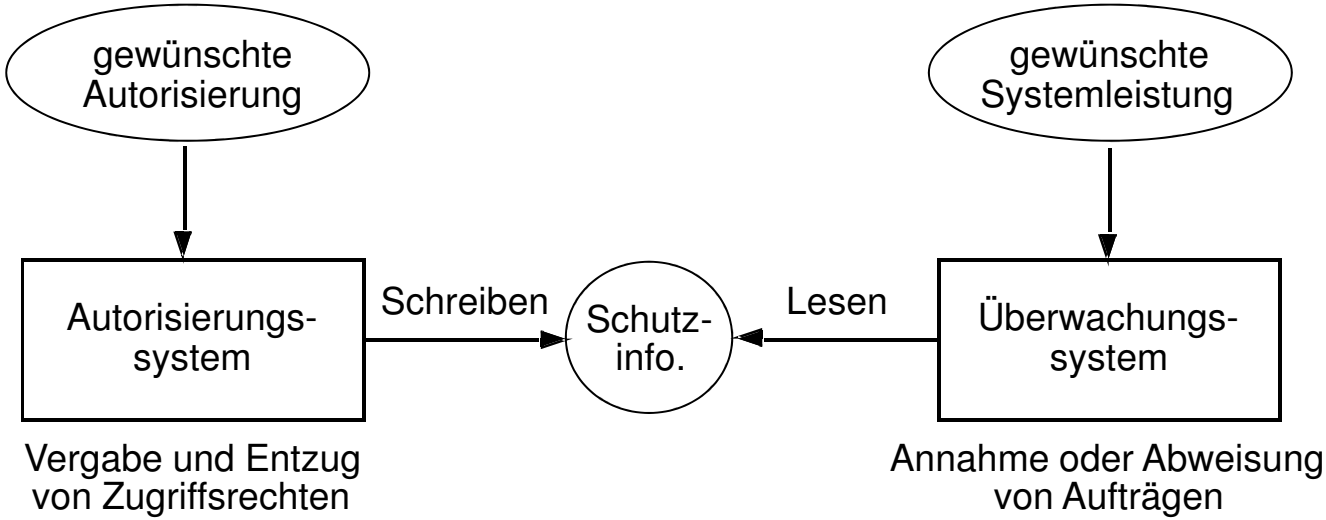
2. Leichte Handhabbarkeit der Daten (Forts.)

- **Gegensatz: Prozedurale DB-Sprachen**
  - Programmierer als Navigator
  - satzweiser Zugriff über vorhandene Zugriffspfade
  - z.B. bei hierarchischen Datenmodellen
    - unnatürliche Organisation bei komplexen Beziehungen (n:m)
    - Auswertungsrichtung ist vorgegeben

Anforderungen an ein DBS (9)

3. Kontrolle der Datenintegrität

- **Automatisierte Zugriffskontrollen (Datenschutz)**
  - separat für jedes Datenobjekt
  - unterschiedliche Rechte für verschiedene Arten des Zugriffs
  - **Idealziel**: „least privilege principle"



Anforderungen an ein DBS (10)

3. Kontrolle der Datenintegrität (Forts.)

- **Erhaltung der logischen Datenintegrität (system enforced integrity)**
  - Beschreibung der „Richtigkeit" von Daten durch Prädikate und Regeln
  - „Qualitätskontrollen" bei Änderungsoperationen
  - aktive Maßnahmen des DBS erwünscht (ECA-Regeln)

Anforderungen an ein DBS (11)

3. Kontrolle der Datenintegrität (Forts.)

- **Transaktionskonzept** (Durchsetzung der ACID-Eigenschaften)

May all your transactions commit and never leave you in doubt. (J. Gray)

  - „Schema-Konsistenz (**C**) aller DB-Daten wird bei Commit erzwungen
  - ACID impliziert Robustheit, d. h., DB enthält nur solche Zustände, die explizit durch erfolgreich abgeschlossene TA erzeugt wurden
    - **D**auerhaftigkeit (Persistenz): Effekte von abgeschlossenen TA gehen nicht verloren
    - **A**tomarität (Resistenz): Zustandsänderungen werden entweder, wie in der TA spezifiziert, vollständig durchgeführt oder überhaupt nicht
  - Im Mehrbenutzerbetrieb entsteht durch nebenläufige TA ein Konkurrenz-verhalten (concurrency) um gemeinsame Daten, d. h., TA geraten in Konflikt
    - **I**solationseigenschaft: TA-Konflikte sind zu verhindern oder aufzulösen

Anforderungen an ein DBS (12)

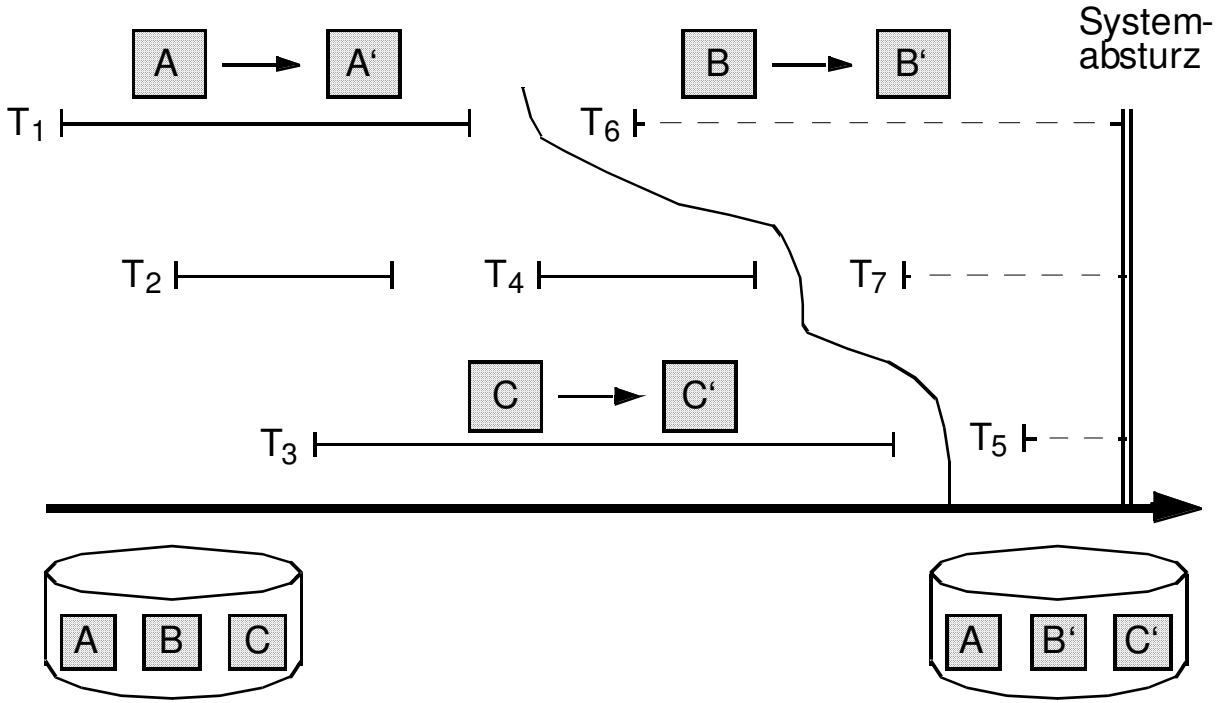
3. Kontrolle der Datenintegrität (Forts.)

- **Erhaltung der physischen Datenintegrität**
  - Periodisches Erstellen von Datenkopien
  - Führen von Änderungsprotokollen für den Fehlerfall (Logging)
  - Bereitstellen von Wiederherstellungsalgorithmen im Fehlerfall (Recovery)
    - **Garantie** nach erfolgreichem Neustart:
      jüngster transaktionskonsistenter DB-Zustand
- **Notwendigkeit des kontrollierten Mehrbenutzerbetriebs**
  - logischer Einbenutzerbetrieb für jeden von n parallelen Benutzern (Leser + Schreiber)
  - geeignete Synchronisationsmaßnahmen zur gegenseitigen Isolation
  - angepasste Synchronisationseinheiten (z. B. Sperrgranulate) mit abgestuften Zugriffsmöglichkeiten
    - **Ziel**: möglichst geringe gegenseitige Behinderung

Anforderungen an ein DBS (13)

3. Kontrolle der Datenintegrität (Forts.)

- **Zu: Erhaltung der physischen Datenintegrität**



Anforderungen an ein DBS (14)

3. Kontrolle der Datenintegrität (Forts.)

- **Zu: Erhaltung der physischen Datenintegrität (Forts.)**
  - **DBMS garantiert physische Datenintegrität**
    - bei jedem Fehler (z. B. Ausfall des Rechners, Absturz des Betriebssystems oder des DBMS, Fehlerhaftigkeit einzelner Transaktionsprogramme) wird eine „korrekte" Datenbank rekonstruiert
    - nach einem (Teil-)Absturz ist immer der jüngste transaktionskonsistente Zustand der DB zu rekonstruieren, in dem alle Änderungen von Transaktionen enthalten sind, die vor dem Zeitpunkt des Fehlers erfolgreich beendet waren (T_1 bis T_4) und sonst keine
    - automatische Wiederherstellung nach Neustart des Systems
  - **Maßnahmen beim Wiederanlauf (siehe auch Beispiel)**
    - Ermittlung der beim Absturz aktiven Transaktionen (T_5, T_6, T_7)
    - Rücksetzen (UNDO) der Änderungen der aktiven Transaktionen in der Datenbank (B' $\rightarrow$ B)
    - Wiederholen (REDO) der Änderungen von abgeschlossenen Transaktionen, die vor dem Absturz nicht in die Datenbank zurückgeschrieben waren (A $\rightarrow$ A')

Anforderungen an ein DBS (15)

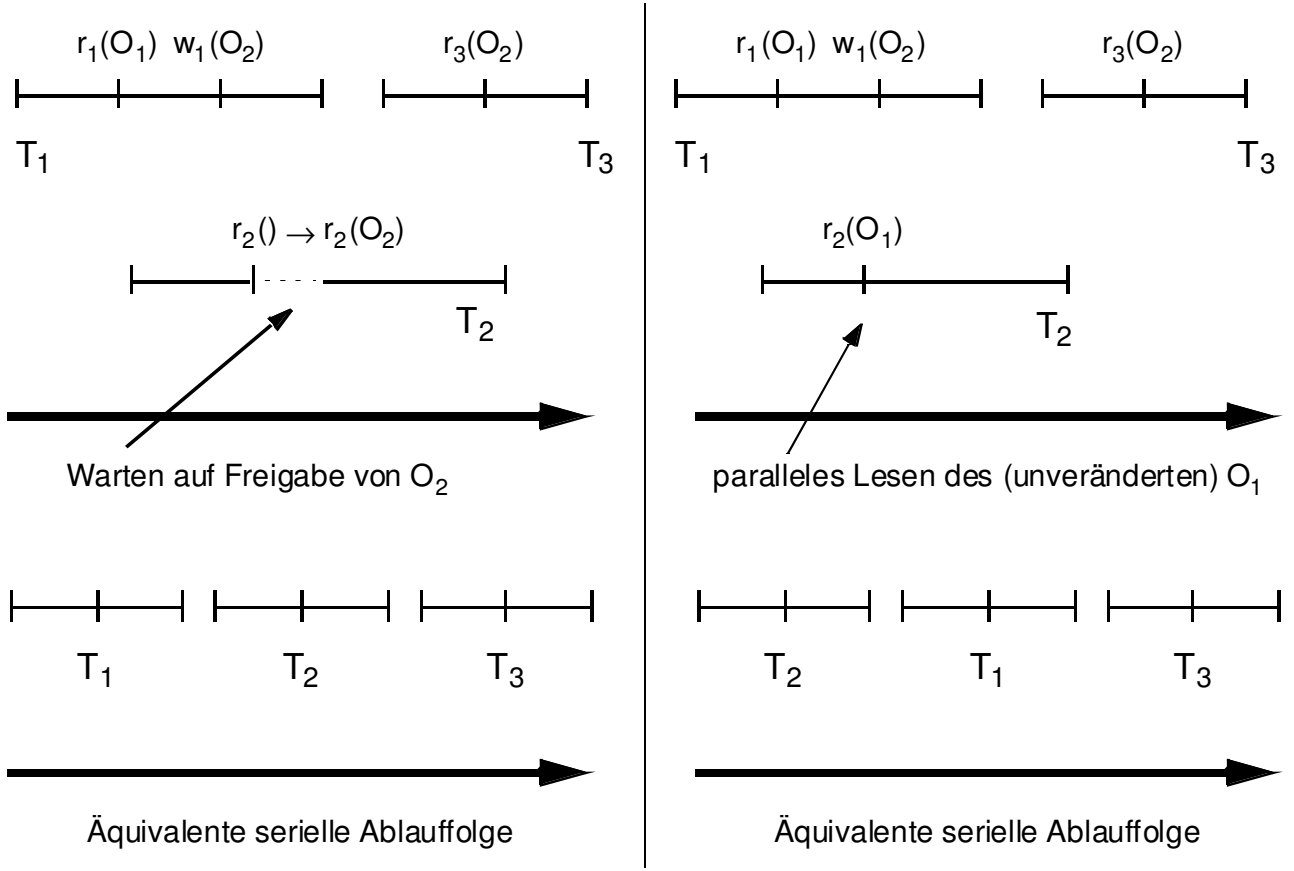
3. Kontrolle der Datenintegrität (Forts.)

- **Zu: Notwendigkeit der Kontrolle des Mehrbenutzerbetriebs**
- Beim **logischen Einbenutzerbetrieb** hat jede der parallel aktiven Transaktionen den ‚Eindruck', als liefe sie alleine ab, d. h., logisch bilden alle Transaktionen eine serielle Ablauffolge
- **Synchronisationskomponente** des DBMS umfasst alle Maßnahmen zur Sicherstellung der Ablaufintegrität (Isolation der parallelen Transaktionen)
- **Formale Definition:** Eine parallele Ablauffolge von Transaktionen ist genau dann korrekt synchronisiert, wenn es eine zu dieser Ablauffolge äquivalente (bezüglich ihrer Lese- und Schreibabhängigkeiten (r, w)) serielle Ablauffolge gibt, so dass jede Transaktion T_i in der seriellen Reihenfolge dieselben Werte liest und schreibt wie im parallelen Ablauf.

Anforderungen an ein DBS (16)

3. Kontrolle der Datenintegrität (Forts.)

- **Zu: Notwendigkeit der Kontrolle des Mehrbenutzerbetriebs (Forts.)**



Anforderungen an ein DBS (17)

4. Leistung und Skalierbarkeit

- **DBS-Implementierung gewährleistet**
  - **Effizienz** der Operatoren (möglichst geringer Ressourcenverbrauch)
  - **Verfügbarkeit** der Daten (Redundanz, Verteilung usw.)
- **Ausgleich von Leistungsanforderungen, die im Konflikt stehen**
  - globale Optimierung durch den DBA (Rolle des internen Schemas)
  - ggf. Nachteile für einzelne Anwendungen
- **Effizienz des Datenzugriffs**
  - Zugriffsoptimierung durch das DBS, nicht durch den Anwender
  - Anlegen von Zugriffspfaden durch den DBA,
    Auswahl idealerweise durch das DBS

Anforderungen an ein DBS (18)

4. Leistung und Skalierbarkeit (Forts.)

- **Leistungsbestimmung**
  - Maßzahlen für Leistung
    - Durchsatz: Anzahl abgeschlossener TA pro Zeiteinheit (meist Sekunde)
    - Antwortzeit: Zeitbedarf für die Abwicklung einer TA
  - Rolle von Benchmarks: TPC-C, TPC-H, TPC-W, TPC-R, . . .

Transaction Processing Council: www.tpc.org

- **Skalierbarkeit**
  - Software- und Hardware-Architektur sollen hinsichtlich des DBS-Leistungsverhaltens automatisch durch Hinzufügen von Ressourcen (CPU's, Speicher) skalieren
    - Scaleup: bei Wachstum der Anforderungen (DB-Größe, Transaktionslast)
    - Speedup: zur Verringerung der Antwortzeit

Anforderungen an ein DBS (19)

5. Hoher Grad an Daten-Unabhängigkeit

- **Konventionelle Anwendungsprogramme (AP) mit Dateizugriff**
  - Nutzung von Kenntnissen der Datenorganisation und Zugriffstechnik
  - gutes Leistungsverhalten, aber . . . ?
- **Datenabhängige Anwendungen sind äußerst unerwünscht**
  - Rolle des Datenmodells: Vergleiche relationales und hierarchisches Datenmodell
  - Verschiedene Anwendungen brauchen verschiedene Sichten auf dieselben Daten
  - Änderungen im Informationsbedarf sowie bei Leistungsanforderungen erzwingen Anpassungen bei Speicherungsstrukturen und Zugriffsstrategien
- deshalb: *möglichst starke Isolation der APs von den Daten*
  sonst: extremer Wartungsaufwand für die APs

Anforderungen an ein DBS (20)

5. Hoher Grad an Daten-Unabhängigkeit (Forts.)

- **Realisierung verschiedener Arten von *Daten-Unabhängigkeit*:**
  - **Minimalziel:** physische Daten-Unabhängigkeit (durch das BS/DBS)
    - **Geräte**unabhängigkeit
    - **Speicherungsstruktur**-Unabhängigkeit
  - logische Daten-Unabhängigkeit (vor allem durch das Datenmodell!)
    - **Zugriffspfad**-Unabhängigkeit
    - **Datenstruktur**-Unabhängigkeit

TA-Konzept (1)

- **Gefährdung der DB-Konsistenz**

	Korrektheit der Abbildungshierarchie	*Übereinstimmung zwischen DB und Miniwelt*
Durch das Anwendungs-programm	Mehrbenutzeranomalien **Synchronisation**	Unzulässige Änderungen **Integritätsüberwachung des DBVS** **TA-orientierte Verarbeitung**
Durch das DBVS und die Betriebsumgebung	Fehler auf den Externspeichern **Fehlertolerante Implementierung** **Archivkopien (Backup)**	Undefinierter DB-Zustand nach einem Systemausfall **TA-orientierte Fehlerbehandlung**

TA-Konzept (2)

- **Ablaufkontrollstruktur: Transaktion (TA)**
  - Eine Transaktion ist eine ununterbrechbare Folge von DML-Befehlen, die die Datenbank von einem logisch konsistenten in einen (neuen) logisch konsistenten Zustand überführt.
  - Beispiel eines TA-Programms:

```
BOT
UPDATE Konto
...
UPDATE Schalter
...
UPDATE Zweigstelle
...
INSERT INTO Ablage (...)
COMMIT
```

TA-Konzept (3)

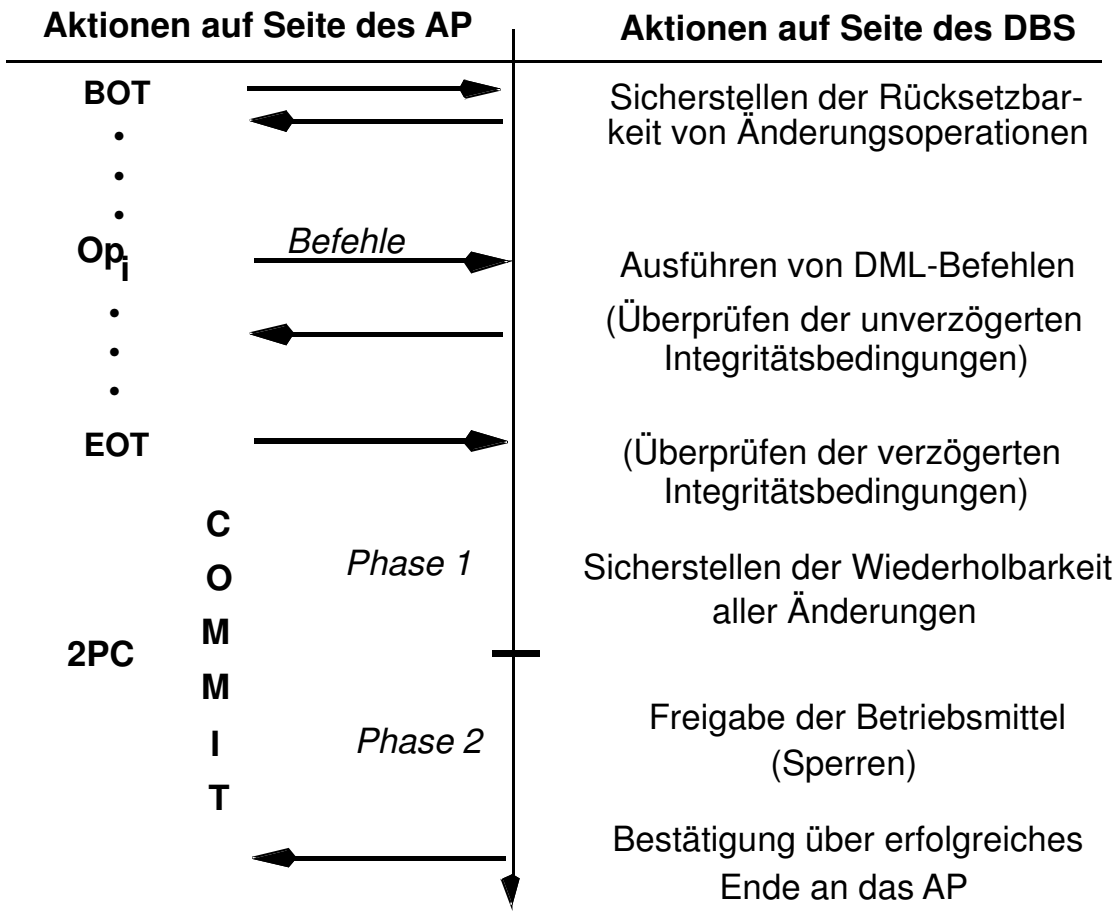
- **Ablaufkontrollstruktur: Transaktion (Forts.)**
  - ACID-Eigenschaften von Transaktionen
    - **Atomicity** (Atomarität)
      - TA ist kleinste, nicht mehr weiter zerlegbare Einheit
      - Entweder werden alle Änderungen der TA festgeschrieben oder gar keine („alles-oder-nichts"-Prinzip)
    - **Consistency**
      - TA hinterlässt einen konsistenten DB-Zustand, sonst wird sie komplett (siehe Atomarität) zurückgesetzt
      - Zwischenzustände während der TA-Bearbeitung dürfen inkonsistent sein
      - Endzustand muss alle definierten Integritätsbedingungen erfüllen

TA-Konzept (4)

- **Ablaufkontrollstruktur: Transaktion (Forts.)**
  - ACID-Eigenschaften von Transaktionen (Forts.)
    - **Isolation**
      - Nebenläufig (parallel, gleichzeitig) ausgeführte TA dürfen sich nicht gegenseitig beeinflussen
      - Parallele TA bzw. deren Effekte sind nicht sichtbar (logischer Einbenutzerbetrieb)
    - **Durability** (Dauerhaftigkeit)
      - Wirkung erfolgreich abgeschlossener TA bleibt dauerhaft in der DB
      - TA-Verwaltung muss sicherstellen, das dies auch nach einem Systemfehler (HW- oder System-SW) gewährleistet ist
      - Wirkung einer erfolgreich abgeschlossenen TA kann nur durch eine sog. kompensierende TA aufgehoben werden

TA-Konzept (5)

- **Schnittstelle zwischen Anwendungsprogramm (AP) und DBS**

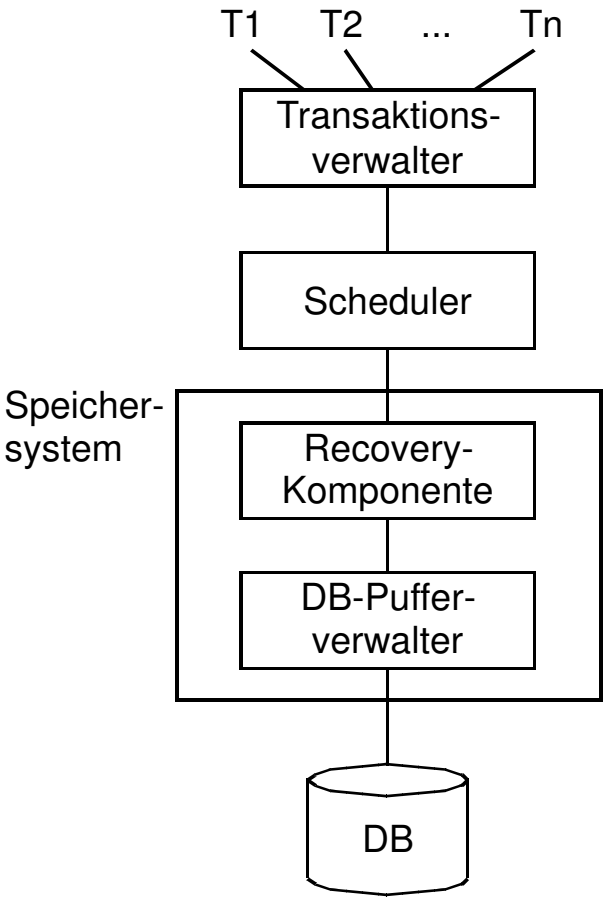


TA-Konzept (6) / TA-Verwaltung (1)

- **Wesentliche Abstraktionen (aus Sicht der DB-Anwendung) zur Gewährleistung einer ‚fehlerfreien Sicht' auf die Datenbank im logischen Einbenutzerbetrieb**
  - Alle Auswirkungen auftretender Fehler bleiben der Anwendung verborgen (failure transparency)
  - Es sind keine anwendungsseitigen Vorkehrungen zu treffen, um Effekte der Nebenläufigkeit beim DB-Zugriff auszuschließen (concurrency transparency)
- **TA-Verwaltung**
  - koordiniert alle DBS-seitigen Maßnahmen, um ACID zu garantieren
  - besitzt zwei wesentliche Komponenten
    - Synchronisation
    - Logging und Recovery
  - kann zentralisiert oder verteilt (z.B. bei VDBS) realisiert sein
  - soll Transaktionsschutz für heterogene Komponenten bieten

TA-Verwaltung (2)

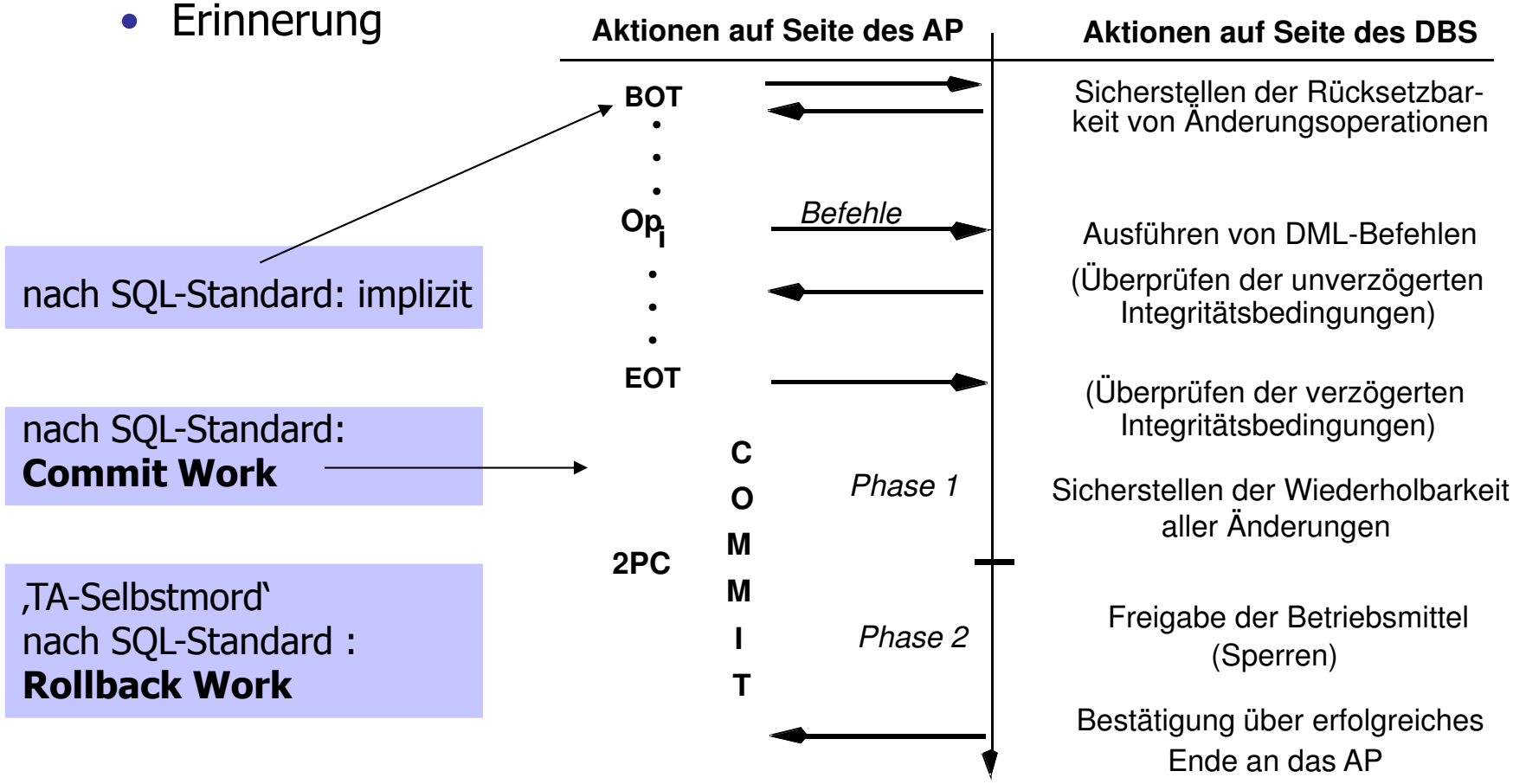
- **Abstraktes Architekturmodell (für das Read/Write-Modell auf Seitenbasis)**



- **Komponenten**
  - **Transaktionsverwalter**
    - Verteilung der DB-Operationen in VDBS und Weiterreichen an den Scheduler
    - zeitweise Deaktivierung von TA (bei Überlast)
    - Koordination der Abort- und Commit-Behandlung
  - **Scheduler** (Synchronisation)
    kontrolliert die Abwicklung der um DB-Daten konkurrierenden TA
  - **Recovery**-Komponente
    sorgt für die Rücksetzbarkeit/Wiederholbarkeit der Effekte von TA
  - DB-Pufferverwalter
    stellt DB-Seiten bereit und gewährleistet persistente Seitenänderungen

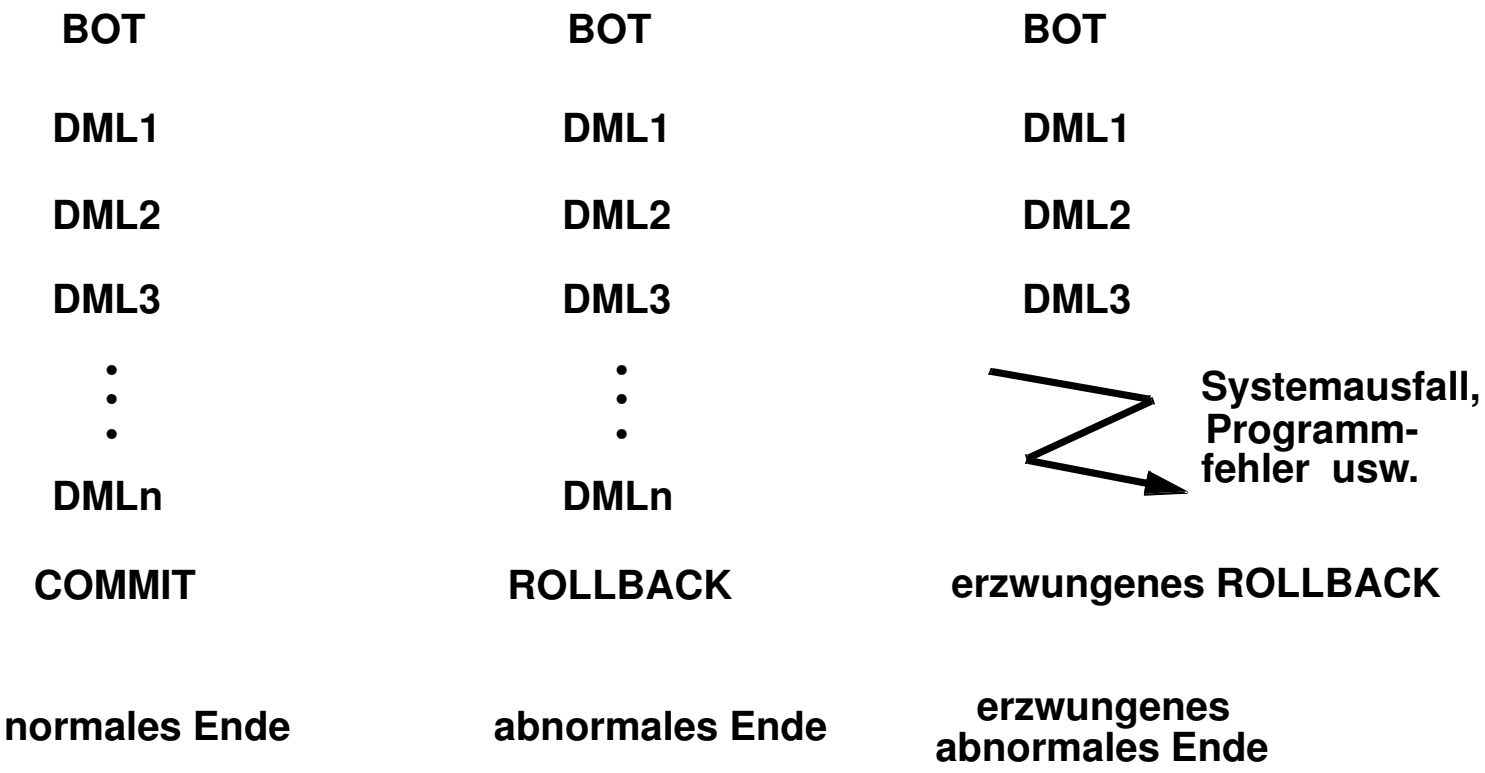
TA-Verwaltung (4)

- **Transaktionsablauf**
  - Erinnerung



TA-Verwaltung (5)

- **Transaktionsablauf (Forts.)**
  - Mögliche Ausgänge einer Transaktion



DB-Recovery (1)

- **Automatische Behandlung aller ‚erwarteten' Fehler durch das DBVS**
- **Voraussetzung**
  - Sammeln redundanter Information während des normalen Betriebs (Logging)
- **Fehlermodell von zentralisierten DBVS**
  - Transaktionsfehler
  - Systemfehler
  - Gerätefehler
- **Achtung: „A recoverable action is 30% harder and requires 20% more code than a non-recoverable action" (J. Gray)**
- **TA-Paradigma verlangt**
  - Alles-oder-Nichts-Eigenschaft von TAs
  - Dauerhaftigkeit erfolgreicher Änderungen
- **Zielzustand nach erfolgreicher Recovery: jüngster transaktionskonsistenter DB-Zustand**
  - Durch die Recovery ist der jüngste Zustand vor Erkennen des Fehlers wiederherzustellen, der allen semantischen Integritätsbedingungen entspricht, der also ein möglichst aktuelles, exaktes Bild der Miniwelt darstellt

DB-Recovery (2)

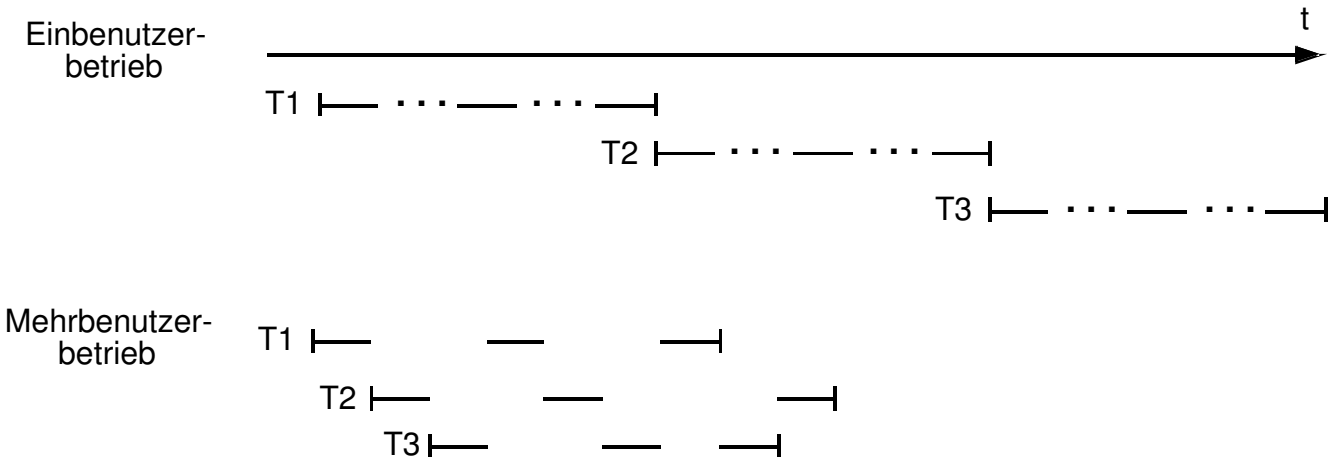
- **Recovery-Arten**
  1. **Transaktions-Recovery**
     - Zurücksetzen einzelner (noch nicht abgeschlossener) Transaktionen im laufenden Betrieb (Transaktionsfehler, Deadlock)
     - Arten:
       - Vollständiges Zurücksetzen auf Transaktionsbeginn (TA-UNDO)
       - Partielles Zurücksetzen auf Rücksetzpunkt (Savepoint) innerhalb der Transaktion
  2. **Crash-Recovery nach Systemfehler**
     - Wiederherstellen des jüngsten transaktionskonsistenten DB-Zustands
     - Notwendige Aktionen
       - (partielles) REDO für erfolgreiche Transaktionen (Wiederholung verlorengegangener Änderungen)
       - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen der Änderungen aus der permanenten DB)

DB-Recovery (3)

- **Recovery-Arten (Forts.)**
  - **3. Medien-Recovery nach Gerätefehler**
    - Spiegelplatten bzw.
    - Vollständiges Wiederholen (REDO) aller Änderungen (erfolgreich abgeschlossener Transaktionen) auf einer Archivkopie
  - **4. 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)

Synchronisation (1)

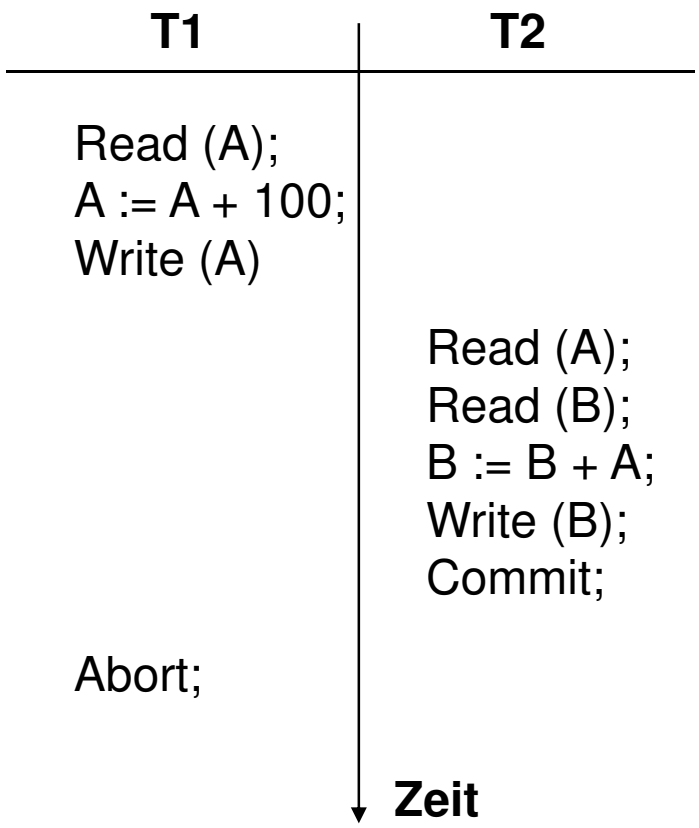
- **Einbenutzer-/Mehrbenutzerbetrieb**



  - CPU-Nutzung während TA-Unterbrechungen
    - E/A
    - Denkzeiten bei Mehrschritt-TA
    - Kommunikationsvorgänge in verteilten Systemen
  - bei langen TAs zu große Wartezeiten für andere TA (Scheduling-Fairness)

Synchronisation (2)

- **Anomalien im unkontrollierten Mehrbenutzerbetrieb**
  1. Abhängigkeit von nicht-freigegebenen Änderungen (Dirty-Read)
     - Geänderte, aber noch nicht freigegebene Daten werden als „schmutzig" bezeichnet (dirty data), da die TA ihre Änderungen bis Commit (einseitig) zurücknehmen kann
     - Schmutzige Daten dürfen von anderen TAs nicht in „kritischen" Operationen benutzt werden

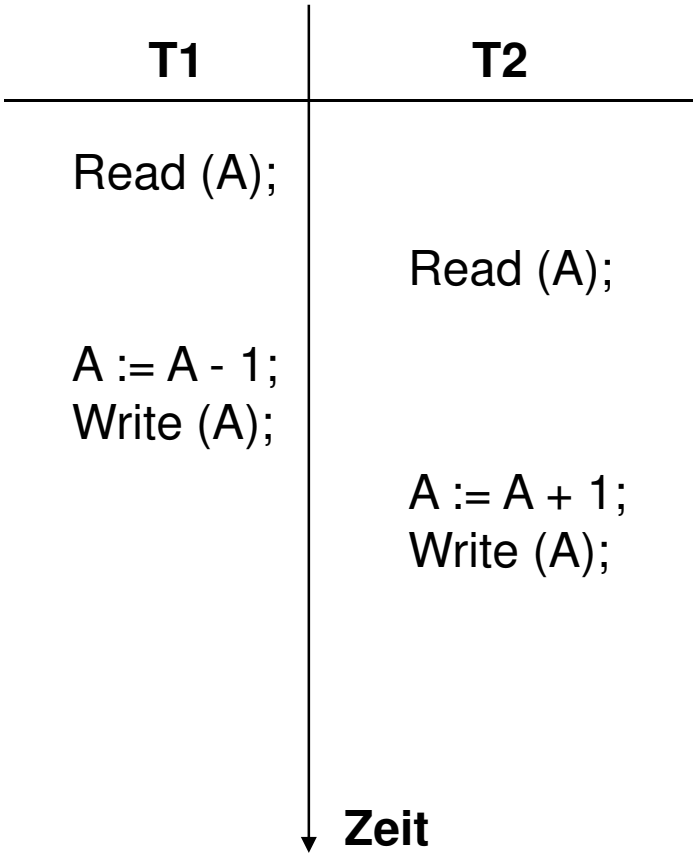


Synchronisation (3)

- **Anomalien im unkontrollierten Mehrbenutzerbetrieb**

  2. Verlorengegangene Änderung (Lost Update)

     - ist in jedem Fall auszuschließen



Synchronisation (4)

- **Anomalien im unkontrollierten Mehrbenutzerbetrieb**

  3. Inkonsistente Analyse (Non-repeatable Read)

Lesetransaktion (Gehaltssumme berechnen)	**Änderungstransaktion**	**DB-Inhalt** (Pnr, Gehalt)
		2345 39.000
		3456 48.000
SELECT Gehalt INTO :gehalt FROM Pers WHERE Pnr = 2345;		
summe := summe + gehalt;		
	UPDATE Pers SET Gehalt = Gehalt + 1000 WHERE Pnr = 2345;	
		2345 40.000
	UPDATE Pers SET Gehalt = Gehalt + 2000 WHERE Pnr = 3456;	
		3456 50.000
SELECT Gehalt INTO :gehalt FROM Pers WHERE Pnr = 3456;		
summe := summe + gehalt;		
		Zeit

Synchronisation (5)

- **Anomalien im unkontrollierten Mehrbenutzerbetrieb**

  4. Phantom-Problem

Lesetransaktion **(Gehaltssumme überprüfen)**	**Änderungstransaktion** **(Einfügen eines neuen Angestellten)**
SELECT SUM (Gehalt) INTO :summe **FROM Pers** **WHERE Anr = 17;**	
	INSERT INTO Pers (Pnr, Anr, Gehalt) **VALUES (4567, 17, 55.000);**
	UPDATE Abt **SET Gehaltssumme = Gehaltssumme + 55.000** **WHERE Anr = 17;**
SELECT Gehaltssumme INTO :gsumme **FROM Abt** **WHERE Anr = 17;**	
IF gsumme <> summe THEN **<Fehlerbehandlung>;**	

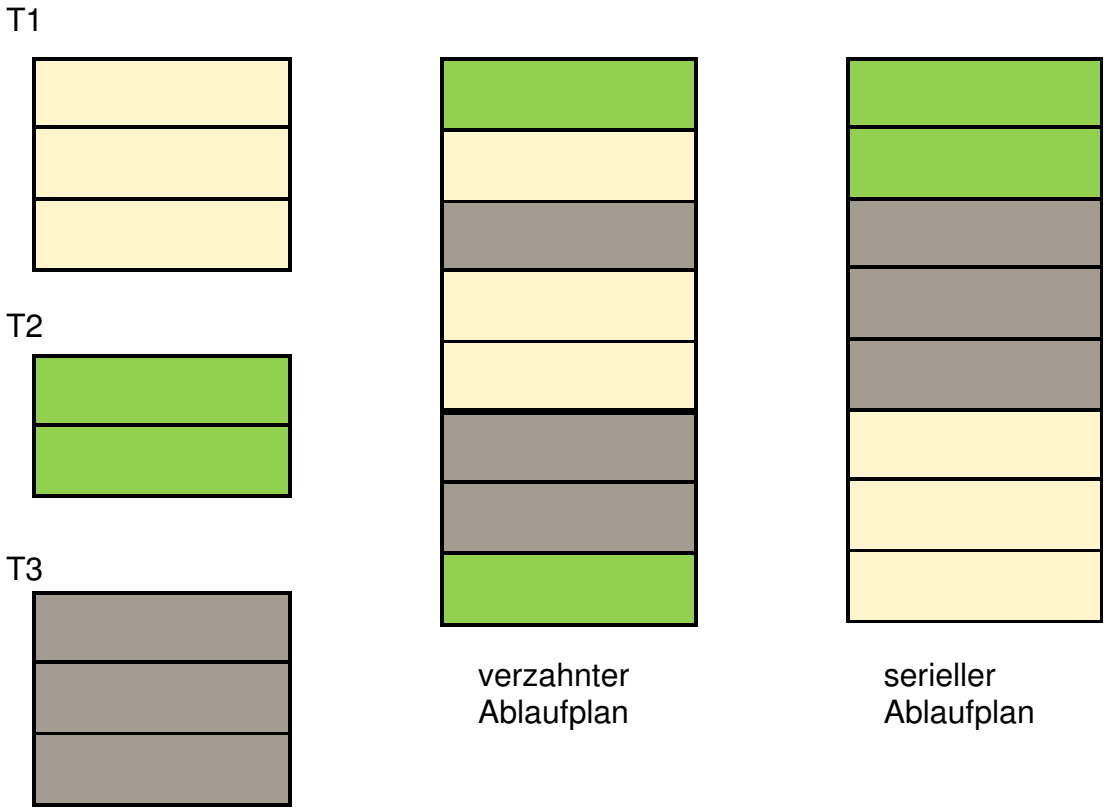
Zeit

Synchronisation (6)

- **Korrektheit – Vorüberlegungen**
  - einzelne Transaktion T
    - wenn T allein auf einer konsistenten DB ausgeführt wird, dann terminiert T (irgendwann) und hinterlässt die DB in einem konsistenten Zustand
    - während der TA-Verarbeitung gibt es keine Konsistenzgarantien!
  - mehrere TAs
    - wenn Transaktionen seriell ausgeführt werden, dann bleibt die Konsistenz der DB erhalten
    - Ziel der Synchronisation: logischer Einbenutzerbetrieb, d.h. Vermeidung aller Mehrbenutzeranomalien

Synchronisation (7)

- **Korrektheit – Vorüberlegungen (Forts.)**
  - mehrere TAs



Synchronisation (8)

- **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.

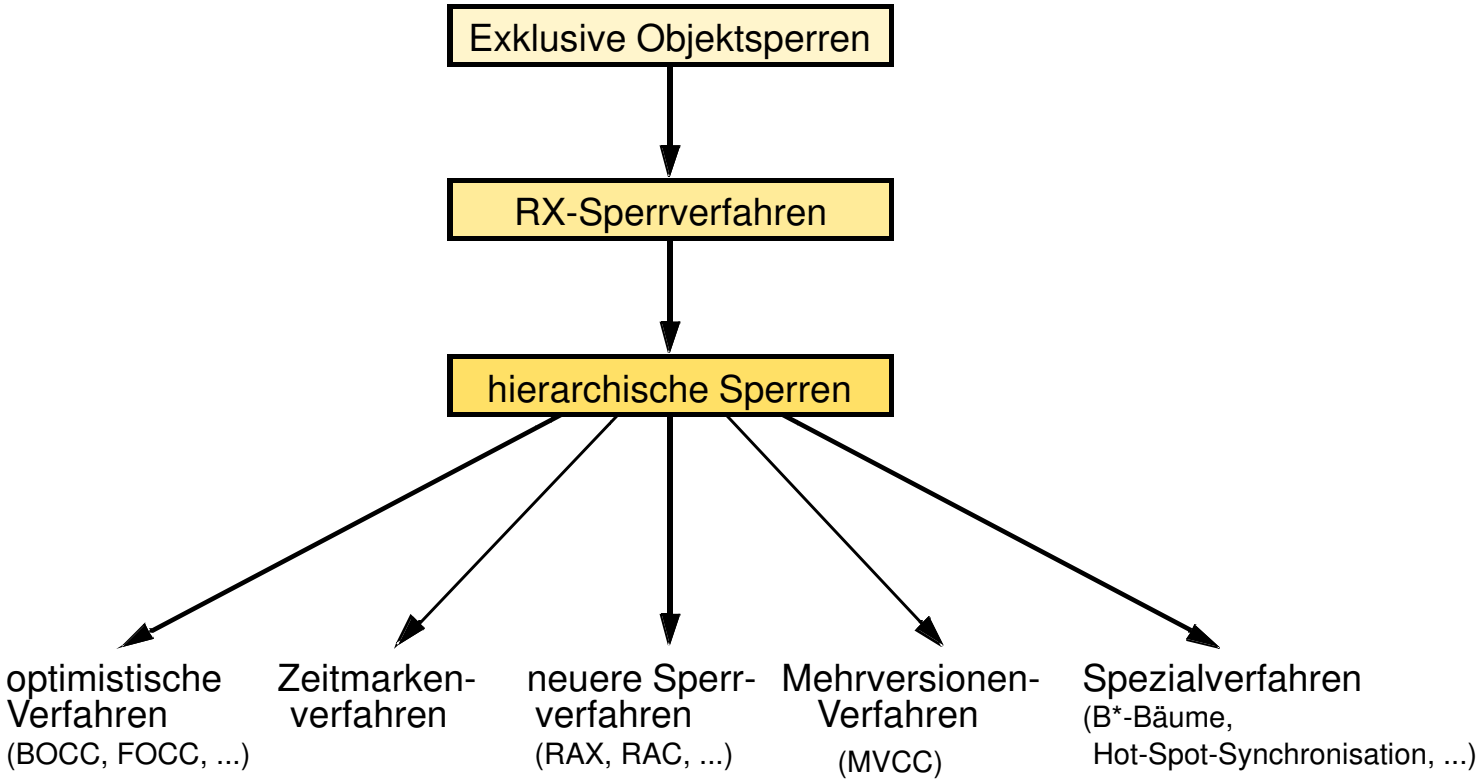
  - Hintergrund:
    - Serielle Ablaufpläne sind korrekt
    - Jeder Ablaufplan, der denselben Effekt wie ein serieller erzielt, ist akzeptierbar

Synchronisation (9)

- **Einbettung des DB-Schedulers**
  - als Komponente der Transaktionsverwaltung zuständig für I von ACID
  - kontrolliert die beim TA-Ablauf auftretenden Konfliktoperationen (Read/Write, Write/Read, Write/Write) und garantiert insbesondere, dass nur „serialisierbare" TA erfolgreich beendet werden
  - nicht serialisierbare TAs müssen verhindert werden; dazu ist Kooperation mit der Recovery-Komponente erforderlich (Rücksetzen von TA).
- **Zur Realisierung der Synchronisation gibt es viele Verfahren**
  - Pessimistisch
  - Optimistisch
  - Versionsverfahren
  - Zeitmarkenverfahren
  - etc.
- **Sperrbasierte (pessimistische) Verfahren**
  - bei einer Konfliktoperation blockieren sie den Zugriff auf das Objekt
  - universell einsetzbar
  - es gibt viele Varianten

Synchronisation (10)

- **Historische Entwicklung von Synchronsiationsverfahren**



Synchronisation (11)

- **RX-Sperrverfahren**
  - Sperrmodi
    - Sperrmodus des Objektes: NL (no lock), R (read), X (exclusive)
    - Sperranforderung einer Transaktion: R, X
  - Kompatibilitätsmatrix

	NL	S	X
S	+	+	-
X	+	-	-

    - Falls Sperre nicht gewährt werden kann, muss die anfordernde TA warten, bis das Objekt freigegeben wird (Commit/Abort der die Sperre besitzenden TA)
    - Wartebeziehungen werden in einem Wait-for-Graph verwaltet

Synchronisation (12)

- **RX-Sperrverfahren (Forts.)**
  - Beispiel: Ablauf von Transaktionen
    (aus Sicht des Schedulers; an der SQL-Schnittstelle ist die Sperranforderung und –freigabe nicht sichtbar)

T1	T2	a	b	Bemerkung
		NL	NL	
lock(a, X)		X		
...				
	lock(b, R)		R	
	...			
lock(b, R)			R	
	lock(a, R)	X		T2 wartet
...				
unlock(a)		NL → R		T2 wecken
...	...			
unlock(b)			R	

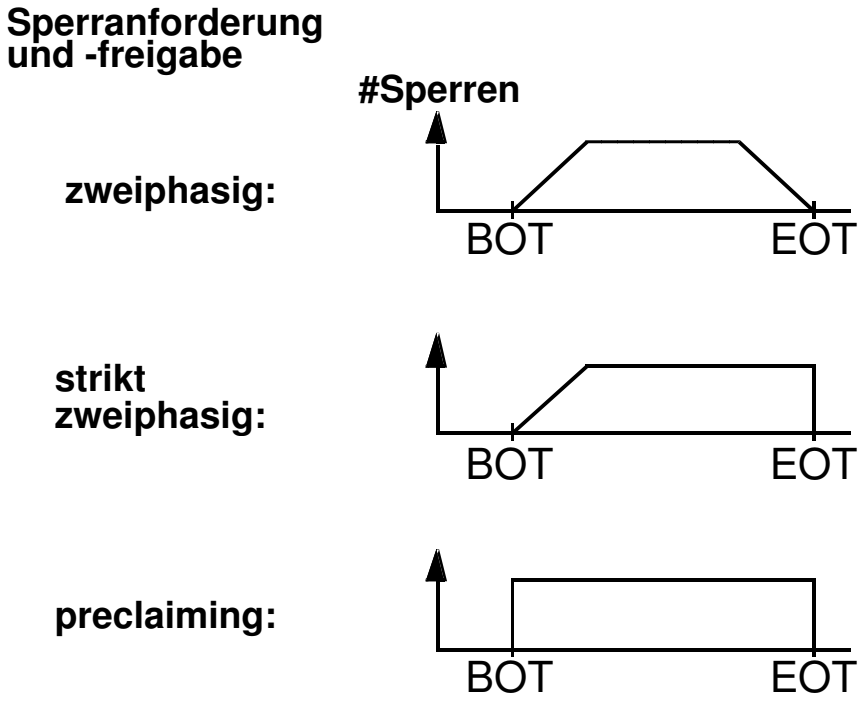
Synchronisation (13)

- **RX-Sperrverfahren (Forts.)**
  - Einhaltung folgender Regeln gewährleistet Serialisierbarkeit:
    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:
    - Anfordern von Sperren erfolgt in einer Wachstumsphase
    - Freigabe der Sperren in Schrumpfungsphase
    - Sperrfreigabe kann erst beginnen, wenn alle benötigten Sperren gehalten werden
    5. Spätestens bei Commit sind alle Sperren freizugeben

Eswaran, K.P. et al.: The notions of consistency and predicate locks in a data base system, Comm. ACM 19:11, 1976, pp. 624-63

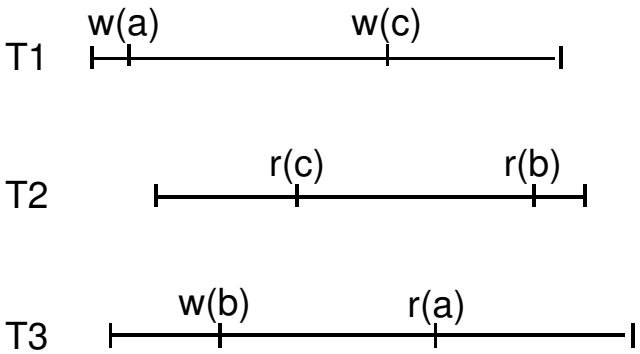
Synchronisation (14)

- **RX-Sperrverfahren (Forts.)**
  - Formen der Zweiphasigkeit
    - Praktischer Einsatz erfordert **striktes 2PL**
    - Gibt alle Sperren erst bei Commit frei
    - Verhindert kaskadierendes Rücksetzen

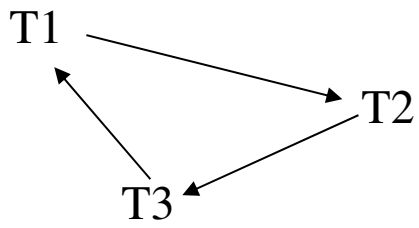


Synchronisation (15)

- **RX-Sperrverfahren (Forts.)**
  - Deadlocks/Verklemmungen
    - Auftreten von Verklemmungen ist **inhärent** und kann bei pessimistischen Methoden (blockierende Verfahren) nicht vermieden werden
  - Beispiel einer nicht-serialisierbaren Historie, die zu einer Verklemmung führt



Warte-Beziehungen:



Synchronisation (16)

- **RX-Sperrverfahren (Forts.)**
  - Allgemeine Forderungen
    - Wahl des gemäß der Operation schwächst möglichen Sperrmodus
    - Möglichkeit der Sperrkonversion (upgrading), falls stärkerer Sperrmodus erforderlich
    - Anwendung: viele Objekte sind zu lesen, aber nur wenige zu aktualisieren
- **Erweiterung: RUX**
  - Ziel: Verhinderung von Konversions-Deadlocks
  - U-Sperre für Lesen mit Änderungsabsicht (Prüfmodus)
  - bei Änderung Konversion U $\rightarrow$ X, andernfalls U $\rightarrow$ R (downgrading)
  - Symmetrische Variante: Unsymmetrische Variante (z.B. in IBM DB2)

	R	U	X
R	+	+	-
U	+	-	-
X	-	-	-

	R	U	X
R	+	-	-
U	+	-	-
X	-	-	-

Synchronisation (17)

- **Konsistenzebenen**
  - Motivation
    - Serialisierbare Abläufe
      - gewährleisten „automatisch" Korrektheit des Mehrbenutzerbetriebs
      - erzwingen u. U. lange Blockierungszeiten paralleler Transaktionen
    - „Schwächere" Konsistenzebene
      - bei der Synchronisation von Leseoperationen
      - erlaubt höhere Parallelitätsgrade und Reduktion von Blockierungen, erfordert aber Programmierdisziplin!
      - Inkaufnahme von Anomalien reduziert die TA-Behinderungen

Synchronisation (20)

- **Konsistenzebenen (Forts.) – in SQL**
  - SQL erlaubt Wahl zwischen vier Konsistenzebenen (Isolation Level)
  - Konsistenzebenen sind durch die Anomalien bestimmt, die jeweils in Kauf genommen werden
  - Abgeschwächte Konsistenzanforderungen betreffen nur Leseoperationen!
  - Lost Update muss generell vermieden werden, d. h., Write/Write-Abhängigkeiten müssen stets beachtet werden

Konsistenz-ebene	Anomalie		
	Dirty Read	Non-Repeatable Read	Phantome
Read Uncommitted	+	+	+
Read Committed	-	+	+
Repeatable Read	-	-	+
Serializable	-	-	-

Zusammenfassung

- **DBS-Charakteristika**
  - Zentralisierte Verwaltung der operationalen Daten (Rolle des DBA)
  - Adäquate Schnittstellen (Datenmodell und DB-Sprache)
  - Datenkontrolle, insbes. zentrale Kontrolle der Datenintegrität und kontrollierter Mehrbenutzerbetrieb
  - Leistung und Skalierbarkeit
  - Hoher Grad an Daten-Unabhängigkeit
- **Wiederholung Transaktionskonzept**
  - ACID-Eigenschaften
  - TA-Verwaltung besitzt zwei wesentliche Komponenten
    - Synchronisation: Serialisierbarkeit als Korrektheitskriterium
    - Logging und Recovery