

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm Spruth

SS 2003

Teil 9

Transaktionsverarbeitung

Vorlesungsübersicht

Client/Server Systeme II

**Prof. Dr.-Ing. Wilhelm G. Spruth
SS2003**

**Fortsetzung der Vorlesung aus dem Wintersemester 2002/2003.
Erster Termin: Montag, 5. Mai, 2003, 11:15 - 12:45, grosser Hörsaal,
Sand 6/7.**

Es werden die folgenden Themen behandelt:

- **Transaktionsverarbeitung**
- **CICS Transaktionsmonitor**
- **ERP Systeme, SAP System R/3 Transaktionsmonitor**
- **Asynchrone Systeme, Message Based Queuing, MQSeries**
- **Java Implementierungsbeispiele:
Enterprise Java Beans, WebSphere Web Anwendungsserver**
- **Objektorientierte Client Server Systeme :
CORBA, RMI und DOT/Net, Web Services**
- **Schnittstellen zum Internet: OS/390 Internet Services**

Vertiefung durch ein Praktikum im WS 2003/2004

Client/Server Systeme II

SS 2002

Literatur

R. Buck-Emden: „Die Client/Server Technologie des SAP System R/3“. Addison-Wesley 1996.

J. Gray, A. Reuter: „Transaction Processing, Morgan Kaufmann Publishers, 1993.

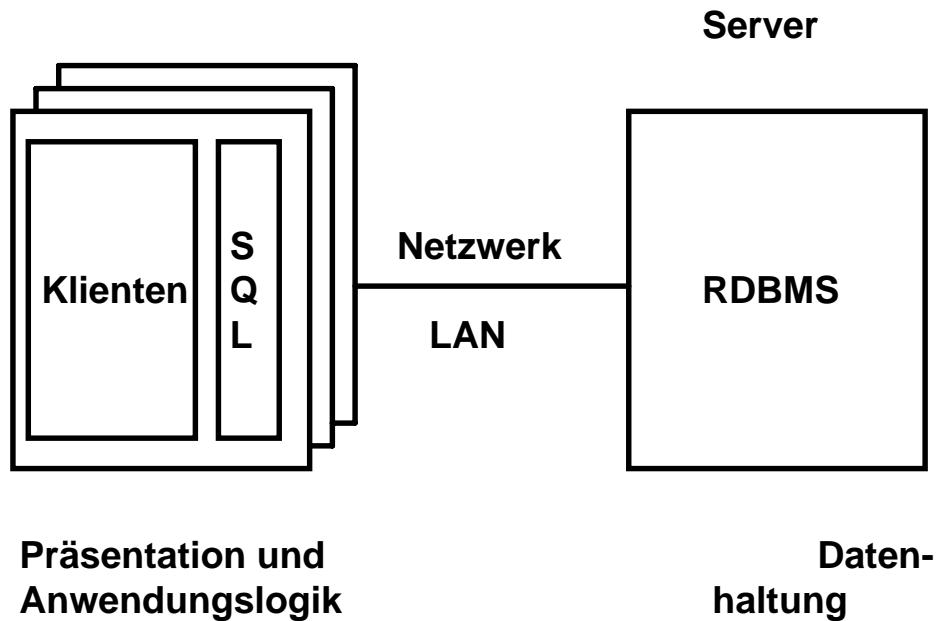
Udo Kebschull, Paul Herrmann, Wilhelm G: Spruth: „Einführung in z/OS und OS/390“. Oldenbourg-Verlag, 2002, ISBN 3-486-27214-4

J. Horswill: „Designing & Programming CICS Applications“. O’Reilly, 2000.

R. Orfali, D. Harkey: „Client/Server Programming with JAVA and CORBA“. John Wiley & Sons, 2nd edition, 1998.

L. Will: „Administration des SAP System R/3“. Addison-Wesley 1997.

W. Zack: „ Windows 2000 and Mainframe Integration“. Macmillan Technical Publishing, 1999.



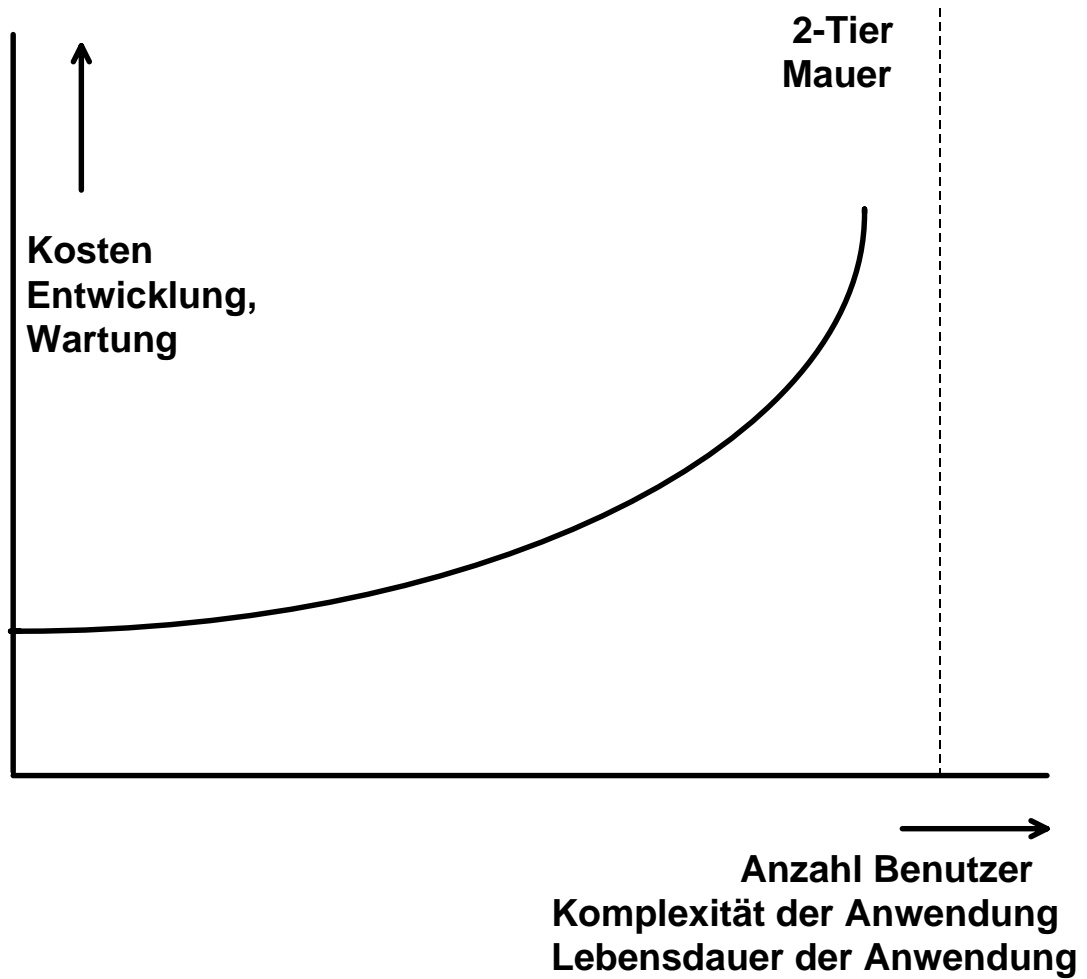
Annahmen:

- < 200 Klienten**
- < 100 000 Transaktionen / Tag**
- LAN Umgebung**
- 1 oder wenige Server**
- Mäßige Sicherheitsanforderungen**

2-Tier Client/Server Architektur

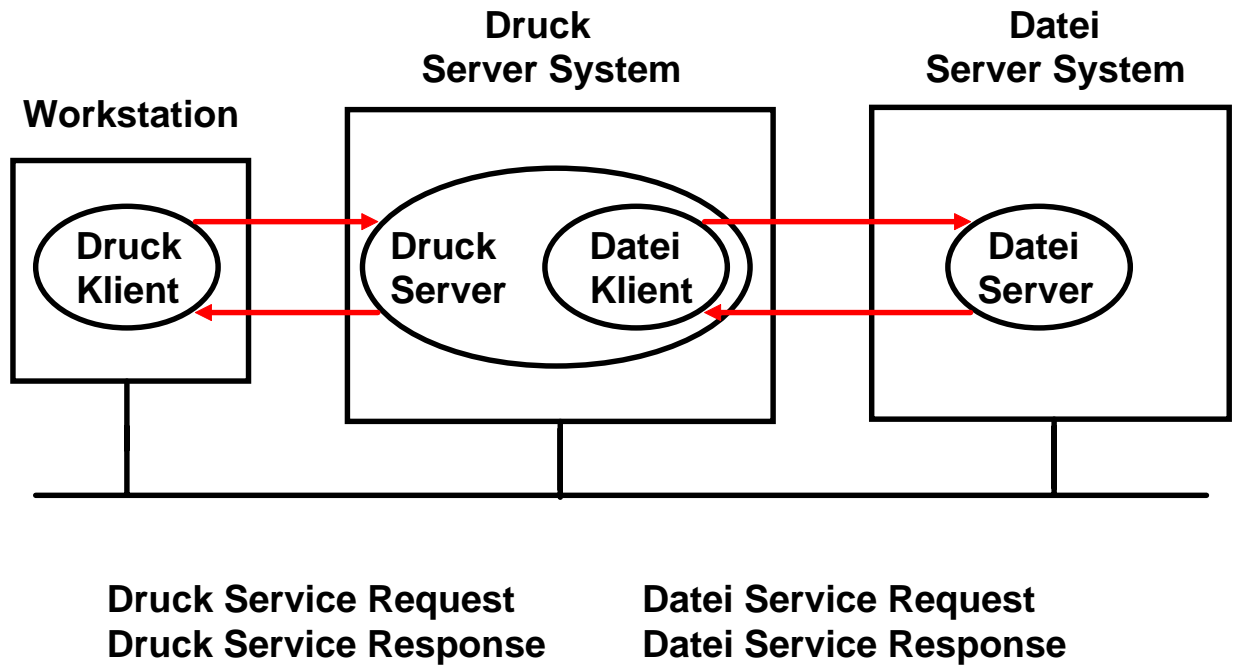
Zweistufige Client/Server Architektur

Anwendungsentwicklung mit Power Builder oder Visual Basic



Skalierung der 2-Tier Architektur

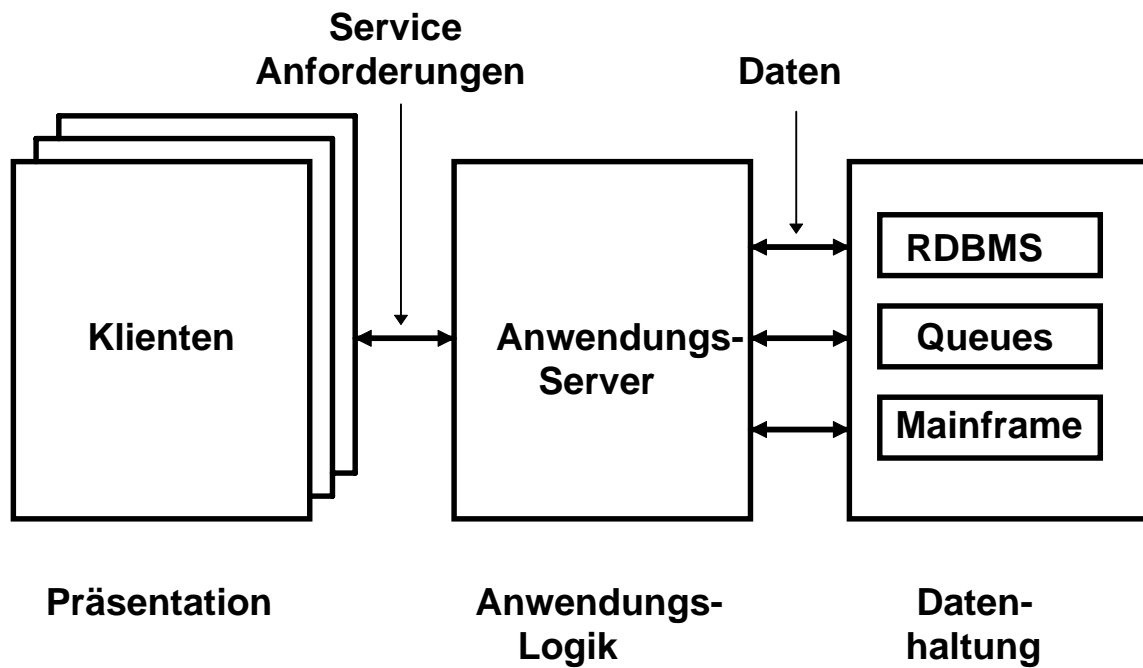
Datenvolumen auf dem Netzwerk
 Datensatz Lock Contention
 Verteilung der Klienten Software
 Sicherheit durch ACL's (hat ein Benutzer Zugriff zu den Daten,
 ist die Benutzung nicht kontrolliert).



Client/Server Betriebsmodell

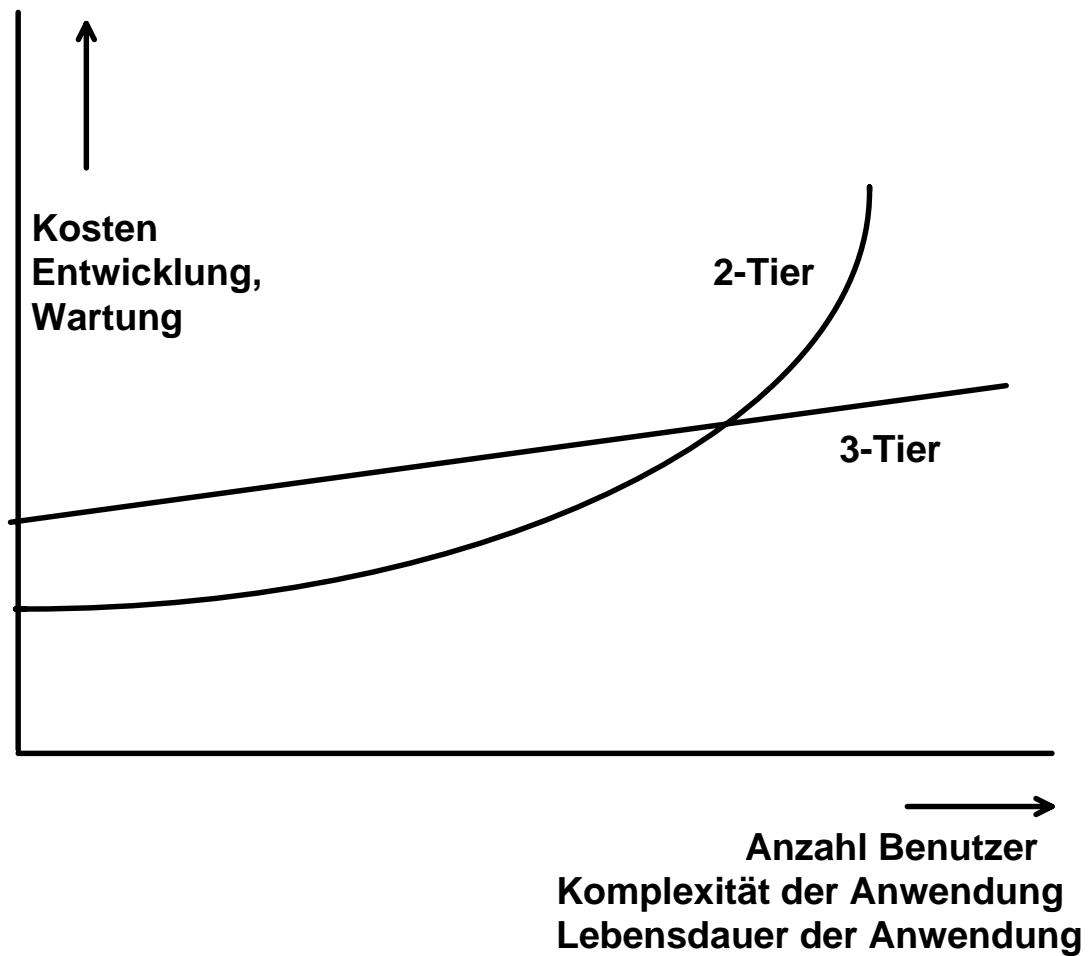
Ein System kann gleichzeitig Klient und Server sein.

Häufigste Ausprägung als 3-Tier Anwendungsserver (z.B. SAP R/3).



3-Tier Client/Server Architektur

Dreistufige Client/Server Architektur



Skalierung der 3-Tier Architektur

Service Anforderungen generieren weniger Datenvolumen
Anwendungsserver optimiert Lock Contention
Mehrfache Anwendungsserver möglich (Anwendungsreplikation)
Zugriffskontrolle auf Service Basis

Embedded SQL, Beispiel für C (1)

Ein Anwendungsprogramm implementiert typischerweise Datenbankzugriffe über eingebettete SQL-Anweisungen. Diese werden durch "exec sql" eingeleitet und durch ein spezielles Symbol (hier ";") beendet. Dies erlaubt einem Precompiler die Unterscheidung der exec sql Anweisungen von anderen Anweisungen.

```
main( )
{
    .....
    .....
    exec sql insert into PERS (PNR, PNAME) values (4711, 'Ernie');
    .....
    .....
}
```

Embedded SQL, Beispiel für C (2)

```
exec sql include sqlca; /* SQL Communication Area */
main ()
{
exec sql begin declare section;
  char  X[8];
  int   GSum;
exec sql end declare section;
exec sql connect to dbname;
exec sql insert into PERS (PNR, PNAME) values (4711, 'Ernie');
exec sql insert into PERS (PNR, PNAME) values (4712, 'Bert');
printf("ANR ? "); scanf(" %s", X);
exec sql select sum (GEHALT) into :GSum from PERS where ANR = :X;
printf("Gehaltssumme: %d\n", GSum)
exec sql commit work;
exec sql disconnect;
}
```

Anmerkungen

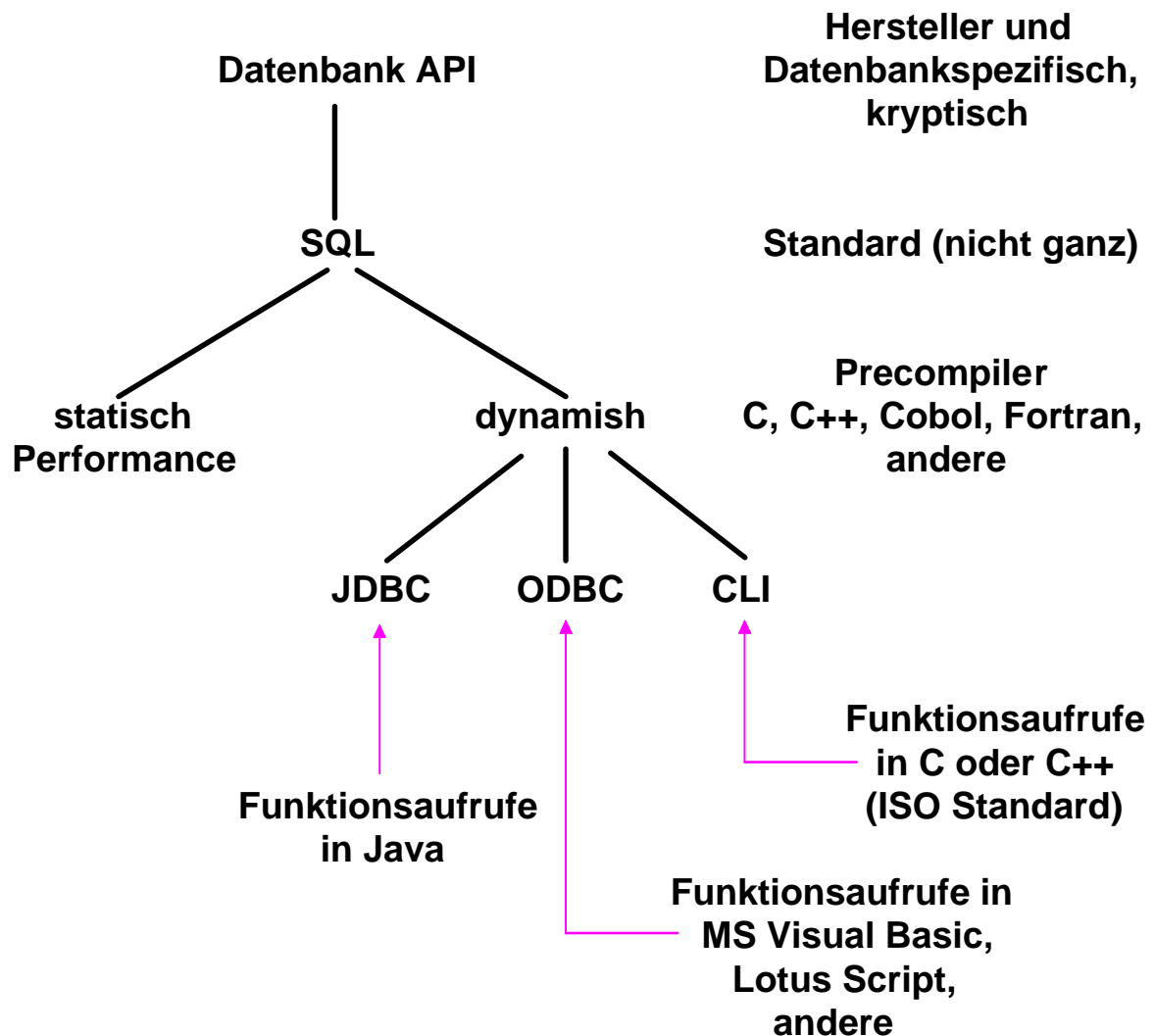
Eingebettete SQL-Anweisungen werden durch "EXEC SQL" eingeleitet und durch spezielles Symbol (hier ";") beendet, um einem Precompiler die Unterscheidung von anderen Anweisungen zu ermöglichen

Der Oracle oder DB/2 Precompiler greift sich die exec sql Befehle heraus und übersetzt sie in Anweisungen, die der C-Compiler versteht.

Die connect Anweisung baut die Verbindung zwischen Klienten und Server auf.

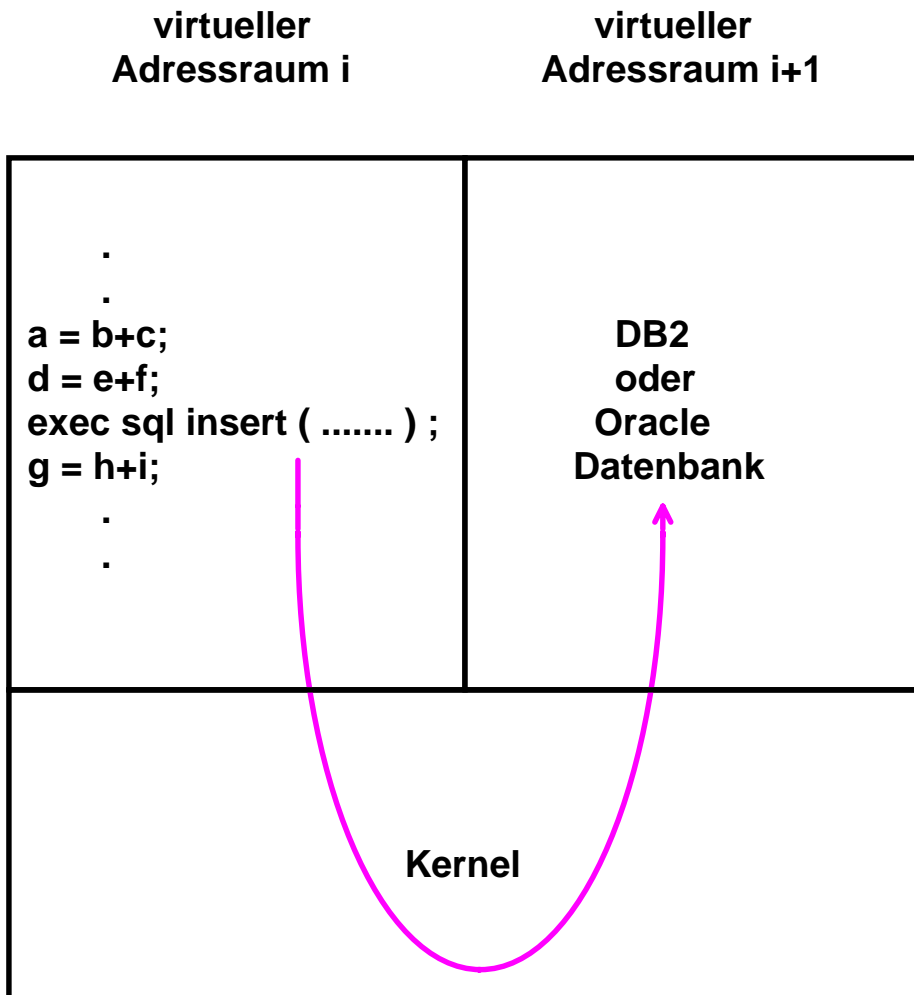
Es wird eine Kopie von einem Teil der DB Tabelle erstellt, gegen die alle SQL Befehle Änderungen vornehmen. Die commit Anweisung macht die vorhergehenden SQL Befehle entgültig.

Kommunikationsbereich SQLCA (Rückgabe von Statusanzeigern u.ä.)



Zugriff auf SQL Datenbank

Dynamisches SQL ermöglicht Angabe von Parametern erst zur Laufzeit. Z.B. Benutzer gibt gewünschten Datenbanknamen und Tabellennamen in Datenfeld einer HTML Seite ein.

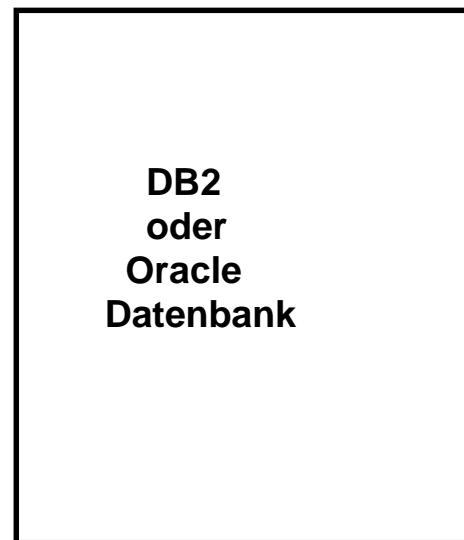
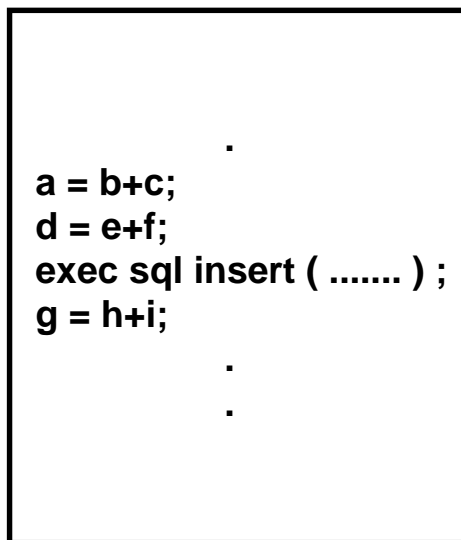


SQL Datenbank Zugriff auf dem gleichen Rechner

In einem Datenbanksystem wie Oracle oder DB2 hat der sql insert Aufruf ACID Eigenschaften

Rechner 1

Rechner 2



**Kommunikationsmechanismus TCP/IP, NetBios, IPX oder SNA,
mit Socket, RPC, SMB, APPC oder CPI-C Protokoll**

ACID Eigenschaften für eine Gruppe von SQL Aufrufen

Embedded SQL Transaktion

```
DebitCreditApplication( )
{
    receive input message;
    exec sql BEGIN WORK;                /* start transaction */
    Abalance = DoDebitCredit (.....);
    if (Abalance < 0 && delta < 0)
        { exec sql ROLLBACK WORK; }
    else {
        send output message;
        exec sql COMMIT WORK;          /* end transaction */
    }
}
```

Anmerkung

Dies repräsentiert einen „Transactional RPC (TRPC)“. Der Server führt die Anweisungen zwischen

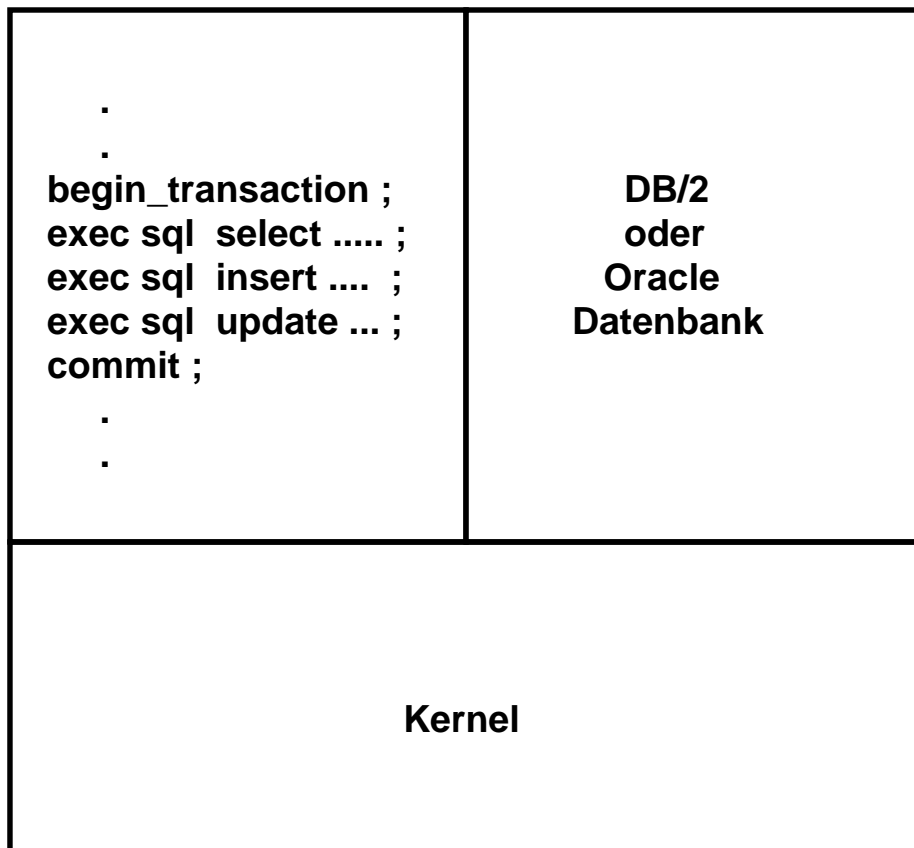
exec sql BEGIN WORK

und

exec sql COMMIT WORK

als Work Unit entsprechend den ACID Anforderungen aus.

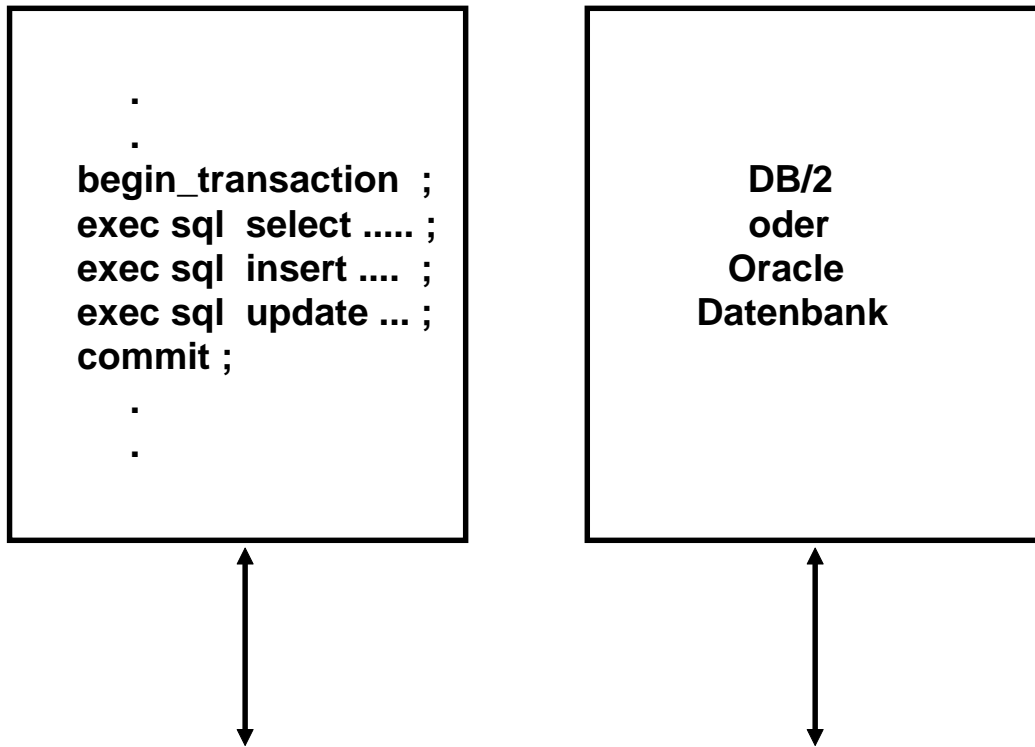
virtueller Adressraum i virtueller adressraum i+1



ACID Eigenschaften für eine Gruppe von SQL Aufrufen

Rechner 1

Rechner 2



Kommunikationsmechanismus TCP/IP oder SNA,
mit Socket, RPC, APPC oder CPI-C Protokoll

**ACID Eigenschaften für eine Gruppe
von SQL Aufrufen**

Embedded SQL Transaktion

```
DebitCreditApplication( )
{
    receive input message;
    exec sql BEGIN WORK;                /* start transaction */
    Abalance = DoDebitCredit (.....);
    if (Abalance < 0 && delta < 0)
        { exec sql ROLLBACK WORK; }
    else {
        send output message;
        exec sql COMMIT WORK;          /* end transaction */
    }
}
```

Anmerkung

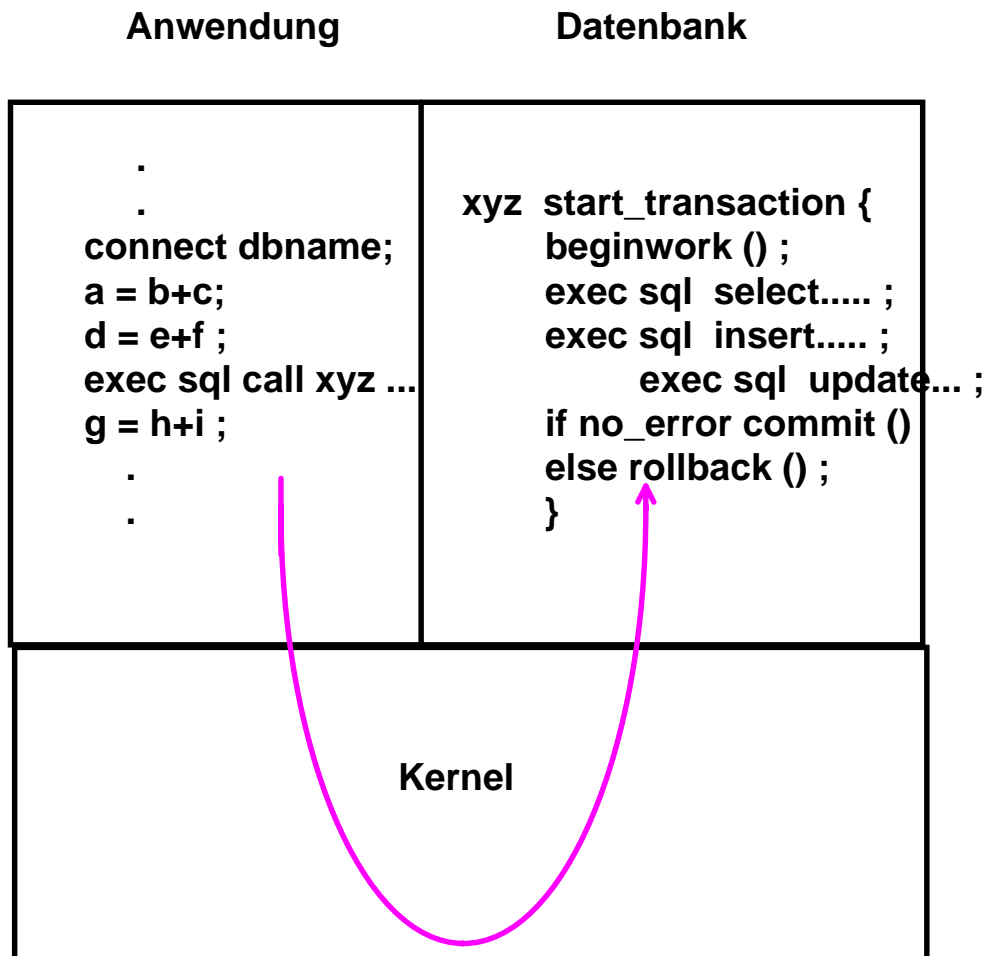
Dies repräsentiert einen „Transactional RPC (TRPC)“. Der Server führt die Anweisungen zwischen

exec sql BEGIN WORK

und

exec sql COMMIT WORK

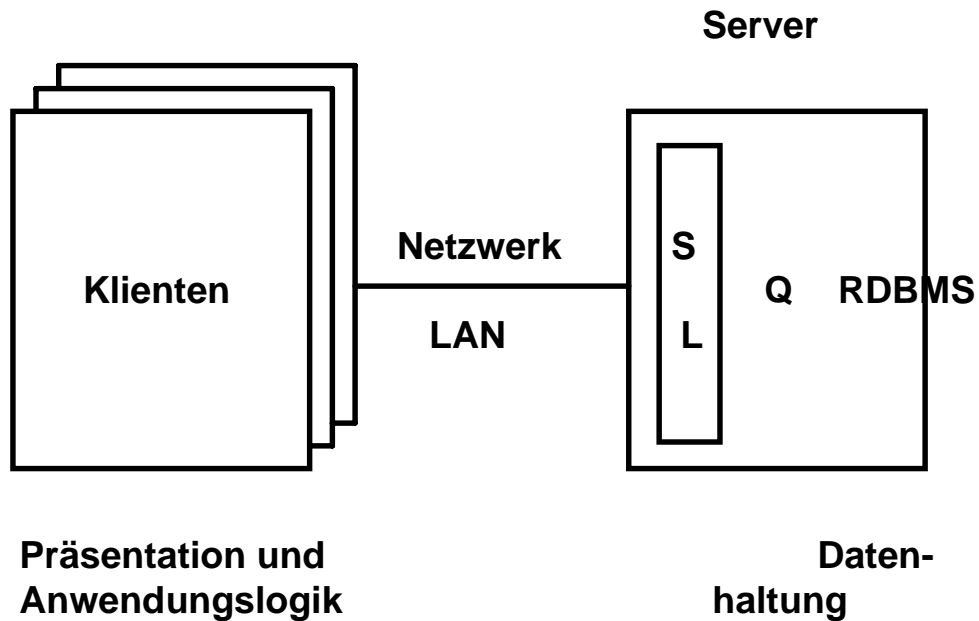
als Work Unit entsprechend den ACID Anforderungen aus.



Stored Procedure

Die Stored Procedure führt eine Gruppe von zusammenhängenden SQL Statements aus. Die Gruppe hat ACID Eigenschaften.

Stored Procedures werden manchmal als „TP light“ bezeichnet, im Gegensatz zu einem „TP heavy“ Transaktionsmonitor. Letzterer startet eigene Prozesse für mehrfache Aufrufe; innerhalb der Prozesse können nochmals Threads eingesetzt werden.



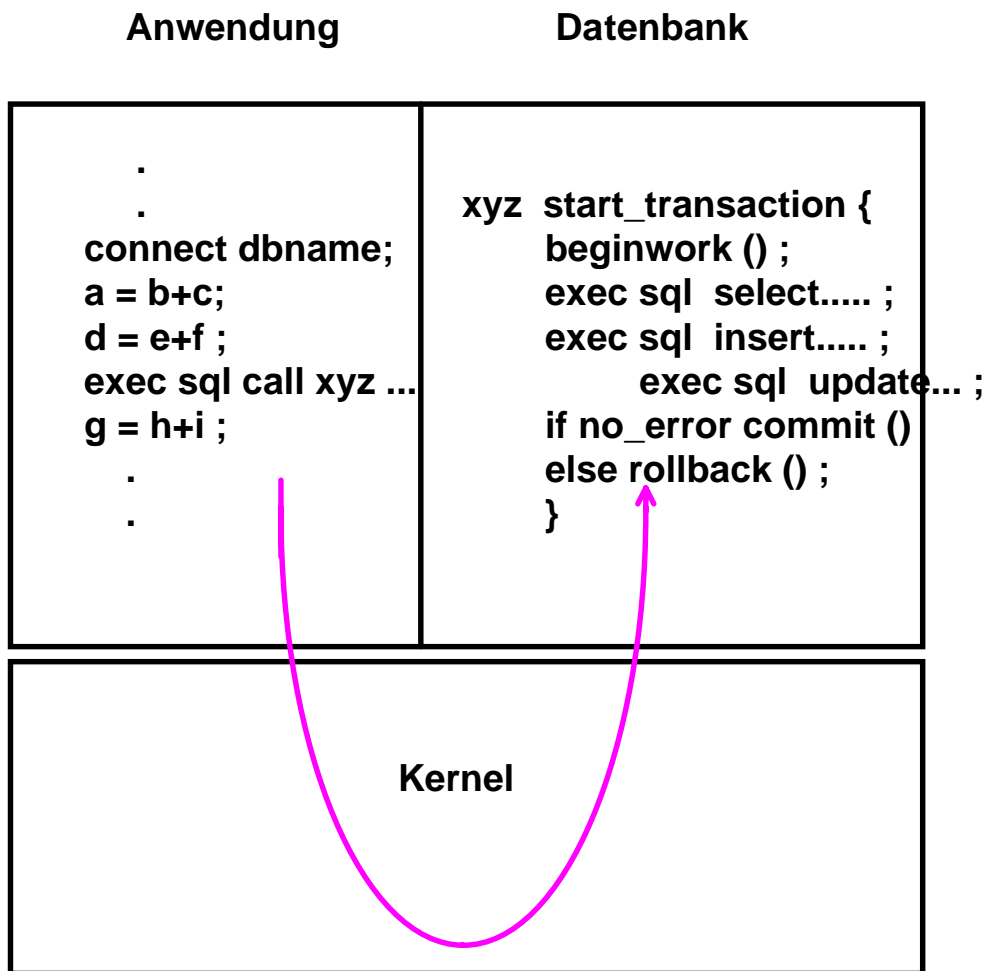
Annahmen:

- < 200 Klienten
- < 100 000 Transaktionen / Tag
- LAN Umgebung
- 1 oder wenige Server
- Mäßige Sicherheitsanforderungen

2-Tier Client/Server Architektur

Arbeiten mit Stored Procedures

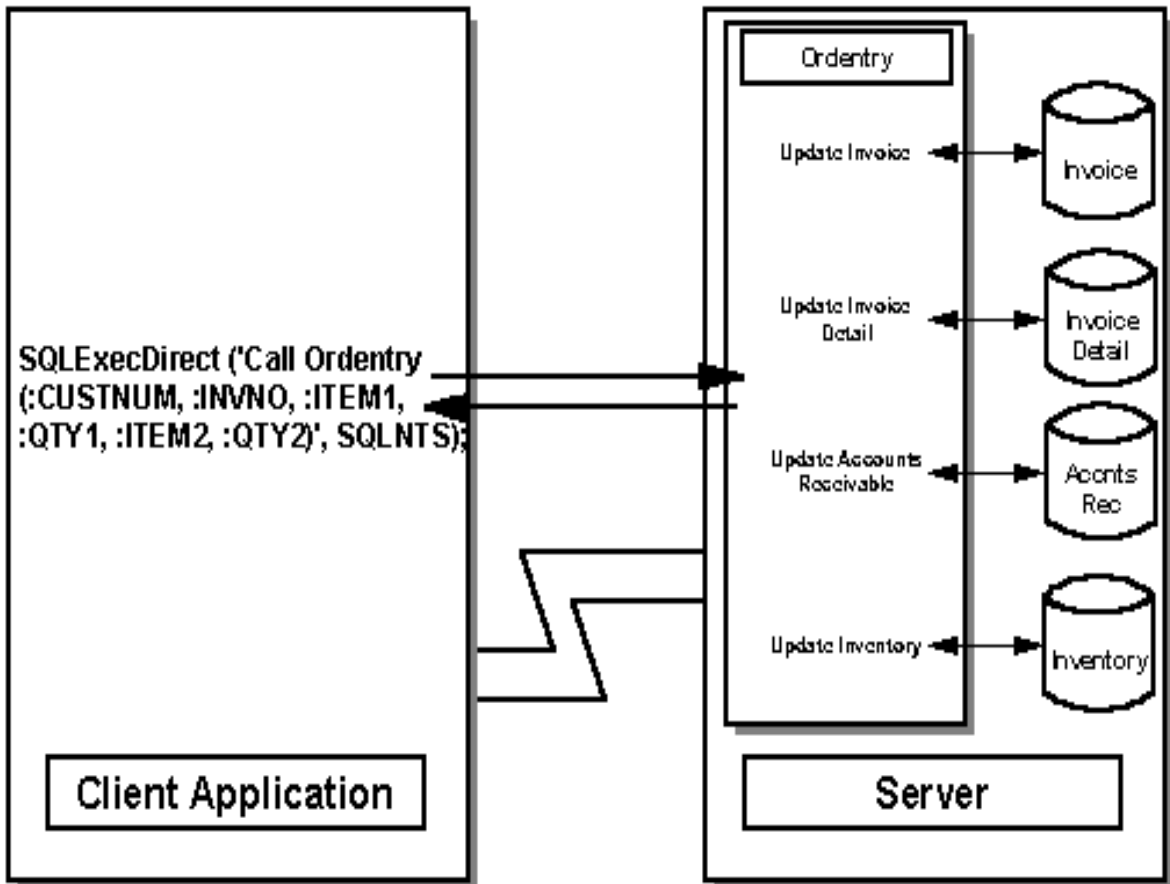
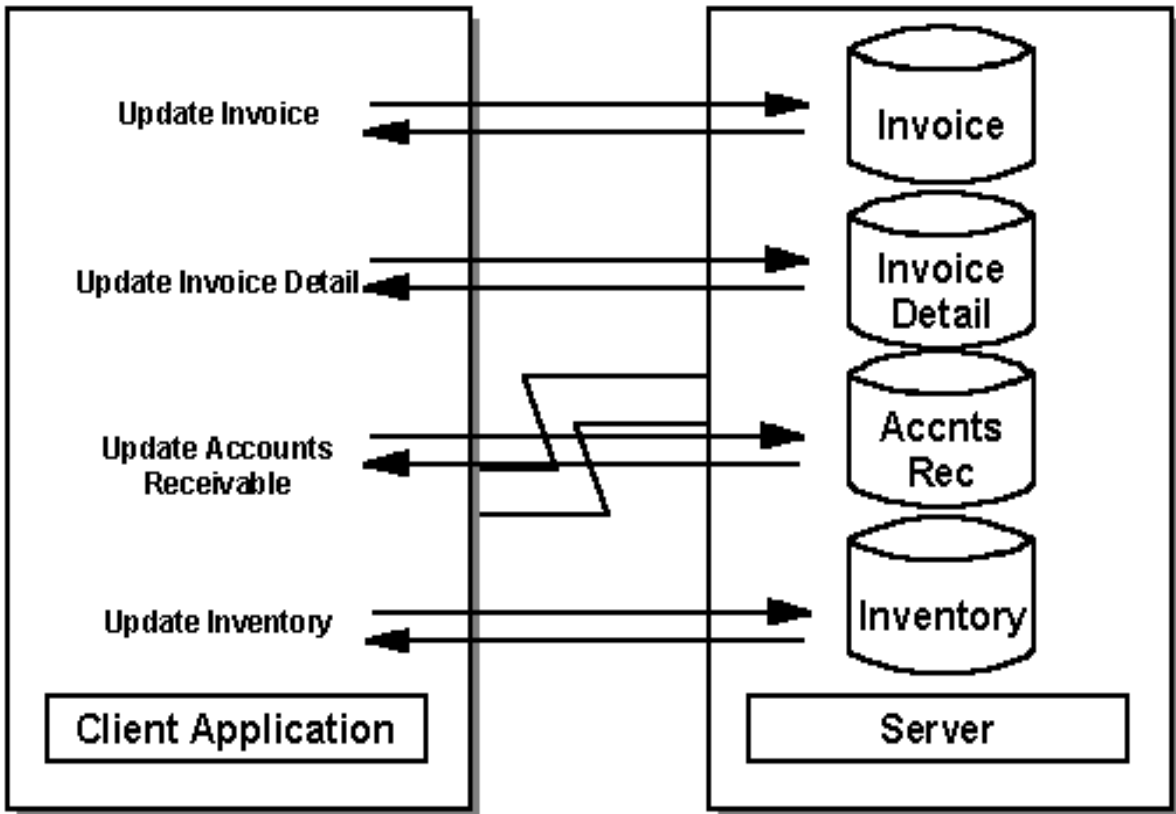
Eine Gruppe von SQL Anweisungen werden packetiert und als Teil der Datenbanksoftware ausgeführt. Nur der Aufruf der Stored Procedure (einschließlich Parameter) geht über das Netz. Der Aufruf kann z.B. über einen RPC erfolgen.



Stored Procedure

Die Stored Procedure führt eine Gruppe von zusammenhängenden SQL Statements aus. Die Gruppe hat ACID Eigenschaften.

Stored Procedures werden manchmal als „TP light“ bezeichnet, im Gegensatz zu einem „TP heavy“ Transaktionsmonitor. Letzterer startet eigene Prozesse für mehrfache Aufrufe; innerhalb der Prozesse können nochmals Threads eingesetzt werden.



Stored Procedures

Stored Procedures bündeln SQL Statements bei Zugriffen auf Relationale Datenbanken. Sie ersetzen viele, vom Klienten an den Server übergebene, SQL Statements durch eine einzige Stored Procedure Nachricht

Beispiel:

Bei einem Flugplatzreservierungssystem bewirkt eine Transaktion die Erstellung oder Abänderung mehrerer Datensätze:

- **Passenger Name Record (neu)**
- **Flugzeugauslastung (ändern)**
- **Platzbelegung (ändern)**
- **Sonderbedingungen (z.B. vegetarische Verpflegung) (ändern)**

Der Einsatz von Stored Procedures kann das Leistungsverhalten wesentlich verbessern

Nur eine Stored Procedure innerhalb eines Datenbank Programms kann in jedem Augenblick aktiv sein.

Beispiel: Datenbankprogramm mit Prozeduren für insert und delete.

System blockiert weitere RPC Aufrufe bis der laufende Aufruf abgeschlossen ist.

Stored Procedure Datenbank Klient

Jeder Datenbank Hersteller stellt eigenen proprietären Klienten zur Verfügung.

Klient enthält Driver, der proprietäres „Format und Protokoll“ (FAP) definiert.

FAP unterstützt mehrere Schicht 4 Stacks (TCP/IP, SNA, NetBios, ...)

Stored Procedures werden vom Datenbank Server in einer Library abgespeichert und mit einem Namen aufgerufen.

Das Klienten Anwendungsprogramm stellt Verbindung zur Datenbank her mit

```
EXEC SQL CONNECT TO dbname
```

und ruft Stored Procedure auf mit

```
EXEC SQL CALL ServProgName (parm1, parm2)
```

Hierbei ist:

parm1 Variablen Name (Puffer) der eine Struktur definiert (als SQLDA bei DB2 UDB bezeichnet), die zum Datenaustausch in beiden Richtungen benutzt wird.

parm2 Variablen Name (Puffer) der eine Struktur definiert die für Return Codes und Nachrichten an den Klienten benutzt wird.

Leistungsverhalten

Meßergebnisse, einfache TP1 Transaktion, OS2, Intel 80486

Dynamic SQL 2,2 Transaktionen/s

Static SQL 3,9 Transaktionen/s

Stored Procedure 10,9 Transaktionen/s

R. Orfali, D. Harkey: „Essential Client/Server Survival Guide“. John Wiley, 1994, S. 178

ACID Implementierung

optimistischer Ansatz:

Daten mit Zeitstempel (oder Versionsnr.) versehen
z.B. zusätzliches Feld in SQL Tabelle

Daten verarbeiten

if Zeitstempel unchanged then commit, else rollback

pessimistischer Ansatz:

Daten mit Lock versehen

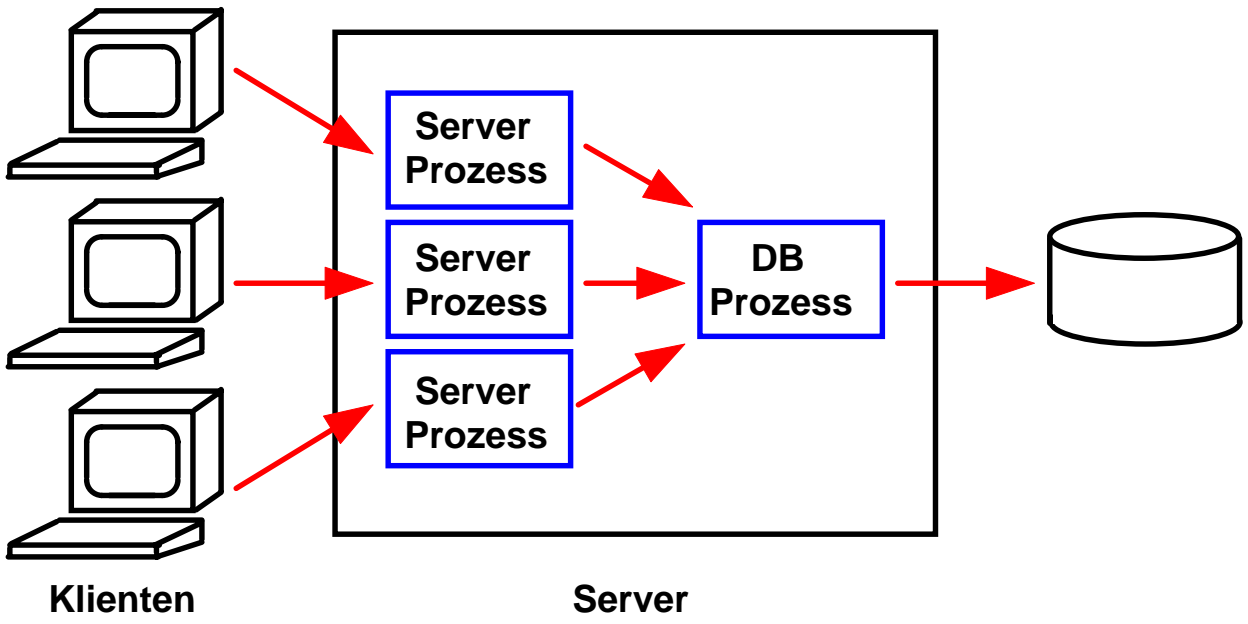
Daten verarbeiten

Ergebnis speichern

reset lock

Der optimistische Ansatz geht von der Annahme aus, daß während der Verarbeitungszeit kein anderer Prozeß auf die gleichen Daten zugreift. Falls doch, dann rollback.

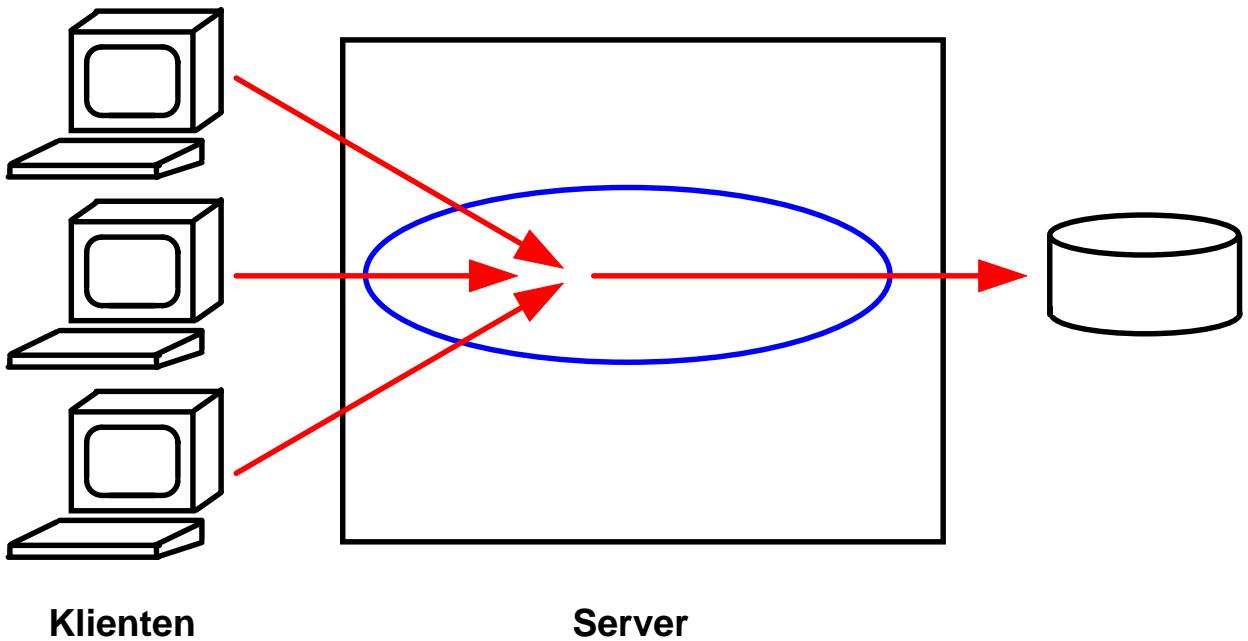
Bei starker Belastung steigt die Anzahl der rollbacks exponentiell an. Deshalb hier Locks einsetzen und Prozesse auf explizite Datenfreigabe warten lassen.



1 Prozess pro Klient

Beispiele: DB2, Informix, Oracle V6

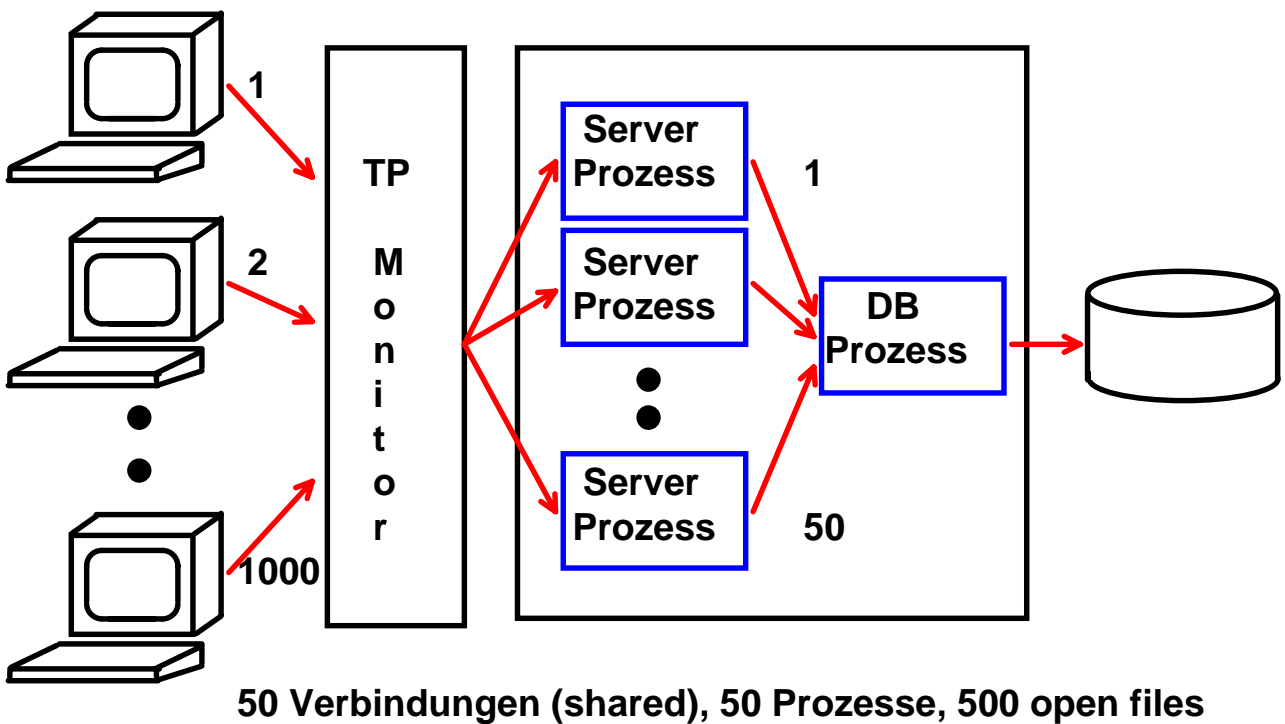
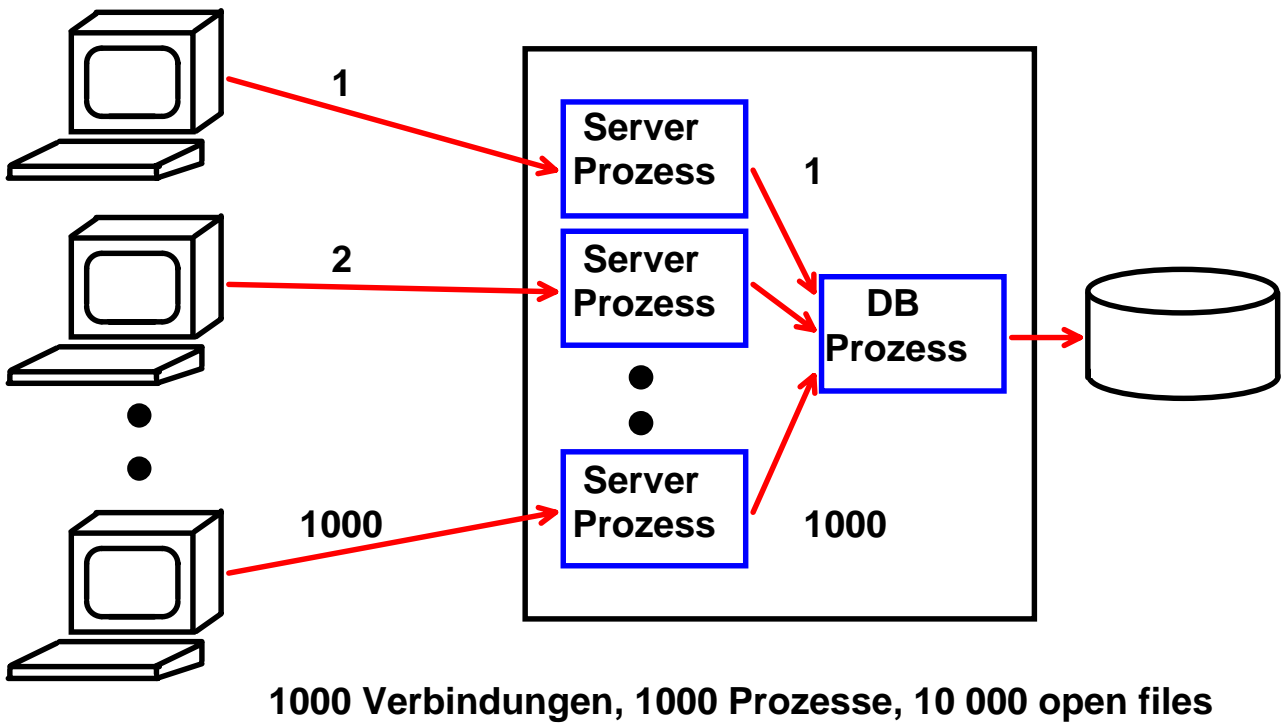
Vorteil: robust



Multithreaded Verarbeitung

alle Server Arbeiten, einschl. DB, als ein einziger multithreaded Prozess. Beispiele: Sybase, SQL Server

Vorteil: Leistungsverhalten



Transaktionsverarbeitung mit und ohne TP Monitor

Transaktionsverarbeitungssystem

Transaction Processing System, TP-System

besteht aus:

- **Anwendungen**
- **Datenbank(en)**
- **Netzwerksteuerung**
- **Entwicklungswerkzeuge**
- **Transaktionsmonitor (TP Monitor)**

Ein Transaktionsmonitor ist eine Softwarekomponente, welche den atomaren Charakter vieler gleichzeitig ablaufender Transaktionen sicherstellt. Der TP Monitor stellt die Kernfunktionen für ein Transaktionsverarbeitungssystem bereit. Hierzu gehören:

- **Message Queuing**
- **Lock Verwaltung**
- **Log Verwaltung**
- **2-Phase Commit Synchronisation**
- **Rollback Funktion**
- **Laststeuerung (Load Balancing)**

cs 0823 ww6

rahm 02-99

Stored Procedures vs. TP Monitor

2 Phase Commit

Es können mehrere TP Monitore involviert sein

Heterogene Datenbanken

Leistung

De facto alle TP Benchmarks werden mit TP Monitoren gefahren

cs 0857 ww

wgs 03-99

Beispiel für Transaktionsmonitore

BEA Tuxedo (AT&T→Novell → BEA)

IBM CICS

IBM IMS-DB/DC

IBM TPF

Microsoft Transaction Server (MTS)

SAP R/3

Siemens UTM

Tandem Pathway

Transarc (OSF) Encina

Eigenschaften eines Transaktionsmonitors:

hohe Verfügbarkeit

kurze Antwortzeit (< 0.3 Sek. erwünscht)

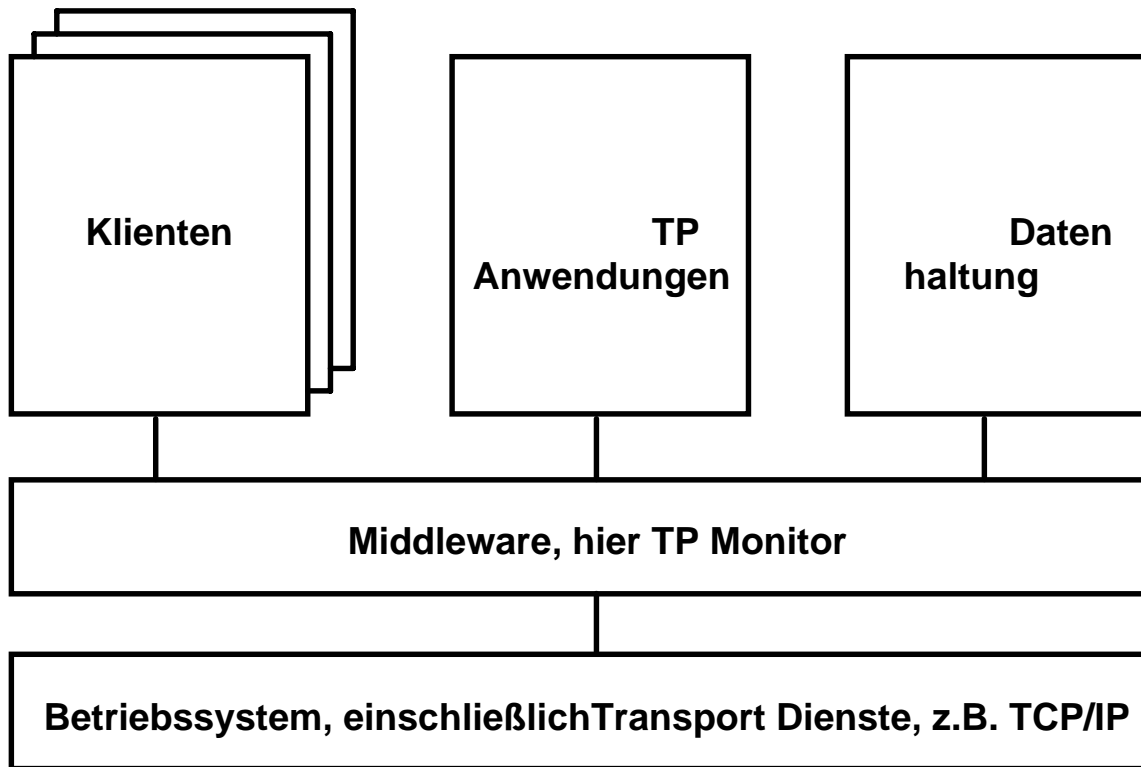
geeignet für hohes Verkehrsaufkommen

niedrige Kosten pro Transaktion

Integrität beim Zugriff auf gemeinsam genutzte

Ressourcen

Einordnung des TP Monitors



Es wäre denkbar, die TP Monitor Funktion in das Betriebssystem einzubauen. Dies ist z.B. beim Compaq/Tandem „Guardian“ und beim IBM „TPF“ Betriebssystem der Fall. Ein normales Betriebssystem ist jedoch strukturiert, vor allem Stapelverarbeitung und lang dauernde interaktive Time-Sharing Sitzungen zu unterstützen. Deshalb setzt im Regelfall der TP Monitor als ein einziger Anwendungsprozess auf dem Betriebssystem auf.

Hierbei vermeidet der TP Monitor nach Möglichkeit die Nutzung der Betriebssystem Funktionen. Um Leistung und Durchsatz zu optimieren hat er z.B. eigene Message Handling und Queuing Einrichtungen und evtl. (z.B. bei CICS) sein eigenes File System.

Stored Procedures vs. TP Monitor

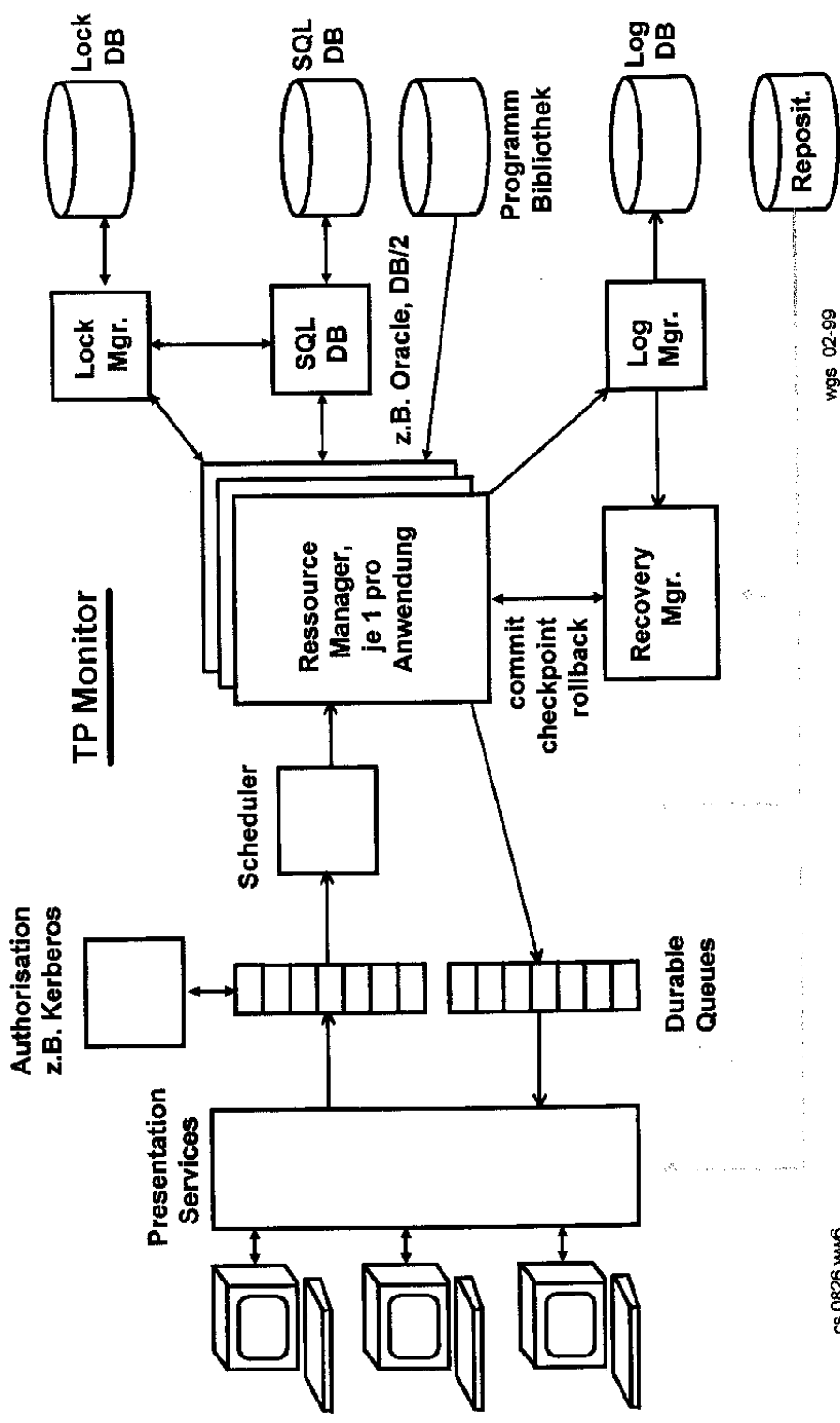
2 Phase Commit

Es können mehrere TP Monitore involviert sein

Heterogene Datenbanken

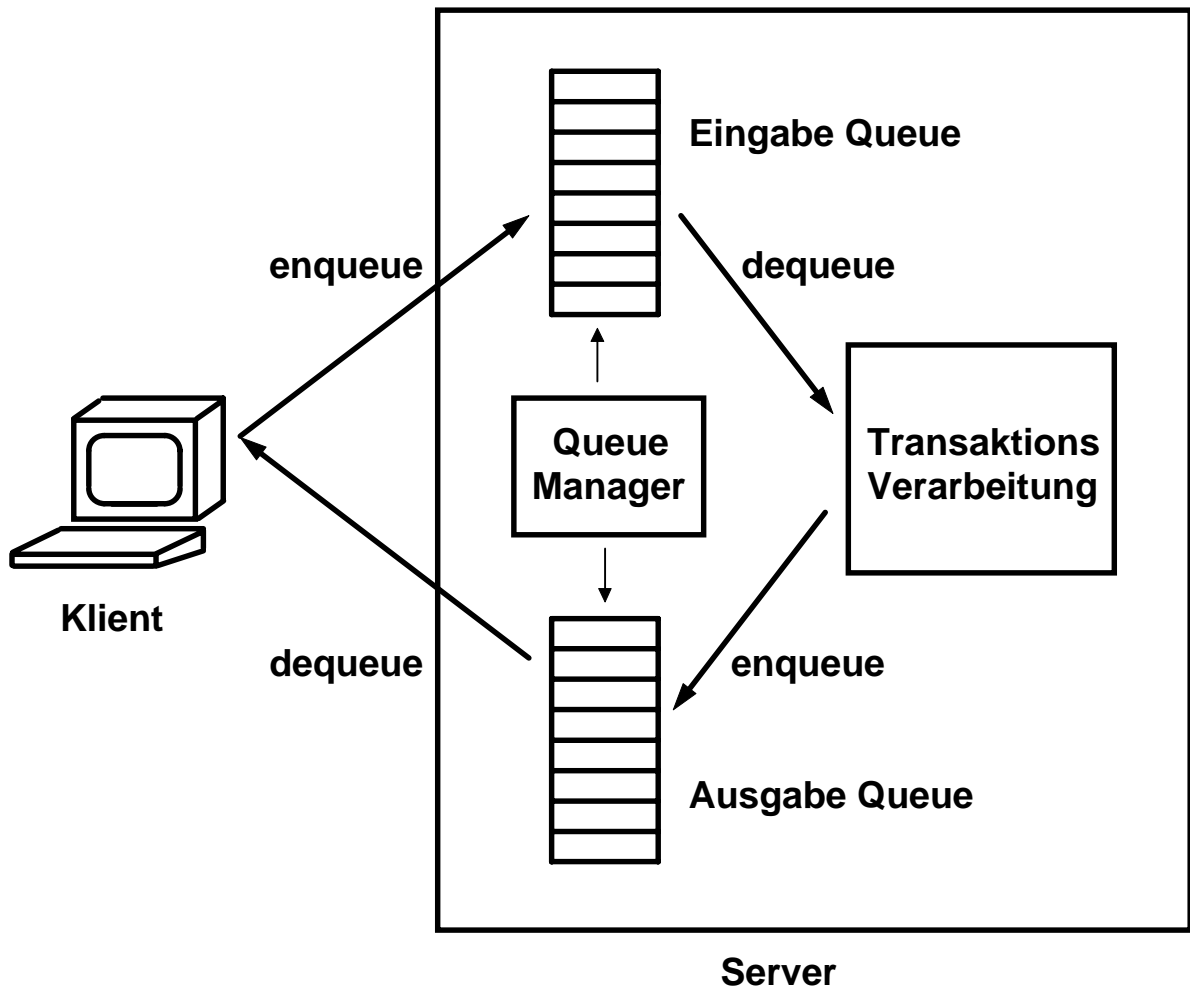
Leistung

De facto alle TP Benchmarks werden mit TP Monitoren gefahren



wgs 02-99

cs 0826 ww6



Queue Manager

Jede einzelne Transaktion wird in 3 Subtransaktionen aufgelöst, die alle eigene ACID Eigenschaften haben.

Subtransaktion 1 nimmt die Eingabenachricht entgegen, stellt sie in die Eingabe Queue, und commits.

Subtransaktion 2 dequeues die Nachricht, verarbeitet sie, stellt das Ergebnis in die Ausgabequeue, löscht den Eintrag in der Eingabequeue und commits.

Subtransaktion 3 übernimmt das Ergebnis von der Ausgabequeue, übergibt das Ergebnis an den Klienten, löscht den Eintrag in der Ausgabequeue und commits.

Ein separater Queue Manager ist optimiert für diese Aufgabe.

Komponenten eines TP Monitors

Endbenutzer kommunizieren mit dem TP Monitor mit Hilfe von Nachrichten.

Presentation Services bilden die Datenausgabe auf die GUI des Benutzers ab.

Eingabe-Nachrichten werden mit einer trid (Transaktions ID) versehen und in einer Warteschlange gepuffert. Aus Zuverlässigkeitsgründen muß diese einen Systemabsturz überleben. Eine ähnliche Warteschlange existiert für die Ausgabe von Nachrichten.

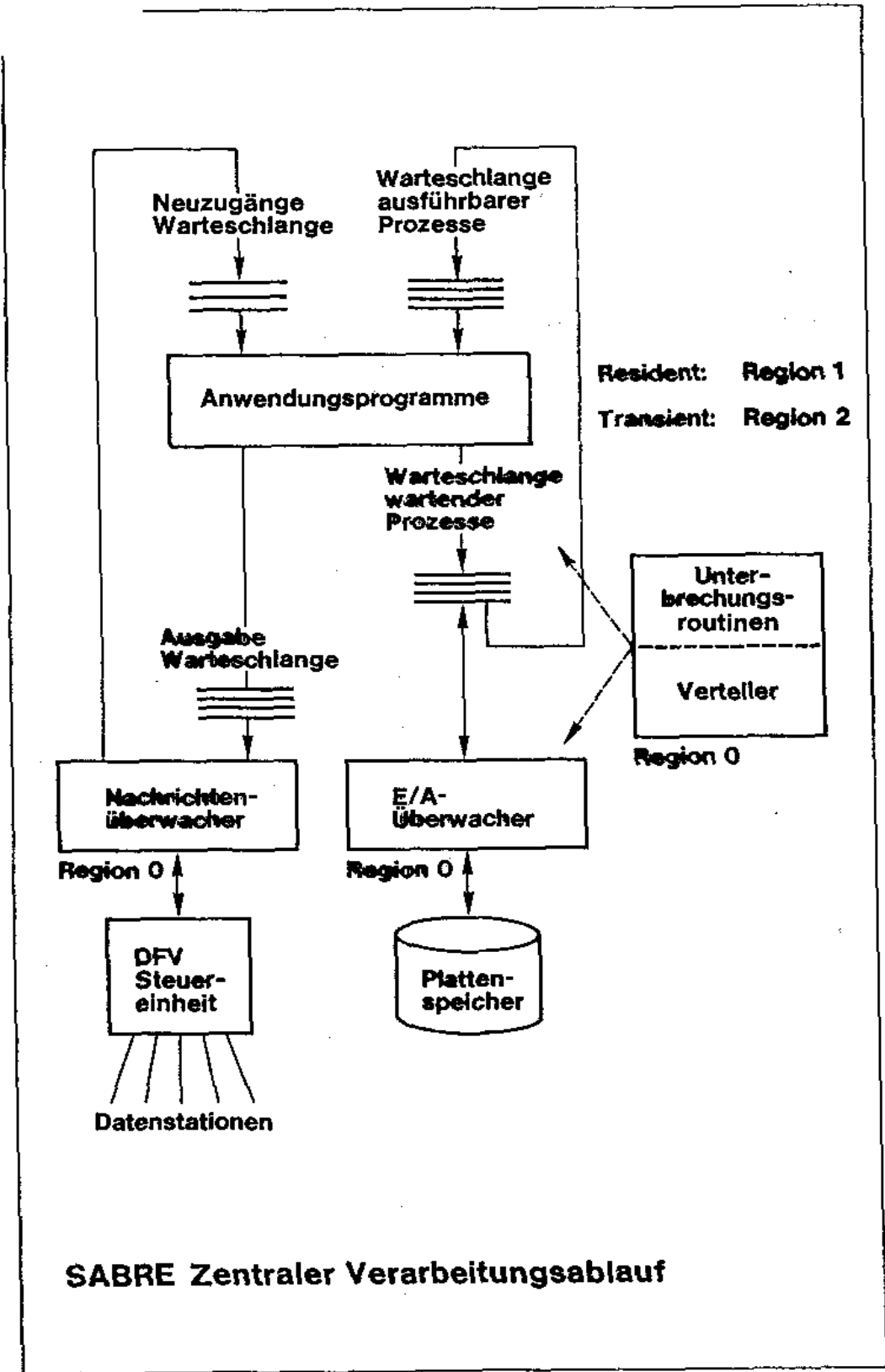
Der Scheduler verteilt eingehende Bearbeitungsanforderungen auf die einzelnen Server Prozesse.

Ein TP Monitor bezeichnet seine Server Prozesse als Ressource Manager. Es existiert ein Ressource Manager pro (aktive) Anwendung. Ressource Manager sind multithreaded; ein spezifischer Ressource Manager für eine bestimmte Anwendung ist in der Lage, mehrere Transaktionen gleichzeitig zu verarbeiten.

Der Lock Manager blockiert einen Teil einer Datenbanktabelle. In Zusammenarbeit mit dem Datenbanksystem stellt er die „Isolation“ der ACID Eigenschaft sicher.

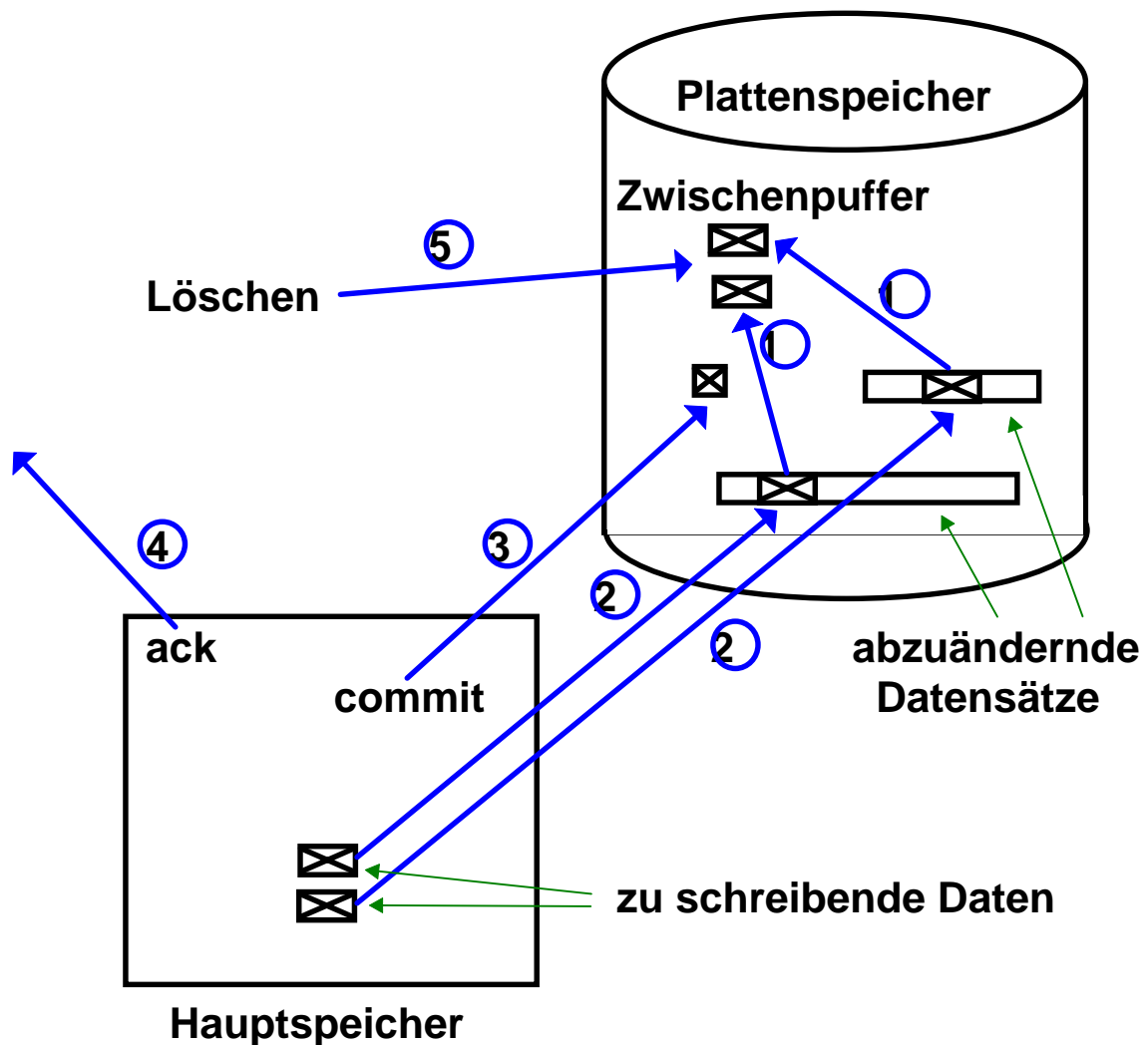
Der LOG Manager hält alle Änderungen gegen die Datenbank fest. Mit Hilfe der Log Datenbank kann der Recovery Manager im Fehlerfall den ursprünglichen Zustand der Datenbank wiederherstellen (Atomizität der ACID Eigenschaft).

In dem Repository verwaltet der TP Monitor Benutzerdaten und -rechte, Screens, Datenbanktabellen, Anwendungen sowie zurückliegende Versionen von Anwendungen und Prozeduren.



Backward Recovery

Der Transaktionsmonitor stellt sicher, daß im Fehlerfall der teilweise Ablauf einer Transaktion rückgängig gemacht wird, und daß alle abgeänderten Felder einer Datenbank wieder in ihren ursprünglichen Zustand zurückbersetzt werden.



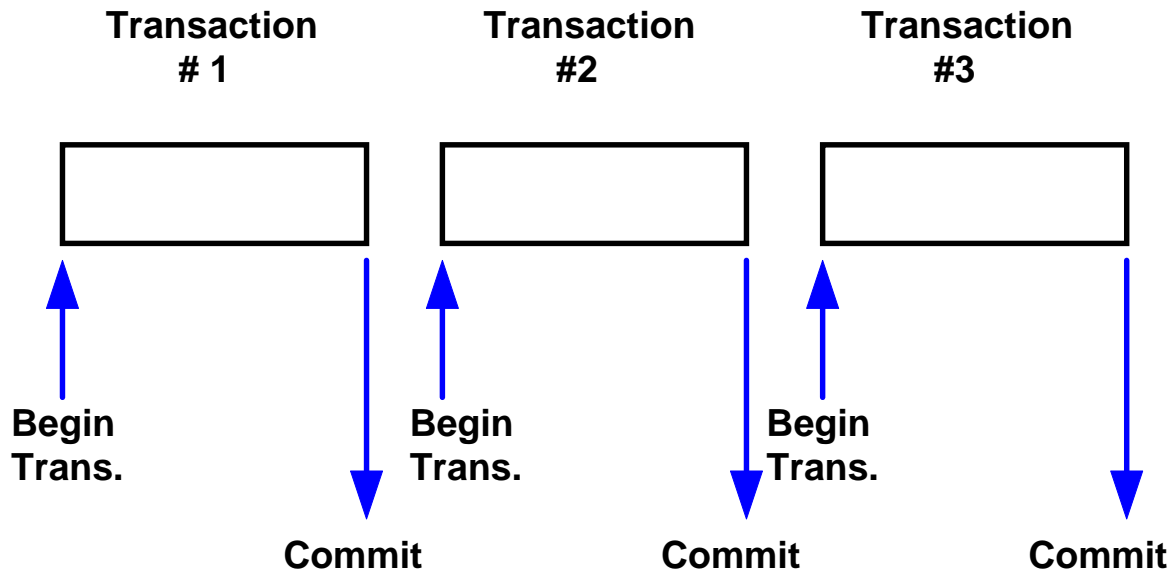
Backward Recovery

Andere Bezeichnungen: backout, roll back abort

1. abzuändernde Information in Puffer zwischenspeichern
2. Datensätze überschreiben
3. Commit Status festhalten. Damit ist es geschehen
4. erfolgreiches Commit dem Benutzer mitteilen
5. Zwischenpuffer löschen

Flat Transaction

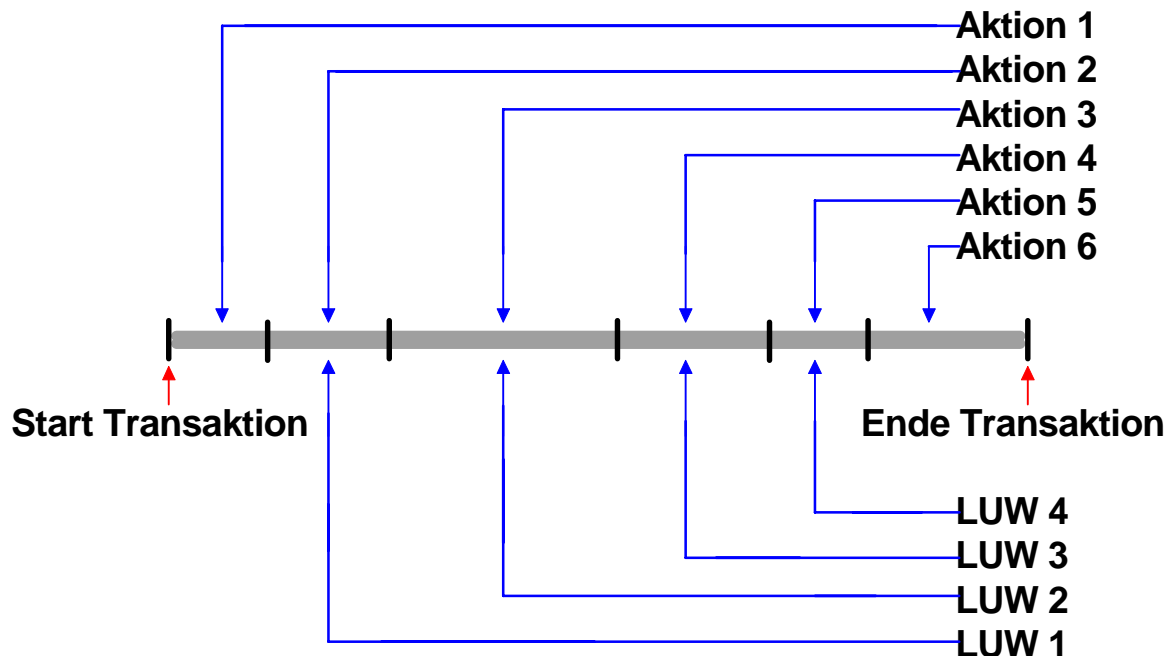
```
start_transaction {  
    beginwork ( );  
    .....  
    .....  
    .....  
    .....  
    .....  
    .....  
    .....  
    .....  
    .....  
    if no_error commit ( ) else rollback ( ) ;  
}
```



Flat Transaction

Alle Arbeit innerhalb einer Flat Transaction findet auf der gleichen Ebene statt. Die Transaktion überlebt entweder mit allen Teilfunktionen (commit) oder es erfolgt ein rollback einschließlich aller Teilfunktion (abort).

Logical Unit of Work LUW



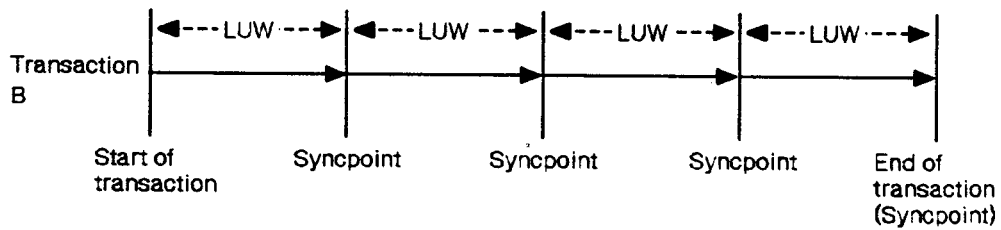
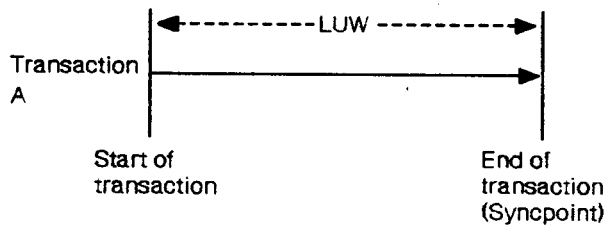
Aktionen sind Bausteine, aus denen der (zeitliche) Arbeitsablauf eines Resource Managers (Server) besteht.

Nicht geschützte (unprotected) Aktionen haben keine ACID Eigenschaften.

Geschützte (protected) Aktionen haben ACID Eigenschaften und werden als „Logical Unit of Work“ (LUW) bezeichnet.

Teilweise abgeschlossene LUW's können rückgängig gemacht werden.

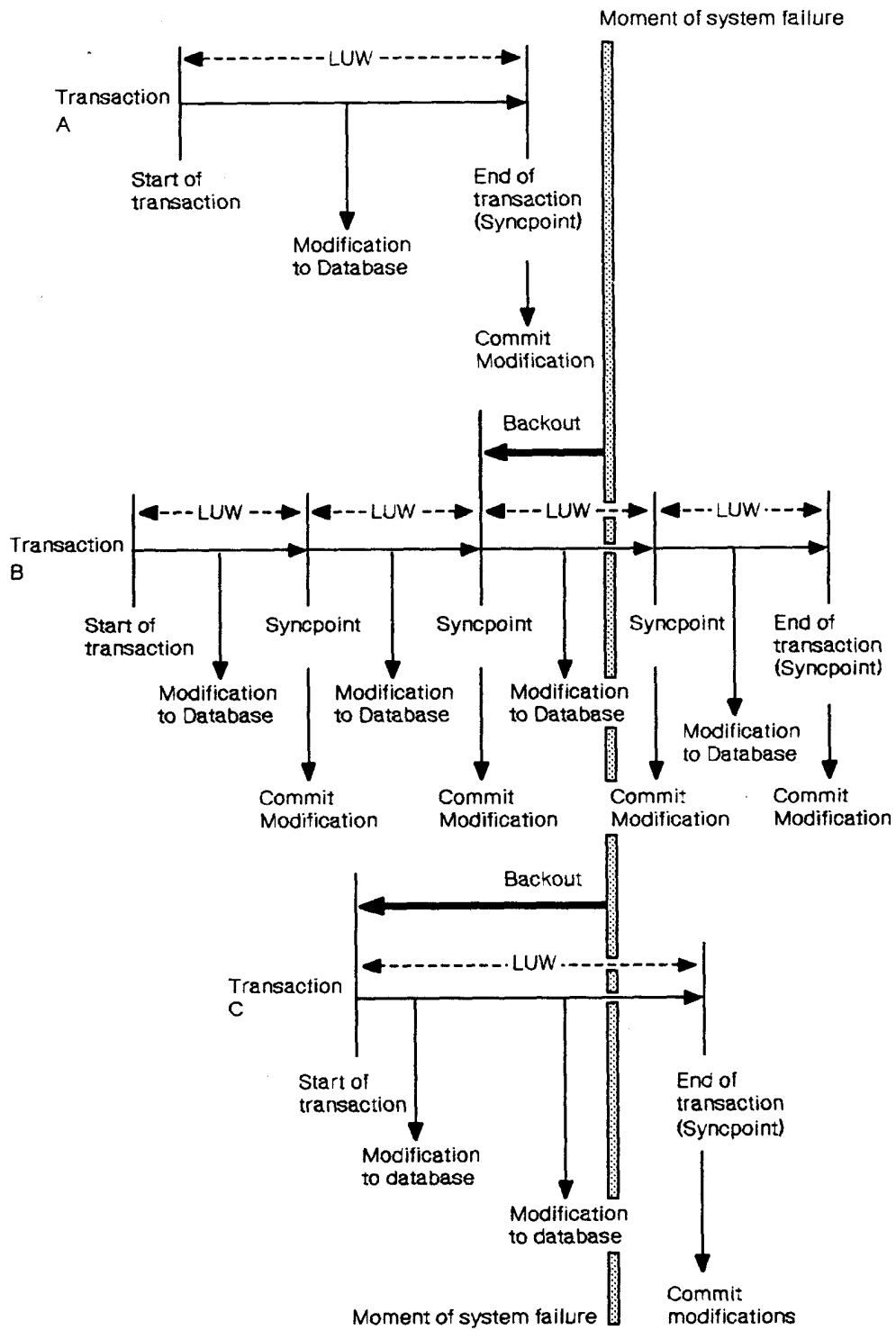
Reale (real) Aktionen beeinflussen die physikalische Umwelt auf eine Art, die nur schwer oder garnicht rückgängig zu machen ist (roll back). Beispiele: Ein Loch bohren, Geldausgabe am Automaten, vertrauliche Daten an den falschen Empfänger senden. ACID Eigenschaften für reale Aktionen mögen schwierig oder unmöglich zu implementieren sein.



„Logical Units of Work „ (LUW) und Sync Punkte

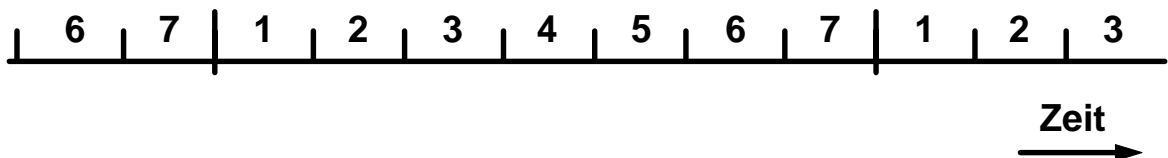
Andere Bezeichnungen: Aktionen und Save Points

Sync Punkte bewirken, daß ROLLBACK WORK nur bis zu dem angegebenen sync Punkt zurücksetzt.



LUW Backout

1. Darlehenskonto abrechnen, Saldo um Tilgungsrate verändern
2. Tilgung und Zinsen im laufenden Konto (Kontokorrent) auf der Sollseite buchen
3. Globales Limit überprüfen
4. Bilanzpositionen (Konten)
5. G+V Positionen (Gewinn- und Verlust Konten)
6. Zinsabgrenzung monatlich für jährliche Zinszahlung
7. Bankmeldewesen (ein Kunde nimmt je 90 000.- DM bei 10 Banken auf, läuft am Stichtag)

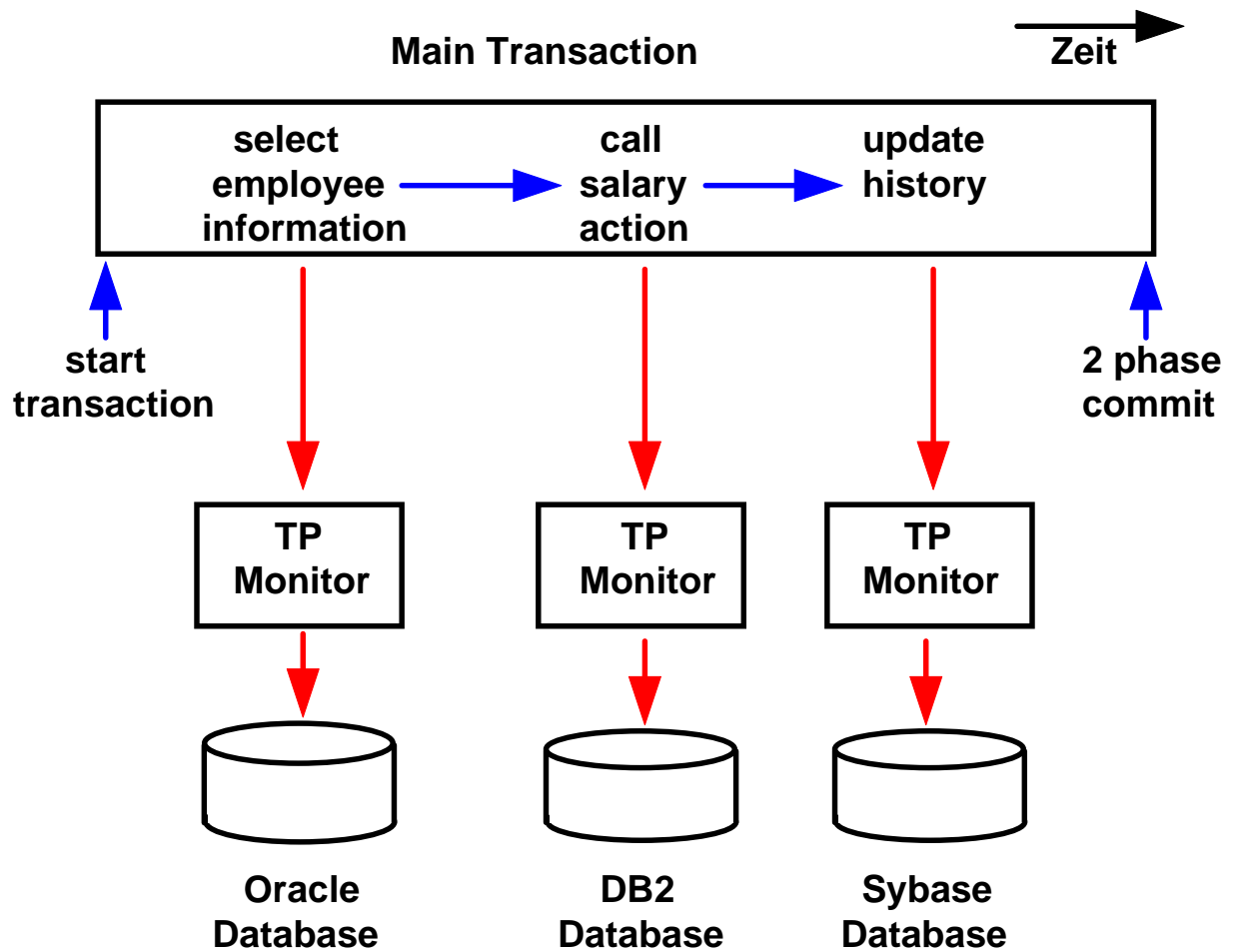


**Abarbeitung in Reihenfolge
evt. keine Sync Points erforderlich**

Buchungsvorgänge, monatl. Kreditabrechnung

1. Darlehenskonto abrechnen, Saldo um Tilgungsrate verändern
2. Tilgung und Zinsen im laufenden Konto (Kontokorrent) auf der Sollseite buchen
3. Globales Limit überprüfen
4. Bilanzposditionen (Konten)
5. G+V Positionen (Gewinn- und Verlust Konten)
6. Zinsabgrenzung monatlich für jährliche Zinszahlung
7. Bankmeldewesen (1 Kunde nimmt je 900 000.- DM bei 5 Banken auf, läuft am Stichtag)

Buchungs- Vorgang	1	2	3	4	5	6	7
Kunde Nr.							
1							
2							
3							
.							
.							
.							
900							



Distributed Flat Transaction

Zwei-Phasen-Festübergabe

Two-phase Commit

Erforderlich bei der gleichzeitigen Änderung mehrerer Datenbanken, z.B. Banküberweisung von Bank A nach Bank B

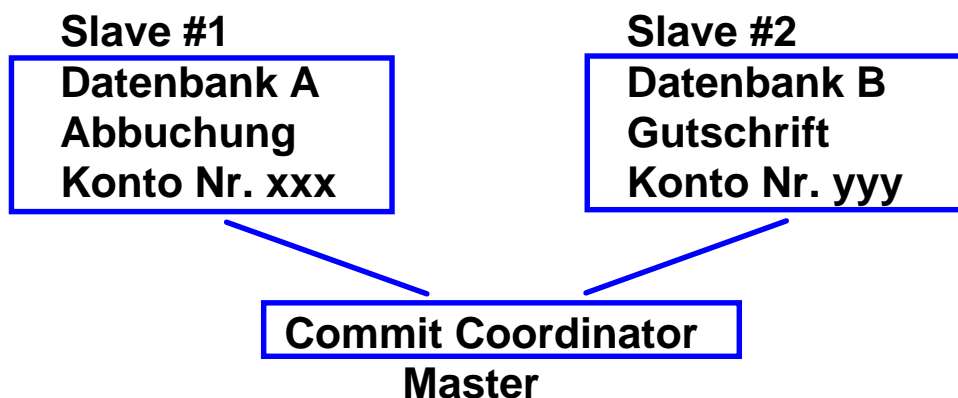
Elektronisches Clearinghaus sendet Nachrichten an beide Banken

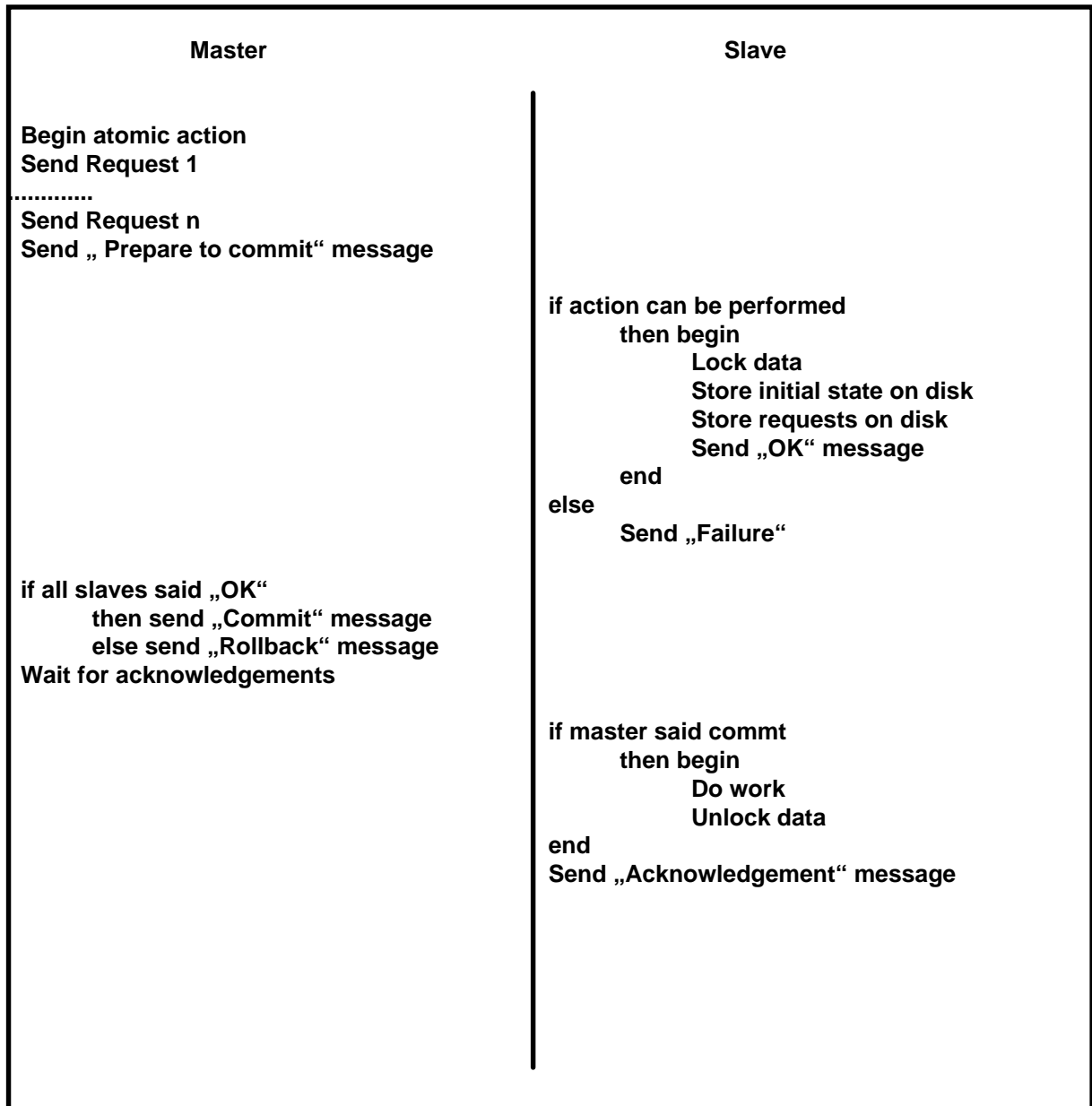
Problem, wenn entweder Abbuchung oder Gutschrift nicht erfolgt

Daher atomare Transaktionen erforderlich

Konsistenz wird erreicht durch einen „Master“ (Commit-Koordinator), der die Arbeit von „Slaves“ überwacht

Clearinghaus übernimmt Rolle des Master, die beiden Banken sind Slaves. Two-phase-Commit-Protokoll stellt sicher, daß Abbuchung und Gutschrift entweder beide erfolgen, oder beide nicht erfolgen





2-Phase Commit Protokoll