

Middleware kombiniert Funktionen wie RPC, MQ Series, CORBA und DCOM ORB`s, sowie Funktionen der Schicht 6 (Datenrepräsentation, Sicherheit, Verzeichnisdienste usw.).

Die entsprechenden Middleware Funktionen sind auf allen Klienten- und Server Rechnern installiert.

Middleware

Middleware-Funktionen

RPC

Message-Schnittstellen

Peer-to-Peer-Schnittstellen

Directory-Dienste

Zeitdienste

Sicherheit / Authentication

Verschlüsselung

Datensyntaxdienste

Verteilte Dateisysteme (NFS, AFS, DFS)

Groupware

Netzwerkmanagement

Beispiele für Middleware Implementierungen

Transaktionsmonitore

CICS, Encina, Tuxedo
ERP Systeme: SAP R/3, Baan, ...

Entwicklungsumgebungen

DCE
CORBA
Enterprise Java Beans (EJB)
MS DCOM
PeopleSoft

Web Application Server (z.B. IBM WebSphere)

Groupware

Lotus Notes
MS Exchange

e-Business

Supply Chain Management (SCM)
Customer Relations Management (CRM)
Electronic Commerce
Data Warehouse

System Management

HP OpenView
Tivoli
Computer Associates TNG

Middleware

In vielen Unternehmen besteht die IT Infrastruktur aus unterschiedlichen Hardware- und Betriebssystemplattformen, unterschiedlichen Datenbanken und File Systemen, unbeweglichen existierenden Anwendungen die mit neuen Anwendungen koordiniert werden müssen und einer Mischung von LAN und WAN Protokollen.

Middleware ist eine Infrastruktur für verteilte Anwendungen. Middleware existiert als eine Zwischenschicht zwischen Anwendungen und dem Betriebssystem mit seinen Netzwerkdiensten auf einem Rechner innerhalb eines Netzwerkes. Middleware verfolgt das Ziel, Entwicklungs- und Administrationskosten zu senken und Datenzugriffe zu vereinfachen. Middleware benutzt häufig standardisierte Dienstleistungen als Schnittstele zum Netz, z.B. RPC, CORBA ORB oder ASN.1.

Middleware ist in vielen Formen verfügbar. Das Distributed Computing Environment (DCE) der Open System Foundation stellt einen Versuch dar, Middleware zu standardisieren. Es ist gleichzeitig ein anschauliches Middleware Implementierungsbeispiel. Die wichtigsten Komponenten von DCE sind der DCE RPC, die DCE Directory Services, Threads, Kerberos für die Sicherheits- und Authentifizierungsdienste, Datenrepräsentationsdienste sowie Zeitdienste. Sie werden durch einheitliche Schnittstellen zusammengehalten und integriert. Während DCE bisher nicht sehr erfolgreich war, werden die DCE Komponenten RPC und Kerberos in größerem Umfang eingesetzt.

Datenbank Middleware und Transaction Processing (TP) Monitore sind zwei heute relativ weit verbreitete Middleware Produkte. CICS, Tuxedo, die SAP Grundstruktur, sowie Groupware Produkte wie Lotus Notes, Microsoft Exchange und Novell GroupWise sind Beispiele für anwendungsspezifische geschlossene Middleware, wobei die beiden letzteren gleichzeitig sog. Frameworks enthalten. CORBA ist auf dem Weg, in die Funktion eines offenen Middleware Standards hineinzuwachsen.

Derzeitig wird Middleware häufig innerhalb eines Unternehmens selbst programmiert. Wenn die Komplexität der Situation wächst wird es besser sein, ein kommerzielles Middleware Produkt einzusetzen. Es existiert eine große Anzahl derartiger Produkte, die alle das Ziel haben, die Erstellung von Client/Server Anwendungen (verteilten Anwendungen) zu erleichtern. Datenbank Middleware wird am häufigsten eingesetzt. Hier fehlen viele der funktionellen Eigenschaften im Vergleich zu komplexeren Programmpaketen; dafür sind die Kosten für Lizenzgebühren, Installation, Implementierung und Administration am geringsten. Das Ticket Service System der Deutschen Bundespost ist hierfür ein Beispiel.

Idealerweise würden Anwendungsentwickler einen einzigen Satz von standardisierten Middleware Diensten für verteilte Anwendungen benutzen. Gewünscht sind einheitliche APIs (to invoke services), und FAPs (formats and protocols for interoperation), die ein breites Spektrum von Entwicklungs- und Laufzeitfunktionen abdecken. In der Praxis ist dies mehr eine Zielvorstellung als Realität. Die CORBA Standards dürften auf diesem Weg am weitesten fortgeschritten sein.

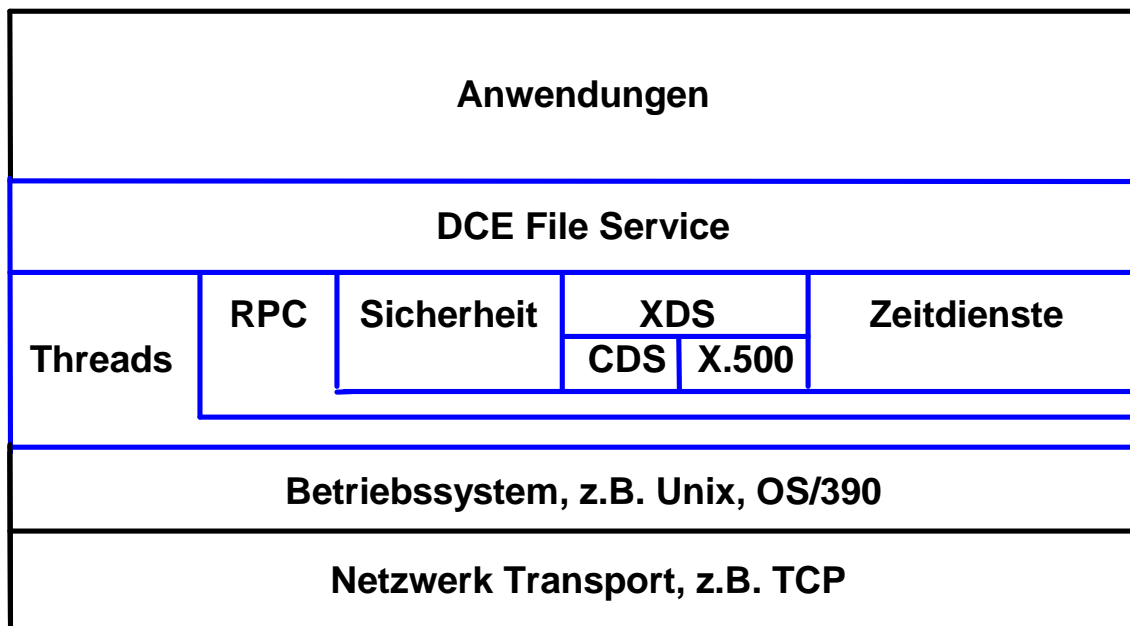
DCE

Distributed Computing Environment

Integrierter Satz an Werkzeugen, Einrichtungen und Diensten

Ermöglicht die Kooperation zwischen unterschiedlichen Rechnern

Schirmt den Benutzer von den technischen Details der Schichten 1-4 ab



Anwendungs-Programmierschnittstellen

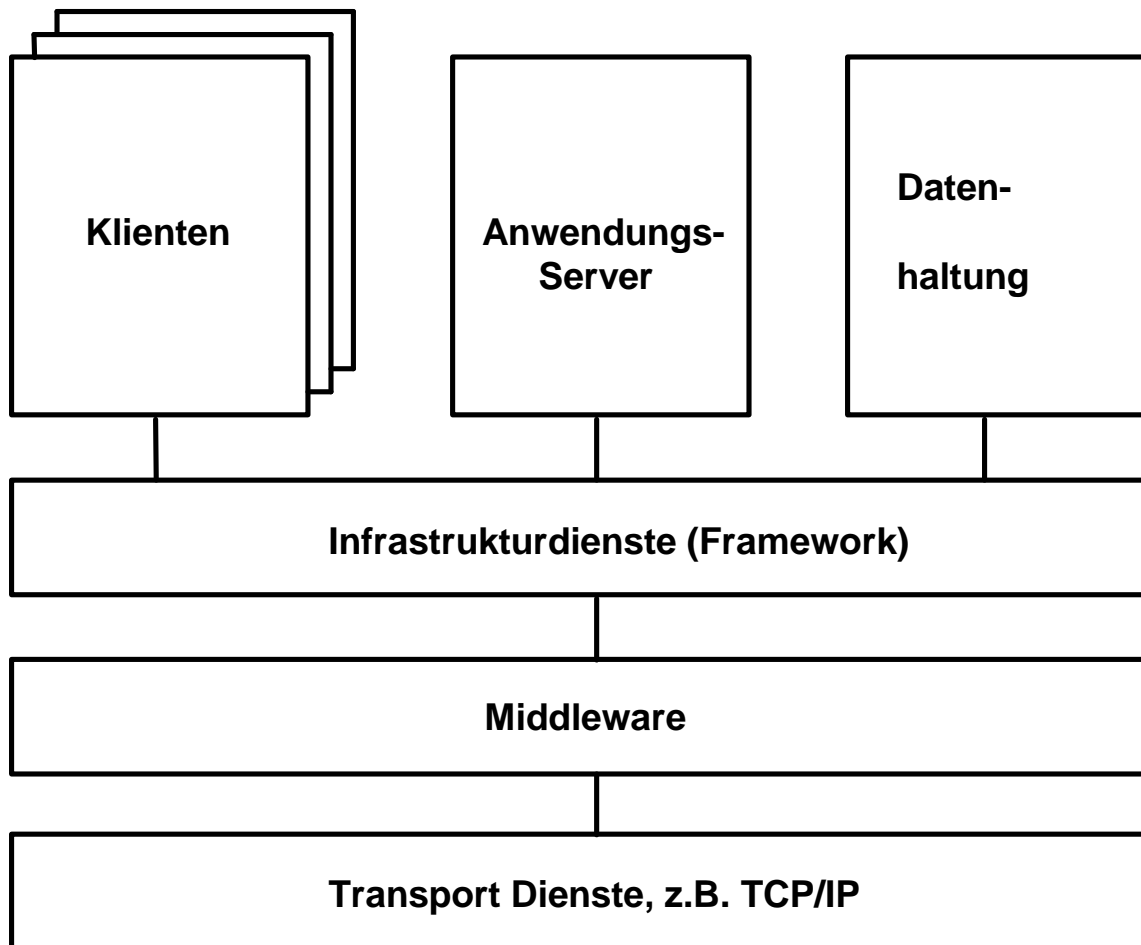
DCE Remote Procedure Call

Kann konfiguriert werden, die Host Computer Sicherheits-einrichtungen zu benutzen

Unterstützt verbindungslose und verbindungsorientierte Protokolle

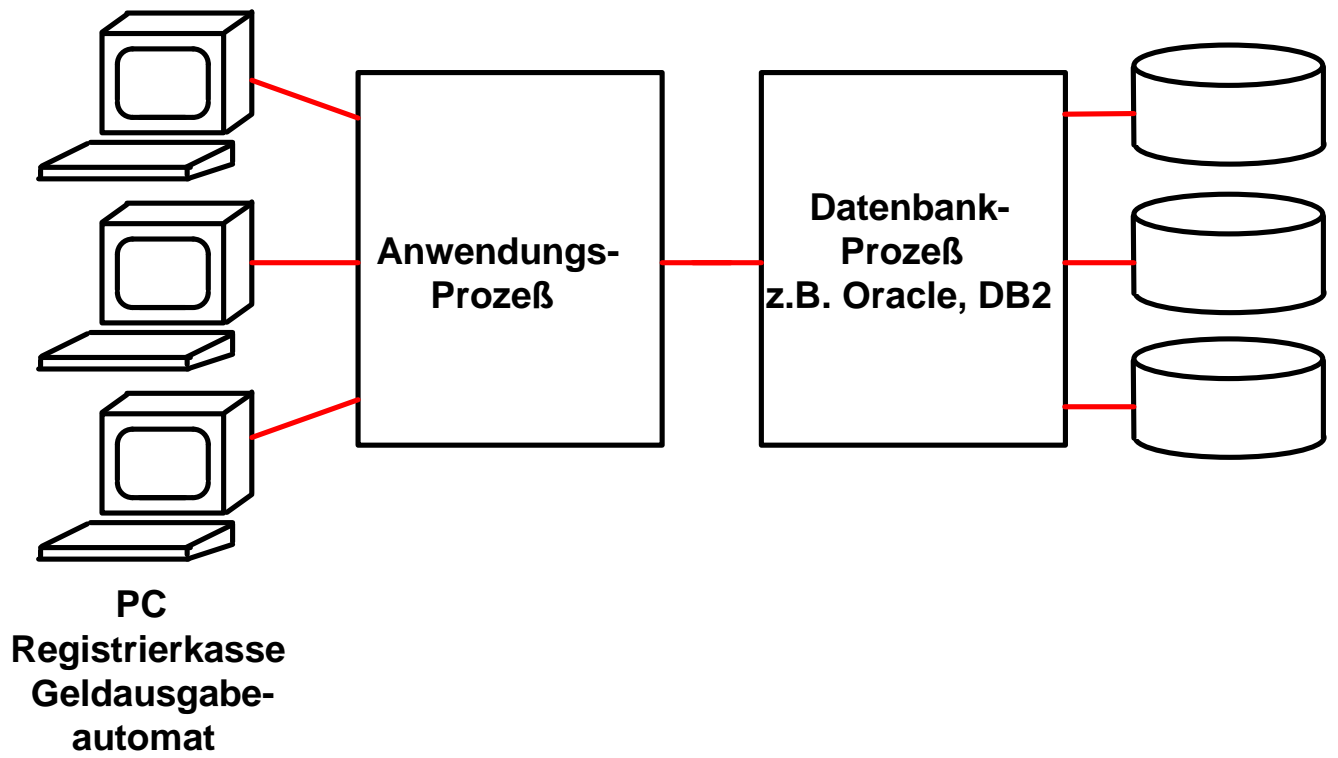
20 unterschiedliche Nachrichten Typen definiert, davon einige nur für den verbindungslosen oder den verbindungsorientierten Transport. Die Nachrichten werden als Protocol Data Units (PDU's) bezeichnet

Der Microsoft RPC ist eine auf DCOM basierende objektorientierte Erweiterung des DCE RPC (auch als „Objekt RPC“ bezeichnet. Die Syntax der übertragenen Pakete ist nahezu vollständig mit der ursprünglichen DCE Spezifikation kompatibel. Wird als Bestandteil von NT ausgeliefert.

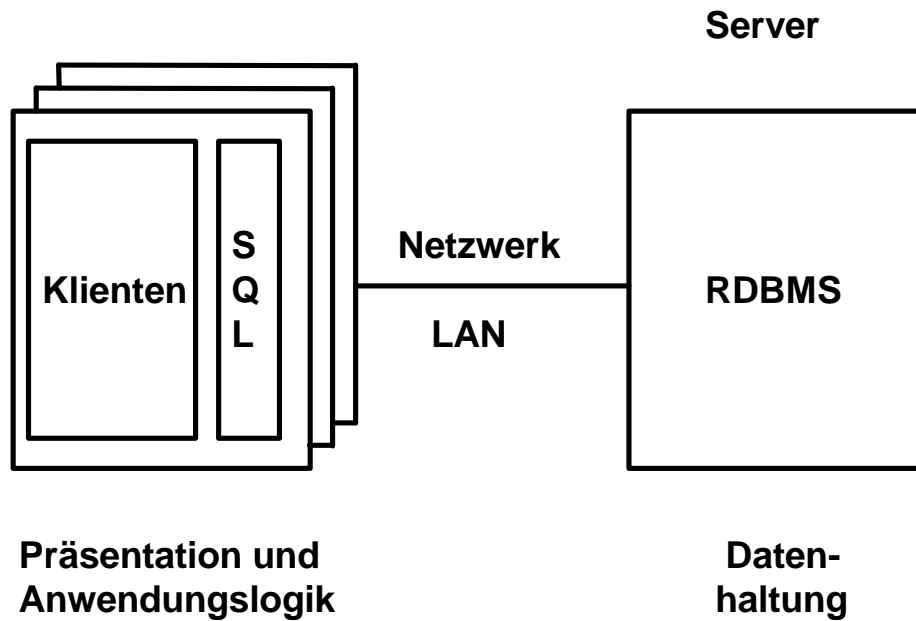


Framework

- Policy Regimes
- Profile Manager
- Tasks
- Scheduling Funktion
- Security Einrichtungen
- Rollback/Recovery
- Administration



Typische Client/Server Anwendung



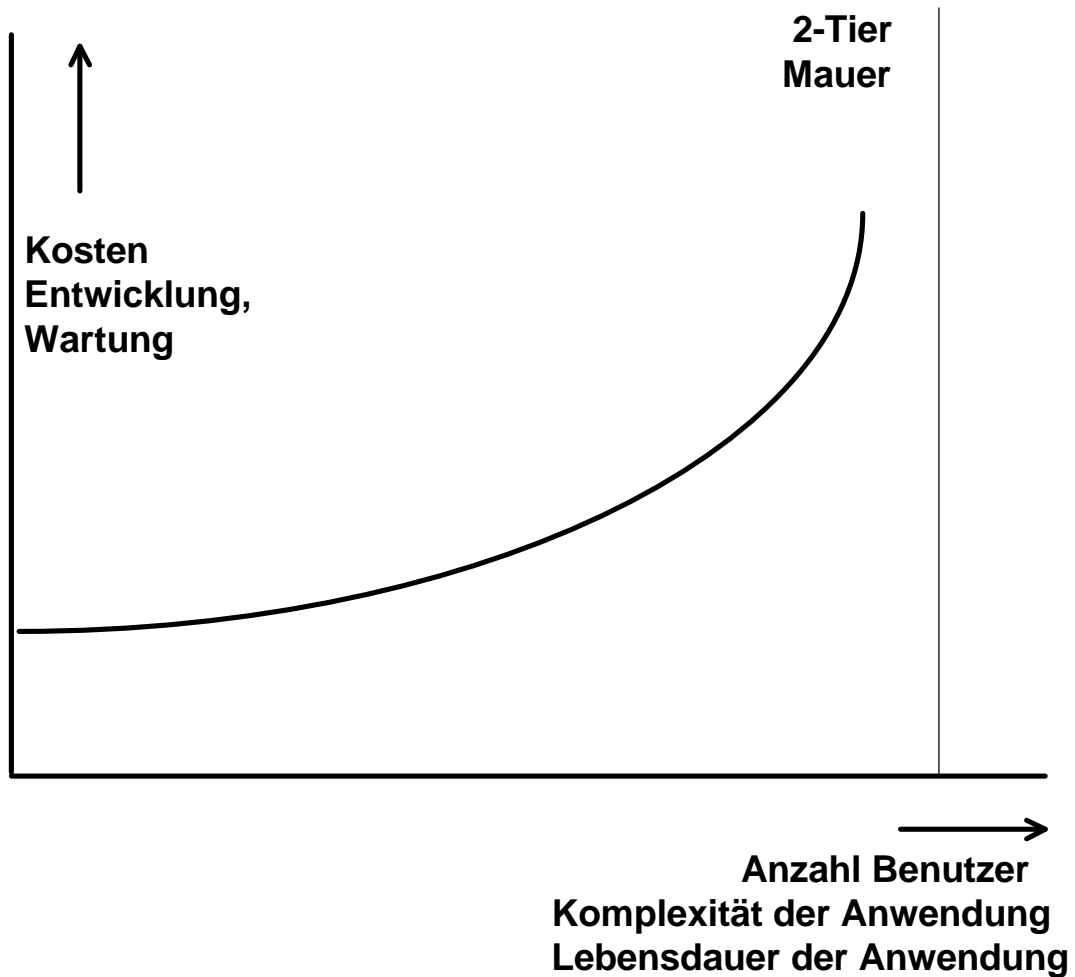
Annahmen:

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

2-Tier Client/Server Architektur

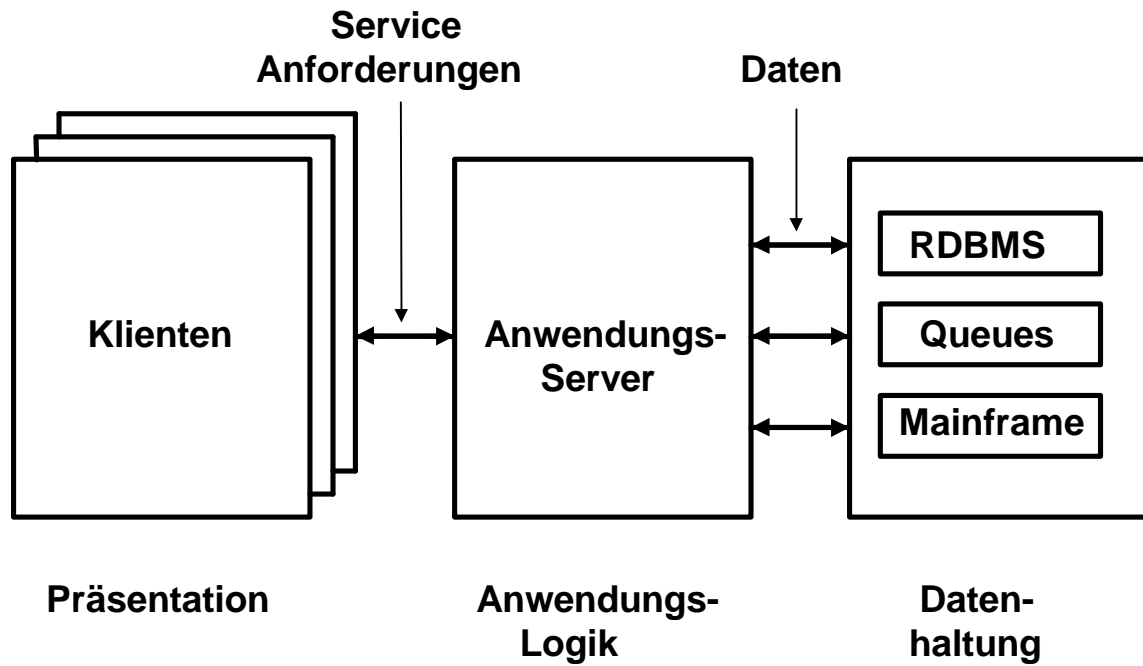
Zweistufige Client/Server Architektur

Anwendungsentwicklung mit Power Builder oder Visual Basic



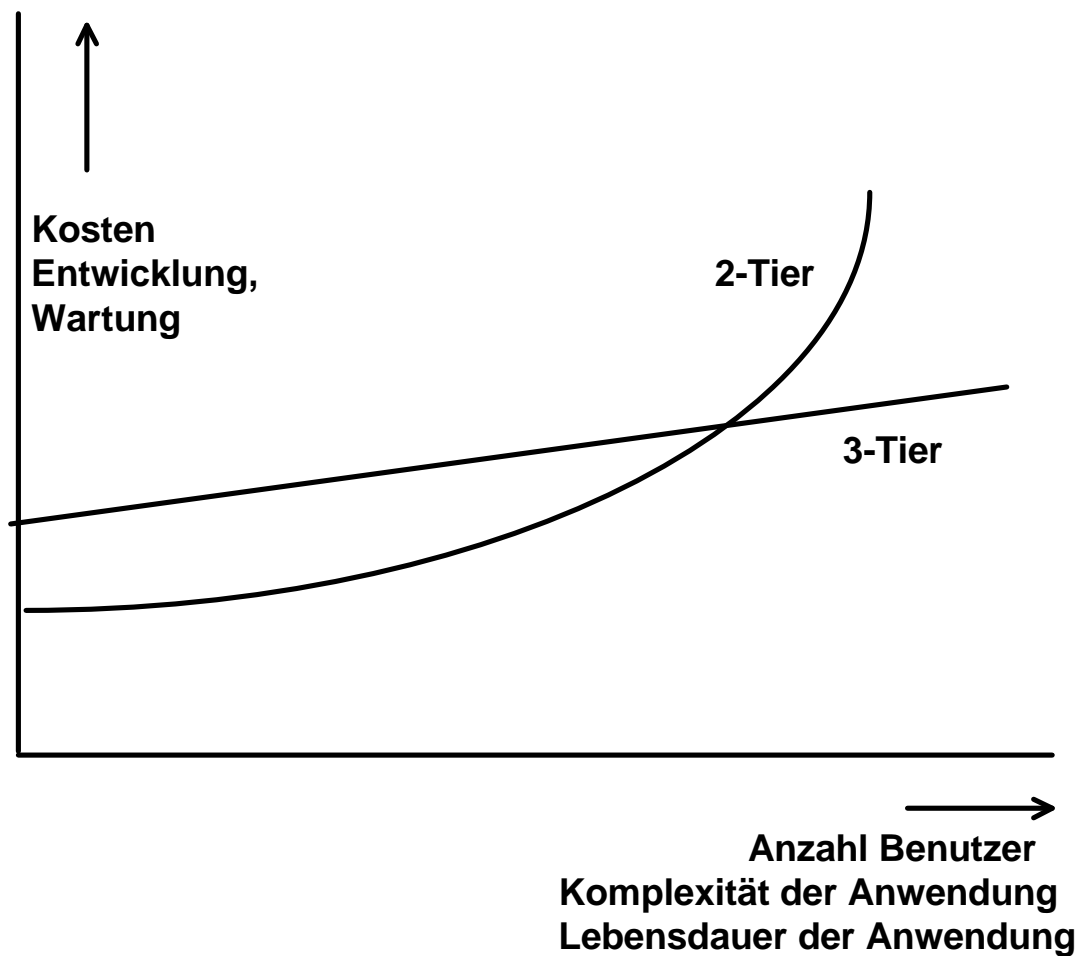
Skalierung der 2-Tier Architektur

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).



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

Transaktionen

Transaktionen sind Client-Server-Anwendungen, welche die auf einem Server gespeicherten Daten von einem definierten Zustand in einen anderen überführen.

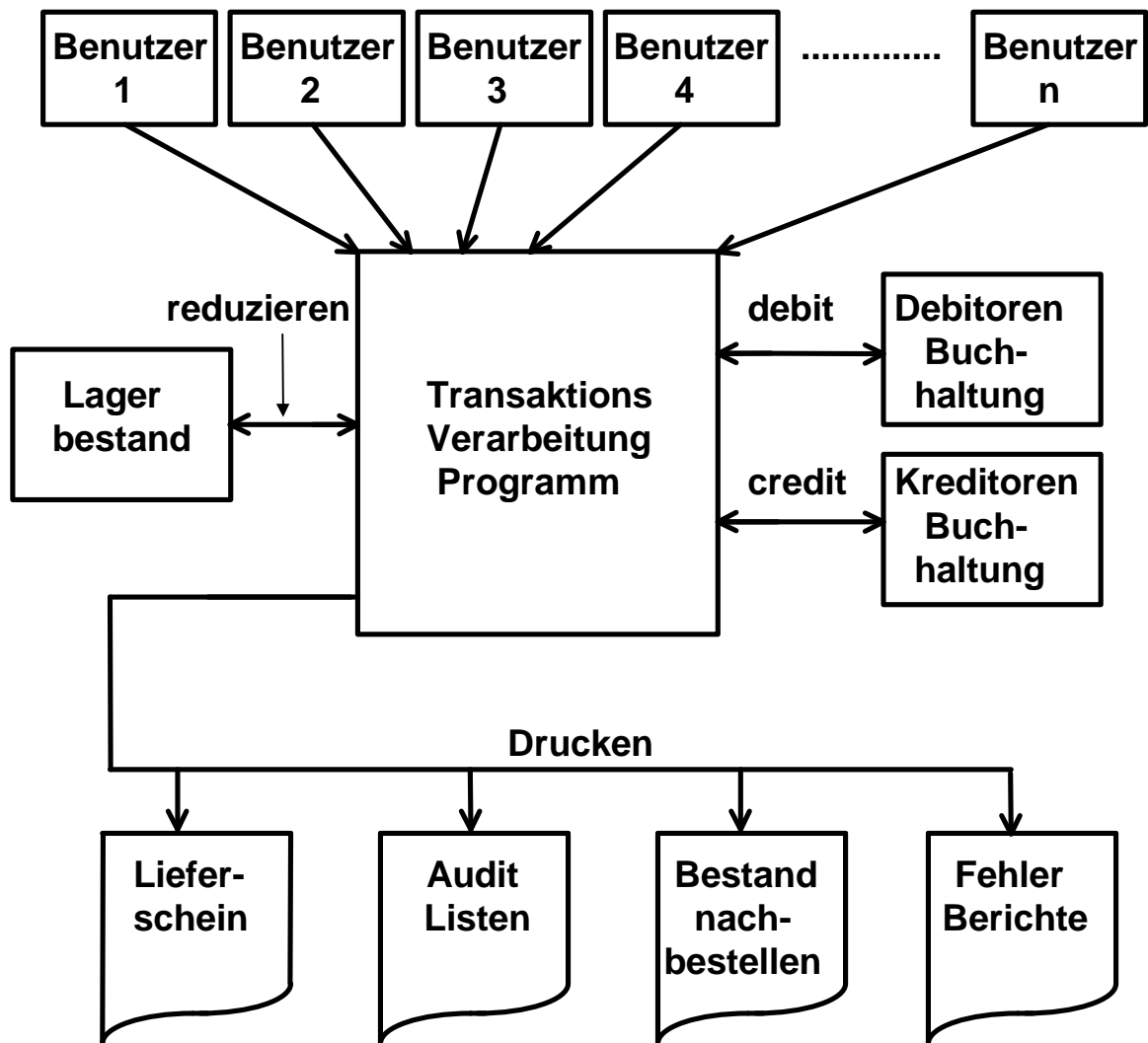
Eine Transaktion ist eine atomare Operation. Die Transaktion wird entweder ganz oder gar nicht durchgeführt.

Eine Transaktion ist die Zusammenfassung von mehreren Datei- oder Datenbankoperationen, die entweder erfolgreich abgeschlossen wird, oder die Datenbank unverändert läßt

Die Datei/Datenbank bleibt in einem konsistenten Zustand: Entweder vor Anfang oder nach Abschluß der Transaktion

Im Fehlerfall, oder bei einem Systemversagen werden alle in Arbeit befindlichen Transaktionen abgebrochen und alle evtl. bereits stattgefundenen Datenänderungen automatisch rückgängig gemacht.

Wird eine Transaktion abgebrochen, werden keine Daten abgeändert



**Beispiel für eine
Transaktionsverarbeitungsanwendung:
Auftragseingang Bearbeitung**

Transactionssysteme

Anwendungsbeispiele:

Auskunftssysteme

Buchungssysteme (z.B. Flugplatzreservierung)

Geldausgabeautomaten

Auftragsbearbeitung

Lagerbestandsverwaltung

ACID Eigenschaften

Atomizität (Atomicity)

Eine Transaktion wird entweder vollständig ausgeführt oder überhaupt nicht

Der Übergang vom Ursprungszustand zum Ergebniszustand erfolgt ohne erkennbare Zwischenzustände, unabhängig von Fehlern oder Crashes. Änderungen betreffen Datenbanken, Messages, Transducer und andere.

Konsistenzerhaltung (Consistency)

Eine Transaktion überführt das System von einem konsistenten Zustand in einen anderen konsistenten Zustand

Daten sind konsistent, wenn sie durch eine Transaktion erzeugt wurden. (Datenkonsistenz kann nicht zu Beginn einer Transaktion überprüft werden).

Isolation

Die Auswirkungen einer Transaktion werden erst nach ihrer erfolgreichen Beendigung für andere Transaktionen sichtbar

Single User Mode Modell. Selbst wenn 2 Transaktionen gleichzeitig ausgeführt werden, wird der Schein einer seriellen Verarbeitung gewahrt.

Dauerhaftigkeit (Durability)

Die Auswirkungen einer erfolgreich beendeten Transaktion gehen nicht verloren

Das Ergebnis einer Transaktion ist real, mit allen Konsequenzen. Es kann nur mit einer neuen Transaktion rückgängig gemacht werden. Die Zustandsänderung überlebt nachfolgende Fehler oder Systemcrashes.

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');
    .....
    .....
}
```

Flat Transaction

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

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

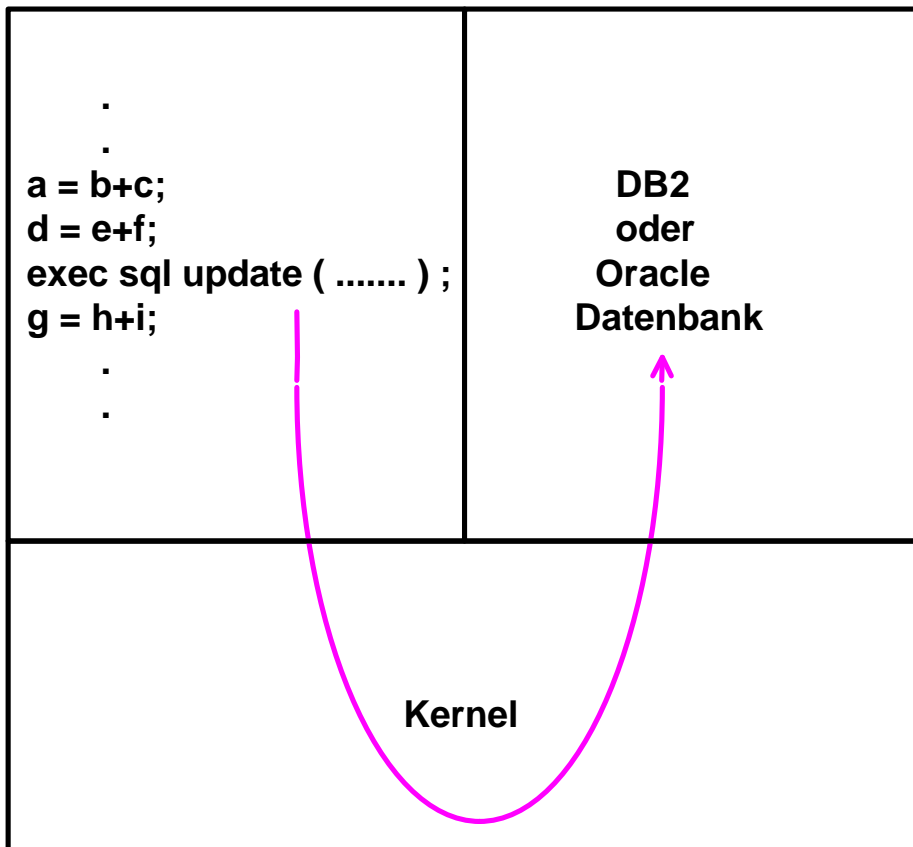
Der Oracle oder DB/2 Precompiler greift sich die exec sql Befehle heraus und übersetzt sie in Anweisungen, die der C-Compiler versteht.

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.)

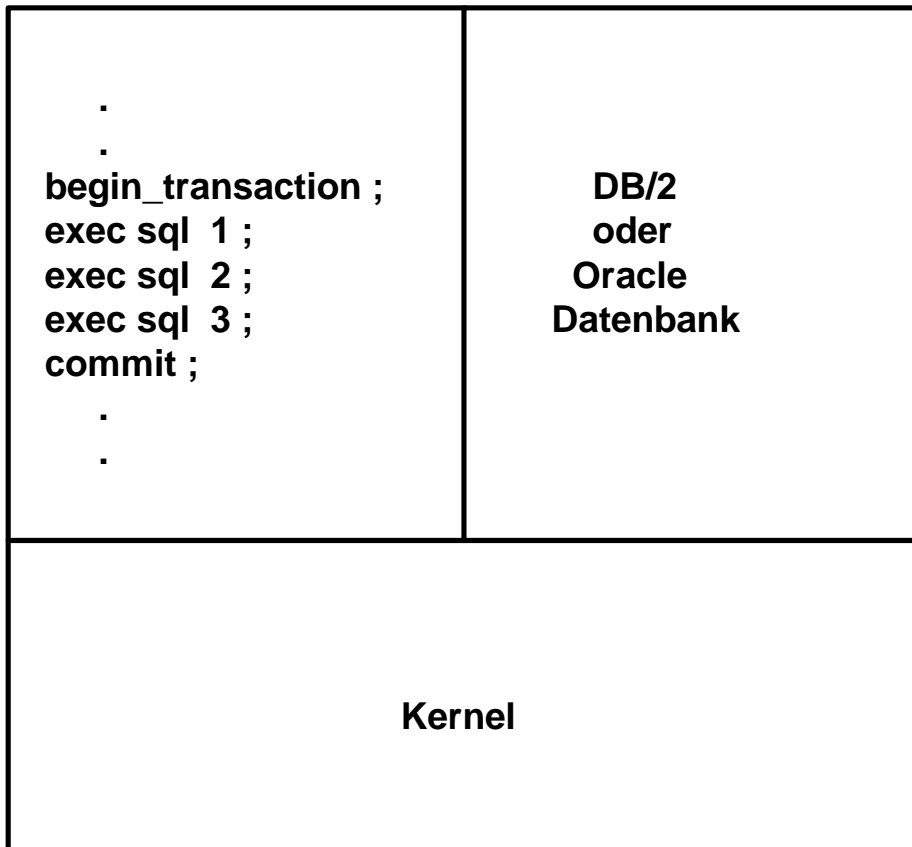
virtueller Adressraum i virtueller adressraum i+1



SQL Datenbank Zugriff auf dem gleichen Rechner

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

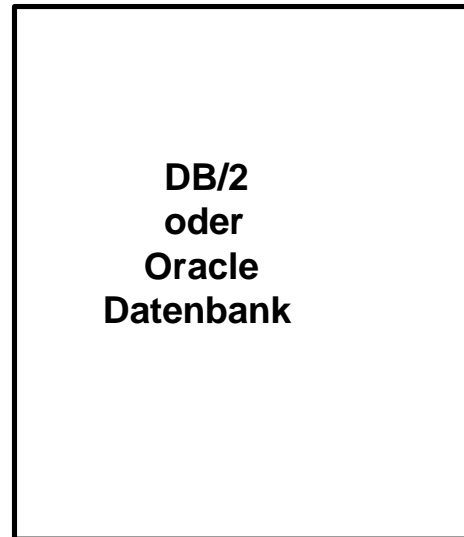
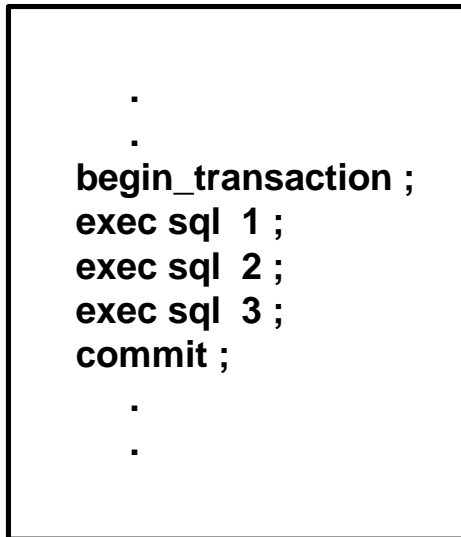
virtueller Adressraum i virtueller adressraum i+1



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

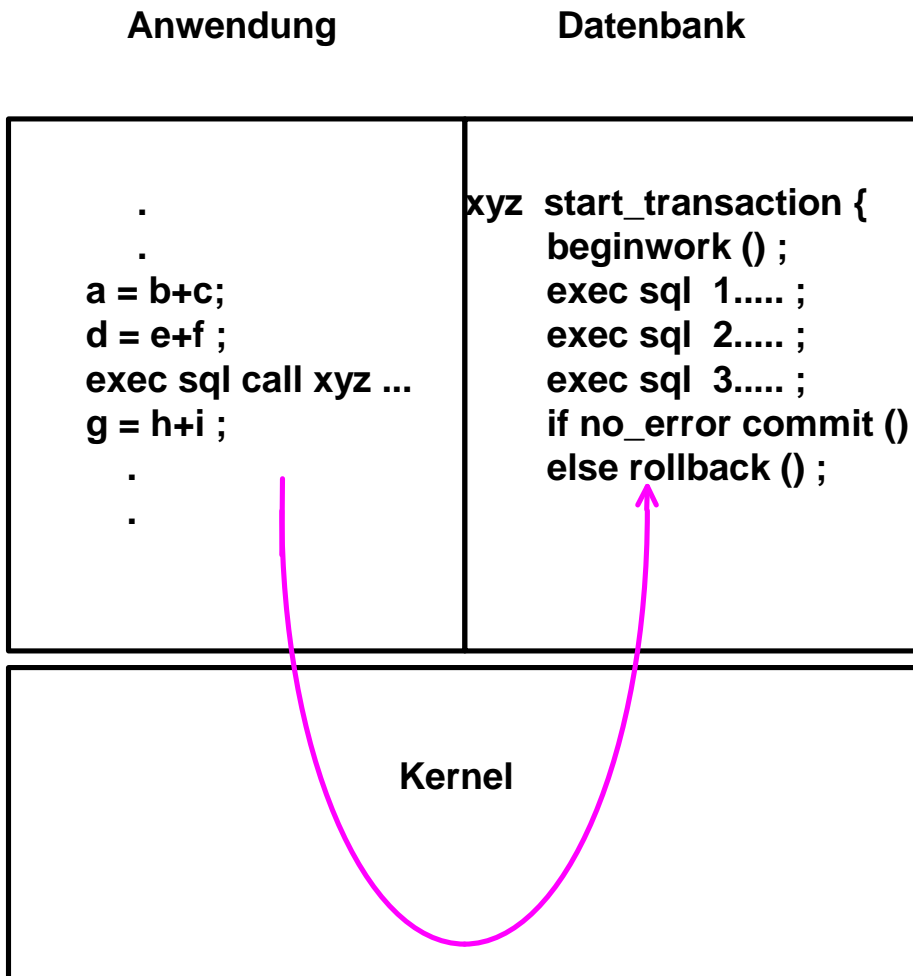
Rechner 1

Rechner 2



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

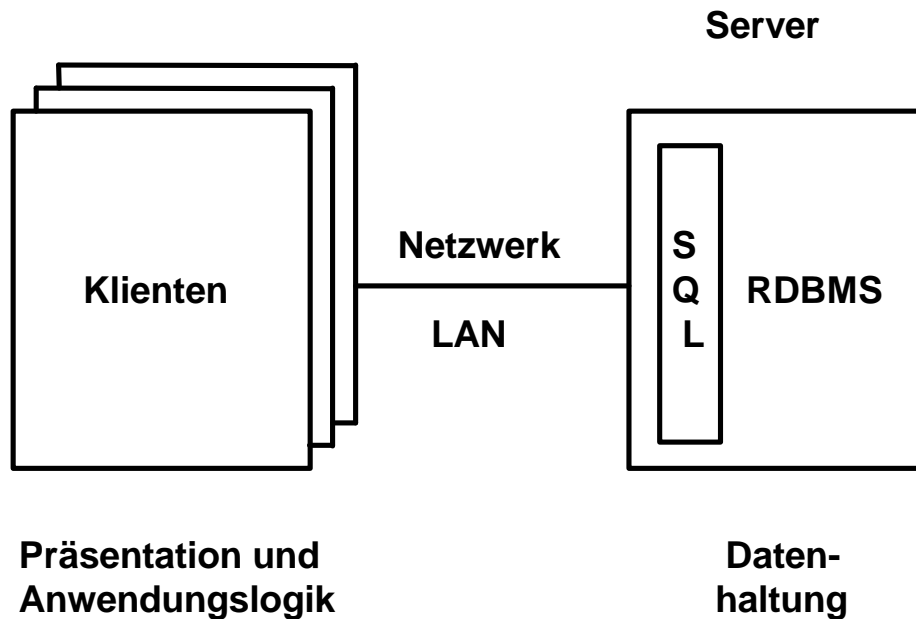
ACID Eigenschaften für eine Gruppe von SQL Aufrufen



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 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.

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.

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

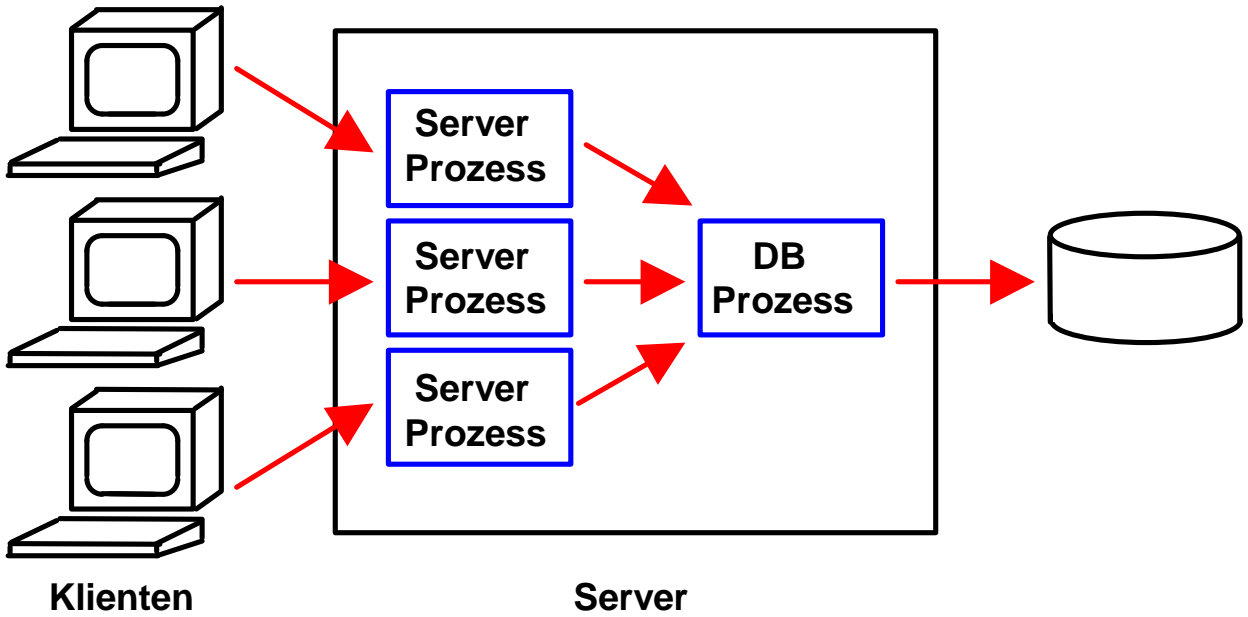
Daten verarbeiten

Ergebnis speichern

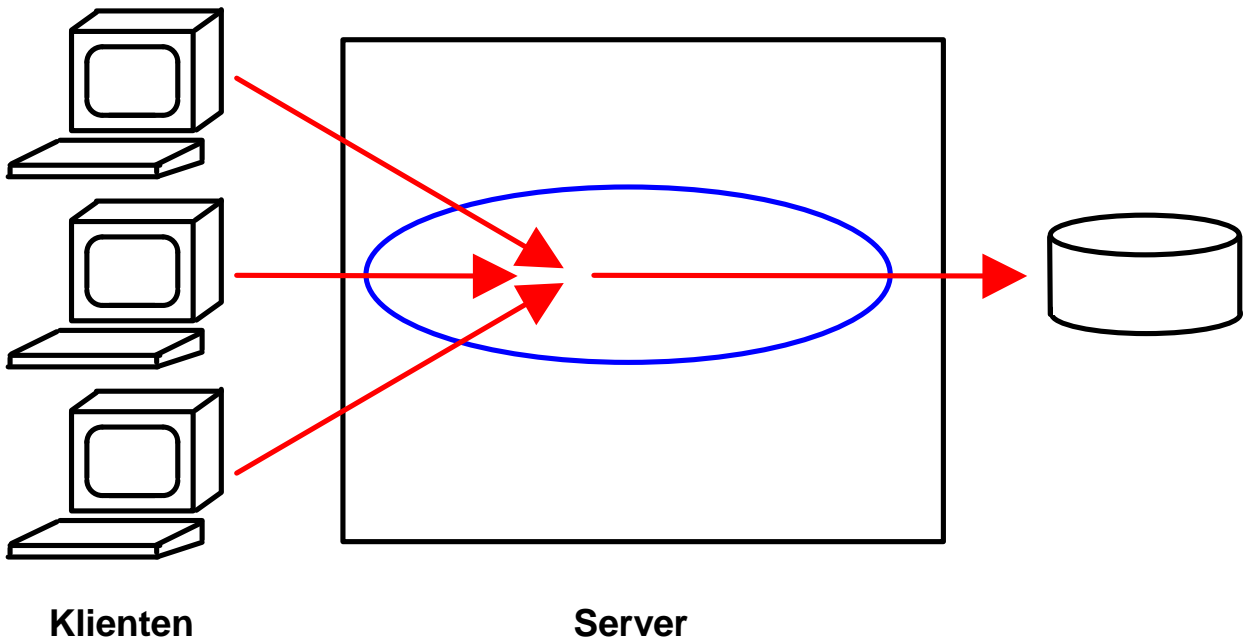
reset lock

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

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)**

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)

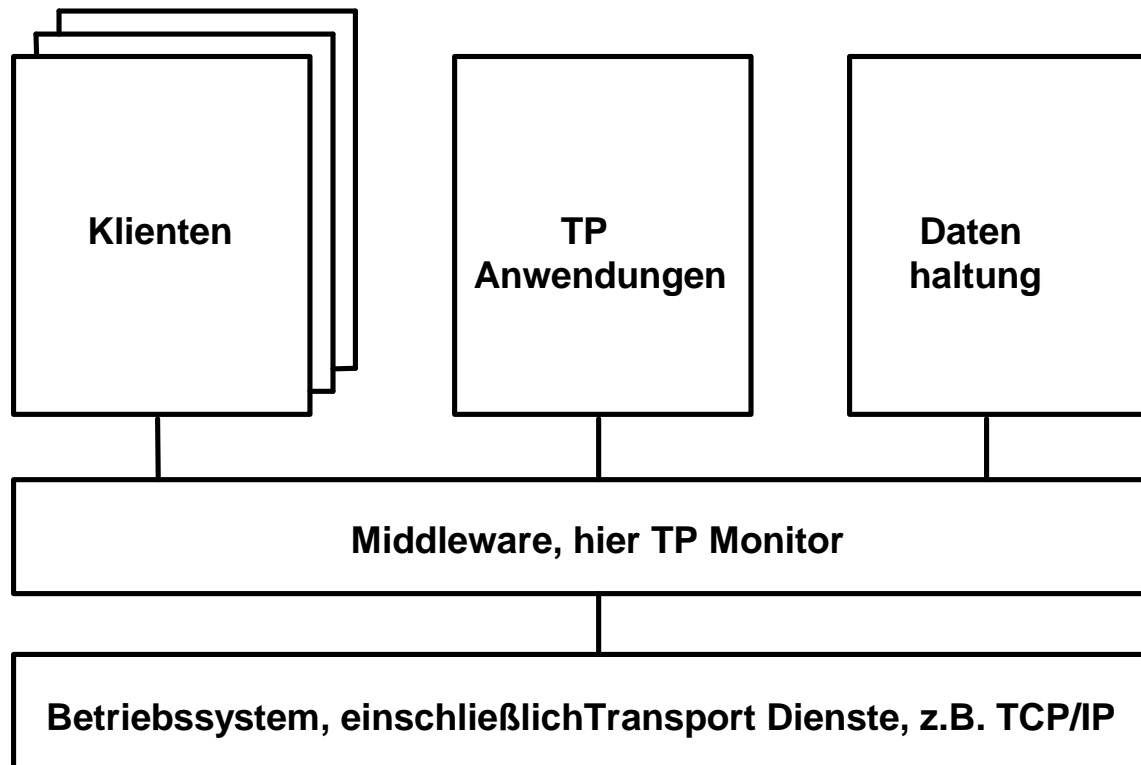
geeignet für hohes Verkehrsaufkommen

niedrige Kosten pro Transaktion

Integrität beim Zugriff auf gemeinsam genutzte

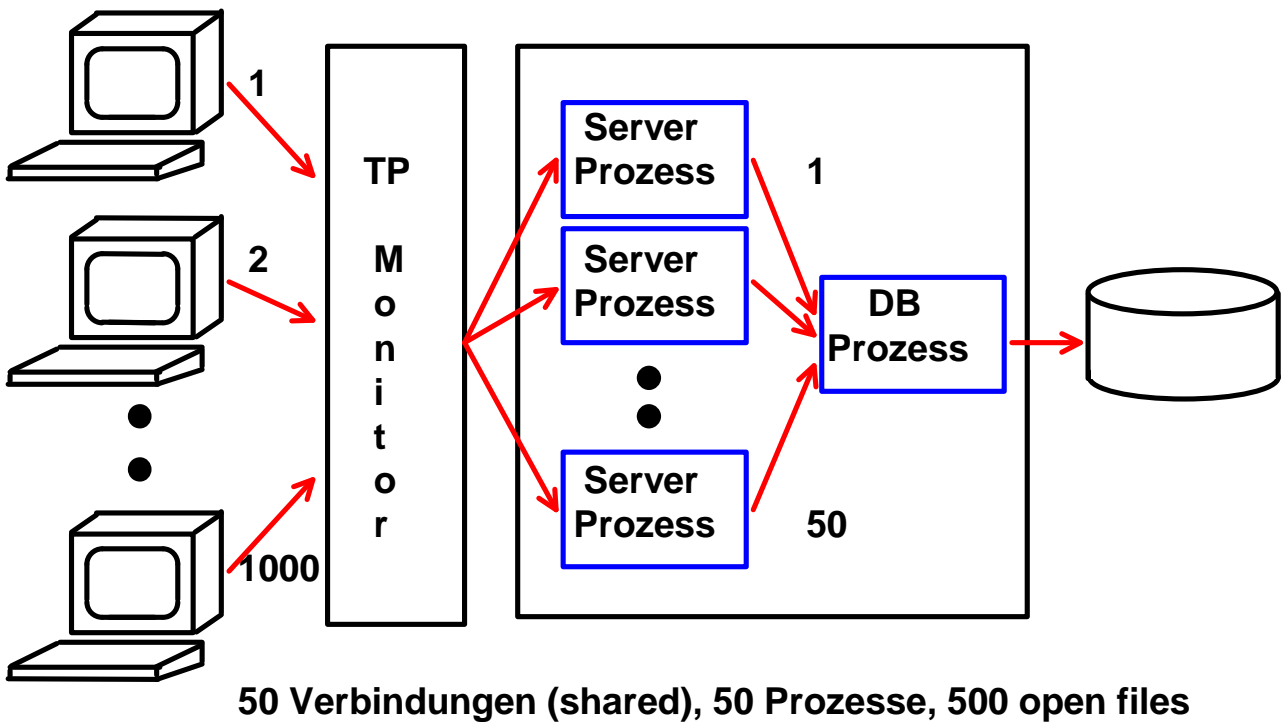
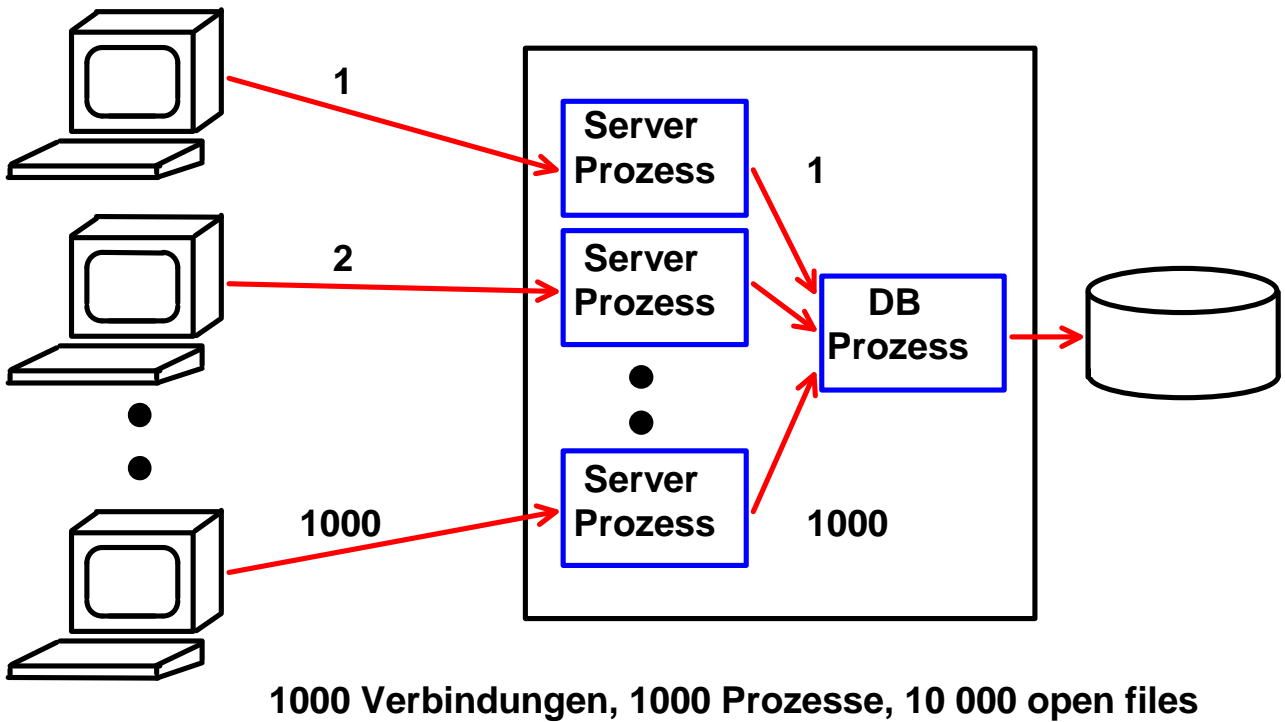
Ressourcen

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.



Transaktionsverarbeitung mit und ohne TP Monitor

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.

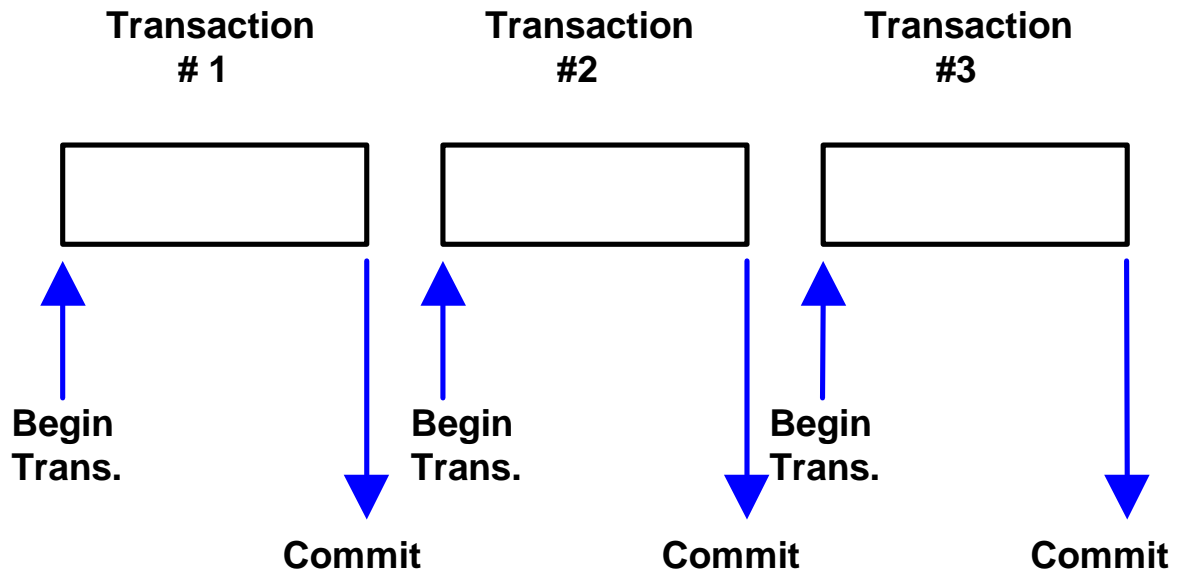
Der Scheduler verteilt eingehende Bearbeitungsanforderungen auf die einzelnen Server Prozesse.

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.

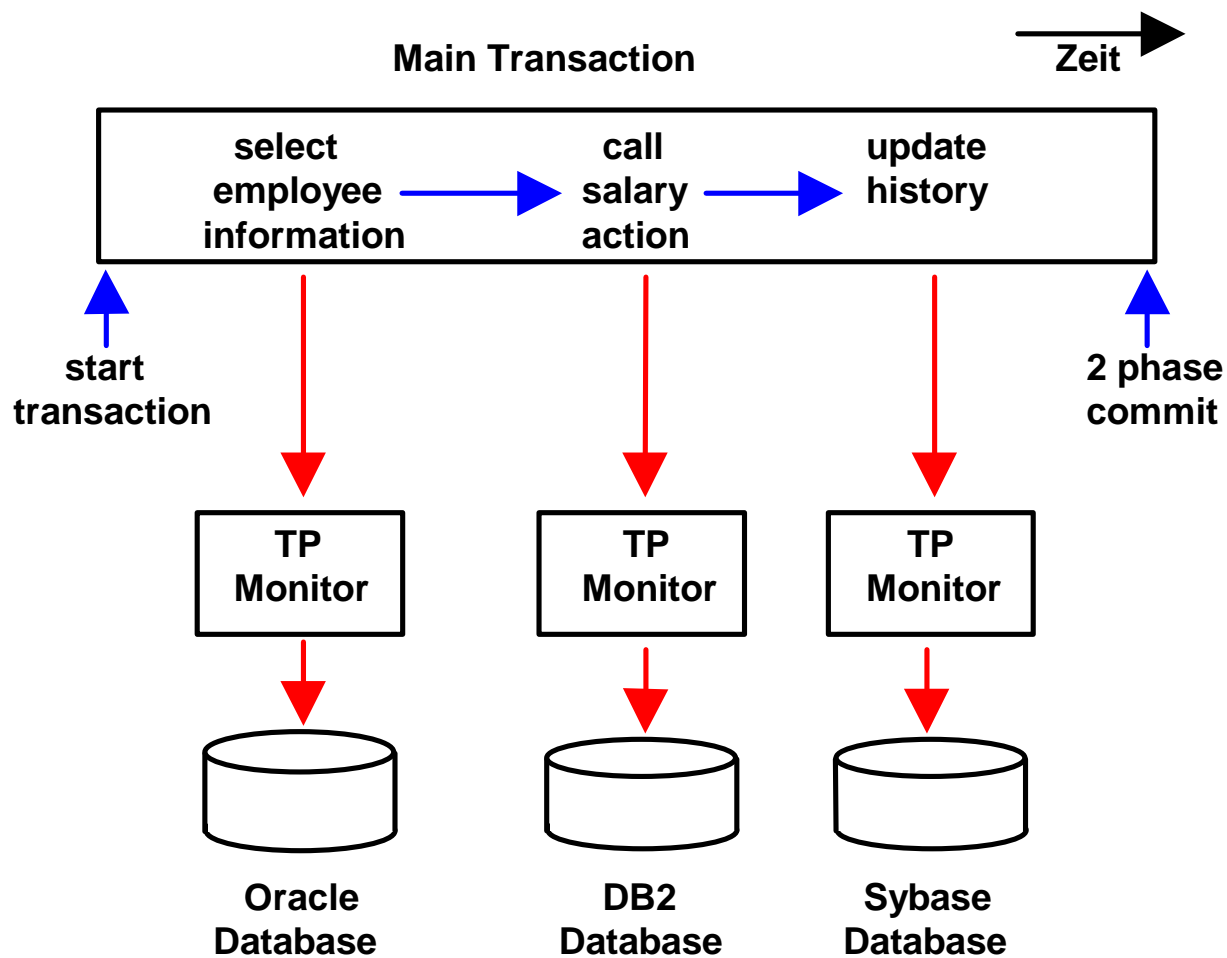
Der LOG Manager hält alle Änderungen gegen die Datenbank fest. Mit Hilfe der Log Datenbank kann der Recovery Manager im Fehlerfall den ursprünglichen Zustand der Datenbank wiederherstellen (Atomizität der ACID Eigenschaft).

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



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).



Distributed Flat Transaction

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

Endgeräte für die Transaktionsverarbeitung

Arbeitsplatzrechner

Windows NT GUI
Browser GUI (Java 1.2 Swing Classes)
Motiv
SAPGUI
3270 GUI

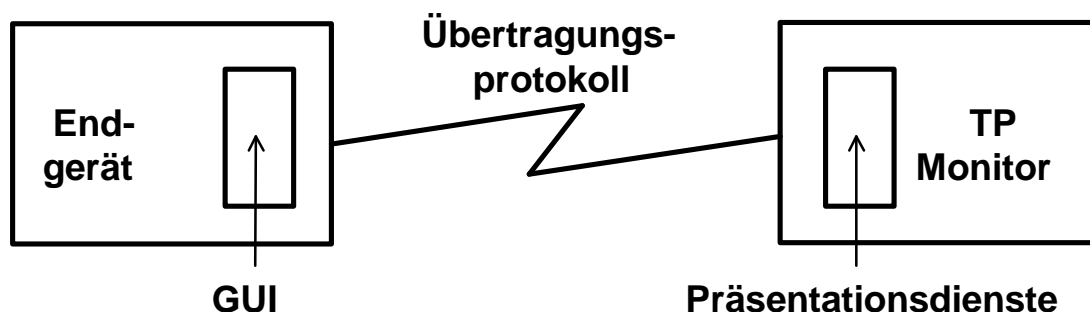
Hand Held Geräte, z.B. Palmtops, Mobiltelefon

Geldausgabeautomaten, Kontoauszugsdrucker

Supermarkt Registrierkasse, Tankstellen Zapfsäule

Produktionssteuerungselektronik

Die GUI ist ein Prozeß in den Endgeräten, welcher für die visuelle Ein/Ausgabe zuständig ist. Es ist die Aufgabe der Präsentationsdienste, Information von/zu den Endgeräte GUI's in geeigneter Form aufzubereiten.



Literatur

- R. Buck-Emden: „Die Client/Server Technologie des SAP System R/3“. Addison-Wesley 1996.**
- R.J. Cypser: „Communications for Cooperating Systems“, Addison-Wesley, 1991.**
- J. Gray, A. Reuter: „Transaction Processing, Morgan Kaufmann Publishers, 1993.**
- P. Heitlinger: „Netzwerk Management“, Thomson Publishing, 1995.**
- Högering/Abek: „Integriertes Netz- und System-Management“, Addison Wesley, Bonn.**
- H.W. Lockhart, Jr.: OSF DCE, McGraw-Hill, 1994.**
- R. Orfali, D. Harkey: Client/Server Programming with JAVA and CORBA, John Wiley & Sons, 1997.**
- L. Will: „Administration des SAP System R/3“. Addison-Wesley 1997.**