

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm Spruth

WS 2003/04

Teil 8

Stored Procedures

Literatur

J. Gray, A. Reuter:

**„Transaction Processing“.
Morgan Kaufmann, 1993.**

P. A. Bernstein:

**„Principles of Transaction Processing“.
Morgan Kaufmann, 1997.**

J. Horswill:

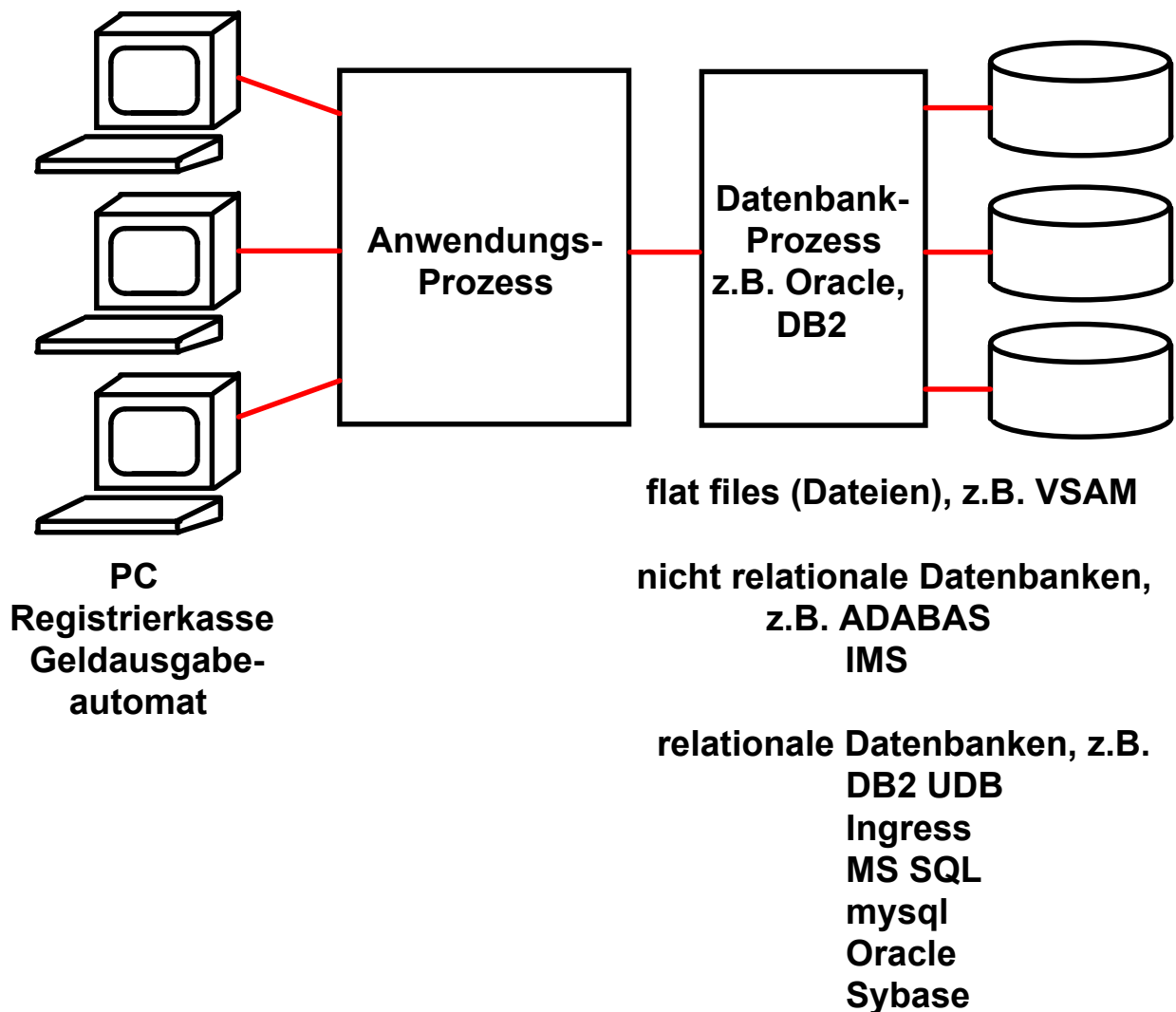
**„Designing and Programming CICS
Applications“. O'Reilly, 2000**

Datenbanken

Eine Datenbank besteht aus einer Datenbasis (normalerweise Plattenspeicher) und Verwaltungsprogrammen (Datenbank Software), welche die Daten entsprechend den vorgegebenen Beschreibungen abspeichern, auffinden oder weitere Operationen mit den Daten durchführen.

Wenn wir im Zusammenhang mit Client/Server Systemen von einer Datenbank sprechen meinen wir in der Regel damit die Verwaltungsprogramme (Datenbank im engeren Sinne).

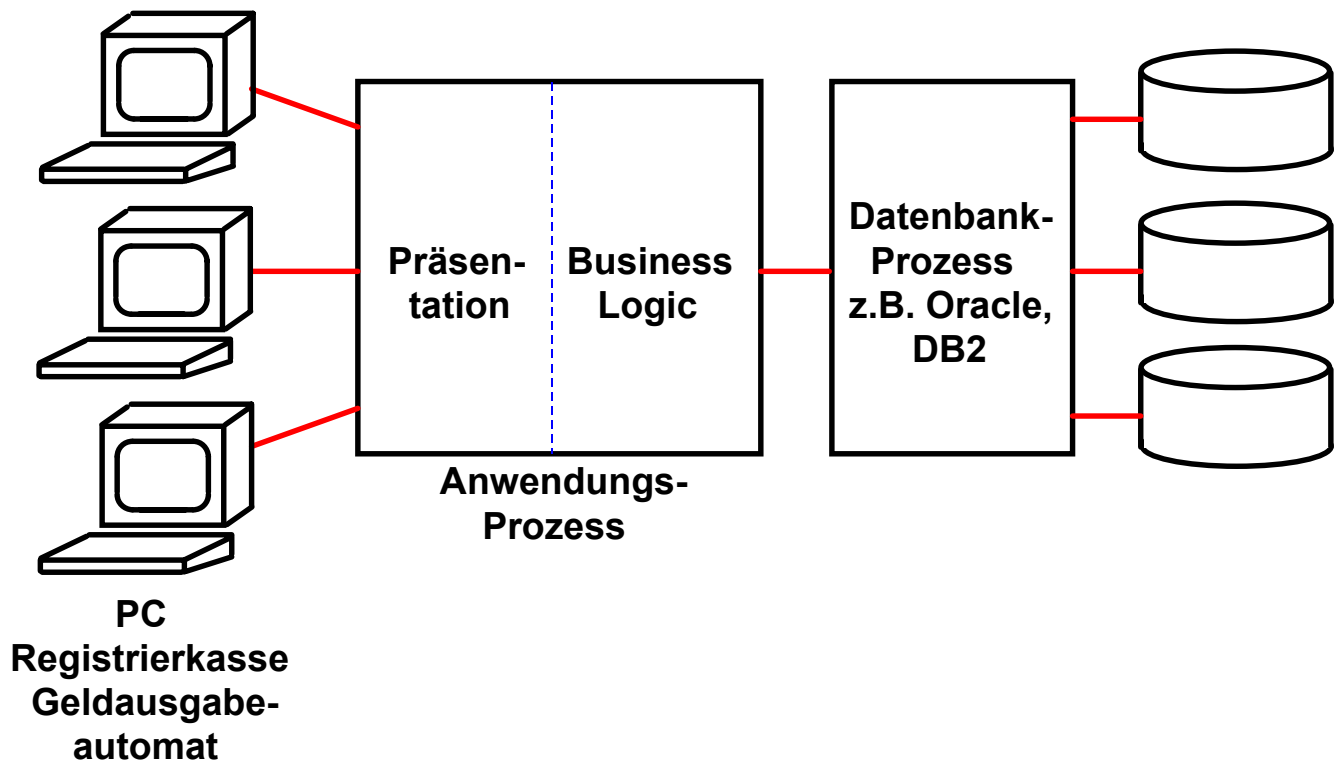
Ein Datenbankprozess läuft fast immer in einem eigenen virtuellen Adressenraum. Häufig sind es sogar mehrere virtuelle Adressenräume (z.B. drei im Fall von OS/390 DB2).



Datenbank Server in einer typische Client/Server Anwendung

Dargestellt ist eine logische Struktur. 2-Tier, 3-Tier oder n-Tier Konfigurationen unterscheiden sich dadurch, wie diese Funktionen auf physikalische Server abgebildet werden.

Im einfachsten Fall (Beispiel Diplomarbeit) sind Anwendung und Datenbank getrennte Prozesse auf dem gleichen Rechner wie der Klient.

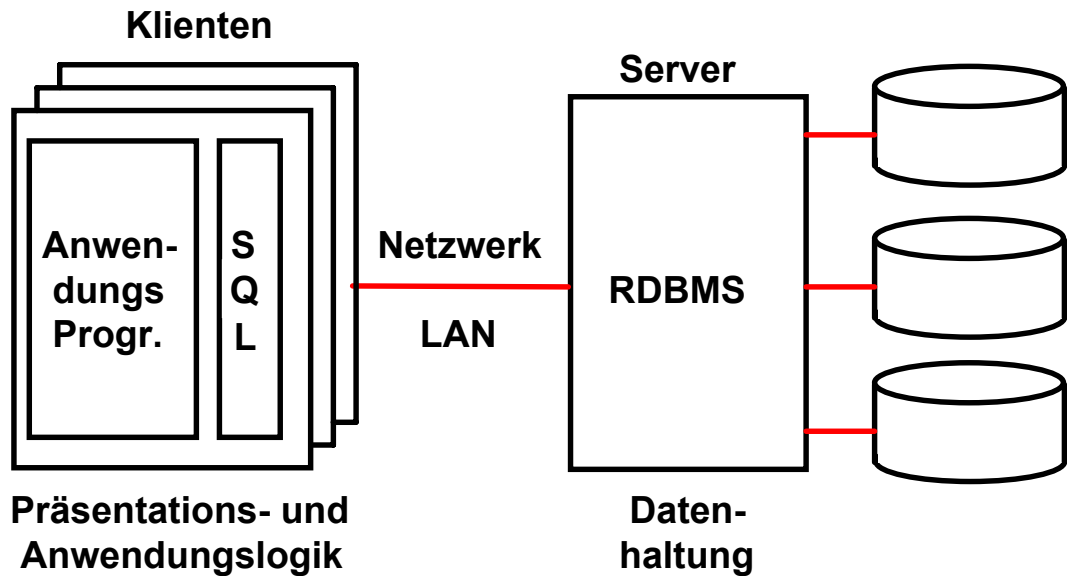


Typische Client/Server Anwendung

Business Logic verarbeitet die Eingabedaten des Endbenutzers und erzeugt Ausgabedaten für den Endbenutzer, z.B. in der Form einer wenig strukturierten Zeichenkette.

Präsentationslogik formt die rohen Ausgabedaten in eine für den Endbenutzer gefällige Form an.

Anwendungsprozess und **Datenbankprozess** laufen in getrennten virtuellen Adressenräumen, ggf. auch auf getrennten physischen Servern.



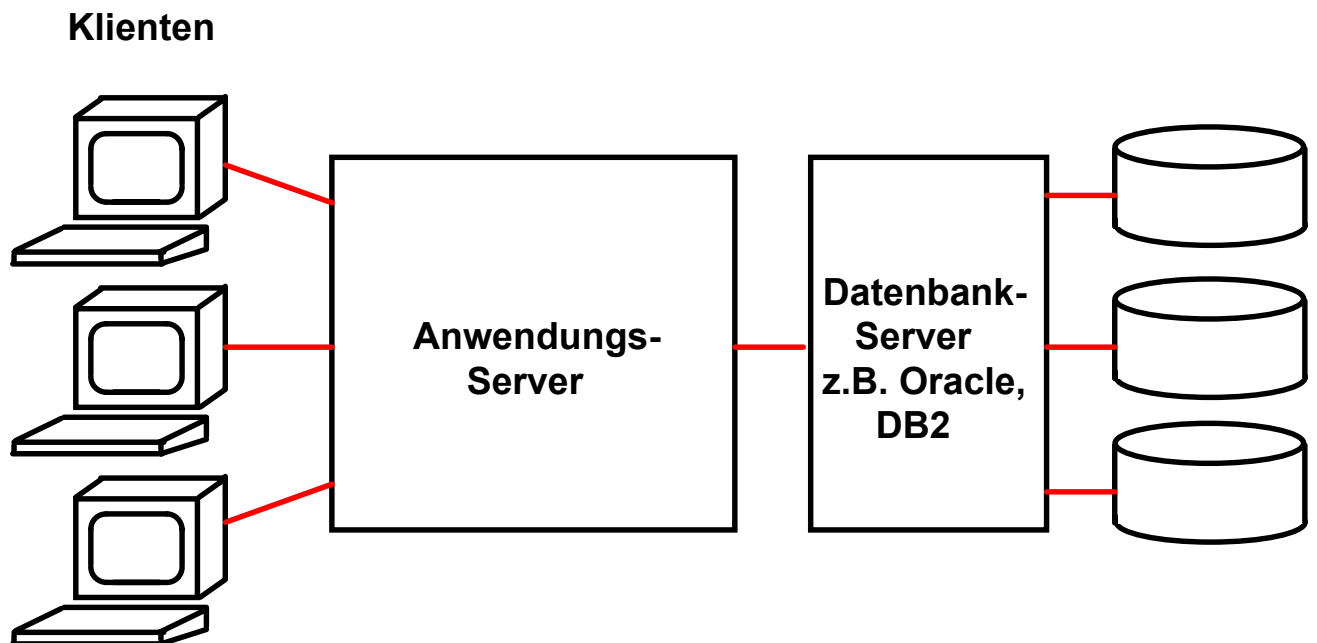
Annahmen:

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

2-Tier Client/Server Architektur

Zweistufige Client/Server Architektur

Anwendungsentwicklung z.B. mit Power Builder oder Visual Basic



3-Tier Client/Server Architektur

Dreistufige Client/Server Architektur

Zwei Arten

Präsentationslogik läuft auf dem Klienten
Beispiel: SAPGUI des SAP R/3 Systems

Präsentationslogik läuft auf dem Anwendungsserver
Beispiel: Servlets und Java Server Pages

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

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.

Transaktionssysteme

Anwendungsbeispiele:

Auskunftssysteme

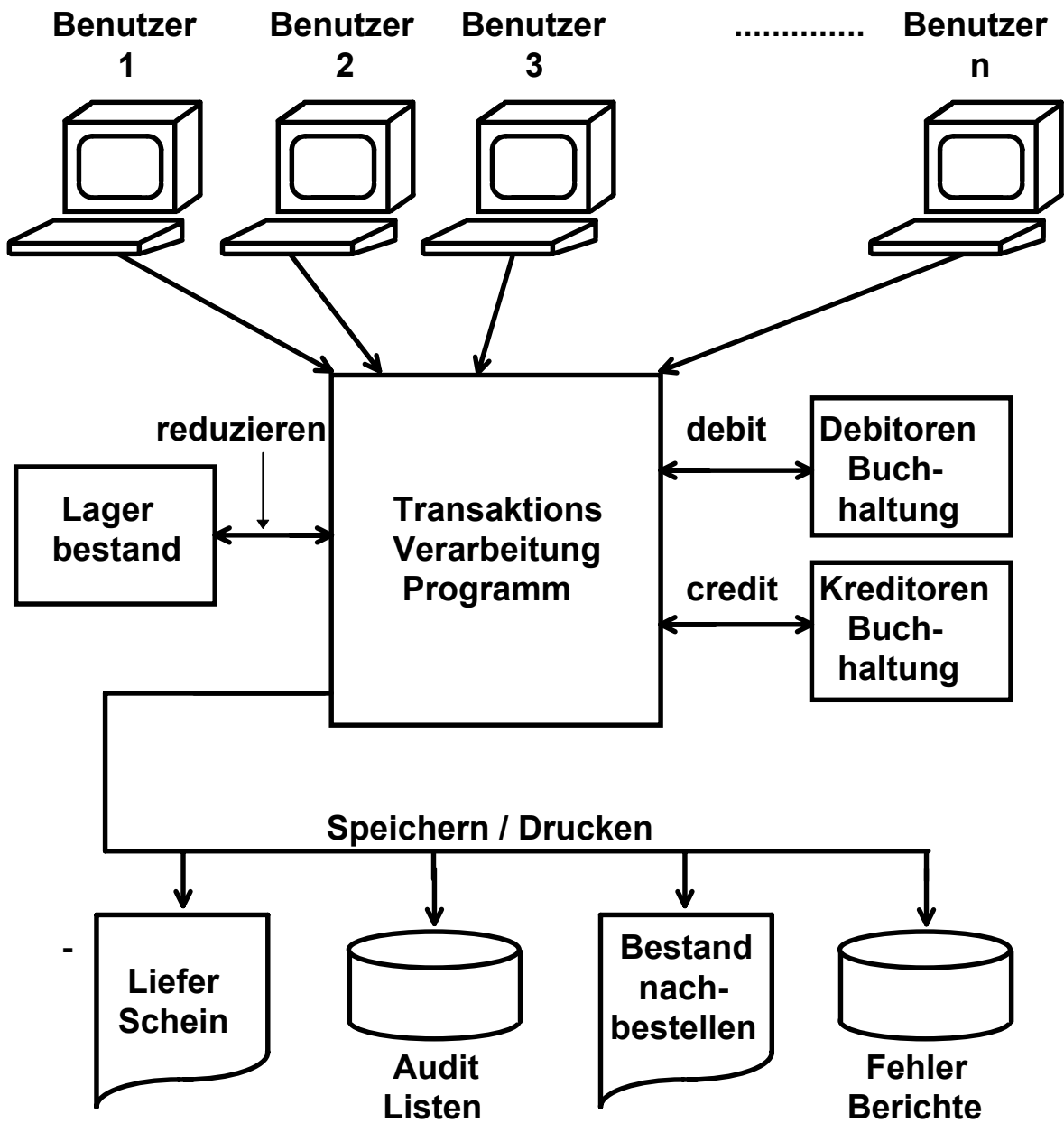
Buchungssysteme (z.B. Flugplatzreservierung)

Geldausgabeautomaten

Auftragsbearbeitung

Lagerbestandsverwaltung

**Etwa 80% aller Datenverarbeitungsvorgänge in der
wirtschaft werden als Transaktionen durchgeführt.**



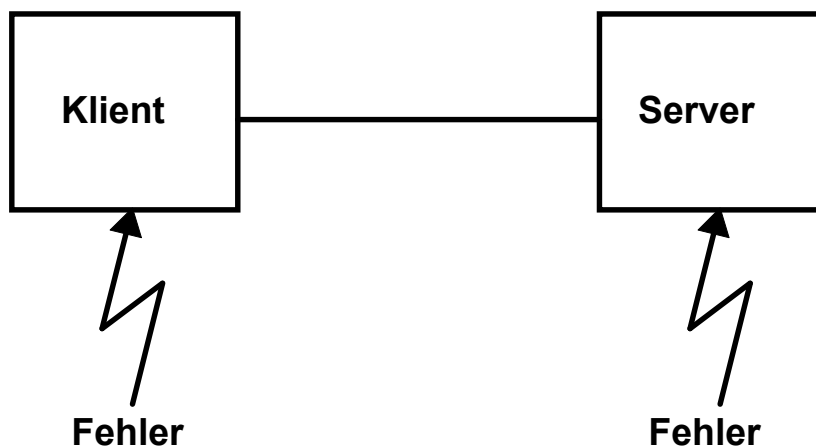
**Beispiel für eine
Transaktionsverarbeitungsanwendung:
Auftragseingang-Bearbeitung**

Fehlerbehandlung

Fehlermöglichkeiten

- **Server kann Prozedur nicht beenden**
(z.B. Rechnerausfall, SW-Fehler)
- **Klient wird während des RPCs abgebrochen**
(z.B. Rechnerausfall, SW-Fehler)

Im Gegensatz zum lokalen Prozeduraufruf sind Fehlerbehandlungsmaßnahmen erforderlich.



Fehlerbehandlung - Ausfall des Servers

1. Idempotente Arbeitsvorgänge

"Idempotent" sind Arbeitsvorgänge, die beliebig oft ausgeführt werden können; Beispiel: Lese Block 4 der Datei xyz. Nicht idempotent ist die Überweisung eines Geldbetrages von einem Konto auf ein anderes.

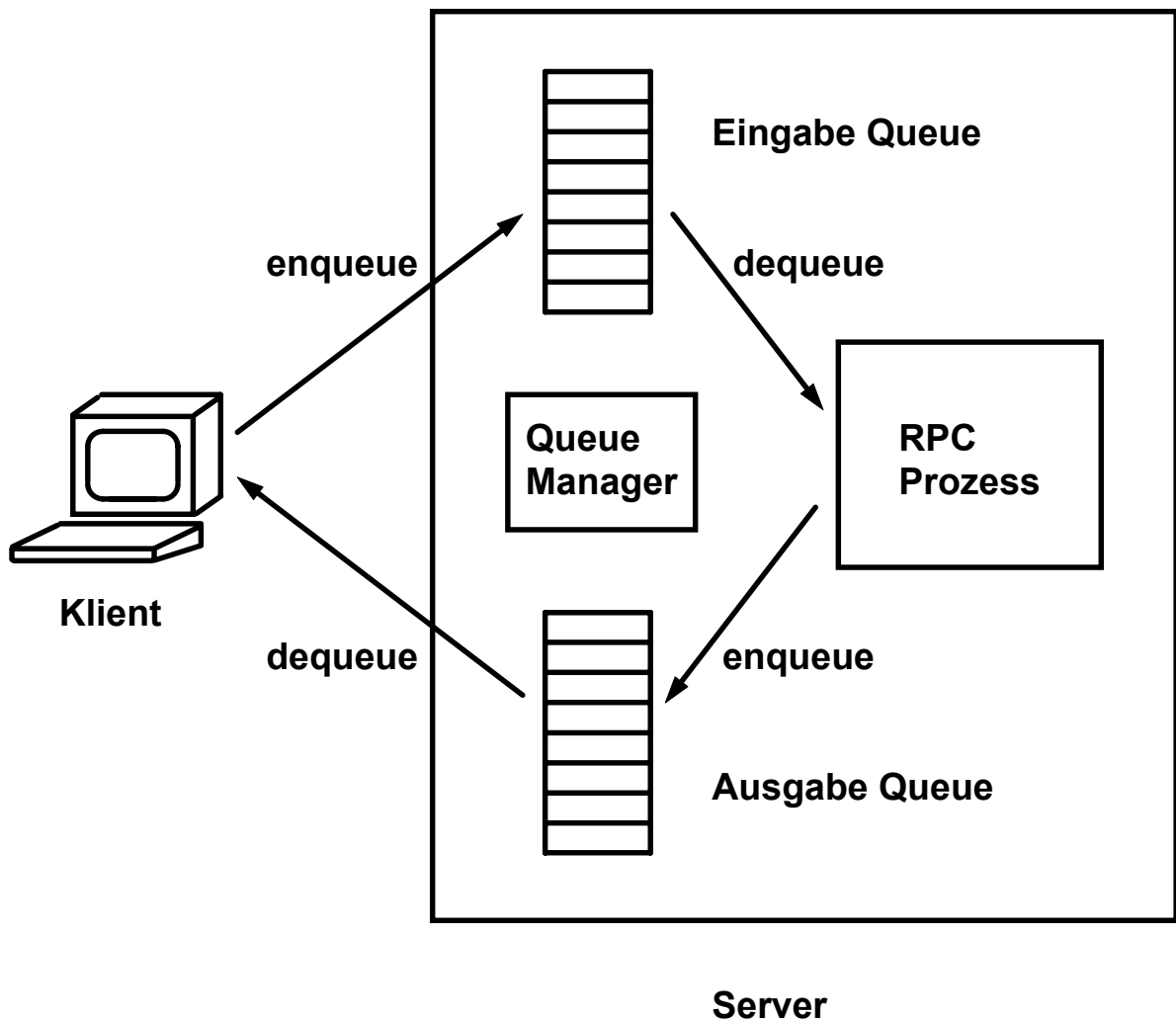
2. Nicht-idempotente Arbeitsvorgänge

Problem: Wie wird festgestellt, ob Arbeitsgang durchgeführt wurde oder nicht.

"Genau einmal" (exactly once).

"Höchstens einmal" (at most once).

"Wenigstens einmal" (at least once). Ideal für idempotente Vorgänge.



Fehlerbehandlung - Ausfall des Klienten

Getrennte Warteschlangen für eingehende und ausgehende Nachrichten lösen viele Probleme. Wenn der Klient abstürzt (Waise), schließt der Server seine Arbeit ab. Das Ergebnis der RPC Verarbeitung steht in der Ausgabe Queue, und kann beim Wiederanlauf des Klienten dort abgeholt werden.

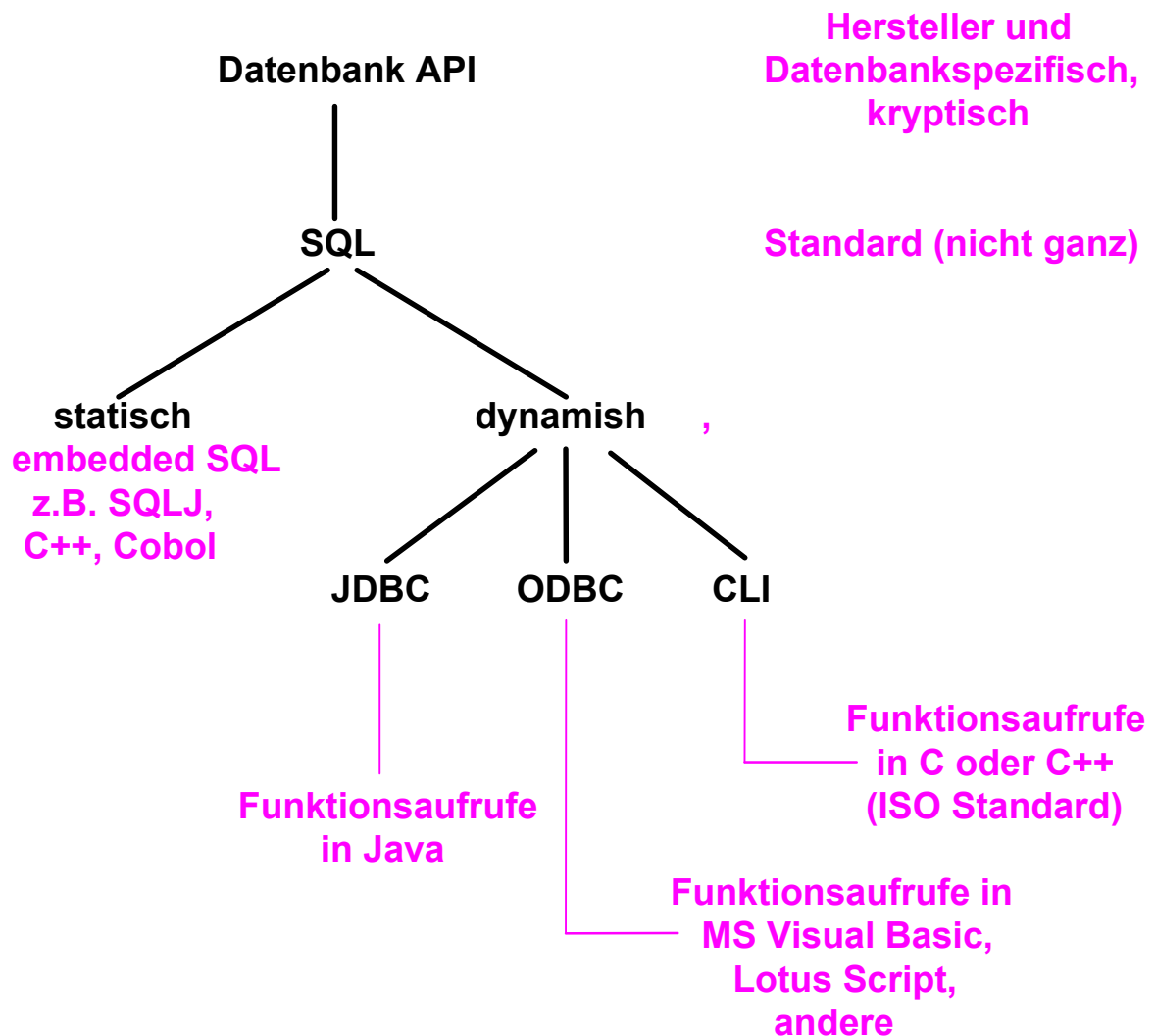
Andere Aufgaben der Queues:

- Prioritätensteuerung
- Lastverteilung auf mehrere Server

Embedded SQL, Beispiel für C (1)

Ein Anwendungsprogramm implementiert typischerweise statische 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');
    .....
    .....
}
```



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 iterator SeatCursor(Integer row, Integer col,
String type, int status);
Integer status = ?;
SeatCursor sc;
#sql sc = {
select rownum, colnum from seats where status <=
:status
};
while(sc.next())
{
#sql { insert into categ values(: (sc.row()),
:(sc.col())) };
}
sc.close();

```

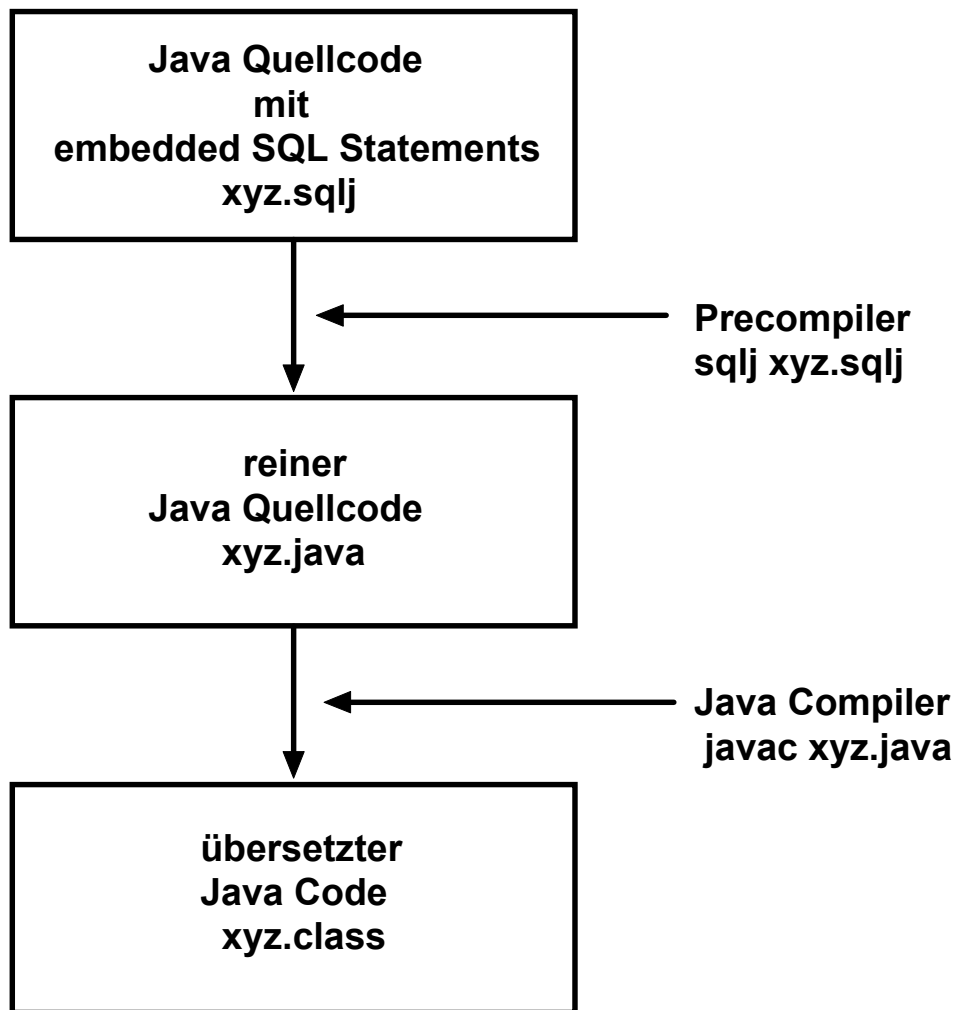
SQLJ

```

Integer status = ?;
PreparedStatement stmt = conn.prepareStatement("select
row, col from seats
where status <= ?");
if (status == null) stmt.setNull(1,Types.INTEGER);
else stmt.setInt(1,status.intValue());
ResultSet sc = stmt.executeQuery();
while(sc.next())
{
int row = sc.getInt(1);
boolean rowNull = sc.isNull();
int col = sc.getInt(2);
boolean colNull = sc.isNull();
PreparedStatement stmt2 =
conn.prepareStatement("insert into categ
values(?, ?)");
if (rowNull) stmt2.setNull(3,Types.INTEGER);
else stmt2.setInt(3,rownum);
if (colNull) stmt2.setNull(4,Types.INTEGER);
else stmt2.setInt(4,colnum);
stmt2.executeUpdate();
stmt2.close();
}
sc.close();
stmt.close();

```

JDBC



Beispiel: SQLJ Precompiler

Embedded SQL bedeutet, wir fügen SQL Kommandos in normale Programmiersprachen wie Cobol, C/C++ oder Java ein.

Der Cobol, C/C++ oder Java Compiler versteht die SQL Kommandos nicht. Sie müssen zunächst mit einem *Pre-Compiler* in Funktionsaufrufe der entsprechenden Programmiersprache übersetzt werden.

Der Pre-Compiler übersetzt embedded SQL Kommandos in entsprechende Datenbank API Calls.

Unterschiedliche Precompiler für DB2, Oracle, Sybase

Literatur: IBM Redbook *e-business Cookbook for z/OS Volume II: Infrastructure*, July 2002, section 4.5.

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

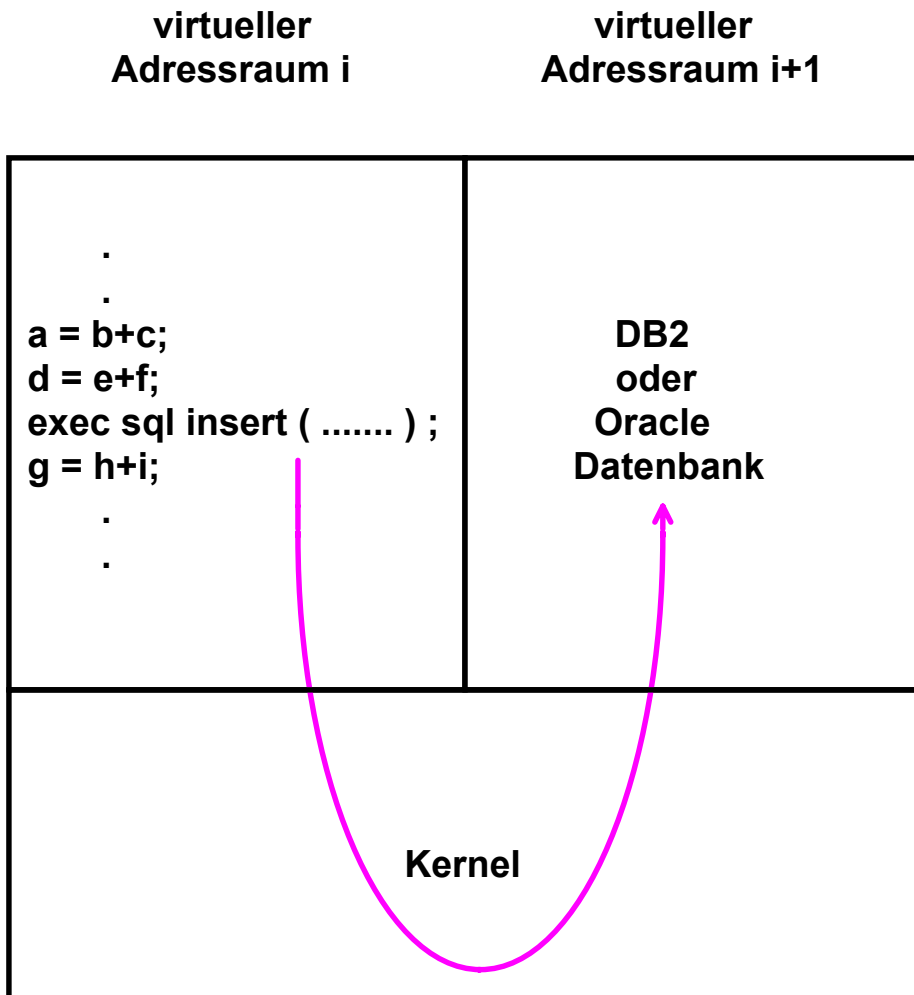
Eingebettete SQL-Anweisungen werden durch "EXEC SQL" eingeleitet und durch spezielles Symbol (hier ";") beendet, um einem Precompiler die Unterscheidung von anderen Anweisungen zu ermöglichen

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.)

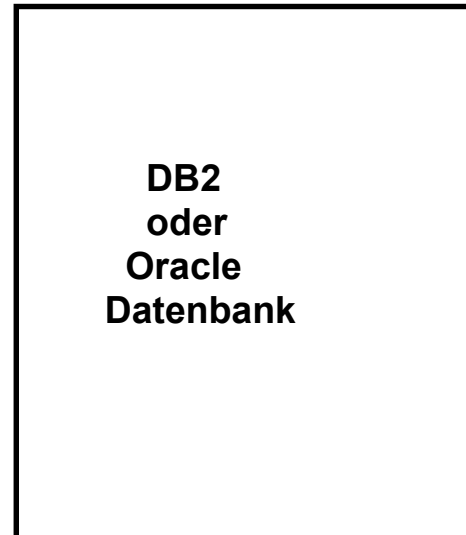
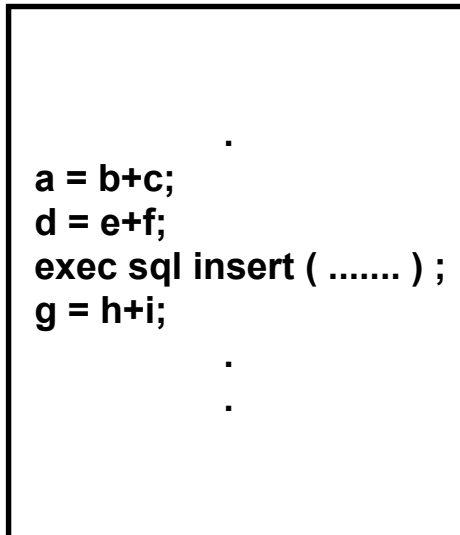


SQL Datenbank Zugriff auf dem gleichen Rechner

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**

**Der Precompiler erzeugt für den SQL Aufruf
Client-Code für den Datenbankserver**

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

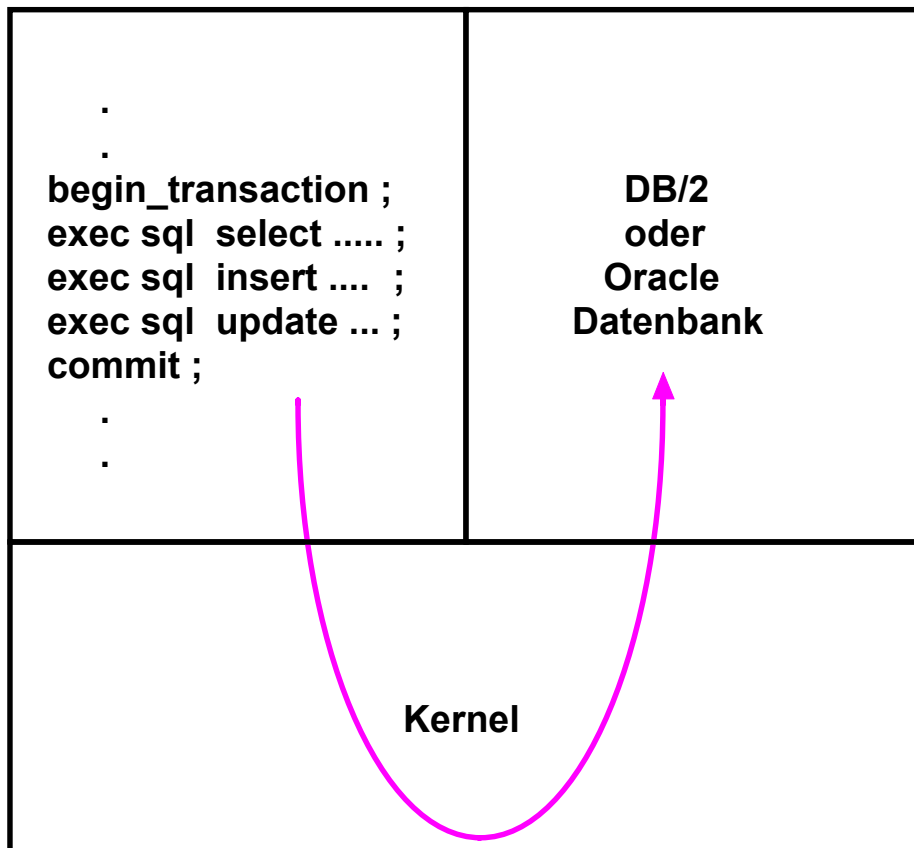
exec sql BEGIN WORK

und

exec sql COMMIT WORK

als Work Unit entsprechend den ACID Anforderungen aus.

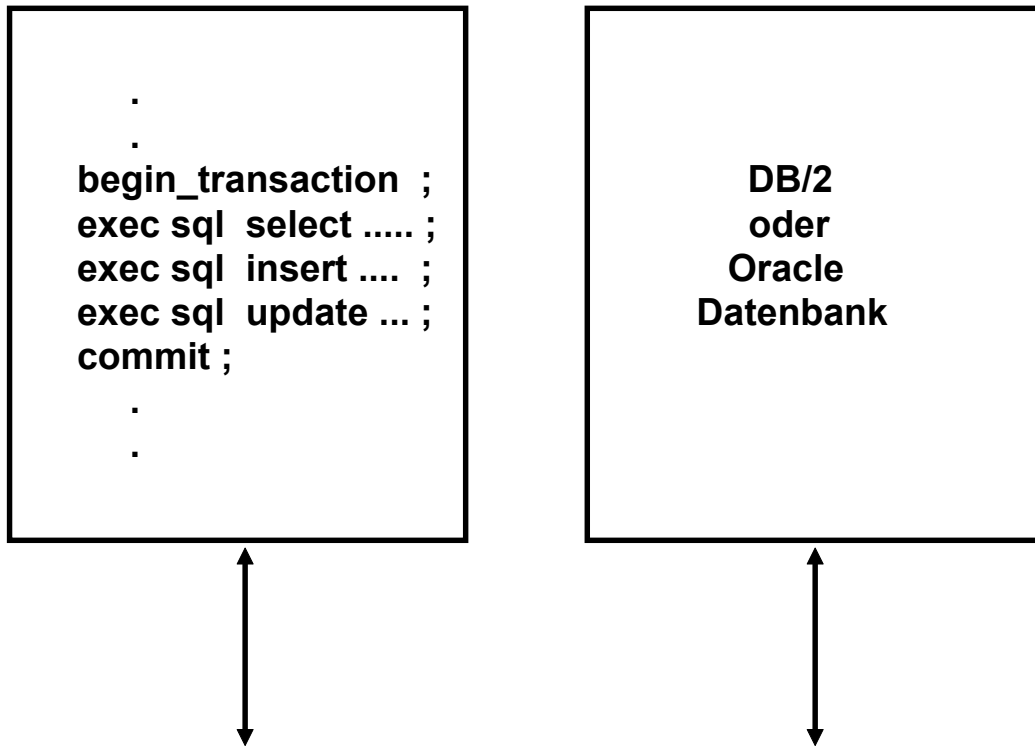
virtueller Adressraum i virtueller adressraum i+1



ACID Eigenschaften für eine Gruppe von SQL Aufrufen

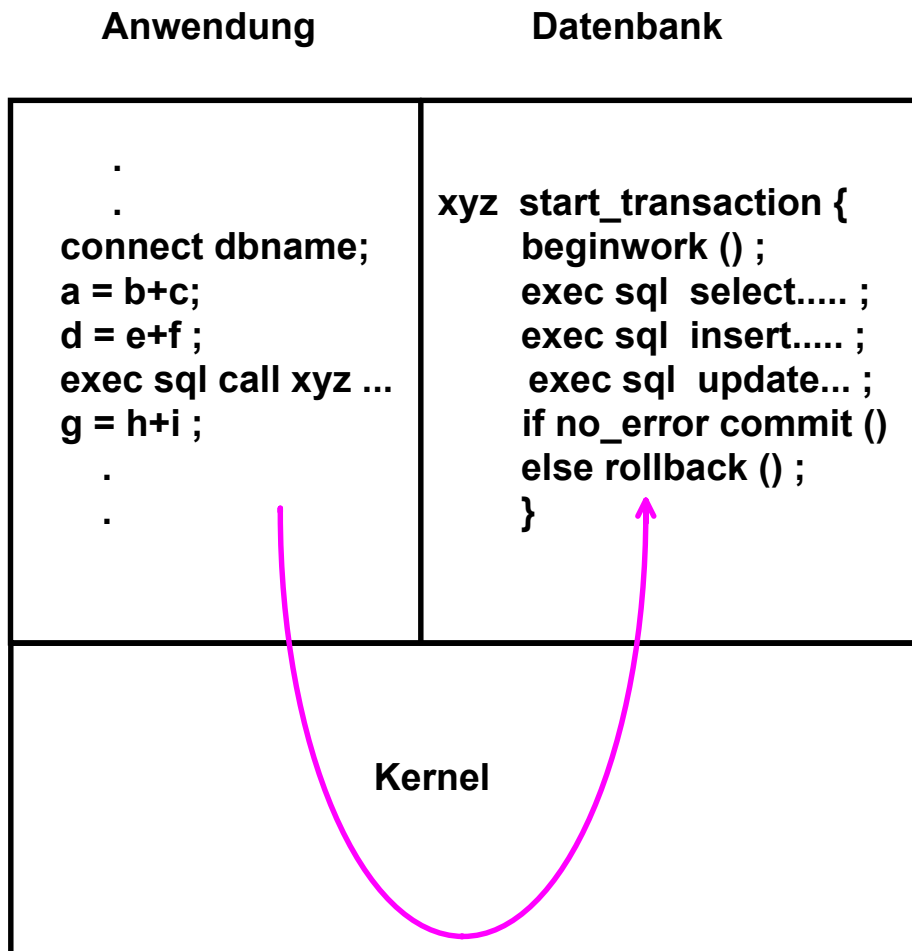
Rechner 1

Rechner 2



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

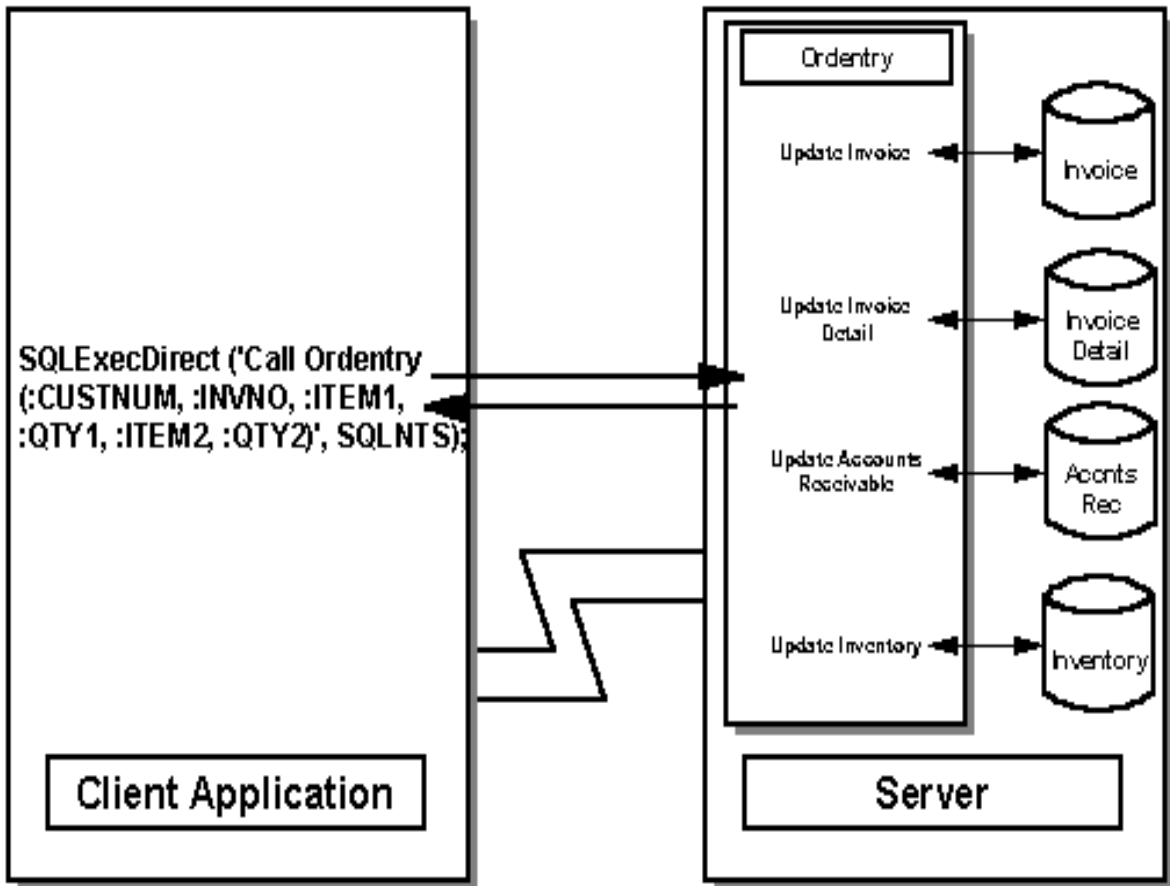
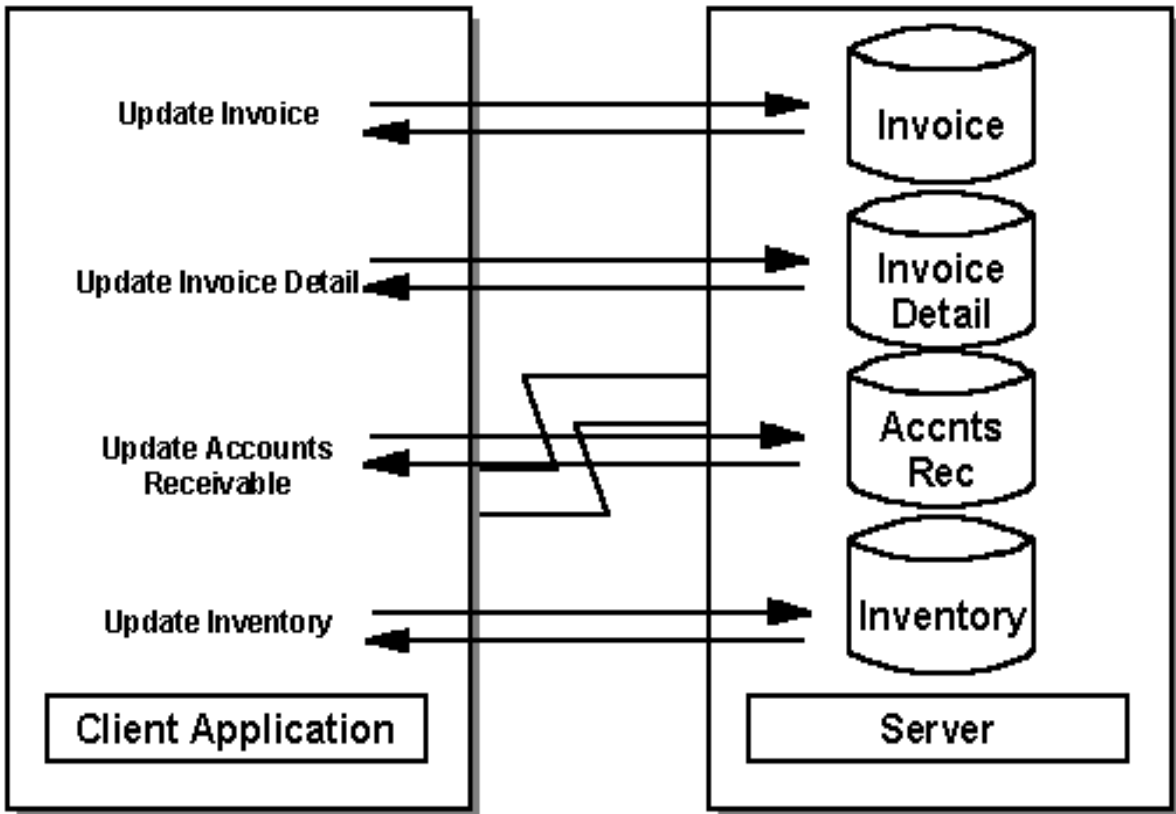
**ACID Eigenschaften für eine Gruppe
von SQL Aufrufen**

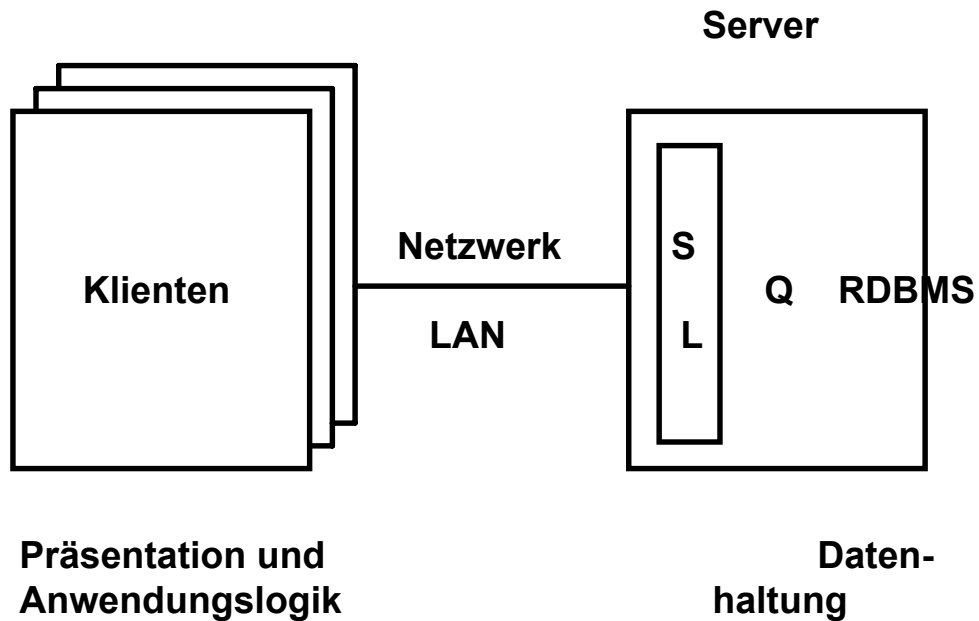


Stored Procedure

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.

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

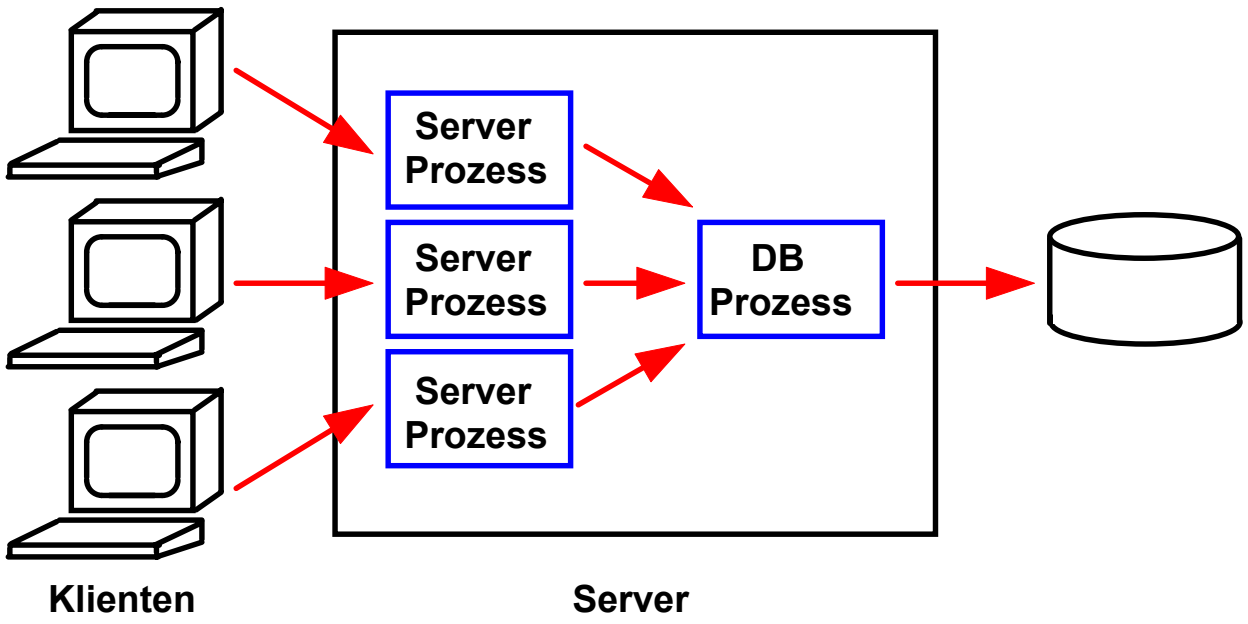
Daten verarbeiten

Ergebnis speichern

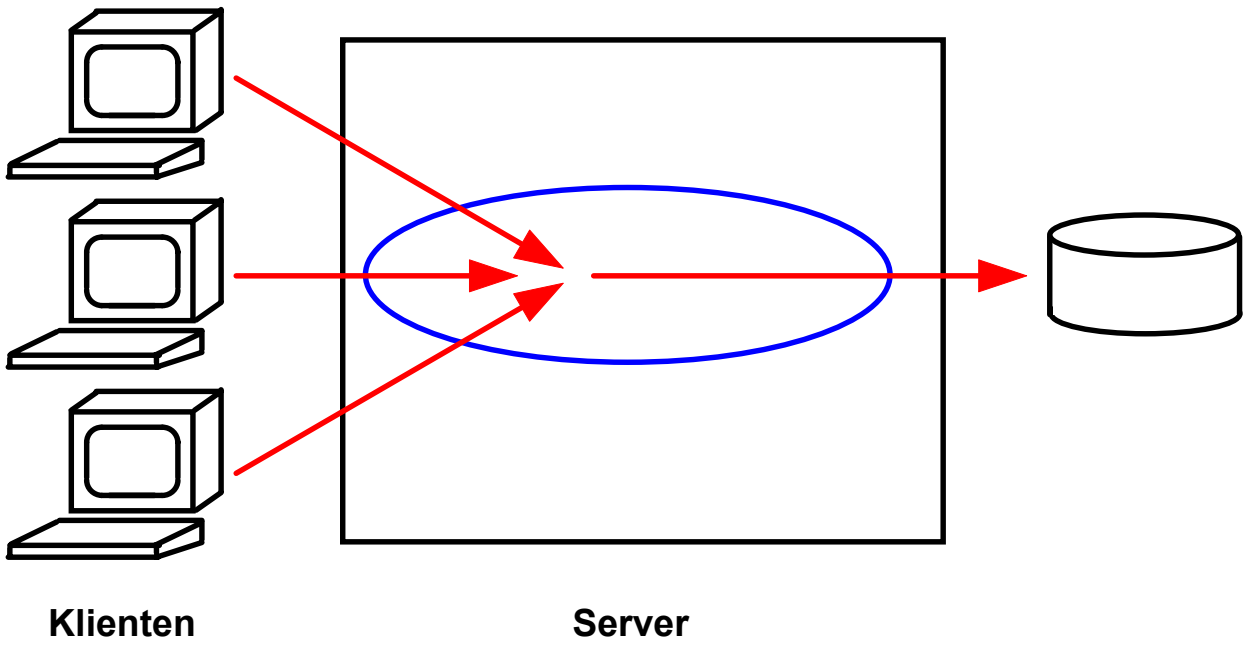
reset lock

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

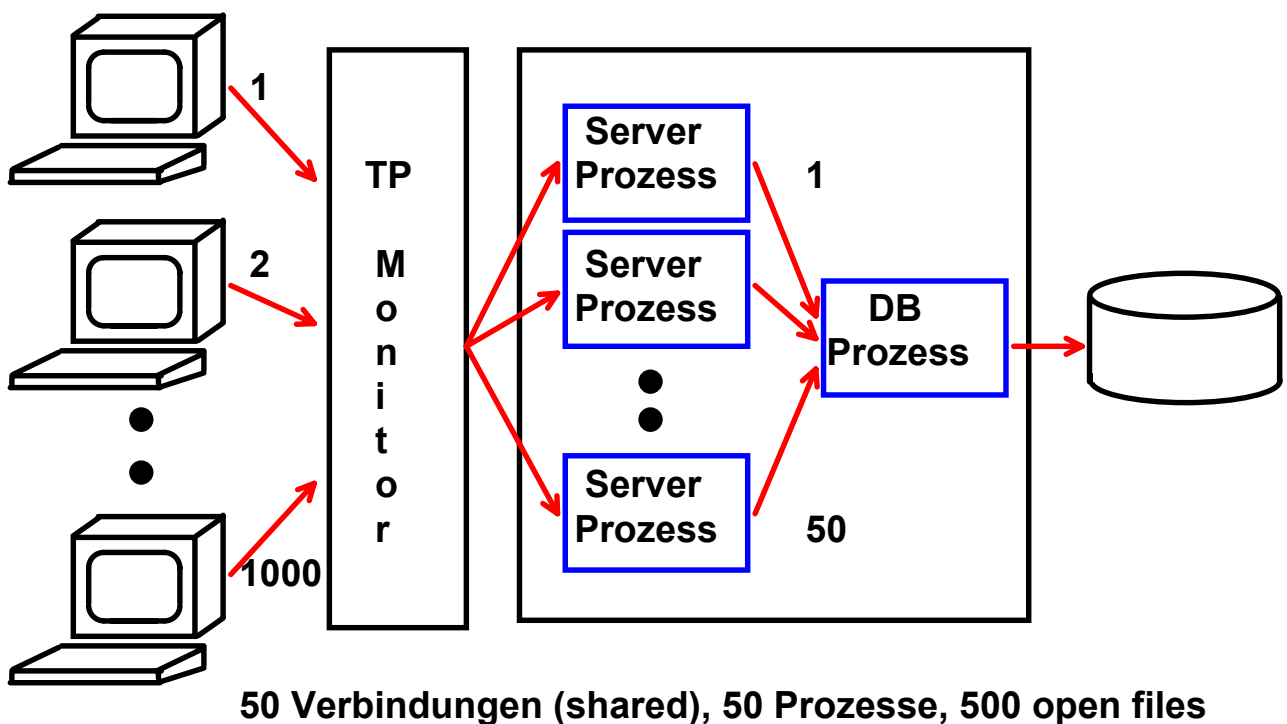
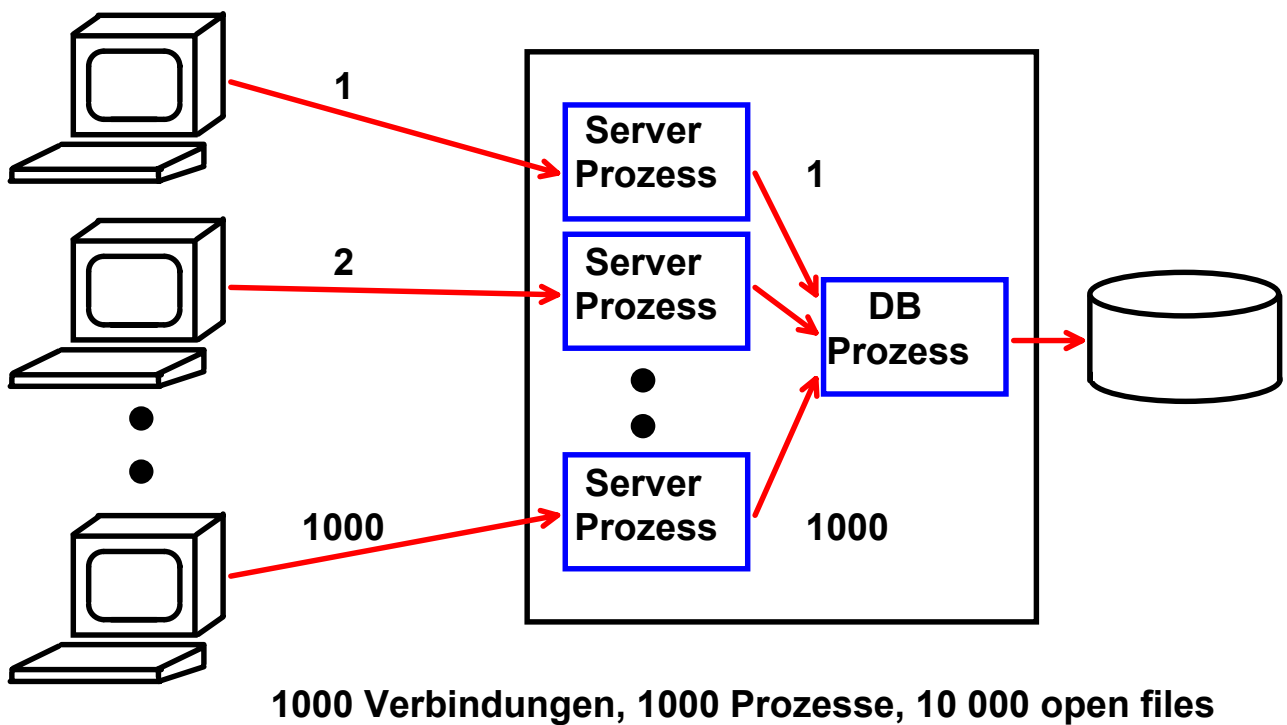
Bei starker Belastung steigt die Anzahl der rollbacks exponentiell an. Deshalb hier Locks einsetzen und Prozesse auf explizite Datenfreigabe warten lassen.



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