UNIVERSITÄT LEIPZIG

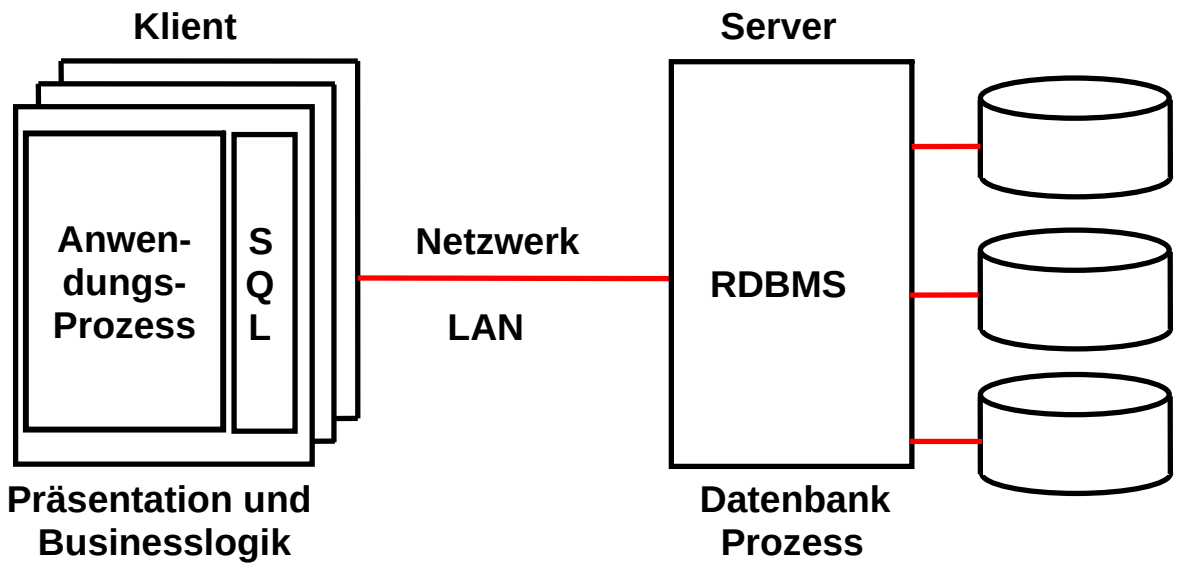
Enterprise Computing
Einführung in das Betriebssystem z/OS

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

WS2012/2013

Transaktionsverarbeitung Teil 3

Stored Procedures

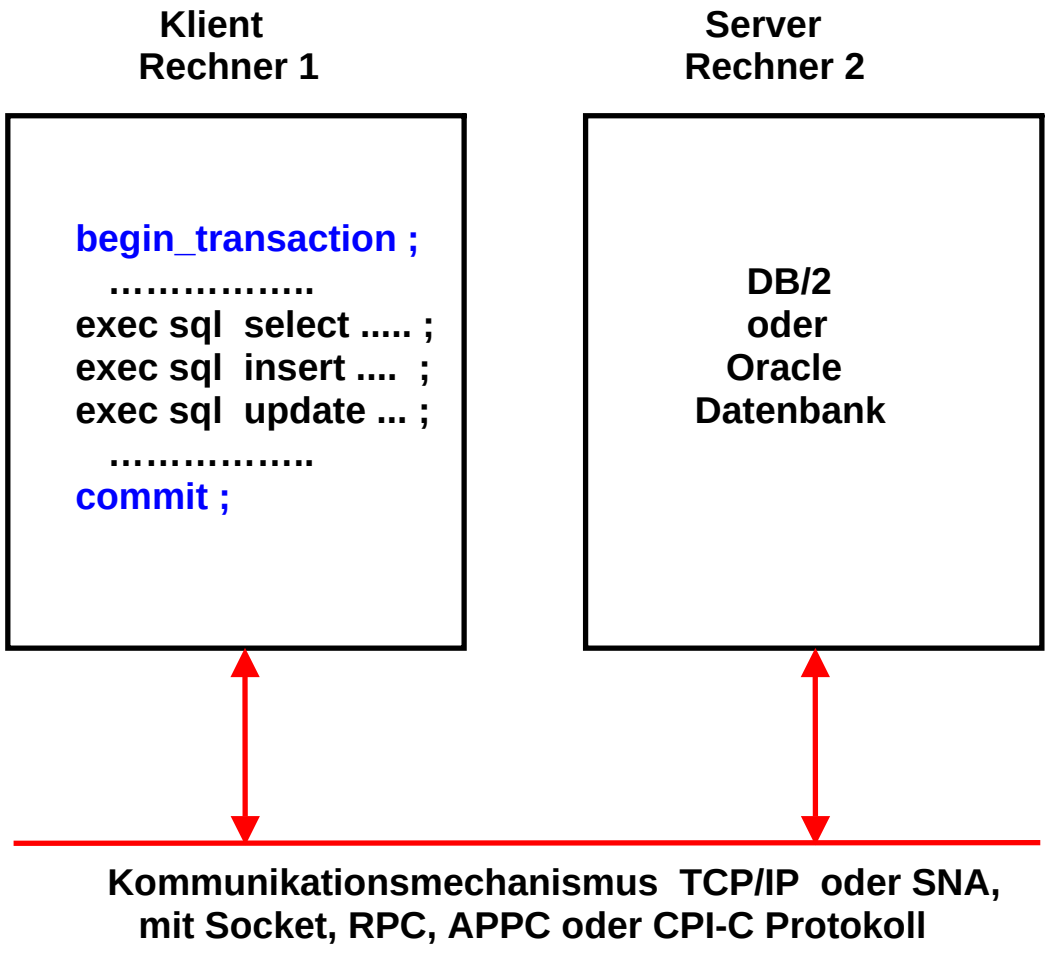


2-Tier Client/Server Architecture

2-stufige Client/Server-Architektur

Der Anwendungsprozess auf dem Klientenrechner sendet SQL Befehle an den Datenbankrechner. Auf dem Client befindet sich eine SQL-Client Komponente, die in der Lage ist, SQL Anfragen über das Netz an den Datenbankserver zu senden.

Der Datenbankprozess stellt sicher, dass jedes einzelne SQL Statement unter Einhaltung der ACID Bedingungen ausgeführt wird.



In vielen Fällen führt der Klient eine Reihe von EXEC SQL Kommandos aus, die insgesamt als Gruppe ACID Eigenschaften haben sollen.

Der Rechner führt die Anweisungen zwischen

exec sql begin_transaction

und

exec sql commit

als Arbeitseinheit (Work Unit) entsprechend den ACID Anforderungen aus.

Dies stellen die beiden Schlüsselwörter begin_transaction und commit sicher.

Eine Gruppe mit mehrfachen SQL Statements, die ACID Eigenschaften haben, werden als Embedded SQL Transaktion bezeichnet.

ACID Eigenschaften für eine Gruppe von SQL Aufrufen

Der Anwendungsprozess verfügt über einen SQL Klienten, der die Kommunikation mit dem Datenbank Prozess herstellt. Dabei ist es gleichgültig, ob der Datenbank Prozess auf dem gleichen Rechner wie der Anwendungsprozess läuft, oder auf einem entfernten (remote) Rechner läuft, der mit dem Anwendungsrechner über TCP/IP verbunden ist.

Embedded SQL Transaktion mit mehrfachen SQL Statements

```
DebitCreditApplication( )
{     receive input message;
      exec sql begin_transaction ;                    /* start transaction */
      Abalance = DoDebitCredit (.....);
                    •
                    •
                    •
          exec sql  select ..... ;
          exec sql  insert ....  ;
          exec sql  update ... ;
                    •
                    •
                    •
      if (Abalance < 0 && delta < 0)
            {   exec sql rollback ;   }
      else  {
            send output message;
            exec sql commit ;                         /* end transaction */
            }
}
```

In dem gezeigten Code Fragment fängt der transaktionale Abschnitt mit exec sql begin_transaction an, und hört mit entweder exec sql commit oder mit exec sql rollback auf. Rollback bewirkt, dass alle Datenbank-Änderungen wieder rückgängig gemacht werden.

An Stelle von Schlüsselworten wie rollback sind auch Schlüsselworte wie abort oder backout gebräuchlich.

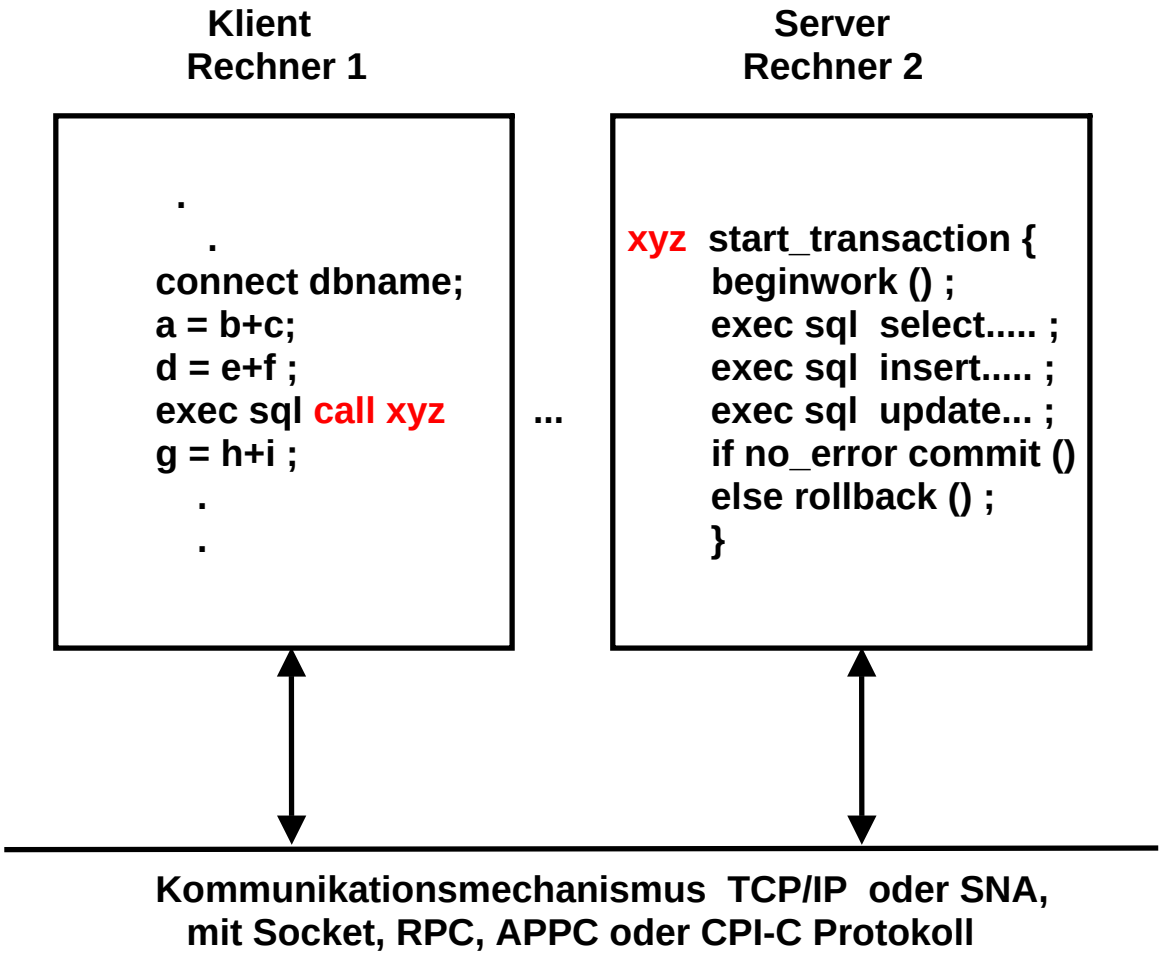
Embedded SQL Transaktion mit mehrfachen SQL Statements

Eine Embedded SQL Transaktion mit mehrfachen SQL Statements stellt ein relativ simples Programmiermodell dar. Sie ist einfach zu entwerfen und zu implementieren. Da der ganze Anwendungscode auf dem Klienten läuft, kann ein einzelner Programmierer die Verantwortung für die ganze Anwendung übernehmen.

Es existiert jedoch eine Reihe von Nachteilen.

Da die gesamte Anwendung auf dem Klienten-Rechner läuft, ist für jedes EXEC SQL Statement eine getrennte Netzwerk-I/O Operation erforderlich. Dies kann die Performance stark herabsetzen. Weiterhin muss das Anwendungsprogramm über detaillierte Kenntnisse des Datenbank-Designs verfügen. Jede Änderung im Datenbank-Design erfordert eine entsprechende Änderung im Anwendungsprogramm. Weiterhin erfordern mehrfache Kopien des Anwendungsprogramms auf mehreren Arbeitsplatzrechnern einen erhöhten Administrationsaufwand.

Stored Procedures lösen diese Probleme.



Mit Hilfe einer “Stored Procedure” wird die Gruppe von SQL Aufrufen nicht auf dem Klienten sondern auf dem Server ausgeführt. Der Klient ruft die Stored Procedure mit Hilfe eines „call“ statements auf.

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

Mit Hilfe des „connect“ Statements wählt der Klient die Datenbank mit dem Namen „dbname“ (an Stelle einer evtl. anderen angeschlossenen Datenbank) aus.

ACID Eigenschaften für eine Gruppe von SQL Aufrufen

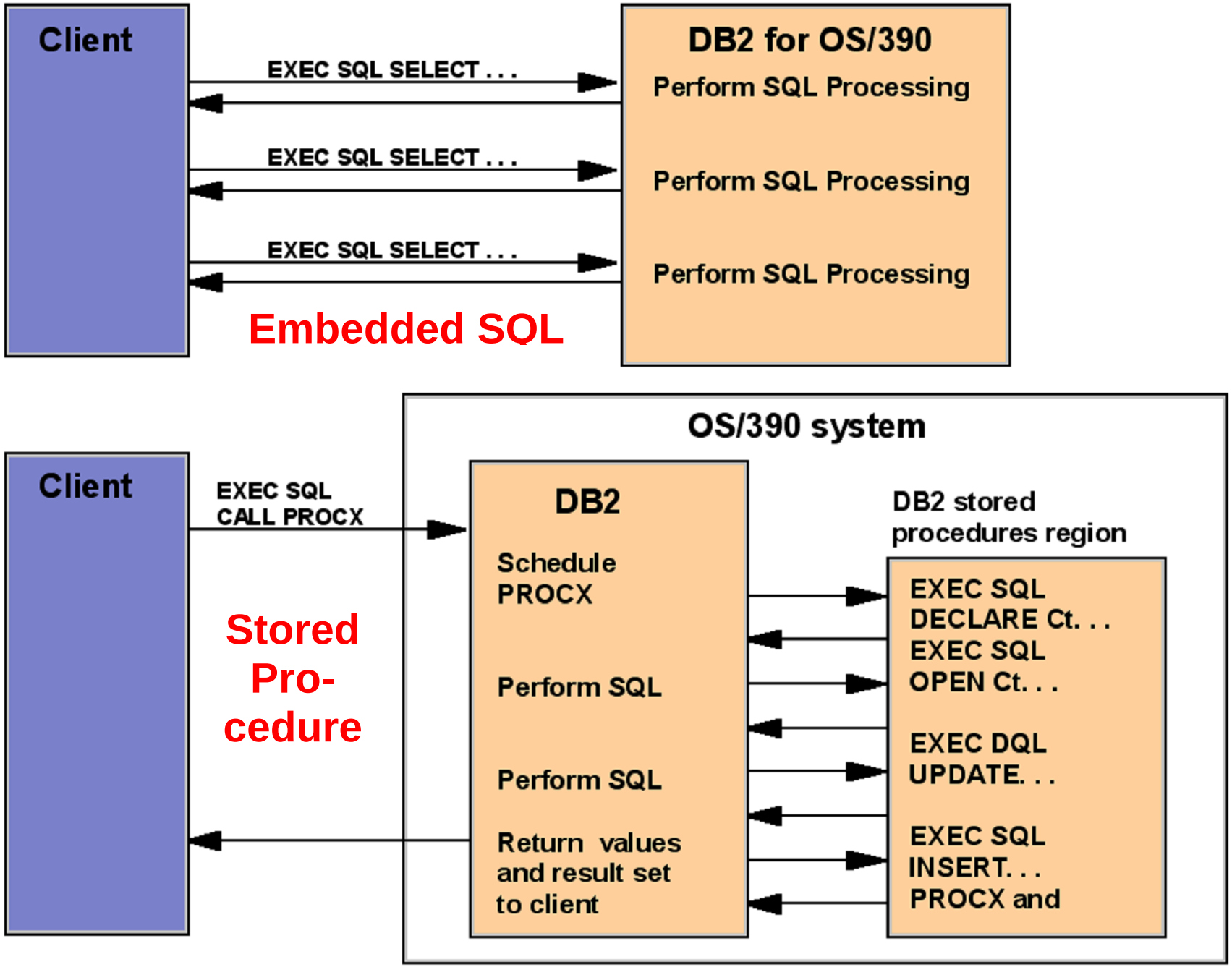
Stored Procedure

Die folgende Abbildung stellt dar, wie durch den Einsatz einer Stored Procedure die Anzahl der SQL Zugriffe über das Netz deutlich reduziert werden kann. Dies verringert das Datenvolumen, welches über das Netz geht.

Zusätzlich wird auch der Aufwand für die TCP/IP Verarbeitung deutlich verringert. Die Stored Procedure läuft in der Regel auf dem gleichen Server wie die Datenbank, und zwar in einem eigenen Address Space, getrennt von dem Datenbank Address Space. Die Kommunikation zwischen den beiden Address Spaces erfolgt aber über den Betriebssystem Kernel und erfordert wesentlich weniger CPU Zyklen als die TCP/IP Verarbeitung.

Als Ergebnis laufen Stored Procedures deutlich performanter als dies bei einzelnen SQL Aufrufen der Fall ist.

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


Client
EXEC SQL SELECT . . .
EXEC SQL SELECT . . .
EXEC SQL SELECT . . .
DB2 for OS/390
Perform SQL Processing
Perform SQL Processing
Perform SQL Processing
Embedded SQL
OS/390 system
Client
EXEC SQL
CALL PROCX
Stored
Pro-
cedure
DB2
Schedule
PROCX
Perform SQL
Perform SQL
Return values
and result set
to client
DB2 stored
procedures region
EXEC SQL
DECLARE Ct. . .
EXEC SQL
OPEN Ct. . .
EXEC DQL
UPDATE. . .
EXEC SQL
INSERT. . .
PROCX and

```
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 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;
}
```

Die hier gezeigte Stored Procedure enthält neben SQL Anweisungen auch Statements in einer regulären Programmiersprache. Eine Stored Procedure kann relativ komplexen Programm Code enthalten.

Traditionell verwenden Stored Procedures unter z/OS hierfür reguläre Programmiersprachen wie COBOL, PL/I, C/C++, Assembler, REXX, and Java (external high-level language stored procedure). z/OS unterstützt neben den traditionellen "external high-level language stored procedures" noch "SQL procedures". SQL procedures sind Stored Procedures, die in der Sprache SQL/PSM (SQL Persistent Modules) geschrieben sind. SQL/PSM ist ein ISO Standard.

Es existiert eine ähnliche, aber proprietäre Programmiersprache PL/SQL (Procedural Language/SQL) für Oracle Datenbanken, die sich in der Syntax an die Programmiersprache Ada angelehnt.

Stored Procedures

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 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.**

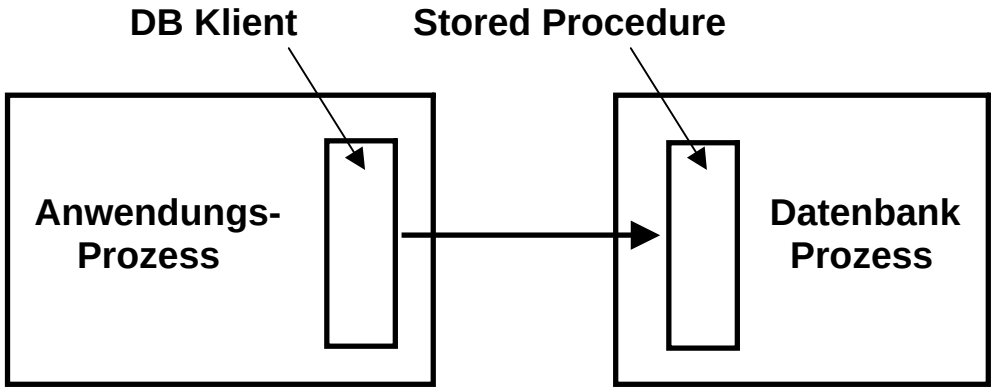
Stored Procedures bündeln SQL Statements bei Zugriffen auf Relationale Datenbanken. Sie ersetzen viele, vom Klienten an den Server übergebene, SQL Statements durch eine einzige Stored Procedure Nachricht

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)**

Stored Procedure Datenbank Klient



Bei einer Stored Procedure führt nicht der Anwendungsprozess, sondern der Datenbankprozess, die Gruppe von SQL Statements aus.

Datenbank Hersteller, z.B. IBM, Oracle, Sybase etc. stellen eigene (proprietäre) Datenbank-Klienten zur Verfügung, die mit den Stored Procedures der Datenbank kommunizieren. Datenbank Klienten sind nicht kompatibel miteinander.

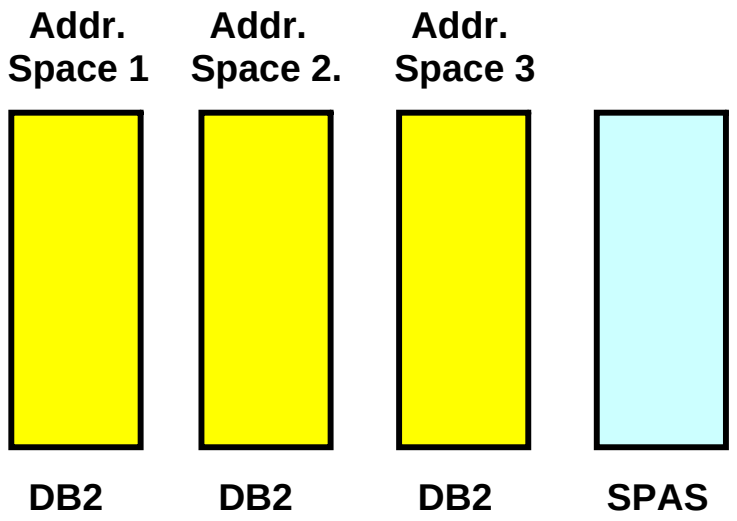
Der Klient enthält einen Treiber (DB Klient), der ein proprietäres “Format and Protocol” (FAP) definiert.

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

DB2 Stored Procedures

DB2 ist verfügbar für z/OS, zVSE, i5/OS (OS/400), and als „DB2 UDB“ (Universal Data Base) für alle verbreiteten Unix (einschließlich Linux) und Windows Betriebssysteme. Es benötigt standardmäßig 3 Address Spaces unter z/OS.

z/OS DB2 Stored Procedures laufen in eigenen Adressenraum (SPAS, Stored Procedure Address Space)



Die Benutzung von Stored Procedures erfordert Verhandlungen mit dem Datenbank Administrator. Das Entwickeln von Stored Procedures ist eine komplexe Aufgabe; das Debuggen kann schwierig sein.

Leistungsverhalten

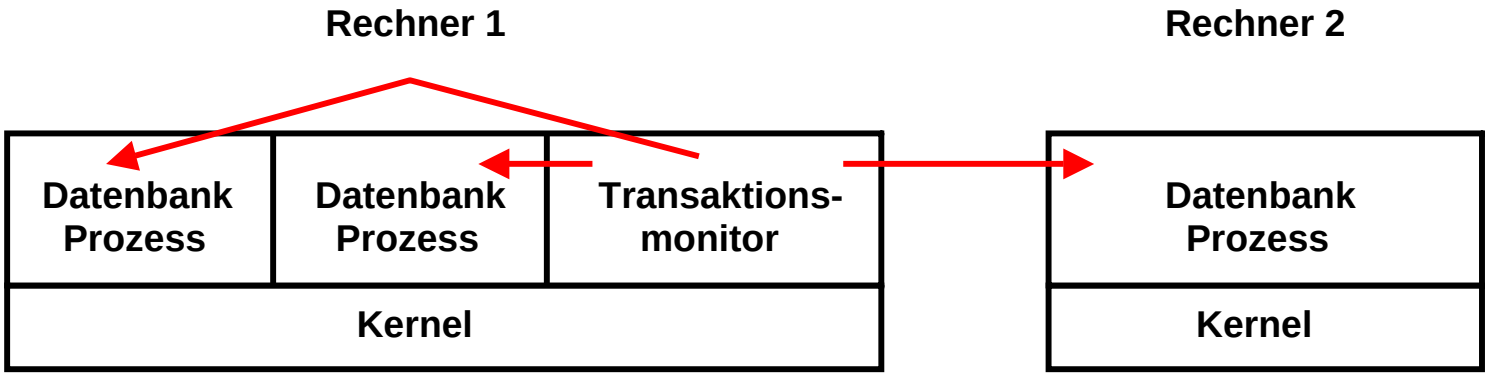
Das Lehrbuch von R. Orfali, D. Harkey: „Essential Client/Server Survival Guide“. John Wiley, 1994, S. 178, enthält einen detaillierten Leistungsvergleich mit einer einfachen Transaktion, für die der vollständige Code vorliegt. Die Messungen wurden unter dem OS/2 Betriebssystem auf einem Intel 80486 Rechner durchgeführt.

Dies sind die Meßergebnisse in ausgeführten Transaktionen pro Sekunde:

Dynamic SQL	**2,2**	**Transaktionen/s**
Static SQL	**3,9**	**Transaktionen/s**
Stored Procedure	**10,9**	**Transaktionen/s**

Zu sehen ist ein deutlicher Performance Vorteil bei der Benutzung von statischen gegenüber dynamischen SQL (Faktor 1,75). Noch deutlicher ist der Performance Vorteil beim Einsatz von Stored Procedures (Faktor 2,79).

Mit einem modernen Rechner wäre die Transaktionsrate (Anzahl ausgeführter Transaktionen pro Sekunde) sehr deutlich höher. Am Verhältnis der Ausführungszeiten würde sich vermutlich nicht viel ändern.



Probleme mit Stored Procedures

Die Stored Procedure Laufzeitumgebung ist Bestandteil eines Datenbankprozesses. Häufig ist es jedoch erforderlich, dass eine transaktionale Anwendung auf zwei oder mehr Datenbanken zugreift. Die Zugriffe müssen die ACID Bedingungen erfüllen.

Ein weiteres Problem tritt auf, wenn eine Transaktion auf mehrere heterogene Datenbanken, z.B. von unterschiedlichen Herstellern zugreift. Beispielsweise ist bei der Buchung einer Urlaubsreise in einem Reisebüro ein Zugriff auf die Datenbank einer Fluglinie sowie auf die Datenbank eines Hotels erforderlich. Diese Datenbanken sind nicht identisch.

Zur Lösung implementiert man die Laufzeitumgebung für die Ausführung von Transaktionen als einen getrennten Prozess, der unabhängig vom Datenbankprozess ist, und als „Transaktionsmonitor“ oder „Transaktionsserver“ bezeichnet wird.

Der Transaktionsmonitor (TP Monitor) Prozess übernimmt die Rolle des DB2 Stored Procedure Address Space (SPAS).

Probleme mit Stored Procedures

Es existieren weitere Vorteile beim Einsatz eines Transaktionsmonitors an Stelle von Stored Procedures:

- **Lastverteilung - Leistungsverhalten - Skalierung**
- **Prioritätssteuerung**
- **Verfügbarkeit (High Availability)**

In Mainframe Installationen werden Transaktionen ganz überwiegend mittels eines Transaktionsmonitors und nicht mittels Stored Procedures ausgeführt. Auch das Leistungsverhalten eines Transaktionsmonitors ist häufig deutlich besser.

Beispiel für Transaktionsmonitore

Transaktionsmonitore sind Softwarepakete (auch als Middleware bezeichnet), die von verschiedenen Herstellern vertrieben werden. Die wichtigsten sind:

CICS der Firma IBM
SAP R/3 der Firma SAP
Transaction Server (MTS) der Firma Microsoft
Tuxedo der Firma Bea (heute ein Tochterunternehmen von Oracle)

Weit mehr als die Hälfte aller täglich weltweit ausgeführten Transaktionen werden von CICS verarbeitet.

Daneben von Bedeutung sind :

IMS-DC der Firma IBM
TPF (Transaction Processing Facility) der Firma IBM
UTM der Firma Siemens
NonStop / Guardian / Pathway der Firma Hewlett Packard

Für die objekt-orientierte Programmierung sind von wachsender Bedeutung:

Corba OTS (Object Transaction Service)
EJB JTS (Enterprise Java Bean Transaction Service)

Unter allen Transaktionsmonitoren hat CICS eine absolut dominierende Position.

Begriffe

Die Begriffe Transaction Monitor, Transaction Server, Transaction Service und Resource Manager werden (grob gesehen) austauschbar verwendet und bedeuten das Gleiche.

Ein Transaction Manager ist eine Schnittstelle zu einem Transaction Monitor