UNIVERSITÄT LEIPZIG

Enterprise Computing
Einführung in das Betriebssystem z/OS

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

WS2012/2013

Transaktionsverarbeitung Teil 1

Einführung

Literatur

J. Gray, A. Reuter:

“Transaction Processing”.
Morgan Kaufmann, 1993.

fast 20 Jahre alt – immer noch das Standard Werk
1070 Seiten !

P. A. Bernstein:

“Principles of Transaction Processing”.
Morgan Kaufmann, 1997.

Wesentlich kürzer

J. Horswill:

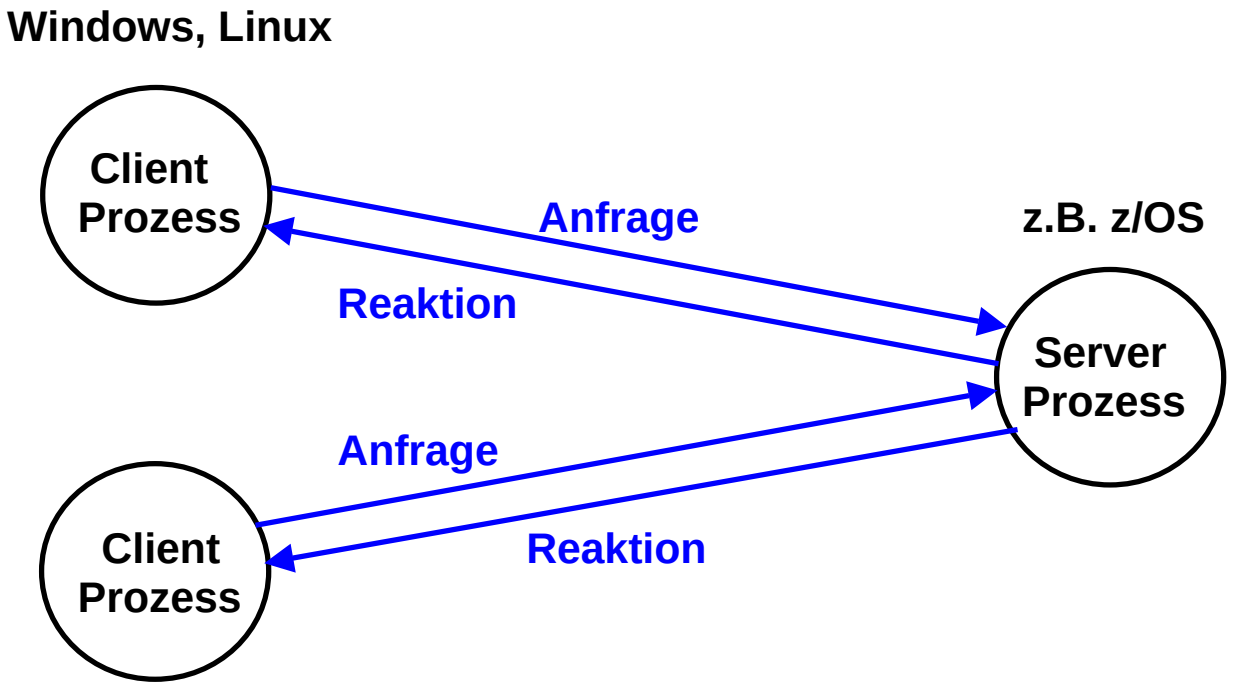
“Designing and Programming CICS Applications”. O’Reilly, 2000

Das beste CICS-spezifische Lehrbuch

Mark Little, Jon Maron, Greg Pavlik:

Java Transaction Processing :
Design and Implementation
Prentice Hall 2004, ISBN 0-13-035290-X

Transaktionale Anwendungen werden in Zukunft in Java entwickelt - vielleicht



Client/Server-Modell

Prozesse auf einem Klienten-Rechner nehmen die Dienstleistungen eines Servers in Anspruch. Ein Server bietet seine Dienste (Service) einer Menge a priori unbekannter Klienten (Clients) an.

Klient:	**Nutzer eines ServerDienstes**
Server:	**Rechner, der Dienst-Software ausführt**
Dienst (Service):	**Software-Instanz, die auf einem oder mehreren Server Rechnern ausgeführt wird**
Interaktionsform:	**Anfrage / Reaktion (Request/Reply)**

Ein Rechner kann gleichzeitig mehrere Serverdienste (Services) anbieten.

Stapelverarbeitung vs. interaktive Verarbeitung

Der größere Teil (etwa 60 %) aller Anwendungen in Wirtschaft und Verwaltung läuft (interaktiv) auf einer Client/Server Konfiguration

Etwa 40 % laufen als Stapelverarbeitungsprozesse (batch processing)

Client/Server Systeme werden auch als „Interaktive“ Systeme, im Gegensatz zur Stapelverarbeitung, bezeichnet.

Die meisten interaktiven Anwendungen werden als „Transaktionen“ ausgeführt.

Datenbanken

Für einfache Interaktionen eines Anwendungsprogramms mit seinen Daten werden „Flat Files“ (z.B. VSAM) eingesetzt. Unter z/OS verringern Access Methods den Programmieraufwand für den Zugriff. Access Methods werden größtenteils vom Betriebssystem Kernel ausgeführt, aber VSAM ist eng in den Benutzer- (Anwendungs-) Prozess integriert.

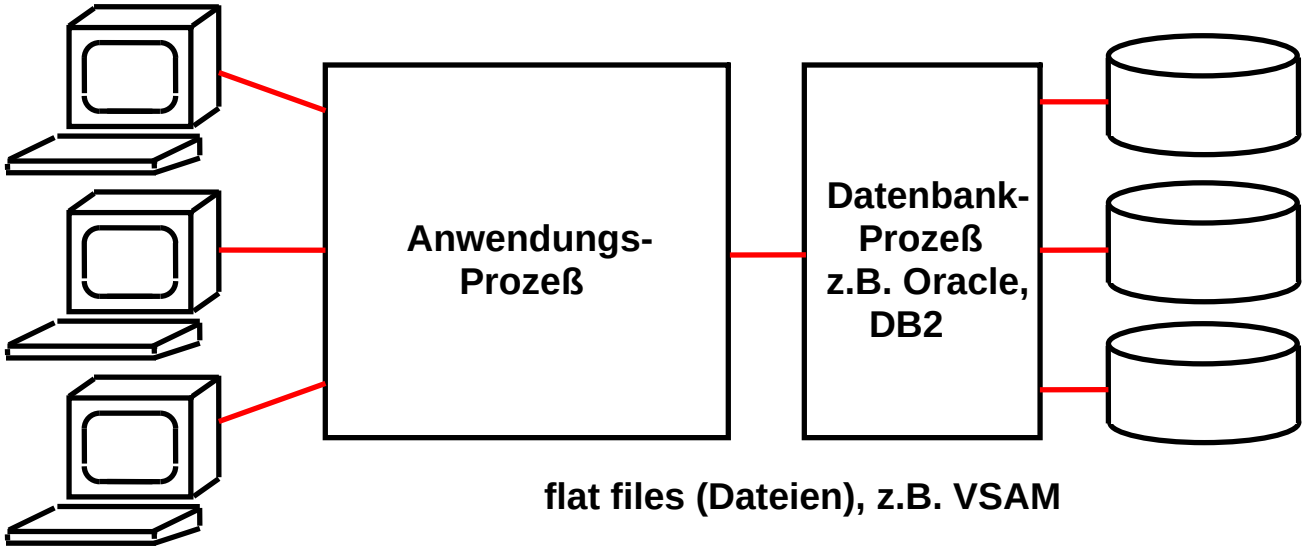
Komplexere Interaktionen unterstützen ein Anwendungsprogramm mit Hilfe eines getrennten „Datenbank“-Prozesses.

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 z/OS DB2).

Auf einem Mainframe dominieren die DB2, IMS und ADABAS Datenbanksysteme.



PC
Registrierkasse
Geldausgabe-
automat

flat files (Dateien), z.B. VSAM

nicht relationale Datenbanken, z.B.
ADABAS
IMS

relationale Datenbanken, z.B.
DB2
Oracle
Ingress
MS SQL
mysql
Sybase

Datenbank Server in einer typische Client/Server Anwendung

Der Server Teil einer Client/Server Anwendung besteht aus zwei Teilen: einem Anwendungsprozess und einem Datenbankprozess.

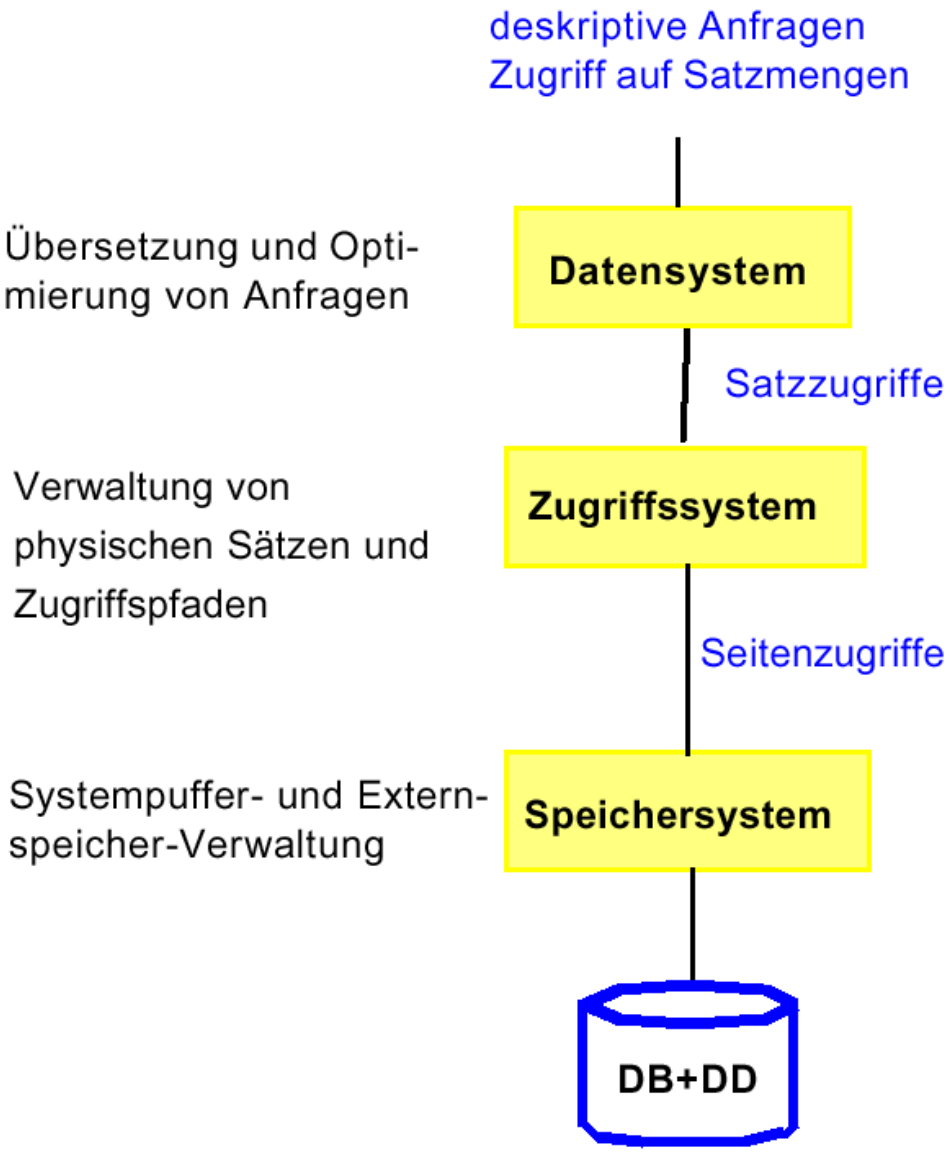
Dargestellt ist eine logische Struktur. 2-Tier, 3-Tier oder n-Tier Konfigurationen unterscheiden sich dadurch, wie diese Funktionen auf physische Server abgebildet werden.

Einige der Alternativen sind:

In einer 2-Tier Konfiguration befindet sich der Anwendungs-prozess auf dem gleichen Rechner wie der Klient.

In einer 3-Tier Konfiguration befinden sich Anwendung und Datenbank als getrennte Prozesse auf getrennten Rechnern.

Komponenten eines Datenbanksystems



Ein Datenbanksystem ist ein eigenständiger Systemprozess, der in einem eigenen virtuellen Adressenraum (Region) läuft (bei z/OS DB2 sind es 3 Regions).

Das Kernelement ist das Zugriffssystem. Es bildet die auf dem Plattenspeicher befindlichen ungeordneten Daten in der Form von Tabellen (z.B. DB2) oder Hierarchien (z.B. IMS) ab.

Das Speichersystem optimiert die Zugriffe auf die Daten eines Plattenspeichers, z. B. durch die Verwaltung von Ein/Ausgabepuffern (Buffer Pool).

Das Datensystem erlaubt Abfragen und Änderungen der Datenbestände mittels einer benutzerfreundlchen Schnittstelle, z.B. SQL.

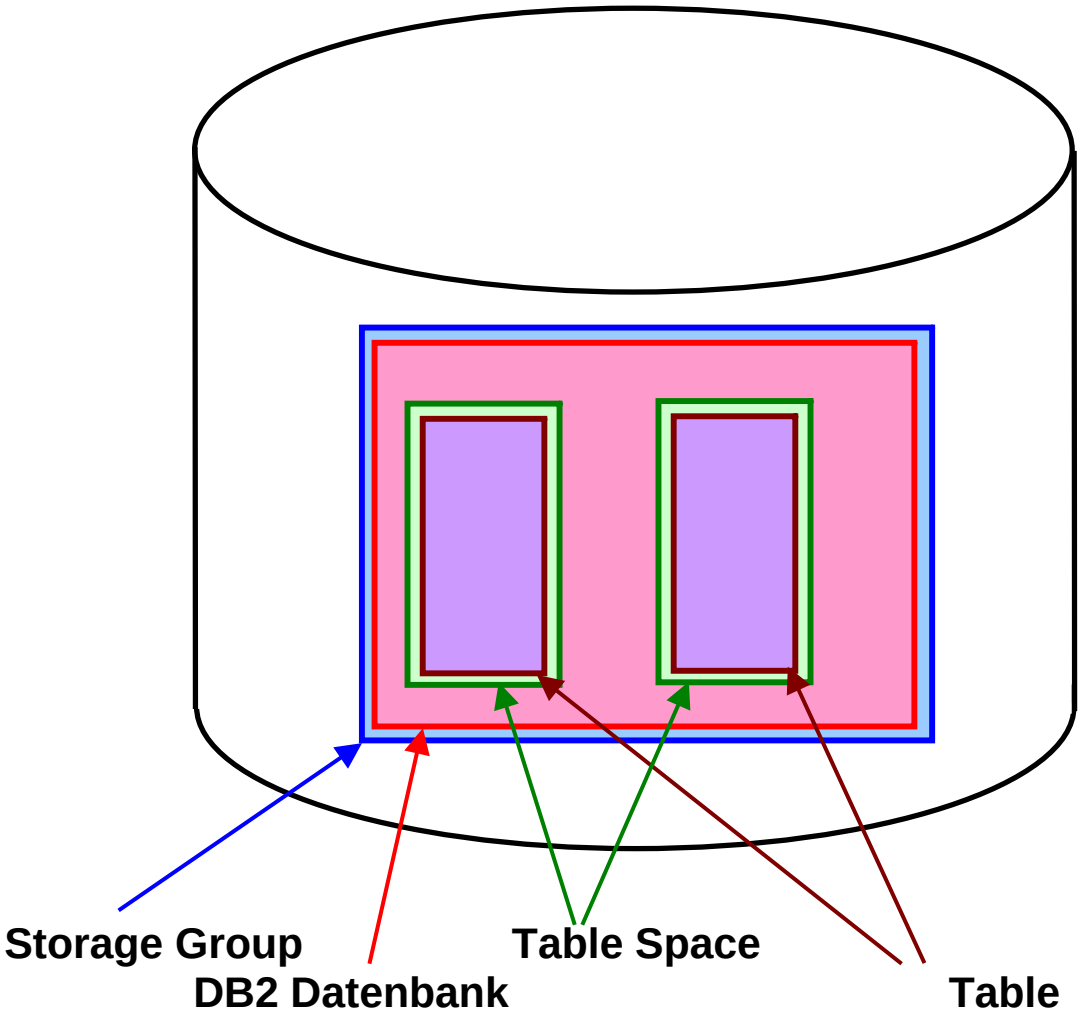
Drei Basiskomponenten eines Datenbanksystems

Das Datensystem übersetzt Anfragen von Transaktionsprogrammen, die z.B. in SQL (Structured Query Language) gestellt werden:

```
Select * FROM PERS
WHERE ANR = ’K55’
```

Das Zugriffssystem setzt die Anfrage in logische Seitenreferenzen um.

Das Speichersystem setzt die logische Seitenreferenzen in physische Seitenreferenzen um. Das DB2 Datenbanksystem speichert alle Daten in „Slots“, deren Größe 4096 Bytes, also der Größe eines Seitenrahmens, entspricht.



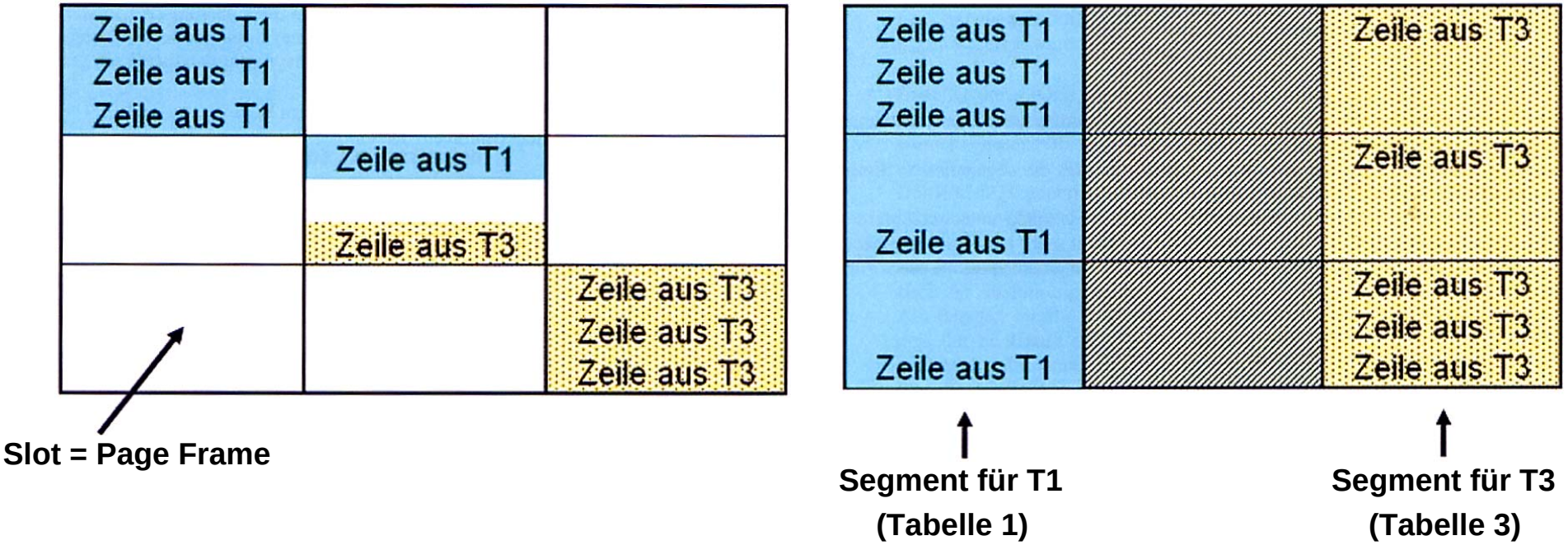
Platzzuweisung für eine DB2 Datenbank

Aus der Sicht des Benutzers besteht eine DB2 Datenbank aus mindestens einer, meistens aber mehreren Tabellen.

Auf einem Plattenspeicher wird hierfür Platz in der Form einer Storage Group angelegt. Diese nimmt eine DB2 Datenbank auf.

Innerhalb der Storage Group werden eine oder mehrere Table Spaces angelegt. Jeder Table Space nimmt eine DB2 Table auf.

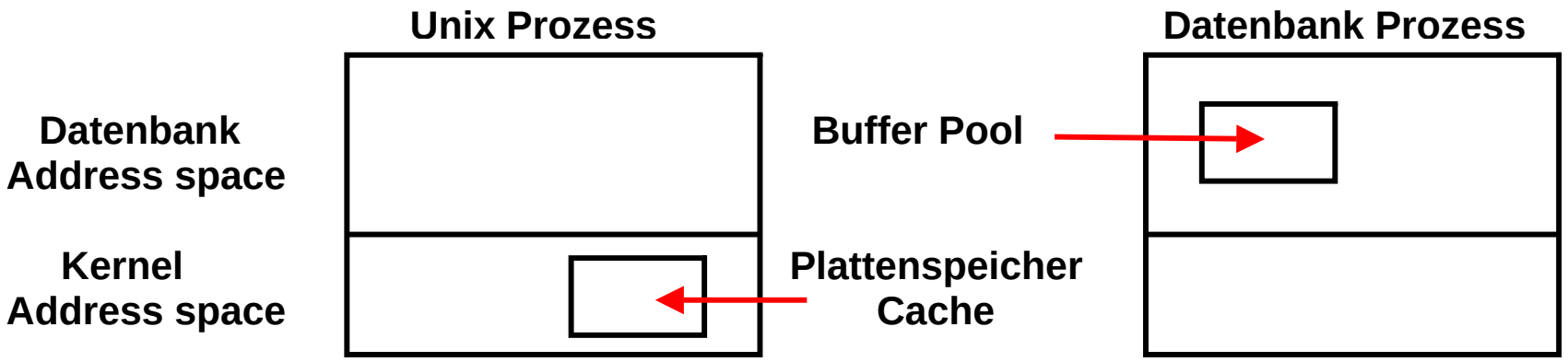
Als Beispiel sei eine DB2 Großrechnerinstallation erwähnt, die mehr als > 50 000 Tables oder Indices für ERP/CRM Anwendungen verwendet.



Simple Table Space **Segmented Table Space**

Ein Table Space besteht aus Slots. Slots sind typischerweise 4096 Bytes groß. Bei einen Zugriff auf die in einer DB2 Tabelle gespeicherten Daten wird ein Slot aus dem Plattenspeicher ausgelesen, und in einen als „Buffer“ bezeichneten 4096 Byte großen Rahmen des realen Hauptspeichers (und die entsprechende Seite des virtuellen Speichers) transportiert.

Ein Table Space kann auch zwei oder mehrere DB2 Tabellen aufnehmen. Bei einem „simple Table Space“ können die Zeilen von zwei Tabellen in dem gleichen Slot untergebracht sein. Der Plattenspeicherbereich von einem einem „segmented Table Space“ ist in mehrere Segmente aufgeteilt. Ein Segment enthält eine bei der Anlage des Table Spaces angegebene Anzahl von Slots (4, 8, ..64). Jedes Segment enthält Einträge von nur einer Tabelle.



Notwendigkeit einer Pufferverwaltung im Datenbanksystem

Unix Systeme verbessern den Zugriff auf Daten, indem sie im Hauptspeicher einen Plattenspeicher-Cache für alle Arten von I/O Daten verwalten. Dieses Konzept existiert nicht unter z/OS.

z/OS DB2 und z/OS IMS speichern eine Anzahl von derzeitig benutzten Daten Items in I/O-„Buffern“ im Hauptspeicher. Ein Buffer speichert in der Regel mehrere Records, z.B. ein VSAM Control Intervall oder einen mehrere Zeilen umfassenden Teil einer DB2 Tabelle. Die Menge aller derzeitig angelegten Buffer wird als Bufferpool bezeichnet.

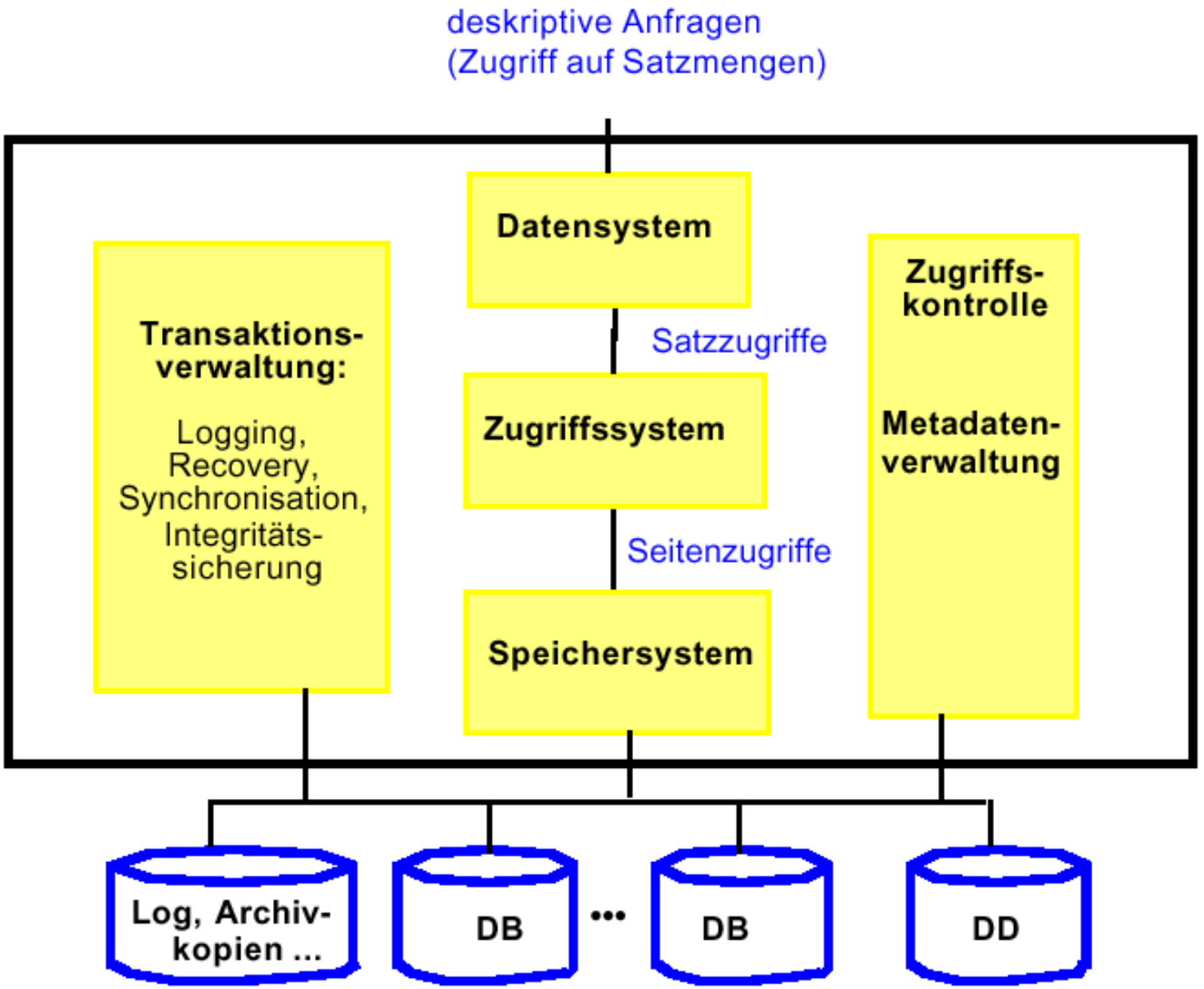
Der Bufferpool befindet sich im Addresspace des Datenbanksystems, welches eine aufwendige Verwaltung seiner I/O Puffer vornimmt. Die Suche im Datenbankpuffer ist in Software implementiert. Typische Referenzmuster in einem Datenbank-System sind:

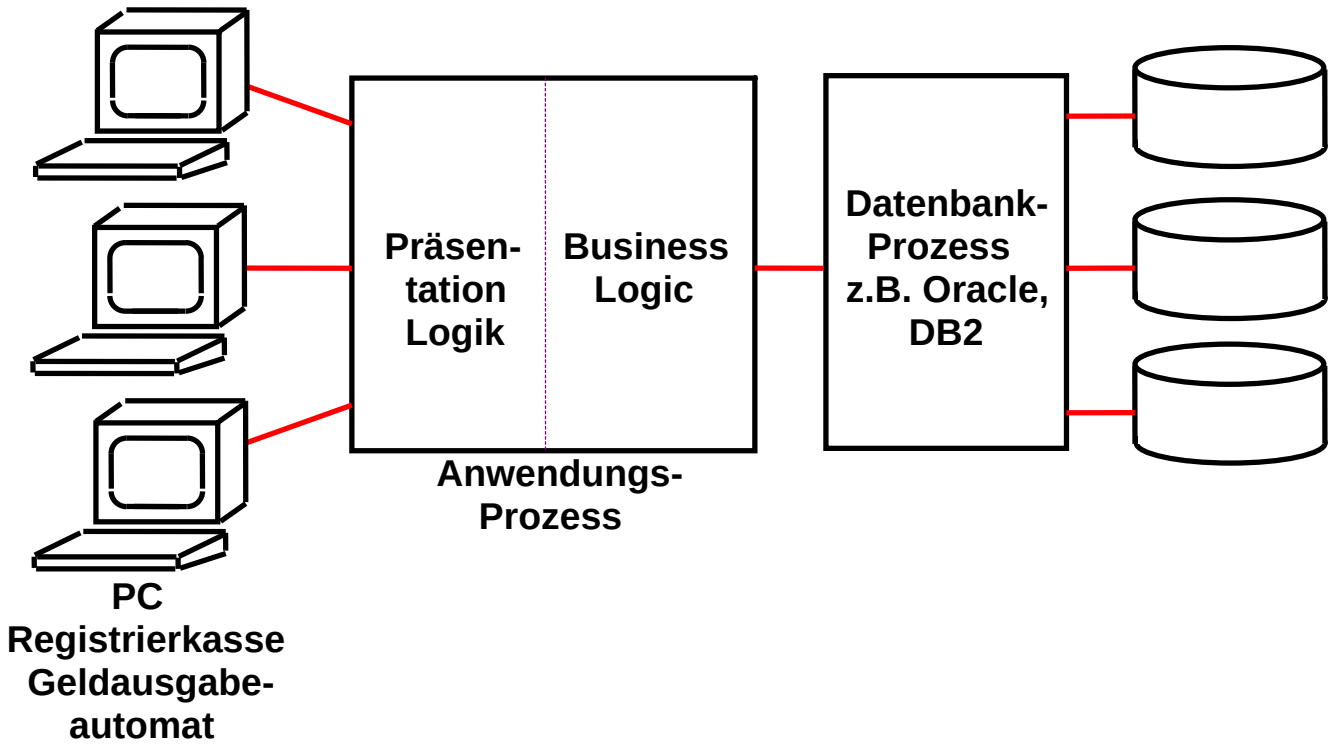
- **Sequentielle Suche (z.B.: Relationen- Scan)**
- **Hierarchische Pfade (z.B.: Suchen über Bäume)**
- **Zyklische Pfade**

Aufbau eines Datenbanksystems

Neben den Komponenten Datensystem, Zugriffssystem und Speichersystem enthält ein Datenbanksystem in der Regel eine Komponente für die Transaktionsverarbeitung, Einrichtungen für die Zugriffskontrolle, Administrationskomponenten sowie Einrichtungen, welche die gespeicherten Daten beschreiben (Metadaten, Data Definition (DD)).

Schließlich sind Einrichtungen für die Verwaltung einer Log-Datei sowie für die Archivierung von Daten vorhanden.





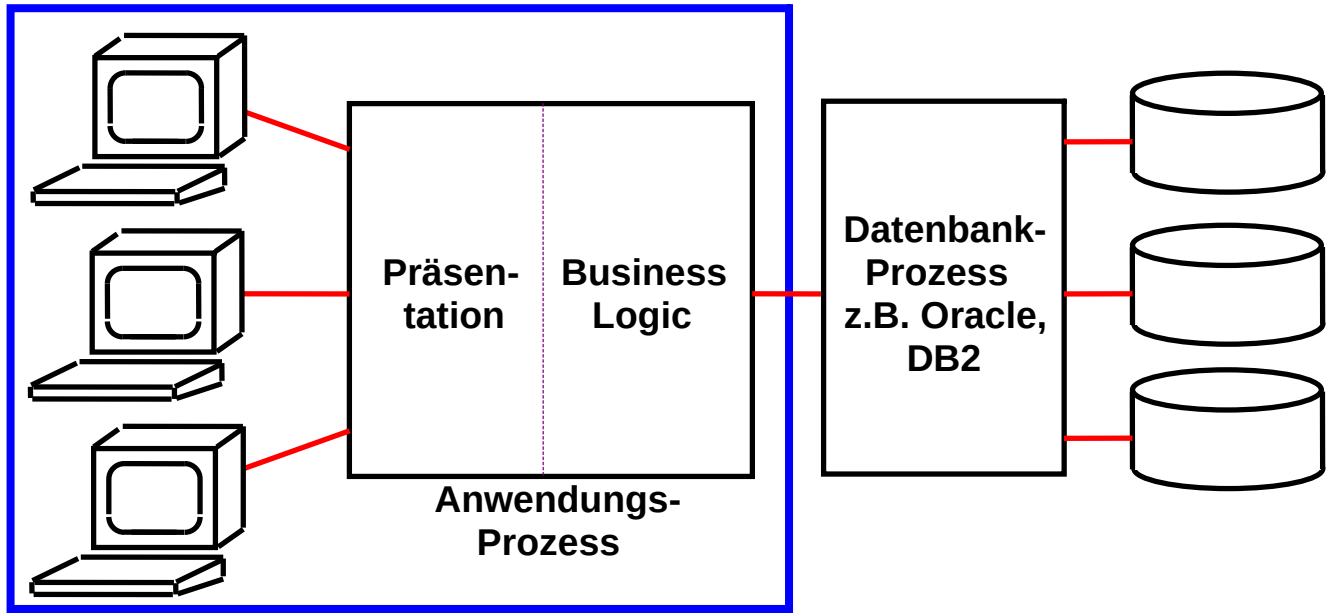
Typische Client/Server Anwendung

Der Anwendungsprozess besteht aus zwei Teilen:

Business Logic (Anwendungslogik) verarbeitet die Eingabedaten des Endbenutzers und erzeugt Ausgabedaten für den Endbenutzer, z.B. in der Form einer wenig strukturierten Zeichenkette (oft als „Unit Record“ bezeichnet).

Präsentationslogik formt die rohen Ausgabedaten in eine für den Endbenutzer gefällige Form um, z.B. in Form einer grafischen Darstellung. Unterschiedliche Klienten können die gleiche Business Logik benutzen um mit unterschiedlicher Präsentationslogik die Ausgabedaten unterschiedlich darzustellen.

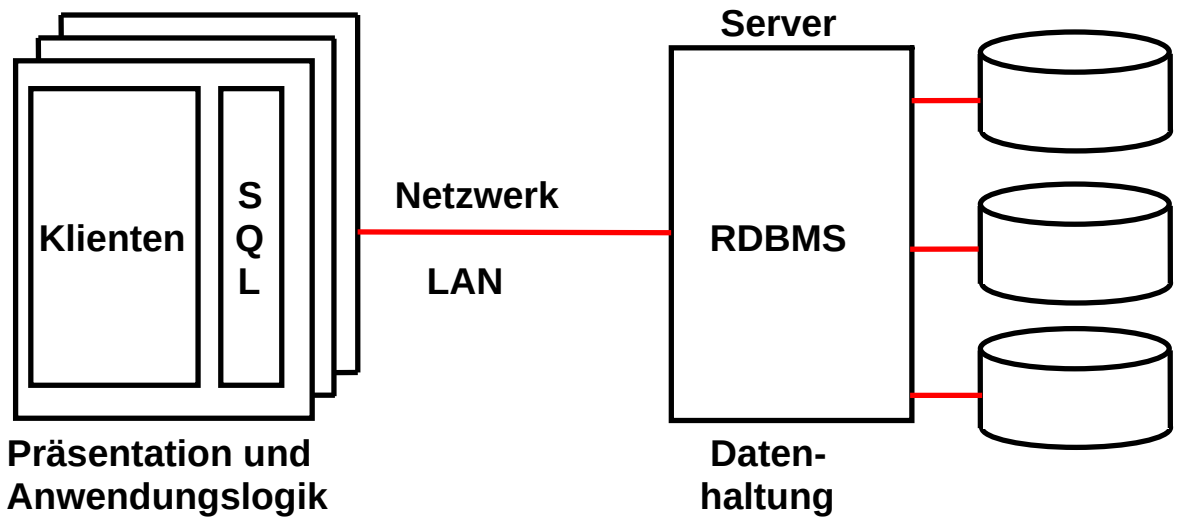
Fat Client



Typische Client/Server Anwendung

In einer als „Fat Client“ bezeichneten Konfiguration laufen Business Logic (Anwendungslogik) und Präsentationslogik beide auf dem Klienten. Auf dem Server läuft nur der Datenbankprozess.

Dies ist eine häufig in kleinen Betrieben anzutreffende Konfiguration, auch als „2-Tier“ bezeichnet. z.B. teilen sich 12 PC Bildschirm-Arbeitsplätze einer Abteilung einen Datenbankserver.



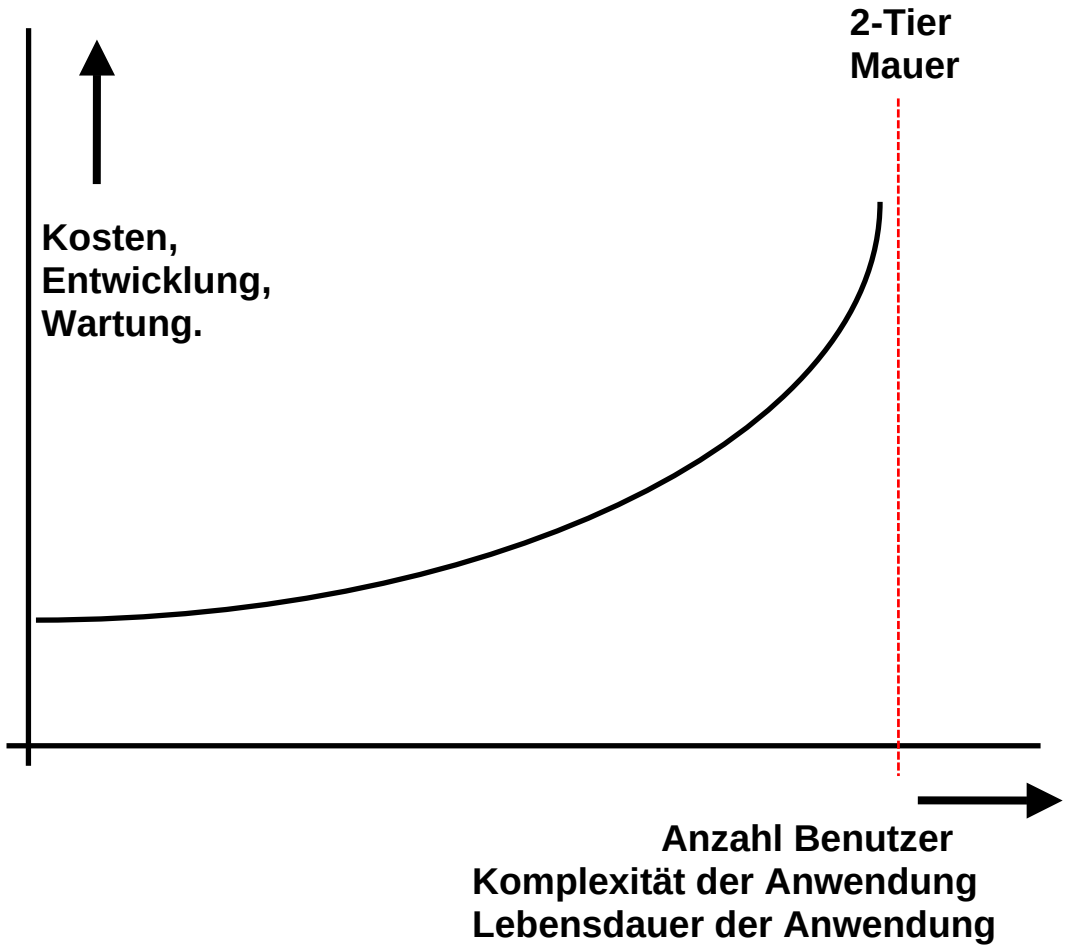
2-Tier Client/Server Architektur

Zweistufige Client/Server Architektur

Auf dem Klientenrechner bedindet sich eine SQL-Client Komponente, die in der Lage ist, SQL Anfragen über das Netz an den Datenbankserver zu senden. Typische Umgebungen haben folgende Eigenschaften:

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

Die Anwendungsentwicklung verwendet häufig Power Builder oder Visual Basic. 2-Tier Konfigurationen werden häufig in reinen Microsoft Umgebungen bei kleinen Unternehmen eingesetzt. Diese Konfiguration skaliert sehr schlecht; deshalb ist sie in größeren Unternehmen nur sehr selten anzutreffen.

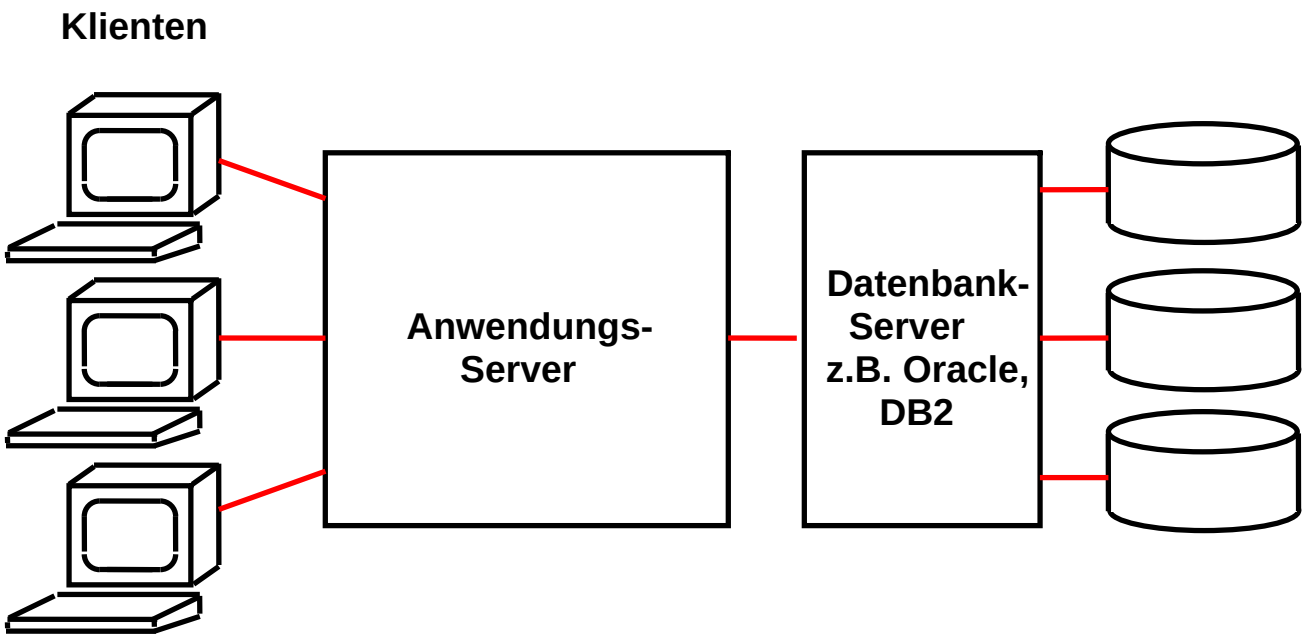


Die 2-Tier Konfiguration ist sehr populär, leicht zu programmieren, skaliert aber schlecht, ist nur bis zu einer maximalen Größe möglich (als 2-Tier Mauer bezeichnet). Gründe sind:

- **Datenvolumen auf dem Netzwerk**
- **Datensatz Lock Contention. Was passiert, wenn 2 Klienten gleichzeitig auf den gleichen Datensatz zugreifen ?**
- **Verteilung (Administration) der Klienten Software auf einer Vielzahl von Klienten Rechnern**

Skalierung der 2-Tier Architektur

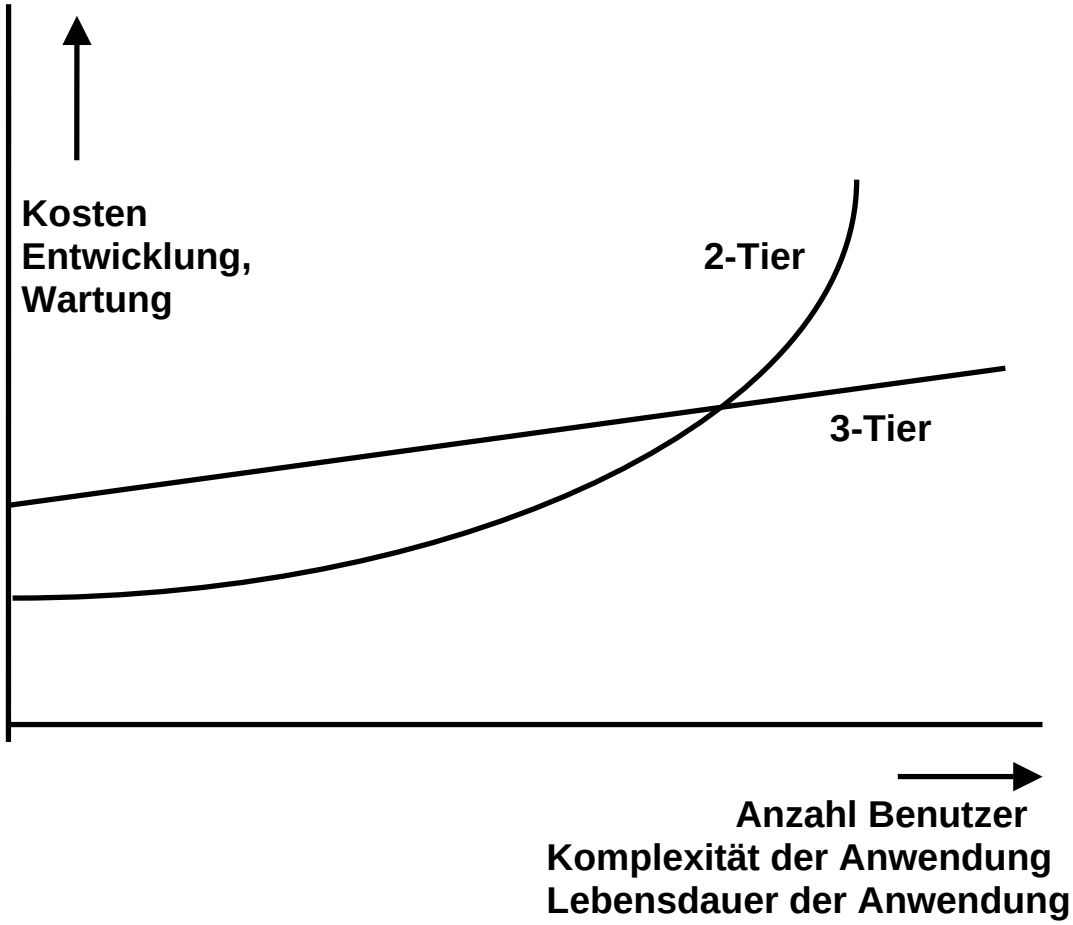
Man spricht von einer „2-Tier Mauer“. Jenseits einer bestimmten Belastungsgrenze ist ein vernünftiger Betrieb nicht mehr möglich, z.B. 100 einfache oder sehr viel weniger komplexe Transaktionen pro Sekunde.



3-Tier Client/Server Architektur
Dreistufige Client/Server Architektur

Bei der 3-Tier Konfiguration existiert ein von den Klienten getrennter Anwendungsserver, wobei entweder zwei getrennte Rechner für Anwendung und Datenbank benutzt werden, oder alternativ der Anwendungs- und der Datenbankprozess auf dem gleichen Rechner laufen. Letzteres ist eine typische z/OS Konfiguration. Für den Anwendungsserver existieren zwei Arten:

- **Präsentationslogik läuft auf dem Klienten. Business Logik läuft auf dem Anwendungsserver. Beispiel: SAPGUI des SAP R/3 Systems**
- **Präsentationslogik und Business Logik laufen auf dem Anwendungsserver Beispiel: Servlets, Java Server Pages und Enterprise Java Beans.**



Die 3-Tier Konfiguration skaliert wesentlich besser als die 2-Tier Konfiguration:

- **Service Anforderungen generieren weniger Datenvolumen**
- **Anwendungsserver optimiert Lock Contention**
- **Mehrfache Anwendungs-server sind möglich (Anwendungsreplikation)**
- **Zugriffskontrolle erfolgt auf Service Basis.**

Skalierung der 3-Tier Architektur

Bei sehr geringer Belastung ist die 2-Tier Konfiguration überlegen – im Mainframe Bereich ein nicht sehr realistischer Fall.

Transaktionen

Transaktionen sind Stapelverarbeitungs- oder 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(en) unverändert läßt

Die Datei/Datenbank bleibt in einem konsistenten Zustand: Entweder der Zustand vor Anfang, oder der Zustand nach erfolgreichem 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. Dieser Vorgang wird als Rollback, Backout oder Abort bezeichnet; die Begriffe sind Synonyme.

Wird eine Transaktion abgebrochen, werden keine Daten abgeändert.

ACID Eigenschaften

Eine Transaktion ist ein Anwendungsprozess, der die ACID Eigenschaften implementiert:

Atomizität (Atomicity)

Konsistenzerhaltung (Consistency)

Isolation

Dauerhaftigkeit (Durability)

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 die transaktionsgeschützten Daten des Systems von einem konsistenten Zustand in einen anderen konsistenten Zustand.

Diese Eigenschaft garantiert, dass die Daten der Datenbank nach Abschluss einer Transaktion schemakonsistent sind, d. h. alle im Datenbankschema spezifizierten Integritätsbedingungen erfüllen.

Daten sind konsistent, wenn sie nur durch eine Transaktion erzeugt oder abgeändert 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.

Dauerhaftigkeit (Durability)

Bei Einhaltung der ACID Eigenschaften wird angenommen, dass

- **alle Daten auf Plattenspeichern (oder Solid State Disks) mit RAID oder vergleichbaren Eigenschaften, oft mit zusätzlicher Spiegelung, gespeichert werden und**
- **alle Fehlerfälle wie Plattenspeichercrasches oder andere katastrophale Ausfälle unbeschadet überstehen.**

Als Persistenz bezeichnet man eine Datenspeicherung, welche die Durability Bedingung der ACID Eigenschaften erfüllt.

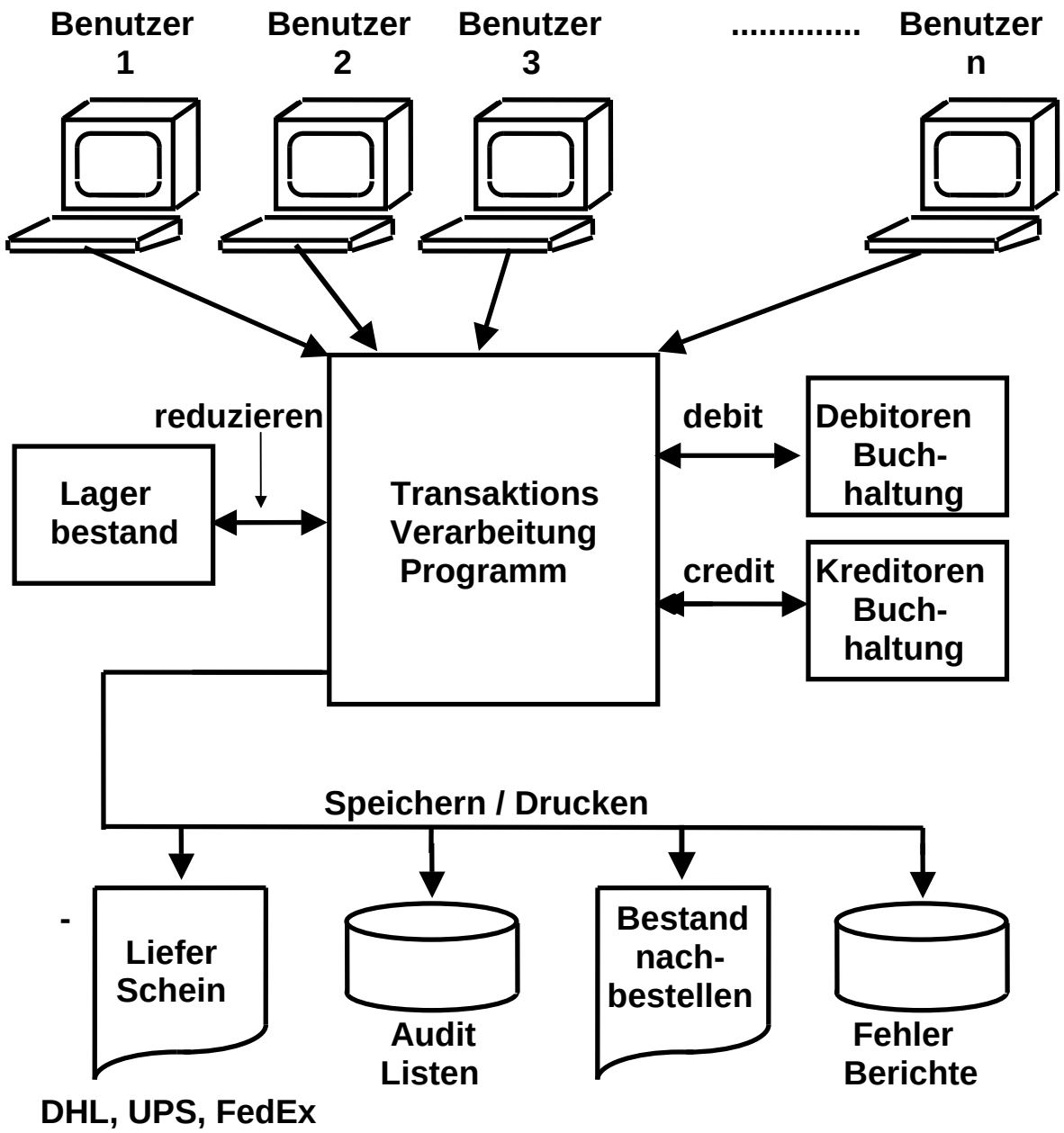
Intel Transactional Synchronization Extentions

Intels nächste Prozessorgeneration Haswell wird als zweiter Prozessor nach IBMs BlueGene/Q Transactional Memory in Hardware unterstützten.

TSX: Transactional Synchronization Extentions, so heißt die Befehlserweiterung, deren Berschreibung man jetzt als bei Intel in der Neufassung der "Intel Architecture Instruction Set Extensions, Programming Reference" herunterladen kann (PDF). Mit TSX soll die Synchronisation zwischen Threads verbessert und vor allem beschleunigt werden. Threads müssen sich bei Zugriffen auf gemeinsame Bereiche miteinander synchronisieren, was viel Zeit kosten kann. Bei Transactional Memory arbeiten die Threads stattdessen zunächst einmal unsynchronisiert und erst beim "Commit" wird überprüft, ob es einen Konflikt gegeben hat. In dem Fall muss die Transaktion verworfen und wiederholt werden, aber das ist vergleichsweise selten.

Konzepte in Software gibt es schon lange, doch die sind bislang zumeist zu ineffizient. Intels TSX bieten dem Programmierer zwei Schnittstellen: Hardware Lock Elison (HLE) mit den neuen Präfixen XACQUIRE und XRELEASE sowie eine Variante namens Restricted Transactional Memoyr RTM, die die neuen Instruktionen XBEGIN, XEND und XABORT bietet. HLE ist die klassische Form, die sich in bestehende Programmkonzepte mit "mutual exclusion" leicht einbringen lässt, RTM ist flexibler, erfordert aber eine Neufassung des Konzepts. (as)

http://www.heise.de/newsticker/meldung/Transactional-Memory-fuer-Intels-Haswell-Prozessor-1432877.html
10.02.2012



Beispiel für eine Transaktions-verarbeitungs-anwendung: Auftragseingang-Bearbeitung

Beispiel für eine Transaktionsverarbeitungsanwendung: Auftragseingang-Bearbeitung

Das Beispiel zeigt das Auftragseingangssystem eines Handelsunternehmens (z. B. Otto Versand), das Aufträge ausliefert, Bezahlungen aufzeichnet, den Orderstatus überprüft und den Warenbestand im Lager überwacht. Die Benutzer sind Kunden, die sich mittels Ihres PCs über das Internet einloggen und jeweils eine Transaktion in der Form eines Bestellvorgangs auslösen. Der Code für ein Beispiel dieser Art ist als ein weit verbreitetes Benchmark (TPC-C) verfügbar (http://www.tpc.org/tpcc/default.asp).

Bei der Bestellung wird der Lagerbestand des bestellten Artikels (z.B. Herrenpullover Gr.43, Bestell Nr. 4711) um die bestellte Menge verringert. Die Debitoren- und Kreditoren-Buchaltung nehmen Finanzbuchungen vor. Ein Paketdienst (z.B. DHL, UPS, Fedex usw.) erhält einen elektronischen Lieferschein und Auslieferungsauftrag. Ein Zulieferant erhält eine Nachbestellung um den Bestand des bestellten Artikels wieder aufzufüllen. Es werden Audit-Listen und Fehlerberichte erstellt.

Eine teilweise ausgeführte Transaktion wird evtl. wieder rückgängig gemacht, z.B. weil der bestellende Kunde seinen Kreditrahmen überzogen hat oder die angegebene Versandanschrift von keinem Paketdienst angefahren werden kann.

Transaktionssysteme

Etwa 80 % aller betrieblichen Aanwendungen werden als Transaktionen verarbeitet. Die transaktionalen Verarbeitungsabläufe erfolgen teilweise interaktiv als Client/Server Anwendungen, teilweise als Stapelverarbeitung (Batch Processing).

<u>Interaktive Beispiele:</u>

Auskunftsysteme
Buchungssysteme (z.B. Flugplatzreservierung)
Geldausgabeautomaten
Auftragsbearbeitung, Buchbestellung bei Amazon
Angebot bei eBay abgeben

<u>Stapelverarbeitung Beispiele:</u>

Monatliche Lohn/Gehaltsabrechnung
Jährliche Bilanz erstellen

Das Verhältnis der Rechnerbelastung interaktiv zu Stapel beträgt bei den meisten Unternehmen etwa 60% zu 40%.