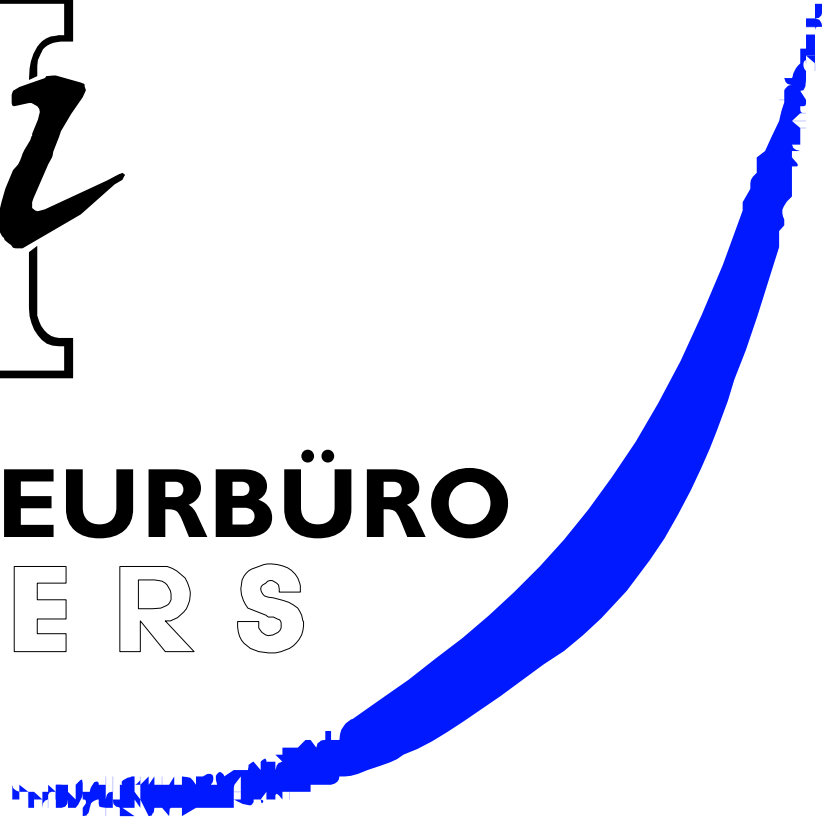




**IBL INGENIEURBÜRO**  
LETTERS



IBL Ingenieurbüro Letters GmbH, Max-Lang-Straße 24, 70771 Leinfelden, Tel: 0711/90374-0, Fax: 0711/90374-20



IBL INGENIEURBÜRO  
LETTERS

Java Forum Stuttgart '99 Stuttgart

# Architektur-Entwurfsmuster für die Realisierung verteilter Anwendungen mit Java und CORBA

Dr. Gerhard Wanner

E-Mail: [gwanner@ibl.de](mailto:gwanner@ibl.de)

# Agenda

## Hinführung

- Architektur: Vorgaben
- Architektur: Übersicht
- Architektur: Details

## Architektur-Entwurfsmuster

- Skalierung, Scheduling
- Datenbankzugriff
- Exceptionhandling in verteilten Systemen
- Objektfreigabe in verteilten Systemen

## Zusammenfassung

# Architektur: Vorgaben



IBL INGENIEURBÜRO  
LETTERS

## Systeme haben einen hohen Interaktionsgrad

- Bearbeitungsdauer eines Use Cases liegt im Minutenbereich (und damit auch die Dauer der Transaktionen)
- Use Cases mit komplexen Benutzerschnittstellen (z.B. Exploreransicht der zu bearbeitenden Businessobjekte)
- Geschachtelte Bearbeitungsschritte mit der Möglichkeit zum Zurücksetzen bereits getätigter Teilprozesse.

## Systeme mit echten Thin Clients

- Sowohl Businessobjekte als auch Use Case-Objekte residieren auf dem Server
- Clients enthalten lediglich GUI-Objekte sowie Proxys auf Use Case- und Businessobjekte

## Folge dieser Voraussetzungen

- Die Systeme können nicht als typische Transaktionssysteme (mit sehr kurzen Transaktionen) realisiert werden
- Es ist kein "formularbasierter" Ansatz möglich

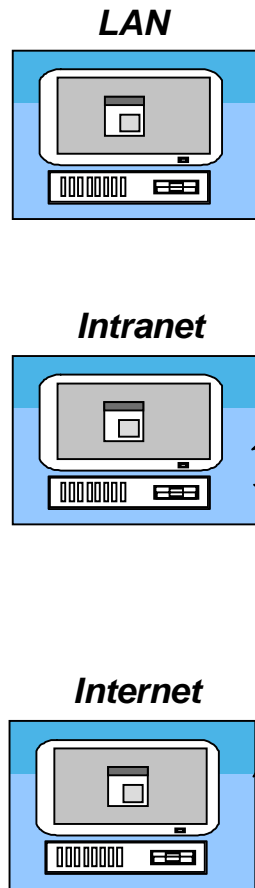
# Architektur: Übersicht

- Systemarchitektur, Verteilung

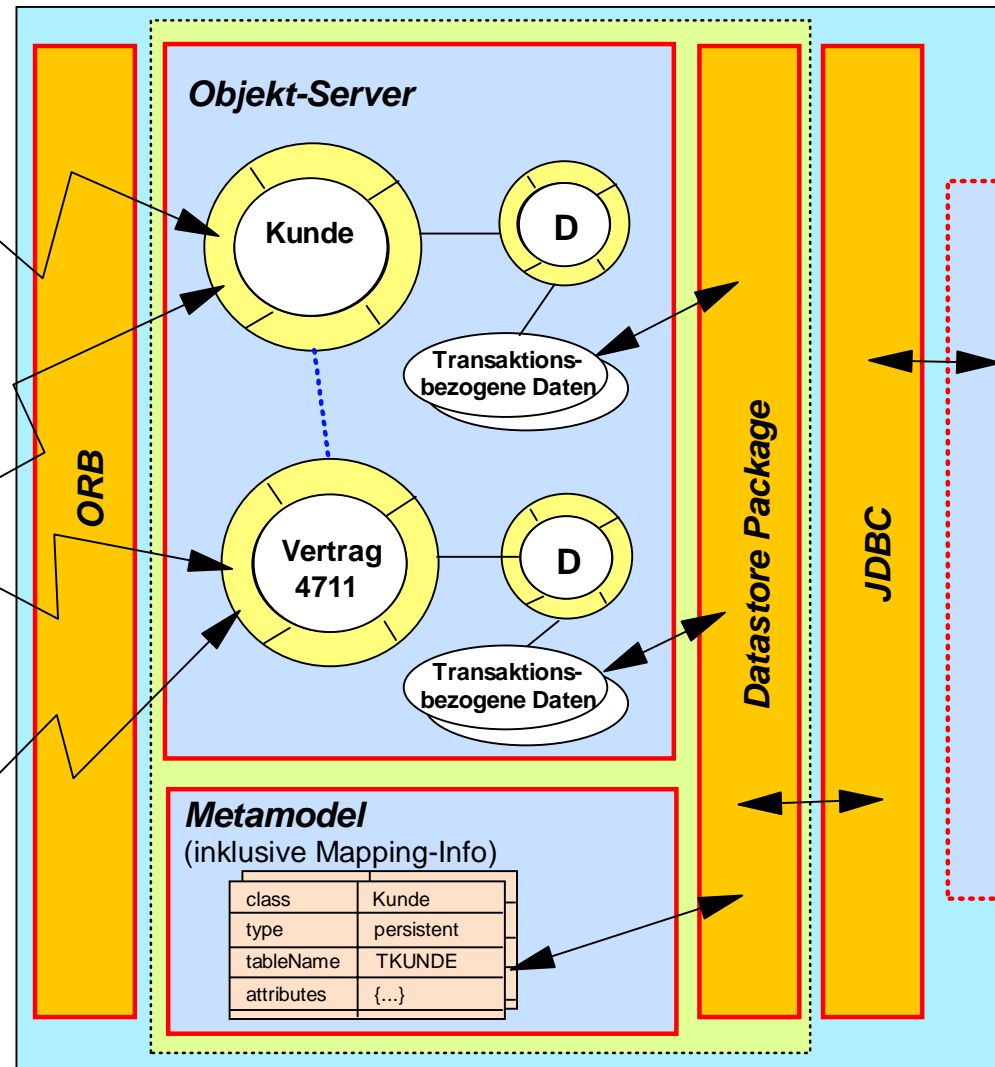


IBL INGENIEURBÜRO  
LETTERS

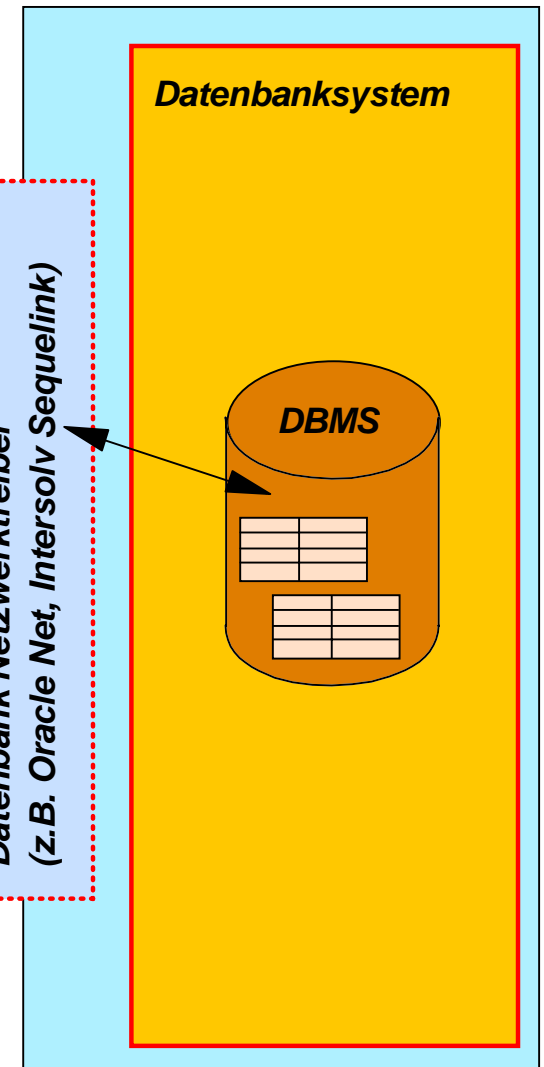
## Clients



## Applikationsserver



## Datenbankserver

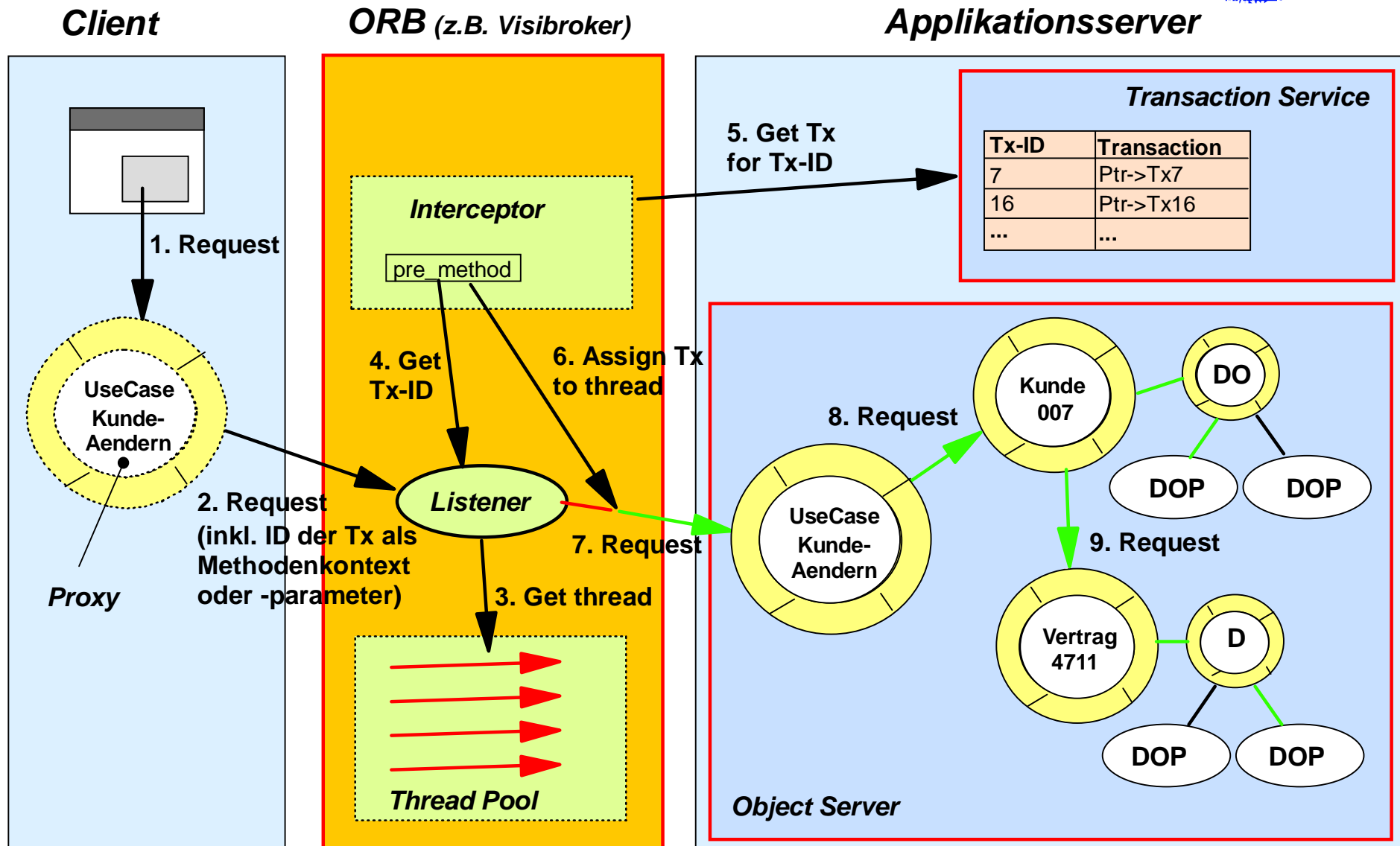


# Architektur: Details

- Zuordnung der Transaktion zum Thread



IBL INGENIEURBÜRO  
LETTERS



# Skalierung (1)

Problem; Lösungsvarianten



IBL INGENIEURBÜRO  
LETTERS

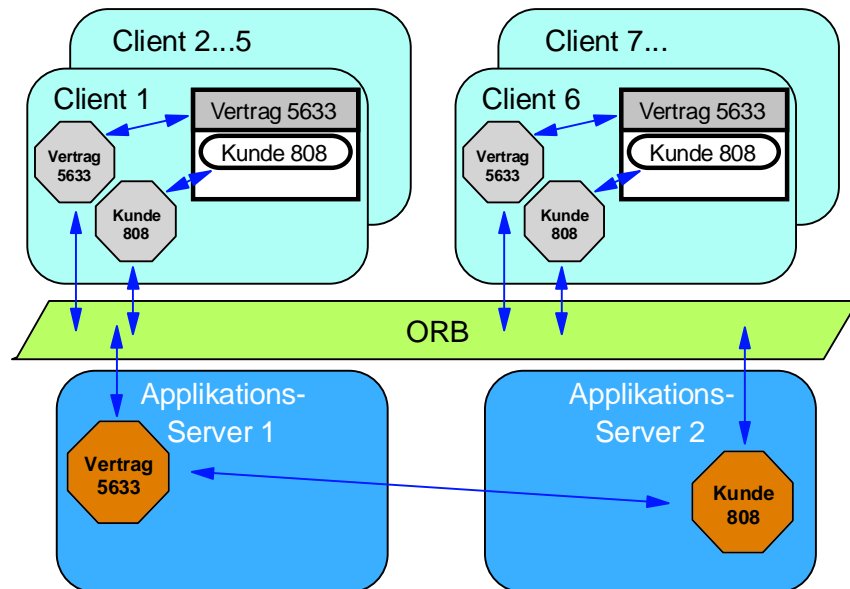
## Problem

- Ein einzelner Applikationsserver ist auch bei massivem Einsatz von Hardware nicht ausreichend:
  - Verfügbarkeit: Der Absturz des (einzigen) Applikationsservers führt zum Komplettausfall des Systems.
  - Skalierung der VM: zu viele Threads lassen die Performance durch Thread-Switching einbrechen.

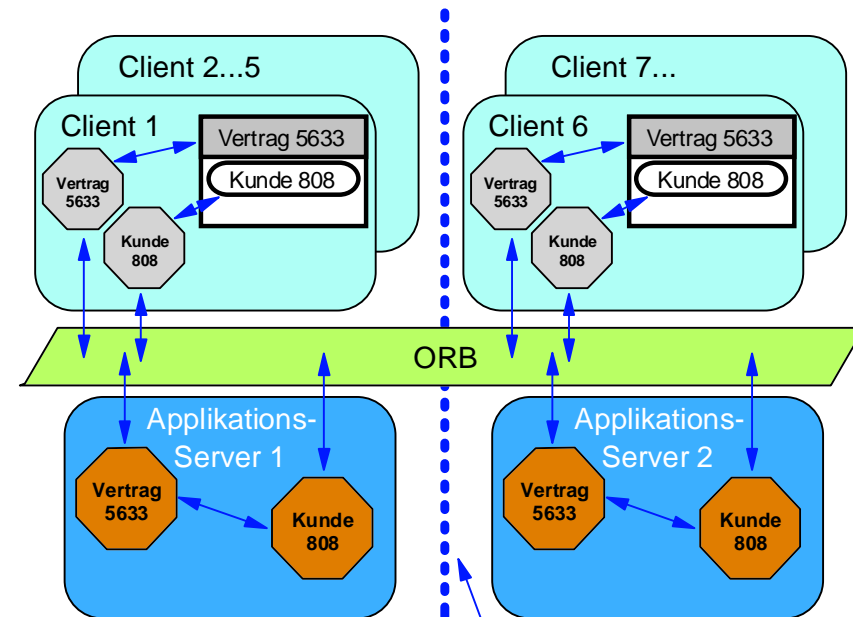
## Lösungsansatz

- Mehrere Applikationsserver
- Zwei Hauptvarianten:

### Singuläre Objekte bzgl. des Gesamtsystems



### Singuläre Objekte pro Applikationsserver



Sichtbarkeitsgrenze

# Skalierung (2)

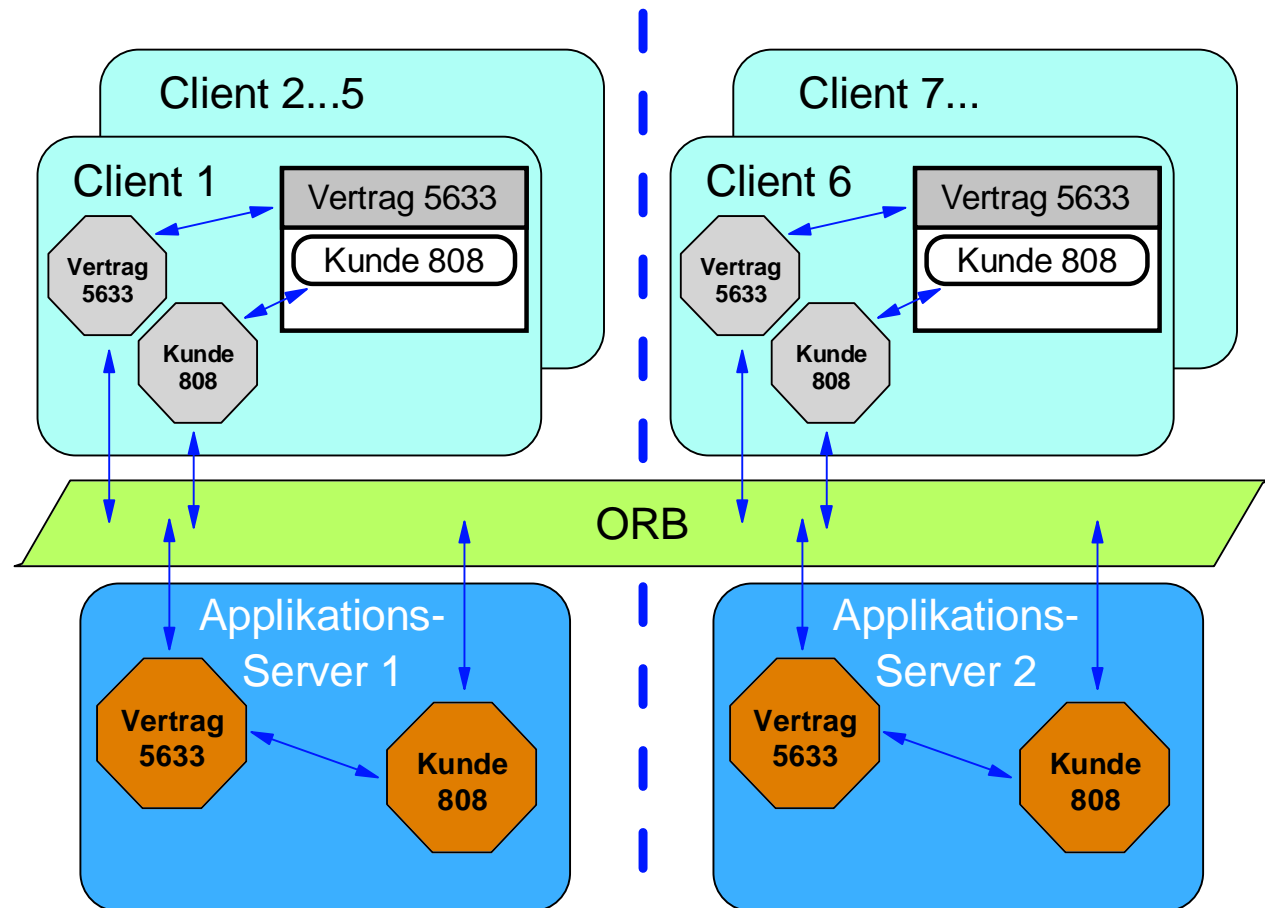
## Mehrere Applikationsserver - Variantenauswahl



IBL INGENIEURBÜRO  
LETTERS

### Lösung

- Mehrere Applikationsserver bedienen jeweils eine bestimmte Anzahl von Clients
  - Jeder Applikationsserver ist für sich ein voll funktionsfähiges System
- Die Vorteile sind
  - Applikationsserver kennen sich untereinander nicht und haben auch keine Abhängigkeiten
  - Behebt die genannten Probleme (Skalierung, Verfügbarkeit)
  - Verteilung der Applikationsserver auf verschiedene physikalische Server problemlos
  - Fat-Clients können parallel eingebunden werden (wegen der notwendigen Synchronisation auf Datenbankebene)
  - Absturz eines Applikationsservers betrifft nur eine begrenzte Anzahl von Clients
  - Kein "Single Point Of Failure"
  - Keine verteilten Transaktionen notwendig
- Die Nachteile sind
  - Mehrere Instanzen desselben Objekts in verschiedenen Applikationsservern möglich, d.h. Notifikation bzw. Nachlesen nötig
  - Synchronisation auf Datenbankebene notwendig (optimistisch oder pessimistisch je nach Anforderung)



# Skalierung (3)

## Scheduling



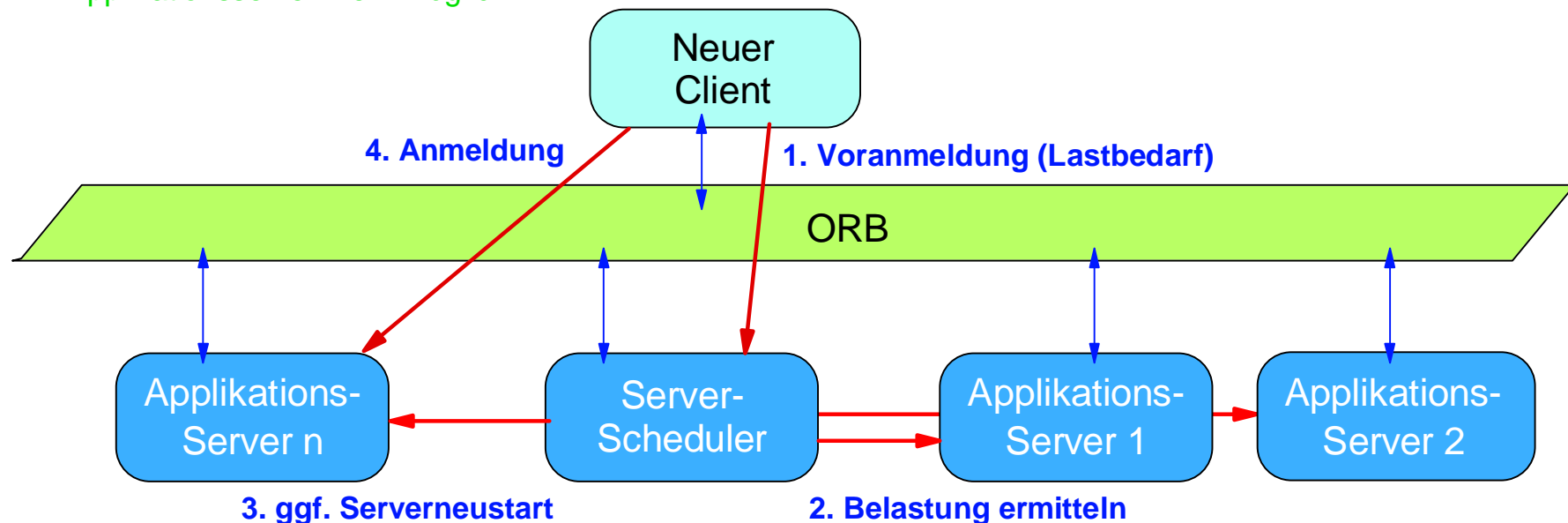
IBL INGENIEURBÜRO  
LETTERS

### Problem

- Wie kann die Last auf mehrere Server verteilt werden?
  - Erkennen von ausgelasteten Servern
  - Spezielle Clients mit hohen Anforderungen

### Lösung

- Scheduler für die Applikationsserver
  - Scheduler ordnet neu gestartete Clients dem am wenigsten belasteten Applikationsserver zu
  - Die Belastung von Applikationsservern wird berechnet aus den Parametern: Anzahl Clients, Anzahl offene Transaktionen, Anzahl instanzierter Objekte, Anzahl Threads, Speicherverbrauch...
  - Scheduler startet bei Bedarf neue Applikationsserver, auch auf anderen Maschinen
  - Clients können ihren Leistungsbedarf beim Anmelden angeben (wichtig z.B. für Batch-Use-Cases)
  - Lastverteilung ggf. auch auf Transaktionsebene denkbar. Allerdings sind dann keine langlebenden Objekte auf dem Applikationsserver mehr möglich!



# Datenbankzugriff

## Connection-Pooling



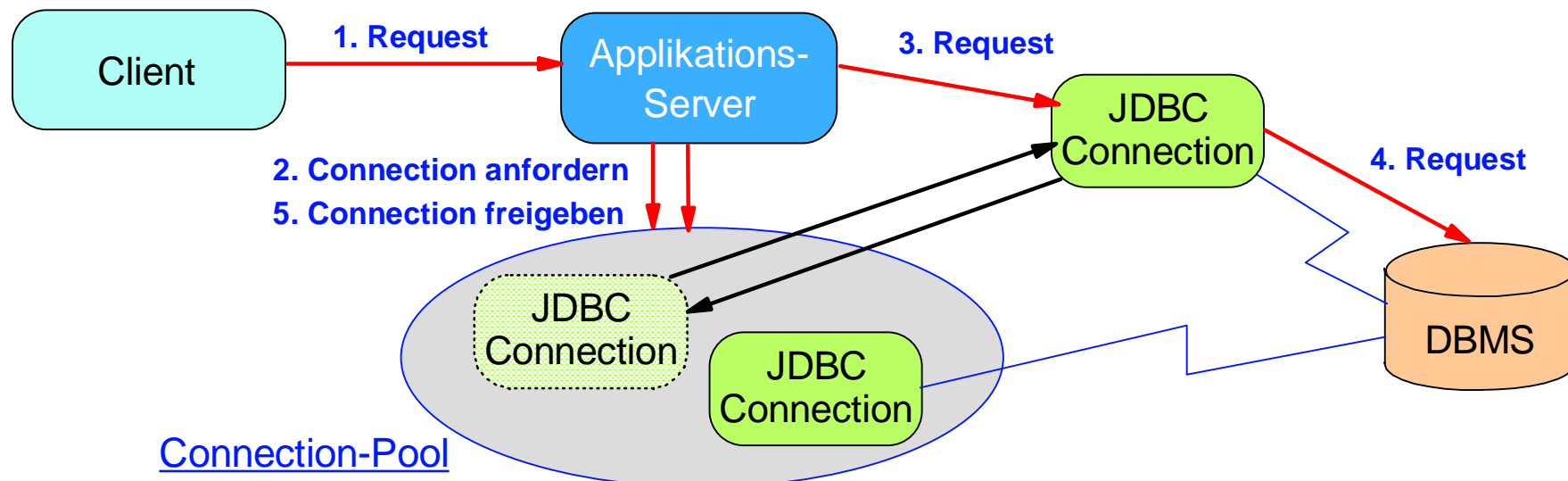
IBL INGENIEURBÜRO  
LETTERS

### Problem

- Gleichzeitig gehen mehrere Client-Requests ein, welche quasi-parallel abgearbeitet werden müssen
- Öffnen und Schließen von JDBC-Connections ist zeitaufwendig

### Lösung

- Öffnen mehrerer JDBC-Connections beim Start des Applikationsservers und vorhalten im "Connection-Pool"
- Wird eine Connection benötigt kann sie dem Pool entnommen und nach der Abarbeitung des Request diesem wieder hinzugefügt werden
- Ist im Pool keine Connection mehr vorhanden, werden dynamisch neue Connections eröffnet



# Datenbankzugriff

## Client-Abbruch



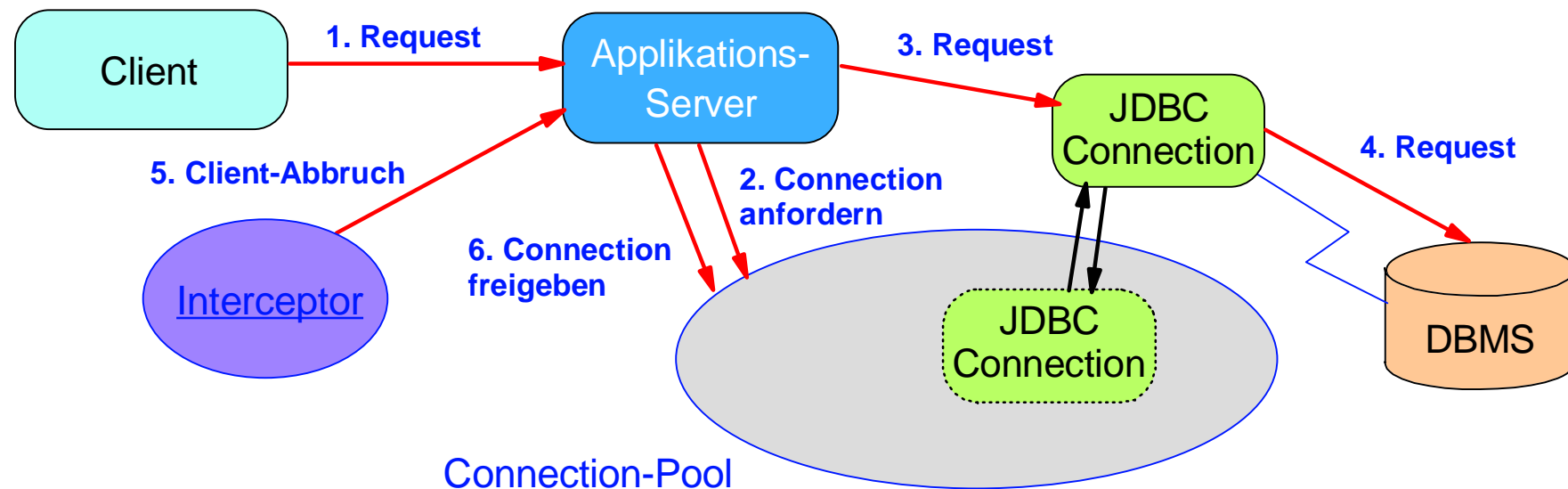
IBL INGENIEURBÜRO  
LETTERS

### Problem

- Bricht ein Client ab, so bleiben benutzte Connections offen
- Dies führt dazu, daß vom Client benutzte Connections nicht mehr in den Pool der verfügbaren Connections zurückgegeben werden
- Das Problem tritt insbesondere dann auf, wenn große Ergebnismengen per Cursor verarbeitet werden

### Lösung

- Abfangen des Client-Abbruchs über Interceptoren beim ORB
- Die dort aufgerufene Methode (*client\_aborted(...)*) setzt alle Transaktionen für den betroffenen Client zurück und schließt danach auch alle Connections



# Datenbankzugriff

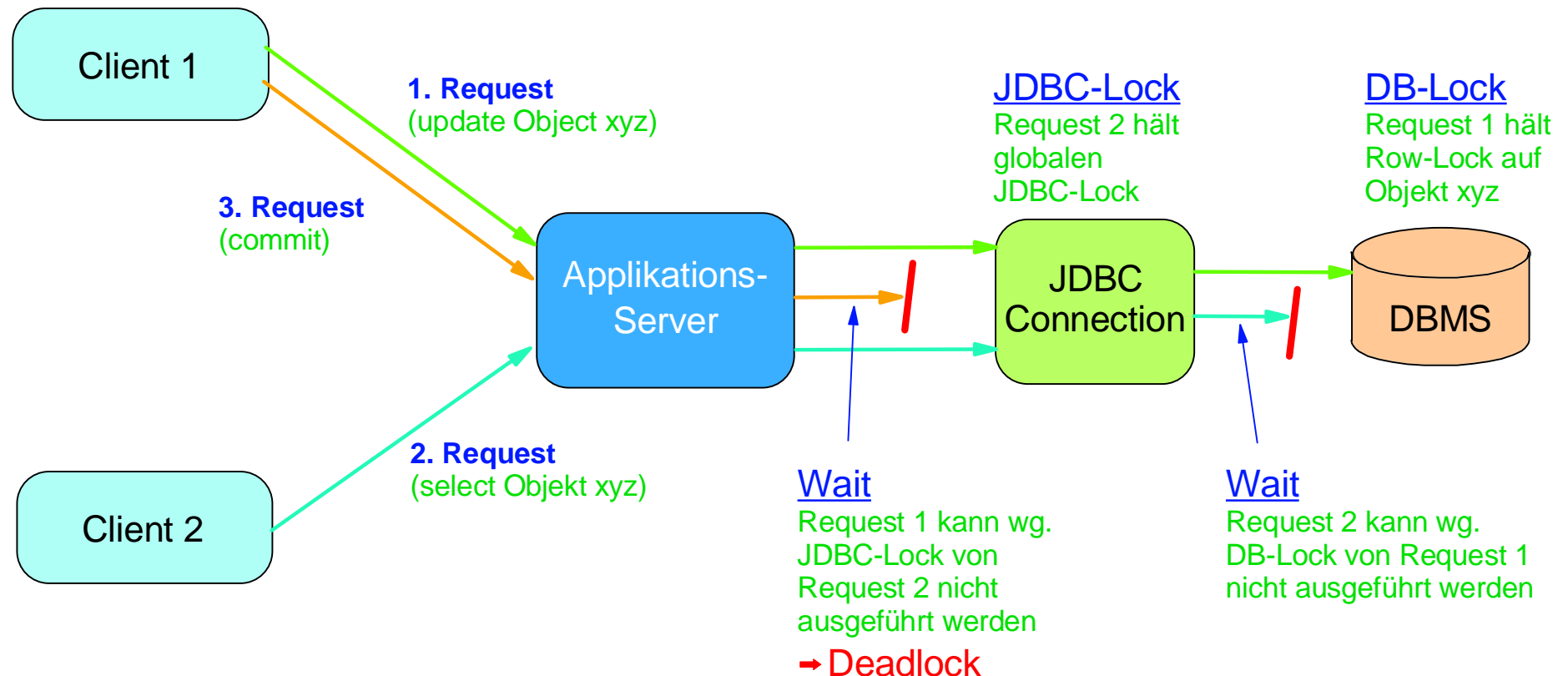
## Serialisierung - Problem



IBL INGENIEURBÜRO  
LETTERS

### Problem

- Einige JDBC-Driver enthalten Thread-Synchronisierungen, so daß immer nur eine Connection tatsächlich benutzt werden kann
  - Verursacht Durchsatzprobleme, da immer nur ein Client-Request datenbankseitig abgearbeitet werden kann
  - Führt zu Deadlock-Situationen durch wechselseitiges Halten von DB-Locks und der Synchronisationssperren.
- Betrifft die JDBC-ODBC-Bridge von Sun und ältere JDBC-Driver von Oracle



# Datenbankzugriff

## Serialisierung - Lösung

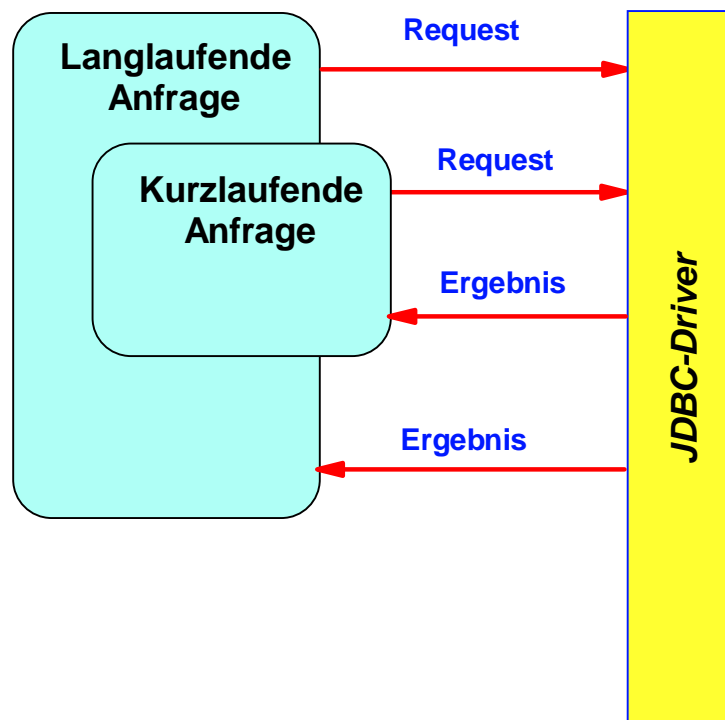


IBL INGENIEURBÜRO  
LETTERS

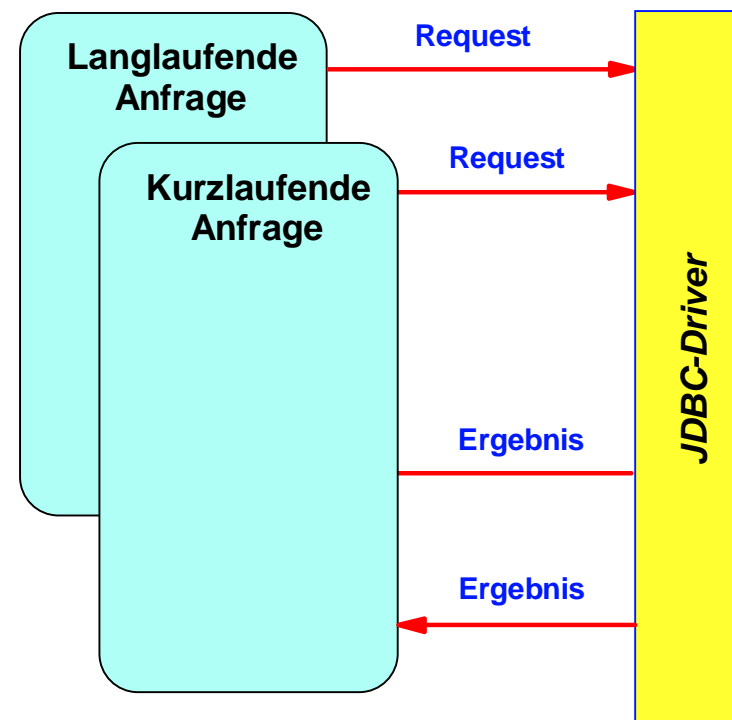
## Lösung

- JDBC-Driver muß echten Parallelbetrieb ermöglichen
- Gilt beispielsweise für die neueren Oracle-JDBC-Driver und für die Intersolv Sequelink Driver
- Überprüfung des korrekten Verhaltens des JDBC-Driver mit Serialisierungstest
  - Ein nach einem langlaufenden Request abgeschickter kurzlaufender Request muß vor dem Ende des Langläufers die angefragten Ergebnisse liefern

### Korrektes Verhalten



### Unbrauchbares Verhalten



# Exceptionhandling

## Behandlung von Exceptions in verteilten Systemen



IBL INGENIEURBÜRO  
LETTERS

### Problem

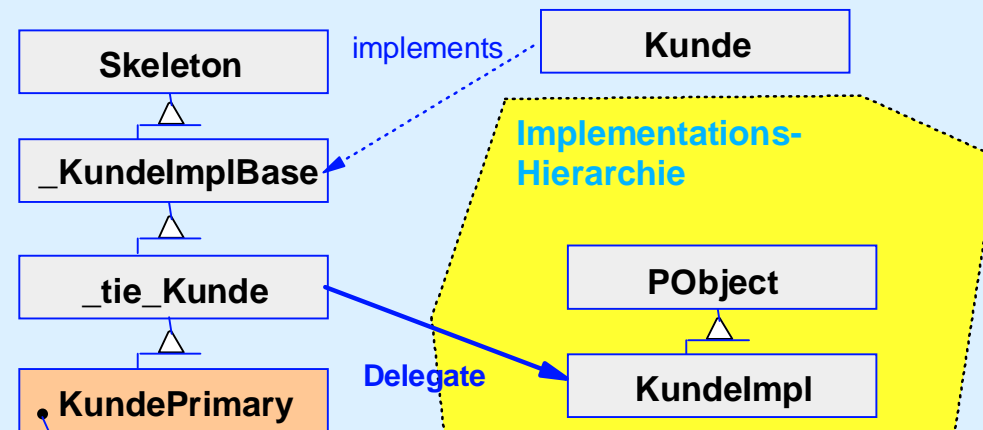
- In verteilten Systemen kann jeder Methodenaufruf potentiell eine Exception verursachen (durch einen Kommunikationsfehler)
- Die Deklaration von Exceptions in jeder Methode ist unerwünscht; sie unterläuft einen flexiblen Verteilungsmechanismus (wahlweise muß jede Methode entspr. Exceptions definieren)
- Jeder Methodenaufruf, der aus dem GUI heraus erfolgt, müßte die geworfenen Exceptions behandeln. Eine zentrale Exceptionbehandlung wird dadurch erschwert

### Lösung

- Realisierung eines Exception-Service
- Die Generierung von Primary-Klassen ermöglicht das Abfangen jeder Exception. Exceptions müssen nicht deklariert und abgefangen werden
- Exceptions müssen nicht idl beschrieben werden; die zentrale Behandlung von Exceptions wird durch den ExceptionHandler ermöglicht (auch die Ausgabe der Fehlermeldung)

#### Objektstruktur auf dem Server

Erweiterung des Tie-Ansatzes für das Exceptionhandling



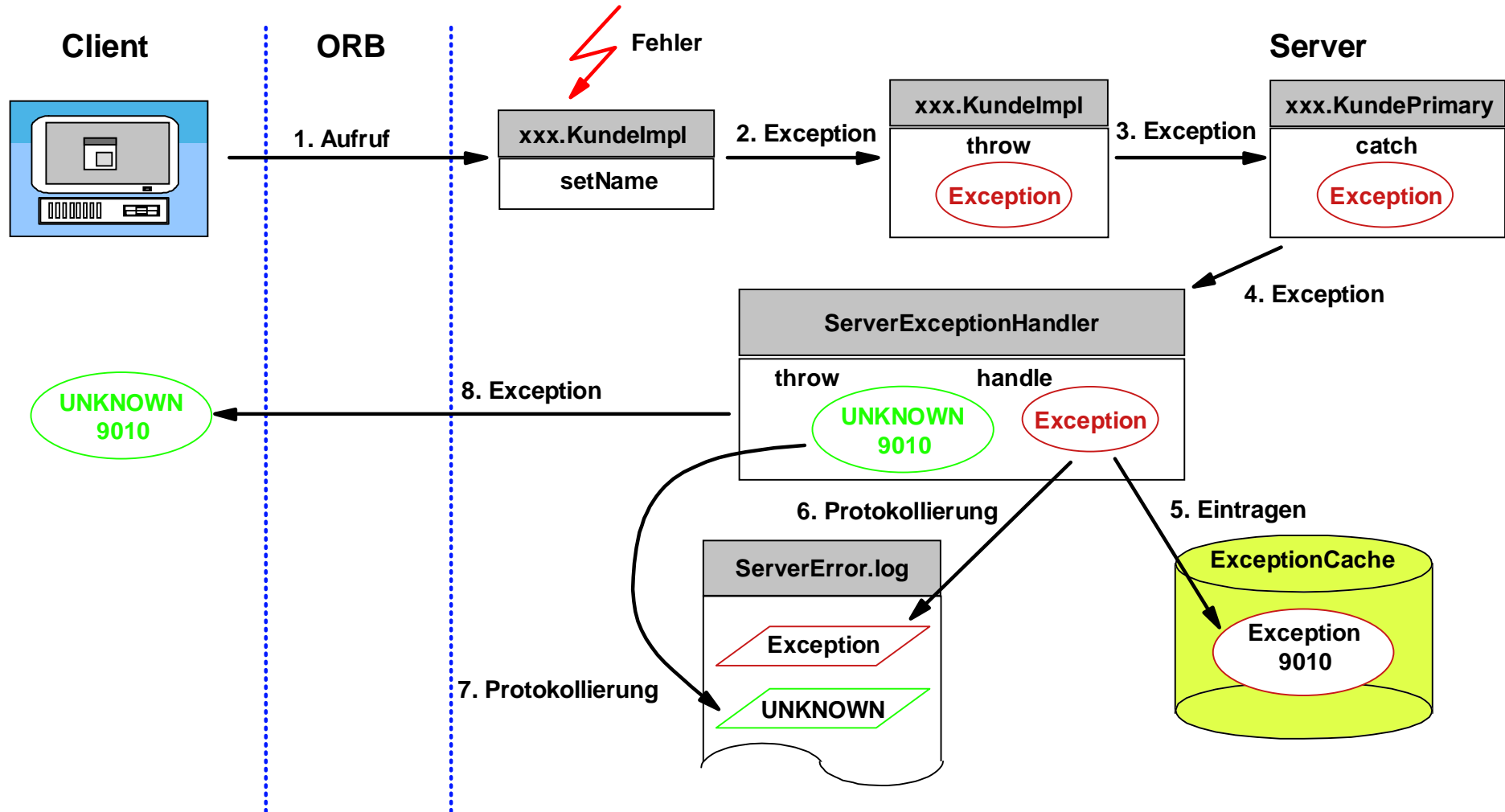
Überschreiben der Methode `_execute(...)` ermöglicht das Abfangen aller in den `*Impl` geworfenen Exceptions bevor das ORB Exception-Handling greift

# Exceptionhandling

Funktionsablauf auf dem Server



IBL INGENIEURBÜRO  
LETTERS

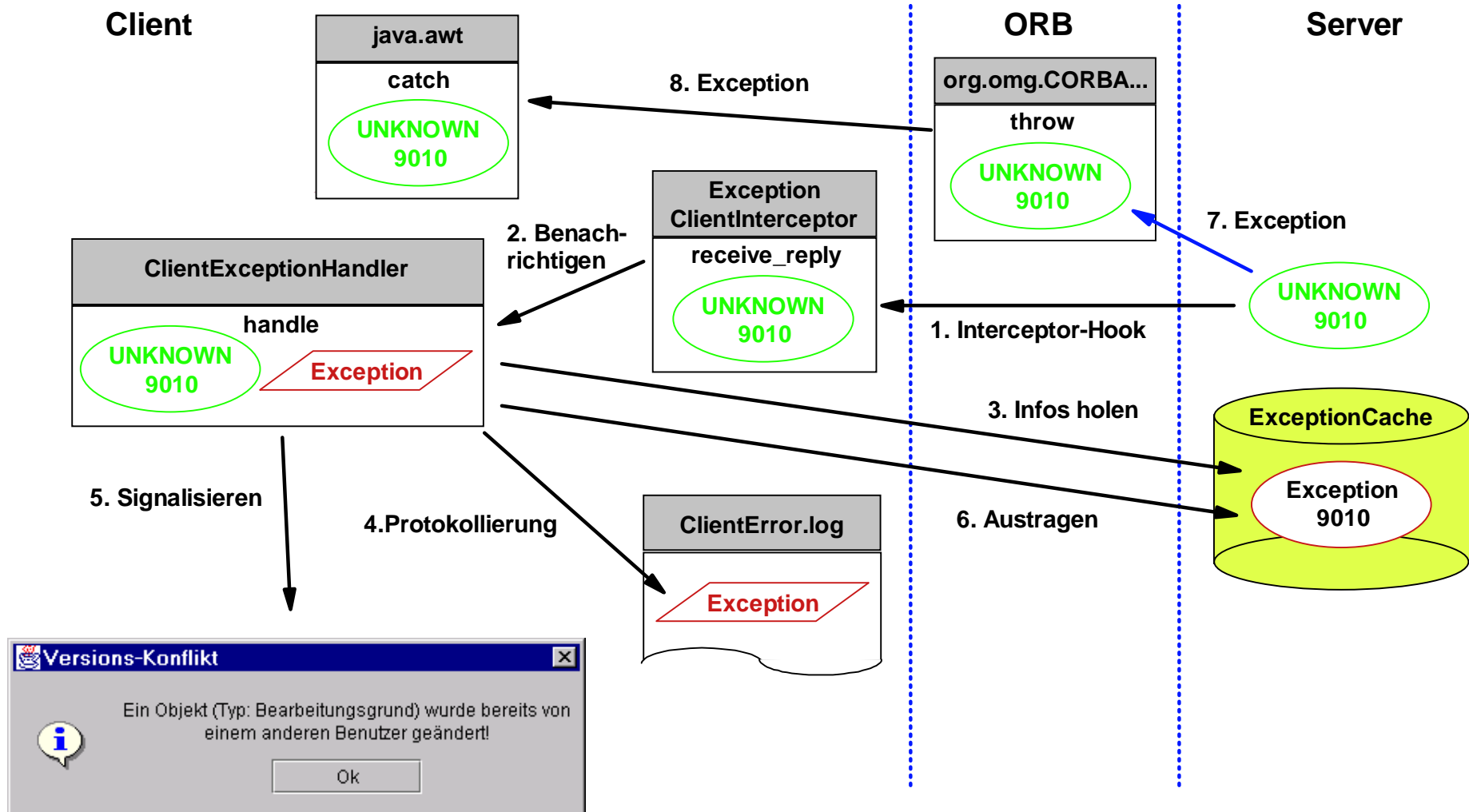


# Exceptionhandling

Funktionsablauf auf dem Client



IBL INGENIEURBÜRO  
LETTERS



# Garbage-Collection

## Freigabe von Objekten in verteilten Systemen



IBL INGENIEURBÜRO  
LETTERS

### Problem

- In verteilten Systemen kann ein Objekt im Objektserver nicht einfach deregistriert werden, da noch Referenzen von Clients darauf bestehen können (die Verwendung eines deregistrierten Objekts führt zu einer Ausnahme)
- Ohne Deregistrierung von Objekten kann aber die Java Garbage-Collection nichts bewirken, da der ORB Referenzen für registrierte Objekte besitzt
- Das dauerhafte Festhalten von Objekten führt aber unweigerlich irgendwann zum Überlauf des Objektserverns!

### Lösung

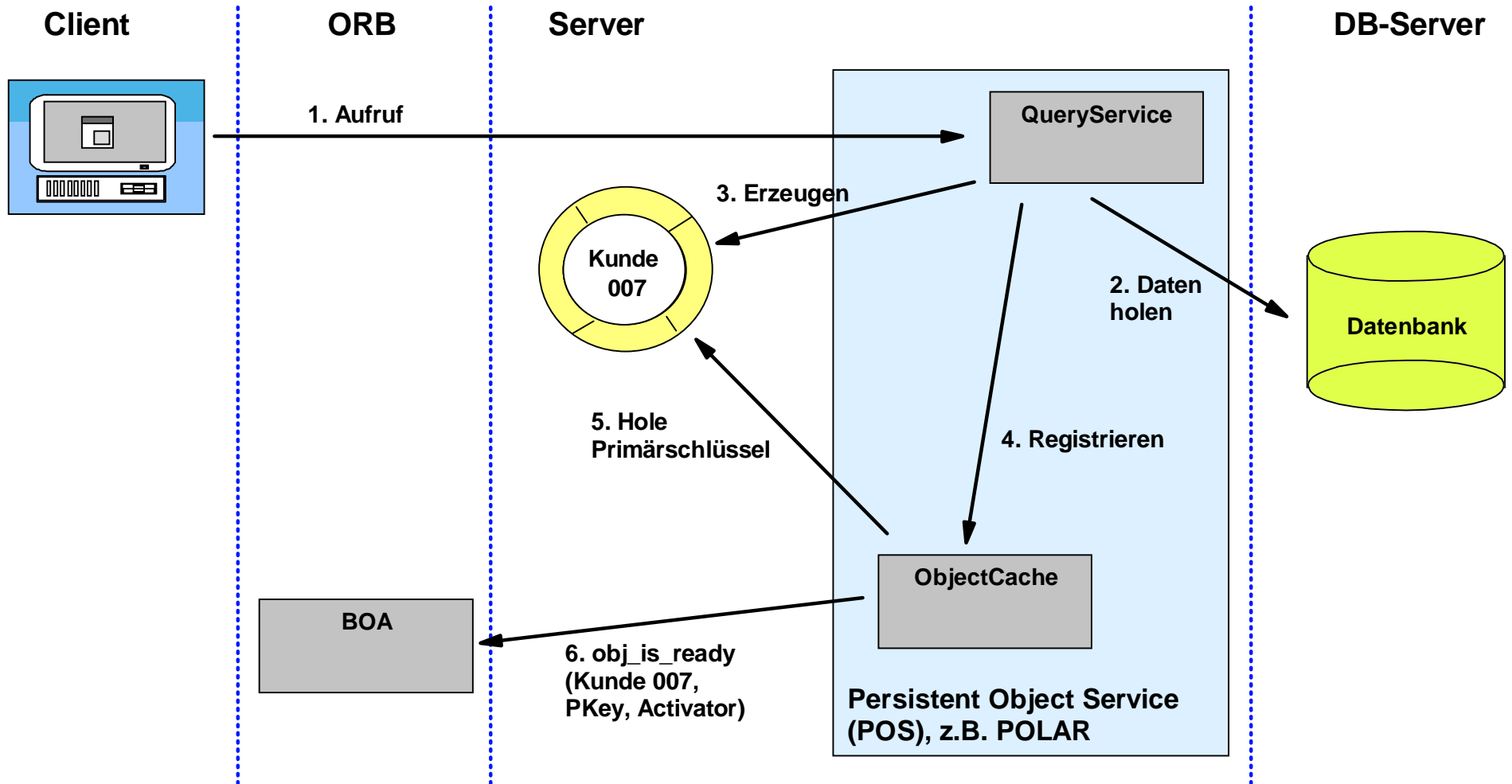
- Nutzung des Activator-Konzepts
  - Objekte werden bei der Instanziierung mit ihrem Primärschlüssel am Activator angemeldet
  - Beim Bereinigen des Objekt-Cache werden die entfernten Objekte beim ORB deregistriert
  - In einer laufenden Transaktion modifizierte Objekte werden von der Bereinigung ausgenommen, da diese Daten besitzen, die nicht aus der Datenbank restaurierbar sind.
  - Beim Zugriff auf ein deaktiviertes Objekt kann der Activator dieses Objekt transparent für den Client erneut instanzieren (z.B. durch Nachladen von der Datenbank)
- Durch die Deaktivierung werden alle Referenzen, die der ORB auf ein Objekt besitzt, freigegeben. Es kann daher von der Garbage-Collection aus dem Speicher entfernt werden, sobald kein Client mehr eine Referenz auf das Objekt hält

# Garbage-Collection

## Objektinstanzierung



IBL INGENIEURBÜRO  
LETTERS

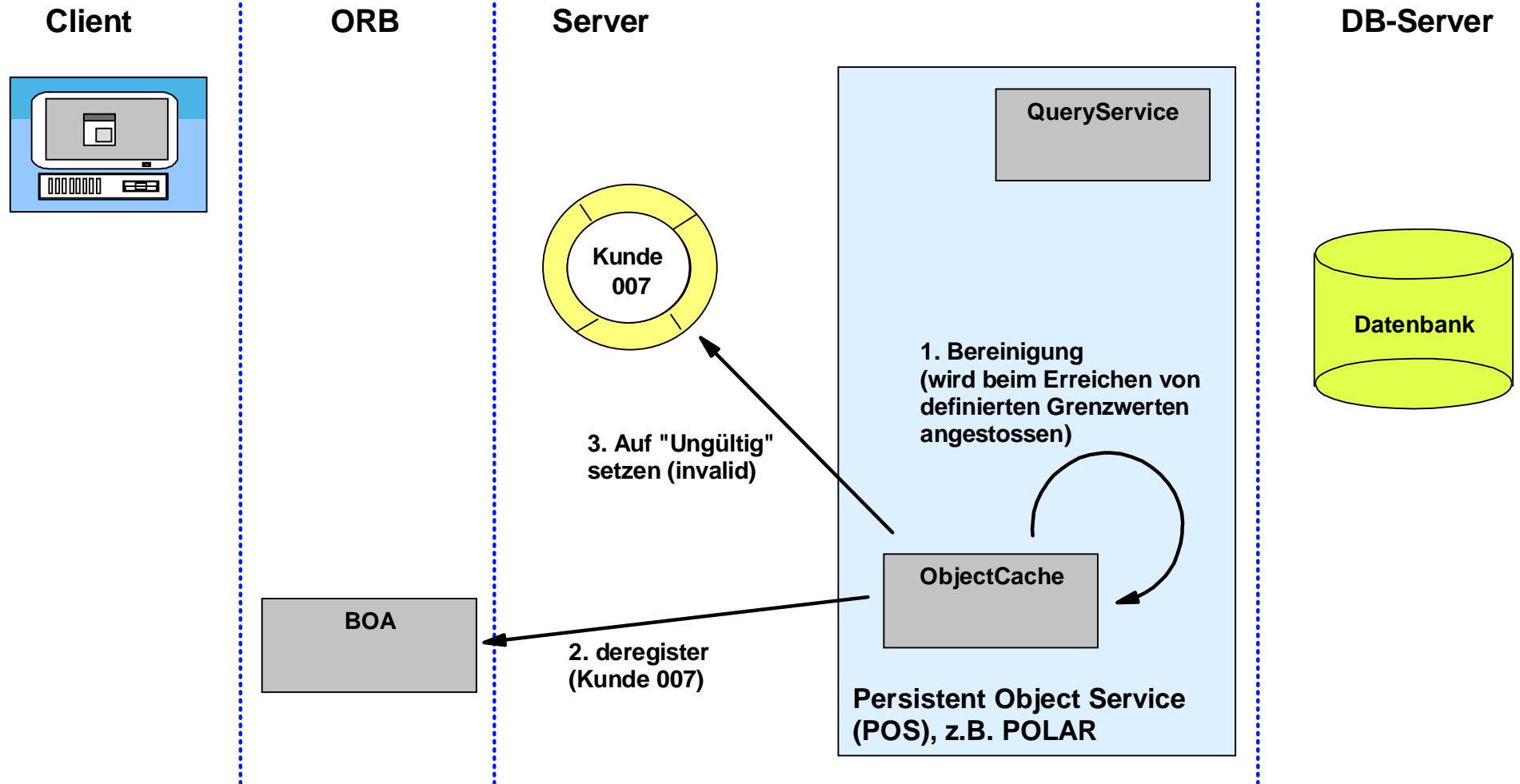


# Garbage-Collection

Objektfreigabe



IBL INGENIEURBÜRO  
LETTERS

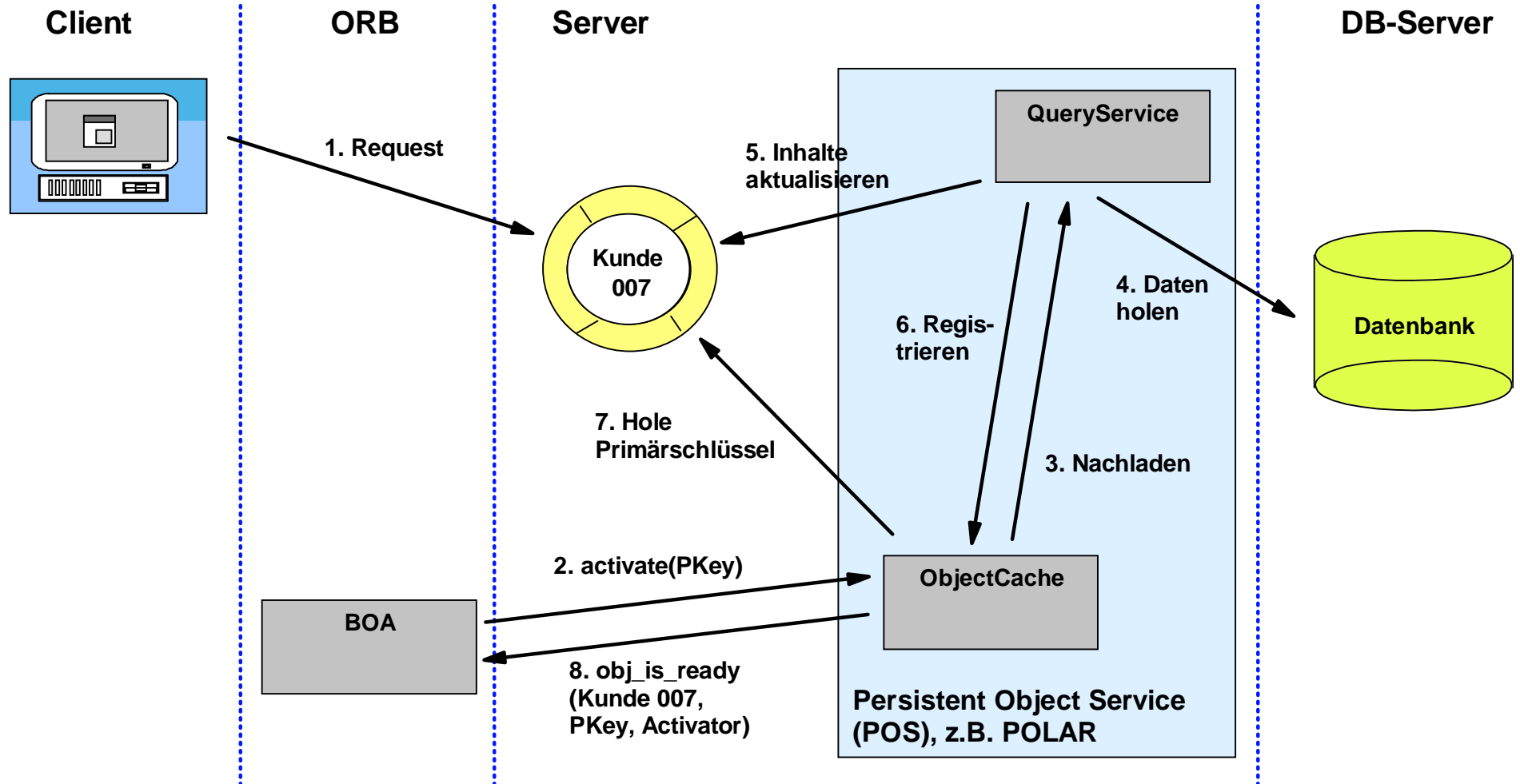


# Garbage-Collection

## Objektreaktivierung



IBL INGENIEURBÜRO  
LETTERS



# Zusammenfassung



IBL INGENIEURBÜRO  
LETTERS

- Systeme mit den genannten modernen Architekturen sind inzwischen in Produktion
- Die Art der Anwendung hat starke Auswirkungen auf die Architektur. Es gibt nicht die Architektur die alle Probleme löst
- Die Realisierung von voll multithreadingfähigen Applikationsservern hat seine Tücken, insbesondere bei der Synchronisation von Client-Requests
- Der Skalierungs-Aspekt muß rechtzeitig berücksichtigt werden. Die Art und Weise wie Skalierung erreicht wird ist stark von der gewählten Architektur abhängig
- Architekturen dieser Art benötigen teilweise völlig neue Lösungsansätze, da der Verteilungsaspekt ein anderes Systemverhalten bewirkt (Beispiel: Exceptionhandling oder Speicherfreigabe)
- Die vorgestellten Architektur-Entwurfsmuster greifen diese Lösungsansätze auf und sollten helfen, eine entsprechende Anwendungsarchitektur zu realisieren