

---

# Kapitel 8

## Verteilte Datenbanken

---

Folien zum Datenbankpraktikum  
Wintersemester 2010/11 LMU München

© 2008 Thomas Bernecker, Tobias Emrich  
unter Verwendung der Folien des Datenbankpraktikums aus dem Wintersemester 2007/08 von Dr. Matthias Schubert

---

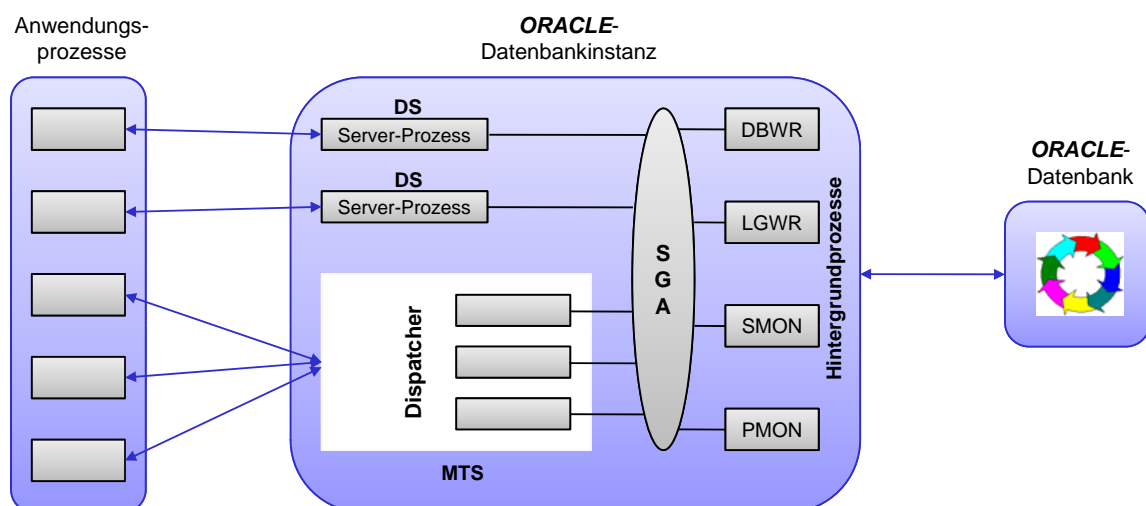
## Übersicht

- 8.1 Client-Server-Architektur
- 8.2 Unterstützte Rechnerarchitekturen
- 8.3 Oracle Parallel Server
- 8.4 Verteilte Datenbanken

## Client-Server-Architektur

- Ein Datenbanksystem beinhaltet in der Regel einen Datenbank-Server, der die ihm angetragenen Datenbankoperationen ausführt. In der *ORACLE-Architektur* bildet die sogenannte **ORACLE-Instanz** den Kern des Datenbanksystems.
- Eine *ORACLE-Instanz* verwaltet genau eine zugehörige Datenbank. Es besteht also eine logische Trennung zwischen der Datenbank und der verwaltenden Instanz. Eine Instanz kann zu verschiedenen Zeitpunkten an verschiedene Datenbanken gekoppelt sein.
- *ORACLE* unterscheidet zwei Konfigurationsvarianten, welche die Kommunikation der Anwendungsprozesse mit der Datenbank-Instanz festlegen:
  - **Dedicated-Server-Configuration (DS)**: An jeden Anwendungsprozess wird ein eigener Server-Prozess gebunden, der die Anforderungen des Anwendungsprozesses bearbeitet.
  - **Dynamic Multi-Threaded-Server-Configuration (MTS)**: Die Anwendungsprozesse kommunizieren mit einem *ORACLE-Dispatcher*, der die Anforderungen in eine Warteschlange in der **System Global Area (SGA)** einreicht. Der Auftrag wird dann durch einen beliebigen vom Dispatcher zugeordneten Server-Prozess bearbeitet.

- Die beiden Konfigurationsvarianten schließen sich dabei nicht gegenseitig aus, sondern können zeitgleich im System laufen.



- Bei der Verbindung mit einer *ORACLE-Instanz* lässt sich die Konfigurationsvariante einstellen, z.B. (bei entsprechenden Einträgen in der Datei '*tnsnames.ora*')

```
CONNECT <User>/<Passwort>@db_mts
```

```
CONNECT <User>/<Passwort>@db_ds
```

# Übersicht

- 8.1 Client-Server-Architektur
- 8.2 Unterstützte Rechnerarchitekturen
- 8.3 Oracle Parallel Server
- 8.4 Verteilte Datenbanken

## Unterstützte Rechnerarchitekturen

- *ORACLE* unterstützt im wesentlichen vier Rechnerarchitekturen:
  - **Ein-Prozessor-System (EP)**: Ein im allgemeinen leistungsstarkes Rechnersystem mit einem Prozessor.
  - **Symmetrisches Multi-Prozessor-System (SMP)**: Das Rechnersystem beinhaltet mehrere gleichberechtigte Prozessoren, die einen gemeinsamen Hauptspeicher teilen (*Shared-Everything-* oder *Shared-Memory-System*).
  - **Cluster-System (CS)**: Das System besteht aus mehreren unabhängigen Rechnern mit jeweils eigenen Hauptspeicher. Auf die Sekundärspeichersysteme (Festplatten) hat allerdings jeder Rechner Zugriff (*Shared-Disk-System*).
  - **Massiv-Parallel-Systeme (MPP)**: Diese Systeme bieten eine hohe Anzahl von Einzelprozessoren (bis zu mehreren 1000), die über einen eigenen Hauptspeicher und einen eigenen Sekundärspeicher verfügen (*Shared-Nothing-System*).
- In der Standardausstattung unterstützt *ORACLE* sowohl *EP-* als auch *SMP-* Architekturen. Um *CS-* und *MPP-* Architekturen verwenden zu können, ist eine spezielle parallele Erweiterung von *ORACLE* erforderlich (*Oracle Parallel Server*, kurz *OPS*).

# Übersicht

- 8.1 Client-Server-Architektur
- 8.2 Unterstützte Rechnerarchitekturen
- 8.3 Oracle Parallel Server**
- 8.4 Verteilte Datenbanken

## Oracle Parallel Server

- Die OPS-Option sorgt dafür, dass *ORACLE* aus der vorhandenen Rechnerarchitektur optimalen Nutzen zieht. Jeder Rechner erhält eine *ORACLE*-Instanz (pro Datenbank auf dem Rechner) mit den jeweiligen Hintergrundprozessen und einer eigenen SGA.
- Wird eine Datenbank durch mehrere (auf verschiedene Rechner verteilte) *ORACLE*-Instanzen verwaltet, dann können in Abhängigkeit von den Anwendungen folgende Vorteile erzielt werden:
  - Ist die Anwendung durch kurze Transaktionen mit wenig CPU- und I/O-Aufwand charakterisiert (z.B. OLTP), dann werden Transaktionen auf die beteiligten Prozessoren verteilt, wodurch eine Erhöhung des Durchsatzes erreicht wird.
  - Zeichnet sich die Applikation durch aufwändige Transaktionen mit hohen CPU- und I/O-Kosten aus (z.B. OLAP), kann die Ausführung einer Transaktion auf den beteiligten Prozessoren parallel verarbeitet werden. Dies führt zu einer Beschleunigung der Anwendung (Speed-Up).
- Die OPS-Option bietet eine erhöhte Fehlertoleranz gegenüber Rechnerausfällen. Fällt ein Rechner mit einer aktiven *ORACLE*-Instanz aus, dann stellt *ORACLE* die Konsistenz der betroffenen Datenbanken durch eine andere Instanz automatisch wieder her.

# Übersicht

- 8.1 Client-Server-Architektur
- 8.2 Unterstützte Rechnerarchitekturen
- 8.3 Oracle Parallel Server
- 8.4 Verteilte Datenbanken

## Verteilte Datenbanken

- Sind in einem Rechnerverbund mehrere Datenbanken (mit ihren *ORACLE*-Instanzen) aktiv, ist der Zusammenschluss zu einer **verteilten Datenbank** möglich. Im Gegensatz zu einem parallelen Server sind die Daten dauerhaft über die beteiligten Instanzen partitioniert und der Ausfall einer Instanz kann nicht durch eine andere Instanz abgefangen werden.
- Eine verteilte Datenbank bietet neben dem entfernten Datenbankzugriff noch weitere Unterstützung:
  - SQL-Anfragen können sich auf Tabellen verschiedener Datenbanken beziehen. Somit sind datenbankübergreifende Joins möglich. *ORACLE* kümmert sich intern um eine geeignete Aufteilung der SQL-Anweisung und das Zusammenführen der Einzelergebnisse zum Gesamtergebnis.
  - Es lassen sich Transaktionen bilden, die Operationen auf mehreren Datenbanken erfordern.
  - *ORACLE* verwaltet bei der Ausführung die beteiligten Rechnerplattformen (Windows NT, Unix) und die darunterliegenden Netzwerk-Protokolle.

## Einbettung von Anwendungen in die verteilte Umgebung

Um Anwendungen in die verteilte Umgebung einzubetten, sind folgende Schritte notwendig:

1. Alle Datenbankinstanzen müssen administrativ dem Datenbanksystem als `<DB-Alias>` bekannt gemacht werden. Dafür steht die Datei '*tnsnames.ora*' zur Verfügung, in welche die zu den `<DB-Alias>` gehörenden Rechneradressen eingetragen werden müssen.

2. Verbinden mit einer Datenbank:

```
CONNECT <User>/<Passwort>@<DB-Alias>
```

3. Um von dem Programm aus auch auf andere Datenbanken zuzugreifen, ist ein sogenannter *Datenbank-Link* erforderlich. Durch diesen wird die Anwendung mit der entsprechenden Datenbank verbunden:

```
CREATE DATABASE LINK <DB-Link> USING <DB-Alias2>;
```

4. Von nun an kann ein SQL-Befehl über `<Name>@<DB-Link>` Bezug auf Schemaobjekte dieser Datenbank (Tabellen, Sichten, Prozeduren, etc.) nehmen.

## Sichten und Synonyme

- Um die Handhabung zu vereinfachen, sind Sichten und Synonyme nützlich.
- Eine Sicht lässt sich wie folgt definieren, so dass anschließend die Tabelle über `<Sicht>` direkt, d.h. ohne Angabe des Datenbank-Links, angesprochen werden kann.

```
CREATE VIEW <Sicht> AS SELECT * FROM <Tabelle>@<DB-Link>;
```

- Ein Synonym kann in ähnlicher Weise eingerichtet werden. Der Synonymname fungiert anschließend wie ein Tabellename.

```
CREATE SYNONYM <Synonym> FOR <Tabelle>@<DB-Link>;
```

## Datenbank-Operationen

Die Datenbank-Operationen werden von *ORACLE* wie folgt auf das verteilte System abgebildet:

- **Einfache verteilte Leseoperationen:** Es werden nur Objekte aus genau einer Remote-Datenbank angefragt (es sind keine lokalen Objekte an der Operation beteiligt). In diesem Fall wird die Anweisung an die entsprechende *ORACLE*-Instanz weitergereicht.

*Beispiel:* `select * from mitarbeiter@db1 where salary>5000;`

- **Komplexe verteilte Leseoperationen:** An der Operation sind Objekte aus mehreren Datenbanken beteiligt. Der lokale Server zerlegt die SQL-Anweisung in Subbefehle und reicht diese an die entsprechenden *ORACLE*-Instanzen weiter.

*Beispiel:* `select * from mitarbeiter@db1 m, abteilung@db2 a  
where ... ;`

**Verteilte Schreiboperationen** sind grundsätzlich durch das Transaktionskonzept abgesichert:

- **Lokale Transaktion:** Es werden nur Objekte in der lokalen Datenbank geändert. Die Verarbeitung läuft wie im nicht-verteilten Fall.
- **Remote Transaktion:** Nur Tabellen, die sich auf genau einem Remote-Server befinden, werden verändert. Diese Operationen werden durch die gewöhnliche Transaktionsverwaltung auf dem Remote-Server verarbeitet.

*Beispiel:* `update mitarbeiter@db1 set gehalt = gehalt * 1.2  
where salary>5000 ;`

- **Verteilte Transaktion:** Es werden Tabellen auf mehreren Datenbank-Servern verändert. Die Bearbeitung solcher Transaktionen stützt sich auf das 2-Phasen-Commit-Protokoll.

## Transaktionskontrolle

ORACLE verwendet hierfür ein **2-Phasen-Commit-Protokoll**:

○ 1. Phase: *Abstimmung der Teilnehmer*

Die erste Phase besteht aus dem Vorbereiten der Transaktion, in der die beteiligten Instanzen die lokale Transaktion durchführen ohne abschließend ein Commit abzusetzen. Jede Instanz sendet ein „Vote Commit“ oder ein „Vote Abort“ an den Koordinator und wartet anschließend auf dessen globale Entscheidung.

○ 2. Phase: *Entscheidung des Koordinators*

Der Koordinator erhält die Ergebnisse aller Teilnehmer und fällt eine globale Entscheidung. Liefern alle Teilnehmer „Vote Commit“, so entscheidet der Koordinator „commit“. Liefert (mindestens) ein Teilnehmer „Vote Abort“, so entscheidet der Koordinator „abort“.

## Replikation

- Im engen Zusammenhang mit verteilten Datenbanken steht das Konzept der *Replikation*. Hier werden Daten aus Gründen der besseren Verfügbarkeit oder des schnelleren lokalen Zugriffs redundant in mehreren Datenbanken gehalten. Das Datenbanksystem muss dann gewährleisten, dass die redundanten Daten miteinander abgeglichen werden.
- ORACLE bietet drei Replikationsmechanismen. Die ersten beiden beruhen auf einem Master/Slave-Prinzip, bei dem die Slave-Datenbank nur lesenden Zugriff erhält.
  - **Asynchrone Replikation**: Die redundanten Daten in der Slave-Datenbank werden periodisch auf den aktuellen Stand der Master-Datenbank gebracht. ORACLE bietet hierfür den Schnappschuss-Mechanismus an.

*Beispiel:*     CREATE SNAPSHOT <Schnappschuss>  
                  REFRESH <Refresh\_Klausel> AS ... ;



- **Synchrone Replikation:** Die redundanten Daten in der Slave-Datenbank werden gleichzeitig mit den Daten in der Master-Datenbank aktualisiert.
  - **Synchrone Lese/Schreib-Replikation:** Änderungen sind sowohl auf der Slave- als auch auf der Master-Datenbank erlaubt. Die Datenbanken werden nach jeder Änderung auf den neuesten Stand gebracht.
- Die Synchronisation der replizierten Datenbankobjekte wird dabei über *interne Trigger* realisiert. Diese werden von *ORACLE* automatisch generiert und sind in eine übergreifende Transaktionskontrolle eingebettet (garantieren also konsistente Replikate). Für das Management von Replikation bietet *ORACLE* als Tool den **Replication Manager** an.