Kapitel 8
Verteilte Datenbanken

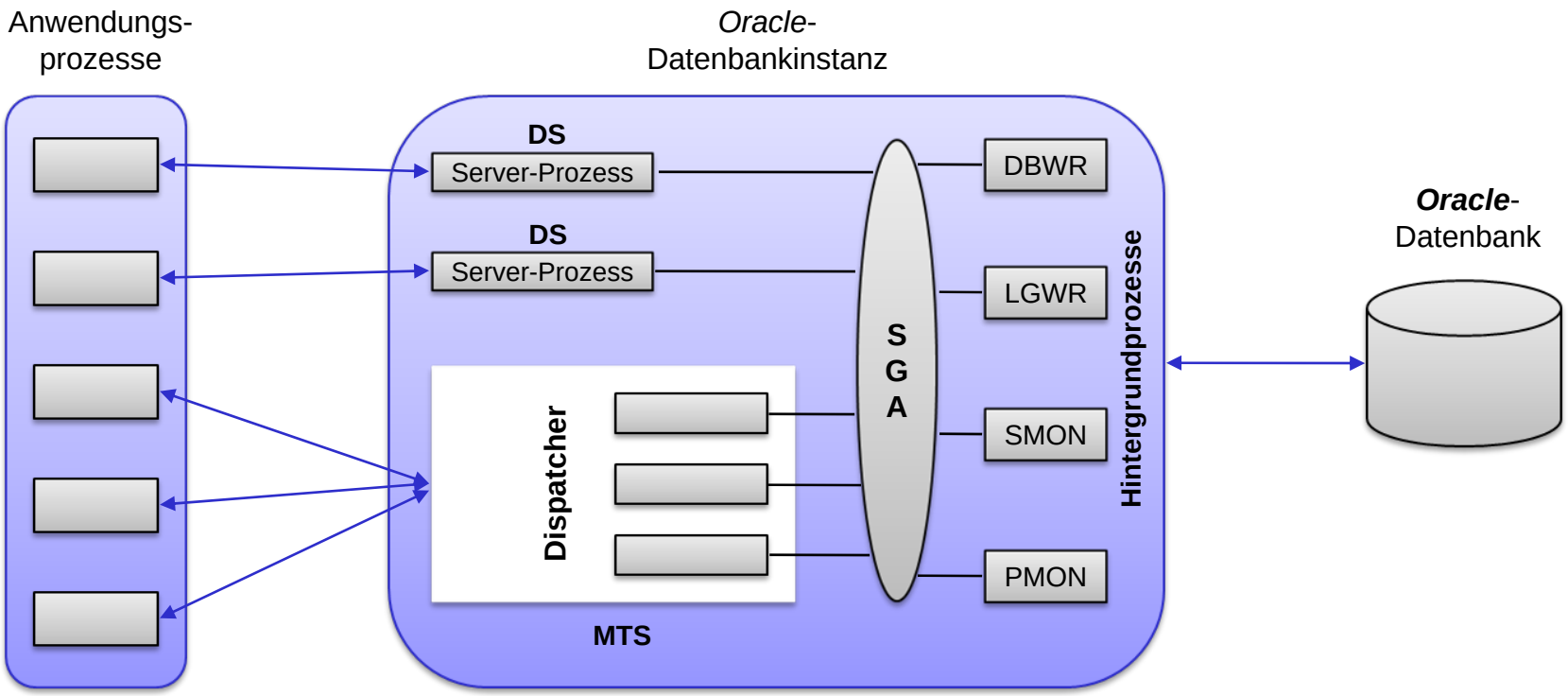
Folien zum Datenbankpraktikum
Wintersemester 2012/13 LMU München

Übersicht

Client-Server-Architektur

- Datenbankserver: führt die ihm angetragenen Datenbankoperationen aus
  Oracle-Architektur: Kern des DBS ist die ***Oracle*-Instanz**
- Eine *Oracle*-Instanz verwaltet genau eine zugehörige Datenbank
  $\Rightarrow$ logische Trennung zwischen der Datenbank und Instanz
- Eine Instanz kann zu verschiedenen Zeitpunkten an verschiedene Datenbanken gekoppelt sein.
- Kommunikation der Anwendungsprozesse mit der Datenbankinstanz in *Oracle*:
  - **Dedicated-Server-Configuration (DS)**: eigener Serverprozess für jeden Anwendungsprozess
  - **Dynamic Multi-Threaded-Server-Configuration (MTS)**: Der *Oracle*-Dispatcher reiht die Anforderungen der Anwendungsprozesse in eine Warteschlange in der **System Global Area (SGA)** ein; Bearbeitung durch einen beliebigen (vom Dispatcher zugeordneten) Server-Prozess

- Kombination der beiden Varianten möglich



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

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

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

Übersicht

Unterstützte Rechnerarchitekturen

- *Oracle* unterstützt im wesentlichen vier Rechnerarchitekturen:
  - ***Ein-Prozessor-System (EP)***
  - ***Symmetrisches Multi-Prozessor-System (SMP)*:** mehrere gleichberechtigte Prozessoren, gemeinsamer Hauptspeicher
    $\Rightarrow$ *Shared-Everything-* oder *Shared-Memory-System*
  - ***Cluster-System (CS)*:** mehrere unabhängige Rechner mit jeweils eigenem Hauptspeicher, gemeinsames Sekundärspeichersystem
    $\Rightarrow$ *Shared-Disk-System*
  - ***Massiv-Parallel-Systeme (MPP)*:** hohe Anzahl von Einzelprozessoren (bis zu mehreren 1000), mit eigenem Haupt- und Sekundärspeicher
    $\Rightarrow$ *Shared-Nothing-System*

- *Oracle*-Standardausstattung: *EP* und *SMP*

- Verwendung von *CS* und *MPP*: parallele Erweiterung von *Oracle* (*Oracle Parallel Server*, *OPS*) erforderlich

Übersicht

Oracle Parallel Server

- OPS-Einsatz: *Oracle* zieht aus der vorhandenen Rechnerarchitektur optimalen Nutzen
- Pro Datenbank auf einem Rechner: *Oracle*-Instanz mit den jeweiligen Hintergrundprozessen eigener SGA
- Durch mehrere (auf verschiedene Rechner verteilte) *Oracle*-Instanzen verwaltete Datenbank:
  - Anwendung mit kurzen Transaktionen mit wenig CPU- und I/O-Aufwand (z.B. OLTP): Verteilung der Transaktionen auf die beteiligten Prozessoren $\Rightarrow$ Erhöhung des Durchsatzes
  - Aufwändige Transaktionen mit hohen CPU- und I/O-Kosten (z.B. OLAP): Parallele Verarbeitung der Transaktionen auf den beteiligten Prozessoren $\Rightarrow$ Beschleunigung der Anwendung
- Erhöhte Fehlertoleranz gegenüber Rechnerausfällen mit aktiver Instanz: Wiederherstellung der Konsistenz der betroffenen Datenbanken durch eine andere Instanz

Übersicht

Verteilte Datenbanken

- Rechnerverbund mit mehreren aktiven Datenbanken (und deren *Oracle*-Instanzen): möglicher Zusammenschluss zu einer *verteilten Datenbank*
- Unterschied zu parallelem Server: Daten sind dauerhaft über die beteiligten Instanzen partitioniert, der Ausfall einer Instanz kann nicht kompensiert werden
- Unterstützung durch verteilte Datenbank:
  - Remote-Datenbankzugriff
  - SQL-Anfragen auf Tabellen verschiedener Datenbanken möglich
    $\Rightarrow$ datenbankübergreifende Joins
    $\Rightarrow$ geeignete Aufteilung der SQL-Anweisung und Zusammenführung der Einzelergebnisse zum Gesamtergebnis durch *Oracle*
  - Transaktionen mit Operationen auf mehreren Datenbanken möglich
  - Verwaltung der Plattformen und Netzwerkprotokolle durch *Oracle*

Einbettung von Anwendungen in die verteilte Umgebung

1. Bekanntmachung der Datenbankinstanzen als `<DB-Alias>` mit den zugehörigen Rechneradressen an das DBS in der Datei '***tnsnames.ora***'
2. Verbinden mit einer Datenbank:

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

3. Zugriff auf andere Datenbanken durch *Datenbank-Links*:

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

4. Bezug eines SQL-Befehls auf Schemaobjekte dieser Datenbank (Tabellen, Sichten, Prozeduren, etc.) über `<Name>@<DB-Link>`

Sichten und Synonyme

- Einfachere Handhabung der Daten
- Viewdefinition ohne Angabe des Datenbank-Links:

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

- Ähnlich: Synonymdefinition

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

  Der Synonymname fungiert anschließend wie ein Tabellenname.

- Unterschied?

Datenbank-Operationen

- ***Einfache verteilte Leseoperation***
  ⇒ Anfrage auf Objekte aus genau einer Remote-Datenbank, keine lokalen Objekte
  ⇒ Weiterleitung der Anfrage an die entsprechende *Oracle*-Instanz:

```
select * from mitarbeiter@db1 where gehalt > 5000;
```

- ***Komplexe verteilte Leseoperation***
  ⇒ Objekte aus mehreren Datenbanken
  ⇒ Zerlegung der SQL-Anweisung durch lokalen Server in Subbefehle, Weiterleitung an die entsprechenden *Oracle*-Instanzen:

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

- ***Verteilte Schreiboperationen***
  ⇒ Absicherung durch das Transaktionskonzept

  - *Lokale Transaktion*:
    ⇒ Änderung von Objekten in der lokalen Datenbank
    ⇒ Verarbeitung läuft wie im nicht-verteilten Fall
  - *Remote Transaktion*:
    ⇒ Änderung von Objekten von genau einem Remote-Server
    ⇒ gewöhnliche Transaktionsverwaltung auf dem Remote-Server:

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

  - *Verteilte Transaktion*:
    ⇒ Änderung von Objekten auf mehreren Datenbank-Servern
    ⇒ Bearbeitung gemäß dem 2-Phasen-Commit-Protokoll

Transaktionskontrolle durch das *2-Phasen-Commit-Protokoll*

- *1. Phase: Abstimmung der Teilnehmer*

  ⇒ Durchführung der lokalen Transaktionen auf den Instanzen ohne Commit
  ⇒ 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*

  ⇒ Liefern alle Teilnehmer „*Vote Commit*", so entscheidet der Koordinator „*commit*"
  ⇒ Liefert (mindestens) ein Teilnehmer „*Vote Abort*", so entscheidet der Koordinator „*abort*"

Replikation

- Redundante Speicherung der Daten in mehreren Datenbanken
  $\Rightarrow$ bessere Verfügbarkeit und schnellerer lokaler Zugriff
- Abgleich der redundanten Daten durch das DBMS nötig
- *Oracle* bietet drei Replikationsmechanismen:
  - ***Asynchrone Replikation**:* Periodische Aktualisierung der redundanten Daten in der Slave-Datenbank bzgl. des Standes der Master-Datenbank:

    ```
    CREATE SNAPSHOT <Schnappschuss>
    REFRESH <Refresh_Klausel> AS ... ; -- wann, was?
    ```

  - ***Synchrone Replikation***: Gleichzeitige Aktualisierung der redundanten Daten in der Slave-Datenbank mit den Daten in der Master-Datenbank
  - ***Synchrone Lese/Schreib-Replikation***: Änderungen auf Master- und Slave-Datenbank erlaubt, Aktualisierung nach jeder Änderung

Replikation

- Synchronisation über *interne Trigger*

  ⇒ automatische Generierung durch *Oracle*

  ⇒ Einbettung in eine übergreifende Transaktionskontrolle

  ⇒ Management der Replikation durch den ***Replication Manager*** von *Oracle*