Ausarbeitung

Replikation in Datenbanken

Yves Adler (05IND)

25. Juni 2008

Inhaltsverzeichnis

1 Einführung

> `Replication is the process of copying and maintaining database objects, such as tables, in multiple databases that make up a distributed database system.` (Ora07a)

Als Replikation (im Zusammenhang mit Datenbanken) wird der Prozess zur mehrfachen Datenspeicherung und Datenpflege in (verteilten) Datenbank-/Informationssystemen bezeichnet. Dieses Vorgehen gewinnt mit dem wachsenden Datenaufkommen und der zunehmenden Verteilung von Datenbanksystemen immer mehr an Bedeutung. Allerdings entsteht auch ein nicht unerheblicher Mehraufwand beim Design, der Implementierung und der Wartung des Datenbanksystems.

Folgende Gründe für den Einsatz von Replikation werden in (Ora07a) genannt:

- **Verfügbarkeit/Recovery:** Der Ausfall einer Kopie der Daten stellt keinen „Totalausfall" des Gesamtsystems dar, sollten diese Daten repliziert auf einem anderen Server zur Verfügung stehen.
- **Performance:** Replikation *kann* schnellen Zugriff auf Daten gewährleisten, so ist es möglich die Last auf mehrere Server zu verteilen und dadurch die Last des Gesamtsystems zu reduzieren. Ein Nutzer greift dann typischerweise auf eine lokale oder Netztopologisch „nahe" Kopie der Daten zu, dadurch werden die lokalen Antwortzeiten auf eine Anfrage reduziert. Dies betrifft jedoch zumeist nur Lesetransaktionen.
- **Disconnected Computing:** Ein Nutzer kann bestimmte Daten lokal replizieren, mit Ihnen *Offline* arbeiten und später wieder mit dem Gesamtsystem synchronisieren.
- **Reduzierung der Netzwerklast:** Sind die Daten über verschiedene Geografische Lokationen (Bsp: Niederlassungen einer Bank) verteilt, so kann Replikation die Netzwerklast im Vergleich zu einem zentralisierten System drastisch reduzieren.

Diese Vor- und Nachteile kann man beim Design eines Datenbanksystems durch die Verwendung verschiedener Replikationsverfahren unterschiedlich gewichten, je nach Anwendungsszenario. Einen „Goldenen Weg" gibt es dabei allerdings nicht, auch ist die Replikation von Daten kein Allheilmittel um Performancesteigerungen zu erzielen, so sind die Lösungen oft sehr Problemspezifisch und eine gute Lösung muss stets in ihrem Umfeld betrachtet werden. Das heißt es ist generell schwierig „gute" Lösungen von einem Umfeld in ein anderes zu übernehmen.

1.1 Anwendunsszenarien

In den letzten Jahren haben sich die Einsatzszenarien (Multimedia, www, etc.) für Informationssysteme stark gewandelt. Damit gehen erhöhte Anforderungen an Skalierbarkeit, Verfügbarkeit und Performance einher, die durch den Einsatz von Replikation adressiert werden sollen.

Ein paar ausgewählte Anwendungsszenarien:

- **Skalierbarkeit**
  Bei Internetdienstleisten wie z.B. Google, YouTube und Amazon steht die Skalierbarkeit der Leselast ganz klar im Vordergrund, so müssen tausende Anfragen von unterschiedlichen Nutzern möglichst effizient, dezentral und schnell zu beantworten.

  Dafür werden große Webserver-Farmen betrieben, welche wiederum auf verteilte Datenbanksysteme zurückgreifen. Nur durch Replikation (eine lokale „Kopie" der Daten) ist es den verschiedenen Niederlassungen möglich optimale Antwortzeiten zu erzielen.

- **Mobile Computing**
  Für Außendienstmitarbeiter ist es häufig nötig auf deren mobilen Endgeräten einen Teil der Firmendaten (z.B.: Die Daten der durch den Außdendienstmitarbeiter betreuten Kunden) zu replizieren. So kann ein Außendienstmitarbeiter „offline" an den Daten arbeiten und diese später mit der Firmendatenbank abgleichen.

- **Höhere Verfügbarkeit**
  Wenn in verteilten Datenbanksystemen zusätzlich zur Verteilung auch noch Replikation eingesetzt wird, kann der Betrieb beim Ausfall eines Knotens aufrechterhalten werden. In sogenannten „Hot-Standby-Konfigurationen" (bzw. „Warm-Standby") sorgt eine Master/Slave-Konfiguration für eine erhöhte Ausfallsicherheit, in dem der Slave alle Daten des Masters repliziert und im Falle eines Ausfalls des Masters sofort einspringt.

- **Datawarehouse**
  Im Datawarehouse/Datamarten-Umfeld kommt es häufig zur „manuellen" Replikation von Daten. So werden die Daten aus Quellsystemen extrahiert, in das Datawarehouse geladen und dort weiter verarbeitet.

  Dies geschieht häufig aus zwei Gründen: Entlastung des Quellsystems (die eigentliche Rechenarbeit läuft dann im DWH) und eine „Zentralisierung" der im DWH-Scope liegenden Daten für bestimmt Auswertungen.

2 Grundlegende Konzepte

Fast jedes Datenbanksystem bietet Hauseigene Werkzeuge um Replikation zu managen. Dabei ist jedoch auffällig, das es keineswegs einen Standard gibt, lediglich ein kleinster gemeinsamen Nenner ist zu finden. Aus diesem Grund werde ich mich nachfolgend auf die Grundlegenden Konzepte beschränken und an geeigneter Stelle auf die Datenbanksystemspezifischen Möglichkeiten eingehen.

Entscheidet man sich für den Einsatz von Replikation in einer Datenbank / einem Datenbanksystem, so muss man diese Entscheidung schon beim Design der Datenbank einfließen lassen. So bieten die meisten Datenbanksysteme die Möglichkeit nahezu alle Datenbankobjekte (vom Package bis zur Tabelle, etc.) zu replizieren. Es liegt beim Nutzer, welche Objekte er für Kritisch/Replikationswürdig hält.

Da Replikation zumeist im Zusammenhang mit verteilten Datenbanksystemen (VDBS) anzutreffen ergeben sich starke Zusammenhänge zwischen der Fragmentierung des VDBS und der darin genutzten Replikation. So kann die horizontale und/oder vertikale Fragmentierung einer Relation grundsätzlich redundanzfrei oder mit Replikation erfolgen, man spricht dann auch von „redundanzfreier Allokation“ bzw. „Allokation mit Replikation“. Auf diese Weise ist es möglich eine Replikation mit sehr feiner Granularität aufzubauen, sollte dies benötigt werden.

Nutzt man Replikation in einem verteilten Datenbanksystem, so muss man eine geringere Transparenz der Daten in Kauf nehmen, da diese explizit an verschiedene Orte verteilt und redundant gespeichert werden.

Allerdings sollte man sich stets bewusst sein, das es sich bei den meisten verteilten Vorgehensweisen um ein Abwägen der Vor- und Nachteile handelt. Es ist sicherlich ein Kinderspiel und binnen kürzester Zeit ein funktionierendes Datenbankumfeld mittels Replikation zu Grunde zu richten.

2.1 symmetrische vs. asymmetrische Replikation

symmetrische Replikation
Bei der Symmetrischen Replikation wird ein Update auf allen Rechnern, bzw. auf allen Replikaten des Replikationsverbundes durchgeführt, man spricht auch von „Update anywhere"/„Update everywhere".

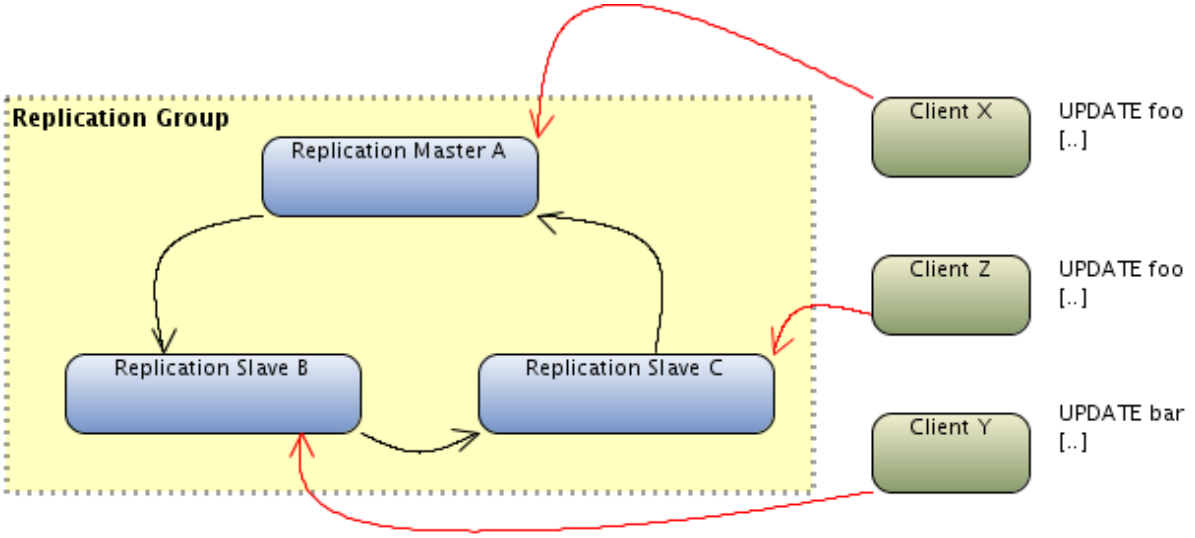

Abbildung 1: Symmetrische Replikation

Wie auf dieser Abbildung zu sehen ist kann es zu Konflikten bei gleichzeitigen Update eines Datenbankobjekts durch zwei oder mehr Clients (hier X und Z) auf unterschiedlichen Knoten des Replikationsverbundes kommen.

Es ist abzuwägen, ob die entstehende Lastverteilung zwischen den Knoten den gestiegenen Kommunikationsaufwand für die Synchronisation der Knoten im Replikationsverbund rechtfertigt.

asymmetrische Replikation
Bei einer Asymmetrischen Replikation wird ein Update der Daten nur auf einem Rechner(verbund)/Replikat, der sogenannten „Primary Copy" bzw. dem „Update Master", zugelassen und durchgeführt. Dieser Rechner(verbund) propagiert nach erfolgtem Update die Änderungen an den Rest des Replikationsverbundes.

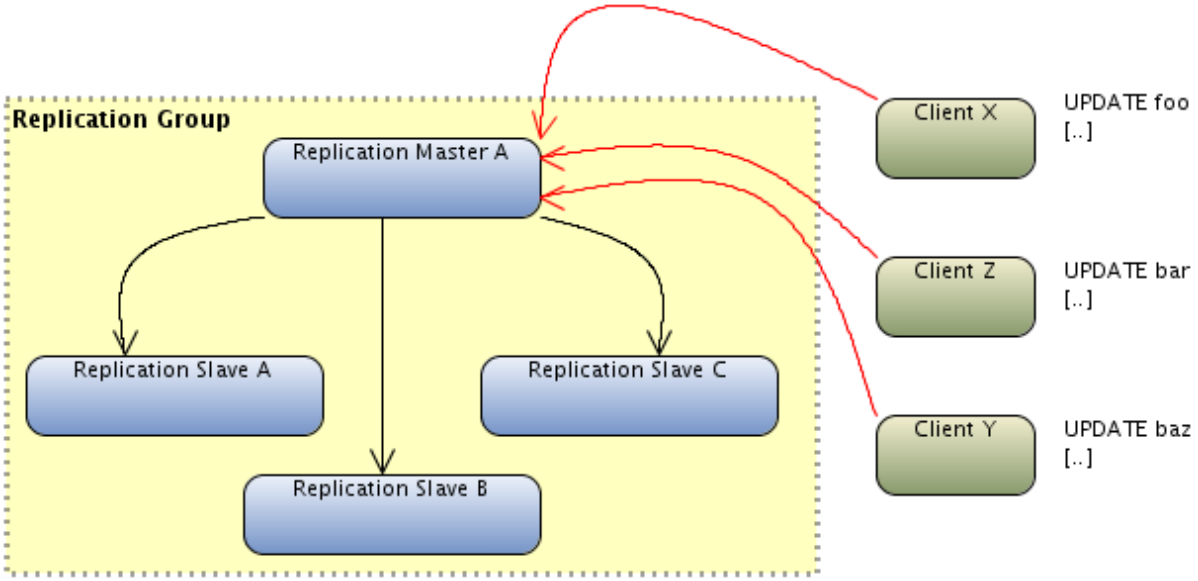

Abbildung 2: Asymmetrische Replikation

Gegenüber der symmetrischen Replikation kann es hier nicht zu Konflikten bei einem Update von einem Datenbankobjekt kommen, da nur ein Knoten zugelassen ist. Dieser muss jedoch alle Änderungen an den Replikationsverband propagieren. Zusätzlich trägt er die komplette Rechenlast aller Updates, welche bei der symmetrischen Replikation auf die verschiedenen Knoten des Verbundes verteilt sind.

2.2 synchrone vs. asynchrone Replikation

synchrone Replikation („eager"/„pessimistic" Replication)
Bei der synchronen Replikation wird eine Transaktion zeitgleich auf allen Replikaten durchgeführt und ist erst dann erfolgreich beendet, wenn die Transaktion auf allen Replikaten erfolgreich durchgeführt wurde. Sollte auf einem Replikat die lokale Transaktion scheitern, so ist auch die globale Transaktion gescheitert.

Mit der synchronen Replikation werden mögliche Konflikte vor dem Commit der Transaktion erkannt. Auf diese Weise kann die Atomarität der Transaktion direkt gesichert werden. Meist wird dies mit Hilfe des „2-Phase-Commit" (2PC) Protokolls, oder einer Variante von diesem (2PC*, 3PC) sichergestellt.

Daraus entsteht auch direkt der Hauptnachteil der synchronen Replikation in Form des Kommunikationsoverheads welcher bei der (Synchronikations-)Komminikation zwischen den Knoten des Replikationsverbundes entsteht (dies äußert sich in längeren Antwortzeiten).

asynchrone Replikation („lazy"/„optimistic" Replication)
Bei der asynchronen Replikation ist es erlaubt, das sich die Replikate unterscheiden, d.h. es wird lediglich sichergestellt das das replizierte Datenbankobjekt konvergiert.

Die asynchrone Replikation kann keine Konfliktfreiheit nach erfolgtem Commit garantieren, es wird lediglich sichergestellt das die Daten konvergieren. Also sind zumindest die mehrzahl der Replikate, welche von einer Transaktion betroffen sind, nach dem Commit auf dem aktuellen Stand. Letztendlich kann die Asynchrone Replikation so einen großen Performancegewinn bei Schreiboperationen erzielen, jedoch auf kosten eines Mehraufwands bei Leseoperationen, da hier mehr Replikate gelesen werden müssen um mögliche Konflikte auszuschließen.

Spezialfall: Prozedurale Replikation (Oracle)
Bei der Prozeduralen Replikation werden die PL/SQL Funktionen und Packages repliziert um dann jeweils auf den lokalen Replikaten ausgeführt zu werden. Das heißt, wenn eine Funktion von einem Client aufgerufen wird, so werden nicht die Ergebnisdaten anschließend synchronisiert (nachdem diese von einem Knoten berechnet wurden) sondern alle Knoten, welche ein Replikat der betroffenen Daten halten berechnen parallel diese Funktion auf dem jeweils lokalen Replikat.

2.3 Konfliktauflösung

Allgemeine Konfliktauflösung

- Timestamp basierte Verfahren:
  - *„Latest Timestamp"*:
    Tritt ein Konflikt zwischen zwei Replikaten bei einem Timestamp basierten Replikationsverfahren auf, so wird das Replikat mit dem „jüngeren" Timestamp gewählt. Diese Strategie ist zwar Problembehaftet, erfüllt aber trotzdem das Konvergenzkriterium.
  - *„Earliest Timestamp"*:
    Stellt das Gegenteil der „Latest Timestamp" Methode dar, wählt also das „älteste" Replikat.
- Master/Slave - *„Mastercopy wins"*:
  Bei Konflikten gewinnt immer das Replikat auf dem Master-Server, dies kommt häufig beim „Disconnected Computing" zum Einsatz.
- *„Site Priority"*:
  Bei Prioritätsbasierten Konfliktauflösungsverfahren werden nicht alle Replikationsstandardorte als gleichberechtigt erachtet. Dies ist meist der Fall, wenn ein Standard den Anspruch auf „präzisereërhebt.
- *„Notice User"*:
  Wird oft als letzte Möglichkeit angesehen, wenn andere Verfahren scheitern wird der Nutzer aufgefordert den entstandenen Konflikt manuell aufzulösen.

Attributbasierte/Spaltenbasierte Konfliktauflösung

- *„Additive and Average"*: Bei zwei konkurrierenden Werten wird einfach der Durchschnitt der beiden gebildet. Je nach dem, ob man „Additive" oder „Average" verwendet wird dabei auf eine andere Formel zurückgegriffen.
- *„Minimum and Maximum"*: Bei zwei konkurrierenden Werten wird einfach das Minimum, bzw. Maximum der beiden gebildet.
- *„Priority Group"*: Kommt bei konkurrierenden Zugriffen auf eine Relation zum Einsatz. Dabei werden die Spalten der Relation mit Prioritäten versehen, die Spalte mit der höchsten Priorität bekommt als erstes den Zugriff.

3 Replikation in der Paxis

In diesem Abschnitt möchte ich einige Replikationsverfahren von Oracle kurz vorstellen. Dabei will ich jedoch lediglich einen groben Überblick geben, detailierte Beschreibungen zu den Möglichkeiten der Datenbanksysteme finden sich in folgenden Dokumenten:

- Oracle 10g: (Ora07a) und (Ora07b)
- PostgreSQL 8.3: (Pos08)
- MySQL 5.1: (MyS08)

3.1 Spezielle Replikationsverfahren

- **Dateisystem Replikation** - Dabei wird die Replikation von dem Datenbanksystem ins Dateisystem verlagert.
- **„Snapshot-Replication": Replikation im Datawarehouse Umfeld** - Stellt eine „One-Way"-Replikation dar, meist liefert ein Quellsystem (regelmässig) Daten an ein Datawarehouse, welche dort geladen und somit repliziert werden. Dies kann einerseits durch einen Export/Import-Mechanismus oder durch einen Database-Link und Materialized Views geschehen.

3.2 Replikation mit Oracle 10g

Oracle 10g kann folgende Datenbankobjekte replizieren:

- Tabellen (oder Partitionen von Tabellen), Views / Object Views
- Indexe und Indextypes
- Trigger
- Packages, Package Bodies sowie Stored Procedures / Functions
- User-Defined Types / Type Bodies und User-Defined Operators
- Synonyms

Grundlage für jede Replikation mit Oracle 10g sind sogenannte „Replication Groups", in diesen werden die zu replizierenden Objekte organisiert/gruppiert.

Dies soll die Administration erleichtern, da sich Objekte welche logisch zusammenhängen leicht gruppieren lassen. Allerdings kann ein zu replizierendes Objekt nur zu genau einer Gruppe zugeordnet werden.

3.2.1 Multimaster Replikation

> `Multimaster replication (also called peer-to-peer or n-way replication) enables multiple sites, acting as equal peers, to manage groups of replicated database objects. Each site in a multimaster replication environment is a master site, and each site communicates with the other master sites.` (Ora07a, Seite 1-4)

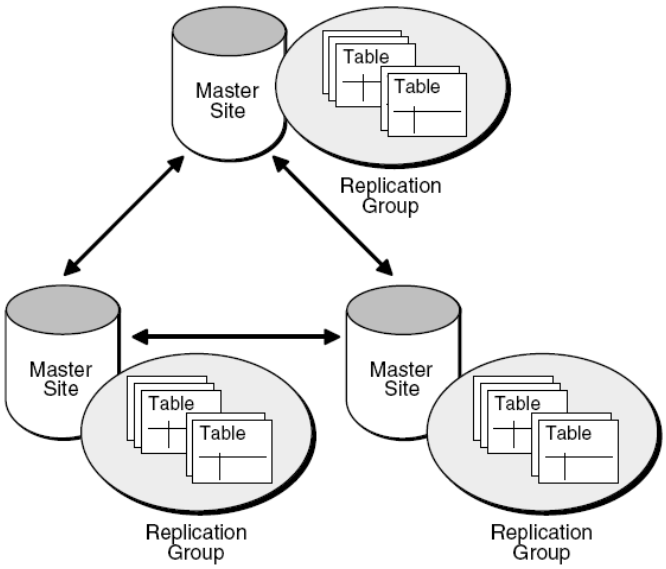

Abbildung 3: Oracle: Multimaster Replikation - (Ora07a, Seite 1-5)

Als „Multimaster"-Replikation wird von Oracle eine symmetrische Replikation, welche Standardmässig asynchron arbeitet bezeichnet. Das heißt alle „Master"-Server in einer „Multimaster"-Replikations-Umgebung kommunizieren miteinander um Konvergenz der replizierten DB-Objekte sicherzustellen. Dabei ist hervorzuheben, das natürlich nur Server, welche dieselbe Replikationsgruppe beherbergen miteinander kommunizieren müssen.

Allerdings biete Oracle die Möglichkeit eine synchrone „Multimaster"-Replikation zu nutzen - siehe (Ora07a, Seite 2-29 und 2-30). Ebenfalls ist es möglich optional in ei-

ner „Multimaster"-Replikations-Umgebung die Prozedurale Replication von Oracle zu nutzen.

3.2.2 Materialized View Replikation

> `A materialized view contains a complete or partial copy of a target master from a single point in time.`
> (Ora07a, Seite 1-5)

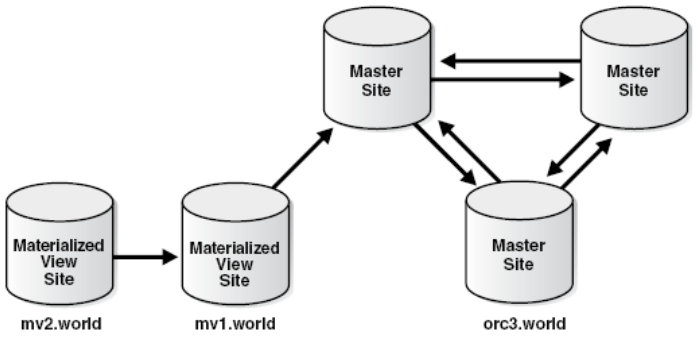

Abbildung 4: Oracle: „Multitier Materialized Views" Replikation - (Ora07a, Seite 3-23)

Die (Read-Only) materialized View Replikation ist technisch sicherlich die einfachste in Oracle 10g zur Verfügung stehende Replikation. Sie wird auch als „Single-Master"-Replikation bezeichnet und kann sowohl synchron, als auch asynchron ablaufen.

Die materialized View Replikation...

- ...ermöglicht schnellen lokalen Zugriff
- ...reduziert die Last auf dem Master-Server
- ...kann die Sicherheit erhöhen, da nur bestimmte Daten (ein beliebiges Subset) vom Master repliziert werden können.
- ...bietet die Möglichkeit zum erstellen von „Disconnected Materialized View Enviroments"

Eine interessante Variante stellt die „updateable" materialized Views, welche dem Nutzer lokal Änderungen an der materialized View erlauben. Diese lokalen Änderungen werden dann an den Master übermittelt und von dort weiter zu allen Knoten der Replication Group.

Die materialized-View-Replikation unterstützt folgende Refresh-Arten:

- **Fast refresh** - Nutzt das materialized View Log um nur die Spalten, welche sich seit dem letzten Update verändert haben zu aktualisieren.
- **Complete refresh** - Aktualisiert die komplette materialized View
- **Force refresh** - führt (wenn möglich) ein „Fast refresh" aus, sollte dies nicht möglich sein wird eine komplette Aktualisierung („Complete Refresh") durchgeführt.

3.2.3 „Multimaster und Materialized View Hybrid" - Replikation

In Oracle 10g gibt es die Möglichkeit Multimaster und Materialized View Replikation zu kombinieren. Dies ist sicherlich ein in der Praxis häufig anzutreffendes Vorgehen, da es die Vorteile beider Replikationsstrategien kombiniert ohne nennenswerte Nachteile zu generieren.

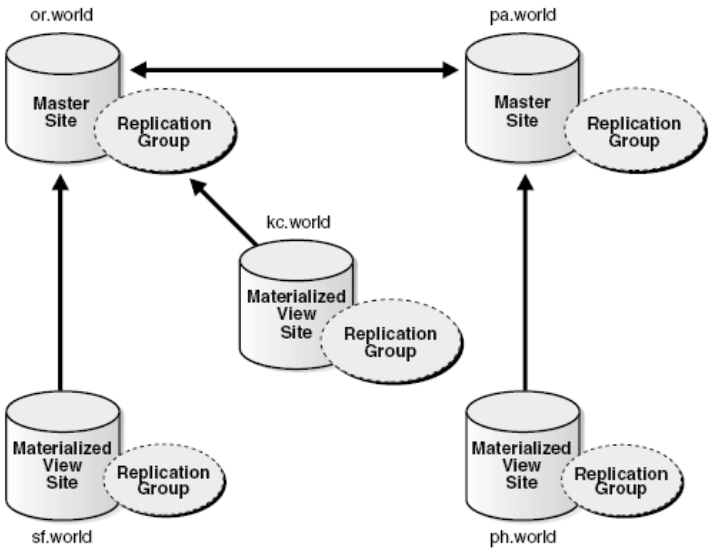

Abbildung 5: Oracle: „Hybrid Configuration" Replikation - (Ora07a, Seite 1-10)

Literatur

(KE04) Kemper, Alfons und André Eickler: *Datenbanksysteme - Eine Einführung*, Kapitel Verteilte Datenbanken, Seiten 443–478. Oldenbourg, München, 5., aktualisierte und erweiterte Auflage, 2004.

(Kud07) Kudrass, Thomas (Herausgeber): *Taschenbuch Datenbanken*, Kapitel Verteilte und förderierte Datenbanksysteme, Seiten 394–426. Hanser, München, 2007.

(MyS08) MySQL AB: *MySQL 5.1 Referenzhandbuch*, 2008.

(Ora07a) Oracle: *Oracle® Database Advanced Replication 10g Release 2 (10.2) - Part Number B14226-02*, 2007.

(Ora07b) Oracle: *Oracle® Database Advanced Replication Management API Reference 10g Release 2 (10.2) - Part Number B14227-02*, 2007.

(Pet06) Petković, Dušan: *SQL Server 2005 - Eine umfassende Einführung*, Kapitel Datenreplikation, Seiten 485–499. dpunkt.verlag, Heidelberg, 1. Auflage, 2006.

(Pos08) The PostgreSQL Global Development Group: *PostgreSQL 8.3.1 Documentation*, 2008.

(SKS02) Silberschatz, Abraham, Henry F. Korth und S. Sudarshan: *Database System Concepts*. McGraw Hill, New York, 4. Auflage, 2002.