Bei der Entscheidung für eine ETL-Technik mit hohen Aktualitätsansprüchen (Near-Real-Time) sind Effizienz und Stabilität zwei wesentliche Kriterien. Dies gilt natürlich für die Transformation der Daten innerhalb des Data Warehouse, ganz besonders aber auch für die Extraktion der Daten aus den Quellsystemen und für die Befüllung – den Load der Staging Area. Der Artikel stellt eine Lösung vor, die auf Oracle Data Guard und Transportable Tablespaces basiert.

Effizientes und einfaches Staging in „Near Real Time"-Data-Warehouses

Mathias Zarick und Karol Hajdu, Trivadis AG

Die Märkte in zahlreichen Branchen werden immer unberechenbarer und kurzlebiger. Viele Fachanwender sind inzwischen nicht mehr bereit, Tage oder mehrere Stunden auf die neuesten Werte zu warten. Sie benötigen kürzere Latenzzeiten. Das „Near Realtime"-Data-Warehouse hat das Licht der Welt erblickt. Gleich, für welchen Ansatz man sich entscheidet, ein wesentlicher Aspekt bleibt immer die Verfügbarkeit einer Staging Area mit extrem kurzer Latenzzeit.

Unterschiedliche Lösungen mit verschiedenen Vorteilen

Es gibt zahlreiche technische Ansätze zur Datenextraktion und Befüllung von Staging Areas (siehe Tabelle 1). Die Ansätze unterscheiden sich dabei in den folgenden grundlegenden Kriterien:

- Menge der transferierten Daten, die für eine Aktualisierung benötigt werden
- Vollständigkeit der transferierten Daten
- Einfluss auf die Performance der Quellsysteme
- Einfluss auf die Verfügbarkeit und Stabilität der Quellsysteme
- Kosten
  - Hardwarekosten und Software-Lizenzen sowie Implementierungsaufwände
  - Komplexität im Betrieb (Aufwände, Stabilität)

Falls das Quellsystem eine Oracle-Datenbank nutzt, gibt es eine effektive Lösung für die Extraktion der Daten und die Befüllung der Staging Area.

Der hier vorgestellte Ansatz basiert auf Oracle Data Guard, Flashback und Transportable Tablespaces. Die Lösung bietet größtenteils dieselben Vorteile wie andere Replikationstechniken, zum Beispiel Oracle Streams (Asynchronous Change Data Capture) oder Oracle GoldenGate:

- Nur relativ kleine Datenmengen müssen transferiert werden
- Es gibt nur einen geringen Einfluss auf die Performance und Verfügbarkeit des Quellsystems

Konzept / Implementierung	Transferiertes Datenvolumen	Vollständigkeit	Einfluss auf Performance des Quellsystems	Einfluss auf Verfügbarkeit / Stabilität des Quellsystems	Komplexität des Betriebs
Daten-Vollextraktion	☹	☺	☹	😐	☺
Marker-basierte Extraktion	☺	😐	😐	😐	☺
Journal-basierte Extraktion	☺	☺	☹	😐	😐
Oracle Streams	☺	☺	☺/😐	☺	😐/☹
Oracle Golden Gate	☺	☺	☺/😐	☺	😐/☹

Tabelle 1: Überblick über die Aktualisierungsmöglichkeiten für Data-Warehouse-Staging-Areas

Konzept / Implementierung	Zu transferierende Datenmengen	Komplexität	Performance-einbußen im Quellsystem	Verfügbarkeit des Quellsystems	Betriebs-komplexität
Neue Lösung mit Data Guard und Transportable Tablespaces	☺	☺	☺	☺	☺

Tabelle 2: Neue Lösung ermöglicht eine deutlich geringere Betriebskomplexität als Streams oder GoldenGate

- Es werden alle Arten von Änderungen bemerkt und transferiert

Die hier vorgestellte Lösung hat allerdings eine signifikant geringere Betriebskomplexität als Oracle Streams oder Oracle GoldenGate (siehe Tabelle 2).

Eine Lösung mit Data Guard

Guard verwaltet Standby-Datenbanken als Kopien von Primär-Datenbanken. Eine Standby-Datenbank kann dabei auch direkt für die Aktualisierung einer Staging Area in einem Data Warehouse genutzt werden. Die Idee basiert auf Data Guards Fähigkeit, eine physische Standby-Datenbank zeitweilig für Schreib- und Leseoperationen zu öffnen und sie anschließend wieder in den Ursprungszustand zurückzustellen, den sie beim Öffnen hatte. Dies wird durch die Flashback-Database-Funktion und die sogenannten „garantierten Restore-Points" ermöglicht. Abbildung 1 zeigt eine auf dem Data Warehouse Host (DWH) eingerichtete physische Standby-Datenbank für die OLTP-DB. Die primäre Datenbank liegt dabei auf dem Host-OLTP.

Aus der Staging Area lesen

Benötigt nun ein ETL-Programm im Data Warehouse die Daten aus der Staging Area, wird folgender Prozess angestoßen:

- Der Recovery-Prozess auf der Standby-Datenbank wird gestoppt
- Die physische Standby-Datenbank wird zu einer sogenannten „Snapshot"-Datenbank konvertiert und damit im Read-Write-Modus geöffnet
- Tablespaces der Standby-Datenbank (hier der Tablespace CRM) werden mit der „Transportable Tablespace"-Funktion in die DWH-Datenbank eingehängt
  - Der CRM-Tablespace in der Snapshot-Datenbank wird dabei auf „Read-only" gesetzt
  - Die Metadaten (Tabellendefinitionen, Indexdefinitionen etc.) des Tablespace werden in die DWH-Datenbank geladen

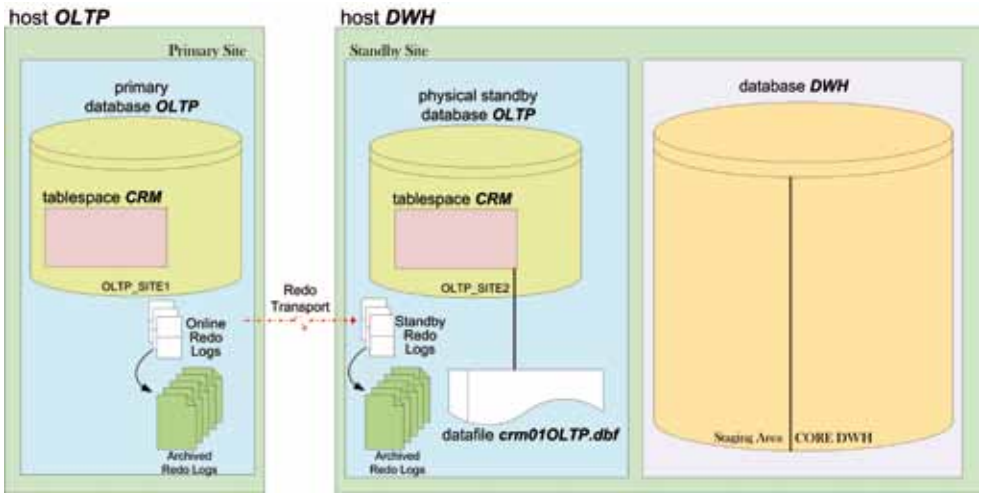

Abbildung 1: Auf dem DWH-Host wird eine physische Standby-Datenbank für OLTP mit Data Guard erstellt

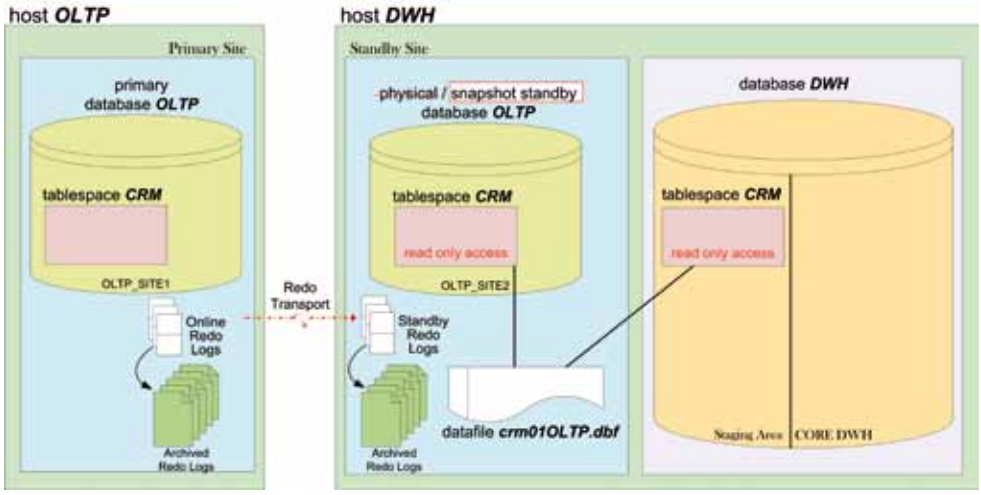

Abbildung 2: Auf dem DWH-Host ist die Datendatei crm01OLTP.dbf nun Teil beider Datenbanken (Read-only)

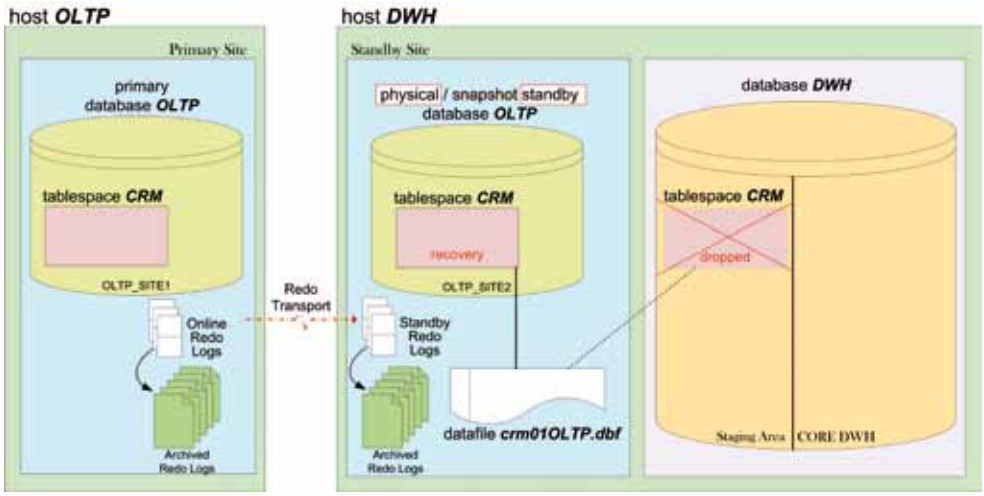

Abbildung 3: Tablespace CRM wird aus der DWH-Datenbank gelöscht und die Snapshot-Standby-Datenbank in eine physische Standby-Datenbank zurückkonvertiert

- Die Datendatei crm01OLTP.dbf ist jetzt Teil beider Datenbanken (Snapshot-Standby-OLTP und -DWH, siehe Abbildung 2)
- Das ETL-Programm kann nun die Daten aus der Staging Area lesen

Aktualisieren der Staging Area

Sobald nun neue Daten in den CRM-Teil der Staging Area geladen werden sollen, sind diese zu aktualisieren (siehe Abbildung 3):

- Der eingebundene Tablespace CRM wird aus der DWH-Datenbank gelöscht (dropped)
- Die Snapshot-Standby-Datenbank wird zurück in eine physische Standby-Datenbank konvertiert
- Gleichzeitig startet der Recovery-Prozess für alle Datendateien der OLTP-Standby-Datenbank

Diese Lösung hat folgende Vorteile:

- Die Staging Area beinhaltet alle Daten, nicht nur die Inkremente
- Es entsteht keine zusätzliche Last auf dem OLTP-Host
- Es werden keine Datendateien kopiert
- Die zwischen OLTP und DWH transferierte Datenmenge ist ausschließlich durch die Datenänderungen im OLTP-System bestimmt
- Redo-Log-Informationen werden fortlaufend übertragen, dabei ist es unerheblich, ob die Standby-Datenbank im Snapshot-Standby- oder im Physical-Standby-Modus ist
- Die Laufzeit des Aktualisierungsprozesses der Staging Area entspricht dem Refresh der Standby-Datenbank und ist nicht von der Größe des Tablespace abhängig, sondern nur von der Menge der Datenänderungen seit der letzten Aktualisierung
- Einmal aufgesetzt, ist der Betrieb dieser Lösung sehr einfach
- Es werden weder „Remote-Queries“ noch „Distributed Joins“ benötigt
- Unter Berücksichtigung des Einflusses auf das Quellsystem und der Last auf dem DWH-Host ist die hier vorgestellte Lösung die effizienteste
- Nur Redo-Logs werden benötigt, keinerlei weitere Strukturen
- Die Lösung arbeitet nur auf Basis von Datenblockänderungen, nicht auf SQL-Level

Dazu müssen einige technische Voraussetzungen erfüllt sein. Diese lassen sich in verschiedene Kategorien unterteilen:

- *Identische Zeichensätze in allen Datenbanken*
  Um Transportable Tablespaces nutzen zu können, müssen die Zeichensätze der OLTP- und der DWH-Datenbank identisch sein.
- *Self-Contained Tablespace Sets*
  Um eine Menge, also ein Set von Tablespaces, von einer zu einer anderen Datenbank zu transportieren, muss diese „Self-Contained“ sein.
  Das bedeutet, dass man kein Set von Tablespaces transportieren kann, das Objekte mit abhängigen Objekten wie zum Beispiel Materialized Views oder Tabellenpartitionen auf anderen Tablespaces enthält, solange diese Objekte nicht alle auf einmal in einem Set transportiert werden.
- *Benötige Oracle-Datenbank-Releases*
  Die OLTP-Datenbank muss unter Oracle 10*g* oder höher laufen. Dabei ist Oracle 0 empfehlenswert, da das Snapshot-Standby-Feature damit verfügbar ist. Mit Oracle 10*g* wird diese Funktionalität manuell nachgebildet, indem man einen garantierten Restore Point in der Standby-Datenbank erzeugt, bevor man sie im Read-Write-Mode öffnet. Das bedeutet folgende Einschränkungen mit 10*g*:
  - Es muss ein wenig programmiert werden, was allerdings keine große Sache und relativ simpel ist.
  - Der Redo-Transport zwischen der Primary- und der Standby-Datenbank wird während der Phase, in der letztere Read-Write geöffnet ist, angehalten. Mit 11*g* bleibt der Log-Transport dagegen die ganze Zeit über aktiv.

  Um die Transportable Tablespaces nutzen zu können, muss die Data-Warehouse-Datenbank dasselbe oder ein höheres Release als die Standby-Datenbank haben.
- *Benötigte Oracle-Lizenzen*
  Die Lösung benötigt eine Oracle-Datenbank-Enterprise-Edition-Lizenz für den OLTP-Host und den DWH-Host. Alle weiteren benötigten Features wie Data Guard, Transportable Tablespaces und die Snapshot-Datenbank (letztere ab 11*g*) sind darin bereits enthalten. Darüber hinaus ist keine weitere Option notwendig – weder Active Data Guard (eine kostenpflichtige Option von 11*g* mit „Real Time Query” sowie inkrementellem Backup auf der Standby-Datenbank) noch Partitioning.

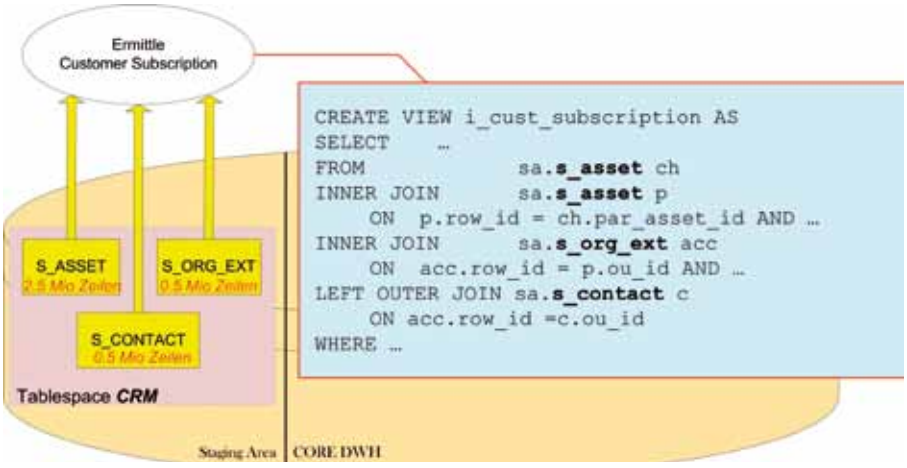

Abbildung 4: Der Transformationsprozess liest Siebel-Tabellen und transformiert die Daten in eine neue Struktur namens „Customer Subscription”

Ein Beispiel aus der realen Welt

Die vorgestellte Lösung arbeitet mit praktisch jeder Oracle-basierten Anwendung – sei es eine Standardanwendung oder eine Eigenentwicklung. Um den Ansatz an einem realistischen Beispiel zu demonstrieren, wird ein Auszug aus dem Datenbank-Schema einer CRM-Applikation namens „Siebel“ verwendet, einer weitverbreiteten CRM-Lösung von Oracle. Im Beispiel werden als repräsentative Auswahl der Siebel-Tabellen mit hoher Komplexität und Kardinalität die Tabellen „S_CONTACT“, „S_ORG_EXT“ und „S_ASSET“ verwendet.

Folgende typische Data-Warehouse-Situation wird angenommen: ETL-Prozesse sollen den Inhalt aus den Siebel-Tabellen extrahieren, in eine neue Zielstruktur namens „Customer Subscription“ transformieren und dafür eine entsprechende View verwenden (siehe Abbildung 4).

Das Data Warehouse muss dabei nicht nur jeweils den letzten gültigen Datensatz in „Customer Subscription“ speichern, sondern alle jemals bekannten historischen Zustände. Der ETL-Prozess muss also immer den neuesten „Abzug“ der drei Tabellen mit dem Inhalt im Core-DWH vergleichen, dann – sofern Änderungen, Löschungen oder neue Datensätze gefunden werden – neue Versionsdatensätze im Zielmodell erzeugen und schließlich die alten Datensätze als „ab jetzt ungültig“ markieren. Dieses Verfahren nennt man „Historisierung“ (siehe Abbildung 5).

Angenommen, der Data-Warehouse-Architekt trifft folgende Design-Entscheidungen:

- Wegen zahlreicher Abhängigkeiten innerhalb der Quellsystem-Tabellen und -Prozesse soll die Staging Area immer den kompletten Satz an Daten enthalten und nicht nur die Inkremente
- Der Transfer von vielen Millionen Records pro Nacht ist nicht akzeptabel
- Im Quellsystem gibt es keine Datensatz-Marker oder Journal-Tabellen, die eine zuverlässige Identifikation von Inkrementen ermöglichen
- Der Architekt entscheidet sich für den Einsatz der hier vorgestellten Lösung

Wegen der hohen Kardinalität der Daten (viele Millionen Datensätze) wird zudem eine gute Skalierbarkeit des zugrundeliegenden DWH-Datenmodells angenommen. Die wichtigsten Schritte zum Aufbau und Betrieb der Lösung sehen folgendermaßen aus: Oracle Data Guard wird entsprechend den bereits erwähnten Vorgaben aufgesetzt. Es wird eine Data-Guard-Broker-Konfiguration erzeugt, wobei der Protection-Mode auf „Maximum Performance” (Default) belassen und der Log-Transport auf „asynchron“ gesetzt wird. In den beiden Datenbanken DWH und OLTP wird die Rolle „dwh_sa_crm_role“ erzeugt:

```
CREATE ROLE dwh_sa_crm_role;
```

Als nächstes wird dieser Rolle in der OLTP-Datenbank das SELECT-Privileg für alle relevanten Siebel-Tabellen eingeräumt:

```
GRANT SELECT ON s_contact TO
dwh_sa_crm_role;
GRANT SELECT ON s_org_ext TO
dwh_sa_crm_role;
GRANT SELECT ON s_asset TO dwh_
sa_crm_role;
```

Schließlich muss noch der Eigentümer der transportierten Tabellen in der Data-Warehouse-Datenbank erzeugt werden:

```
CREATE USER crm IDENTIFIED BY
thisIsASecretPassword;
```

Dieser User benötigt im Übrigen weder das „create session”- noch das „create table”-Privileg.

Solange der Aktualisierungsprozess ausgeführt wird, sind die CRM-Daten in der Staging Area nicht verfügbar. Die Datendateien des CRM-Tablespace auf dem DWH-Host sind gegenwärtig exklusiv der physischen Standby-Datenbank des OLTP-Systems (hier OLTP_SITE2) zugeordnet. Das Konzept kann natürlich dahingehend erweitert werden, mehrere Tablespaces sogar von mehreren Datenbanken zu transferieren.

Um den Aktualisierungsprozess zu beenden, sind die folgenden Schritte nötig:

Zunächst wird die physische Standby-Datenbank in eine Snapshot-Standby-Datenbank konvertiert sowie anschließend der betroffene Tablespace von OLTP_SITE2 auf Read-only gesetzt und via Datapump und Datenbank-Link ins Data-Warehouse eingebunden:

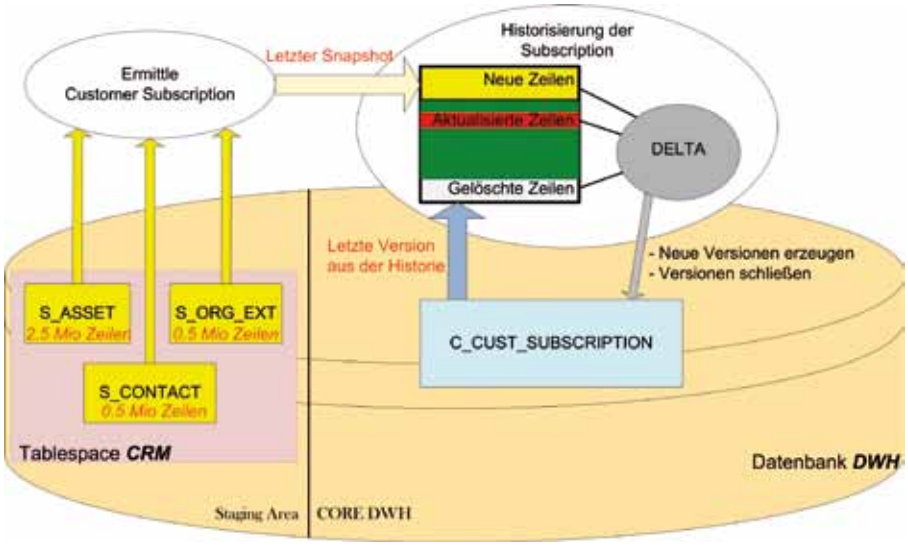

Abbildung 5: ETL vergleicht den neuen „Abzug“ der drei „Customer Subscription“-Tabellen mit den letzten bekannten Daten aus dem Core Data Warehouse und erzeugt wenn nötig neue Versionen der Daten

```
SQL> alter tablespace crm read
only;
# impdp system@DWH logfile=imp_
crm.log network_link=OLTP_SNAP
transport_tablespaces=CRM
transport_datafiles=d:\oradata\
oltp\crm01oltp.dbf
```

Wichtig: Sofern 11.2.0.2 genutzt wird, ist es aufgrund von Bug 10185688 nötig, entweder XDB in der Quelldatenbank oder den verfügbaren Patch zu installieren.

Beim Transfer können die zu transferierenden Tabellen angegeben werden. Außerdem kann man wählen, ob man Indizes, Objekt-Privilegien, Tabellen-Trigger oder Tabellen-Constraints importieren möchte.

Um den Aktualisierungsprozess zu starten, sind folgende Maßnahmen erforderlich:

Zunächst wird der CRM-Tablespace aus der DWH-Datenbank entfernt:

```
SQL> drop tablespace crm inclu-
ding contents;
DGMGRL> convert database ‚OLTP_
SITE2‘ to physical standby
```

Es wird nun geprüft, ob die Daten im CRM-Tablespace der Standby-Datenbank wieder aktuell genug für eine erneute Einbindung in das DWH sind. Dazu wird der Data Guard Broker befragt:

```
DGMGRL> show database ‚OLTP_
SITE2‘;
Database - OLTP_SITE2
  Role:            PHYSICAL
STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds
  Apply Lag:       11 minutes
23 seconds
  Real Time Query: OFF
  Instance(s):
    oltp

Database Status:
SUCCESS
```

Als Aktualisierungsmethode setzt die Standby-Datenbank das äußerst effiziente parallele Media-Recovery ein. Es arbeitet blockorientiert und ist damit deutlich schneller und Ressourcen-schonender als die Mechanismen von GoldenGate oder Streams, bei denen die Änderungen aus den Logs extrahiert und per SQL Datensatz für Datensatz abgearbeitet werden.

Fazit

Eine Staging Area profitiert von der beschriebenen Lösung in folgenden Fällen am meisten:

- Das Quellsystem weist große Datenmengen mit komplexen Beziehungen auf: Es gibt Datenextraktion aus dedizierten Online-Transaction-Applikationen wie CRM oder SCM (Supply Chain Management), bei denen mehrere Millionen individuelle Kunden, Verträge, Produkte oder Produktkomponenten, Warenlager etc. verwaltet werden
- Die Anforderungen an die Aktualität im Data Warehouse sind besonders hoch

Literatur und Links

1. Trivadis Whitepaper „Solution for Staging Area in Near Real-Time DWH – Efficient in Refresh and Easy to Operate”: http://www.trivadis.com/uploads/tx_cabagdownloadarea/WhitePaper_Solution_for_Staging_Area_01.pdf
2. Oracle Data Guard Concepts and Administration: http://download.oracle.com/docs/cd/E11882_01/server.112/e17022/toc.htm
3. Oracle Database PL/SQL Packages and Types Reference – Chapter 46: http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16760/d_datpmp.htm
4. Data Warehousing mit Oracle – Business Intelligence in der Praxis. Chapter 3.4. Jordan et al., Hanser, 2011

Karol Hajdu
Trivadis Delphi GmbH
karol.hajdu@trivadis.com
Mathias Zarick
mathias.zarick@trivadis.com