Oracle Database 10*g* Data Guard

Claudia Hüffer, Mike Dietrich
Oracle Deutschland GmbH
Hamburg und München

Schlüsselworte
Standby-Datenbank, Schatten-Datenbank, Hochverfügbarkeit, Real-Time-Apply, No-Data-Loss, init.ora-Parameter, Logical Standby Datenbank, Physical Standby Datenbank, Data Guard, Data Guard Broker, Data Guard Manager, Real Application Cluster (RAC), Maximum Availability Architecture (MAA).

Zusammenfassung
Der vorliegende Beitrag gibt einen technischen Überblick über die Erweiterungen der Oracle Data Guard Technologie mit Oracle Database 10*g*. Zunächst wird die Basis-Architektur von Oracle Data Guard erläutert. Dabei werden die beiden Varianten der Standby Datenbank, Logical und Physical Standby, vorgestellt.
Im Anschluß daran werden die Neuerungen im Bereich Data Guard mit Oracle Database 10*g* erläutert. Dazu gehören neue init.ora-Parameter, Realtime Apply, Erweiterungen der Logical Standby Datenbank und Erweiterungen des Data Guard Brokers und des Data Guard Managers.
Im Anhang sind Auszüge aus den Initialisierungsdateien für eine Beispiel-Konfiguration bestehend aus einer Primär-Datenbank, einer Physical und einer Logical Standby Datenbank aufgelistet.

Einleitung
Die Bereitstellung von hoch- oder höchstverfügbaren Systemen gewinnt zunehmend an Bedeutung. Möchte man ein hochverfügbares System bereitstellen, so muß dieses System in der Lage sein, auf alle möglichen Fehlerszenarien reagieren zu können.
Betrachtet man die Ursachen für Systemausfälle, so kann man diese zunächst grob in planmäßige und unplanmäßige Systemausfälle unterteilen.
Unplanmäßige Ausfallzeiten werden durch System/Hardware-Ausfälle, menschliche Fehler und Desaster-Situationen verursacht. Planmäßige Ausfallzeiten können durch Wartungsarbeiten an der Hardware oder Software entstehen.
Oracle bietet eine Reihe von Funktionalitäten, um Ausfallzeiten in allen Szenarien verhindern oder minimieren zu können. *Abbildung 1* zeigt die Szenarien und die entsprechenden Oracle Lösungen.
Neben diesen sehr spezifischen Lösungen können durch Einsatz von Standby Datenbank-Technologie in vielen Fällen Ausfallzeiten verringert und das Produktiv-System entlastet werden. Darüberhinaus läßt sich durch Kombination eines RAC-Produktionssystems mit einer RAC-Data Guard-Umgebung ein höchstverfügbares Datenbank-System aufbauen. Die verwendete Architektur entspricht dann der von Oracle vorgeschlagenen Maximum Avaliability Architecture (MAA), die

maximalen Schutz bietet. Weitergehende Informationen zur MAA finden sich im Oracle TechNet (otn.oracle.com). *Abbildung 2* zeigt die Einsatz-Möglichkeiten von Data Guard in den verschiedenen Fehler-Szenarien.

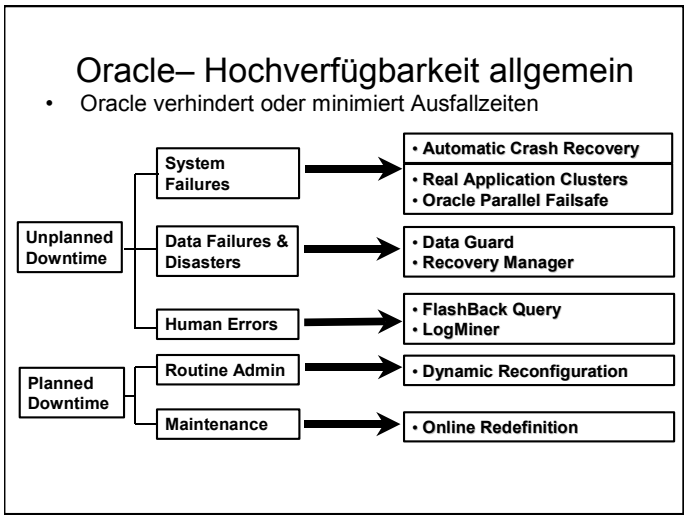

Abbildung 1: Hochverfügbarkeit mit Oracle, allgemein

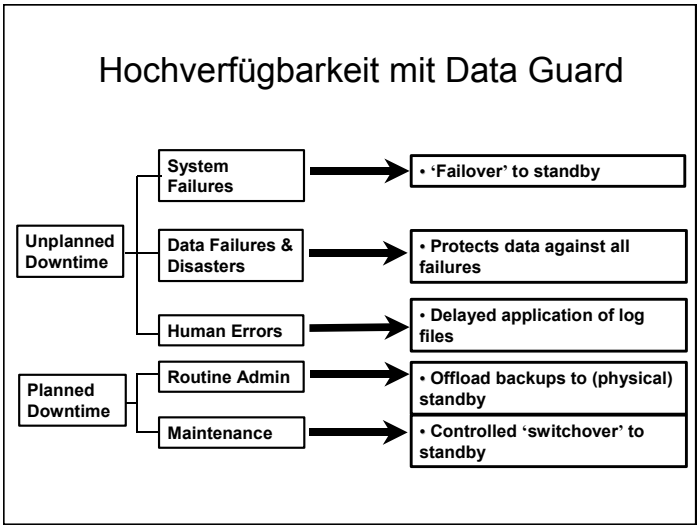

Abbildung 2: Hochverfügbarkeit mit Oracle Data Guard

Die Oracle Standby-Technologie ist seit ihrer Einführung mit Oracle 7.3 immer weiter entwickelt worden. Mit Oracle 8.0 bzw. 8*i* wurden der sogenannte „managed recovery mode“ und die Möglichkeit des Readonly-Zugriffes für Lesezugriffe eingeführt. Mit Oracle 9*i* Release 1 wurde

durch die Einführung der Broker-Architektur die Administration der Standby-Umgebung deutlich vereinfacht. Durch Erweiterungen des Log Transport Services konnte die Datenverfügbarkeit auf dem Standby-System gesteigert werden, so dass nun sogenannte No-Data-Loss-Umgebungen aufgebaut werden können. Ferner wurden verschiedene Protection Modi und das sogenannte *Graceful Switchover* eingeführt.
Mit Oracle 9*i* Release 2 wurde als weitere Variante der Standby-Datenbank die Logical Standby Datenbank eingeführt. Der Einsatz-Schwerpunkt der Logical Standby Datenbank ist im Bereich Reporting zu sehen, wohingegen die Physical Standby Datenbank als blockweise identische Kopie der Pirmär-Datenbank im Bereich Desaster-Schutz die ideale Lösung darstellt.

Die Erweiterungen von Oracle Data Guard mit Oracle Database 10*g* liegen im Bereich der erleichterten Administrierbarkeit sowohl im Single Instance, als auch im RAC-Umfeld, erweiterten Support für verschiedene Datentypen bei der Logical Standby Datenbank und Realtime Apply für (Nahezu-)Echtzeit Reporting und schnelleres Switchover/Failover.
Desweiteren wurde mit Oracle Database 10*g* die „flashback database“ Funktionalität eingeführt. Diese läßt sich auch in Verbindung mit Data Guard einsetzen, wenn es um die Bereiche „Human Errors“ oder Neuaufbau nach einem Failover geht.

Architektur
Bevor auf die einzelnen Neuerungen mit Oracle Database 10*g* eingegangen wird, erläutert ein Überblick das Konzept der Data Guard Architektur.

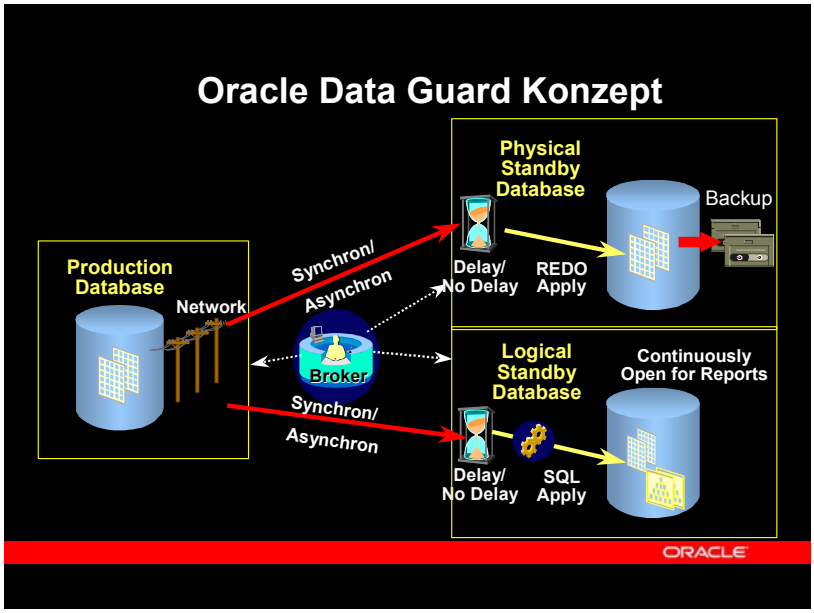

Abbildung 3: Oracle Data Guard Konzept

Ausgehend aus einem Backup der Primär-Datenbank können eine Physical und/oder eine Logical Standby Datenbank aufgebaut werden. Insgesamt können bis zu 9 Standby Datenbanken direkt an eine Primär Datenbank angeschlossen werden. Sollten weitere Standby Datenbanken benötigt werden, so können diese kaskadierend an die bestehenden Standby Datenbanken angeschlossen werden. Das Primär System kann eine Single-Instance oder RAC Datenbank sein. Ebenso kann die Standby Datenbank als Single-Instanz oder als RAC betrieben werden.

Änderungen, die auf dem Primär-System stattfinden, werden durch die sogenannten Log-Transport Services auf die Standby-Datenbank übertragen. Diese Übertragung kann entweder durch den Archiver (ARCH) nach einem Logswitch auf dem Primärsystem durchgeführt werden, oder - zeitnaher - beim Schreiben der Inhalte des Logbuffers durch den Logwriter (LGWR). Mit Verwendung des LGWR kann man je nach Konfiguration den Datenverlust bei Totalausfall des Primär-Systems minimieren oder ganz ausschließen (no-data-Loss-Umgebung, Maximum Protection Mode).
Zur Erleichterung der Administration einer Data Guard Umgebung kann die Broker-Architektur eingesetzt werden. Ein zusätzlicher Prozess, der Data Guard Broker (DMON), überwacht regelmäßig das Gesamt-System. Der Data Guard Broker kann über zwei APIs angesprochen werden.Das sind das Command-Line-Interface `dgmgrl` und die graphische Oberfläche Data Guard Manager (DGM), die in den Enterprise Manager integriert ist. Der DGM erlaubt mithilfe von Assistenten auch das Erstellen von Standby Datenbanken.

Physical Standby Datenbank

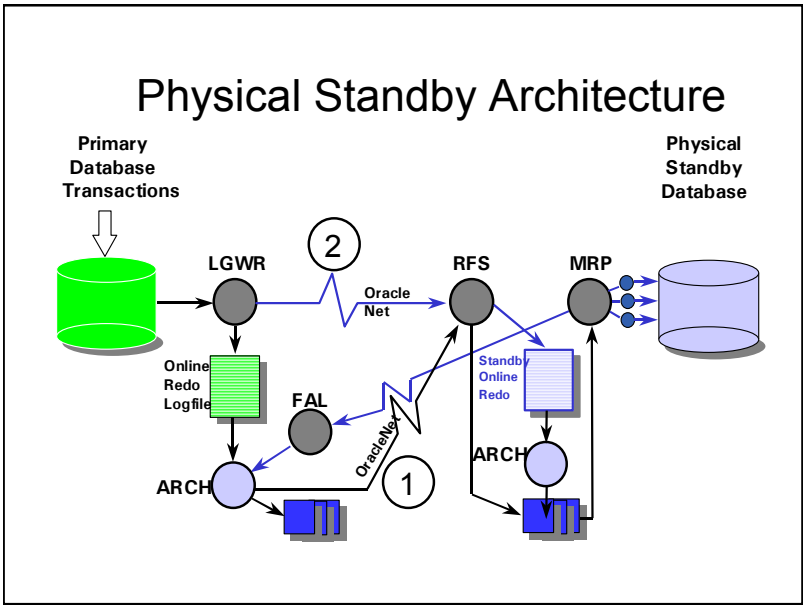

Abbildung 4: Physical Standby Datenbank Architektur

Transaktionen, die in der Primärdatenbank stattfinden, werden im current Online Redo Log protokolliert. Bei Verwendung des Archivers als Log Transport Service ① wird bei einem Logswitch das gerade beschriebene Online Redo Log lokal in ein Verzeichnis weggesichert und mit Oracle Net zur Standby-Datenbank übertragen. Dort nimmt der RFS (remote file service-Prozess) die Informationen entgegen und legt die Datei im „standby_archive_dest"-Verzeichnis ab. Sie steht dann für das Apply zur Verfügung.

Alternativ kann auch der Logwriter (LGWR) den Log Transport übernehmen ②. In diesem Fall wird immer dann, wenn der Logwriter in das current Online Redo Log schreibt, also wenn ein Commit stattgefunden hat oder der Inhalt des Logbuffers weggeschrieben wird, dieselbe Information mit Oracle Net auf die Standby Datenbank übertragen. Dort nimmt der RFS Prozess die Informationen entgegen. Werden auf der Standby Datenbank sogenannte Standby Online Redo Logs (SRL) verwendet, dann wird die Information dort eingetragen. Werden keine SRLs verwendet, wird direkt in ein archiviertes Redo Log auf der Standby-Seite geschrieben. Bei Verwendung des Logwriters liegen die Informationen über Änderungen direkt auf der Standby-Seite vor. Somit kann auch im Falle eines Totalausfalles der Primär Datenbank, bei Verwendung von SRLs, bis zur letzten comitteten Transaktion recovered werden, so dass keine abgeschlossenen Transaktionen verloren gehen. Kommt es nun zu einem Logswitch auf der Primärseite, wird dieser direkt auf dem Standby-System ausgelöst und die entstehende archivierte Redo Log Datei steht spätestens jetzt für das Apply zur Verfügung. (Verwendung von Real-Time-Apply siehe weiter unten).

Möchte man die Standby Datenbank dazu einsetzen, Systemausfälle durch menschliche Fehler (versehentliches Löschen von Daten) zu vermeiden, so kann man das Apply optional durch ein sogenanntes Delay verzögern. Bei der Physical Standby Datenbank ist der MRP (Managed Recovery Process) für das Apply verantwortlich. D.h., die Änderungen, die auf der Primär Datenbank stattgefunden haben, werden durch ein Recovery in der Physical Standby Datenbank nachgezogen. Daher ist die Physical Standby Datenbank blockweise 1:1 identisch mit der Primär Datenbank und erlaubt zum einen einen optimalen Desaster-Schutz, zum anderen kann die Physical Standby Datenbank für Backups herangezogen werden und somit das Primärsystem entlasten.

Logical Standby Datenbank

Bei der Logical Standby Datenbank kann ebenfalls zwischen dem Archiver und Logwriter als Log Transport Service gewählt werden. Auch stehen die Informationen spätestens nach einem Logswitch für das Apply zur Verfügung. (Realtime Apply siehe weiter unten).

Der grundlegende Unterschied zwischen der Physical und der Logical Standby Datenbank liegt im **Apply** Mechanismus. Während bei der Physical Standby Datenbank das Apply in einer blockweise identischen Datenbank durch ein Recovery stattfindet, ist die Logical Standby Datenbank nicht mehr blockweise 1:1 identisch. Sie befindet sich auch nicht im Recovery Mode, sondern steht auch während des Applys den Anwendern lesend und unter bestimmten Voraussetzungen auch schreibend zur Verfügung. Für das Apply werden mit Logmining-Technologie die Inhalte der Redo

Logs analysiert und aus diesen Informationen SQL-Statements rekonstruiert. Diese SQL-Statements werden dann in der offenen Datenbank angewendet.
Dadurch kann die Logical Standby Datenbank während des Apply für Reportingaufgaben genutzt werden. Ferner erlaubt das PL-SQL-Paket DBMS_LOGSTDBY eine Manipulation des Apply-Mechanismus (z.B. Definition von Filtern).
Da die Logical Standby Datenbank zur Zeit noch nicht alle Datentypen unterstützt, kann sie nur unter bestimmten Voraussetzungen auch als Desaster-Schutz eingesetzt werden.

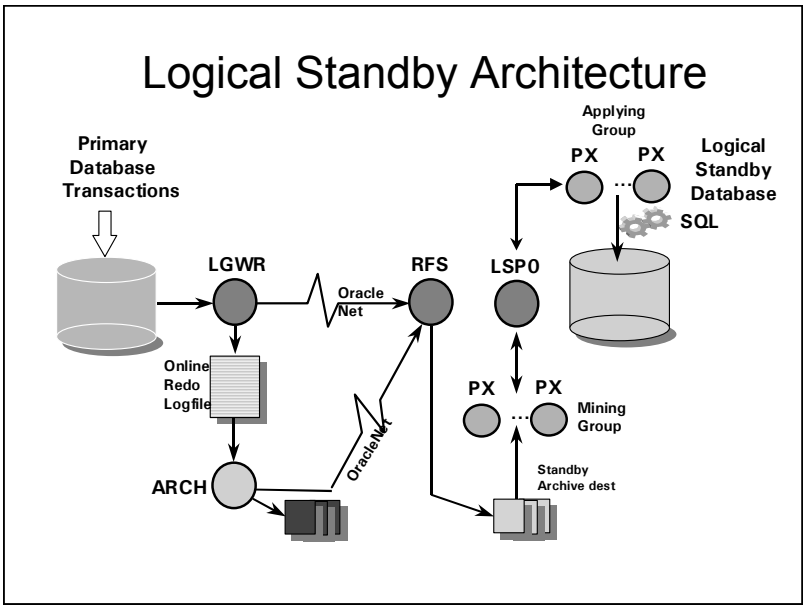

Abbildung 5: Logical Standby Datenbank Architektur

Folgende Datentypen werden von der Logical Standby Datenbank unterstützt:
`CHAR, NCHAR, VARCHAR2 und VARCHAR, NUMBER, DATE, TIMESTAMP (WITH TIME ZONE/WITH LOCAL TIMEZONE), INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND, RAW, CLOB, NCLOB, LONG, LONG RAW, BINARY_FLOAT, BINARY_DOUBLE`

Folgende Datentypen werden derzeit nicht unterstützt: `BFILE, ROWID, UROWID,` user-defined types, object types, `REF`s, `VARRAY`s, nested tables, XMLType.
Zur Überprüfung, ob in der Primär Datenbank nicht unterstützte Datentypen verwendet werden, kann der DBA die View `DBA_LOGSTDBY_UNSUPPORTED` abfragen.

Neuerungen in Oracle Database 10*g* Data Guard

Im Folgenden werden die Erweiterungen von Oracle Database 10*g* Data Guard beschrieben.

Data Guard Broker-Unterstützung für RAC Systeme

Der Data Guard Broker unterstützt mit Oracle Database 10*g* auch Data Guard Konfigurationen, bei denen die Primär Datenbank und/oder die Standby Datenbank als RAC-System betrieben werden.
Bis Oracle 9i einschließlich mußten Data Guard Konfigurationen, bei denen RAC Datenbanken beteiligt waren, mit SQL*Plus manuell administriert werden.
Die folgende Abbildung zeigt den neuen Data Guard Manager, der in den Enterprise Manager integriert ist. Die Data Guard Konfiguration besteht aus einer RAC Primär Datenbank, einer Physical und einer Logical Standby Datenbank.

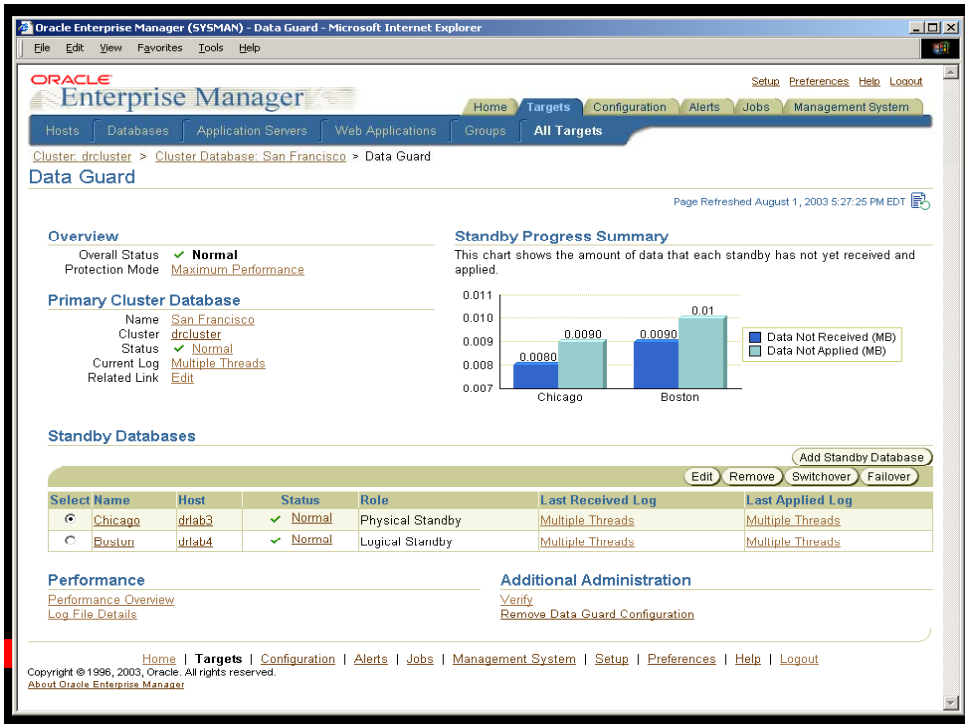

Abbildung 6: Data Guard Manager GUI

Neue init.ora/spfile-Parameter

Mit Oracle Database 10*g* sind eine Reihe neuer Datenbank-Parameter für Data Guard eingeführt worden. Desweiteren wurde die Liste der Attribute zum Parameter `log_archive_dest_n` erweitert.
Durch diese neuen Parameter ist es nun möglich, eine Konfiguration so aufzubauen, dass auch beim Rollenwechsel nach einem Failover oder Switchover keinerlei Parameter-Anpassungen vorgenommen werden müssen. D.h., die Initialisierungs-Parameter der beteiligten Datenbanken haben für alle Rollen Gültigkeit.

Die neuen Parameter werden anhand eines Beispiels erläutert. Die Beispiel-Konfiguration besteht aus einer Single-Instance Primär Datenbank, einer Physical und einer Logical Standby Datenbank.

In dieser Konfiguration ergeben sich die für Data Guard relevanten Parameter für die Primär Datenbank wie folgt:

```
*.db_unique_name='Redwood_Shores'
*.db_file_name_convert='/oradata/datafiles/tdsphys','/oradata/datafiles/tdsprod',
                       '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsprod'
*.db_recovery_file_dest='/oradata/flash_recovery_area'
*.db_recovery_file_dest_size=2147483648
*.fal_client='tdsprod'
*.fal_server='tdsphys','tdsrpt'
*.log_archive_config='DG_CONFIG=(Redwood_Shores,Austin_Datacenter,SanFrancisco)'
*.log_archive_dest_1='LOCATION=/app/oracle/product/admin/tdsprod/archive
                      VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_2='LOCATION=/app/oracle/product/admin/tdsprod/archive/SRLs
                      VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLES) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_3='SERVICE=tdsphys LGWR async
                      VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=Austin_Datacenter'
*.log_archive_dest_4='SERVICE=tdsrpt
                      VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=SanFrancisco'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_dest_state_2='ENABLE'
*.log_archive_dest_state_3='ENABLE'
*.log_archive_dest_state_4='ENABLE'
*.log_archive_format='arch_%t_%s_%r.arc'
*.log_file_name_convert='/oradata/datafiles/tdsphys','/oradata/datafiles/tdsprod',
                        '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsprod'
*.standby_archive_dest='/app/oracle/product/admin/tdsprod/archive'
*.standby_file_management='auto'
```

Die neuen Parameter oder Attribute sind fett unterlegt.
`db_unique_name` ist die eindeutige Bezeichnung für eine Site (=Datenbank) in einer Data Guard Konfiguration.
`log_archive_config` gibt an, welche Sites Bestandteil der Data Guard Konfiguration sind. Im obigen Beispiel besteht die Konfiguration aus den drei Sites `Redwood_Shores`, `Austin_Datacenter` und `SanFrancisco`. Die Namen für die Sites werden bei den jeweiligen Datenbanken mit `db_unique_name` eindeutig festgelegt.
Der Site-Name taucht dann auch noch mal beim `log_archive_dest_`*`n`* Parameter als neues Attribut `db_unique_name` auf.
Durch die Verwendung des neuen `VALID_FOR`-Attributes des `log_archive_dest_`*`n`*-Parameters, kann man eine Data Guard Umgebung ohne Parameter-Anpassungen beim Rollenwechsel betreiben.
Erläuterung der obigen Parameter:
Bei `log_archive_dest_1` ist angegeben, dass diese Einstellung für Online (Redo) Logfiles gilt, egal, in welcher Rolle sich die Datenbank befindet (ALL_ROLES). Der Parameter `log_archive_dest_2` dagegen wird nur dann ausgewertet, wenn sich die Datenbank in der Standby-Rolle befindet (STANDBY_ROLE) und dann gilt der Parameter auch nur, wenn Standby Redo Logs verwendet werden. In dem Fall werden nach einem Logswitch die archivierten Redo Logs im Verzeichnis `/app/oracle/product/admin/tdsprod/archive/SRLs`

abgelegt.
`Log_archive_dest_3` und `_dest_4` beschreiben die Übertragung von Log-Informationen an Standby Datenbanken. Dies geschieht nur dann, wenn die Datenbank Primär-Datenbank ist (PRIMARY_ROLE). Im ersten Fall wird dazu der Logwriter (LGWR) verwendet und im zweiten Fall wird der Archiver (Default) verwendet.

Würde im obigen Beispiel nun ein Switchover stattfinden, wäre die Datenbank also Standby Datenbank, so würden die Parameter `dest_3` und `dest_4` nicht mehr ausgewertet.
Eine manuelle Anpassung von Parametern nach einem Switchover/Failover ist im Gegensatz zu 9*i* dann nicht mehr notwendig.

Die Parameter `log_file_name_convert` und `db_file_name_convert` sind nicht grundsätzlich neu. Neu ist lediglich, daß diese Parameter nun auch bei Logical Standby Datenbanken verwendet werden können.
`Log_archive_format`, `log_archive_dest_state_n`, `standby_file_management`, `standby_archive_dest`, `fal_server` und `fal_client` sind nicht neu, sondern nur der Vollständigkeit halber mit aufgelistet.

Mit Oracle Database 10*g* wurde auch eine Flash Recovery Area neu eingeführt. Diese wird für unterschiedliche Zwecke genutzt. Im Zusammenhang mit Data Guard kann man diese Flash Recovery Area in Verbindung mit der „flashback database"-Technologie verwenden. Dazu legt man mit `db_recovery_file_dest` das Verzeichnis und mit `db_recovery_file_dest_size` die Größe der Flash Recovery Area fest. Dort werden dann die für „flashback database" relevanten Dateien abgelegt.

Erweiterungen der Logical Standby Datenbank
Mit Oracle Database 10*g* sind einige Erweiterungen bei der Logical Standby Datenbank implementiert worden.

Der Aufbau einer Logical Standby Datenbank kann in 10*g* jetzt auch online erfolgen. Ein Stoppen der Primär Datenbank ist jetzt nicht mehr erforderlich. Beim manuellen Aufbau der Logical Standby Datenbank wurden außerdem die notwendigen Schritte deutlich vereinfacht.

Die Logical Standby Datenbank kann nun auf von dem Primärsystem abweichenden Verzeichnisstrukturen aufgebaut werden, ohne daß der DBA im laufenden Betrieb eingreifen muß. Ermöglicht wird dies dadurch, dass die Logical Standby Datenbank nun die Konvertierungsparameter `db_file_name_convert` und `log_file_name_convert` unterstützt. Setzt man `standby_file_management=AUTO`, wird beim Anlegen einer neuen Datendatei in der Primärdatenbank diese Aktion automatisch auch auf der Logical Standby Datenbank nachgezogen, selbst wenn die Strukturen voneinander abweichen.

Beginnend mit Oracle Database 10*g* unterstützt die Logical Standby Datenbank jetzt auch Standby Online Redo Logs. Dadurch wird die Datensicherheit bei der Logical Standby Datenbank gesteigert und es können sogenannte „no-data-loss-Umgebungen“ aufgebaut werden.
Sehr interessant ist der Einsatz von Standby Redo Logs bei der Logical Standby Datenbank in Verbindung mit dem Realtime Apply, das ein Nahezu-Echtzeit-Reporting ermöglicht.

Mit Oracle Database 10*g* wurde der Support für die verschiedenen Datentypen erweitert. Neu ist der Support für IOTs (unter bestimmten Voraussetzungen) und für den Spaltentyp LONG. Damit wird der Einsatz von Logical Standby Datenbanken auch bei vielen „Alt“-Systemen, die noch LONG-Spalten verwenden, möglich.

Real Time Apply
Bis Oracle 9*i* einschließlich erfolgte das Apply der Informationen aus den archivierten Redo Logs erst **nach** einem Logswitch – unabhängig davon, ob mit dem LGWR oder dem ARCH als Log Transport Service gearbeitet wurde. Mit Oracle Database 10*g* ist erstmals ein Realtime Apply möglich. Dies bedeutet, dass Informationen, die in den SRLs stehen, bei Verwendung des LGWR direkt für das Apply herangezogen werden.
Dadurch ergeben sich zwei Vorteile: zum einen ist der Zeitaufwand bei einer Switchover- oder Failover-Situation deutlich geringer, zum anderen erlaubt das Realtime Apply bei der Logical Standby Datenbank (nahezu) Echtzeit Reporting und damit aktuelle Daten.

Bei der Physical Standby Datenbank wird das Realtime Apply gestartet mit:
```
alter database recover managed standby database using current logfile;
```

Bei der Logical Standby Datenbank wird das Realtime Apply gestartet mit:
```
alter database start logical standby apply immediate;
```

Flashback Database und Data Guard
Flashback Database ist eine Funktionalität, die mit Oracle Database 10*g* eingeführt wird. Diese Technologie erlaubt das Zurückversetzen der **gesamten** Datenbank in einen Zustand in der Vergangenheit. Dazu muß zuvor die Flashback Funktionalität in der Datenbank mit:
`alter database flashback on;` aktiviert worden sein. Es werden dann in der Flash Recovery Area entsprechende Flashback Log Dateien abgelegt, die ein Zurückversetzen der gesamten Datenbank erlauben. Wie weit der DBA in die Vergangenheit zurückgehen kann, hängt zum einen vom Parameter `db_flashback_retention_target` und zum anderen von der Größe der Recovery Area ab. Die Flash Recovery Area kann für die Flashback Logs, die inkrementellen Backups und die archivierten Redo Logs verwendet werden.

Flashback Database kann in Verbindung mit Data Guard u.a. in folgenden Szenarien genutzt werden: Abfangen von „human errors“ und Wiederaufbau einer Standby Datenbank nach Failover.
Benutzerfehler konnte man in 9*i* und kann man auch in 10*g* noch durch ein *delayed apply* abfangen. Alternativ dazu kann der DBA in10*g* auch „Flashback Database“ verwenden. Dazu wird auf der

Standby Datenbank zunächst mit `alter database flashback on;` der „Flashback Database“-Mechanismus eingeschaltet. Kommt es nun in der Primär Datenbank zu einem Benutzerfehler, z.B. einem versehentlichen Löschen von Dateninhalten, so kann man die Standby Datenbank mit `flashback database to scn <scn_vor_fehler>;` auf einen Zeitpunkt vor dem Löschen zurücksetzen. Dann öffnet man die Standby Datenbank *readonly* und überprüft das Ergebnis. Hat man den gewünschten Zeitpunkt erreicht, kann man entweder ein Failover auf die Standby Datenbank durchführen, oder auch nur die fehlenden Informationen exportieren und in die Primär Datenbank importieren. Anschließend würde man das Recovery wieder starten und bis zum aktuellen Stand recovern. Für welche Lösung sich der DBA entscheidet, hängt von der Komplexität des fehlerhaften Löschprozessen oder Updates ab.

Sehr interessant wird die „Flashback Database“ Funktionalität im Zusammenhang mit Failover-Situationen. Bis Oracle 9*i* einschließlich ist die ehemalige Primär Datenbank nach einem Failover zur Standby Datenbank für das Gesamt-System verloren. Die alte Primär Datenbank konnte nach einem Failover nicht mehr in das Gesamt-System eingebettet werden und es mußte eine neue Standby Datenbank aus einem Backup der neuen Primär Datenbank aufgebaut werden.

Mit Oracle Database 10*g* kann in all den Fällen, bei denen nach einem Failover doch wieder auf die alte Primär Datenbank zugegriffen werden kann, und bei denen auf der Primär Datenbank **vor** dem Failover „Flashback Database“ aktiviert war, diese Datenbank gerettet werden. Das macht ein sogenanntes *Reinstantiate* überflüssig.
Nach einem Failover ermittelt der DBA dazu zunächst auf der neuen Primär Datenbank die SCN, zu der das Failover stattgefunden hat (Beispiel mit Physical Standby Datenbank):

```
select standby_became_primary_scn from v$database;
```

Die alte Primär Datenbank wird nun „gemounted“ und in die Vergangenheit mit `flashback database to scn <standby_became_primary>;` zurückversetzt. Anschließend wird dort ein Standby-Controlfile erzeugt, die alte Primär Datenbank als Physical Standby Datenbank *gemounted* und die in der Zwischenzeit aufgelaufenen archivierten Redo Logs eingespielt. Somit kann der DBA ohne ein *Reinstantiate* zu einer Data Guard Konfiguration zurückkehren. Wenn gewünscht, kann dann ein Switchover zur Ursprungs-Situation durchgeführt werden (nähere Informationen hierzu befinden sich im Kapitel 10 der Data Guard Dokumentation).

Anhang
Die dem Paper zugrundeliegenden Tests wurden mit Oracle Database 10*g* Beta2 auf RedHat Linux durchgeführt. Es handelte sich dabei um eine Data Guard Umgebung bestehend aus eine Primär Datenbank (tdsprod) , einer Physical Standby Datenbank (tdsphys) und einer Logical Standby Datenbank (tdsrpt). Die für Data Guard relevanten init.ora Parameter sind hier aufgelistet:

Auszug aus der inittdsprod.ora:

```
*.db_allowed_logon_version=10
*.db_file_name_convert='/oradata/datafiles/tdsphys','/oradata/datafiles/tdsprod',
```

```
                        '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsprod'
*.db_name='tdsprod'
*.db_recovery_file_dest='/oradata/flash_recovery_area'
*.db_recovery_file_dest_size=2147483648
*.db_unique_name='Redwood_Shores'
*.fal_client='tdsprod'
*.fal_server='tdsphys','tdsrpt'
*.log_archive_config='DG_CONFIG=(Redwood_Shores,Austin_Datacenter,SanFrancisco)'
*.log_archive_dest_1='LOCATION=/app/oracle/product/admin/tdsprod/archive
              VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_2='LOCATION=/app/oracle/product/admin/tdsprod/archive/SRLs
              VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLES) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_3='SERVICE=tdsphys LGWR async
              VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=Austin_Datacenter'
*.log_archive_dest_4='SERVICE=tdsrpt
              VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=SanFrancisco'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_dest_state_2='ENABLE'
*.log_archive_dest_state_3='ENABLE'
*.log_archive_dest_state_4='ENABLE'
*.log_archive_format='arch_%t_%s_%r.arc'
*.log_file_name_convert='/oradata/datafiles/tdsphys','/oradata/datafiles/tdsprod',
                        '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsprod'
*.standby_archive_dest='/app/oracle/product/admin/tdsprod/archive'
*.standby_file_management='auto'
```

Auszug aus der inittdsphys.ora:

```
*.db_allowed_logon_version=10
*.db_file_name_convert='/oradata/datafiles/tdsprod','/oradata/datafiles/tdsphys',
                        '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsphys'
*.db_name='tdsprod'
*.db_recovery_file_dest='/oradata/flash_recovery_area'
*.db_recovery_file_dest_size=2147483648
*.db_unique_name='Austin_Datacenter'
*.fal_client='tdsphys'
*.fal_server='tdsprod','tdsrpt'
*.instance_name='tdsphys'
*.log_archive_config='DG_CONFIG=(Redwood_Shores,Austin_Datacenter,SanFrancisco)'
*.log_archive_dest_1='LOCATION=/app/oracle/product/admin/tdsphys/archive
                VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=Austin_Datacenter'
*.log_archive_dest_2='LOCATION=/app/oracle/product/admin/tdsphys/archive/SRLs
                VALID_FOR=(STANDBY_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=Austin_Datacenter'
*.log_archive_dest_3='SERVICE=tdsprod LGWR async
                VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_4='SERVICE=tdsrpt
                VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=SanFrancisco'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_dest_state_2='enable'
*.log_archive_dest_state_3='enable'
*.log_archive_dest_state_4='enable'
*.log_archive_format='arch_%t_%s_%r.arc'
*.log_file_name_convert='/oradata/datafiles/tdsprod','/oradata/datafiles/tdsphys',
               '/oradata/datafiles/tdsrpt','/oradata/datafiles/tdsphys'
*.standby_archive_dest='/app/oracle/product/admin/tdsphys/archive'
*.standby_file_management='auto'
```

Auszug aus der inittdsrpt.ora:

```
*.db_allowed_logon_version=10
*.db_file_name_convert='/oradata/datafiles/tdsprod','/oradata/datafiles/tdsrpt',
                        '/oradata/datafiles/tdsphys','/oradata/datafiles/tdsrpt'
```

```
*.db_name='tdsrpt'
*.db_recovery_file_dest='/oradata/flash_recovery_area'
*.db_recovery_file_dest_size=2147483648
*.db_unique_name='SanFrancisco'
*.fal_client='tdsrpt'
*.fal_server='tdsprod','tdsphys'
*.instance_name='tdsrpt'
*.log_archive_config='DG_CONFIG=(Redwood_Shores,Austin_Datacenter,SanFrancisco)'
*.log_archive_dest_1='LOCATION=/app/oracle/product/admin/tdsrpt/archive
              VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=SanFrancisco'
*.log_archive_dest_2='LOCATION=/app/oracle/product/admin/tdsrpt/archive/SRLs
              VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE) DB_UNIQUE_NAME=SanFrancisco'
*.log_archive_dest_3='SERVICE=tdsprod
              VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=Redwood_Shores'
*.log_archive_dest_4='SERVICE=tdsphys
              VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=Austin_Datacenter'
*.log_archive_dest_state_1='ENABLE'
*.log_archive_dest_state_2='enable'
*.log_archive_dest_state_3='enable'
*.log_archive_dest_state_4='enable'
*.log_archive_format='arch_%t_%s_%r.arc'
*.log_file_name_convert='/oradata/datafiles/tdsprod','/oradata/datafiles/tdsrpt',
                        '/oradata/datafiles/tdsphys','/oradata/datafiles/tdsrpt'
*.standby_archive_dest='/app/oracle/product/admin/tdsrpt/archive'
*.standby_file_management='auto'
```

Quellen:

Oracle Data Guard, Concepts and Administration, 10*g* Release 1 (10.1), Part No. B10823-01

Kontaktadressen:

Oracle Deutschland GmbH
Claudia Hüffer
Notkestr. 15
D-22607 Hamburg

Telefon: +49(0)40-89091135
Mobil: +40(0)177-5949135
Fax: +49(0)40-89091250
E-Mail: Claudia.Hueffer@oracle.com
Internet: www.oracle.com/de

Oracle Deutschland GmbH
Mike Dietrich
Riesstr. 25
D-80992 München

Telefon: +49(0)89-14301037
Mobil: +40(0)177-5940026
Fax: +49(0)89-14302977
E-Mail: Mike.Dietrich@oracle.com