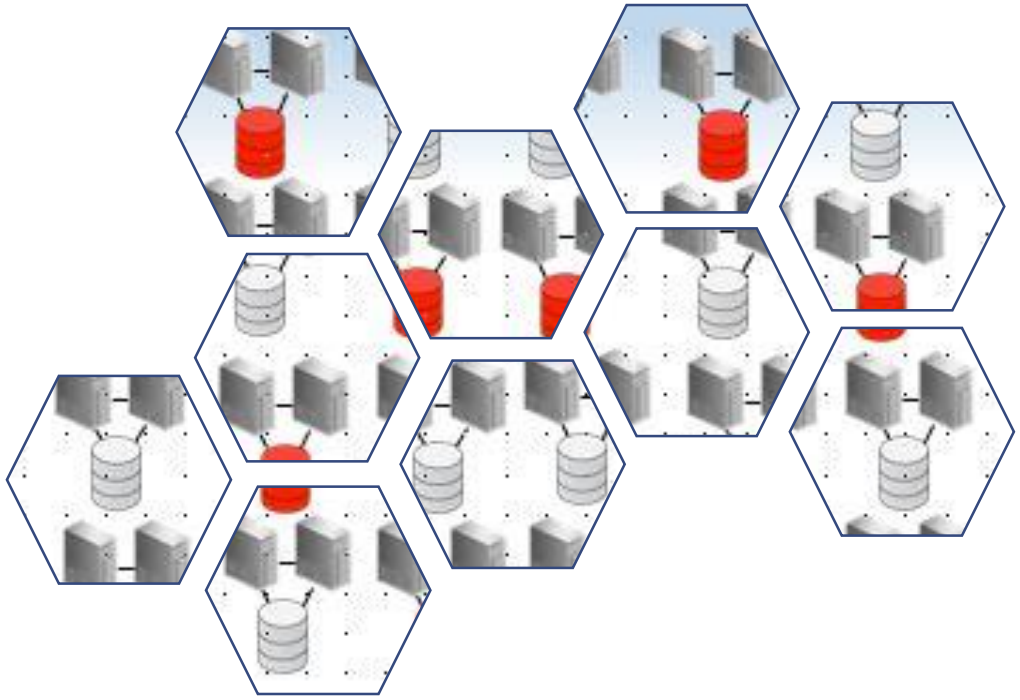
Rolling Patching mit DBMS_ROLLING und Active Data Guard

Nikolaus Eichler
Selbständiger Berater
nik.eichler@gmail.com



Agenda

- Rolling Patching – Historie
- Einsatzzweck des DBMS_ROLLING Packages
- Voraussetzungen für das Rolling Patching
- Anwendungsbereiche für das DBMS_ROLLING Package
- Ablauf der Prozedur-Aufrufe

Rolling Patching - Historie

- 11.2.0.1 und älter: Primary und Physical Standby Datenbank mussten den gleichen Release-Stand haben
- Seit 11.2.0.2: Data Guard Standby-First Installable Patches für Physical Standby Datenbanken
- Seit 10.2.3: Logical Standby Datenbank
- Seit 11.1: Transient Logical Standby Datenbank und das 'physru' Bourne Shell Script
- Seit 12.1: DBMS_ROLLING Package – ohne Broker Unterstützung
- Seit 12.2: DBMS_ROLLING Package – mit Broker Unterstützung

Einsatzzweck des DBMS_ROLLING Packages

Datenbank Upgrades und Patching mit DBMS_ROLLING

- Automatisierung zahlreicher manueller Schritte bei Rolling Operationen mit Active Data Guard und Transient Logical Standby
- Rolling Upgrades werden einfacher und robuster

Anwendungsbereiche

- Upgrades von Datenbanken
- Online Patching von Datenbanken
- Nicht-partitionierte Tabellen partitionieren
- BasicFiles LOBs in SecureFiles LOBs umwandeln
- CLOBs abgespeicherte XMLType-Dokumente als Binary XML abspeichern
- Tabellen mit OLTP Compression komprimieren

1. Voraussetzungen für den Einsatz von DBMS_ROLLING

- Active Data Guard Lizenz
- Protection Level muss entweder **Maximum Availability** oder **Maximum Performance** sein
- Flashback Database muss auf allen beteiligten Datenbanken eingeschaltet sein
- Das '**Fast Start Failover**' Feature muss '*disabled*' sein (*gilt auch noch für 18c*)

2. Voraussetzungen für den Einsatz von DBMS_ROLLING

Feststellen, ob 'Logical Apply' bestimmte Datentypen nicht replizieren kann

Ein Abfrage auf DBA_LOGSTDBY_EDS_SUPPORTED zeigt, welche Tabellen Kandidaten für **Extended Datatype Support** (EDS) sind:

```
SELECT * FROM DBA_ROLLING_UNSUPPORTED;
```

Extended Datatype Support

Das DBMS_LOGSTDBY Package

Beispiel

Auf der **Primary** Datenbank werden Trigger und Schattentabellen erzeugt:

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE',
table_name =>'CUSTOMERS');
```

Extended Datatype Support

Das DBMS_LOGSTDBY Package

Beispiel

Auf der **Primary** Datenbank werden Trigger und Schattentabellen erzeugt:

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE',
table_name =>'CUSTOMERS');
```

Auf der Logical **Standby**-Datenbank wird zunächst das Apply gestoppt. Dann werden mit:

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE', -
table_name =>'CUSTOMERS',p_dblink => 'dblink_to_primary');
```

die Tabellendaten auf der Logical Standby bereitgestellt und die Replikation aktiviert.

Extended Datatype Support

Das DBMS_LOGSTDBY Package

Beispiel

Auf der **Primary** Datenbank werden Trigger und Schattentabellen erzeugt:

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE',
table_name =>'CUSTOMERS');
```

Auf der Logical **Standby**-Datenbank wird zunächst das Apply gestoppt. Dann werden mit:

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE', -
table_name =>'CUSTOMERS',p_dblink => 'dblink_to_primary');
```

die Tabellendaten auf der Logical Standby bereitgestellt und die Replikation aktiviert.

Abschließend wird das Apply wieder gestartet.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

Nicht unterstützte Datatypen in einer Logical Standby Database (12.2)

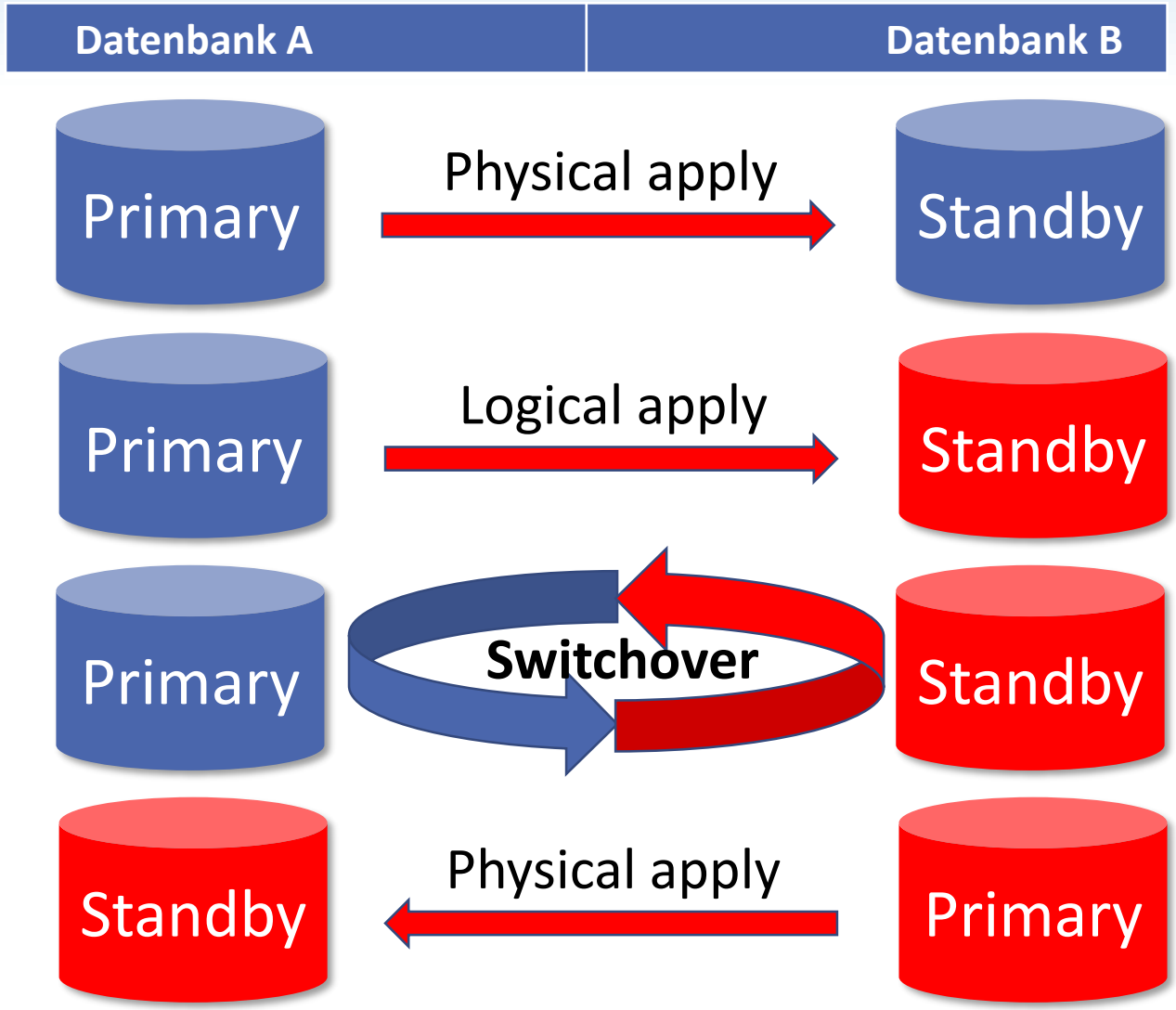
- BFILE
- ROWID, UROWID
- Nested tables
- Objects with nested tables
- Identity columns

Neu in DBMS_ROLLING 12.2

- Ab Oracle RDBMS 12c Release 2 (*12.2.0.1*) kann der Data Guard Broker während eines DBMS_ROLLING Rolling Upgrades eingeschaltet bleiben
- Upgrade von Database Vault möglich
- Upgrade von Label Security möglich

Rolling Patching - Prinzip

=rel. n =rel. n+1



Rolling Patching - Prinzip

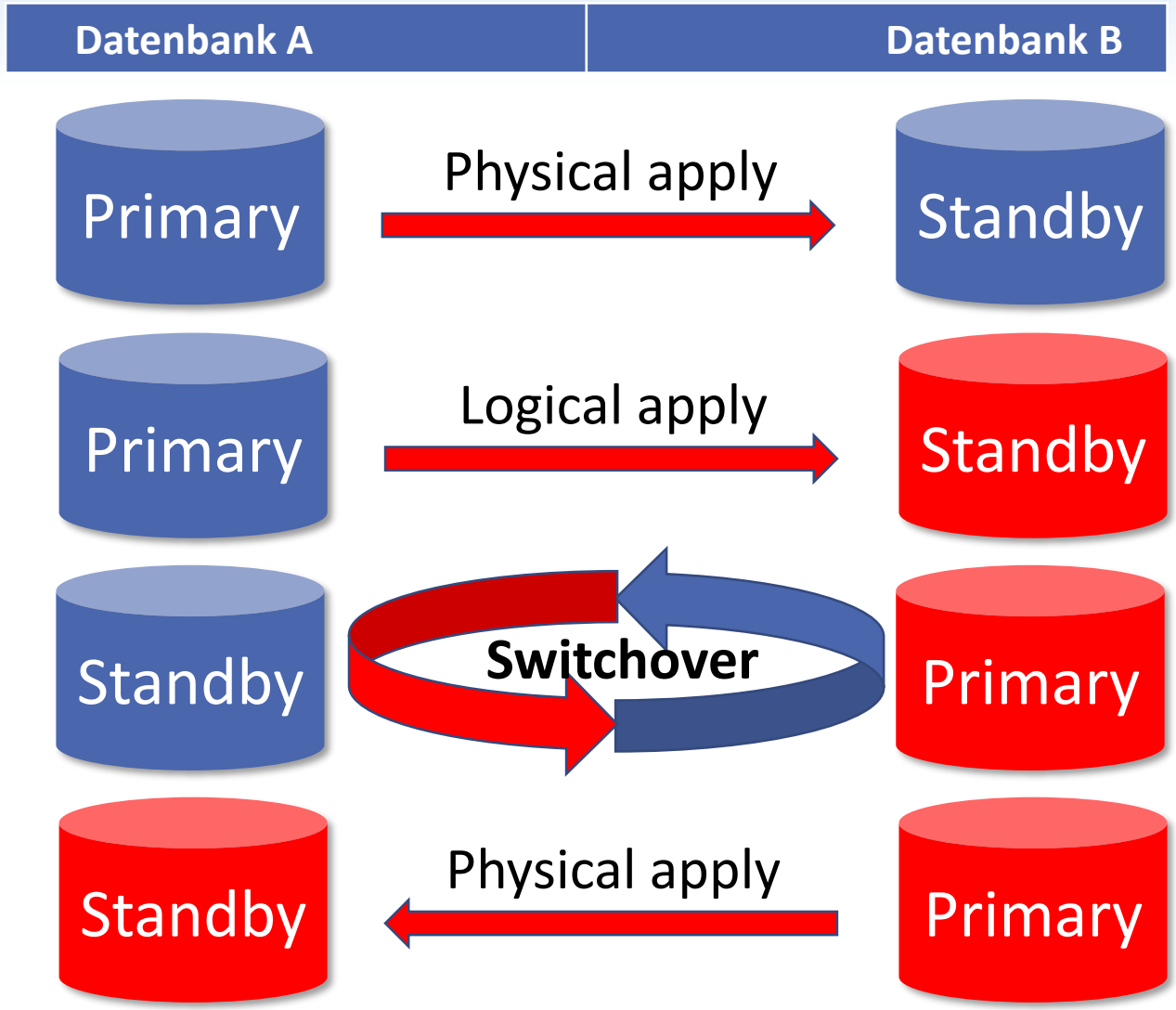
=rel. n =rel. n+1

Ausgangskonfiguration

Umwandlung in Transient Logical Standby und **Upgrade** der Standby

Rollentausch und Patchen der ehemaligen Primary Datenbank

Umwandlung in Physical Standby Datenbank



Alle DBMS_ROLLING Prozeduren im Überblick

Prozedur	Beschreibung
INIT_PLAN	Legt die künftige Primary Datenbank fest und generiert Default-Werte
SET_PARAMETER	Modifiziert Parameter für die Durchführung des Plans
BUILD_PLAN	Validiert Plan-Parameter und generiert oder modifiziert den Plan
START_PLAN	Startet die ROLLING-Operation
SWITCHOVER	Führt den Rollentausch zwischen der Primary- und der Transient Logical Standby-Datenbank durch.
FINISH_PLAN	Schliesst die ROLLING-Operation ab
DESTROY_PLAN	Löscht den Plan und deallokiert alle Ressourcen
ROLLBACK_PLAN	Rollt den Plan komplett zurück. Das geht aber nur vor einem Switchover!

Alle DBMS_ROLLING Prozeduren im Überblick

Prozedur	Beschreibung
INIT_PLAN	Legt die künftige Primary Datenbank fest und generiert Default-Werte
SET_PARAMETER	Modifiziert Parameter für die Durchführung des Plans
BUILD_PLAN	Validiert Plan Parameter und generiert oder modifiziert den Plan
START_PLAN	Startet die ROLLING-Operation
SWITCHOVER	Führt den Rollentausch zwischen der Primary- und der Transient Logical Standby-Datenbank durch.
FINISH_PLAN	Schliesst die ROLLING-Operation ab
DESTROY_PLAN	Löscht den Plan und deallokiert alle Ressourcen
ROLLBACK_PLAN	Rollt den Plan komplett zurück. Das geht aber nur vor einem Switchover!

Spezifizierung

Kompilierung

Ausführung

DBMS_ROLLING.INIT_PLAN

Zunächst wird die zukünftige Primary-Datenbank (die jetzige Standby) festgelegt:

```
SQL> EXEC DBMS_ROLLING.INIT_PLAN(future_primary => 'stdby')
```

Hier wird die Eingabe des ***DB_UNIQUE_NAME*** Parameters erwartet.

Hinweis: Auf Gross-/Kleinschreibung achten !

DBMS_ROLLING.SET_PARAMETER

Modifiziert Parameter für die Durchführung des Plans

```
DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_WAIT', '1');

DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_TIME', '60');

DBMS_ROLLING.SET_PARAMETER (name=>'LOG_LEVEL',value=>'FULL');
```

DBMS_ROLLING.BUILD_PLAN

Generiert und validiert den Plan und modifiziert die Plan-Parameter

```
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
```

Übliche Laufdauer: ca. 1s

Die DBA_ROLLING_PARAMETERS View

Die Plan Parameter anzeigen lassen

```
SELECT SCOPE, NAME, CURVAL FROM DBA_ROLLING_PARAMETERS ORDER
BY SCOPE, NAME;
```

SCOPE	NAME	CURVAL
prod	INVOLVEMENT	FULL
prod	MEMBER	NONE
stdby	INVOLVEMENT	FULL
stdby	MEMBER	TRAILING
	ACTIVE_SESSIONS_TIMEOUT	3600
	ACTIVE_SESSIONS_WAIT	0
	BACKUP_CONTROLFILE	rolling_change_backup.f
	DGBROKER	1
	DICTIONARY_LOAD_TIMEOUT	3600
	DICTIONARY_LOAD_WAIT	0
	DICTIONARY_PLS_WAIT_INIT	300

Die DBA_ROLLING_PLAN View

Den Rolling Plan anzeigen lassen

```
SELECT instid, target, phase, description
FROM DBA_ROLLING_PLAN;
```

```
INSTID TARGET     PHASE        DESCRIPTION
------ ---------- ------------ ------------------------------------------------------
     1 prod       START        Notify Data Guard broker that DBMS_ROLLING has started
     2 stdby      START        Notify Data Guard broker that DBMS_ROLLING has started
     3 prod       START        Verify database is a primary
     4 prod       START        Verify MAXIMUM PROTECTION is disabled
     5 stdby      START        Verify database is a physical standby
     6 stdby      START        Verify physical standby is mounted
     7 stdby      START        Verify future primary is configured with standby redo logs
     8 prod       START        Verify server parameter file exists and is modifiable
     9 stdby      START        Verify server parameter file exists and is modifiable
   ...
   ...
   ...
    83 prod       FINISH       Notify Data Guard broker that DBMS_ROLLING has finished
    84 stdby      FINISH       Notify Data Guard broker that DBMS_ROLLING has finished
    85 stdby      FINISH       Restore Supplemental Logging

85 rows selected.
```

DBMS_ROLLING.BUILD_PLAN

Fehlerverhalten

```
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
```

```
BEGIN DBMS_ROLLING.BUILD_PLAN; END;
*
ERROR at line 1:
ORA-45438: database is not in mounted mode
ORA-06512: at "SYS.DBMS_ROLLING", line 16
ORA-06512: at line 1
```

Fehlerbeseitigung auf der Standby Seite:

```
DGMGRL> edit database 'stdby' set state = 'apply-off';
SQL> shutdown immediate
SQL> startup mount
```

Danach ließ sich BUILD_PLAN problemlos erneut starten

DBMS_ROLLING.BUILD_PLAN

Fehlerverhalten

INSTID	TARGET	PHASE	DESCRIPTION
1	prod	START	Notify Data Guard broker that DBMS_ROLLING has started
2	stdby	START	Notify Data Guard broker that DBMS_ROLLING has started
3	prod	START	Verify database is a primary
4	prod	START	Verify MAXIMUM PROTECTION is disabled
5	stdby	START	Verify database is a physical standby
6	stdby	START	Verify physical standby is mounted
7	stdby	START	Verify future primary is configured with standby redo logs
8	prod	START	Verify server parameter file exists and is modifiable
9	stdby	START	Verify server parameter file exists and is modifiable
10	prod	START	Verify Data Guard broker configuartion is enabled
11	stdby	START	Verify Data Guard broker configuartion is enabled
12	prod	START	Verify Fast-Start Failover is disabled
13	stdby	START	Verify Fast-Start Failover is disabled
14	prod	START	Verify fast recovery area is configured

DBMS_ROLLING.START_PLAN

Fehlerbehandlung und Restart-Fähigkeit

```
SQL> EXEC DBMS_ROLLING.START_PLAN;
 ORA-45489: required condition was not
satisfied.
*
ERROR at line 1:
instruction execution failure1:
```

```
DGMGRL> stop observer
Observer stopped.
DGMGRL> disable fast_start failover;
Disabled.
```

```
SQL> EXEC DBMS_ROLLING.START_PLAN;
PL/SQL procedure successully completed.
```

Durchlaufzeit ca. 20 – 30s

Die DBA_ROLLING_EVENTS View

Wo werden Fehler protokolliert?

```
SELECT SYSDATE,TYPE "TYPE",MESSAGE from DBA_ROLLING_EVENTS
WHERE TYPE IN ('WARNING','ERROR','NOTICE')
```

```
TYPE         MESSAGE
------------ ------------------------------------------------------------
ERROR        DBMS_ROLLING.BUILD_PLAN halted due to error
ERROR        failed with status 45489 in instruction 12
ERROR        failure while executing one or more instructions from batch 7
ERROR        DBMS_ROLLING.START_PLAN failed due to error
NOTICE       database stdby must be upgraded, open read/write, and running the high
             er version binary before DBMS_ROLLING.SWITCHOVER can be invoked
```

DBA_ROLLING_PLAN:

```
11      stdby   START   Verify Data Guard broker configuartion is enabled
12      prod    START   Verify Fast-Start Failover is disabled
13      stdby   START   Verify Fast-Start Failover is disabled
14      prod    START   Verify fast recovery area is configured
```

DBMS_ROLLING Events im Alert-File

```
RTS(14212): completed instruction 10 from plan revision 1
2018-10-30T14:01:35.683441+01:00
RTS(14212): executing rolling upgrade instruction 12 from plan revision 1
2018-10-30T14:01:35.683512+01:00
RTS(14212): failed on instruction 12 from plan revision 1
Dumping error context:
2018-10-30T14:09:32.811776+01:00
Fast-Start Failover (FSFO) has been disabled on "prod"
2018-10-30T14:11:18.262229+01:00
RTS(14212): executing rolling upgrade instruction 12 from plan revision 1
2018-10-30T14:11:18.262340+01:00
RTS(14212): completed instruction 12 from plan revision 1
2018-10-30T14:11:18.280436+01:00
RTS(14212): executing rolling upgrade instruction 14 from plan revision 1
2018-10-30T14:11:18.288517+01:00
```

DBMS_ROLLING.START_PLAN

Data Guard Broker – Transient Logical Standby

```
DGMGRL> show configuration

Configuration - prodstdby

  Protection Mode: MaxPerformance
  Members:
  prod  - Primary database
    stdby - Transient logical standby database

Fast-Start Failover: DISABLED

Configuration Status:
    ROLLING DATABASE MAINTENANCE IN PROGRESS
```

DBMS_ROLLING.START_PLAN

Broker Warnung ´- Transient Logical Standby

```
DGMGRL> show database stdby

Database - stdby

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-OFF
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: (unknown)
  Real Time Query:    OFF
  Instance(s):
    stdby

  Database Warning(s):
    ORA-16866: database converted to transient logical standby database for rolling database maintenance

Database Status:
WARNING
```

Patching- und Testphase

Nach erfolgtem DBMS_ROLLING.START_PLAN

Wenn das DBMS_ROLLING.START_PLAN beendet ist, ist der Zeitpunkt für das Patching der Standby-Seite gekommen.
Beispiel:

```
$opatch prereq …
$opatch apply
$datapatch
```

Nach dem Patching

Bevor DBMS_ROLLING.SWITCHOVER aufgerufen wird

- Die Änderungen auf der Primärseite müssen auf die *'Transient Logical Standby*' Datenbank angewendet werden. Die Datenbank muss zum Schreiben geöffnet sein.
- Für Multitenant Datenbanken gilt:
  - Die TNS Services, mit denen man den Datenbank-Connect macht, müssen alle auf den **Root-Container** der Datenbank zeigen
  - Alle Container Datenbanken (Root und PDBs) müssen vor Aufruf von DBMS_ROLLING.SWITCHOVER Read/Write geöffnet sein, damit die Logical Apply Engine beim Anwenden der Änderungen in den PDBs nicht stehenbleibt

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

DBMS_ROLLING.SWITCHOVER

Fehlerbehandlung und Restart-Fähigkeit

```
SQL> EXEC   DBMS_ROLLING.SWITCHOVER;
```

Falls das Switchover probiert wird, ohne dass ein Logical Apply erfolgt ist:

```
BEGIN dbms_rolling.switchover; END;

*
ERROR at line 1:
ORA-45427: logical standby Redo Apply process was not running
ORA-06512: at "SYS.DBMS_ROLLING", line 89
ORA-06512: at line 1
```

Auch hier reicht es, den Logical Apply nachzuholen und das Switchover zu wiederholen

Auch wenn der DG Broker aktiv ist, muss ein 'Switchover' mit der DBMS_ROLLING.SWITCHOVER Prozedur durchgeführt werden! (*gilt auch noch für 18c*)

Nach dem DBMS_ROLLING.SWITCHOVER

Patching- und Testphase

Jetzt kann die alte Primary/neue Standby gepatcht werden
Beispiel:

```
$opatch prereq …
$opatch apply
$datapatch
```

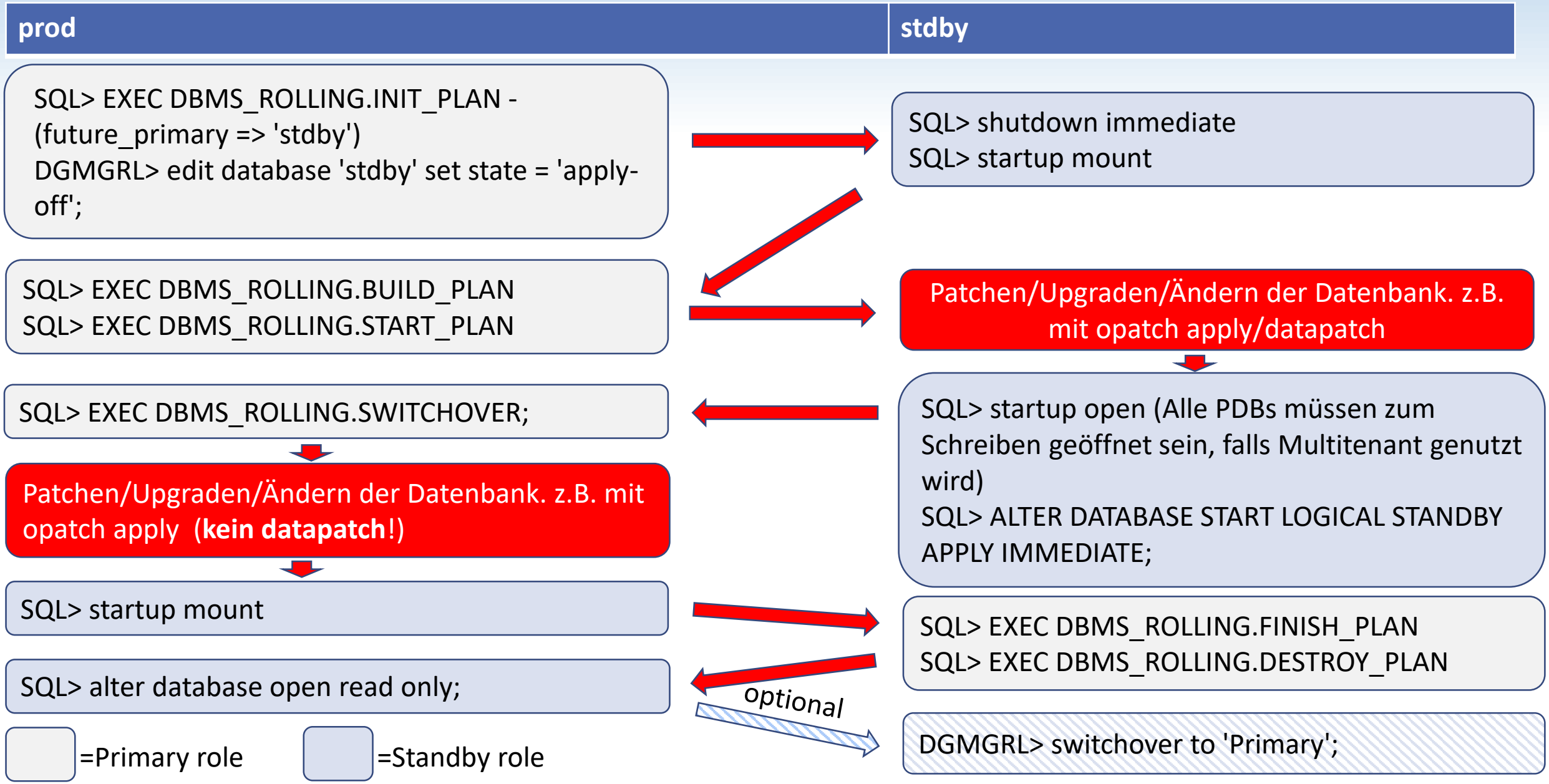
DBMS_ROLLING.FINISH_PLAN

Auf ehemaliger Standby/neuer Primary aufrufen

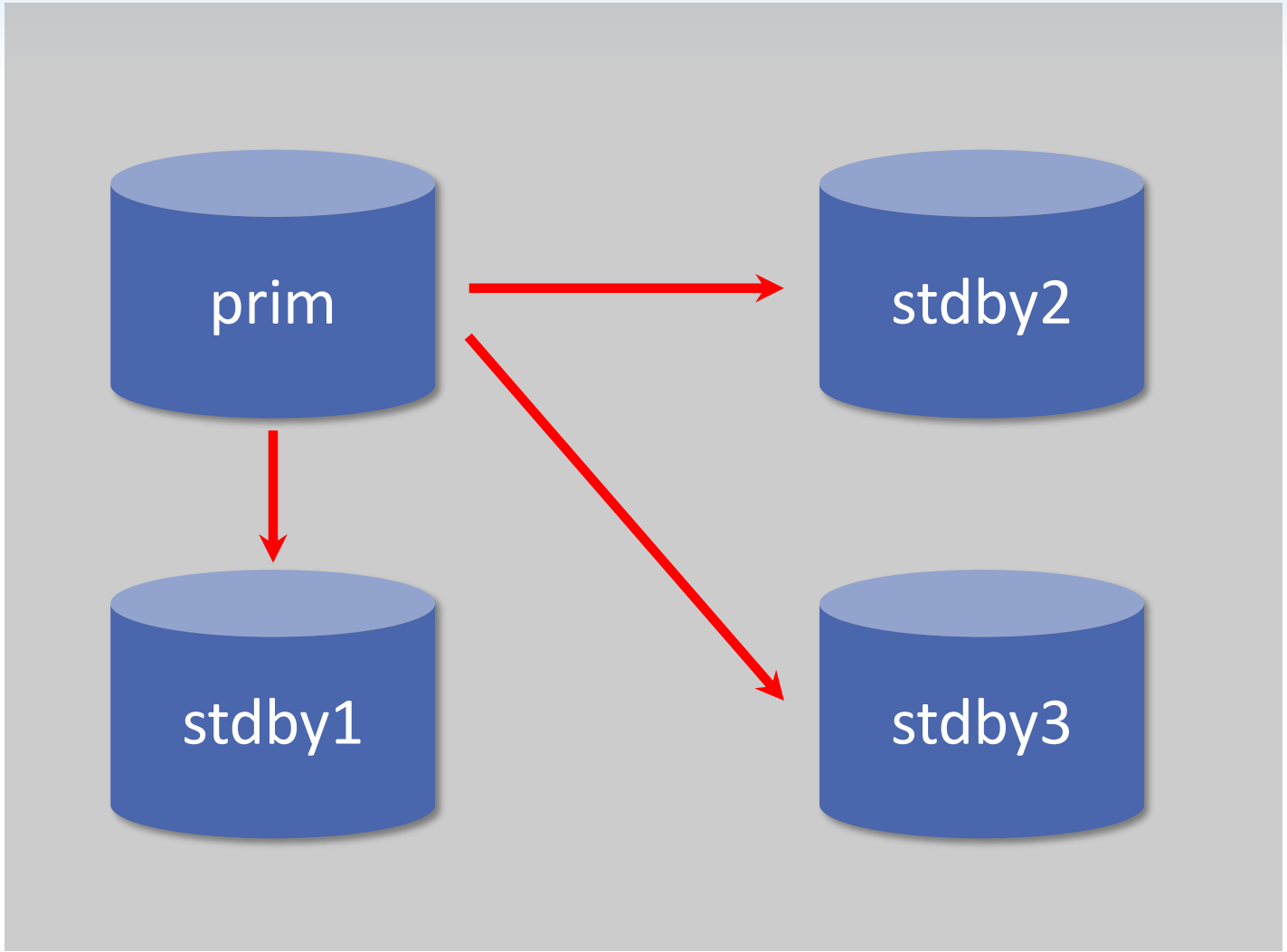
```
SQL> EXEC DBMS_ROLLING.FINISH_PLAN
BEGIN dbms_rolling.finish_plan; END;


*
ERROR at line 1:
ORA-45438: database is not in mounted mode
ORA-06512: at "SYS.DBMS_ROLLING", line 36
ORA-06512: at line 1
```

Nach Starten der ehemaligen Primary in den Mount Status lief auch diese Prozedur fehlerfrei durch

DBMS_ROLLING Ablauf-Zusammenfassung

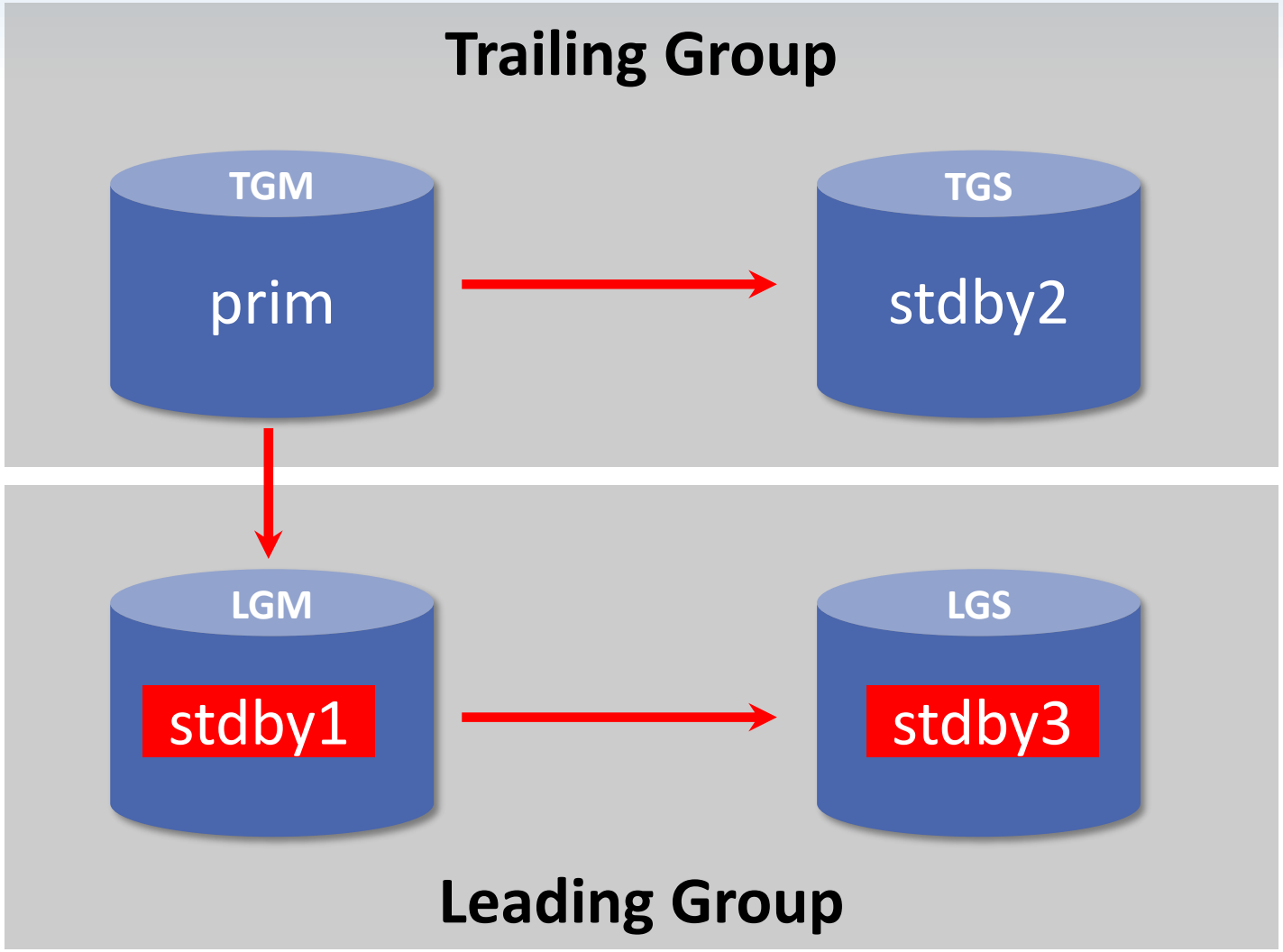


DBMS_ROLLING – Absicherung während des Patchings



Wenn sowohl Primär- als auch Standby-Datenbank während des Patchings abgesichert werden sollen, müssen mehrere Standby-Datenbanken konfiguriert werden

DBMS_ROLLING – Absicherung während des Patchings




```
DBMS_ROLLING.INIT_PLAN
(future_primary =>'stdby1')
```

```
DBMS_ROLLING.SET_PARAMETER(
scope=>'stdby3',name=>
'member' value=>'leading')
```

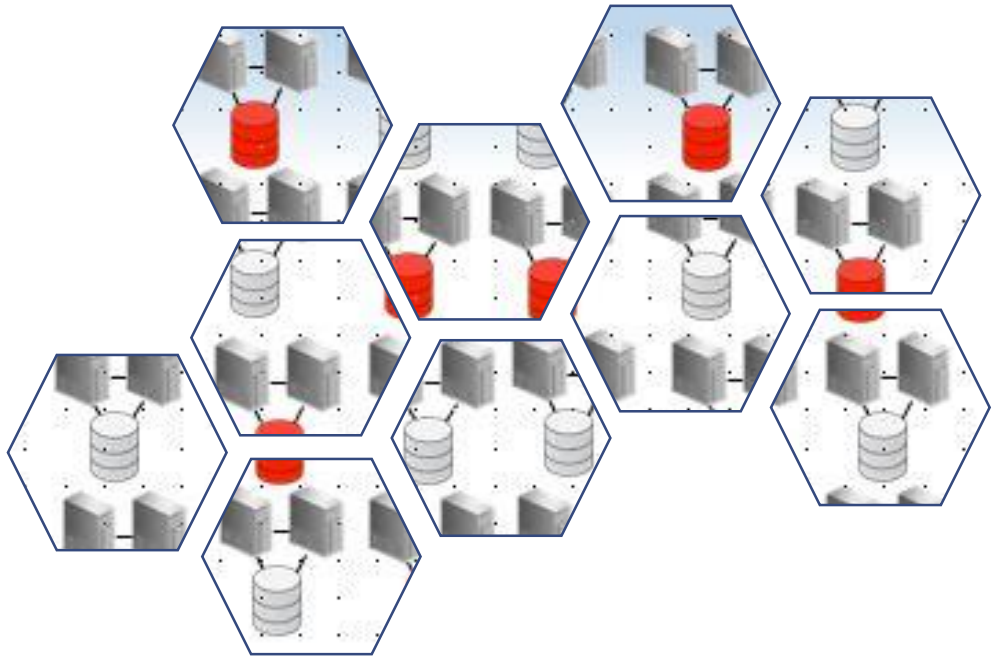
Abkürzungen:

TGM Trailing Group Master
TGS Trailing Group Standbys
LGM Leading Group Master
LGS Leading Group StandbysA

Zusammenfassung

- Das DBMS_ROLLING Package vereinfacht den komplexen Vorgang eines ROLLING Patchings erheblich
- Die Fehlermeldungen sind größtenteils eindeutig
- Eine durch einen Fehler unterbrochene DBMS_ROLLING Prozedur konnte in den von uns beobachteten Fällen nach Beseitigung der Ursache einfach erneut aufgerufen werden.
- Die Ausfallzeit für die User beschränkt sich auf die Zeit für das 'Switchover' und die Zeit für den Reconnect auf die neue Primary Datenbank.

Haben Sie noch Fragen?



Nikolaus Eichler
Selbständiger Berater
Am Nachbarsberg 18
D-42781 Haan

Telefon: +49 (0) 160-96227899
E-Mail nik.eichler@gmail.com