

### **Listing zum Monatstipp Mai 2009:**

An den folgenden Beispielen können Sie die Packages `vergleich_11g` und `vergleich_10g` testen.

Der Einfachheit halber darf nur ein DBA das Package verwenden, da nur der Ersteller eines Vergleichs die weiteren Prozeduren aufrufen und den Vergleich löschen kann (alternativ müsste man die Packages mit Invoker Rights erstellen und einige Erweiterungen einbauen).

Für die Beispielszenarios wurde das Package `vergleich_11g` in einer 11g-DB installiert, die Remote-DB war eine 10g Express-Edition.

In der Basis-Datenbank wird ein User mit DBA-Rechten eingerichtet, im Beispiel Kirk genannt, der in den Schemata Scotty und Spock (in der letzten Zeit kam zuviel Startrek im Fernsehen) je einen Klon von `dba_objects` erstellt

Für den Vergleich zwischen 2 Datenbanken über einen DB-Link wird noch ein weiterer Klon im Schema Scotty der Remote-DB angelegt.

### **Vorgehensweise für Oracle11g:**

Lassen Sie zuerst das Skript `vorbereitung_vergleich_11g.sql` laufen und installieren Sie dann das Package `vergleich_11g` über die Skripte `vergleich_11g_header.sql` und `vergleich_11g_body.sql` im Schema Kirk.

#### **1. Vergleich aller Spalten zweier Tabellen innerhalb eines Schemas**

```
conn kirk/uhura
set serveroutput on
BEGIN
    vergleich_11g.vergleichen(
        p_vergleichsname    => 'scotty_object1_object1a',
        p_schema_lokal      => 'scotty',
        p_basistabelle      => 'object1',
        p_vergleichstabelle => 'object1a');
END;
```

#### **Ausgabe:**

```
aktuelle Scan-Nr: 1
Die Objekte stimmen überein
```

#### **2. Vergleich aller Spalten zweier Tabellen in 2 verschiedenen Schemata.**

```
BEGIN
    vergleich_11g.vergleichen(
        p_vergleichsname    => 'scotty_object1_spock_object2',
        p_schema_lokal      => 'scotty',
        p_basistabelle      => 'object1',
        p_schema_remote     => 'spock',
        p_vergleichstabelle => 'object2');
END;
```

#### Ausgabe:

aktuelle Scan-Nr: 2  
Die Objekte stimmen überein

### 3. Vergleich von 2 Tabellen über einen Database Link unter Benutzung einer Spaltenliste

Da der Besitzer der Tabellen in beiden Fällen der gleiche ist (Scotty), muss der Name des Remote-Schemas nicht angegeben werden.

```
BEGIN
    vergleich_11g.vergleichen(
        p_vergleichsname => 'scotty_object1a_object1_rem',
        p_schema_lokal   => 'scotty',
        p_basistabelle   => 'object1a',
        p_vergleichstabelle => 'object1_rem',
        p_dblinkname     => 'sc_rem',
        p_spaltepk       => 'object_id',
        p_spalte1        => 'object_type',
        p_spalte2        => 'object_name',
        p_spalte3        => 'last_ddl_time');
END;
```

#### Ausgabe:

aktuelle Scan-Nr: 3  
Die Objekte stimmen überein

### 4. Vergleich zweier Tabellen in 2 verschiedenen Schemata unter Benutzung einer Spaltenliste

Jetzt bauen wir ein paar Unterschiede in die Tabelle scotty.object1 ein:

```
UPDATE scotty.object1
SET object_name = 'A', object_type = 'B', last_ddl_time = sysdate
WHERE MOD(object_id, 5000) = 0;
DELETE FROM scotty.object1 WHERE MOD(object_id, 4999) = 0;
DELETE FROM spock.object2 WHERE MOD(object_id, 3999) = 0;
INSERT INTO scotty.object1 (object_id, object_name) VALUES (90000, 'Mueller');
INSERT INTO scotty.object1 (object_id, object_name) VALUES (90001, 'Maier');
INSERT INTO spock.object2 (object_id, object_name) VALUES (90002, 'Lang');
INSERT INTO spock.object2 (object_id, object_name) VALUES (90003, 'Kurz');
COMMIT;
```

→

12 Zeilen wurden aktualisiert.  
12 Zeilen wurden gelöscht.  
15 Zeilen wurden gelöscht.  
1 Zeile wurde erstellt.  
1 Zeile wurde erstellt.  
1 Zeile wurde erstellt.  
1 Zeile wurde erstellt.

Wenn man nur eine Auswahl von Spalten angibt, kann man den Vergleich beschleunigen. Der Primärschlüssel muss immer dabei sein. Für das Beispiel werden die Spalten object\_id, object\_name, object\_type und last\_ddl\_time verwendet.

Diesmal sollen die differierenden Datensätze ermittelt und festgehalten werden, also wird der

Parameter `p_diff_bericht` auf `TRUE` gesetzt und die Details des Vergleichs mittels der Prozedur `vergleich_11g.diff_uebersicht` in der Tabelle `Ausgabe` im Schema `Kirk` festgehalten. Hier wird allerdings nur festgehalten, welche Zeilen in einer der beiden Tabellen fehlen, bzw. sich unterscheiden.

```
BEGIN
  vergleich_11g.vergleichen(
    p_vergleichsname => 'scotty_o1_spock_o2_4_spalten',
    p_schema_lokal   => 'scotty',
    p_basistabelle   => 'object1',
    p_schema_remote  => 'spock',
    p_vergleichstabelle => 'object2',
    p_spaltepk       => 'object_id',
    p_spalte1        => 'object_type',
    p_spalte2        => 'object_name',
    p_spalte3        => 'last_ddl_time',
    p_diff_bericht  => TRUE);
END;
```

### Ausgabe

```
aktuelle Scan-Nr: 4
Es wurden 43 Unterschiede gefunden.
12 Zeilen haben unterschiedliche Werte
14 Zeilen fehlen in der Basistabelle
17 Zeilen fehlen in der Vergleichstabelle
```

```
SELECT index_wert, in_basis, in_vergleich FROM ausgabe;
```

→

INDEX_WERT	IN_BASIS	IN_VERGLEICH	
90001	ja	nein	-- Datensatz nur in Basistabelle vorhanden
90000	ja	nein	
90002	nein	ja	-- Datensatz nur in Vergleichstabelle vorhanden
90003	nein	ja	
3999	ja	nein	
5000	ja	ja	-- in beiden Tabellen vorhanden, Werte unterschiedlich
4999	nein	ja	
10000	ja	ja	
...			

Die Tabelle `Ausgabe` wird bei jeder neuen Durchführung der Prozedur `diff_uebersicht` geleert. Wenn Sie das nicht wünschen, kommentieren Sie in Zeile 132 des Package Bodys den Befehl `EXECUTE IMMEDIATE 'TRUNCATE TABLE ausgabe'; aus`

### 5. Synchronisation der Tabellen nach dem Vergleich

Unter Angabe des Vergleichnamens und der beim Vergleich ausgegebenen `Scan_id` (der sogenannten `parent_scan_id`) kann man 2 Tabellen synchronisieren, wenn der Vergleich Unterschiede hervorbrachte. Die `parent_scan_id` kann man übrigens auch über die View `user_comparison_scan` abfragen.

Mit `p_richtung = 1` wird die Vergleichstabelle überschrieben, `p_richtung = 2` überschreibt die Basistabelle.

```
BEGIN
  vergleich_1lg.synchronisieren(
    p_vergleichsname => 'scotty_o1_spock_o2_4_spalten',
    p_scan_id => 4,
    p_richtung => 1);
END;
```

**Ausgabe:**

```
29 Zeilen in der Vergleichstabelle überschrieben
14 Zeilen in der Vergleichstabelle gelöscht
```

[6. Vergleich von 2 Tabellen \(Spaltenliste\) über einen Database Link inklusive Auslesen der Unterschiede und Synchronisation.](#)

Man kann die Synchronisation auch direkt in der Prozedur `vergleichen` über den Parameter `p_sync` veranlassen.

```
BEGIN
  vergleich_1lg.vergleichen(
    p_vergleichsname      => 'scotty_object1_object1_rem',
    p_schema_lokal        => 'scotty',
    p_basistabelle        => 'object1',
    p_schema_remote       => 'scotty',
    p_vergleichstabelle   => 'object1_rem',
    p_dblinkname          => 'sc_rem',
    p_spaltepk            => 'object_id',
    p_spalte1             => 'object_type',
    p_spalte2             => 'object_name',
    p_spalte3             => 'last_ddl_time',
    p_diff_bericht        => TRUE,
    p_sync                => 1);
END;
```

**Ausgabe:**

```
aktuelle Scan-Nr: 32
Es wurden 26 Unterschiede gefunden.
12 Zeilen haben unterschiedliche Werte
12 Zeilen fehlen in der Basistabelle
2 Zeilen fehlen in der Vergleichstabelle
14 Zeilen in der Vergleichstabelle überschrieben
12 Zeilen in der Vergleichstabelle gelöscht
```

## **Vorgehensweise für Oracle10g:**

Für die Beispielszenarios wurde das Package `vergleich_10g` auf einer 10g-DB installiert, die Remote-DB war ebenfalls eine 10g-DB auf einem anderen Rechner.

DBMS\_COMPARISON nutzt für den Vergleich der Tabellen die Ora\_Hash-Funktion, die seit Oracle10g implementiert ist. Darauf basiert auch das Package `vergleich_10g`.

Lassen Sie zuerst das Skript `vorbereitung_vergleich_10g.sql` (mit angepassten Connect-Strings) laufen und Installieren Sie dann das Package `vergleich_10g` über die Skripte `vergleich_10g_header.sql` und `vergleich_10g_body.sql` im Schema Kirk.

### **1. Vergleich aller Spalten zweier Tabellen innerhalb eines Schemas**

Rufen Sie zuerst die Prozedur `vergleich_10g.hash_vergl` auf, die die Hash-Werte der beiden Tabellen (für alle oder bis zu 4 Spalten) miteinander vergleicht. (Dafür wird zuerst über die private Funktion `vergleich_10g.spalten` eine konkatenierte Spaltenliste für jede der Tabellen erzeugt, über die die - ebenfalls private - Funktion `vergleich_10g.hash_sum` den Hash-Wert ermittelt).

```
SET TIMING ON
SET SERVEROUTPUT ON
BEGIN
  vergleich_10g.vergl_hash(
    p_schema_lokal => 'scotty',
    p_basistabelle => 'object1',
    p_vergleichstabelle => 'object1a');
END;
```

#### **Ausgabe:**

```
Hash-Wert der Basistabelle scotty.object1: 304028237432843
Hash-Wert der Vergleichstabelle scotty.object1a: 304028237432843
Keine Unterschiede gefunden
```

### **2. Vergleich aller Spalten zweier Tabellen in 2 verschiedenen Schemata.**

```
BEGIN
  vergleich_10g.vergl_hash(
    p_schema_lokal => 'scotty',
    p_basistabelle => 'object1',
    p_schema_remote => 'spock',
    p_vergleichstabelle => 'object2');
END;
```

#### **Ausgabe:**

```
Hash-Wert der Basistabelle scotty.object1: 304028237432843
Hash-Wert der Vergleichstabelle spock.object2: 304028237432843
Keine Unterschiede gefunden
```

### 3. Vergleich zweier Tabellen in 2 verschiedenen Schemata unter Benutzung einer Spaltenliste

Jetzt bauen wir wieder ein paar Unterschiede in die Tabelle scotty.object1 ein:

```
UPDATE scotty.object1
SET object_name = 'A', object_type = 'B', last_ddl_time = sysdate
WHERE MOD(object_id, 5000) = 0;
DELETE FROM scotty.object1 WHERE MOD(object_id, 4999) = 0;
DELETE FROM spock.object2 WHERE MOD(object_id, 3999) = 0;
INSERT INTO scotty.object1 (object_id, object_name) VALUES (90000, 'Mueller');
INSERT INTO scotty.object1 (object_id, object_name) VALUES (90001, 'Maier');
INSERT INTO spock.object2 (object_id, object_name) VALUES (90002, 'Lang');
INSERT INTO spock.object2 (object_id, object_name) VALUES (90003, 'Kurz');
COMMIT;
```

und vergleichen scotty.object1 und spock.object2 noch einmal

```
BEGIN
  vergleich_10g.vergl_hash(
    p_schema_lokal => 'scotty',
    p_basistabelle => 'object1',
    p_schema_remote => 'spock',
    p_vergleichstabelle => 'object2');
END;
```

#### **Ausgabe:**

```
Hash-Wert der Basistabelle scotty.object1: 303978466770671
Hash-Wert der Vergleichstabelle spock.object2: 303961876549947
Die Tabellen stimmen nicht überein
```

Wenn sich die Tabellen unterscheiden, kann man über die Prozedur

vergleich\_10g.vergl\_diff die Differenzen für den Vergleich von bis zu 4 Spalten in eine dynamisch erstellte Tabelle diff eintragen lassen (Für die Bearbeitung aller Spalten wäre der Einsatz von DBMS\_SQL nötig. Das folgt in einer späteren Version).

Per Default steht der Parameter p\_sync der Prozedur auf FALSE, d.h. es wird anschließend keine Synchronisation angestoßen.

Der dynamisch erstellte SQL-Befehl basiert auf dem Ansatz von Tom Kyte et al., (s.

Literaturstelle 2 im Monatstipp).

Er lautet in diesem Beispiel:

```
INSERT INTO diff
SELECT object_id,object_type,object_name,last_ddl_time,
CASE COUNT(src1) WHEN 0 THEN 'fehlt' ELSE 'vorhanden' END in_basis,
CASE COUNT(src2) WHEN 0 THEN 'fehlt' ELSE 'vorhanden' END in_vergleich
FROM
(SELECT object_id,object_type,object_name,last_ddl_time,
1 src1, TO_NUMBER(null) src2
FROM scotty.object1
UNION ALL
SELECT object_id,object_type,object_name,last_ddl_time,
TO_NUMBER(null) src1, 2 src2
FROM spock.object2)
```

```
GROUP BY object_id,object_type,object_name,last_ddl_time
HAVING COUNT(src1) <> COUNT(src2) ORDER BY 1
```

```
BEGIN
  vergleich_10g.vergl_diff (
    p_schema_lokal => 'scotty',
    p_basistabelle => 'object1',
    p_schema_remote => 'spock',
    p_vergleichstabelle => 'object2',
    p_spaltepk => 'object_id',
    p_spalte1 => 'object_type',
    p_spalte2 => 'object_name',
    p_spalte3 => 'last_ddl_time');
END;
```

### Ausgabe:

Es wurden 64 abweichende Datensätze gefunden und in die Tabelle Diff eingetragen

### Einträge in der Tabelle:

<u>OBJECT ID</u>	<u>OBJECT TYPE</u>	<u>OBJECT NAME</u>	<u>LAST DDL TIME</u>	<u>IN BASIS</u>	<u>IN VERGLEICH</u>
3999	VIEW	EXU8USR	15.10.07	vorhanden	fehlt
4999	VIEW	DBA_SNAPSHOTS	03.03.09	fehlt	vorhanden
5000	B	A	06.05.09	vorhanden	fehlt
5000	SYNONYM	DBA_SNAPSHOTS	15.10.07	fehlt	vorhanden
7998	PACKAGE	DBMS_APPCTX	15.10.07	vorhanden	fehlt
9998	SYNONYM	DBA_ADVISOR_SQLW_PARAM..	15.10.07	fehlt	vorhanden
10000	B	A	06.05.09	vorhanden	fehlt
10000	SYNONYM	USER_ADVISOR_SQLW_PARAM..	15.10.07	fehlt	vorhanden
11997	INDEX	WM\$RIC_TABLE_PT_IDX	15.10.07	vorhanden	fehlt
....					

### rot markiert:

Im Gegensatz zu den Einträgen in der Tabelle Ausgabe (Package vergleich\_11g) kommen die Index-Werte in der Tabelle diff im Fall der in Scotty.Object1 aktualisierten Datensätze doppelt vor, da die Datensätze in beiden Tabellen vorhanden sind.

### blau markiert:

Diese Datensätze müssen bei der Synchronisation in die Vergleichstabelle insertiert werden.

### grün markiert:

Diese Datensätze müssen bei der Synchronisation aus der Vergleichstabelle gelöscht werden.

## 4. Synchronisation der Tabellen in verschiedenen Schemata der gleichen Datenbank

Anschließend an den Vergleich kann man über die Prozedur vergleich\_10g.sync

- die angegebenen Spalten der Vergleichstabelle auf die entsprechenden Werte in der Basistabelle setzen,
- die nur in der Vergleichstabelle vorhandenen löschen und
- die im Vergleich zu Basistabelle fehlenden Datensätze insertieren.

Die Prozedur bearbeitet nur die Datensätze, die in der Tabelle Diff eingetragen wurden, deshalb geht das Ganze sehr schnell über die Bühne.

**Es darf zwischenzeitlich kein neuer Vergleich mit anderen Tabellen durchgeführt worden sein!!!**

Alternativ kann man den Parameter p\_sync beim Aufruf der Prozedur `vergleich_10g.vergl_diff` auf TRUE setzen (der Default-Wert ist FALSE), dann wird der Synchronisationsprozeß automatisch veranlasst, s.u.)

```
BEGIN
  vergleich_10g.sync (
    p_schema_lokal => 'scotty',
    p_basistabelle => 'object1',
    p_schema_remote => 'spock',
    p_vergleichstabelle => 'object2',
    p_spaltepk => 'object_id',
    p_spalte1 => 'object_type',
    p_spalte2 => 'object_name',
    p_spalte3 => 'last_ddl_time');
END;
```

Im Beispiel werden folgende DML-Befehle durchgeführt:

- **Update:** Der fett gedruckte Select filtert die Datensätze aus der Tabelle diff, deren Indexwerte doppelt vorkommen.

```
MERGE INTO spock.object2 o USING diff d ON (o.object_id = d.object_id)
WHEN MATCHED THEN UPDATE SET o.object_type = d.object_type,
o.object_name = d.object_name, o.last_ddl_time = d.last_ddl_time
WHERE o.object_id IN (SELECT object_id FROM diff
GROUP BY object_id
HAVING COUNT(object_id) =2)
AND d.in_basis = 'vorhanden';
```

- **Delete:**

```
DELETE FROM spock.object2
WHERE object_id IN
-- Indexwerte in der Tabelle diff, die nur in der Vergleichstabelle vorkommen
(SELECT object_id FROM diff WHERE in_vergleich = 'vorhanden'
MINUS
-- Indexwerte, die in Diff doppelt vorkommen
SELECT object_id FROM diff GROUP BY object_id HAVING COUNT(object_id) =2)
```

- **Insert:**

```
INSERT INTO spock.object2
SELECT * FROM scotty.object1
WHERE object_id IN
-- Indexwerte in der Tabelle diff, die nur in der Basistabelle vorkommen
(SELECT object_id FROM diff WHERE in_vergleich = 'fehlt')
```

```
MINUS
```

```
-- Indexwerte, die in Diff doppelt vorkommen
```

```
SELECT object_id FROM diff GROUP BY object_id HAVING COUNT(object_id) =2)
```

### 5. Vergleich und Synchronisation von 2 Tabellen über einen DB-Link unter Benutzung einer Spaltenliste

In diesem Beispiel wird die Synchronisation gleich an den Vergleich angeschlossen.

Da der Name des Schemas in der lokalen und der Remote-Datenbank der gleiche ist (Scotty), muss er nicht als Parameter übergeben werden.

Der Update über einen Merge macht bei Verwendung eines DB-Links Probleme (ORA-02064: Verteilter Vorgang nicht unterstützt), deshalb muss man in diesem Fall - wiederum basierend auf den Einträgen in der Diff-Tabelle - pauschal alle überflüssigen und älteren Datensätze der Vergleichstabelle löschen und dann die fehlenden aus der Basistabelle ergänzen.

```
BEGIN
```

```
    vergleich_10g.vergl_diff (  
        p_schema_lokal => 'scotty',  
        p_basistabelle => 'object1',  
p_vergleichstabelle => 'object1_rem',  
        p_link => 'sc_rem',  
        p_spaltepk => 'object_id',  
        p_spalte1 => 'object_type',  
        p_spalte2 => 'object_name',  
        p_spalte3 => 'last_ddl_time',  
        p_sync => TRUE);
```

```
END;
```

#### **Ausgabe:**

Es wurden 44 abweichende Datensätze gefunden und in die Tabelle Diff eingetragen.

Synchronisation gestartet

Synchronisation abgeschlossen