Kleiner Helfer: Mit

"Smart Replication"

Daten effizient, robust und flexibel verteilen

Vortragsunterlagen

Installationsskript (Sql*Plus)
Überwachungsaufruf (Sql)
Anwender-Dokumentation (html/zip)

DOAG-Konferenz 2009
17. bis 19. November 2009
Nürnberg
http://www.doag.org

Carsten Kaftan
Sanacorp Pharmahandel GmbH
Teamleiter Datenbankadministration
Semmelweisstraße 4
D-82152 Planegg
C.Kaftan@sanacorp.de

Version 1.0
– 29.10.2009 –

1 Überblick

Bei ca. 70 über Deutschland verteilten, produktiven Datenbanken sind Replikation und Datenverteilung für uns ein wichtiges Thema. Advanced Replication und Streams bieten nicht für alle Anforderungen die notwendige Unterstützung und sind durch einige oftmals unnötige Eigenschaften aufwendig.

Die Eigenentwicklung "Smart Replication", ein in PL/SQL geschriebenes Tool, konzentriert sich auf die stabile und anpassungsfähige Weitergabe von Datenänderungen; mit elementaren Verfahren werden diese erfaßt und in die gewünschten Ziele übertragen.

Der Vortrag richtet sich an Konferenzteilnehmer die eine schlichte, schnell und allgemein einsetzbare Lösung für Replikationsaufgaben suchen. Die nötigen Skripte zur Installation und Einrichtung von Smart Replication sowie die Anwender-Dokumentation sind diesem pdf-Dokument als Anlagen beigefügt.

2 Prinzip

Eine ***Smart Replication*** bezieht sich auf eine Quelltabelle (bzw. einen Quellview) und ggf. mehrere gleichartige Zieltabellen; es lassen sich beliebig viele voneinander unabhängige Smart Replications einrichten die mit einfachen Methoden wie Zwischen- und temporären Tabellen, Triggern und Jobs die Daten aus der Quelle heraus auf die gewünschten Ziele verteilen – innerhalb einer oder zwischen verschiedenen Datenbanken. Im folgenden soll der Aufbau einer Smart Replication dargestellt werden (vgl. Abb. 1).

Zunächst müssen die Quelldaten so wie im Ziel benötigt bereitgestellt werden; im einfachsten Fall sind Zieltabelle und Quelltabelle von gleicher Struktur, in anderen Fällen kann etwa ein passender View auf die Quelltabellen eingerichtet werden. Die Konfiguration der Smart Replication legt Quell- und Ziel-Datenobjekte, zugehörige Schlüsselfelder (i.a. den Primärschlüssel) und Übertragungsparameter fest (***Quelle***, ***Ziel*** und

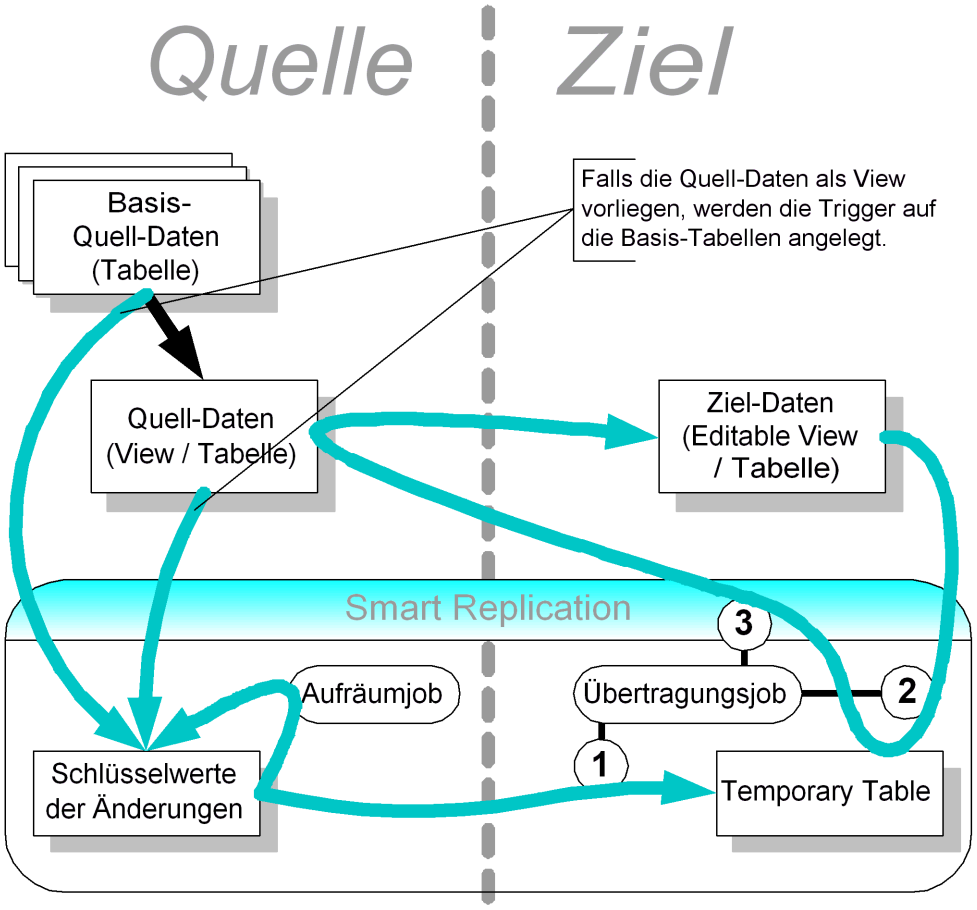

Abb. 1: Schematischer Datenfluß bei Smart Replication

Transfer). Bei Änderung eines Quelldatensatzes wird der zugehörige Schlüssel in die Schlüsselwerte-der-Änderungen-Tabelle eingefügt. Die Erfassung der Datenänderungen erfolgt am einfachsten über Trigger (***Auslöser***) auf die Quelltabelle bzw. auf die Basistabellen zum Quellview, es ist aber auch möglich die zu übertragenden Schlüssel gezielt in die Smart-Replication-Tabelle einzutragen. Beim Eintrag werden neben dem Schlüssel auch der vorgesehene Übertragungszeitraum und die Priorität festgelegt.

Ein ***Aufräumjob*** soll gewährleisten daß sich keine überflüssigen Einträge ansammeln. So werden gegebenenfalls Einträge zusammengefaßt, etwa nach Mehrfach-Änderungen eines Datensatzes bzw. Überschneidungen in den vorgesehenen Übertragungsfenstern. Gemäß der jeweiligen Konfiguration werden auch "Ewigkeits-Einträge" entfernt, deren Übertragungsfenster sehr weit in der Zukunft liegen. Ewigkeits-Einträge können bei Historisierungen entstehen, falls etwa ein Datensatz mit Gültigkeit bis zum 31.12.9999 ("für immer") geändert wird und die zugehörige Smart Replication so konfiguriert ist, daß sie für Gültig-Von- und Gültig-Bis-Datümer Aktualisierungen vormerkt.

Der ***Übertragungsjob*** wird bei Änderungen in der Schlüsselwerte-der-Änderungen-Tabelle aktualisiert und plant sich für den Beginn des nächsten Übertragungsfensters und danach solange in den vorgegebenen Zeitabständen ein, bis alle anliegenden Änderungen in der konfigurierten Blockgröße abgearbeitet sind. Danach plant sich der Übertragungsjob für den Beginn des nächstfolgenden Übertragungsfensters bzw. – falls keine weiteren Datenänderungen vorliegen – für "nie" ein. Die von diesem Job aufgerufene Datenübertragung läuft in folgenden Schritten ab (vgl. Abb. 1):

(1) Der nächste Datensatzblock wird nach Priorität und vorgesehenem Übertragungsfenster mit der konfigurierten Anzahl an Einträgen zusammengestellt und in die Temporary-Table im Ziel übertragen.

(2) Einträge mit Schlüsseln aus der Temporary-Table werden aus der Zieltabelle gelöscht.

(3) Einträge mit Schlüsseln aus der Temporary-Table werden von der Quelltabelle in die Zieltabelle kopiert.

Das Verfahren ***Smart Replication*** ist im Package ***DBA_REPLIC*** kodiert, hier finden sich die Prozeduren zur Einrichtung, Administration und Entfernung von Smart Replications. Beim Erstellen der ersten Smart Replication werden einige nötige Hilfsstrukturen – Views, Functions, Jobs, usw. – erstellt die beim Entfernen der letzten Smart Replication auch wieder entfernt werden.

3 Handhabung

Im folgenden soll beispielhaft dargestellt werden, wie Smart Replication eingerichtet und benutzt werden kann. Es soll die zeitnahe Übertragung aller Datenänderungen aus einer Quelltabelle in eine gleichstrukturierte und gleichnamige Zieltabelle im gleichen Schema auf einer anderen Datenbank aufgebaut werden. (Die durchzuführenden Schritte sind mit schattiertem, grauen Hintergrund dargestellt.)

Ausgefeiltere Konfigurationsmöglichkeiten werden in diesem Kapitel nicht berücksichtigt, diese werden in der gezippt beigefügten Anwender-Dokumentation ***Smart Replication - Handbuch.html*** erläutert.

3.1 Installation

Mit dem Package DBA_REPLIC werden Hilfsstrukturen im Schema REPLIC installiert, Smart Replications eingerichtet und konfiguriert, usw. DBA_REPLIC ist bzgl. der fachlichen Logik eigenständig, verwendet aber Routinen aus einem "DBA_%-Framework" – etwa bezüglich Logdatei-Schreibung und Fehlerbehandlung –, die zunächst bereitgestellt werden müssen. Der gesamte DBA_%-Code wird im Schema ADMIN eingerichtet.

Das beigefügte Installationsskript ***SmartReplication_Installation.sql*** (vgl. Kapitel 3.1.2) erstellt das ADMIN-Schema mit einem reduzierten DBA_%-Framework, das DBA_REPLIC-Package sowie den User REPLIC. Nach Ausführung dieses Skriptes können Smart Replications eingerichtet werden.

Ebenfalls als Anlage steht auch der Überwachungsaufruf ***SmartReplication_Monitoring.sql*** (vgl. Kapitel 3.1.3) zur Verfügung, der den Gesamtzustand aller eingerichteten Smart Replications überprüft.

3.1.1 Hinweise

- Smart Replication wird auf 9i- und 10g-Datenbanken eingesetzt und enthält keine besonderen Versionsabhängigkeiten, die Verwendung mit 11g wurde bisher aber nicht getestet.
  Für 10g-Datenbanken gilt allerdings: Smart Replication arbeitet in einigen Situationen sehr unperformant wenn abhängige Cursor nach einem Analyze nicht invalidiert werden. Die datenbankweite Einstellung

  ```
  DBMS_STATS.SET_PARAM('NO_INVALIDATE','FALSE');
  ```

  oder eine entsprechende Anpassung der Prozedur ***DBA_TOOL.AnalyzeOneTable*** behebt das Problem.
- Für das Logging wird mit Bezug auf das Dateisystem (entsprechend der Installationsvariable ***PATH_PROT***) ein Datenbank-Directory ***PROT*** eingerichtet. In dieses Verzeichnis wird die Datei ***SmartReplication.log*** geschrieben.
- Das Logging ist sehr ausführlich, alle Übertragungen werden summarisch in der Logdatei protokolliert. Falls die Bewirtschaftung der Log-Dateien nicht ausreichend effizient ist oder auf eine ausführliche Protokollierung kein Wert gelegt wird läßt sich der Code diesbezüglich anpassen – detaillierte Auswertungen zum Übertragungsvolumen usw. (Abfragen auf ***ALERT_REPLIC_LOG***) können dann aber nicht mehr oder nur noch eingeschränkt durchgeführt werden.
- Das Leserecht auf das Directory PROT, das Ausführungsrecht auf DBA_REPLIC und verschiedene Rechte auf REPLIC-Objekte werden an ***PUBLIC*** gegrantet.
- Bei schemaübergreifenden Replikationen benötigt der Ziel-Owner das ***SELECT ANY TABLE***-Recht um auf die Quelltabelle und Hilfstabellen im Quellschema zugreifen zu können (ein schemabezogenes SELECT-Recht bietet Oracle nicht an; tabellenbezogene Grants können auf die erst zu erstellenden Hilfstabellen nicht im Voraus vergeben werden).
- Das DBA_%-Framework-Package ***DBA_TOOL*** enthält eine Hilfsprozedur ***CallExecuteImmediateInSchema***, die Operationen in allen Schemas mit Owner-Berechtigungen erlaubt. Falls das zu riskant ist muß diese Prozedur eingeschränkt werden (etwa auf Ausführungen im REPLIC-Schema).

- Für jede Smart Replication werden zwei Jobs erzeugt; bei zahlreichen Replikationen entstehen entsprechend viele Datenbank-Jobs.
- Um Probleme mit der USER-Zuordnung bei Datenbank-Jobs zu umgehen, werden von einigen Prozeduren Updates auf die Systemtabelle ***DBA_JOBS*** ausgeführt (entsprechend den vorgeschlagenen Verfahren z.B. aus den Metalink-Notes 228516.1 – "How to copy (export/import) the Portal database schemas of IAS 9.0.2 to another database", 399230.1 – "Troubleshooting Chapter 11 Staging a Test Environment from a Production Environment", usw.).
- Durch die Trigger-Abhängigkeiten invalidieren eventuelle Code-Änderungen bei Smart Replication ggf. fachliche Objekte, es können in dieser Situation einmalig Fehlermeldungen auftreten (***ORA-04068: existing state of packages has been discarded*** oder ***ORA-04061: existing state of ... has been invalidated***). Änderungen am Code von DBA_REPLIC und den Triggern sollten also während Wartungsintervallen erfolgen und anschließend die invaliden Objekte compiliert werden. Reine Konfigurationsänderungen sind jederzeit problemlos möglich.
- Die angelegten Hilfsobjekte beziehen sich teilweise auf Schema-Tabellen bzw. Views und werden durch Anhängen eines Suffixes an den Objektnamen benannt: Tabellen und Views die in Smart Replication verwendet werden, dürfen – ohne Anpassungen in DBA_REPLIC – keine Namen von mehr als 22 Zeichen Länge aufweisen.

3.1.2 SmartReplication_Installation.sql

Das Installationsskript ***SmartReplication_Installation.sql*** ist in diesem pdf-Dokument als Anlage enthalten. Bitte legen Sie vor Ausführung des Skripts geeignete Installationsparameter fest (Zeilen 30-36, siehe unten). Das Skript muß als SYS bzw. mit SYSDBA-Berechtigungen ausgeführt werden, da einige Grants auf SYS-Objekte vergeben werden und ein Directory angelegt wird. Das Installationsskript muß auf jeder an einer Smart Replication beteiligten Datenbank ausgeführt werden, also Quell- und Zieldatenbanken[1].

```
Aufruf des Installationsskripts mit geeigneten Parametern

-- Installationsparameter:
set define on;
-- Die Tablespace für die ADMIN-Daten- und Indexsegmente:
define TABLESPACE_DTA     = 'USERS'
define TABLESPACE_NDX     = 'USERS'
-- Die Paßwörter für die Benutzer ADMIN und REPLIC:
define PASSWORD_ADMIN     = 'Lege mich fest!'
define PASSWORD_REPLIC    = 'Lege mich fest!'
-- Das Protokollverzeichnis (auf Dateiebene):
define PATH_PROT          = '/prot/oracle/'
```

3.1.3 SmartReplication_Monitoring.sql

Ebenfalls enthalten ist ***SmartReplication_Monitoring.sql***; dieser Sql-Aufruf überprüft die eingerichteten Smart Replications auf potentielle Probleme; gegebenenfalls werden Fehlermeldungen ausgegeben (die in der Anwender-Dokumentation erklärt werden). Eine Überwachung entsprechend den Sql-Abfragen in diesem Skript – das hauptsächlich als Muster diesen soll – sollte für jede Datenbank mit Smart Replications eingerichtet werden.

```
Bei einem produktiven Einsatz von Smart Replication sollten
entsprechende Überprüfungen regelmäßig ausgeführt werden.
```

1) Selbstverständlich kann eine Datenbank Quell- und Zieldatenbank zugleich sein.

3.2 Vorbereitung einer Smart Replication

Die Quell- und Zieldatenbanken müssen einander erreichen können, deshalb werden insgesamt sechs Datenbank-Links benötigt (die Anlage dieser Datenbanklinks ist nur einmal pro Datenbankpaar notwendig; Aufruf mit entsprechenden Angaben in der Quell- und in der Zieldatenbank, als SYS, ADMIN bzw. REPLIC):

```
-- als SYS:
create public database link &&ZIEL-/QUELL-DATENBANK
       using '&&ZIEL-/QUELL-DATENBANK';

-- als ADMIN
create database link &&ZIEL-/QUELL-DATENBANK
       connect to ADMIN identified by &&PASSWORD_ADMIN
       using '&&ZIEL-/QUELL-DATENBANK';

-- als REPLIC
create database link &&ZIEL-/QUELL-DATENBANK
       connect to REPLIC identified by &&PASSWORD_REPLIC
       using '&&ZIEL-/QUELL-DATENBANK';
```

Der Tabellen-Owner benötigt einige Rechte (dieser Grant muß nur einmal pro Schema ausgeführt werden; Aufruf in der Quell- und in der Zieldatenbank):

```
grant CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER,
      CREATE DATABASE LINK, UNLIMITED TABLESPACE to &&SCHEMA;
```

3.3 Einrichtung einer Smart Replication

Die einmalig pro Datenbank, Datenbankpaar und Datenschema durchzuführenden Installationen und Vorbereitungen sind abgeschlossen, jetzt können die einzelnen Smart Replications eingerichtet werden.

Einrichtung des Ziels (Aufruf in der Zieldatenbank):

```
begin
  ADMIN.DBA_REPLIC.SmartTargetAdd('&&SCHEMA','&&TABELLE');
end;
/
```

Einrichtung der Quelle (dieser und die weiteren Aufrufe in der Quelldatenbank):

```
begin
  ADMIN.DBA_REPLIC.SmartSourceAdd('&&SCHEMA','&&TABELLE')
end;
/
```

Einrichtung des Transfers:

```
begin
  ADMIN.DBA_REPLIC.SmartTransferAdd('&&SCHEMA','&&TABELLE',
                                    '&&QUELLDATENBANK','&&ZIELDATENBANK');
end;
/
```

Erstellung des Skripts für die Auslöser-Trigger:

```
begin
  ADMIN.DBA_REPLIC.SmartScriptTrigger('&&SCHEMA','&&TABELLE');
end;
/
```

Durch die Erstellung des Skripts sind die Trigger noch nicht angelegt worden, dazu muß das generierte Skript in einem weiteren Schritt ausgeführt werden. Dies gibt die Möglichkeit zur Kontrolle und Anpassung – das Anlegen dieser Trigger ist der entscheidende Punkt bei der Einrichtung; in diesem Schritt werden die Datenstrukturen mit der Replikation verbunden. Nach diesem Schritt werden alle Datenänderungen repliziert:

```
Ausführung des im vorhergehenden Schritt erzeugten Skripts.
```

3.4 Status-Abfragen

Mit Abfragen auf ***SmartReport***, ***SmartStatusFct***, ***SMARTSTATUS*** und ***ALERT_REPLIC_LOG*** lassen sich die Informationen zu Status und Konfiguration der Smart Replications erhalten.

Umfassende Status- und Konfigurations-Abfrage (Aufruf in der Quell- oder in der Zieldatenbank):

```
set echo off heading off feedback off
set linesize 200 pagesize 0
select TEXT_CONTENT from table(ADMIN.DBA_REPLIC.SmartReport('%'));
commit;
```

Schnelle Abfrage auf eventuelle Probleme (dieser und alle weiteren Aufrufe in den Quelldatenbanken):

```
select   SR_OWNER, SR_OBJECT,
         SR_OVERDUE, SR_CURRENT,
         trunc(case when SR_SCHEDULED_OVERDUE-SR_SCHEDULED_FIRST >= .9/24
                         and SR_SCHEDULED_FIRST < SR_DATEREF
                    then 100*(SR_DATEREF-SR_SCHEDULED_FIRST)/
                              (SR_SCHEDULED_OVERDUE-SR_SCHEDULED_FIRST)
                    else NULL end) PROZENT,
         case when nvl(SR_JOB_NEXTDATE,to_date('9999','yyyy')) >
                   greatest(SR_SCHEDULED_FIRST, SR_DATEREF + 5/1440)
              then 'JOB_NEXTDATE > SCHEDULED_FIRST'
              else 'ok'
              end NEXTDATE,
         SR_SCHEDULED_FIRST, SR_SCHEDULED_OVERDUE,
         SR_JOB_NEXTDATE, SR_JOB_FAILURES, SR_JOB_BROKEN
    from table (REPLIC.SmartStatusFct('%', '%', null, 1))
   where nvl(SR_SCHEDULED_FIRST, SYSDATE) < SYSDATE - 5/1440
      or nvl(SR_JOB_FAILURES, 0) > 0
      or nvl(SR_JOB_BROKEN, 'N') != 'N'
      or nvl(SR_JOB_NEXTDATE,to_date('9999','yyyy')) >
         greatest(SR_SCHEDULED_FIRST, SR_DATEREF + 5/1440)
order by 1, 2;
```

Besonders geeignet für datenbankübergreifende Abfragen (hier lassen sich Functional Tables wie SmartStatusFct nicht direkt verwenden):

```
select SR_OWNER "Schema", count(1) "Anzahl der Quellen" from (
select   *
    from REPLIC.SMARTSTATUS@DATENBANK01
   where SR_SOURCE_TYPE = 'LOCAL'
union all
select   *
    from REPLIC.SMARTSTATUS@DATENBANK02
   where SR_SOURCE_TYPE = 'LOCAL')
group by SR_OWNER
order by SR_OWNER;
```

Analyse der Logdatei nach "Übertragungsvolumen pro Stunde" :

```
with info as (select OBJEKT, ZEIT, SEKUNDEN, ZEILEN from (
  select VON ZEIT, (BIS-VON)*24*60*60 SEKUNDEN, OBJEKT, ZEILEN from (
    select OBJEKT, ZEIT VON,
           lead(zeit, 1) over (partition by OBJEKT
                               order by ZEIT, ISTSTART desc, ISTENDE) BIS,
           ZEILEN, IstStart, IstEnde, TEXT from (
      select ZEIT, OBJEKT, TEXT,
             case when TEXT like '%will now be transferred%'
                  then 1 else 0 end IstStart,
             case when TEXT like '%arget(s) completed.%'
                  then 1 else 0 end IstEnde,
             case when TEXT like '%will now be transferred%'
                  then to_number(substr(TEXT, 1, instr(TEXT, ' ')))
             else NULL end ZEILEN from (
        select to_date(substr(TEXT, 1, 20), 'dd.mm.yyyy, hh24:mi:ss') ZEIT,
               substr(TEXT, 24, instr(TEXT, ', SmartExecute')-24) OBJEKT,
               substr(TEXT, instr(TEXT, ', SmartExecute: ')+
                            length(', SmartExecute: ')) TEXT
          from REPLIC.ALERT_REPLIC_LOG
         where TEXT like '%SmartExecute%'))
       where IstStart = 1 or IstEnde = 1 )
     where IstStart = 1
     order by OBJEKT, VON))
select OBJEKT, trunc(ZEIT, 'HH'), sum(Zeilen) ZEILEN,
       sum(Sekunden) SEKUNDEN, count(1) AUFRUFE
  from info
 group by OBJEKT, trunc(ZEIT, 'HH')
 order by 1, 2;
```

3.5 Troubleshooting

Falls ein Ziel vorübergehend – etwa aufgrund eines Netzwerkproblems – nicht mehr erreichbar ist läßt sich der betroffene Transfer deaktivieren, die anderen Ziele werden normal weiterversorgt. Die Änderungen die der deaktivierte Transfer nicht übertragen konnte werden später nach Reaktivierung nachgefahren.

Deaktivierung eines Transfers (dieser und der nächste Aufruf in den Quelldatenbanken):

```
begin
  ADMIN.DBA_REPLIC.SmartTransferSetStatus(
    '&&SCHEMA', '&&TABELLE', '&&Ziel', '%', '%', 0);
end;
/
```

Aktivieren des (aller) deaktivierten Transfers und automatisches Nachfahren der angefallenen Änderungen:

```
begin
  ADMIN.DBA_REPLIC.SmartTransferSetStatus('%', '%', '%', '%', '%', 1);
end;
/
```

4 Beispiel – Musterdurchlauf

Eine Datenbank DEMO.world enthalte in der Tabelle STAMM.HIST historisierte Stammdaten. Diese sollen zum jeweils aktuellen Stand im Schema ZACK in der Tabelle AKT zur Verfügung gestellt werden.

Anlegen der Beispieltabellen:

```
create user STAMM identified by STAMM;
grant connect, resource to STAMM;

create table STAMM.HIST (
  NAME    varchar2(300),
  VON     date,
  BIS     date,
  EINTRAG varchar2(4000) );

ALTER TABLE STAMM.HIST ADD (
  CONSTRAINT HIST_PK PRIMARY KEY (NAME, VON) );


create user ZACK identified by ZACK;
grant connect, resource to ZACK;

create table ZACK.AKT (
  NAME    varchar2(300),
  EINTRAG varchar2(4000) );

ALTER TABLE ZACK.AKT ADD (
  CONSTRAINT AKT_PK PRIMARY KEY (NAME) );
```

Füllen mit Werten:

```
insert into STAMM.HIST ... ;
```

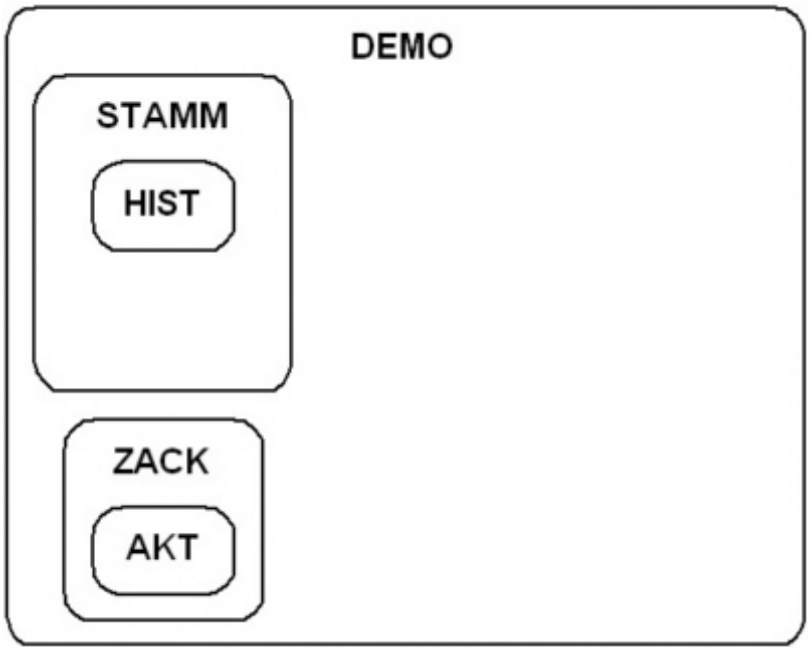

Abb. 2: Ausgangssituation in der DEMO-Datenbank

Installieren von Smart Replication:

```
Aufruf des Installationsskripts mit folgenden Parametern:

-- Installationsparameter:
set define on;
-- Die Tablespace für die ADMIN-Daten- und Indexsegmente:
define TABLESPACE_DTA     = 'USERS'
define TABLESPACE_NDX     = 'USERS'
-- Die Paßwörter für die Benutzer ADMIN und REPLIC:
define PASSWORD_ADMIN     = 'ADMIN'
define PASSWORD_REPLIC    = 'REPLIC'
-- Das Protokollverzeichnis (auf Dateiebene):
define PATH_PROT          = '/dbtmp/'
```

Bereitstellen der Quelldaten in geeigneter Form:

```
create view STAMM.HIST_AKT as
     select NAME, EINTRAG
       from HIST
      where sysdate between VON and BIS;
```

Vergabe der notwendigen Berechtigungen:

```
grant CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER,
      CREATE DATABASE LINK, UNLIMITED TABLESPACE to STAMM;
grant CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER,
      CREATE DATABASE LINK, UNLIMITED TABLESPACE to ZACK;
grant SELECT ANY TABLE to ZACK;
```

Einrichten der Senke:

```
exec ADMIN.DBA_REPLIC.SmartTargetAdd('ZACK', 'AKT');
```

```
22.10.2009, 14:37:47 - SmartSetup: Table "REPLIC.SMART_OBJECTS" created.
22.10.2009, 14:37:47 - SmartSetup: Table "REPLIC.SMART_TRANSFERS" created.
22.10.2009, 14:37:47 - SmartSetup: Table "REPLIC.ALERT_REPLIC_LOG" created.
22.10.2009, 14:37:47 - SmartSetup: Procedure "REPLIC.JOB_SETNEXTDATE" created.
22.10.2009, 14:37:47 - SmartSetup: Type "REPLIC.SmartStatusRec" created.
22.10.2009, 14:37:47 - SmartSetup: Type "REPLIC.SmartStatusTbl" created.
22.10.2009, 14:37:47 - SmartSetup: Function "REPLIC.SmartStatusFct" created.
22.10.2009, 14:37:47 - SmartSetup: View "REPLIC.SMARTSTATUS" created.
22.10.2009, 14:37:47 - SmartSetup: View "REPLIC.SMARTDEPENDENCIES" created.
22.10.2009, 14:37:47 - SmartSetup: Temporary Table "REPLIC.SMARTDEPENDENCIES#TMP" created.
22.10.2009, 14:37:48 - ZACK.AKT, SmartTargetAdd: Configuration-Entry created.
22.10.2009, 14:37:48 - ZACK.AKT, SmartTargetAdd: Table "ZACK.AKT#REP_TMP" created.
22.10.2009, 14:37:48 - ZACK.AKT, SmartTargetAdd: Procedure "ZACK.AKT#REP_PRC" created.
22.10.2009, 14:37:48 - ZACK.AKT, SmartTargetAdd: Configuration-Entry updated.
```

Einrichten der Quelle:

```
exec ADMIN.DBA_REPLIC.SmartSourceAdd('STAMM', 'HIST_AKT', 'NAME');
```

```
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Configuration-Entry created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Sequence "STAMM.HIST_AKT#REP_SEQ" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Table "STAMM.HIST_AKT#REP" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Index "STAMM.HIST_AKT#REP_NX1" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Job "1" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Tidy-Job "3" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Trigger "STAMM.HIST_AKT#REP_TR1" created.
22.10.2009, 14:38:05 - STAMM.HIST_AKT, SmartSourceAdd: Trigger "STAMM.HIST_AKT#REP_TR2" created.
22.10.2009, 14:38:06 - STAMM.HIST_AKT, SmartSourceAdd: Configuration-Entry updated.
```

Einrichten des Transfers:

```
exec ADMIN.DBA_REPLIC.SmartTransferAdd(
       'STAMM', 'HIST_AKT', '', '', 'ZACK', 'AKT');
```

```
22.10.2009, 14:38:24 - STAMM.HIST_AKT, SmartTransferAdd: Transfer to ZACK.AKT added.
```

Erstellen des Auslöser-Trigger-Skripts:

```
begin
  ADMIN.DBA_REPLIC.SmartScriptTrigger
  ( sOwner => 'STAMM',
    sObject => 'HIST_AKT',
    sBaseTableOwner => 'STAMM',
    sBaseTableName => 'HIST',
    sBaseTableDateBegin => 'VON',
    sBaseTableDateEnd => 'BIS' ) ;
end;
/
```

```
-- Sql-Code für einen Trigger bezüglich "STAMM.HIST_AKT"
-- Vor Ausführung bitte prüfen und anpassen!
create trigger STAMM.HIST#REP_TR001
  after delete or insert or update
  on STAMM.HIST
  for each row
begin
  if 1 != ADMIN.DBA_REPLIC.SmartTriggerDoFire(iFireInReplication => 0) then return; end if;
  if deleting then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:old.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
  elsif inserting then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:new.NAME, greatest(:new.VON, SYSDATE), greatest(:new.VON, SYSDATE)+interval'1'hour,
                                   DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:new.NAME, greatest(:new.BIS, SYSDATE)+1/86400, greatest(:new.BIS, SYSDATE)
                                   +1/86400+interval'1'hour, DEFAULT);
  elsif updating then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:old.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:new.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:new.NAME, greatest(:new.VON, SYSDATE), greatest(:new.VON, SYSDATE)+interval'1'hour,
                                   DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN, REP#TARGET_WINDOW_END,
                                   REP#PRIORITY)
    values (:new.NAME, greatest(:new.BIS, SYSDATE)+1/86400, greatest(:new.BIS, SYSDATE)
                                   +1/86400+interval'1'hour, DEFAULT);
  end if;
end;
/
```

Ausführen des Auslöser-Trigger-Skripts:

```
create trigger STAMM.HIST#REP_TR001
  after delete or insert or update
  on STAMM.HIST
  for each row
begin
  if 1 != ADMIN.DBA_REPLIC.SmartTriggerDoFire(iFireInReplication => 0) then
                        return; end if;
  if deleting then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:old.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
  elsif inserting then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:new.NAME, greatest(:new.VON, SYSDATE), greatest(:new.VON,
                        SYSDATE)+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:new.NAME, greatest(:new.BIS, SYSDATE)+1/86400,
                        greatest(:new.BIS, SYSDATE)+1/86400+interval'1'hour,
                        DEFAULT);
  elsif updating then
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:old.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:new.NAME, SYSDATE, SYSDATE+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:new.NAME, greatest(:new.VON, SYSDATE), greatest(:new.VON,
                        SYSDATE)+interval'1'hour, DEFAULT);
    insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END, REP#PRIORITY)
    values (:new.NAME, greatest(:new.BIS, SYSDATE)+1/86400,
                        greatest(:new.BIS, SYSDATE)+1/86400+interval'1'hour,
                        DEFAULT);
  end if;
end;
/
```

Initialisieren der Zieltabelle:

```
insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END)
     select NAME, VON, VON+interval'1'hour
       from STAMM.HIST
      where VON > SYSDATE;

insert into STAMM.HIST_AKT#REP(NAME, REP#TARGET_WINDOW_BEGIN,
                        REP#TARGET_WINDOW_END)
     select NAME, BIS, BIS+interval'1'hour
       from STAMM.HIST
      where BIS > SYSDATE;

insert into ZACK.AKT
     select * from STAMM.HIST_AKT
     minus
     select * from ZACK.AKT;

commit;
```

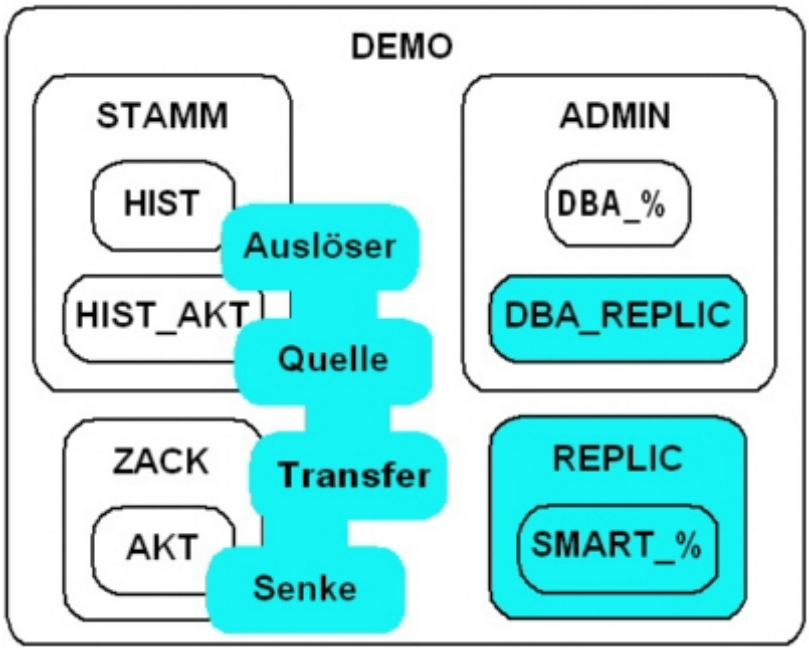

Abb. 3: Die Smart Replication ist fertig eingerichtet

Ausgabe des umfassenden Reports:

```
set echo off heading off feedback off
set linesize 200 pagesize 0
select TEXT_CONTENT from table(ADMIN.DBA_REPLIC.SmartReport('%'));
commit;
```

```
Smart-Replication-Konfiguration und Status
__________________________________________



Grundlegende Konfiguration und Status
=====================================

Datenbank-Name: "DEMO.WORLD"
Bezugszeit: 22.10.2009, 15:18:00
Die Smart-Replication-Umgebung ist vollständig eingerichtet.


Smart Replications
==================


Smart-Replication "STAMM.HIST_AKT"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Konfiguration
-------------
                      Schema: STAMM
                      Objekt: HIST_AKT
                  Schlüssel: NAME
                Einrichtung: Quelle

Datenfluß
---------
     STAMM.HIST >>>
 STAMM.HIST_AKT >>>
    >>> STAMM.HIST_AKT >>>
           >>> ZACK.AKT
```

```
Einträge in STAMM.HIST_AKT
--------------------------
           Bisher bearbeitet: 20.000
                  Überfällig:            0
         Vorliegend, aktuell:        4.180 im Zeitraum 22.10.2009, 14:45:10 - 22.10.2009, 16:18:00
          Vorliegend, gesamt:       15.697 im Zeitraum 22.10.2009, 14:45:10 - 22.10.2009, 17:06:50
             Davon ohne Redo:       15.697 im Zeitraum 22.10.2009, 14:45:10 - 22.10.2009, 17:06:50

Job-Status
----------
                       Job-ID: 1
                               Fehlerfreie Ausführung
                   Karenzzeit: 10 Sekunden
                   Begrenzung: maximal 1000 Einträge pro Aufruf
               Letzter Aufruf: 22.10.2009, 15:15:32
                Dieser Aufruf: 22.10.2009, 15:18:28
               Nächster Aufruf: 22.10.2009, 15:18:24

Aufräumen bei STAMM.HIST_AKT
----------------------------
             Insgesamt gelöscht: 13.042 Einträge
              Letzte Ausführung: 22.10.2009, 15:15:57
               Zuletzt gelöscht: 2.726 Einträge

Aufräumjob-Status
-----------------
                       Job-ID: 3
                               Fehlerfreie Ausführung
                   Karenzzeit: 300 Sekunden
                Idle-Abstand: 3600 Sekunden
                     Ewigkeit: ab "to_date('31123000', 'ddmmyyyy')", aktuell: 31.12.3000, 00:00:00
               Overlap-Analyse: maximal 100 Einträge pro Aufruf
                    Löschungen: maximal 10000 Einträge pro Aufruf
               Letzter Aufruf: 22.10.2009, 15:05:27
                Dieser Aufruf: 22.10.2009, 15:15:57
               Nächster Aufruf: 22.10.2009, 15:15:57

Senken für STAMM.HIST_AKT
-------------------------
ZACK.AKT
                       Status: Der Transfer ist aktiv.
               Datenbank-Links: - keine Verwendung von Datenbank-Links -


Smart-Replication "ZACK.AKT"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Konfiguration
-------------
                       Schema: ZACK
                       Objekt: AKT
                   Schlüssel: NAME
                  Einrichtung: Senke

Datenfluß
---------
 STAMM.HIST_AKT >>>
        ZACK.AKT >>>
          >>> ZACK.AKT

Quellen für ZACK.AKT
--------------------
STAMM.HIST_AKT
                       Status: Der Transfer ist aktiv.
               Datenbank-Links: - keine Verwendung von Datenbank-Links -
                    Überfällig: 0 Einträge
                       Aktuell: 4180 Einträge
                        Gesamt: 15697 Einträge im Zeitraum:
                                22.10.2009, 14:45:10 - 22.10.2009, 17:06:50
                           Job: 1
                                Fehlerfreie Ausführung
                                22.10.2009, 15:15:32 - letzter Aufruf
                                22.10.2009, 15:18:28 - dieser Aufruf
                                22.10.2009, 15:18:24 - nächster Aufruf
```

5 Eigenschaften

Die Schlüsselqualitäten von Smart Replication sind:

- Massenänderungen werden zügig abgearbeitet und führen nicht zu Ressourcenengpässen.
- Die Übertragung von Änderungen läßt sich priorisieren – etwa bei Online-Anwendungen gegenüber Batch-Abläufen – und zeitlich steuern.
- Einfache Konfiguration:
  Weitgehend automatische Einrichtung, jederzeitige Anpassung der Einstellungen, unkomplizierte Änderung von Datenstrukturen, problemlose Erweiterung etwa um neue Zieldatenbanken, usw.
- Sichere und flexible Administration:
  Durchgängige Kontrolle über den aktuellen Status, direkter Zugriff auf die Smart-Replication-Komponenten, im Bedarfsfall sind eine gezielte Drosselung der Übertragungsraten, die Unterbrechung und Umleitung von Datenübertragungen, usw. möglich.

Wesentliche fachliche Unterschiede zu Streams bzw. Advanced Replication:

- Die zu übertragenden Inhalte stehen nicht von vorneherein fest, sondern nur die den Datensatz identifizierenden Schlüsselwerte: die zum **Übertragungszeitpunkt** – nicht dem Änderungszeitpunkt – in der Quelle konkret vorhandenen Werte werden im Ziel eingefügt.
- Hohe "Fehlertoleranz": Beispielsweise gibt es keinen Feldinhaltsvergleich zwischen alten und neuen Einträgen der ggf. zu einer Fehlermeldung führt – der Inhalt zum Zeitpunkt der letzten Datensatz-Übertragung "gewinnt".
  Ein Kollisionshandling ist deshalb nicht ohne weiteres möglich; gegebenenfalls läßt sich der Übertragungsschlüssel geeignet erweitern – etwa unter Einbeziehung des letzten Datensatz-Update-Zeitpunktes – um diese Thematik abzudecken.
- Keine Transaktionstreue und keine Einzelsatzverarbeitung: Smart Replication überträgt in unabhängigen Transaktionen mehrere, ggf. geeignet zusammengefaßte Änderungen auf einmal.

Weitere Merkmale:

- Abweichend vom Default kann konfiguriert werden, daß von Smart Replication übertragene Änderungen in einer anschließenden Smart Replication weiter übertragen werden – damit sind Kettenübertragungen möglich. Der Standard – keine Fortpflanzung von Änderungen – erlaubt netzförmige Synchronisation ("Multi-Master"), dazu müssen alle Quellen mit allen Zielen über Smart Replications verbunden werden.
- Mit der Verwendung von (Editable) Views lassen sich zu denselben Tabellen verschiedene Smart Replications einrichten.
- Mit der ***SmartTruncate***-Prozedur können Quell- und Zieltabelle(n) truncated werden, der Truncate wird dabei sofort – also ohne Berücksichtigung eines Vorgabe-Zeitfensters – durchgeführt.
- Wird der Smart-Replication-Session-Status ***InReorganisation*** auf 1 gesetzt so werden nachfolgende Datenänderungen in dieser Session nicht übertragen – bis ***InReorganisation*** wieder auf 0 zurückgesetzt wird.
- Einträge in der Schlüsselwerte-der-Änderungen-Tabelle werden nur durch den Aufräumjob oder den Übertragungsjob entfernt. Falls das gewünschte Übertragungsfenster in der Zukunft liegt können sich also überflüssige Einträge ansammeln, z.B. wenn der Datensatz auf den sich ein Eintrag bezieht inzwischen gelöscht wurde. Für die korrekte Datenübertragung ist dies aber ohne Bedeutung.
- Änderungen werden zu den vorgesehenen Übertragungsfenstern sofort übertragen, es gibt kein Warten auf den Ablauf eines regelmäßigen Refresh-Intervalls oder ähnliches.

6 Verwendungsmöglichkeiten

Smart Replication kann dort eingesetzt werden, wo Datenänderungen aufbereitet und an anderer Stelle "eingearbeitet" werden sollen; z.B. in folgenden Szenarien:

- Bereitstellen aktueller Daten aus historisierten Tabellen:
  Häufig werden Daten mit Gültigkeitszeiträumen versehen. Zur Performance-Steigerung lassen sich mit Smart Replication die jeweils aktuellen Daten in einer Arbeitstabelle für schnelle Abfragen zur Verfügung stellen. Zu den in den Datensätzen angegebenen Gültig-Von- und Gültig-Bis-Zeitpunkten wird der jeweils zugehörige Eintrag in der Arbeitstabelle aktualisiert.
- Inkrementelles "ETL" in ein Datawarehouse:
  Das Übertragungsfenster der Datensätze wird auf den gewünschten (nächtlichen) Zeitpunkt festgelegt: Tagsüber stehen im Datawarehouse die Vortagesdaten konsistent zur Verfügung, nachts werden die Änderungen integriert.
- "Fast Refresh ohne Limits":
  Gibt es einen Zusammenhang zwischen Änderungen in Basistabellen und betroffenen Ergebnis-Datensätzen, kann dieser für eine zeilenweise Aktualisierung genutzt werden indem die zugehörigen Auslöser-Trigger entsprechend formuliert werden. Formale Restriktionen (bzgl. Model-Clause, Having, Sysdate, usw.) gibt es keine.
- Ressourcenschonende, zeitnahe Übertragung auch bei geringen Änderungsraten:
  Änderungen werden sofort erkannt und weitergeleitet; in Zeiträumen in denen keine Bearbeitungen stattfinden werden keine Ressourcen verbraucht – kein Polling, keine zeitabhängigen Refreshs, usw.
- "Multi-Master-Replikation" in eine zentrale Zieltabelle:
  Ein sternförmiger Datenfluß aus den Niederlassungen ins Zentrum zur dortigen Weiterbearbeitung läßt sich ohne Tricks (bei Advanced Replication müssen hier z.B. die zentralen Snapshot-Log-Tabellen truncated werden) umsetzen.
- ...

7 Einsatz in der Sanacorp

Am 30.01.2008 wurde der Entschluß gefaßt, ein allgemein einsetzbares, massendatenänderungstaugliches und ressourcenschonendes Replikations-Verfahren zu entwickeln, das Zeitsteuerung und Priorisierung ermöglicht. Bis zum 14.05.2008 war das Verfahren in der ersten Version fertig und wurde produktiv eingesetzt. Inzwischen wird – nach einigen Verbesserungen – Smart Replication auch für betriebswichtige Daten verwendet. Beispiele:

- Nächtliche Übertragung der Stammdaten(änderungen) aus der zentralen Datenbank ***DB*** in die Auswertungsdatenbank ***EBEN***.
- Aus der vielfach historisierten, weitestgehend normalisierten Stammdatenhaltung in der zentralen ***DB***-Datenbank werden die jeweils aktuellen Einträge ermittelt und aufbereitet in die Hochverfügbarkeitsdatenbank ***ZACK*** übertragen (z.B. Kundendaten für die Auftragserfassung).
- Ersatz aller Updateabler Snapshots durch Smart Replications – dadurch wird deutlich mehr Flexibilität erreicht, vor allem bei der anstehenden Einrichtung zweier neuer Niederlassungen.
- Viele Faktura-Tabellen mit kleinen Änderungsraten werden aus den Niederlassungen in die zentrale Datenbank repliziert.
- Stoßweise Änderungsprozesse – Einspielen von vollständigen Preislisten, usw. – werden von Smart Replication "gedrosselt" an die Bestimmungsorte übertragen; Ad-Hoc-Änderungen dürfen überholen.

Meßergebnisse zum Ressourcenverbrauch (die Messungen wurden vor der Einführung von Smart Replication für eine betriebswichtige Datenbank in der Testumgebung durchgeführt; die Übertragungen waren so konfiguriert daß keine spürbare Beeinträchtigung des Betriebes zu befürchten war):

- Bei normalem Replikationsvolumen steigt die CPU-Last der DB nur geringfügig (ca. 2 Prozentpunkte) an.
- Bei Massenänderungen steigt die CPU-Last der DB um bis zu 10 Prozentpunkte an. Es können ca. 100.000 Sätze je Minute repliziert werden (wenn 40.000 Sätze/Übertragung konfiguriert sind).

Aktueller Status:

963.739.325 übertragene Sätze (Übertragungsjob)

81.733.784 gelöschte Sätze (Aufräumjob)

660 Smart-Replication-Quellen

37 Datenbanken mit Smart-Replication

(produktive Datenbanken, 26.10.2009, 12:48)