ritconsult

Ing. Mag. Reinhard Reisinger
IT Consulting

Schillerstraße 24/18
A-4020 Linz, Austria
Tel. +43 (0)699/111 99 325
eMail: r.reisinger@ritconsult.at

Migration
TABSYS nach TABEX/4

Leitfaden

Inhaltsverzeichnis

Beispielübersicht

1 Einleitung

Nachfolgend eine Anleitung wie System- und Konfigurationstabellen aus TABSYS in TABEX/4 übernommen werden können.

Es wird ausdrücklich darauf hingewiesen, dass die Anleitung nur als Unterstützung für eine in Eigenregie und Eigenverantwortung durchzuführende Migration dienen soll. Weiters wird darauf hingewiesen, dass für die Beispiel SQL-Befehle und für die Beispiel TABEX/4 Utility Abläufe keine Garantie auf Korrektheit und Fehlerfreiheit gegeben werden kann.

Das Dokument ist in wesentlichen Bereichen eine Aufzeichnung der durchgeführten Tätigkeiten im Zuge der Ablöse von TABSYS auf DB2/LUW nach TABEX/4 3.2.0 (Linux) mit Daten in DB2/LUW. Wenn die Migration aus anderen Quellumgebungen erfolgen soll beziehungsweise in eine andere Zielumgebung, so sind die jeweiligen systemspezifischen Befehle anzupassen.

Die Migration wurde nach TABEX/4 3.2.0 durchgeführt. Version 3.2.0 ist die Mindestvoraussetzung für diese Migrationsanleitung. Höhere TABEX/4 Versionen sind entsprechend den aktuellen Datenstrukturen mit den richtigen Dateninhalten zu versorgen.

2 Voraussetzungen

Um die Migration anhand dieses Leitfadens durchführen zu können sind folgende Voraussetzungen zu erfüllen.

- umfassende Administrationskenntnisse in TABEX/4
- umfassende SQL Kenntnisse
- Verständnis der TABSYS Systemtabellen
- Kenntnisse in der jeweiligen Betriebssystem Skriptsprache (shellscript, windows batch, …)

3 Vorbereitende Arbeiten

3.1 Zugangs- und Zugriffsberechtigungen

Es werden für die Migration folgende Zugangs- und Zugriffsberechtigungen benötigt

- TABEX/4 Administrator
- DB2 Benutzer mit „create table“ Rechten
- Systembenutzer/Dienstbenutzer mit DB2 Profil auf die Zieldatenbank
- Systembenutzer mit Schreibrechten in TABEX/4 Utility-Verzeichnisse (../work/inp, ../work/jcl, …)

3.2 Generischer Utility-Aufruf

Zur Umstellung werden eine Reihe von TABEX Utility-Jobs benötigt. Um nicht für jeden TABEX Utility-Job eine eigene TABEX Utility Jobcontrol schreiben zu müssen kann ein generischer Utility-Aufruf konfiguriert werden. Parameter für diesen Aufruf ist dann nur mehr der jeweils unterschiedliche TABEX Utility Input (die Utility Befehlssequenz).

3.2.1 Verzeichnisstruktur für generische Utility-Aufrufe

Um über die Ergebnisse der benötigten Utility-Aufrufe einen möglichst guten Überblick zu bewahren wird die Anlage folgender Verzeichnisstruktur empfohlen.

```
…<BOIROOT>/<mywork>/jcl
                   /inp
                   /log
```

In /jcl werden die JobControlLanguage Steuersequenzen abgelegt
In /inp werden die TABEX/4 Utility-Befehlssequenzen abgelegt
In /log werden die Logdateien der Utility-Aufrufe abgelegt

3.2.2 Shell-Script für generischen Utility-Aufruf

Aufruf der BOI Laufzeitumgebung mit JobControl „util_standard“ und der Utility-Befehlssequenz laut Aufrufparameter.

```
Beispiel:
> exec_utl reorg_database
```

```
# ============================================
# call a TABEX Utility
# with generic input-file (linux)
# --------------------------------------------
# (c) ritconsult, Linz, Austria, 2010
# ============================================
# !/bin/sh

this_TABEX=$BOIROOT/work/ehkv_tabex
TMP_BIN=$BOIROOT/bin
TMP_JCL=$this_TABEX/jcl
TMP_LOG=$this_TABEX/log
TMP_INP=$this_TABEX/inp

THISUTL=TABN01
THISJCL=util_standard
THISSCR=$1

echo "$0 starting 390 emulation"
echo "$0 starting emulation in: $TMP_BIN"
echo "$0 using job control in : $TMP_JCL"
echo "$0 starting utility ....: $THISUTL"
echo "$0 using jobcontrol ....: $THISJCL"
echo "$0 running skript ......: $THISSCR"

rm -f $TMP_LOG/$THISSCR.log

$TMP_BIN/boirun $TMP_JCL/$THISJCL.jcl $4 $5 $6 $7 $8 $9
echo "... done"
echo ":-) $0 thanks 4 using tabex/4"
```

Beispiel 1: exec_util – shell script - generischer Utility-Aufruf

3.2.3 Jobcontrol für generischen Utility-Aufruf

```
/* ========================================================== */
/* TABEX/4 Job Control Language                               */
/* call a utility with dynamic command input file             */
/* ========================================================== */
/* (c) ritconsult, Linz, Austria, 2010                        */
/* ========================================================== */

/* THISSCR set in environment to TABEX-path                   */
DEFINE   MYSCR   <<THISSCR>>

INCLUDE "<TABEXJCL>/sysjcl.jcl"        /* system settings      */
INCLUDE "<TABEXJCL>/stdtdb.jcl"        /* standard databases   */
```

```
TERMTYPE = NONE                          /* no terminal          */
SYSSWI   = <SWI-BAT>                     /* systemswitch batch   */
MODULE   = TABUCOMP                      /* module               */
EXEMODE  = STANDALONE                    /* mode                 */
SYSPARM  = '<SYSPCPG>,X=TABN0X,U=TABN02B'

/* additional allocated databases                                */
FILE = TABVSAM '<TABEXTDB>/.tdb' <TDB>

/* system IO files                                               */
FILE = SYSIN     '<TabexWork>/ehkv_tabex/inp/<MYSCR>.inp'  <INP>
FILE = SYSPRINT  '<TabexWork>/ehkv_tabex/log/<MYSCR>.lst'  <LST>
FILE = BOIINP    '<TabexWork>/ehkv_tabex/inp/<MYSCR>.txt'  <INP>
FILE = BOIPRINT  '<TabexWork>/ehkv_tabex/log/<MYSCR>.txt'  <PRT>
```

Beispiel 2: util_standard - JCL - generischen Utility-Aufruf

3.3 TABEX Migrationsdatenbank

Die Migration wird für die TABEX relevanten Steuereinträge in einem zweistufigen Verfahren durchgeführt.
In Schritt eins werden die Daten aus den TABSYS Konfigurationstabellen mittel SQL Abfragen in TABEX Tabellen in der TABEX Migrationsdatenbank übernommen.
In Schritt zwei werden diese Steuereinträge für TABEX mit den bestehenden TABEX Steuereinträgen zusammengeführt.

3.3.1 Anlegen TABEX Migrationsdatenbank

Wenn für die Migration nicht eine bestehende TABEX Datenbank verwendet werden kann, dann ist mit den TABEX Utilities eine entsprechende TABEX „Migrationsdatenbank“ zu erzeugen.

```
- ============================================
- create a TABEX/4 database (Windows XP)
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN01
SWITCHUTL,TABN01
-
- set parameter 1 to database file
SETPAR,1
<path>/tdb/TABVSAM.tdb
-
alloc (and create) file dynamically
ALLOC,TABVSAM,%PAR01,"NOALIAS DIO  FIX 8185 NOCRLF NOLEN EBC UNB"
-
- activate allocated database
DB=TABVSAM
-
- initialize database with 20 records (this is very small!!!)
INIT,20,T3C,REUSE
-
- release database
FREE,TABVSAM
-
- end of create database
END
```

Beispiel 3: create_database - utility input - Anlage Datenbank

4 TABEX Tabellenkatalog

4.1 Anlegen eines TABEX Tabellenkatalogs

Der TABEX/4 Tabellenkatalog muss entsprechend den Vorgaben von BOI erstellt werden.
Der Creator/das Schema und der Tabellenname sind frei wählbar. Die Felddefinitionen und die Schlüsseldefinitionen müssen der Vorgaben von BOI entsprechen.

```
-- create a TABEX4 tablecatalog (DB2/LUW)
-- replace scheme, tablename and dbspace when needed
-- ========================================
-- (c) ritconsult, Linz, Austria, 2010
-- ========================================
CREATE TABLE "TABEX"."BOICAT"  (
                "OBJID" CHAR(10) NOT NULL ,
                "OTYPE" CHAR(8) NOT NULL ,
                "STMP_D" CHAR(26) NOT NULL ,
                "DATE_R" CHAR(10) NOT NULL ,
                "USER_A" CHAR(8) NOT NULL ,
                "USER_C" CHAR(8) NOT NULL ,
                "STMP_C" CHAR(26) NOT NULL ,
                "USER_M" CHAR(8) NOT NULL ,
                "STMP_M" CHAR(26) NOT NULL ,
                "OQUAL" CHAR(250) NOT NULL )
               IN "USERSPACE1" ;

ALTER TABLE "TABEX"."BOICAT"
       ADD PRIMARY KEY
              ("OBJID",
               "OTYPE",
               "STMP_D");


ALTER TABLE "TABEX"."BOICAT"
       ADD UNIQUE
              ("OQUAL",
               "STMP_D");
```

Beispiel 4: create_catalog - SQL - Anlage TABEX Katalog

Für historisierte Tabellen kann im Bedarfsfall ein weiterer Tabellenkatalog angelegt werden. Es sind dann in TABEX zwei Datenbankverbindungen mit den unterschiedlichen Katalogen zu konfigurieren.

```
-- create a TABEX4 tablecatalog (DB2/LUW)
-- replace scheme, tablename and dbspace when needed
-- ========================================
-- (c) ritconsult, Linz, Austria, 2010
-- ========================================
CREATE TABLE "TABEX"."BOICATHI"  (
…
```

Beispiel 5: create_catalog_hi - SQL - Anlage TABEX Katalog

4.2 Befüllen des TABEX Tabellenkatalogs

Der TABEX Tabellenkatalog ist im Wesentlichen aus dem TABSYS Verzeichnis (TBTS_VERZEICHNIS) zu befüllen. Können die Daten nicht mit einer einzelnen Abfrage extrahiert werden, dann sind mehrere SQL Selects für die Befüllung des TABEX Tabellenkatalogs auszuführen.

4.2.1 Übernahme der Standardtabellen

```
-- load boicat from tabsys verzeichnis (DB2/LUW)
-- alter sql statement when/where needed
-- ========================================
-- (c) ritconsult, Linz, Austria, 2010
```

```
-- ========================================
connect to <database> user <user>;
delete from TABEX.BOICAT;
insert into TABEX.BOICAT
select substr(TABNAME, 8, 10)                    as OBJID,
       char('DATA', 8)                           as OTYPE,
       char(' '  , 26)                           as STAMP_D,
       cast(current date as char(10))            as DATE_R,
       char('TABADM', 8)                         as USER_A,
       char('SR20621', 8)                        as USER_C,
       cast(current timestamp as char(26))       as STAMP_C,
       char(' ', 8)                              as USER_M,
       char('' , 26)                             as STAMP_M,
       char(TABNAME, 250)                        as OQUAL
from TABSYS.TBTS_VERZEICHNIS
where substr(TABNAME, 7, 1) = '.' and basis_table_qual <> 'TABSYS'
union all
select substr(TABNAME, 5, 10)                    as OBJID,
       char('DATA', 8)                           as OTYPE,
       char(' '  , 26)                           as STAMP_D,
       cast(current date as char(10))            as DATE_R,
       char('TABADM', 8)                         as USER_A,
       char('SR20621', 8)                        as USER_C,
       cast(current timestamp as char(26))       as STAMP_C,
       char(' ', 8)                              as USER_M,
       char('' , 26)                             as STAMP_M,
       char(TABNAME, 250)                        as OQUAL
from TABSYS.TBTS_VERZEICHNIS
where substr(TABNAME, 4, 1) = '.';
commit;
```

Beispiel 6: load_catalog - SQL - Befüllung TABEX Katalog

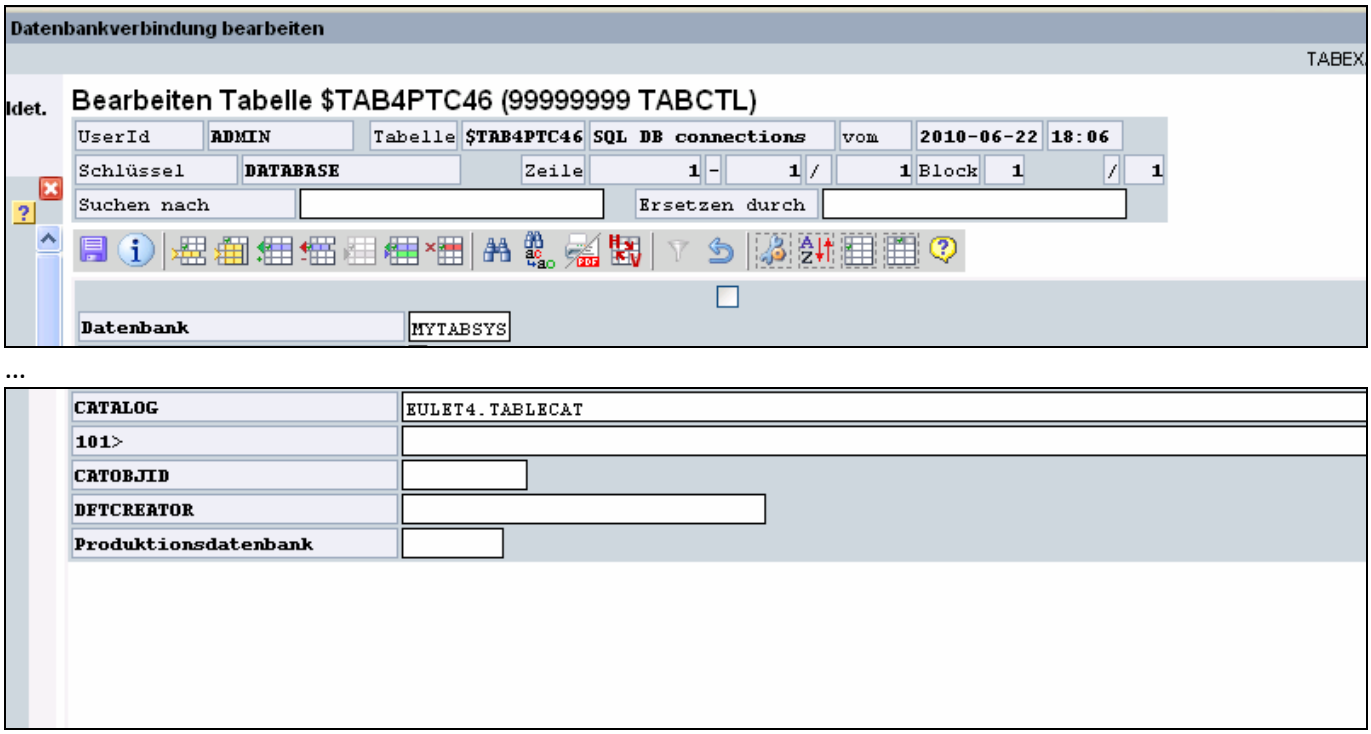
4.2.2 Übernahme der Historientabellen

```
-- load boicathi from tabsys verzeichnis (DB2/LUW)
-- alter sql statement when/where needed
-- ========================================
-- (c) ritconsult, Linz, Austria, 2010
-- ========================================
connect to <database> user <user>;
delete from TABEX.BOICATHI;
insert into TABEX.BOICATHI
select CHAR(HI_TABLE_NAME, 10)                   as OBJID,
       char('DATA', 8)                           as OTYPE,
       char(' '  , 26)                           as STAMP_D,
       cast(current date as char(10))            as DATE_R,
       char('TABADM', 8)                         as USER_A,
       char('SR20621', 8)                        as USER_C,
       cast(current timestamp as char(26))       as STAMP_C,
       char(' ', 8)                              as USER_M,
       char('' , 26)                             as STAMP_M,
       char(trim(HI_TABLE_QUAL) concat '.' concat trim(HI_TABLE_NAME), 250) as OQUAL
from TABSYS.TBTS_VERZEICHNIS
WHERE HI_TABLE_QUAL ^= ' ' ;
commit;
```

Beispiel 7: load_catolog_hi - SQL - Befüllung TABEX Katalog (Historientabellen)

4.3 TABEX Tabellenkatalog in Web Interface bekanntmachen

Der neu angelegte TABEX Tabellenkatalog muss für die TABEX Datenbankverbindung bekannt gemacht werden.



Wenn es sich um eine neue Datenbank handelt, dann sind gegebenenfalls noch weitere TABEX Steuertabellen (siehe BOI Dokumentation) zu pflegen.

5 Migration TABSYS Konfigurationen nach TABEX

5.1 Benutzergruppen

5.1.1 Extrahieren Benutzergruppen

Die Tabellen werden aus TBTS_... extrahiert und in der Migrationsdatenbank zwischengespeichert

```
- ============================================
- load TABEX usergroups from sql select
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set active database to ...
DB=TABVSAM
-
- open a sql connection using ODBC DSN
SQLOPEN,'DSN=DBXHHR03;UID=<userid>;PWD=*****'
-
- create table TSYS_STUG from SELECT-Statement
- alter sql statement when/where needed
SQLTAB,TSYS_STUG,00000000,EE,1,16,,S,Y,Y
SELECT distinct +
char(':' concat substr(BENUTZERGRUPPE, 1, 7), 8) UGR, +
char(substr(TUSER,1,7), 8) USR +
FROM TABSYS.TBTS_BENUTZERGRU +
where substr(tuser,2,1) between '1' and '9'
-
- end of load usergroup
END
```

Beispiel 8: load_usergrp - utility input - Extraktion Benutzergruppen

5.1.2 Zusammenführung Benutzergruppen

Die Benutzergruppen in TABEX werden in Datenbank OPSMSG in Tabelle $TABI4STUG verwaltet
Mit dem Utility Befehl UPDATE werden bestehende Datensätze überschrieben und allfällige neue angelegt.

```
- ============================================
- merge TABEX/4 usergroups with table TSYS_STUG
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set database to OPSMSG
DB=OPSMSG
-
- update usergroup table $TAB4STUG and
- insert all records of TSYS_STUG of databas TABSYS
UPDATE,$TABI4STUG,99999999,,,C
DATA,TSYS_STUG,T,TABVSAM
-
- end of merge usergroup
END
```

Beispiel 9: merge_usergrp - utility input - Zusammenführung Benutzergruppen

5.2 Benutzer

5.2.1 Extrahieren Benutzer

Um nur jene Benutzer zu extrahieren, die auch einer Gruppe zugeordnet sind, werden auch die Benutzer aus der TABSYS Tabelle TBTS_BENUTZERGRU gelesen.
In diesem Script wird das Passwort – abgelaufen Flag auf „L“ gesetzt. Eine Umsetzung auf den Wert „I“ – (passwort initial und somit abgelaufen) erfolgt in einem der Folgeschritte. Es wird hier allen Benutzern das TABEX Profil „$_EH_GERM“ zugewiesen und dass Passwortfeld mit 32 Leerzeichen (Füllzeichen aufgefüllt)

```
- ============================================
- load TABEX users from sql select
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set active database to ...
DB=TABVSAM
-
- open a sql connection using ODBC DSN
SQLOPEN,'DSN=DBXHHR03;UID=<user>;PWD=*****'
-
- create table TSYS_USER from SELECT-Statement
- alter sql statement when/where needed
- flag L will be reset to I later on
SQLTAB,TSYS_USER,00000000,EE,1,8,,S,Y,Y
SELECT DISTINCT char(substr(TUSER,1,7), 8)  as UID,      +
char('L', 1)           AS FLAG,     +
char('$_EH_GERM', 10)  AS PROFILE,  +
char(' ', 32)          AS PWD       +
FROM TABSYS.TBTS_BENUTZERGRU +
where substr(tuser,2,1) between '1' and '9'
-
- end of load usergroup
END
```

Beispiel 10: load_user - utility input - Extraktion Benutzer

5.2.2 Zusammenführung Benutzer

Die Benutzergruppen in TABEX werden in Datenbank OPSMSG in Tabelle $TABI4BTLO verwaltet
Mit dem Utility Befehl UPDATE werden bestehende Datensätze überschrieben und allfällige neue angelegt.

```
- ============================================
- merge TABEX/4 user with table TSYS_USER
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set database to ...
DB=OPSMSG
-
- update user table $TABI4BTLO and
- insert all records of TSYS_USER of databas TABSYS
UPDATE,$TABI4BTLO,99999999,,,C
DATA,TSYS_USER,T,TABVSAM
-
- end of merge users
END
```

Beispiel 11: merge_user - utility input - Zusammenführung Benutzer

5.2.3 Passwort für Benutzer initialisieren

Mit Hilfe des Programms FDTSETMPWD (**f**ield **d**eveloped **t**ool „**set m**ultiple **p**ass**w**or**d**“) können die Passwörter aller neuen Benutzer auf das Passwort eines Referenzbenutzers gesetzt werden. Das neue Passwort wird bei allen Benutzern auf abgelaufen (FLAG = I) gesetzt.

```
$TABX3SP02 JOHNDOE          SSL Programmverzeichnis              22.07.201
Command ==> run TDTSETMPWD                                        Scroll =

Trace NNN  Comp NNN     Date 99999999    Src INTSRC   Mod INTMOD
-----------------------------------Used 042%-------------------------------
* ...Name... Funkt ..Vers.Information.. .Guelt-Ab. ..E-Tag... .E-Zeit. .
  FDTSETMPWD       pwd.set.multiple     00.00.0000 20.07.2010 23:21:57 J
```

...

```
insert the name of the user table..($TABI4BTLO)

insert the name of the user table..... (OPSMSG)

insert the reference user for PWD........ (BOI)

insert the update flag value ..............(L)

INFO update password for all users with FLAG=L
INFO set password in table $TABI4BTLO in database OPSMSG
INFO using reference user BOI       for initial password
INFO reference password is   , ~ëù< É|_¤| kK |_¤| kK |_¤| kK
```

5.3 Objektgruppen

5.3.1 Extrahieren Objektgruppen (Tabellengruppen)

Tabellengruppen können aus den TABSYS Berechtigungen abgeleitet werden. Durch Auswahl der entsprechenden Zeilen aus den TABSY Tabellen und Gruppierung dieser Zeilen mittels SQL Sprachvorrat können die Gruppendefinition für TABEX/4 Tabellengruppen erstellt werden.

```
- ============================================
- load table group from sql select
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set active database to ...
DB=TABVSAM
-
- open a sql connection using ODBC DSN
SQLOPEN,'DSN=DBXHHR03;UID=<user>;PWD=*****'
-
- create table TSYS_... from SELECT-Statement
SQLTAB,TSYS_OGAG1,00000000,EE,1,39,,S,Y,Y
select distinct                 +
char('T', 1)        as TYPE,    +
char(' ', 8)        as SPACE,   +
char(' ', 8)        as INSTANCE, +
char(':AGA01', 10)    as GROUP,    +
smallint(1)         as ORDER,   +
char('T09Y******', 10) as OBJ,   +
char('+', 1)        as ELEM     +
from tabex.boicat a             +
join tabsys.tbts_security b on a.oqual = b.tabname +
where b.ber_TSUSER like 'AGA%'
-
- end of load objectgroup :AGA01
END
```

Beispiel 12: load_objectgrp_aga01 - utility input - Extraktion Objektgruppe

Die Definition der Tabellengruppen kann im Prinzip in beliebiger Komplexität erfolgen.

```
- ============================================
- load table group from sql select
- --------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ============================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set active database to
DB=TABVSAM
-
- open a sql connection using ODBC DSN
SQLOPEN,'DSN=DBXHHR03;UID=<user>;PWD=*****'
-
- create table TSYS_... from SELECT-Statement
SQLTAB,TSYS_OGB17,00000000,EE,1,39,,S,Y,Y
select distinct                 +
char('T', 1)        as TYPE,    +
char(' ', 8)        as SPACE,   +
char(' ', 8)        as INSTANCE, +
char(':B17', 10)    as GROUP,   +
smallint(1)         as ORDER,   +
char(substr(a.objid,1,4) concat '******', 10) as OBJ,   +
char('+', 1)        as ELEM     +
from tabex.boicat a             +
join tabsys.tbts_security b on a.oqual = b.tabname +
where (b.ber_TSUSER like 'KOOR17%' or b.ber_TSUSER like 'WKV%' +
or b.ber_TSUSER like 'INK%') and substr(a.objid,1,4) +
```

```
in ('T03Y', 'T77Y', 'T92Y') +
union   +
select distinct +
char('T', 1)        as TYPE, +
char(' ', 8)        as SPACE, +
char(' ', 8)        as INSTANCE, +
char(':B17', 10)    as GROUP,    +
smallint(1)         as ORDER,    +
char(a.objid, 10)   as OBJ,      +
char('+', 1)        as ELEM      +
from tabex.boicat a              +
join tabsys.tbts_security b on a.oqual = b.tabname +
where (b.ber_TSUSER like 'KOOR17%' or b.ber_TSUSER like 'WKV%' +
or b.ber_TSUSER like 'INK%') and substr(a.objid,1,4) +
not in ('T03Y', 'T77Y', 'T92Y')
-
- end of load group :B17
END
```

Beispiel 13: load_objectgrp_b17 - utility input - Extraktion Objektgruppe

5.3.2 Zusammenführen Objektgruppen (Tabellengruppen)

Die Tabellengruppen in TABEX werden in Datenbank OPSMSG in Tabelle $TABI4BTLO verwaltet
Mit dem Utility Befehl UPDATE werden bestehende Datensätze überschrieben und allfällige neue angelegt.

```
- ==========================================
- merge TABEX/4 tablegroups with TSYS_...
- ------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ==========================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set database to OPSMSG
DB=OPSMSG
-
- update user table $TABI4STOG and
- insert all records of TSYS_OGB17 of database TABVSAM
UPDATE,$TABI4STOG,99999999,,,C
DATA,TSYS_OGB17,T,TABVSAM
- repeat UPDATE and DATA for each group
-
- end of merge tablegroups
END
```

Beispiel 14: merge_objectgrp - utility input - Zusammenführung Objektgruppe(n)

5.4 Berechtigungen

5.4.1 Menüberechtigungen

Menüberechtigungen sind im TABEX/4 Berechtigungssystem zu konfigurieren.
Berechtigungstyp = M

5.4.2 Datenbankberechtigungen

Datenbankberechtigungen sind im TABEX/4 Berechtigungssystem zu konfigurieren.
Berechtigungstyp = D

5.4.3 Datenbankschemaberechtigungen

Datenbankschemaberechtigungen (Creator) sind im TABEX/4 Berechtigungssystem zu konfigurieren.
Berechtigungstyp = C

5.4.4 Tabellenberechtigungen

Tabellenberechtigungen sind im TABEX/4 Berechtigungssystem zu konfigurieren.
Berechtigungstyp = T

5.5 Referenztabellen (pulldown lists (look-up), Prüftabellen, …)

Wenn DB2 Tabellen in TABEX/4 als Referenztabellen verwendet werden sollen, dann sind hierfür folgende Voraussetzungen zu schaffen.

- Vergabe eines 10-stelligen TABEX Tabellennamens für die Tabelle
- Herstellung eine Verbindung zwischen TABEX/4 und den Inhalten in der DB2 Datenbank

Mittels TABEX/4 Utilities und dem Konstrukt einer TABEX/4 „SQLT-Tabelle“ können diese Voraussetzungen geschaffen werden. Für die Verwendung dieser Tabellen ist zu gewährleisten, dass zum Verwendungszeitpunkt eine Verbindung zur Datenbank mit den Tabelleninhalten vorhanden ist.

```
- ==========================================
- create a TABEX reference to a DB2 table
- ------------------------------------------
- (c) ritconsult, Linz, Austria, 2010
- ==========================================
- switch to utility TABN02
SWITCHUTL,TABN02
-
- set active database to ...
DB=TABVSAM
-
- open a sql connection using ODBC DSN
SQLOPEN,'DSN=DBXHHR03;UID=<user>;PWD=*****'
-
- create table LU_SPRACHE from SELECT * from TABADM.T99YSPR
- keep connection in table, do not save any data in TABEX
SQLTAB,LU_SPRACHE,00000000,*,,,TABADM.T99YSPR,N,N
-
- end of create reference
END
```

Beispiel 15: create_lookup - utility input - Anlage Referenztabelle

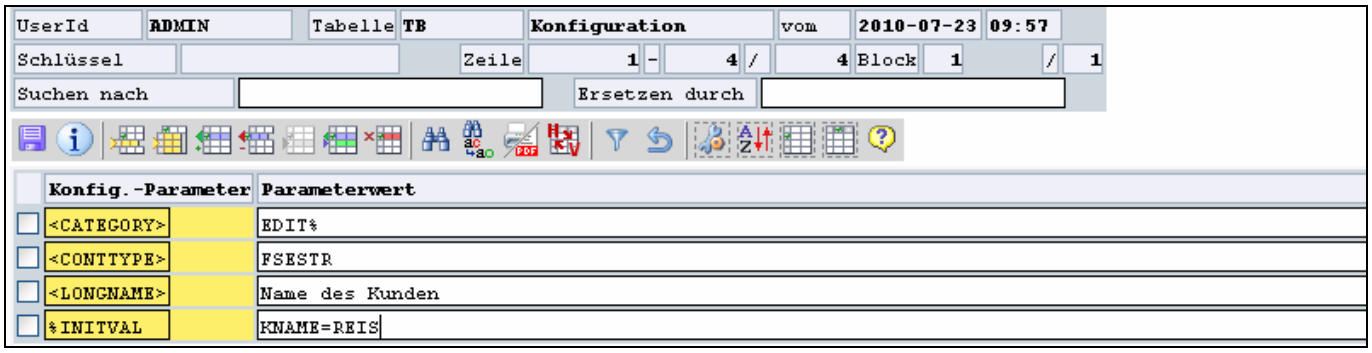
Nachdem die Referenztabelle in TABEX erstellt worden ist, kann die Konfiguration der gewünschten Funktionalität analog zur Verwendung von Standard TABEX Tabellen erfolgen.

5.6 Initialwerte, Feldschutz, diverse Formatierungen

Ein Großteil der TABSYS Funktionalität in diesem Bereich ist mit den “Benutzerdefinierten Einstellungen” in TABEX/4 abzubilden.

Folgende Objekte und Beziehungen sind hierfür zu konfigurieren.

5.6.1 „benutzerspezifische“ (Tabellen) Einstellung

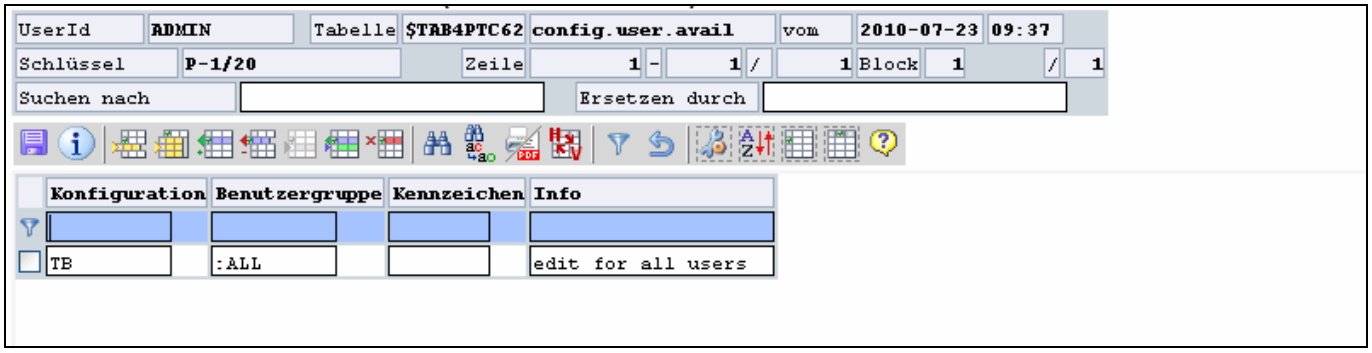


Beispiel 16: Initialwerte - TABEX/4 Konfiguration

Setzt Initialwert für Feld KNAME auf „REIS“. Die Konfiguration ist in einer TABEX Tabelle „TB“ abgelegt.

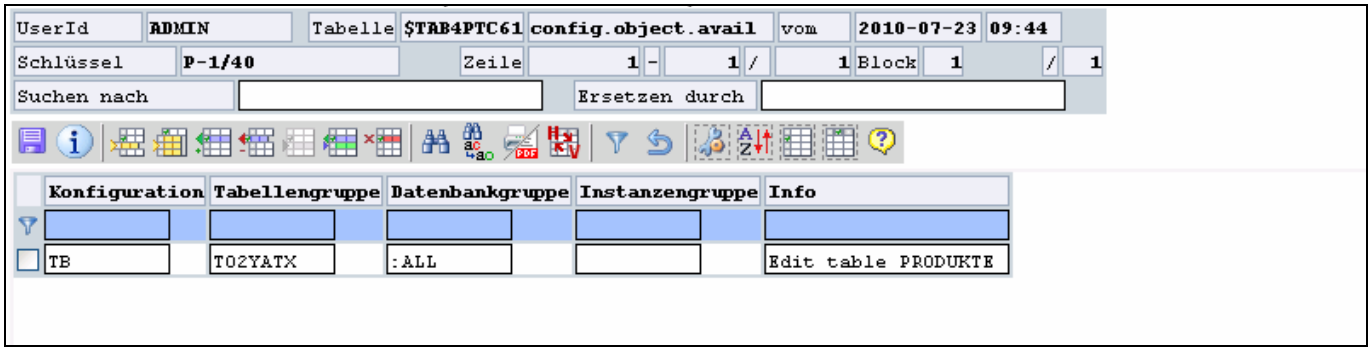
5.6.2 Tabelleneinstellungen Benutzern zuweisen.

Über diese Einstellung werden Tabelleneinstellungen einem oder mehreren Benutzern zugewiesen.



5.6.3 Tabelleneinstellungen Objekten zuweisen

Über diese Einstellung werden Einstellungen einer oder mehreren Tabellen in einer oder mehreren Datenbank(en) zugewiesen.



Hier werden die Einstellungen aus Tabelle „TB“ der Tabelle T02YATX in allen Datenbanken zugewiesen.