Zusammenführung zweier Systeme zu einem System mit zwei Mandanten

Sprache: **Deutsch** · English

Zusammenführung zweier Systeme zu einem System mit zwei Mandanten

> **Zwei Nuclos-Systeme desselben Nuclets zu einem Mandanten-System zusammenführen – Schritt für Schritt mit SQL und pg_dump.**

HOW-TO | ADMINISTRATOR | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

Vorher: Backup & Konzept

Erstelle vor jedem Schritt ein Backup der Datenbank. Lies zuvor die Seite Mandant. Beispiele beziehen sich auf **PostgreSQL**; für andere Datenbanken sind Anpassungen nötig.

Dieses How-to führt zwei Systeme, die auf demselben Nuclet basieren, zu einem System mit **zwei Mandanten** zusammen. Die Daten jedes Systems werden je einem Mandanten zugeordnet; zunächst werden keine Daten geteilt. Ein *einzelnes* System machst du über die Punkte 2, 3 und 7 mandantenfähig.

1. Voraussetzungen

- beide Systeme: gleiche Nuclos- und Nucletversionen
- gleiches Datenbanksystem/Codierung (z.B. Postgres 12, UTF-8)
- gleicher DB-Schemaname (angenommen)

2. Mandanten anlegen & Ebene angleichen

Auf beiden Systemen unter *Administration Mandant* je einen Mandanten anlegen, dann die UID der Mandantenebene per SQL angleichen:

```
INSERT INTO t_md_mandator_level(
        struid, strname, intversion, blnshowname, datcreated, strcreated, datchanged, strchanged
) VALUES(
        'tmp', 'tmp', 1, false, current_date, 'me', current_date, 'me'
);
UPDATE t_md_mandator SET struid_t_md_mandator_level = 'tmp';
UPDATE t_md_mandator_level SET struid = '<struid auf anderem System>' WHERE struid <> 'tmp';
UPDATE t_md_mandator SET struid_t_md_mandator_level = '<struid auf anderem System>';
DELETE FROM t_md_mandator_level WHERE struid = 'tmp';
```

Mandantenabhängige Nucletparameter und Ressourcen (Logos o.ä.) setzen; alle übrigen Parameter auf beiden Systemen gleich halten.

Benutzer vorbereiten

Es darf keine identischen Benutzernamen geben – auf einem System umbenennen und alle Benutzer dem jeweiligen Mandanten zuordnen (Sammelbearbeitung):

```
UPDATE t_md_user set struser = struser || '-<Mandant>';
```

3. Mandantenfähigkeit für BOs aktivieren

Pro Businessobjekt über die Management Console aktivieren und den initialen Mandanten setzen:

```
-package <nuclet-package> -bo “<BO-Name>” -level 1 -initial <Mandantenname> [-uniqueMandator]
```

Das Flag `-uniqueMandator` macht den Mandanten Teil des Unique-Keys. Für viele BOs erzeugt folgendes SQL ein Migrationsskript:

```
set search_path to <schema_name>;

SELECT (
        'UPDATE t_md_entity SET ' || E'\n' ||
        'blnmandatorunique = true, ' || E'\n' ||
        'struid_t_md_mandator_level= ' ||
                '''' || (SELECT struid FROM t_md_mandator_level LIMIT 1) || '''' || ';' || E'\n' ||

        '-- Lege Spalte für Mandator in BO-Tabellen an'  || E'\n' ||
        (SELECT
                 STRING_AGG(
                        'ALTER TABLE ' || strdbentity || ' ADD COLUMN IF NOT EXISTS struid_nuclosmandator
character varying(128); ' ||  E'\n' ||
                         'UPDATE ' || strdbentity || ' SET struid_nuclosmandator = ' || '''' || (SELECT struid
FROM t_md_mandator LIMIT 1) || '''' || ';' || E'\n' ||
                        'ALTER TABLE ' || strdbentity || ' ALTER COLUMN struid_nuclosmandator SET NOT NULL',
                        ';' || E'\n'
                )
        FROM t_md_entity
        WHERE strvirtualentity IS NULL
        ) || ';' || E'\n'
)
```

```
psql [-U "<connect with db user>"]-qAtX <DB-name> < migrate_entities.sql > do_migrate_entities.sql
psql [-U "<connect with db user>"] <DB-name> < do_migrate_entities.sql
```

Caches/Constraints neu aufbauen

Nach nachträglichen Änderungen auf der Management Console ausführen: `invalidate caches`, `rebuild classes`, `rebuild constraints and indexes`.

4. Datenexport

Verwaiste Dokumentreferenzen zuerst entfernen:

```
DELETE FROM <schema_name>.t_md_importfile WHERE struid_documentfile = 'null';
DELETE FROM <schema_name>.t_ud_go_document WHERE struid_t_ud_documentfile = 'null';
DELETE FROM <schema_name>.t_ud_documentfile WHERE struid = 'null';
```

Großteil der Daten als COPY-Statements exportieren (`<nuclet_prefix>` = lokaler Nuclet-Identifizierer):

```
pg_dump --file "my_exported_data.sql" --host "<db host>" --port "<db port>" [--username "<connect with db
user>"] [--no-password] --verbose --format=p --blobs --data-only --no-owner --no-privileges --no-tablespaces --
no-unlogged-table-data --no-comments  --schema "<schema_name>" -t "^<schema_name>.<nuclet_prefix>*" -t
<schema_name>.t_md_mandator -t <schema_name>.t_md_mandator_accessible -t <schema_name>.t_md_user -t
<schema_name>.t_md_role_user -t <schema_name>.t_md_mandator_param_value -t <schema_name>.
t_md_mandator_role_user -t <schema_name>.t_md_passwordhistory -t <schema_name>.t_md_user_setting -t
<schema_name>.t_ud_searchfilter_user -t <schema_name>.t_ud_genericobject -t "<schema_name>.t_ud_document*" -t
"<schema_name>.t_ud_go_*" -t <schema_name>.t_ud_history -t <schema_name>.t_ud_lock  -t <schema_name>.
t_ud_entityobject_relation -t <schema_name>.t_ud_logbook -t <schema_name>.t_ud_timelimittask -t "<db name>"
```

Den Nuclet-Prefix auf den des Zielsystems anpassen:

```
sed -iE "s/<nuclet_prefix>/<nuclet_prefix-zielsystem>/g" my_exported_data.sql
```

Restliche Daten als INSERT (mit `--on-conflict-do-nothing`) exportieren:

```
/usr/bin/pg_dump --file "more_exported_data.sql"  --host "<db host>" --port "<db port>" [--username "<connect
with db user>"] [--no-password] --verbose --format=p --blobs --data-only --no-owner --no-privileges --no-
tablespaces --no-unlogged-table-data --no-comments --inserts --on-conflict-do-nothing --schema "<schema_name>" -
t <schema_name>.t_md_workspace  -t "<schema_name>.t_md_preference*"  "<schema_name>.t_ud_print*" "<db name>"
```

5. Datenimport

Auf der Zieldatenbank zunächst temporär alle Foreign-Key-Constraints deaktivieren (Skript analog zu Schritt 3 erzeugen), dann importieren:

```
psql [-U "<connect with db user>"] -a -v ON_ERROR_STOP=1 -f my_exported_data.sql <DB-name>
psql [-U "<connect with db user>"] -a -v ON_ERROR_STOP=1 -f more_exported_data.sql <DB-name>
```

Anschließend das `documents`-Verzeichnis des anderen Systems in das Zielverzeichnis kopieren.

6. Inhaltliche Anpassungen

SQL-Objekte (VLPs, Views, Funktionen, Datenquellen für Reports/Formulare) müssen ggf. händisch angepasst werden – das ist jeweils eine fachliche Frage.

Verwandte Seiten

Mandant

Konzept.

Öffnen

Backup

Backup.

Öffnen