Export/Import mit PostgreSQL Werkzeugen

(Stand BPS 2.24.4)

Vorbemerkung

Die alten BPS Skripte export.js und import.js sind praktisch für einige Fälle. Vorteil dieser Skripts ist die Unabhängigkeit vom Datenbanktyp, es ist also damit auch möglich ein BPS Schema von Oracle nach PostgreSQL oder umgekehrt umzuziehen. Grosser Nachteil ist jedoch, dass diese Skripte viel langsamer ausgeführt werden als native Datenbank Programmen wie PostgreSQL pg_dumpall/pg_dump/pg_restore. Deshalb bevorzugen viele Administratoren die direkte Verwendung von pg_dump/pg_dumpall, was besonders für 1:1-Backups der ganzen Datenbank absolut in Ordnung ist.

Ab BPS 2.24 stehen nun die nachfolgend beschriebenen Skripte zur Verfügung welche den Export und Import eines einzelnen Schemas mit den PostgreSQL Programmen massiv vereinfachen.

Traditionelle Methode

Das Exportieren eines BPS-Schemas zu Sicherungszwecken kann mit PostgreSQL Werkzeugen problemlos durchgeführt werden, indem entweder die vollständige Datenbank mit pg_dumpall oder nur das Schema mit pg_dump exportiert wird. Das Wiederherstellen in einer vollständigen Datenbank, in der noch keine der Schema- oder BPS-Rollen oder Benutzer vorhanden ist, ist unkompliziert. Wenn jedoch nur ein einzelnes Schema zu übertragen ist sind einige weitere Aktionen erforderlich: Erstellen der Tablespaces und der Rollen schema_ *usr* und schema_ *gst* vor dem Import und rekreieren der Datenbankbenutzer nach dem Import in der Benutzerverwaltung des BPS Arbeitsplatzes.

Es gibt jedoch Fälle, in denen mehr Aktionen erforderlich sind, z.B. das Kopieren auf einen anderen Schemanamen oder das Reorganisieren, weil etwas beschädigt ist (Indizes, Berechtigungen, Trigger, Funktionen ...). Weitere Informationen finden sie unter Export/Import Übersicht.

Hier kommen die Skripte bpspgexp.cmd und bpspgimp.cmd ins Spiel, da sie die meisten Schritte automatisieren die für einen sauberen Export/Import oder eine Reorganisation erforderlich sind.

BPS Skripte für PostgreSQL Werkzeuge

bpspgexp exportiert einerseits die wichtigen Tabellendaten und Sequenzen per pg_dump in eine dmp-Datei. Der beste Zeitpunkt für den Export eines Produktionsschemas ist während niemand anderes daran arbeitet. Nichtsdestotrotz exportiert pg_dump einen Schnappschuss des Zeitpunkts zu dem der Export gestartet wurde, sodass es immer eine konsistente Kopie ergibt.

Vor dem Import mit **bpspgimp** muss auf dem Ziel-PostgreSQL-Server ein sauberes BPS-Schema der gleichen Version bereits vorhanden sein. Das kann ein schon vorhandenes Schema sein oder eines das frisch mit dem BPS Assistenten für neue Datenbanken erstellt wird. Die allfällig dort bereits vorhandenen Tabellendaten und Sequenzen werden erst gelöscht, und danach aus der Exportdatei

frisch importiert. Auf diese Weise bleiben alle Objekte, Berechtigungen usw. in einem sauberen Zustand und Sie haben nachher eine einwandfreie BPS Datenbank.

Nach dem Import mit bpspgimp müssen sie nur noch die BPS-Datenbankbenutzer in der Benutzerverwaltung des BPS Arbeitsplatzes neu generieren.

Die Skripte müssen mit den Anmeldeinformationen des Benutzers postgres oder eines anderen Benutzers mit Superuser-Berechtigung ausgeführt werden.

Instruktionen

Export durchführen

Öffnen sie eine BPS Kommandozeile, erzeugen sie ein Arbeitsverzeichnis und wechseln sie dorthin:

```
mkdir C:\dev\bpspgdmp
cd /d C:\dev\bpspgdmp
```

Prüfen sie dass psql, pg_dump and bpspgexp im PATH gefunden werden:

```
psql -V
pg_dump -V
bpspgexp
```

Beachten sie, dass die Version von pg_dump gleich oder neuer als jene des PostgreSQL Servers sein muss.

Skript ausführen

```
bpspgexp postgres:********@vmpg13/bpsdb lu_buv
```

Beispielausgabe

```
C:\dev\bpspgdmp>bpspgexp postgres:********@vmpg13/bpsdb lu_buv
Checking psql login and schema
Checking for custom tables and sequences
No custom tables or sequences found
Exporting to lu_buv-0-2024004-20210428.pgd
Script bpspgexp completed.

C:\dev\bpspgdmp>
```

Im aktuellen Arbeitsverzeichnis finden sie die Datei lu_buv-0-2024004-20210428.pgd welche durch pg_dump erzeugt wurde.

```
C:\dev\bpspgdmp>dir
 Datenträger in Laufwerk C: ist System
 Volumeseriennummer: 7A49-8313
```

```
 Verzeichnis von C:\dev\bpspgdmp

11.05.2021  12:04    <DIR>          .
11.05.2021  12:04    <DIR>          ..
11.05.2021  11:23         2’780’996 lu_buv-0-2024004-20210428.pgd
               1 Datei(en),        954'422 Bytes
               2 Verzeichnis(se), 48'421'363'712 Bytes frei

C:\dev\bpspgdmp>
```

Import Durchführen

In unserem Beispiel haben wir die Datei lu_buv-0-2024004-20210428.pgd mit dem Export des Schemas LU_BUV von einem anderen Datenbankserver. Das Schema soll auf den Zielserver vmpg12 und hier ebenfalls in ein Schema names LU_BUV importiert werden.

- Kreieren sie ein Arbeitsverzeichnis von wo aus bpspgimp.cmd ausgeführt werden soll.
- Kopieren sie die Datei lu_buv-0-2024004-20210428.pgd in das Arbeitsverzeichnis.
- Prüfen/vorbereiten des Schema LU_BUV auf dem Server vmpg12:
  - Falls das Schema noch nicht existiert, erzeugen sie es mit Hilfe des BPS Assistenten für neue Datenbanken.
  - Falls das Schema schon existiert aber eine alte Version ist, aktualisieren sie es mit dem BPS Assistenten für die Datenbankaktualisierung.
  - Falls das Schema schon existiert aber möglicherweise beschädigt ist, löschen sie es mit dem BPS Assistenten für Datenbank Löschung, und kreieren sie es neu mit Hilfe des BPS Assistenten für neue Datenbanken.
- In allen Fällen stellen sie sicher dass das Schema auf dem Zielsystem die gleiche Version hat wie jenes des Quellsystems, sonst wird der Import möglicherweise fehlschlagen. (Zu sehen in den BPS Einstellungen unter Central System Settings/Install/DbLevel)
- Öffnen sie eine BPS Kommandozeile und führen sie nachfolgendes aus

In das Arbeitsverzeichnis wechseln

```
cd /d C:\dev\bpspgdmp
```

Kontrollieren dass psql, pg_restore and bpspgimp gefunden werden

```
psql -V
pg_restore -V
bpspgimp
```

Skript starten

```
bpspgimp postgres:******@vmpg12/postgres lu_buv-0-2024004-20210428
```

Beispielausgabe

```
C:\dev\bpspgdmp>bpspgimp postgres:********@vmpg12/postgres
lu_buv-0-2024004-20210428
```

```
Check dump file and tool presence
Check psql login and target schema
Create temporary files
Disable all triggers
Truncate tables
Import tables and sequences with pg_restore
Enable all triggers
Analyze tables
Disable auditing
Restore install settings
Rebuild shadows
Enable auditing
Script bpspgimp completed.

C:\dev\bpspgdmp>
```

Im aktuellen Arbeitsverzeichnis finden sie die Datei lu_buv-0-2024004-20210428_imp.log mit dem Log der Importdetails.

```
C:\dev\bpspgdmp>dir
 Datenträger in Laufwerk C: ist System
 Volumeseriennummer: 7A49-8313

 Verzeichnis von C:\dev\bpspgdmp

11.05.2021  12:04    <DIR>          .
11.05.2021  12:04    <DIR>          ..
28.04.2021  16:51         2’780’996 lu_buv-0-2024004-20210428.pgd
28.04.2021  16:57            16’668 lu_buv-0-2024004-20210428_imp.log
               2 Datei(en),      2'797'424 Bytes
               2 Verzeichnis(se), 48'421'363'712 Bytes frei

C:\dev\bpspgdmp>
```

Kundenspezifische Tabellen und Sequenzen

Wenn sie eigene Datenbank-Objekte innerhalb der BPS Datenbank angelegt haben, so *sollte* die Benennung den BPS Konventionen folgen und jeweils ein c (fur „Custom“) vorangestellt haben:

Objekttyp	**BPS Eigene Objekte**	**Kundenspezifische Objekte**
TABLE	t_* ta_* tl_* tn_* …	**c**t_* …
SEQUENCE	s_* sl_* …	**c**s_* …
VIEW	v_* va_* …	**c**v_* …
INDEX	i_* …	**c**i_* …
FUNCTION	f_* …	**c**f_* …
PROCEDURE	p_* …	**c**p_* …
TRIGGER	tr_* tra_* trd#_* trl_* trp#_* …	**c**tr_* …
TYPE	tp_* …	**c**tp_* …
PACKAGE	ptr_* ptrp_* …	**c**ptr_* …

Wenn sie bei der Benennung ihrer kundenspezifischen Objekte generell ein c voranstellen, so ist einerseits garantiert dass sie jetzt und in Zukunft nie in Konflikt mit den BPS eigenen Objekten kommen.

Ausserdem profitieren sie dann davon dass die hier beschriebenen Export- und Import-Programme auch die Inhalte ihrer mit ct_* benannten Tabellen, sowie den aktuellen Stand der mit cs_* benannten Sequenzen sichern und zurückspielen können.

Alle anderen Objekte ausser Tabellen und Sequenzen sind für den Export/Import dieser Skripts irrelevant. Natürlich benötigen sie die auch für ihre Applikationen, aber jene Objekte werden nicht durch diese Skripts erzeugt sondern durch ihre eigenen Setup-Skripte.

Der Exportskript schaut standardmässig ob es irgendwelche Tabellen ct_* oder Sequenzen cs_* gibt, und exportiert diese automatisch mit.

Dateien welche nicht nur BPS eigene Daten sondern auch kundenspezifische enthalten, erkennen sie am Namen. Sie haben nach dem Schema eine 1 statt eine 0:

lu_buv-**0**-2024004-20210428.pgd –> Enthält keine kundenspezifischen Daten
lu_buv-**1**-2024004-20210428.pgd –> Enthält kundenspezifische Daten

Möglicherweise haben sie zwar solche kundenspezifischen Objekte, möchten diese aber doch nicht exportieren (wir sind ihnen z.B. dankbar wenn sie uns ggf. einen Export ohne diese senden). Sie können das beim Export erreichen indem sie am Schluss des Exportbefehls `nocustom` anhängen, z.B:

```
C:\dev\bpspgdmp>bpspgexp postgres:********@vmpg13/bpsdb lu_buv nocustom
```

Der Importskript erkennt am Dateinamen ob er kundenspezifische Daten mitimportieren muss, dort müssen sie also nichts spezielles angeben. Beim Import müssen jedoch die kundenspezifischen Objekte im Zielschema bereits vorhanden und auf dem gleichen Versionsstand sein wie beim Quellschema. Das machen Sie analog den BPS Objekten. Wenn sie z.B. ein neues Schema mit Hilfe des BPS Assistenten für neue Datenbanken inizialisieren, führen sie danach auch die Skripts aus welche ihre kundenspezifischen Objekte generieren. Danach klappt dann auch der Import inklusive den Daten in ihren kundenspezifischen Tabellen und den Zählerständen ihrer kundenspezifischen Sequenzen.

In ein Schema mit anderem Namen importieren

Die Schritte sind grundsätzlich die gleichen wie oben wo wir in das Schema LU_BUV importiert haben.

Diesesmal wollen wir von lu_buv-0-2024004-20210428.pgd in das Schema BPSTEST importieren.

Statt das Schema LU_BUV auf dem Zielserver vorzubereiten, tun wir das diesmal einfach für das Schema BPSTEST .

Ein Schema LU_BUV muss in diesem Fall nicht auf dem Zielserver existieren, allerdings schadet es auch nicht.

Diese Methode erlaubt es also auch in ein anderes Schema auf dem Quellserver zurück zu importieren, ohne das Original-Schema dabei zu verändern.

Das können sie also gut verwenden um ihr Produktions-Schema auf ein Test- oder Qualitäts-Schema zu kopieren.

Skript starten, aber dieses mal hängen wir den Namen des Ziel-Schemas noch an

```
bpspgimp postgres:******@vmpg12/postgres lu_buv-0-2024004-20210428 bpstest
```

Einfach, oder?