PostgreSQL Datenbank unter Windows sichern und wiederherstellen

Kathegorie
PostGreSQL

Denise
Robinson

Postgres bietet drei unterschiedliche Ansätze zum Sichern von Daten:

- SQL-Dump
- Backup auf Dateisystemebene
- Kontinuierliche Archivierung

Wir wollen Dir hier einen kurzen Einblick in diese verschiedenen Methoden geben.

So erstellst du eine PostgreSQL-Dump-Datei

In Postgres wird zum Extrahieren einer Datenbank in eine Skriptdatei oder eine andere Archivdatei ein Dienstprogramm namens **pg_dump** verwendet. Ein entscheidender Vorteil dieses Dienstprogramms besteht darin, dass man seine Dumps auf neueren Versionen von PostgreSQL oder auf Computern mit unterschiedlichen Architekturen wiederherstellen kann. Die anderen Sicherungsmethoden sind an eine bestimmte Serverversion und -architektur gebunden.
Die einfachste Verwendung dieses Befehls ist:

```
pg_dump database_name > database.sql
```

oder

```
pg_dump database_name -f database.sql
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz

info@madafa.de
madafa.de

+49 6131 3331612

Dieser Befehl erstellt eine SQL-Datei mit Befehlen, die zum Neuerstellen der Datenbank in dem Zustand erforderlich sind, in dem sie sich zum Zeitpunkt der Sicherung befand.

Pg_dump im Batch ausführen

Wenn Du eine Datenbank automatisch sichern möchtest, musst Du möglicherweise eine Passwortabfrage entfernen. Dies ist hilfreich, wenn ein Backup ausgeführt wird, wenn kein Benutzer anwesend ist, um ein Passwort einzugeben. Dies erreichst Du, indem Du der Variable **PGPASSWORD** ein Kennwort zuweist:

```
SET PGPASSWORD=my_password pg_dump -U postgres database_name > database.sql
```

Wenn Du das Kennwort nicht in einer Batchdatei behalten möchtest, kannst Du die Anmeldeinformationen alternativ im folgenden Format in **%APPDATA%\postgresql\pgpass.conf** ablegen:

```
hostname:port:database:username:password
```

Sternchen können den Namen des Hosts und der Datenbank ersetzen.
Mit den folgenden Befehlen wird ein Verzeichnis erstellt und der Datensatz als ein Batch zur Datei hinzugefügt:

```
cd %appdata% mkdir postgresql cd postgresql echo localhost:5432:my_
database:postgres:my_password >> pgpass.conf
```

Sichern eines Remote-Servers

Um einen Remote-Server zu sichern, fügst Du die Parameter **-h** und **-p** hinzu:

```
pg_dump -h host_name -p port_number database_name > database.sql
```

Sichern einer einzelnen Tabelle

Um eine einzelne Tabelle zu sichern, benutzt Du den Parameter **-t**:

```
pg_dump -t table_name database_name > table.sql
```

In diesem Fall gibt pg_dump keine anderen Datenbankobjekte aus, die mit der ausgewählten Tabelle verknüpft sind. Dies bedeutet, dass es keine Garantie dafür gibt, dass Du diesen Speicherauszug in einer sauberen Datenbank ohne Fehler wiederherstellen kannst.
In alten (vor 8.2) PostgreSQL-Versionen hat **-t table_name** alle Tabellen mit dem angegebenen Namen ausgegeben. Moderne Postgres-Systeme geben alles aus, was in Ihrem Standardsuchpfad sichtbar ist. Wenn Du zum alten Verhalten zurückkehren möchtest, kannst Du den Befehl **-t "*.table_name"** verwenden.

Komprimieren des Sicherungsskripts

Um die Ausgabedatei zu komprimieren, musst Du den Parameter **-Z** verwenden:

```
pg_dump -Z6 database_name > database.gz
```

Dieser Befehl bewirkt, dass die gesamte Ausgabedatei so komprimiert wird, als ob sie mit einer Komprimierungsstufe von 6 durch **gzip** geführt worden wäre (sie kann von 0 bis 6 variieren).
Eine weitere Option zum Abrufen einer kleineren Sicherungsdatei ist die Verwendung des benutzerdefinierten Dateiformats für die Sicherung.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Die PostgreSQL-Dump-Datei wiederherstellen

Da die von **pg_dump** generierten Textdateien eine Reihe von SQL-Befehlen enthalten, können sie dem Dienstprogramm **psql** zugeführt werden. Die Datenbank selbst wird nicht von psql erstellt, daher musst Du sie zuerst selbst aus **template0** erstellen. Der allgemeine Befehl zum Wiederherstellen eines Dumps lautet also:

```
createdb -T template0 database_name psql database_name < database.sql
```

Bevor Du mit der Wiederherstellung eines SQL-Dumps und der Neuerstellung der Objekte mit den ursprünglichen Berechtigungen beginnst, musst Du sicherstellen, dass alle Benutzer, denen Berechtigungen für Objekte erteilt wurden oder die Objekte in der hochgeladenen Datenbank besitzen, bereits vorhanden sind. Andernfalls schlägt der Wiederherstellungsprozess fehl.

Wiederherstellen einer Remote-Datenbank

Um eine Datenbank auf einem Remote-Server wiederherzustellen, kannst Du **psql** mit den Parametern **-h** und **-p** verbinden:

```
psql -h host_name -p port_number database_name < database.sql
```

Mit folgendem Befehl ist möglich, eine Datenbank direkt von einem Server auf einen anderen zu dumpen, da pg_dump und psql in Pipes schreiben oder daraus lesen können:

```
pg_dump -h source_host database_name | psql -h destination_host database_name
```

Folgender Befehl dupliziert eine Datenbank:

```
createdb -T template0 new_database pg_dump existing_database | psql new_database
```

Fehlerbehandlung

Wenn ein SQL-Fehler auftritt, wird das **psql**-Skript standardmäßig weiterhin ausgeführt. Dieser Ablauf kann geändert werden, indem **psql** mit der Variable **ON_ERROR_STOP** ausgeführt wird. Somit wird psql mit einem Exit-Status von 3 beendet. Dies sieht wie folgt aus:

```
psql --set ON_ERROR_STOP=on database_name < database.sql
```

Wenn ein Fehler auftritt, erhälst Du eine teilweise wiederhergestellte Datenbank. Um dies zu vermeiden und die Wiederherstellung abzuschließen, stellst Du ein, dass ein ganzer Dump als einzelne Transaktion wiederhergestellt wird. Dazu verwendest Du den Parameter -1:

```
psql --set ON_ERROR_STOP=on -1 database_name < database.sql
```

Mehrere PostgreSQL-Datenbanken gleichzeitig sichern

Pg_dump kann jeweils nur eine Datenbank dumpen, wobei Informationen zu Tabellenräumen oder Rollenverteilungen in diesem Dump nicht berücksichtigt werden. Es gibt ein Dienstprogramm **pg_dumpall** , das das bequeme speichern des gesamten Inhalts eines Datenbankclusters unterstützt. Es behält Rollen- und Tabellenbereichsdefinitionen (clusterweite Daten) bei und führt Sicherungen jeder Datenbank in einem bestimmten Cluster durch. **pg_dumpall** funktioniert folgendermaßen: Es gibt Befehle zum Neuerstellen von Tabellenräumen, leeren Datenbanken und Rollen aus und ruft dann **pg_dump** für jede Datenbank auf. Obwohl jede Datenbank intern konsistent ist, werden Snapshots verschiedener Datenbanken möglicherweise nicht vollständig synchronisiert.

Die grundlegende Verwendung dieses Befehls lautet wie folgt:

```
pg_dumpall > all_databases.sql
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Psql und der Parameter **-f** können verwendet werden, um den resultierenden Dump wiederherzustellen:

```
psql -f all_databases.sql postgres
```

Unabhängig davon, mit welcher Datenbank Du eine Verbindung herstellst, enthält die über pg_dumpall erstellte Skriptdatei alle erforderlichen Befehle zum Erstellen und Verbinden mit den gespeicherten Datenbanken.

Eine PostgreSQL-Datenbank als Archivdatei im benutzerdefinierten Format sichern

Während das von pg_dump erzeugte Textformat riesige Ausgabedateien erstellen kann und dadurch nicht sehr flexibel ist, verfügt PostgreSQL über eine weitere Funktion, mit der Benutzer das sogenannte „benutzerdefinierte Format" exportieren können. Diese Format wird standardmäßig (ohne zusätzliche Schritte) archiviert, wodurch beim erneuten Import eine erhebliche Flexibilität geboten wird.
Um eine Backupdatei im benutzerdefinierten Dump-Format zu erstellen, musst Du wie folgt den Parameter **-Fc** hinzufügen:

```
pg_dump -Fc database_name > database.dump
```

Mit dem **pg_restore**-Befehl erstellst Du die Datenbank in dem Dateiformat neu. Dadurch kann **pg_restore** selektiv festlegen, was wiederhergestellt werden soll, oder sogar Elemente vor der Wiederherstellung neu anordnen.
Verwende folgenden Befehl um das benutzerdefinierte Dateiformat wiederherzustellen:

```
pg_restore -d Datenbankname database.dump
```

Weitere PostgreSQL-Sicherungsformate

Der **pg_dump** bietet zwei weitere Ausgabedateiformate: **directory** und **tar**. Beide werden mit **pg_restore** wiederhergestellt.
Um ein Archiv im Directoryformat zu erstellen, musst Du den Parameter **-Fd** verwenden:

```
pg_dump -Fd database_name -f database.dump
```

Es wird ein Verzeichnis erstellt, das die Dump-Objekte in einem maschinenlesbaren Format beschreibt, das pg_restore lesen kann.
Das **tar**-Format ist mit dem Directoryformat kompatibel: Beim Extrahieren eines Archivs im **tar**-Format wird ein Archiv im Directoryformat erstellt. Allerdings kann die relative Reihenfolge der Tabellendaten während des Wiederherstellungsprozesses nicht geändert werden, wenn Du das tar-Format verwenden.
Mithilfe des Parameters **-Ft** kannst Du eine **tar-Datei** erstellen:

```
pg_dump -Ft database_name -f database.tar
```

Datenbankobjektdefinitionen sichern

Von Zeit zu Zeit müssen nur die Datenbankobjektdefinitionen gesichert werden, sodass Du nur das Schema wiederherstellen kannst.
Verwende den folgenden Befehl, um alle Objekte, z.B. Tabellen, Schemas und Berechtigungen, in allen Datenbanken zu sichern:

```
pg_dumpall --schema-only > definitions.sql
```

Um nur die Rollendefinition zu sichern verwende den folgenden Befehl:

```
pg_dumpall --roles-only> role.sql
```

Mit folgendem Befehl kannst Du die Tabellenraumdefinition sichern:

```
pg_dumpall --tablespaces-only > tablespaces.sql
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Backup auf Dateisystemebene

Eine alternative Sicherungsstrategie besteht darin, die Dateien, die PostgreSQL zum Speichern der Daten in der Datenbank verwendet, direkt zu kopieren. Jede Methode für Dateisystemsicherungen kann verwendet werden, z.B.:

```
xcopy “C:\Program Files\PostgreSQL\12\data” “D:\backup” /E
```

Anschließend kannst Du eine neue Serverinstanz mit demselben Versionsnamen in diesem Ordner starten (beachte, dass Du diesen Befehl auf Administratorebene ausführen musst):

```
pg_ctl start -D "D:\backup"
```

Diese Methode bietet Dir folgende Vorteile:

- × So schnell wie einfaches Kopieren von Dateien
- × Ganze Instanz (Cluster) gesichert
- × Verursacht keine Sperrkonflikte und hängt nicht davon ab, dass andere Verbindungen ihre Sperren aufheben
- × Fast sofortiger Wiederherstellungsprozess – Du musst keine SQL-Anweisungen ausführen, um Daten zurückzugewinnen

und gleichzeitig impliziert es einige Einschränkungen:

- × Erfordert das Herunterfahren der Datenbank
- × Kann nur auf derselben Hauptversion von PostgreSQL wiederhergestellt werden
- × Einzelne Datenbanken oder einzelne Tabellen können nicht wiederhergestellt werden: Man muss alles oder nichts wiederherstellen
- × Erzeugt sehr große Backups, da alle Indexe und Bloats enthalten sind und kann daher viel größer sein als SQL-Dumps

Kontinuierliche Archivierung

Bei der kontinuierlichen Archivierungsmethode wird eine Sicherung auf Dateisystemebene mit einer Sicherung der Write-Ahead-Protokoll-Dateien (WAL), in denen jede an den Datendateien der Datenbank vorgenommene Änderung gespeichert wird, kombiniert.

Wie bei einer einfachen Dateisystemsicherungsmethode kann diese Technik nur die Wiederherstellung eines gesamten Datenbankclusters unterstützen, nicht jedoch einer Teilmenge. Darüber hinaus erfordert dies einen großen Archivspeicher, da eine einfache Basissicherung umfangreich sein kann. Dies ist jedoch die bevorzugte Sicherungsmethode in vielen Situationen, in denen eine hohe Zuverlässigkeit erforderlich ist.
Dieser Weg ist schwieriger zu verwalten als jeder der vorherigen Ansätze, hat jedoch einige wesentliche Vorteile. Für weitere Informationen findest Du hier auf der offiziellen PostgreSQL Seite die komplette Vorgehensweise zur Sicherung und Wiederherstellung.

Bilanz: Du solltest nun unter anderem dazu in der Lage sein, mithilfe des pg_dump- und pg_restore-Befehls PostgreSQL-Datenbanken sichern und wiederherstellen zu können und einen Überblick über alternative Sichgerungsmethoden haben.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612