Aide-mémoire des commandes SQL

Voir http://docs.postgresqlfr.org/9.3/sql-commands.html

ABORT

Équivaut à **ROLLBACK**

ALTER AGGREGATE

Change la définition d'une fonction d'agrégat.
Seul le propriétaire de la fonction d'agrégat peut utiliser ALTER AGGREGATE.
Pour modifier le schéma d'une fonction d'agrégat, il est nécessaire de posséder le droit CREATE sur le nouveau schéma.
Pour modifier le propriétaire de la fonction, il faut être un membre direct ou indirect du nouveau rôle propriétaire, rôle qui doit en outre posséder le droit CREATE sur le schéma de la fonction d'agrégat.
Ces restrictions assurent que la modification du propriétaire ne permet pas d'aller au-delà de ce que permet la suppression et la recréation d'une fonction d'agrégat.
Toutefois, un superutilisateur peut modifier la possession de n'importe quelle fonction d'agrégat.

Syntaxe :

```
ALTER AGGREGATE nom ( type_arg [ , ... ] ) RENAME TO
nouveau_nom
ALTER AGGREGATE nom ( type_arg [ , ... ] ) OWNER TO
nouveau_proprietaire
ALTER AGGREGATE nom ( type_arg [ , ... ] ) SET SCHEMA
nouveau_schema
```

Paramètres

nom
: Le nom (éventuellement qualifié du nom du schéma) de la fonction d'agrégat.

type_arg
: Un type de données en entrée sur lequel la fonction d'agrégat opère. Pour référencer une fonction d'agrégat sans argument, écrivez * à la place de la liste des types de données en entrée.

nouveau_nom
: Le nouveau nom de la fonction d'agrégat.

nouveau_propriétaire
: Le nouveau propriétaire de la fonction d'agrégat.

nouveau_schema
: Le nouveau schéma de la fonction d'agrégat.

ALTER COLLATION
ALTER CONVERSION
ALTER DATABASE
ALTER DEFAULT PRIVILEGES
ALTER DOMAIN

ALTER EXTENSION
ALTER FOREIGN DATA WRAPPER
ALTER FOREIGN TABLE
ALTER FUNCTION
ALTER GROUP
ALTER INDEX
ALTER LANGUAGE
ALTER LARGE OBJECT
ALTER OPERATOR
ALTER OPERATOR CLASS
ALTER OPERATOR FAMILY
ALTER ROLE

Modifier un rôle de la base de données

Syntaxe

```
ALTER USER nom [ [ WITH ] option [ ... ] ]
```

Paramètres

option :

```
SUPERUSER | NOSUPERUSER
      | CREATEDB | NOCREATEDB
      | CREATEROLE | NOCREATEROLE
      | CREATEUSER | NOCREATEUSER
      | INHERIT | NOINHERIT
      | LOGIN | NOLOGIN
      | REPLICATION | NOREPLICATION
      | CONNECTION LIMIT limite_connexion
      | [ ENCRYPTED | UNENCRYPTED ] PASSWORD
'motdepasse'
      | VALID UNTIL 'dateheure'
```

Exemples

```
ALTER USER nom RENAME TO nouveau_nom
ALTER USER nom SET parametre_configuration { TO | = } {
valeur | DEFAULT }
ALTER USER nom SET parametre_configuration FROM CURRENT
ALTER USER nom RESET parametre_configuration
ALTER USER nom RESET ALL
```

ALTER SCHEMA
ALTER SEQUENCE
ALTER SERVER
ALTER TABLE
ALTER TABLESPACE
ALTER TEXT SEARCH CONFIGURATION
ALTER TEXT SEARCH DICTIONARY
ALTER TEXT SEARCH PARSER

ALTER TEXT SEARCH TEMPLATE
ALTER TRIGGER
ALTER TYPE
ALTER USER
Équivaut à **ALTER ROLE**
ALTER USER MAPPING
ALTER VIEW
ANALYZE
BEGIN
CHECKPOINT
CLOSE
CLUSTER
COMMENT
COMMIT
COMMIT PREPARED
COPY
CREATE AGGREGATE
CREATE CAST
CREATE COLLATION
CREATE CONVERSION
CREATE DATABASE
Crée une nouvelle base de données.
Pour créer une base de données, il faut être super-utilisateur ou avoir le droit spécial CREATEDB. Cf. CREATE ROLE
Par défaut, la nouvelle base de données est créée en clonant la base système standard template1.
Un modèle différent peut être utilisé en écrivant TEMPLATE nom.
En particulier, la clause TEMPLATE template0 permet de créer une base de données vierge qui ne contient que les objets standards pré-définis dans la version de PostgreSQL™ utilisée. C'est utile pour ne pas copier les objets locaux ajoutés à template1.

Syntaxe

```
CREATE DATABASE nom
    [ [ WITH ] [ OWNER [=] nom_utilisateur ]
               [ TEMPLATE [=] modèle ]
               [ ENCODING [=] codage ]
           [ LC_COLLATE [=] lc_collate ]
           [ LC_CTYPE [=] lc_ctype ]
               [ TABLESPACE [=] tablespace ]
               [ CONNECTION LIMIT [=] limite_connexion ]
]
```

Paramètres
L'ordre des paramètres optionnels n'a aucune importance.

nom
Le nom de la base de données à créer.
nom_utilisateur
Le nom de l'utilisateur propriétaire de la nouvelle base de

données ou DEFAULT pour l'option par défaut (c'est-à-dire le nom de l'utilisateur qui exécute la commande). Pour créer une base de données dont le propriétaire est un autre rôle, vous devez être un membre direct ou direct de ce rôle, ou être un superutilisateur.

modèle

Le nom du modèle squelette de la nouvelle base de données ou DEFAULT pour le modèle par défaut (template1).

codage

Le jeu de caractères de la nouvelle base de données. Peut-être une chaîne (par exemple 'SQL_ASCII'), un nombre de jeu de caractères de type entier ou DEFAULT pour le jeu de caractères par défaut (en fait, celui de la base modèle). Les jeux de caractères supportés par le serveur PostgreSQL™ sont décrits dans Section 22.3.1, « Jeux de caractères supportés ». Voir ci-dessous pour des restrictions supplémentaires.

lc_collate

L'ordre de tri (LC_COLLATE) à utiliser dans la nouvelle base. Ceci affecte l'odre de tri appliqué aux chaînes, par exemple dans des requêtes avec ORDER BY, ainsi que l'ordre utilisé dans les index sur les colonnes texte. Le comportement par défaut est de choisir l'ordre de tri de la base de données modèle. Voir ci-dessous pour les restrictions supplémentaires.

lc_ctype

La classification du jeu de caractères (LC_CTYPE) à utiliser dans la nouvelle base. Ceci affecte la catégorisation des caractères, par exemple minuscule, majuscule et chiffre. Le comportement par défaut est d'utiliser la classification de la base de données modèle. Voir ci-dessous pour les restrictions supplémentaires.

tablespace

Le nom du tablespace associé à la nouvelle base de données ou DEFAULT pour le tablespace de la base de données modèle. Ce tablespace est celui par défaut pour les objets créés dans cette base de données. Voir CREATE TABLESPACE(7) pour plus d'informations.

limite_connexion

Le nombre de connexions concurrentes à la base de données. -1 (valeur par défaut) signifie qu'il n'y a pas de limite.

La commande CREATE DATABASE ne peut pas être exécutée à l'intérieur d'un bloc de transactions.

Les erreurs sur la ligne « ne peut initialiser le répertoire de la base de données » (« could not initialize database directory » dans la version originale) sont le plus souvent dues à des droits insuffisants sur le répertoire de données, à un disque plein ou à un autre problème relatif au système de

fichiers.

L'instruction DROP DATABASE(7) est utilisée pour supprimer la base de données.

Le programme createdb(1) est un enrobage de cette commande fourni par commodité.

Bien qu'il soit possible de copier une base de données autre que template1 en spécifiant son nom comme modèle, cela n'est pas (encore) prévu comme une fonctionnalité « COPY DATABASE » d'usage général. La limitation principale est qu'aucune autre session ne peut être connectée à la base modèle pendant sa copie. CREATE DATABASE échouera s'il y a une autre connexion au moment de son exécution ; sinon, les nouveaux connexions à la base modèle seront verrouillées jusqu'à la fin de la commande CREATE DATABASE. La Section 21.3, « Bases de données modèles » fournit plus d'informations à ce sujet.

L'encodage du jeu de caractère spécifié pour la nouvelle base de données doit être compatible avec les paramètre de locale (LC_COLLATE et LC_CTYPE). Si la locale est C (ou de la même façon POSIX), alors tous les encodages sont autorisés. Pour d'autres paramètres de locale, il n'y a qu'un encodage qui fonctionnera correctement. (Néanmoins, sur Windows, l'encodage UTF-8 peut être utilisée avec toute locale.) CREATE DATABASE autorisera les superutilisateurs à spécifier l'encodage SQL_ASCII quelque soit le paramètre locale mais ce choix devient obsolète et peut occasionner un mauvais comportement des fonctions sur les chaînes si des données dont l'encodage n'est pas compatible avec la locale sont stockées dans la base.

Les paramètres d'encodage et de locale doivent correspondre à ceux de la base modèle, excepté quand la base template0 est utilisée comme modèle. La raison en est que d'autres bases de données pourraient contenir des données qui ne correspondent pas à l'encodage indiqué, ou pourraient contenir des index dont l'ordre de tri est affecté par LC_COLLATE et LC_CTYPE. Copier ces données peut résulter en une base de données qui est corrompue suivant les nouveaux paramètres. template0, par contre, ne contient aucun index pouvant être affecté par ces paramètres.

L'option CONNECTION LIMIT n'est qu'approximativement contraignante ; si deux nouvelles sessions commencent sensiblement en même temps alors qu'un seul « connecteur » à la base est disponible, il est possible que les deux échouent. De plus, les superutilisateurs ne sont pas soumis à cette limite.

Exemples

Créer une nouvelle base de données :

```
CREATE DATABASE lusiadas;
```

Créer une base de données ventes possédée par l'utilisateur app_ventes utilisant le tablespace espace_ventes comme espace par défaut :

```
CREATE DATABASE ventes OWNER app_ventes
TABLESPACE espace_ventes;
```

Créer une base de données musique qui supporte le jeu de caractères ISO-8859-1 :

```
CREATE DATABASE musique ENCODING 'LATIN1'
TEMPLATE template0;
```

Dans cet exemple, la clause TEMPLATE template0 n'est requise que si l'encodage de template1 n'est pas ISO-8859-1. Notez que modifier l'encodage pourrait aussi nécessiter de sélectionner de nouveaux paramètres pour LC_COLLATE et LC_CTYPE.connexion]]

CREATE DOMAIN
CREATE EXTENSION
CREATE FOREIGN DATA WRAPPER
CREATE FOREIGN TABLE
CREATE FUNCTION
CREATE GROUP
CREATE INDEX
CREATE LANGUAGE
CREATE OPERATOR
CREATE OPERATOR CLASS
CREATE OPERATOR FAMILY

CREATE ROLE

Ajoute un nouveau rôle dans une grappe (cluster) de bases de données PostgreSQL™.
Un rôle est une entité qui peut posséder des objets de la base de données et avoir des droits sur la base.
Il peut être considéré comme un « utilisateur », un « groupe » ou les deux suivant la façon dont il est utilisé.
Cette commande nécessite le droit CREATEROLE ou d'être superutilisateur.
Les rôles sont définis au niveau de la grappe de bases de données, et sont donc valides dans toutes les bases de la grappe.

Syntaxe

```
CREATE ROLE nom [ [ WITH ] option [ ... ] ]
où option peut être :
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
    | CONNECTION LIMIT limite_connexion
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'motdepasse'
    | VALID UNTIL 'heuredate'
    | IN ROLE nom_role [, ...]
    | IN GROUP nom_role [, ...]
    | ROLE nom_role [, ...]
    | ADMIN nom_role [, ...]
    | USER nom_role [, ...]
    | SYSID uid
```

Paramètres

nom

nom du nouveau rôle.

option (valeurs par défaut soulignées ou en gras)

SUPERUSER, NOSUPERUSER

SUPERUSER → **superutilisateur** qui a tous les droits d'accès à la base de données.
NOSUPERUSER → pas superutilisateur

Dangereux, à n'utiliser que si nécessaire. Seul un superutilisateur peut créer un superutilisateur.

CREATEDB, NOCREATEDB

CREATEDB → autorisé à créer des bases de données
NOCREATEDB → pas autorisé à créer des bases de données

CREATEROLE, NOCREATEROLE

CREATEROLE → autorisé à créer de nouveaux rôles
NOCREATEROLE → pas autorisé à créer de nouveaux rôles

un rôle qui possède le droit CREATEROLE peut aussi modifier ou supprimer d'autres rôles.

INHERIT, NOINHERIT

héritage des droits conférables (c'est-à-dire les droits d'accès aux objets de la base de données et les appartenances aux rôles).

INHERIT → ce rôle hérite des privilèges détenus par un rôle dont il est membre direct ou indirect.
NOINHERIT → l'appartenance à un autre rôle ne lui confère que la possibilité d'utiliser SET ROLE pour acquérir les droits de l'autre rôle ; ils ne sont disponibles qu'après cela.

Ne s'applique pas aux attributs de rôle spéciaux configurés par CREATE ROLE et ALTER ROLE.
Par exemple, être membre d'un rôle disposant du droit CREATEDB ne confère pas automatiquement le droit de création de bases de données, même avec INHERIT ; il est nécessaire d'acquérir ce rôle via SET ROLE avant de créer une base de données.

LOGIN, NOLOGIN

précise si un rôle est autorisé à se connecter, c'est-à-dire si le rôle peut être donné comme nom pour l'autorisation initiale de session à la connexion du client.
Un rôle ayant l'attribut LOGIN peut être vu comme un utilisateur.
Les rôles qui ne disposent pas de cet attribut sont

utiles pour gérer les privilèges de la base de données mais ne sont pas des utilisateurs au sens habituel du mot.

REPLICATION, NOREPLICATION
: détermine si un rôle peut initier une réplication en flux ou placer le système en mode sauvegarde ou l'en sortir.
Un rôle ayant l'attribut REPLICATION est un rôle très privilégié et ne devrait être utilisé que pour la réplication.

CONNECTION LIMIT limiteconnexion
: nombre maximum de connexions concurrentes possibles pour le rôle, s'il possède le droit de connexion.
valeur par défaut = -1 ⇒ pas de limite.

PASSWORD motdepasse
: Le mot de passe du rôle.
seulement utile pour les rôles ayant l'attribut LOGIN, mais il est possible d'en définir un pour les rôles qui ne l'ont pas.
Cette option peut être omise si l'authentification par mot de passe n'est pas envisagée.
Si aucun mot de passe n'est spécifié, le mot de passe sera NULL et l'authentification par mot de passe échouera toujours pour cet utilisateur. Un mot de passe NULL peut aussi être indiqué explicitement avec PASSWORD NULL.

ENCRYPTED, UNENCRYPTED
: chiffrement du mot de passe stocké dans les catalogues système.
En l'absence de précision, le comportement par défaut est déterminé par le paramètre de configuration password_encryption.
Si le mot de passe présenté est déjà une chaîne chiffrée avec MD5, il est stocké ainsi, quelque soit le mot-clé spécifié, ENCRYPTED ou UNENCRYPTED (le système ne peut pas déchiffrer la chaîne déjà chiffrée). Cela permet de recharger des mots de passe chiffrés lors d'opérations de sauvegarde/restauration.
D'anciens clients peuvent ne pas disposer du support pour le mécanisme d'authentification MD5, nécessaire pour travailler avec les mots de passe stockés chiffrés.

VALID UNTIL 'dateheure'
: date et heure d'expiration du mot de passe uniquement (pas du rôle)
Les date et heure d'expiration ne sont pas vérifiées lors de connexions à l'aide de méthodes d'authentification qui n'utilisent pas les mots de passe.

Sans précision, le mot de passe est indéfiniment valide.

IN ROLE nom_role
: rôles dont le nouveau rôle est membre.
pas d'option pour ajouter le nouveau rôle en tant qu'administrateur ; cela se fait à l'aide d'une commande GRANT séparée.

ROLE nom_role
: liste les rôles membres du nouveau rôle. Le nouveau rôle devient ainsi un groupe.

ADMIN nom_role
: équivalente à ROLE, mais les rôles nommés sont ajoutés au nouveau rôle avec l'option WITH ADMIN OPTION. Cela leur confère le droit de promouvoir à d'autres rôles l'appartenance à celui-ci.

Tous les attributs positionnés par **CREATE ROLE** peuvent être modifiés ensuite à l'aide de commandes ALTER ROLE.
DROP ROLE permet de supprimer un rôle.
Il est préférable d'utiliser [[internet:grant]] et [[REVOKE]] pour ajouter et supprimer des membres de rôles utilisés comme groupes.

L'attribut INHERIT est la valeur par défaut pour des raisons de compatibilité descendante : dans les précédentes versions de PostgreSQL™, les utilisateurs avaient toujours accès à tous les droits des groupes dont ils étaient membres. Toutefois, NOINHERIT est plus respectueux de la sémantique spécifiée dans le standard SQL.

Le privilège CREATEROLE impose quelques précautions. Il n'y a pas de concept d'héritage des droits pour un tel rôle. Cela signifie qu'un rôle qui ne possède pas un droit spécifique, mais est autorisé à créer d'autres rôles, peut aisément créer un rôle possédant des droits différents des siens (sauf en ce qui concerne la création des rôles superutilisateur). Par exemple, si le rôle « user » a le droit CREATEROLE mais pas le droit CREATEDB, il peut toujours créer un rôle possédant le droit CREATEDB. Il est de ce fait important de considérer les rôles possédant le privilège CREATEROLE comme des superutilisateurs en puissance.

PostgreSQL™ inclut un programme, createuser(1) qui possède les mêmes fonctionnalités que CREATE ROLE (en fait, il appelle cette commande) et peut être lancé à

partir du shell.

L'option CONNECTION LIMIT n'est vérifiée qu'approximativement. Si deux nouvelles sessions sont lancées à peu près simultanément alors qu'il ne reste qu'un seul « emplacement » de connexion disponible pour le rôle, il est possible que les deux échouent. De plus, la limite n'est jamais vérifiée pour les superutilisateurs.

Faites attention lorsque vous donnez un mot de passe non chiffré avec cette commande. Le mot de passe sera transmis en clair au serveur. Ce dernier pourrait être tracer dans l'historique des commandes du client ou dans les traces du serveur. Néanmoins, la commande createuser(1) transmet le mot de passe chiffré. De plus, psql(1) contient une commande \password que vous pouvez utiliser pour modifier en toute sécurité votre mot de passe.

Exemples

Créer un rôle qui peut se connecter mais sans lui donner de mot de passe :

```
CREATE ROLE jonathan LOGIN;
```

Créer un rôle avec un mot de passe :

```
CREATE USER davide WITH PASSWORD 'jw8s0F4';
```

(CREATE USER est identique à CREATE ROLE mais implique LOGIN.)

Créer un rôle avec un mot de passe valide jusqu'à fin 2006. Une seconde après le passage à 2007, le mot de passe n'est plus valide.

```
CREATE ROLE miriam WITH LOGIN PASSWORD
'jw8s0F4' VALID UNTIL '2007-01-01';
```

Créer un rôle qui peut créer des bases de données et gérer des rôles :

```
CREATE ROLE admin WITH CREATEDB CREATEROLE;
```

CREATE RULE
CREATE SCHEMA

CREATE SEQUENCE
CREATE SERVER
CREATE TABLE
CREATE TABLE AS
CREATE TABLESPACE
CREATE TEXT SEARCH CONFIGURATION
CREATE TEXT SEARCH DICTIONARY
CREATE TEXT SEARCH PARSER
CREATE TEXT SEARCH TEMPLATE
CREATE TRIGGER
CREATE TYPE
CREATE USER

Alias de **CREATE ROLE**
Mais avec **LOGIN** par défaut.

CREATE USER MAPPING
CREATE VIEW
DEALLOCATE

DECLARE
DELETE
DISCARD
DO
DROP

DROP AGGREGATE
DROP CAST
DROP COLLATION
DROP CONVERSION
DROP DATABASE

La commande DROP DATABASE détruit une base de données. Elle supprime les entrées du catalogue pour la base et le répertoire contenant les données. Elle ne peut être exécutée que par le propriétaire de la base de données ou le superutilisateur. De plus, elle ne peut être exécutée si quelqu'un est connecté sur la base de données cible, y compris l'utilisateur effectuant la demande de suppression. (On peut se connecter à postgres ou à toute autre base de données pour lancer cette commande.)
DROP DATABASE ne peut pas être annulée. Il convient donc de l'utiliser avec précaution !

Syntaxe

```
DROP DATABASE [ IF EXISTS ] nom
```

Paramètres

IF EXISTS

Ne pas renvoyer une erreur si l'agrégat n'existe pas. Un message d'avertissement est affiché dans ce cas.

name

Le nom de la base de données à supprimer.

DROP DATABASE ne peut pas être exécutée à l'intérieur d'un bloc de transactions.

Cette commande ne peut pas être exécutée en cas de connexion à la base de données cible. Il peut paraître plus facile d'utiliser le programme dropdb(1) à la place, qui est un enrobage de cette commande.

DROP DOMAIN
DROP EXTENSION
DROP FOREIGN DATA WRAPPER
DROP FOREIGN TABLE
DROP FUNCTION
DROP GROUP
DROP INDEX
DROP LANGUAGE
DROP OPERATOR
DROP OPERATOR CLASS
DROP OPERATOR FAMILY
DROP OWNED
DROP ROLE

Supprime le(s) rôle(s) spécifié(s). Seul un superutilisateur peut supprimer un rôle de superutilisateur. Le droit CREATEROLE est nécessaire pour supprimer les autres rôles.
Un rôle ne peut pas être supprimé s'il est toujours référencé dans une base de données du groupe. Dans ce cas, toute tentative aboutit à l'affichage d'une erreur. Avant de supprimer un rôle, il est nécessaire de supprimer au préalable tous les objets qu'il possède (ou de modifier leur appartenance) et de supprimer tous les droits définis par ce rôle. Les commandes REASSIGN OWNED(7) et DROP OWNED(7) peuvent être utiles pour cela.
Néanmoins, il n'est pas nécessaire de supprimer toutes les appartenances de rôle impliquant ce rôle ; DROP ROLE supprime automatiquement toute appartenance du rôle cible dans les autres rôles et des autres rôles dans le rôle cible. Les autres rôles ne sont pas supprimés ou affectés.

Syntaxe

```
DROP ROLE [ IF EXISTS ] nom [, ...]
```

Paramètres

IF EXISTS

Ne pas renvoyer une erreur si l'agrégat n'existe pas. Un message d'avertissement est affiché dans ce cas.

nom

Le nom du rôle à supprimer.

PostgreSQL™ inclut un programme dropuser(1) qui a la même fonctionnalité que cette commande (en fait, il appelle cette commande) mais qui est lancé à partir du shell.

Exemples

Supprimer un rôle :

```
DROP ROLE jonathan;
```

DROP RULE
DROP SCHEMA
DROP SEQUENCE
DROP SERVER
DROP TABLE
DROP TABLESPACE
DROP TEXT SEARCH CONFIGURATION
DROP TEXT SEARCH DICTIONARY
DROP TEXT SEARCH PARSER
DROP TEXT SEARCH TEMPLATE
DROP TRIGGER
DROP TYPE
DROP USER
DROP USER MAPPING
DROP VIEW

END
EXECUTE
EXPLAIN
FETCH
GRANT
INSERT
LISTEN
LOAD
LOCK
MOVE
NOTIFY
PREPARE

PREPARE TRANSACTION

REASSIGN OWNED
REINDEX
RELEASE SAVEPOINT
RESET
REVOKE
ROLLBACK

Annule la transaction en cours et toutes les modifications effectuées lors de cette transaction.

Syntaxe

```
ROLLBACK [ WORK | TRANSACTION ]
```

Paramètres

```
WORK, TRANSACTION
```

mots clés optionnels. Ils sont sans effet.

Notes

L'utilisation de la commande COMMIT(7) permet de terminer une transaction avec succès.
Lancer ROLLBACK en dehors de toute transaction n'a pas d'autre conséquence que l'affichage d'un message d'avertissement.

Exemples

Pour annuler toutes les modifications :

```
ROLLBACK;
```

ROLLBACK PREPARED
ROLLBACK TO SAVEPOINT

SAVEPOINT
SECURITY LABEL
SELECT

SELECT INTO

SET

SET CONSTRAINTS
SET ROLE
SET SESSION AUTHORIZATION
SET TRANSACTION

SHOW
START TRANSACTION
TABLE
TRUNCATE
UNLISTEN
UPDATE
VACUUM
VALUES

WITH