Commandes GRANT REVOKE

Modification des droits sur des objets :

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
  [,...] | ALL [ PRIVILEGES ] }
  ON [ TABLE ] tablename [, ...]
  TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
  ON DATABASE dbname [, ...]
  TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
  ON FUNCTION funcname ([type, ...]) [, ...]
  TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
  ON LANGUAGE langname [, ...]
  TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
  ON SCHEMA schemaname [, ...]
  TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Pour la syntaxe de `REVOKE`, simplement remplacer `GRANT` par `REVOKE` et `TO` par `FROM` ! De plus, l'option `WITH GRANT OPTION` est transférée en tête sous forme `REVOKE [GRANT OPTION FOR]` ....

Composé avec LaTeX, révision 3296

Syntaxe SQL pour PostgreSQL

Fabien Coelho – MINES ParisTech

Conventions

Les convensions suivantes sont utilisées dans la description des syntaxes :
répétition ... **facultatif** `[ ]` **obligatoire** `{ }` **alternative** `|`

Commande SELECT

Extraction de tuple à partir d'une relation (`TABLE VIEW`...).

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
  * | expression [ AS output_name ] [, ...]
  [ FROM from_item [, ...] ]
  [ WHERE condition ]
  [ GROUP BY expression [, ...] ]
  [ HAVING condition [, ...] ]
  [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
  [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
  [ LIMIT { count | ALL } ]
  [ OFFSET start ] ;
```

Où `from_item` peut être :

```
  [ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
  [ LATERAL ] ( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
  [ LATERAL ]function_name ( [ argument [, ...] ] ) [ AS ] alias
        [ ( column_alias [, ...] | column_definition [, ...] ) ]
  [ LATERAL ] function_name ( [ argument [, ...] ] ) AS ( column_def [, ...] )
  [ LATERAL ] from_item [ NATURAL ] join_type from_item
      [ ON join_condition | USING ( join_column [, ...] ) ]
```

Commande UPDATE

Mise à jour d'un tuple dans une relation :

```
UPDATE [ ONLY ] table
  SET column = { expression | DEFAULT } [, ...]
  [ FROM fromlist ]
  [ WHERE condition ] ;
```

Commande DELETE

Efface des tuples d'une relation :

```
DELETE FROM [ ONLY ] table [ WHERE condition ] ;
```

Commande INSERT

Ajoute un nouveau tuple dans une relation :

```
INSERT INTO table [ ( column [, ...] ) ]
  { DEFAULT VALUES
  | VALUES ( { expression | DEFAULT } [, ...] )[, ...]
  | query }
  [ ON CONFLICT [ (key-column) ] DO
    { NOTHING | UPDATE SET column_name = ...[, ...] } ]
```

Commande CREATE TABLE

Création d'une nouvelle table :

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name (
{ column_name data_type [ DEFAULT default_expr ] [ column_constraint [...] ]
| table_constraint
| LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ]
}  [, ... ] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ] ;
```

Où `column_constraint` est :

```
[ CONSTRAINT constraint_name ]
{ NOT NULL
| NULL
| UNIQUE
| PRIMARY KEY
| CHECK (expression)
| REFERENCES reftable [( refcolumn )]
  [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
  [ ON DELETE action ] [ ON UPDATE action ]
}
  [ DEFERRABLE | NOT DEFERRABLE ]
  [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

et `table_constraint` est :

```
[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] )
| PRIMARY KEY ( column_name [, ... ] )
| CHECK ( expression )
| FOREIGN KEY ( column_name [, ... ] )
  REFERENCES reftable [ ( refcolumn [, ... ] ) ]
  [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
  [ ON DELETE action ] [ ON UPDATE action ]
}
[ DEFERRABLE | NOT DEFERRABLE ]
[ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```


Commande ALTER TABLE

Modification de la définition d'une table :

```
ALTER TABLE [ ONLY ] name [ * ]
{ ADD [ COLUMN ] column type [ column_constraint [ ... ] ]
| DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
| ALTER [ COLUMN ] column { SET DEFAULT expression | DROP DEFAULT }
| ALTER [ COLUMN ] column { SET | DROP } NOT NULL
| ALTER [ COLUMN ] column SET STATISTICS integer
| ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
| SET WITHOUT OIDS
| RENAME [ COLUMN ] column TO new_column
| ADD table_constraint
| DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
| RENAME TO new_name
| OWNER TO new_owner
| CLUSTER ON index_name
}
```

Commande CREATE INDEX

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] name ]
  ON table_name [ USING method ]
  ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ]
       [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...] )
  [ WITH ( storage_parameter = value [, ... ] ) ]
  [ TABLESPACE tablespace_name ]
  [ WHERE predicate ]
```

Avec pour `method` : `btree hash gist spgist gin brin`

Command DROP

Efface un objet (avec propagation ou non) :

```
DROP { TABLE | VIEW | DATABASE | INDEX | USER | ... } name
  [ CASCADE | RESTRICT ] ;
```

Commande SHOW SET

Consultation et modification des options :

```
SHOW { option_name | ALL } ;
SET option_name TO new_value ;
```