DDL and DML

Zach Perry

Types of SQL Commands

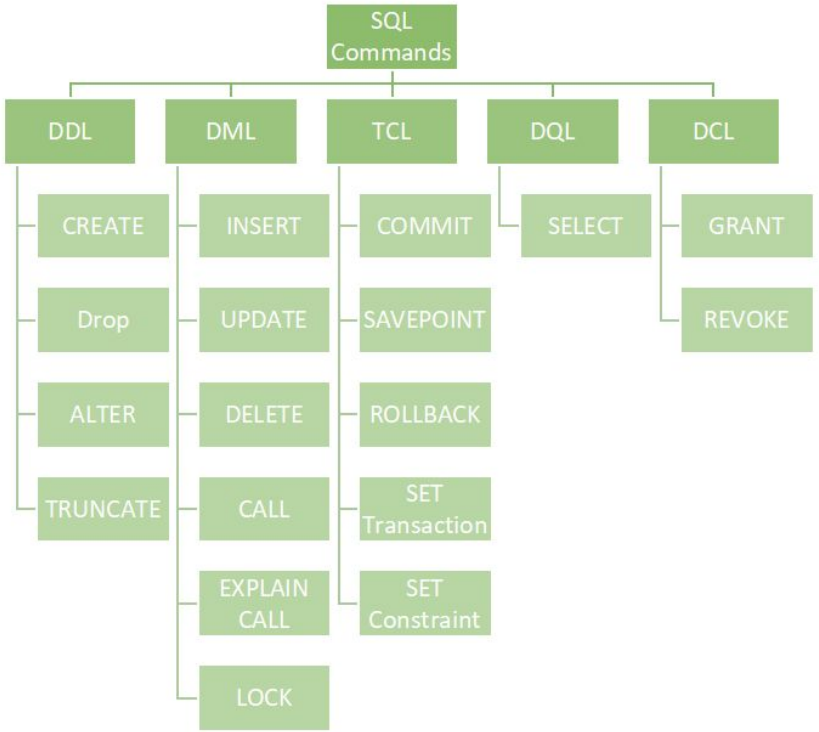
DDL – Data Definition Language

DQL – Data Query Language

DML – Data Manipulation Language

DCL – Data Control Language

TCL – Transaction Control Language



DDL and DML

- DDL - Data Definition Language
  - Define the database schema
  - Interactions with database objects
- DML - Data Manipulation Language
  - Manipulate the data in the database
  - Interactions with data

Basic DDL Commands

- CREATE
- ALTER
- DROP
- TRUNCATE

CREATE

CREATE

- Used to create **database objects**:
  - database
  - table
  - user
  - role
  - view
  - procedure

CREATE

Basic syntax:

CREATE <OBJECT> <name>;

```
create database company
with owner david;

create user salesadmin
with password 'admin@123';
```

CREATE TABLE (basic syntax)

```
CREATE TABLE table_name (
        column_name datatype (size),
        column_name2 datatype (size)
);
```

CREATE TABLE (example)

```
CREATE TABLE student (
        name varchar(100),
        locker_number int,
        student_id int
);
```

CREATE TABLE (full syntax)

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN | DEFAULT } ] [ COMPRESSION compression_method ] [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ PARTITION BY { RANGE | LIST | HASH } ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] ) ]
[ USING method ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]
```

https://www.postgresql.org/docs/current/sql-createtable.html

CREATE TABLE (full syntax)

only create table if another with same name doesn't already exist

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN | DEFAULT } ] [ COMPRESSION compression_method ] [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ PARTITION BY { RANGE | LIST | HASH } ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] ) ]
[ USING method ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace_name ]
```

new table automatically copies all column names, their data types, and their not-null constraints.

https://www.postgresql.org/docs/current/sql-createtable.html

Constraints

```
[ CONSTRAINT constraint_name ]
{ NOT NULL |
  NULL |
  CHECK ( expression ) [ NO INHERIT ] |
  DEFAULT default_expr |
  GENERATED ALWAYS AS ( generation_expr ) STORED |
  GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY [ ( sequence_options ) ] |
  UNIQUE [ NULLS [ NOT ] DISTINCT ] index_parameters |
  PRIMARY KEY index_parameters |
  REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
    [ ON DELETE referential_action ] [ ON UPDATE referential_action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

CREATE TABLE (example)

```
CREATE TABLE student (
        name varchar(100) DEFAULT 'Geoff' ,
        locker_number int CHECK (locker_number > 0),
        student_id int GENERATED ALWAYS AS (locker_number + 5) STORED
);
```

ensures values will be > 0

generates student_id by adding 5 to locker_number

A helpful little note on default

“The default value can be an expression, which will be evaluated whenever the default value is inserted (not when the table is created). A common example is for a timestamp column to have a default of CURRENT_TIMESTAMP, so that it gets set to the time of row insertion” - Documentation

Generated columns

- “A generated column cannot be written to directly. In INSERT or UPDATE commands, a value cannot be specified for a generated column
- Some restrictions for generated columns and tables involving generated columns:
  - The generation expression can only use immutable functions and cannot use subqueries or reference anything other than the current row in any way.
  - A generation expression cannot reference another generated column.
  - A generation expression cannot reference a system column, except tableoid.
  - A generated column cannot have a column default or an identity definition.

CREATE TABLE (named constraints)

“You can also give the constraint a separate name. This clarifies error messages and allows you to refer to the constraint when you need to change it” - Documentation

```
CREATE TABLE student (
        name varchar(100) ,
        locker_number int CONSTRAINT positive_num CHECK (locker_number > 0),
        student_id int GENERATED ALWAYS AS (locker_number + 5) STORED
);
```

CREATE TABLE AS

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name
    [ (column_name [, ...] ) ]
    [ USING method ]
    [ WITH ( storage_parameter [= value] [, ... ] ) | WITHOUT OIDS ]
    [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
    [ TABLESPACE tablespace_name ]
    AS query
    [ WITH [ NO ] DATA ]
```

make a table using values from another table

CREATE TABLE AS (example)

CREATE TABLE student_copy AS SELECT * FROM student;

ALTER

ALTER

- Used to change a database object

- ➢ Add columns
- ➢ Remove columns
- ➢ Add constraints
- ➢ Remove constraints
- ➢ Change default values
- ➢ Change column data types
- ➢ Rename columns
- ➢ Rename tables

ALTER TABLE (add column)

ALTER TABLE student ADD COLUMN fave_class text;

- can include constraints

ALTER TABLE products ADD COLUMN fave_class text CHECK (fave_class <> '');

ALTER TABLE (remove column)

ALTER TABLE products DROP COLUMN description;

- Recall that sometimes other columns might depend on the column we want to drop
- CASCADE will drop anything that depends on our dropped column as well

ALTER TABLE products DROP COLUMN description CASCADE;

ALTER TABLE (add constraint)

ALTER TABLE student ADD CHECK (name <> '');

- to set not null, use this syntax:

ALTER TABLE products ALTER COLUMN product_no SET NOT NULL;

ALTER TABLE (drop constraint)

ALTER TABLE products DROP CONSTRAINT some_name;

- to drop not null constraint, use this syntax:

ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL;

ALTER TABLE (set / remove default)

ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77;

ALTER TABLE products ALTER COLUMN price DROP DEFAULT;

ALTER TABLE (change column datatype)

ALTER TABLE products ALTER COLUMN price TYPE numeric(10,2);

ALTER TABLE (renaming)

- Rename column

ALTER TABLE products RENAME COLUMN product_no TO product_number;

- Rename table

ALTER TABLE products RENAME TO items;

https://www.postgresql.org/docs/current/ddl-alter.html#DDL-ALTER-ADDING-A-CONSTRAINT

DROP

DROP

- **DON’T DO IT**
  - unless you’re really really absolutely sure you want to
- Used to remove database object from the schema

DROP

Syntax:

DROP object object_name ;

Example:

DROP TABLE student;

TRUNCATE

TRUNCATE

- Used to removes the data, but keep the structure
- Faster than DROP
- Example:
    - TRUNCATE TABLE student;

Basic DML Commands

- INSERT
- UPDATE
- DELETE

INSERT

INSERT

- Fill columns from left to right with values given

INSERT INTO products VALUES (1, 'Cheese', 9.99);

- To specify columns to fill to (all others will get default):

INSERT INTO products (product_no, name) VALUES (1, 'Cheese');

https://www.postgresql.org/docs/current/dml-insert.html

INSERT

- Can insert defaults directly

INSERT INTO products (product_no, name, price) VALUES (1, 'Cheese', DEFAULT);

INSERT INTO products DEFAULT VALUES;

INSERT

- Can insert multiple rows at a time

```
INSERT INTO products (product_no, name, price) VALUES
  (1, 'Cheese', 9.99),
  (2, 'Bread', 1.99),
  (3, 'Milk', 2.99);
```

INSERT

- Can insert the result of a query

```
INSERT INTO products (product_no, name, price)
 SELECT product_no, name, price FROM new_products
  WHERE release_date = 'today';
```

UPDATE

UPDATE

To update existing rows, use the UPDATE command. This requires three pieces of information:

1. The name of the table and column to update
2. The new value of the column
3. Which row(s) to update

https://www.postgresql.org/docs/current/dml-update.html

UPDATE

UPDATE products SET price = 10 WHERE price = 5;

- Update all values in column:

UPDATE products SET price = price * 1.10;

- Update multiple columns

UPDATE mytable SET a = 5, b = 3, c = 1 WHERE a > 0;

DELETE

DELETE

- Similar syntax to UPDATE:

DELETE FROM products WHERE price = 10;

- Delete all columns:

DELETE FROM products;