BASES DE DATOS PARA DATA SCIENCE

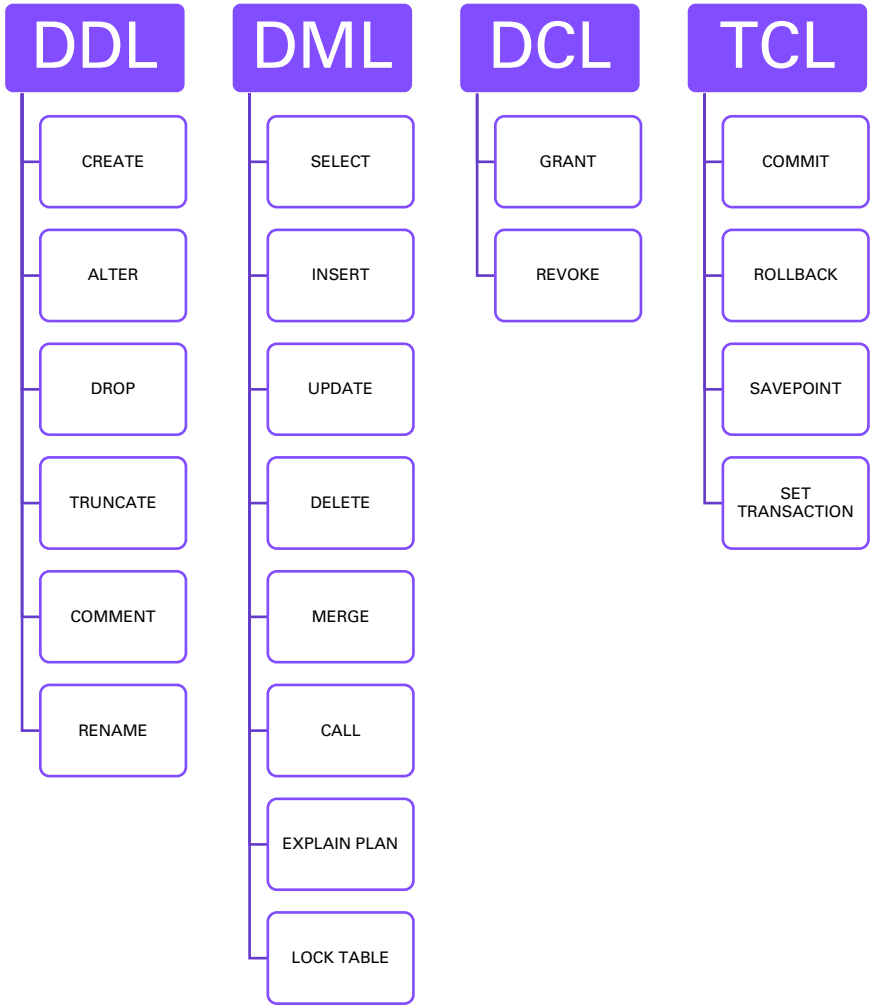
Lopez, Yoel
Pelli, Nahuel

LENGUAJE SQL

Tipos de SQL

En SQL podemos encontrar 4 categorías importantes:

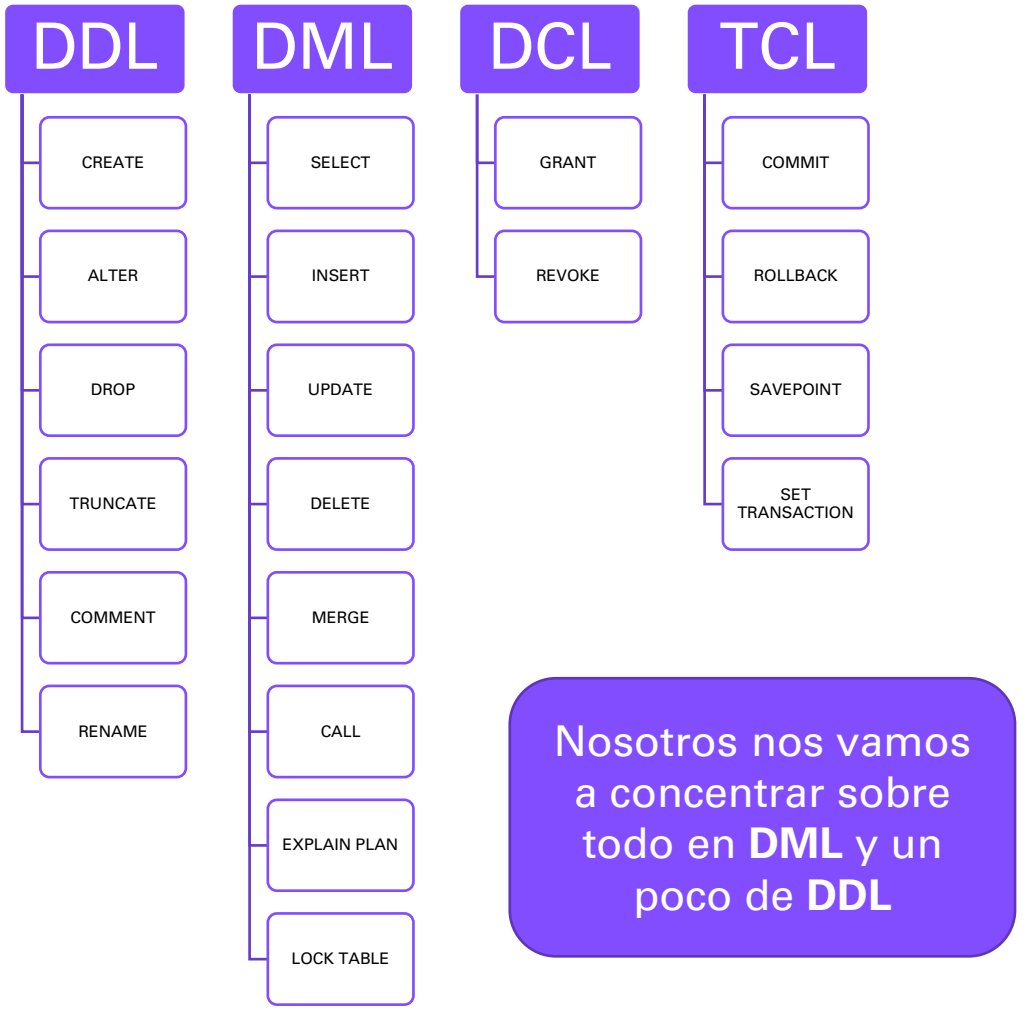
- ***DDL:*** Corresponde a los comandos que nos permiten *definir* la estructura de nuestro MR.
- ***DML:*** Estos comandos nos permiten manipular nuestro MR



Tipos de SQL

En SQL podemos encontrar 4 categorías importantes:

- ***DCL:*** Estos comandos rigen la parte de control y gobierno de nuestros datos
- ***TCL:*** Estos comandos rigen el control sobre las operaciones de DML.



Nosotros nos vamos a concentrar sobre todo en **DML** y un poco de **DDL**

DDL

No... no es dulce de leche

Creación de Elementos

- En PostgreSQL, tenemos que la sentencia **CREATE** nos permite crear distintos elementos en nuestra base de datos. Estos pueden ser:
  - **DATABASE**
  - **TABLE**
  - **INDEX**
  - **ROLE**
  - **USER**
  - ...

- `CREATE DATABASE testDB;`
-
```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```
-
```
CREATE TABLE TestTable AS
SELECT customername, contactname
FROM customers;
```
-
```
CREATE USER name [ [ WITH ] option [ ... ] ];
```

Name	Aliases	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [(*n*)]		fixed-length bit string
bit varying [(*n*)]	varbit [(*n*)]	variable-length bit string
boolean	bool	logical Boolean (true/false)
box		rectangular box on a plane
bytea		binary data (“byte array”)
character [(*n*)]	char [(*n*)]	fixed-length character string
character varying [(*n*)]	varchar [(*n*)]	variable-length character string
cidr		IPv4 or IPv6 network address
circle		circle on a plane
date		calendar date (year, month, day)
double precision	float8	double precision floating-point number (8 bytes)
inet		IPv4 or IPv6 host address
integer	int, int4	signed four-byte integer
interval [***fields***] [(*p*)]		time span
json		textual JSON data
jsonb		binary JSON data, decomposed
line		infinite line on a plane
lseg		line segment on a plane
macaddr		MAC (Media Access Control) address
macaddr8		MAC (Media Access Control) address (EUI-64 format)
money		currency amount
numeric [(*p*, *s*)]	decimal [(*p*, *s*)]	exact numeric of selectable precision
path		geometric path on a plane
pg_lsn		PostgreSQL Log Sequence Number
pg_snapshot		user-level transaction ID snapshot
point		geometric point on a plane
polygon		closed geometric path on a plane
real	float4	single precision floating-point number (4 bytes)
smallint	int2	signed two-byte integer
smallserial	serial2	autoincrementing two-byte integer
serial	serial4	autoincrementing four-byte integer
text		variable-length character string
time [(*p*)] [without time zone]		time of day (no time zone)
time [(*p*)] with time zone	timetz	time of day, including time zone
timestamp [(*p*)] [without time zone]		date and time (no time zone)
timestamp [(*p*)] with time zone	timestamptz	date and time, including time zone
tsquery		text search query
tsvector		text search document
txid_snapshot		user-level transaction ID snapshot (deprecated; see pg_snapshot)
uuid		universally unique identifier
xml		XML data

Creación de Elementos

- Usualmente, en nuestro rol de DS, MLE, etc. Vamos a estar utilizando creación sobre todo de tablas.
- Una parte importante de las tablas es el tipo de datos que contienen en sus columnas. Esto varia ligeramente entre motor y motor, en el caso de postgresSQL, los mas importantes son:
  - Numéricos (Int, Float, etc.)
  - Caracteres (char, varchar, text, …)
  - Fecha y tiempo (timestamp, date, ….)
  - Objetos (XML, json, …)
  - Sequencias autogeneradas (Serial, …)

Modificación de tablas

- En PostgreSQL, tenemos que la sentencia **ALTER** esta nos permite cambiar atributos de nuestro objeto, en particular **ALTER TABLE** nos permite operar sobre nuestras tablas:
  - Agregar columnas
  - Cambiar la definición de las mismas
  - Agregar o borrar *constraints* (relaciones)

- `ALTER TABLE tesTable ADD (column datatype [DEFAULT expr]);`
- `ALTER TABLE tesTable`
  `ALTER COLUMN columnName datatype;`
- `ALTER TABLE tesTable`
  `ALTER COLUMN columnName SET NOT NULL;`
- `ALTER TABLE tesTable`
  `ALTER CHECK ( col::expression )`
- `ALTER TABLE tesTable`
  `ALTER CONSTRAINT [ constraintName ];`
  `CHECK ( col::expression )`

```
ALTER TABLE tesTable
   ALTER CONSTRAINT [ relation_name ];
   [FOREIGN|PRIMARY] KEY (col.O )
   [[REFERENCES] otherTestTable (col.R)]
```

Ejemplo

```
CREATE SEQUENCE "theme_id_seq"
 INCREMENT 1
 MINVALUE 1
 MAXVALUE 2147483647
 START 1
 CACHE 1;
SELECT setval('"public"."theme_id_seq"', 1, TRUE);


CREATE TABLE theme(
    theme_id int DEFAULT nextval('theme_id_seq'::regclass) NOT
NULL,
    name varchar(256),
    parent_id int
    CONSTRAINT pk_themes PRIMARY KEY (theme_id)
);
ALTER TABLE themes ADD CONSTRAINT (fk_themes_id)
 FOREIGN KEY (parent_id) REFERENCES themes (theme_id)
```

Ejemplo

(ooootra opción)

```
CREATE TABLE theme(
    theme_id serial NOT NULL,
    name varchar(256),
    parent_id serial
    CONSTRAINT pk_themes PRIMARY KEY (theme_id)
);

ALTER TABLE themes ADD CONSTRAINT (fk_themes_id)
 FOREIGN KEY (parent_id) REFERENCES themes (theme_id)
```

DML

Consultando datos!

Insertando valores

- INSERT nos permite agregar filas en una tabla, es la única manera que nos provee SQL directamente para ingresar datos.

```
•INSERT INTO tableName[(col1[, col2,…])]
   VALUES (val1[, val2, …]);
```

```
•INSERT INTO tableName
   VALUES (val1, val2, … , val3);
```

```
•INSERT INTO themes(name,parent_id)
   VALUES (‘star wars death star’, 128);
•INSERT INTO sets VALUES
(1,‘dome’,1989,128,129);
```

DUDAS?

ENCUESTA