**Proceso de creación de la base de datos para el proyecto Polimeromics**

### Comprender los datos es el primer paso al comenzar a crear una tabla. El proyecto Polimeromics se concentra en tres bases de datos. Comprender los archivos fuente implica entender las columnas, tipos de datos y posibles valores nulos o inconsistencias, verificar si contiene encabezados claros y si todas las filas tienen el mismo número de columnas, revisar un subconjunto de datos para identificar valores extremos, datos corruptos o caracteres especiales. En la carpeta data están los datos utilizados en este proyecto y su transformación.

### En la carpeta scripts están los scripts que describen el proceso de limpieza y normalización de datos antes descrito puede verificar los procesos y su propósito. Ya limpios, procesados y normalizados tenemos tres archivos .csv:

* Para la data rcsb\_pdb: 'C:/Polimeromics/data/RCSB\_PDB/RCSB\_PDB\_cleaned\_and\_validated.csv'
* Para la data biogrid\_homosapiens:

'C:/Polimeromics/data/BIOGRID-CSV/cleaned\_blocks/merged\_blocks\_no\_duplicates\_processed\_normalized\_utf8.csv'

* Para la data polymers\_dataset:

'C:/Polimeromics/data/polymers\_dataset/polymers\_dataset\_reduced\_with\_metadata.csv'

El volumen de los datos de las tres bases de datos en total es de aproximadamente 339 MB distribuidos en tres archivos. Para la vinculación de los datos utilizare un proveedor de SQL que considero el más económico en este caso railway.com.

Luego de abrir una cuenta en <https://railway.com/> quien prestara el soporte SQL de este proyecto, debes crear una instancia para tu proyecto. Luego de creada la instancia, en la pestaña variables encontraras la información:

DATABASE\_PUBLIC\_URL:

postgresql://postgres:wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI@junction.proxy.rlwy.net:19339/railway

DATABASE\_URL:

postgresql://postgres:wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI@postgres.railway.internal:5432/railway

PGDATA:

/var/lib/postgresql/data/pgdata

PGDATABASE:

railway

PGHOST:

postgres.railway.internal

postgres.railway.internal

PGPASSWORD:

wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI

PGPORT:

5432

PGUSER:

postgres

POSTGRES\_DB:

railway

POSTGRES\_PASSWORD:

wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI

POSTGRES\_USER:

postgres

SSL\_CERT\_DAYS:

820

Con esta información podrás conectarte desde tu consola de **Powershell, Bash** o desde un administrador de base de datos como **pgAdmin 4.**

**Para una conexión desde la terminal usando psql:**

psql -U tu\_usuario -h tu\_host -p tu\_puerto -d tu\_base\_de\_datos

Sustituye tu\_usuario, tu\_host, tu\_puerto, y tu\_base\_de\_datos con las credenciales proporcionadas por Railway.

**Luego te pedirá tu contraseña:**

Suministra la contraseña proporcionada por Railway.

**Para una conexión desde la terminal de pgAdmin:**

Luego de descargada la aplicación puedes en la parte superior izquierda en la pestana “server” , presione “créate” y utiliza las credenciales proporcionadas por Railway.

**Creación de Tablas**

Luego es importante definir el propósito de la tabla, esto implica determinar cómo se usará la tabla, quién la consumirá y qué nivel de precisión es necesario para los tipos de datos. En este proyecto en particular la precisión es muy importante por ser un proyecto de bioinformática sin embargo limite la precisión del proyecto por los costos en almacenamiento que generaba un alto volumen de datos.

### ****2. Validar el archivo fuente****

* Usar herramientas como Python o Excel para:
  + Contar el número de columnas y filas.
  + Detectar valores nulos, duplicados y filas incompletas.
  + Analizar estadísticas básicas de cada columna (mínimo, máximo, promedio, etc.).
  + Normalizar los nombres de las columnas para que sean consistentes y SQL-friendly.

### ****3. Crear un plan de diseño de la tabla****

* **Definir los tipos de datos**:
  + Seleccionar tipos de datos adecuados para cada columna:
    - INTEGER: Para valores numéricos enteros.
    - FLOAT o DECIMAL: Para números con decimales.
    - VARCHAR o TEXT: Para texto.
    - DATE o TIMESTAMP: Para datos temporales.
  + Considerar el rango esperado de valores y el rendimiento (por ejemplo, usar SMALLINT en lugar de INTEGER si los valores son pequeños).
* **Definir restricciones**:
  + Establecer restricciones de integridad para garantizar la calidad de los datos:
    - NOT NULL: Para columnas que deben tener siempre un valor.
    - UNIQUE: Para columnas que no deben tener valores duplicados.
    - CHECK: Para validar rangos o condiciones específicas.
    - DEFAULT: Para columnas con valores predeterminados.
* **Definir claves primarias y foráneas**:
  + Especificar una clave primaria (PRIMARY KEY) para garantizar la unicidad de cada fila.
  + Identificar relaciones con otras tablas mediante claves foráneas (FOREIGN KEY).

Conexión desde la terminal usando psql:

psql -h junction.proxy.rlwy.net -p 19339 -U postgres -d railway

password:

wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI

$env:PGPASSWORD = "wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI"

psql -h junction.proxy.rlwy.net -p 19339 -U postgres -d railway -f "C:/Polimeromics/data/RCSB\_PDB/rcsb\_pdb\_create\_table.sql"

$env:PGPASSWORD = "wDRvQVpvgyyloKpjkUJlkjnNvdsUKbOI"

psql -h junction.proxy.rlwy.net -p 19339 -U postgres -d railway -f "C:/Polimeromics/data/polymers\_dataset/Tables/polymers\_dataset.sql"

Tabla BIOGRID:

CREATE TABLE IF NOT EXISTS biogrid\_homosapiens (

identifier\_id TEXT,

identifier\_type TEXT,

official\_symbol TEXT,

aliases TEXT,

organism\_official TEXT,

score\_1 FLOAT,

score\_2 FLOAT,

hit TEXT,

unique\_id INTEGER

);

**Tabla**

CREATE TABLE IF NOT EXISTS rcsb\_pdb (

entry\_id SERIAL PRIMARY KEY,

experimental\_method TEXT,

matthews\_coefficient FLOAT,

percent\_solvent\_content FLOAT,

crystallization\_method TEXT,

ph FLOAT,

crystal\_growth\_procedure TEXT,

temp\_k FLOAT,

deposition\_date TEXT,

release\_date TEXT,

number\_of\_non\_hydrogen\_atoms FLOAT,

total\_polymer\_instances FLOAT,

total\_polymer\_residues FLOAT,

number\_of\_water\_molecules FLOAT,

disulfide\_bond\_count FLOAT,

molecular\_weight FLOAT,

number\_of\_distinct\_protein\_entities FLOAT,

refinement\_resolution FLOAT,

structure\_determination\_methodology TEXT,

average\_b\_factor FLOAT,

r\_free FLOAT,

r\_work FLOAT,

structure\_title TEXT,

sequence TEXT,

entity\_polymer\_type TEXT,

polymer\_entity\_sequence\_length FLOAT,

entity\_macromolecule\_type TEXT,

total\_polymer\_entity\_instances FLOAT,

molecular\_weight\_entity FLOAT,

macromolecule\_name TEXT,

ec\_number TEXT,

ec\_provenance\_source TEXT,

source\_organism TEXT,

taxonomy\_id FLOAT,

total\_polymer\_residues\_assembly FLOAT,

total\_polymer\_instances\_assembly FLOAT,

oligomeric\_count FLOAT,

oligomeric\_state TEXT,

stoichiometry TEXT,

ligand\_id TEXT,

ligand\_formula TEXT,

ligand\_mw FLOAT,

ligand\_name TEXT,

inchi TEXT

);

Para vincular las tres tablas en PostgreSQL, se necesita establecer relaciones entre ellas mediante

clavesprimarias (primary keys) y claves foráneas (foreign keys). Esto implica identificar los campos que

tienen valores relacionados entre las tablas.

Para la data: biogrid\_homosapiens : unique\_id INTEGER

Para la data: polymers\_dataset\_table; unique\_id INT PRIMARY KEY

Para la data: rcsb\_pdb; entry\_id TEXT

**Estableciendo Relaciones con Claves Foráneas**

**Paso 1: Vincular biogrid\_homosapiens y polymers\_dataset\_table**

Si polymers\_dataset\_table.unique\_id está relacionado con biogrid\_homosapiens.unique\_id, creamos una clave foránea:

Esto asegura que:

* Cada unique\_id en polymers\_dataset\_table debe existir en biogrid\_homosapiens.
* Si eliminas una fila en biogrid\_homosapiens, las filas relacionadas en polymers\_dataset\_table también se eliminarán.