**UNIVERSIDAD VERACRUZANA**

**FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**

**BOCA DEL RÍO, VERACRUZ**

**PROGRAMA EDUCATIVO**

**INGENIERÍA MECATRÓNICA**

**EXPERIENCIA EDUCATIVA**

**TÓPICOS AVANZADOS DE INFORMÁTICA**

**DOCENTE**

**M. YULIANA BERUMEN DÍAZ**

**PROYECTO**

**THE HUNGER GAMES**

**ALUMNO**

**INGRID MELISSA MORA HERNÁNDEZ**

**MATRÍCULA**

**S19003064**

**FECHA**

**14 DE JUNIO DE 2023**

Durante la experiencia educativa de Tópicos Avanzados de Informática, adquirimos conocimientos fundamentales orientados al ámbito de la Ingeniería de Software y la Gestión de Bases de Datos. A lo largo del curso, exploramos diversos temas, como lo fueron el Modelado Orientado a Objetos, la estructura de los Diagramas Entidad-Relación y Diagramas de Clase, los cuales nos permitieron adquirir una visión profunda y entendimiento de los elementos que les conforman, tales como las entidades, atributos, relaciones y la cardinalidad.

Además, nos adentramos en la sintaxis básica del lenguaje SQL, el cual es un lenguaje enfocado en la creación y manipulación de bases de datos y utilizamos el sistema de gestión PostgreSQL, para implementación de éstas.

Como parte integral de nuestro aprendizaje, a manera de cierre “con broche de oro”, se nos encomendó la realización de un proyecto con el objetivo de poner en práctica y aplicar todo lo aprendido durante el semestre. Cada uno de nosotros recibió un planteamiento de problema único, el cual requería el diseño y la implementación de una base de datos utilizando PostgreSQL como sistema de gestión. En adición a esto, se solicitó la creación de una interfaz que permitiera la manipulación eficiente de la base de datos en cuestión. Para este propósito, empleamos Java, con apoyo de alguno de los IDEs recomendados como NetBeans, Eclipse o IntelliJ IDEA.

El objetivo principal de este proyecto es demostrar nuestra habilidad para el desarrollo de soluciones efectivas y eficientes en el ámbito de la gestión de bases de datos y la interfaz de usuario. El presente documento busca presentar un reporte detallado de los procesos y análisis que se llevaron a cabo para el correcto desarrollo y cumplimiento de lo anteriormente mencionado. A lo largo de éste, presentaré el enfoque adoptado, los desafíos encontrados y las soluciones implementadas por mí para la creación de la base de datos y la interfaz correspondientes. Este proyecto final lo vi como una oportunidad para consolidar los conocimientos teóricos adquiridos y aplicarlos en un escenario práctico, así como retomar brevemente mis orígenes como programadora retirada. Así que de antemano quiero externar mi gratitud a usted profesora por toda la experiencia de este semestre, se la estima y aprecia infinitamente ♡.

El problema que se me fue asignado fue el siguiente:

*Juegos del Hambre*

*“Debido a la organización de los próximos juegos del hambre, el presidente de Panem, Coriolanus Snow, decidió que se debe crear un sistema de información a fin de realizar la gestión de las pruebas de entrenamiento que los tributos realizan antes de entrar a la arena. Del análisis realizado por Seneca Crane, el vigilante en jefe, se obtuvo la siguiente información:*

* *El entrenamiento se compone de una serie de pruebas, en cada una de las cuales intervienen los tributos.*
* *Las pruebas son evaluadas por los vigilantes del Capitolio, siendo 1 la puntuación más baja y 12 la más alta.*
* *De cada tributo se desea guardar su CURP, nombre, sexo, edad, habilidad, puntuación obtenida en el espectáculo brindado a los vigilantes del Capitolio y el distrito al que pertenece.*
* *Cada tributo tiene un mentor, del mentor se desea guardar su CURP, nombre, sexo, edad, juego en el que resultó ganador y distrito al que pertenece.*
* *De cada vigilante se desea guardar su CURP, nombre, sexo, edad, puesto y/o especialidad.*
* *Cada Distrito se identifica por un nombre, de ellos se desea guardar en qué se especializa, puestos de trabajo, números de Juegos del Hambre ganados, tributos, cantidad de habitantes, ubicación, clima, porcentaje de hombres y mujeres y nombre de su líder.*
* *Un tributo puede pertenecer solamente a un distrito. A un Distrito pueden pertenecer muchos tributos.*
* *Los distritos son controlados por el Capitolio.*
* *Del Capitolio se desea guardar el nombre del presidente, número de habitantes, ubicación, clima, porcentaje de hombres y de mujeres, y lugares de interés.*
* *Dentro del sistema cada prueba se debe identificar a partir de un código y un nombre, se desea saber también su tipo y grado de dificultad (para esto se utilizará una escala de colores: amarillo, naranja, rojo y negro). Así también se debe registrar el nombre del participante vencedor y el tiempo empleado por este.*
* *La realización de las pruebas se desarrollará a lo largo de varias jornadas, los tributos podrán competir en varias pruebas. Para cada participante en una prueba se deben registrar la fecha en la que participa, el tiempo que le tomó llevarlo a cabo y su puntuación.”*

Lo primero que se realizó fue el diagrama Entidad-Relación, el cual en realidad formó parte de una de las primeras tareas realizadas durante el curso. Este, sin yo saberlo en su momento, me sería de utilidad para este proyecto.

Interfaz de usuario gráfica

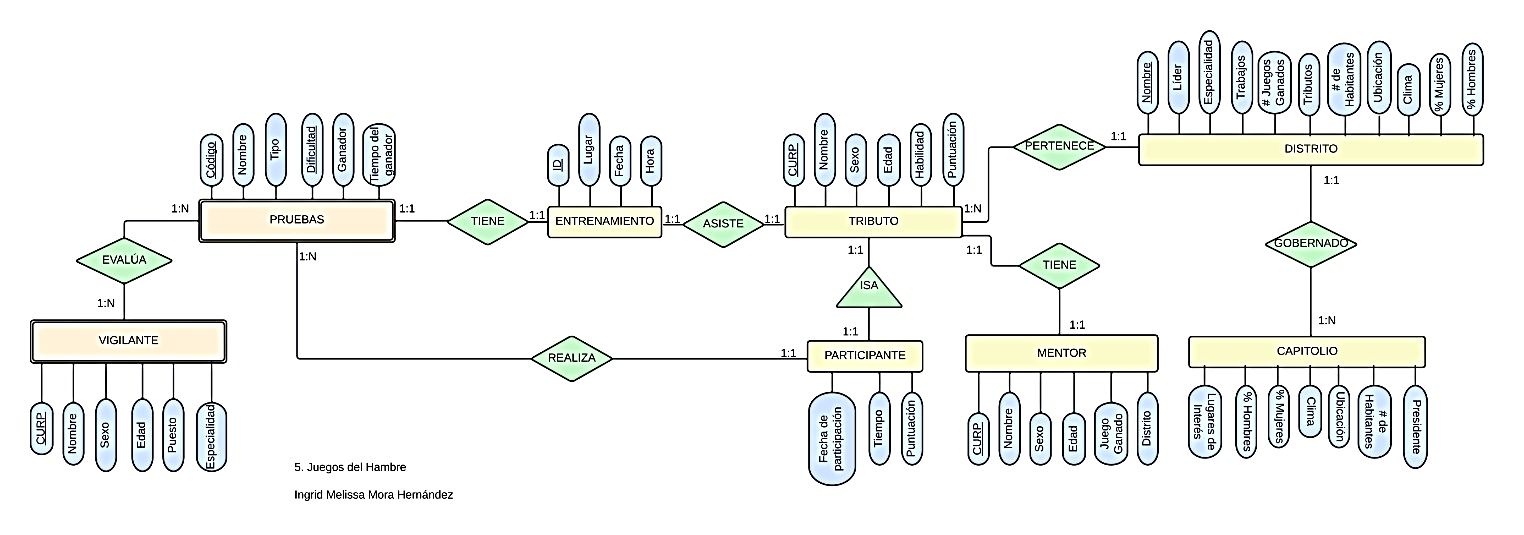
Descripción generada automáticamente

Ilustración 1. Diagrama Entidad-Relación

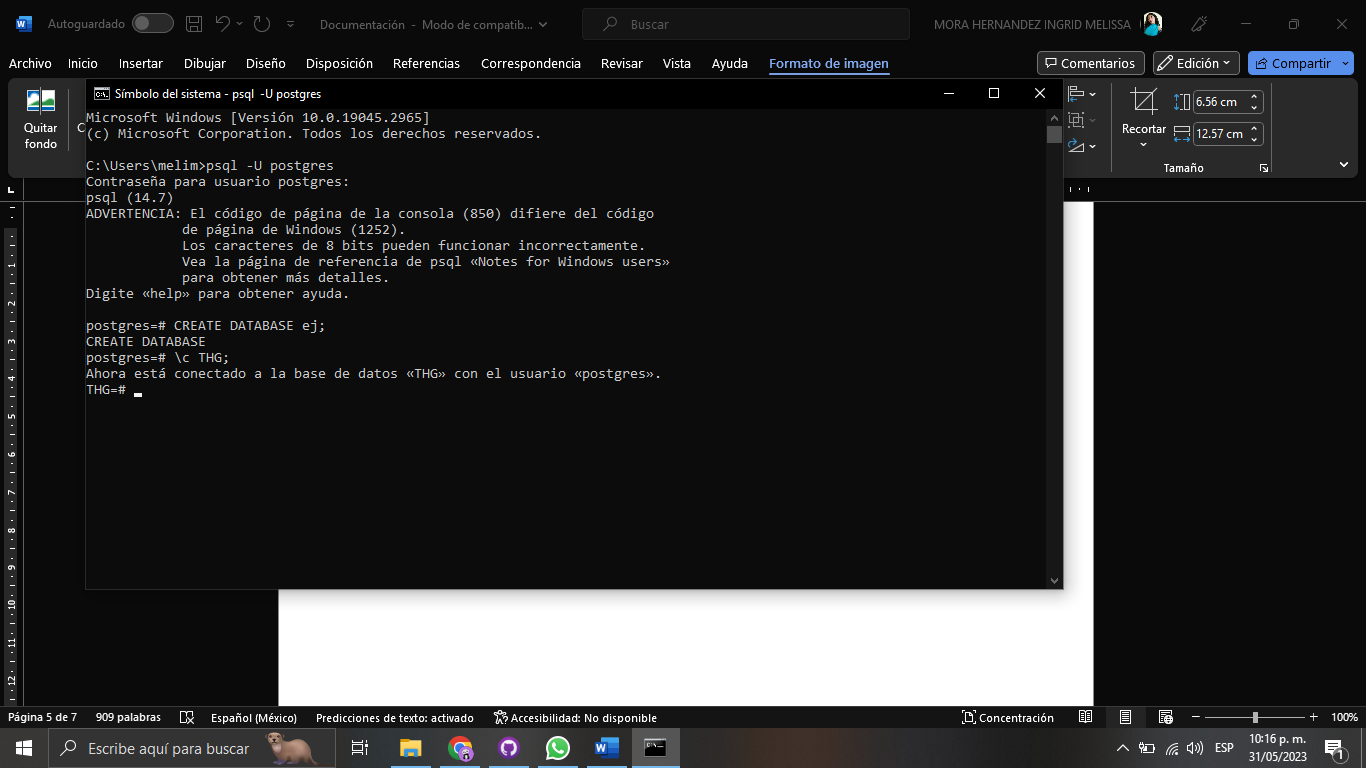
Como siguiente avance, se realizó la conversión de dicho Diagrama Entidad-Relación a su forma de Diagrama de Clases.

Diagrama

Descripción generada automáticamente

Ilustración 2. Diagrama de Clases

Y es en base a este último es que se comenzó a realizar la base de datos de PostgreSQL. Opté por realizar el proceso de la creación de la base de datos directamente en la ventana de comandos por motivos de familiaridad y comodidad.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 3. Creación de la BD en CMD

Posteriormente creé cada una de mis tablas con sus respectivos atributos directo en PostgreSQL:

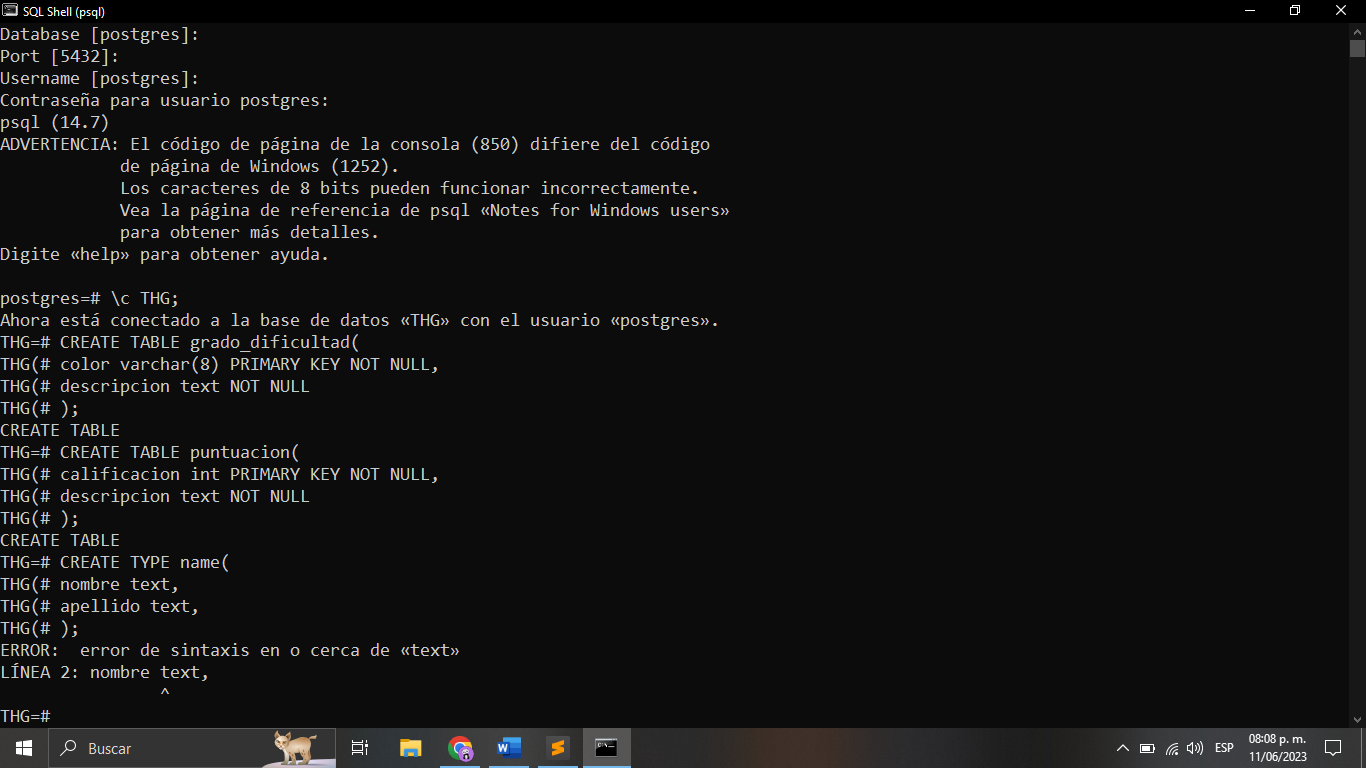


Ilustración 4. Creación de las tablas Grado\_Dificultad y Puntuación

El motivo por el que decidí comenzar por crear estas dos tablas es que, si nos fijamos en el diagrama de clases (*Ilustración 2*), notamos que estas no poseen ningún tipo de dependencia o relación con otras tablas, por lo que su creación e inserción de datos nos representarían ningún efecto colateral negativo para la BD.

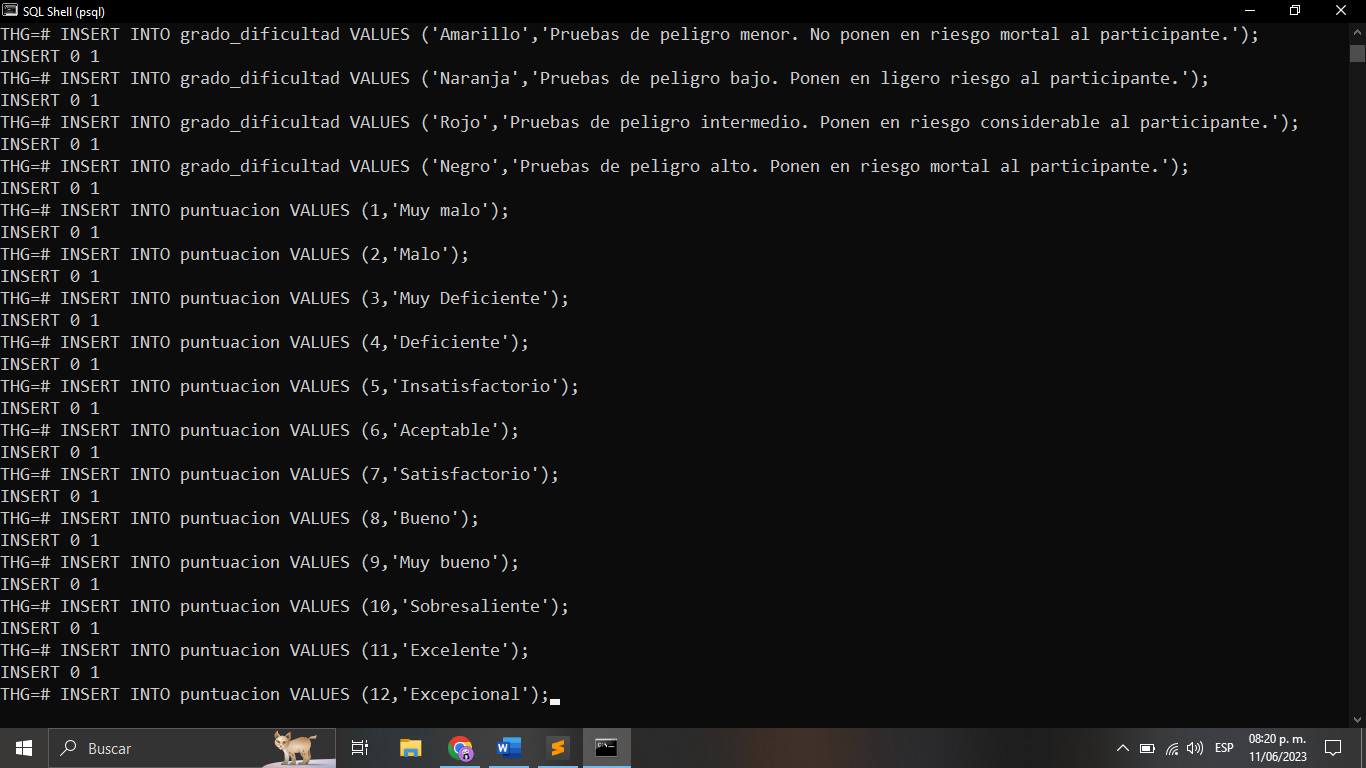


Ilustración 5. Inserción de datos a las tablas Grado\_Dificultad y Puntuación

Creé un *type* llamado **name**, para emplearlo como un tipo de dato “ideal” para las columnas que requieran de un campo de tipo Nombre. También creé la tabla **Persona** que, por el tipo de enfoque que di a mi BD, funciona como una tabla *“padre”* que heredará sus columnas a tablas como **Líder**, **Mentor**, **Tributo** y **Vigilante** con ayuda del comando *INHERITS.*

Captura de pantalla de computadora

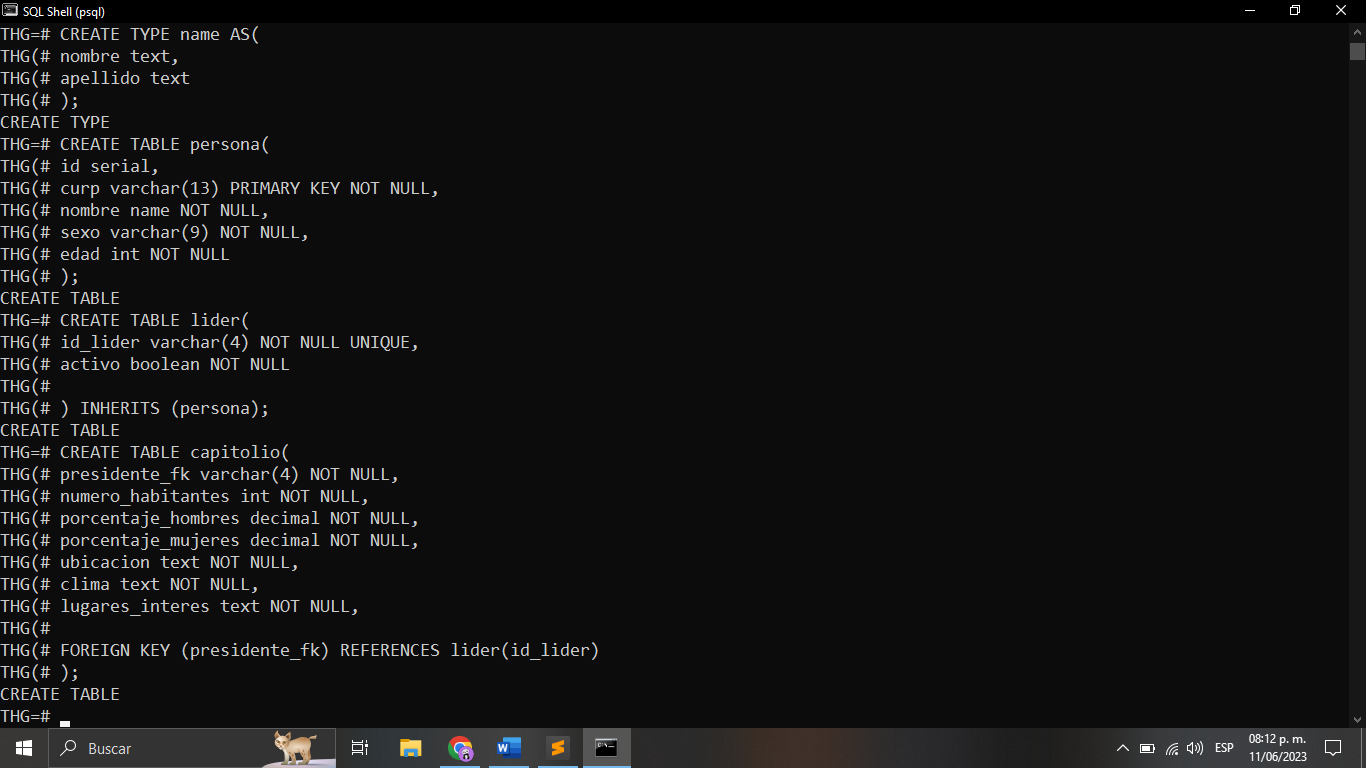
Descripción generada automáticamente

Ilustración 6. Creación del type Name y las tablas Persona y Líder y Capitolio

La tabla **Persona** es a mi parecer, la tabla más importante de toda la BD, y es que es aquella con la que más adelante estableceré conexión en Netbeans para su interacción mediante la creación de nuestra Interfaz Gráfica de Java. A esta tabla también se le asignó una columna llama ID de tipo *serial*, con el propósito de que este atributo nos sea de utilidad más adelante. Por su parte, a la tabla **Líder** se le asignó una columna llama Activo de tipo *boolean* que, de igual manera nos será de utilidad más tarde.

Algo a destacar es que, todos los *inserts* que se realicen en cualquiera de las tablas hijas se verán reflejados también en la tabla **Persona**, sólo que sin mostrar los atributos exclusivos de las tablas hijas. Por ejemplo, si consultamos los *inserts* de la tabla **Líder** desde la tabla **Persona**:

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 7. Consulta de registros en la tabla Persona con el comando SELECT \* FROM

Posteriormente creé la tabla **Capitolio**, en la cuál observamos nuestra primer *foreign key* que establece relación con nuestra tabla **Líder**, y es que el primer registro de nuestra tabla Líder le corresponde a Coriolanus Snow, quien a su vez es el presidente del Capitolio. A manera de corroborar si la relación fue establecida de manera correcta, hacemos uso del comando *JOIN* y validamos:

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 8. Uso del comando JOIN para validar la relación entre las tablas Líder y Capitolio

Creé las tablas **Distrito**, **Mentor** y **Tributo** y agregué sus respectivos *inserts*, me parece importante recalcar que el orden en que realicé la creación e inserciones de dichas tablas fue hecho siguiendo la línea de relación y es que, para que la tabla **Distrito** fuera creada, primeramente debía existir **Líder**, puesto que de ella se obtenía la relación definida en su *foreign key*, de igual manera ocurría con la tabla **Mentor** que requería de **Distrito** y así sucesivamente, con **Tributo** que requería de **Distrito** y **Mentor**.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 9. Creación de las tablas Distrito, Mentor y Tributo

Ahora bien, uno de los atributos que se nos pedían para la tabla **Distrito** era que existiera un registro de los tributos que le pertenecían, sin embargo, en primera instancia esta relación no podía ser creada ya que en ese momento aún no existía una tabla **Tributo** a la que referenciar, así que lo siguiente que hice fue hacer uso del comando *ALTER TABLE* para añadir dichas columnas a mi tabla **Distrito** y del comando *UPDATE*para asignarles valores a esos nuevos campos.

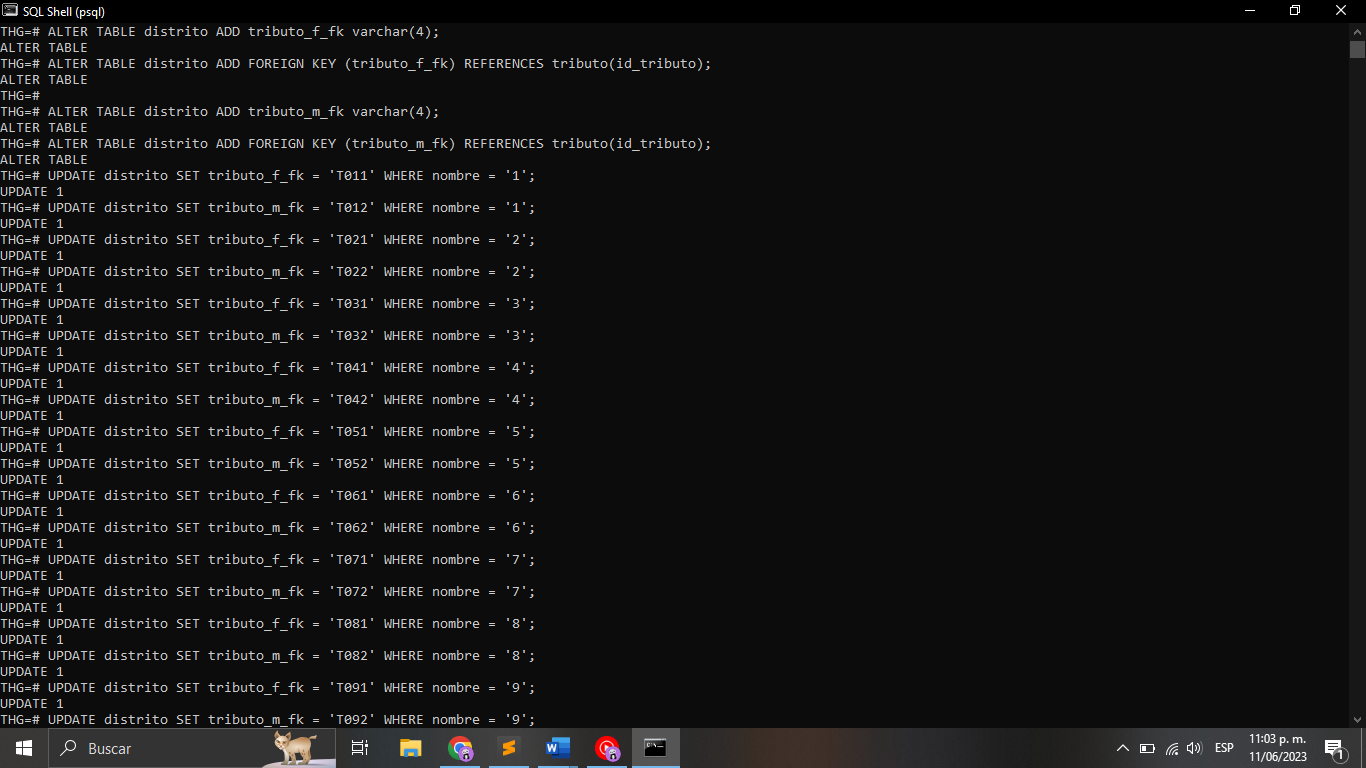


Ilustración 10. ALTER TABLE y UPDATE de la tabla Distrito

Añadí el *CONSTRAINT UNIQUE* a la columna ID de la tabla **Persona** porque me percaté que no había hecho esto anteriormente. Esto con el objeto de evitar ID repetidos que nos pudieran generar problemas más adelante a la hora de interactuar con la Interfaz. También creé las tablas **Vigilante**, **Prueba** y **Participante**, siendo esta última una “tabla relación” que conecta a la tabla **Tributo** con la tabla **Prueba**, esto se hace ya que, al ser una relación de muchos a muchos, se requiere de una tabla para esto.

Y de por último y de manera similar a cómo ocurría con las columnas Tributo\_F y Tributo\_M de la tabla **Distrito**, en la tabla **Prueba** requeríamos que existiera un registro del vencedor de dicha prueba, sin embargo, esta relación no podía ser creada si aún no existía una tabla **Participante** a la cual referenciar, así que nuevamente hice fue hacer uso de los comandos *ALTER TABLE* y *UPDATE*para realizar las modificaciones necesarias a la tabla **Prueba**.

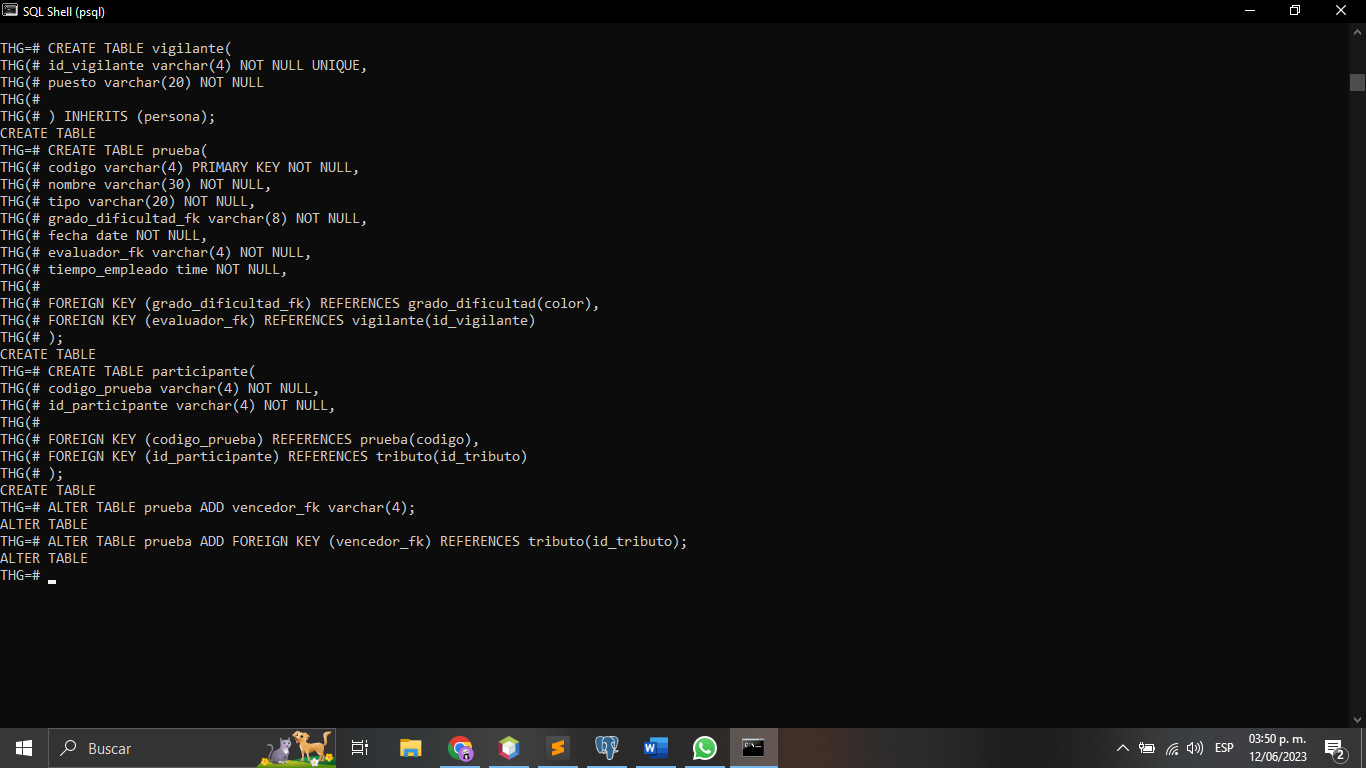
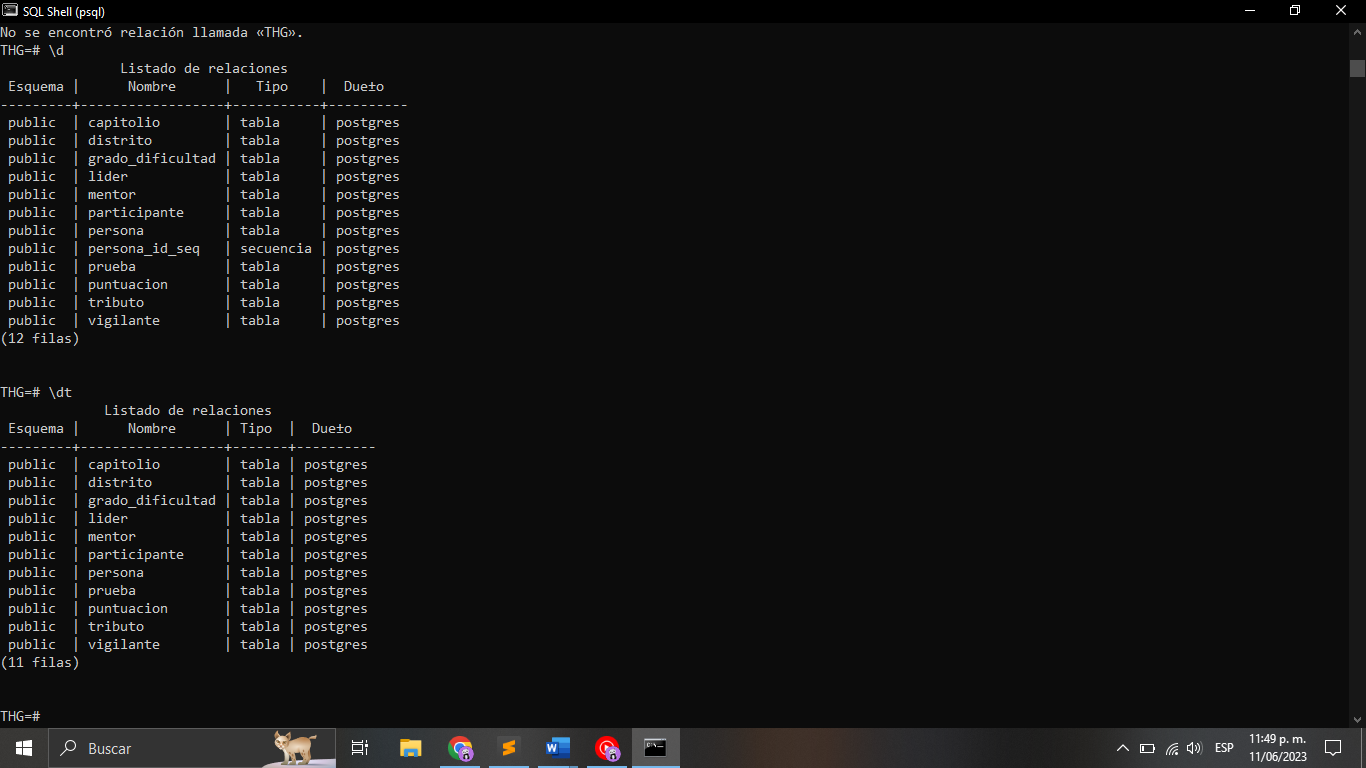


Ilustración 11. Creación de las tablas Vigilante, Prueba y Participante

Con esto finalizaría con la creación de mi Base de Datos, ya sólo para comprobar inspeccioné que todas las tablas fueran creadas correctamente:



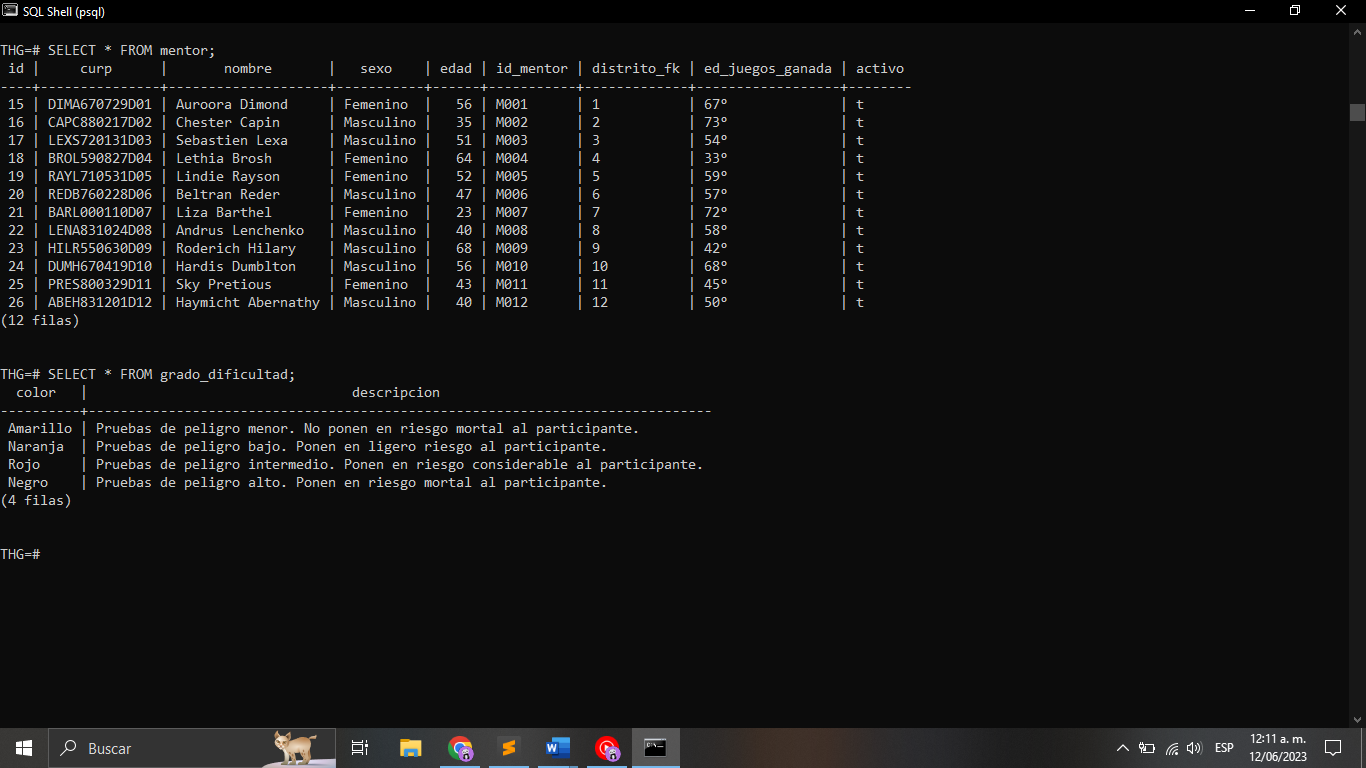
Añadí el atributo activo boolean NOT NULL para todas las tablas hijas de persona para que cuando se elimine el registro en la tabla padre, el estatus cambié en su correspondiente hijo.

Pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente



Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente