RETO GRUPAL

ESTADOS DE ESTADOS UNIDOS

ESCUELA DE TALENTO | FUNDACIÓN NTTDATA

ÁREA II:
BASES DE DATOS

METODOLOGÍA

En este documento presentamos nuestro trabajo correspondiente a la resolución de los ejercicios del Área II - Reto Grupal “Estados de Estados Unidos”.

Para la realización de las tareas descritas en el reto hemos realizado videoconferencias para realizar los ejercicios de forma colaborativa y compartir nuestras propuestas. Asimismo, hemos utilizado plataformas para compartir archivos y un grupo de Whatsapp para que en todo momento todos los integrantes pudiéramos tener acceso a la evolución del trabajo, aportar nuestras opiniones, anotar sugerencias etc, cuando no estábamos conectados.

El proceso ha sido documentado por Ana, quien se ha encargado de la edición de texto e imagen, y ha sido revisado y aprobado por el resto de los componentes del equipo.

Cabe destacar que Luisa, a parte de colaborar con los ejercicios de este reto, planteó las correcciones de código correspondientes al Reto del Área I y subsanó los errores.

Gracias a nuestra profesora Cristina por las clases y aclaraciones en los primeros momentos del reto.

PARTICIPANTES

- JÉSSICA RÍOS
- LUISA ROMERO
- ANA DELGADO
- JAIME SHIMOHIRA
- JESÉ MUÑOZ
- SERGIO BULBARELA POPO

ÍNDICE

EJERCICIO 1

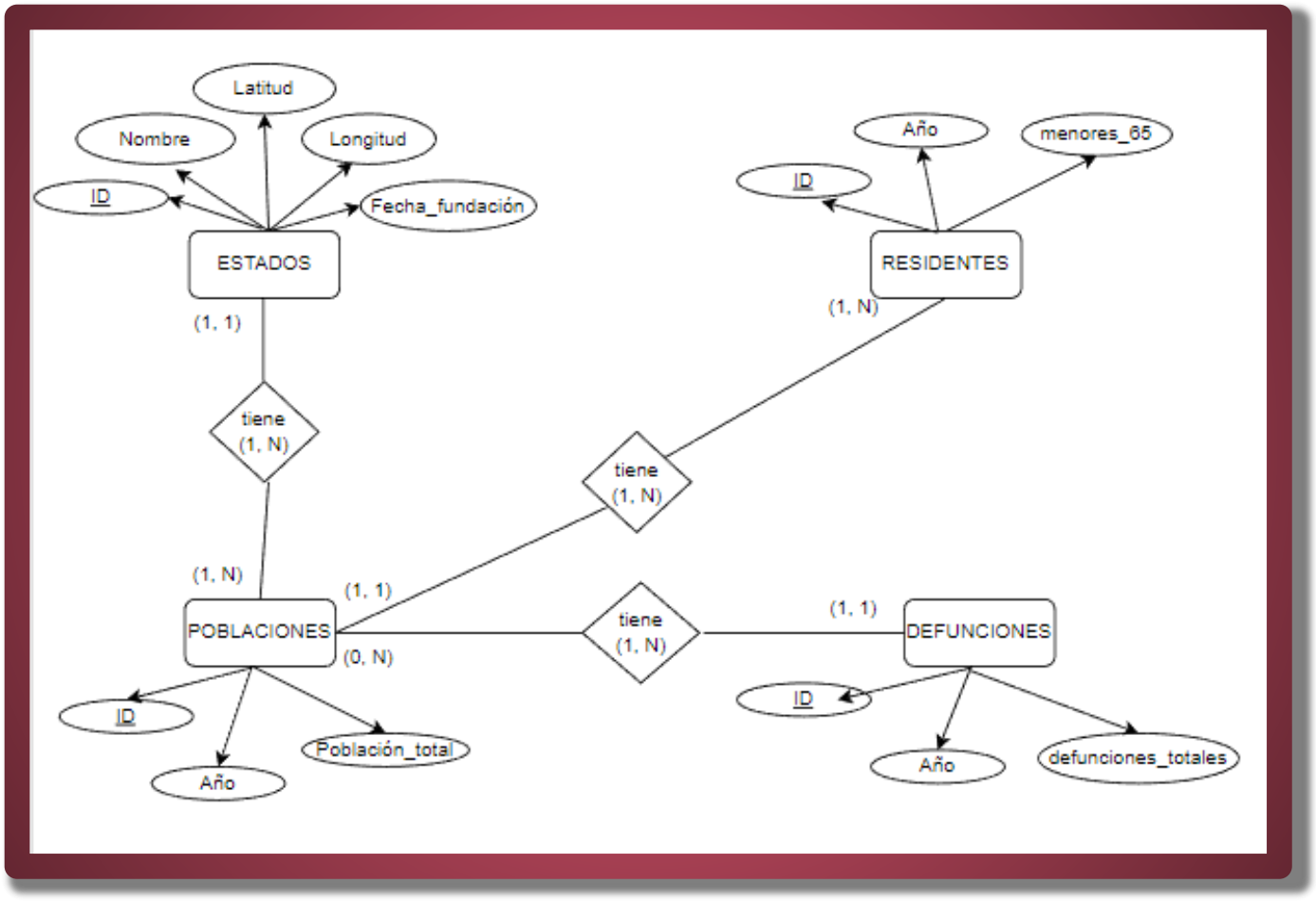
Durante el proceso de creación de los diagramas de entidad – relación, nos surgieron dudas y realizamos cambios constantes, ya que, por los datos que se facilitaban, considerábamos que se podría realizar el ejercicio con una o dos tablas únicamente. No obstante, nuestras propuestas se han basado en cuatro tablas, ya que así se solicitaba en ejercicios posteriores.

Todo el proceso de creación de diagramas se hizo en una reunión conjunta en la que participamos todos los miembros del equipo, tras una primera lluvia de ideas, nos centramos en que la tabla Estados debe ser la tabla principal, no obstante hubo distintas propuestas.

Para proceder con la creación de las tablas y diagramas, Jéssica compartió pantalla para poder ir aplicando de forma síncrona tanto las diferentes propuestas como la decisión definitiva.

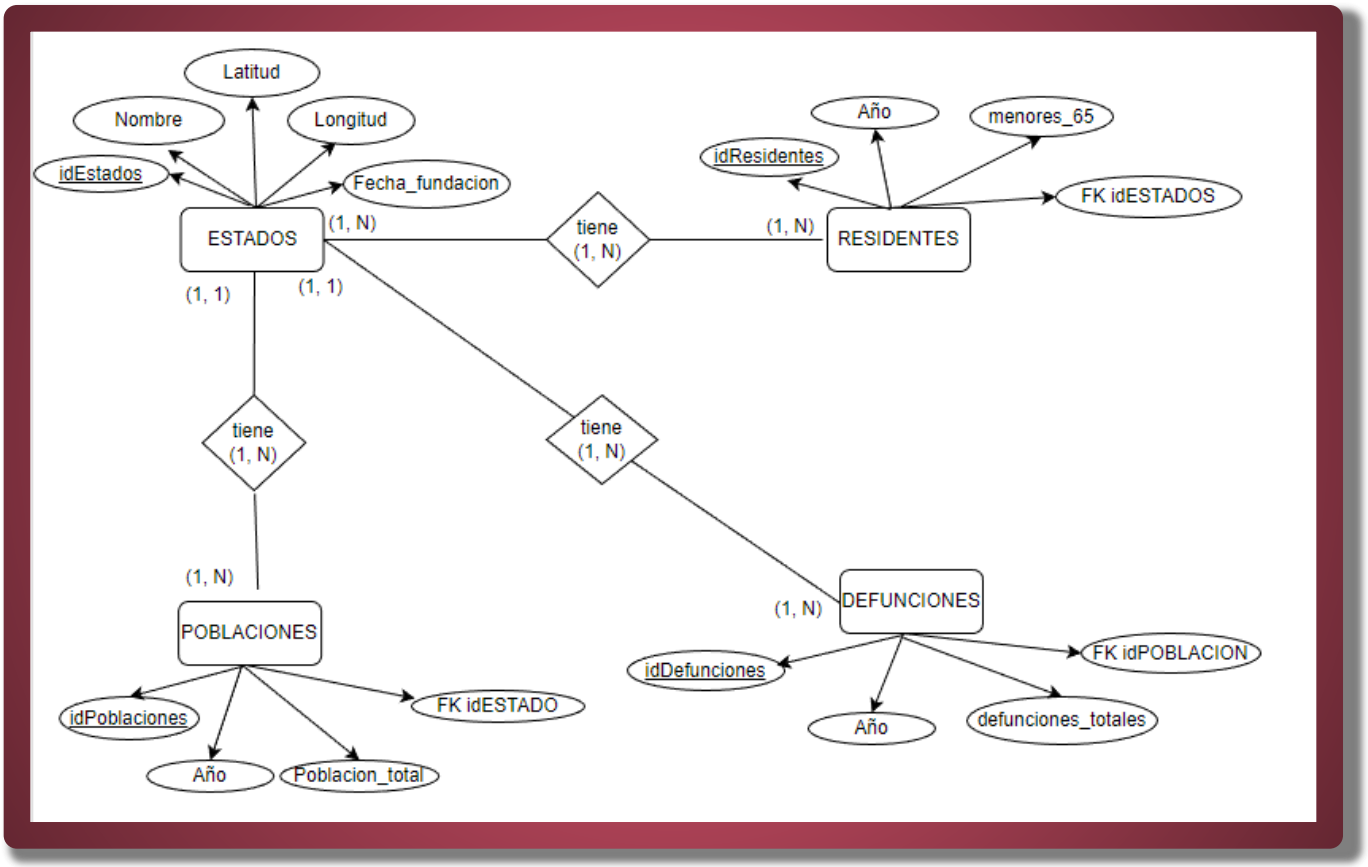
En la primera propuesta, se crearon cuatro tablas: Estados, Poblaciones, Residentes y Defunciones, de manera que la tabla Estados y la tabla Poblaciones se relacionaban entre sí, y las tablas Residentes y Defunciones se relacionaban con la tabla Poblaciones, según la propuesta realizada por Jesé.

En este diagrama podemos apreciar las tablas con los atributos y las relaciones que establecimos entre ellas, de manera que desde la tabla Poblaciones se establecía relación con la tabla Estados, con la tabla Residentes y la tabla Defunciones.



Se establecieron los atributos Ed, Año y población_total para la tabla Población, Id, Año y defunciones_totales para la tabla defunciones, Id, Año y menores_65 para la tabla Residentes y finalmente, para la tabla Estados, Id, Nombre, Longitud, Latitud y fecha_fundación

Durante el debate que se generó, surgió otra idea por parte de Ana y Jaime, que dio lugar a la propuesta final. Para el nuevo diagrama cambiamos la relación entre las tablas de manera que las tablas Residentes y Defunciones se relacionen con la tabla Estados en vez de con la tabla Poblaciones, ya que consideramos Estados la tabla principal.



De esta manera se puede interpretar que un estado puede tener varios residentes, pero los residentes solo pertenecen a un estado concreto, un estado tiene distintas poblaciones, pero la población en sí solo puede pertenecer a un estado, y lo mismo ocurre con las defunciones.

La idea final es consensuada entre todos los miembros del equipo, y determinamos de esa manera las tablas en su forma final.

EJERCICIO 2

Creamos por palabras la estructura de los diagramar ER anteriores. Dado que finalmente estuvimos valorando dos propuestas, hemos decidido plasmar aquí la estructura de ambos diagramas:

Primer planteamiento, desarrollado por Sergio, Jaime y Jesé.

En el primer planteamiento las tablas Estados, Residentes, Poblaciones y Defunciones tendrían los siguientes atributos:

- ESTADOS: ID (PK), Nombre, Latitud, Longitud, Fecha_fundación.
- POBLACIONES: ID (PK), Año, Población_total.
- RESIDENTES: ID (PK), Año, Menores_65.
- DEFUNCIONES: ID (PK), Año, Defunciones_totales.

Las relaciones quedarían explicadas de la siguiente manera:

- La relación de ESTADOS con POBLACIONES tiene cardinalidad de (1, N), es decir, un Estado puede tener muchas poblaciones, aunque cada población sólo podrá pertenecer a un Estado específico.
- La relación de POBLACIONES con RESIDENTES tiene cardinalidad de (1, N), ya que entendemos que una población podrá tener muchos residentes, pero estos residentes sólo podrán pertenecer a una única población.
- POBLACIONES se relaciona también con DEFUNCIONES con una cardinalidad de (1, N), ya que una población podrá tener muchas defunciones y, a su vez, las defunciones pertenecerán a una población en específico.

Planteamiento definitivo, desarrollado por Ana y Jéssica.

En el planteamiento que se usó en el diagrama definitivo tenemos las tablas anteriores: Estados, Residentes, Poblaciones y Defunciones, con los siguientes atributos:

- ESTADOS: ID (PK), Nombre, Latitud, Longitud, Fecha_fundación.
- POBLACIONES: ID (PK), Año, Poblacion_total, idEstado (FK).
- RESIDENTES: ID (PK), Año, Menores_65, idEstado (FK).
- DEFUNCIONES: ID (PK), Año, Defunciones_totales, idEstado (FK).

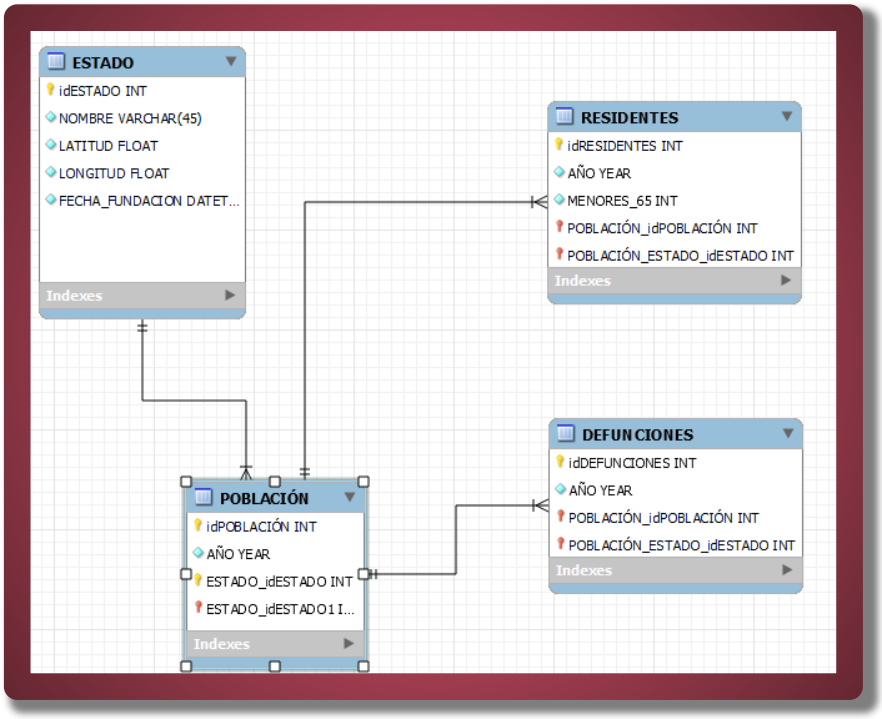
En cuanto a relaciones se realizan los siguientes cambios:

- La relación de ESTADOS con POBLACIONES tiene cardinalidad de (1, N), es decir, un Estado puede tener muchas poblaciones, aunque cada población sólo podrá pertenecer a un Estado específico.
- La relación de ESTADOS con RESIDENTES tiene cardinalidad de (1, N), es decir, un Estado puede tener muchos residentes, pero a su vez cada residente sólo podrá pertenecer a un Estado específico.
- La relación de ESTADOS con DEFUNCIONES tiene cardinalidad de (1, N), es decir, un Estado puede tener muchas defunciones entre sus habitantes, aunque cada defunción sólo podrá contabilizarse en un Estado específico.

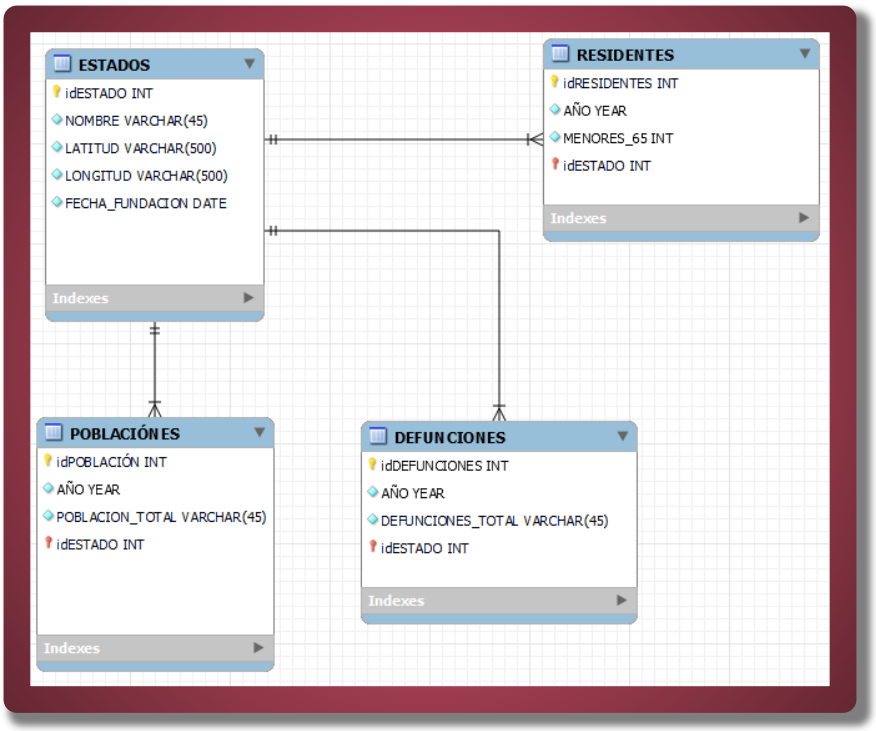
EJERCICIO 3

Para este ejercicio creamos las tablas en Workbench, mediante trabajo colaborativo en videollamada en el que de nuevo Jéssica compartió pantalla y fue siguiendo los pasos. Fueron creadas en base a los diagramas anteriores, en los que se barajaron principalmente dos opciones, que son las siguientes:

Primera opción de modelado sugerida por Luisa;



Opción de modelado definitiva, sugerida por Jaime



EJERCICIO 4

Para poder seguir el proceso que seguimos en este ejercicio, hemos decidido explicarlo enumerando las distintas partes:

-Query y base de datos relacional con SQL en Python
-Base de datos en PHPMYADMIN
-Base de datos no relacional en MongoDB

QUERY Y BASE DE DATOS RELACIONAL CON SQL EN PYTHON

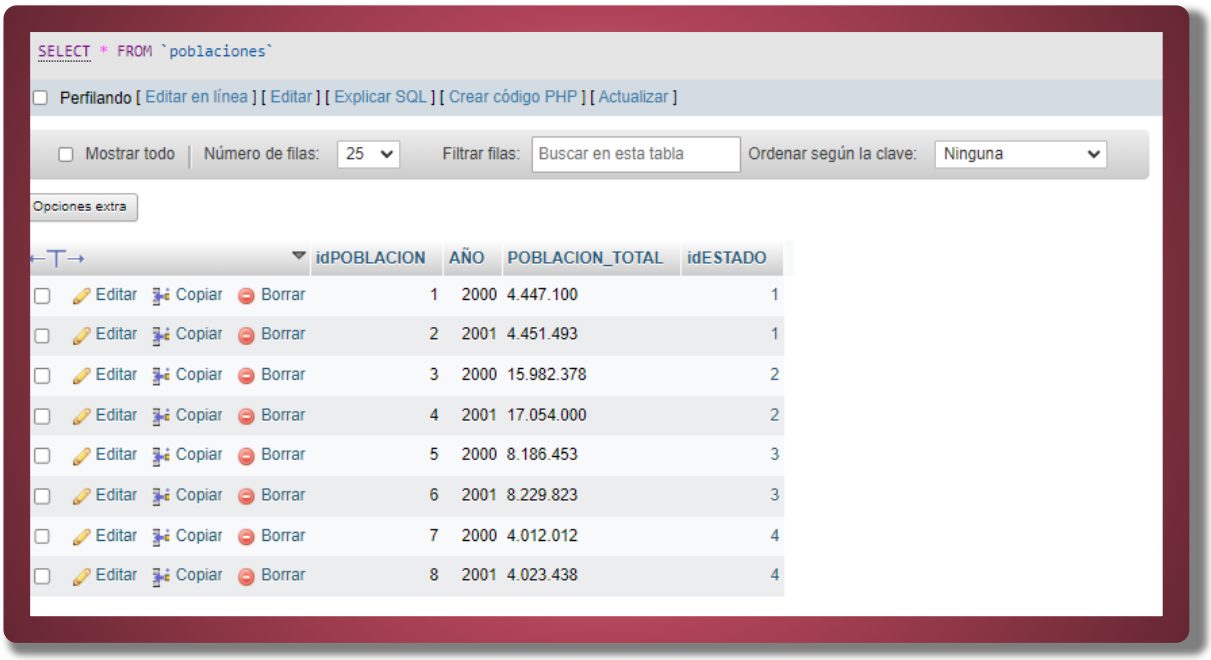
1. <u>Creación de base de datos con sus respectivas tablas por Jéssica:</u>

```python
#Conectar con BBDD
import mysql.connector

conexion=mysql.connector.connect(host="localhost", user="root", password="", database="")

#Crear cursor
cursor = conexion.cursor()

#Ejecutar código SQL
sql = """
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema RETO_ESTADOS
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `RETO_ESTADOS` ;
USE `RETO_ESTADOS` ;

-- -----------------------------------------------------
-- Table `RETO_ESTADOS`.`ESTADOS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `RETO_ESTADOS`.`ESTADOS` (
  `idESTADO` INT NOT NULL,
  `NOMBRE` VARCHAR(45) NOT NULL,
  `LATITUD` VARCHAR(500) NOT NULL,
  `LONGITUD` VARCHAR(500) NOT NULL,
  `FECHA_FUNDACION` DATE NOT NULL,
  PRIMARY KEY (`idESTADO`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `RETO_ESTADOS`.`POBLACIONES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `RETO_ESTADOS`.`POBLACIONES` (
  `idPOBLACION` INT NOT NULL,
  `AÑO` YEAR NOT NULL,
  `POBLACION_TOTAL` VARCHAR(45) NOT NULL,
  `idESTADO` INT NOT NULL,
  PRIMARY KEY (`idPOBLACION`),
    FOREIGN KEY (`idESTADO`)
    REFERENCES `RETO_ESTADOS`.`ESTADOS` (`idESTADO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `RETO_ESTADOS`.`DEFUNCIONES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `RETO_ESTADOS`.`DEFUNCIONES` (
  `idDEFUNCIONES` INT NOT NULL,
  `AÑO` YEAR NOT NULL,
  `DEFUNCIONES_TOTAL` VARCHAR(45) NOT NULL,
  `idESTADO` INT NOT NULL,
  PRIMARY KEY (`idDEFUNCIONES`),
    FOREIGN KEY (`idESTADO`)
    REFERENCES `RETO_ESTADOS`.`ESTADOS` (`idESTADO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `RETO_ESTADOS`.`RESIDENTES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `RETO_ESTADOS`.`RESIDENTES` (
  `idRESIDENTES` INT NOT NULL,
  `AÑO` YEAR NOT NULL,
  `MENORES_65` VARCHAR(45) NOT NULL,
  `idESTADO` INT NOT NULL,
  PRIMARY KEY (`idRESIDENTES`),
    FOREIGN KEY (`idESTADO`)
    REFERENCES `RETO_ESTADOS`.`ESTADOS` (`idESTADO`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
"""


# Dividir el código SQL en consultas individuales
queries = sql.split(';')

# Ejecutar cada consulta individualmente
for query in queries:
    cursor.execute(query)

# Cerrar la conexión
conexion.close()
```

2. Inserción de datos en la base de datos por parte de Jaime

```python
#Conectar con BBDD
import mysql.connector

conexion=mysql.connector.connect(host="localhost", user="root", password="", database="reto_estados")

#Crear cursor
cursor = conexion.cursor()

#Ejecutar código SQL
cursor.execute("SET @OLD_SQL_MODE=@@SQL_MODE")
cursor.execute("SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS")
cursor.execute("SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS")

cursor.execute("SET SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION'")
cursor.execute("SET FOREIGN_KEY_CHECKS=0")
cursor.execute("SET UNIQUE_CHECKS=0")

cursor.execute("SELECT * FROM reto_estados.estados")
result = cursor.fetchall()
for row in result:
    print(row)


cursor.execute("INSERT IGNORE INTO `reto_estados`.`estados` (`idESTADO`, `NOMBRE`, `LATITUD`, `LONGITUD`, `FECHA_FUNDACION`) VALUES ('1', 'Alabama', '33.258.882', '-86.829.534', '1819-12-14')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`estados` (`idESTADO`, `NOMBRE`, `LATITUD`, `LONGITUD`, `FECHA_FUNDACION`) VALUES ('2', 'Florida', '27.756.767', '-81.463.983', '1845-03-03')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`estados` (`idESTADO`, `NOMBRE`, `LATITUD`, `LONGITUD`, `FECHA_FUNDACION`) VALUES ('3', 'Georgia', '32.329.381', '-83.113.737', '1733-02-12')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`estados` (`idESTADO`, `NOMBRE`, `LATITUD`, `LONGITUD`, `FECHA_FUNDACION`) VALUES ('4', 'South Carolina', '33.687.439', '-80.436.374', '1776-03-26')")
conexion.commit()

cursor.execute("SELECT * FROM reto_estados.defunciones")
result = cursor.fetchall()
for row in result:
    print(row)

cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('1', 2000, '10.622', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('2', 2001, '15.912', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('3', 2000, '38.103', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('4', 2001, '166.069', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('5', 2000, '14.804', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('6', 2001, '15.000', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('7', 2000, '8.581', '4')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`defunciones` (`idDEFUNCIONES`, `AÑO`, `DEFUNCIONES_TOTAL`, `idESTADO`) VALUES ('8', 2001, '9.500', '4')")
conexion.commit()

cursor.execute("SELECT * FROM reto_estados.poblaciones")
result = cursor.fetchall()
for row in result:
    print(row)

cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('1', 2000, '4.447.100', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('2', 2001, '4.451.493', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('3', 2000, '15.982.378', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('4', 2001, '17.054.000', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('5', 2000, '8.186.453', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('6', 2001, '8.229.823', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('7', 2000, '4.012.012', '4')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`poblaciones` (`idPOBLACION`, `AÑO`, `POBLACION_TOTAL`, `idESTADO`) VALUES ('8', 2001, '4.023.438', '4')")
conexion.commit()

cursor.execute("SELECT * FROM reto_estados.residentes")
result = cursor.fetchall()
for row in result:
    print(row)

cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('1', 2000, '3.870.598', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('2', 2001, '3.880.476', '1')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('3', 2000, '13.237.167', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('4', 2001, '13.548.077', '2')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('5', 2000, '7.440.877', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('6', 2001, '7.582.146', '3')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('7', 2000, '3.535.770', '4')")
cursor.execute("INSERT IGNORE INTO `reto_estados`.`residentes` (`idRESIDENTES`, `AÑO`, `MENORES_65`, `idESTADO`) VALUES ('8', 2001, '3.567.172', '4')")
conexion.commit()

# Cerrar la conexión
conexion.close()
```

BASE DE DATOS EN PHPMYADMIN por parte de Ana**:**



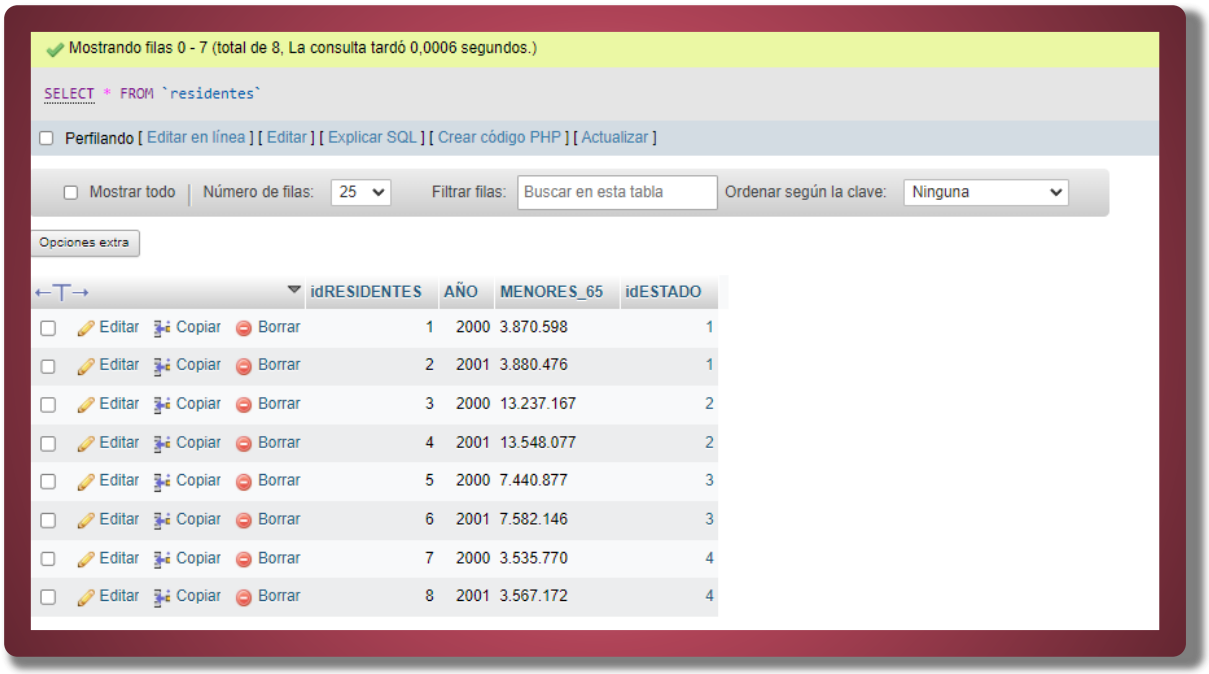
SELECT * FROM `poblaciones`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idPOBLACION	AÑO	POBLACION_TOTAL	idESTADO
Editar Copiar Borrar	1	2000	4.447.100	1
Editar Copiar Borrar	2	2001	4.451.493	1
Editar Copiar Borrar	3	2000	15.982.378	2
Editar Copiar Borrar	4	2001	17.054.000	2
Editar Copiar Borrar	5	2000	8.186.453	3
Editar Copiar Borrar	6	2001	8.229.823	3
Editar Copiar Borrar	7	2000	4.012.012	4
Editar Copiar Borrar	8	2001	4.023.438	4



Mostrando filas 0 - 7 (total de 8, La consulta tardó 0,0006 segundos.)

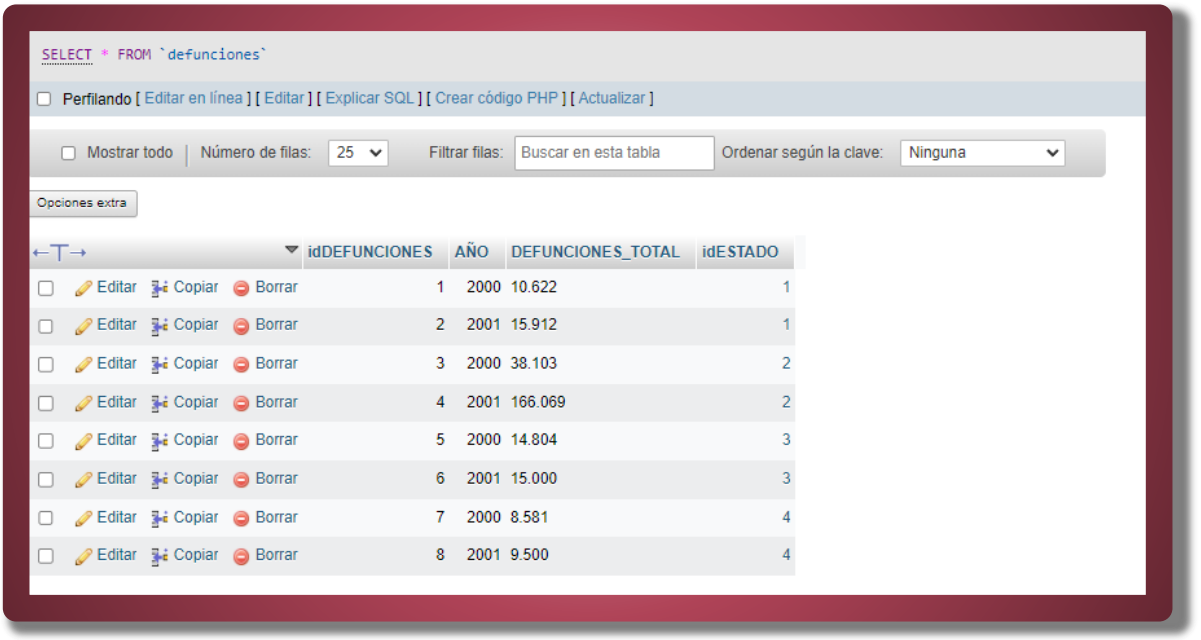
SELECT * FROM `residentes`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idRESIDENTES	AÑO	MENORES_65	idESTADO
Editar Copiar Borrar	1	2000	3.870.598	1
Editar Copiar Borrar	2	2001	3.880.476	1
Editar Copiar Borrar	3	2000	13.237.167	2
Editar Copiar Borrar	4	2001	13.548.077	2
Editar Copiar Borrar	5	2000	7.440.877	3
Editar Copiar Borrar	6	2001	7.582.146	3
Editar Copiar Borrar	7	2000	3.535.770	4
Editar Copiar Borrar	8	2001	3.567.172	4



SELECT * FROM `defunciones`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idDEFUNCIONES	AÑO	DEFUNCIONES_TOTAL	idESTADO
Editar Copiar Borrar	1	2000	10.622	1
Editar Copiar Borrar	2	2001	15.912	1
Editar Copiar Borrar	3	2000	38.103	2
Editar Copiar Borrar	4	2001	166.069	2
Editar Copiar Borrar	5	2000	14.804	3
Editar Copiar Borrar	6	2001	15.000	3
Editar Copiar Borrar	7	2000	8.581	4
Editar Copiar Borrar	8	2001	9.500	4

A partir del modelado de las tablas relacionales, se exportó la petición SQL a través de WorkBench. Jesé hizo esta exportación a partir del archivo de tablas que le pasó Jéssica y que previamente había sido elaborado entre todos en videollamada.

Una vez tenemos las querys elaboradas, Jéssica las pasó a VSCODE, donde se pulió la consulta, eliminando aquellas cuestiones que daban errores (tildes, comillas mal colocadas, líneas sobrantes), y añadiendo todo lo necesario para poder ejecutar los archivos de Python que crearían la base de datos con sus tablas y datos respectivos.

Para esto, con la finalidad de hacer un código más segmentado y legible, decidimos hacer dos archivos, uno que crearía la base de datos y las tablas relacionales, y otro para las inserciones de registros.

Durante el proceso detectamos los siguientes errores, que, tras un análisis de las causas, fueron solventados.

- Errores de sintaxis, debidos a descuidos a la hora de elaborar las tablas.

- Errores al ejecutar el archivo de Python que crearía la base de datos y tablas, ya que, en un primer momento, con el código introducido, sólo se creaba la tabla Estados. La solución fue ejecutar las consultas de manera individual e indicar al sistema que la separación entre unas y otras consultas vendría dada por punto y coma ";":

```python
# Dividir el código SQL en consultas individuales
queries = sql.split(';')

# Ejecutar cada consulta individualmente
for query in queries:
    cursor.execute(query)
```

- Errores al insertar los registros en las tablas. Como daba diferentes errores de sintaxis, se optó por ejecutar las consultas una a una y el archivo dejó de dar errores, no obstante, los datos aun no se estaban insertando. Finalmente, la solución fue añadir un conexion.commit()tras cada conjunto de peticiones.

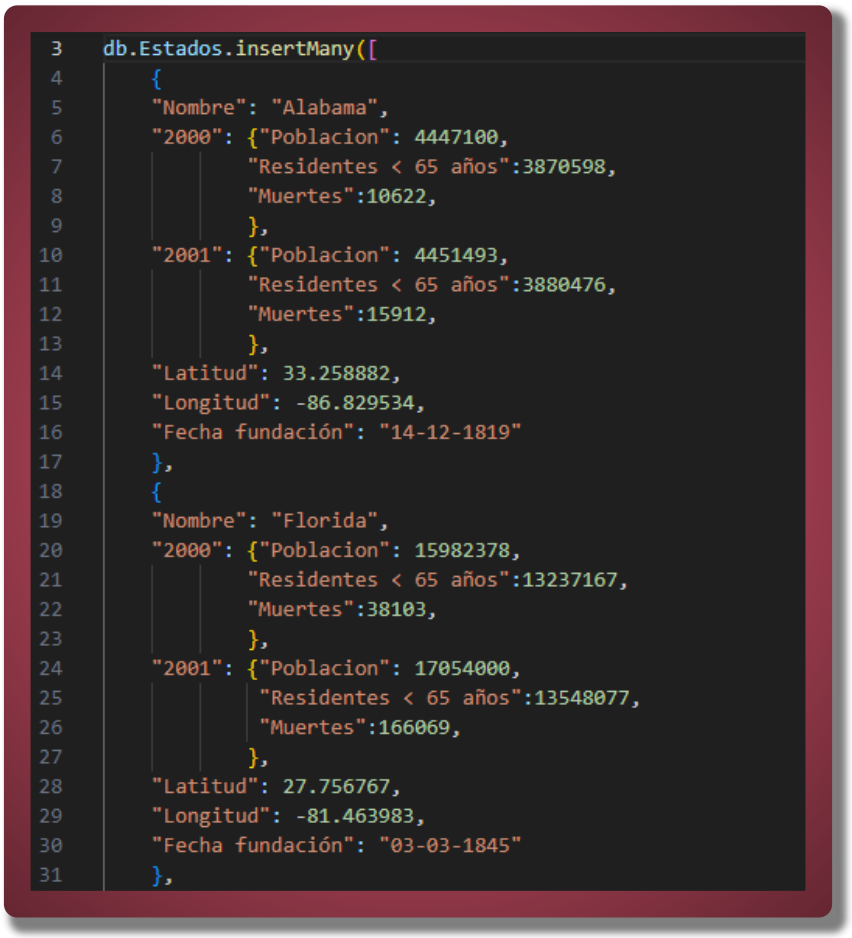
- Como en un primer momento se añadieron varios registros, en una segunda ejecución el sistema aparecía un error ya que los registros estaban duplicados. Para solucionarlo, se añadió IGNORE al lado de INSERT, para que durante el trasvase a la base de datos el sistema ignore los registros que ya estén almacenados y continúe con los que faltan.

BASE DE DATOS NO RELACIONAL MONGODB

Primero se conectó Visual Studio Code con el MongoDB para luego crear la base de datos con la respectiva colección:

```
1   use("Reto_Estados")
2   db.createCollection("Estados")
```

Una vez insertados los documentos en la colección, Jéssica y Jaime mostraron cómo se deberían ingresar los datos para que se encuentre en el formato correcto y probaron si el código funcionaba correctamente.



```
db.Estados.insertMany([
    {
    "Nombre": "Alabama",
    "2000": {"Poblacion": 4447100,
            "Residentes < 65 años":3870598,
            "Muertes":10622,
            },
    "2001": {"Poblacion": 4451493,
            "Residentes < 65 años":3880476,
            "Muertes":15912,
            },
    "Latitud": 33.258882,
    "Longitud": -86.829534,
    "Fecha fundación": "14-12-1819"
    },
    {
    "Nombre": "Florida",
    "2000": {"Poblacion": 15982378,
            "Residentes < 65 años":13237167,
            "Muertes":38103,
            },
    "2001": {"Poblacion": 17054000,
             "Residentes < 65 años":13548077,
             "Muertes":166069,
            },
    "Latitud": 27.756767,
    "Longitud": -81.463983,
    "Fecha fundación": "03-03-1845"
    },
```

```
    {
    "Nombre": "Georgia",
    "2000": {"Poblacion": 8186453,
            "Residentes < 65 años":7440877,
            "Muertes":14804,
            },
    "2001": {"Poblacion": 8229823,
            "Residentes < 65 años":7582146,
            "Muertes":15000,
            },
    "Latitud": 32.329381,
    "Longitud": -83.113737,
    "Fecha fundación": "12-02-1733"
    },
    {
    "Nombre": "South Carolina",
    "2000": {"Poblacion": 4012012,
            "Residentes < 65 años":3535770,
            "Muertes":8581,
            },
    "2001": {"Poblacion": 4023438,
            "Residentes < 65 años":3567172,
            "Muertes":9500,
            },
    "Latitud": 33.687439,
    "Longitud": -80.436374,
    "Fecha fundación": "26-03-1776"
    }
])
```

Finalmente comprobamos que se pueden visualizar los datos en MongoDB, con nombre para la base de datos "Retos_Estados" con su colección llamada "Estados". Asimismo, pudimos comprobar que los cuatro documentos están correctamente insertados.

Cabe destacar que finalmente se realizó una modificación en las tablas, para poder tener un ID autoincremental. La modificación fue realizada por Jéssica y está incluida en los datos adjuntos.

ANEXO

En este anexo hemos creado un acceso a los siguientes documentos correspondientes a la creación de la base de datos y su código:

SQL.PY

INSERCIÓN DE DATOS.PY

MONGODB.PY