COMISIÓN: 53190

ALUMNO: ALONSO MAURICIO JAVIER

TRABAJO PRÁCTICO – ENTREGA FINAL

Instrucciones para una correcta instalación

1. Instalar los scripts en MySQL

Se debe ejecutar los scripts en el siguiente orden, para lograr una correcta implementación del schema:

- 1-Script-DB

Este script crea el shema **proyecto_alonsomauricio**

El schema cuenta con las siguientes tablas:

- Tabla envergadura

```
-- Creación de tablas
CREATE TABLE IF NOT EXISTS envergadura (
    idEnvergadura INT AUTO_INCREMENT,
    tipo_envergadura VARCHAR(30) NOT NULL,
    PRIMARY KEY (idEnvergadura)
);
```

- Tabla estado

```
CREATE TABLE IF NOT EXISTS estado (
    idEstado INT AUTO_INCREMENT,
    nombre_estado VARCHAR(15) NOT NULL,
    PRIMARY KEY (idEstado)
);
```

- Tabla país

```
CREATE TABLE IF NOT EXISTS pais (
    idPais INT AUTO_INCREMENT,
    nombre_pais VARCHAR(50) NOT NULL,
    PRIMARY KEY (idPais)
);
```

- Tabla contacto

```
CREATE TABLE IF NOT EXISTS contacto (
    idContacto INT AUTO_INCREMENT,
    telefono VARCHAR(20),
    email VARCHAR(50) NOT NULL,
    web VARCHAR(100),
    PRIMARY KEY (idContacto)
);
```

- Tabla dep

```
CREATE TABLE IF NOT EXISTS dep (
    idDep INT AUTO_INCREMENT,
    nombre_dep VARCHAR(100) NOT NULL,
    telefono_dep VARCHAR(20),
    email_dep VARCHAR(50) NOT NULL,
    PRIMARY KEY (idDep)
);
```

- Tabla representante

```
CREATE TABLE IF NOT EXISTS representante (
    idRepresentante INT AUTO_INCREMENT,
    nombre_rep VARCHAR(100) NOT NULL,
    dni VARCHAR(8) NOT NULL,
    profesion VARCHAR(50),
    nacionalidad INT,
    PRIMARY KEY (idRepresentante),
    FOREIGN KEY (nacionalidad) REFERENCES pais(idPais)
);
```

- Tabla provincia

```
CREATE TABLE IF NOT EXISTS provincia (
    idProvincia INT AUTO_INCREMENT,
    nombre_provincia VARCHAR(50) NOT NULL,
    PRIMARY KEY (idProvincia)
);
```

- Tabla ciudad

```
CREATE TABLE IF NOT EXISTS ciudad (
    idCiudad INT AUTO_INCREMENT,
    nombre_ciudad VARCHAR(50) NOT NULL,
    PRIMARY KEY (idCiudad)
);
```

- Tabla sector

```
CREATE TABLE IF NOT EXISTS sector (
    idSector INT AUTO_INCREMENT,
    nombre_sector VARCHAR(30) NOT NULL,
    PRIMARY KEY (idSector)
);
```

- Tabla rubro

```
CREATE TABLE IF NOT EXISTS rubro (
    idRubro INT AUTO_INCREMENT,
    actividad VARCHAR(200) NOT NULL,
    idSector INT,
    PRIMARY KEY (idRubro),
    FOREIGN KEY (idSector) REFERENCES sector(idSector)
);
```

- Tabla bancos

```
CREATE TABLE IF NOT EXISTS bancos (
    idBanco INT AUTO_INCREMENT,
    nombre_banco VARCHAR(30),
    PRIMARY KEY (idBanco)
);
```

- Tabla rrhh

```
CREATE TABLE IF NOT EXISTS rrhh (
    idRrhh INT AUTO_INCREMENT,
    num_socios TINYINT NOT NULL,
    num_empleados TINYINT,
    PRIMARY KEY (idRrhh)
);
```

- Tabla asesor

```
CREATE TABLE IF NOT EXISTS asesor (
    idAsesor INT AUTO_INCREMENT,
    nombre_asesor VARCHAR(100) NOT NULL,
    idDep INT,
    PRIMARY KEY (idAsesor),
    FOREIGN KEY (idDep) REFERENCES dep(idDep)
);
```

- Tabla linea

```
CREATE TABLE IF NOT EXISTS linea (
    idLinea INT AUTO_INCREMENT,
    nombre_linea VARCHAR(50) NOT NULL,
    PRIMARY KEY (idLinea)
);
```

- Tabla programa

```
CREATE TABLE IF NOT EXISTS programa (
    idPrograma INT AUTO_INCREMENT,
    nombre_programa VARCHAR(100) NOT NULL,
    descripcion TEXT,
    legajo VARCHAR(20),
    monto INT,
    idEstado INT,
    idLinea INT,
    PRIMARY KEY (idPrograma),
    FOREIGN KEY (idLinea) REFERENCES linea(idLinea),
    FOREIGN KEY (idEstado) REFERENCES estado(idEstado)
);
```

- Tabla ratio

```
CREATE TABLE IF NOT EXISTS ratio (
    idRatio INT AUTO_INCREMENT,
    patneto INT NOT NULL,
    margen DECIMAL(5,2) NOT NULL,
    roa DECIMAL(5,2),
    roe DECIMAL(5,2),
    endeudamiento DECIMAL(5,2),
    PRIMARY KEY (idRatio)
);
```

- Tabla localización

```
CREATE TABLE IF NOT EXISTS localizacion (
    idLocalizacion INT AUTO_INCREMENT,
    nombre_calle VARCHAR(50),
    numeracion_calle INT,
    idCiudad INT,
    idProvincia INT,
    idPais INT,
    PRIMARY KEY (idLocalizacion),
    FOREIGN KEY (idCiudad) REFERENCES ciudad(idCiudad),
    FOREIGN KEY (idProvincia) REFERENCES provincia(idProvincia),
    FOREIGN KEY (idPais) REFERENCES pais(idPais)
);
```

- Tabla empresa

```
CREATE TABLE IF NOT EXISTS empresa (
    idEmpresa INT AUTO_INCREMENT,
    nombre_soc VARCHAR(100) NOT NULL,
    tipo VARCHAR(50) NOT NULL,
    cuit VARCHAR(13) NOT NULL,
    fecha_contrato DATE NOT NULL,
    idEnvergadura INT,
    fecha_i_m_e DATE,
    idRubro INT,
    idBanco INT,
    idRrhh INT,
    idAsesor INT,
    idContacto INT,
    idRepresentante INT,
    idPrograma INT,
    idRatio INT,
    idLocalizacion INT,
    PRIMARY KEY (idEmpresa),
    FOREIGN KEY (idEnvergadura) REFERENCES envergadura(idEnvergadura),
    FOREIGN KEY (idRubro) REFERENCES rubro(idRubro),
    FOREIGN KEY (idBanco) REFERENCES bancos(idBanco),
    FOREIGN KEY (idRrhh) REFERENCES rrhh(idRrhh),
    FOREIGN KEY (idAsesor) REFERENCES asesor(idAsesor),
    FOREIGN KEY (idContacto) REFERENCES contacto(idContacto),
    FOREIGN KEY (idRepresentante) REFERENCES representante(idRepresentante),
    FOREIGN KEY (idPrograma) REFERENCES programa(idPrograma),
    FOREIGN KEY (idRatio) REFERENCES ratio(idRatio),
    FOREIGN KEY (idLocalizacion) REFERENCES localizacion(idLocalizacion)
```

- 2–Script-Triggers

En este script se crean los triggers y las tablas bitacora_empresa y bitacora_programa.

Se toman como tablas para los triggers las tablas empresa y programa. Los triggers registran las incorporación o eliminación de registros de estas tablas, asentando un registro en las tablas de bitacora_empresa y bitacora_programa.

- 3-Script-DatosParametricas

Este script incorpora datos a las tablas paramétricas. Las tablas paramétricas son aquellas que registran datos que después se repiten en varios registros de las otras tablas. También su cantidad de registros tiende a variar menos que en otras tablas. Son tablas paramétricas: envergadura, estado, pais, provincia, dep, sector, línea, asesor, bancos, línea.

- 4-Script-InsercionDatos

Este script incorpora datos en las tablas que no son paramétricas. Se registran datos en las tablas: programa, rubro, ratio, rrhh, contacto, representante, ciudad, localizacion, empresa.

- 5-Script-Vistas

Este script crea cinco vistas:

  - Vista_empresa_completa: En esta vista se muestran TODOS los datos que componen la base

- Vista_estado_empresas_por_departamento: En esta vista se cuentan la cantidad de casos que hay por DEP, discriminando según el estado. Se ordena la información por DEP y luego por Estado
- Vista_cantidad_empresas_por_sector: En esta vista se cuentan la cantidad de casos que hay por sector.
- Vista_empresas_con_ciudad_actividad_y_asesor: En esta vista se enlistan las empresa detallando de qué ciudad son, qué actividad realizan y quién las está asesorando.
- Vista_detalle_empresas_con_representante: En esta vista se enlistan las empresa detallando su envergaruda, cuántos socios y empleados tienen, el nombre de su representante y la nacionalidad.

- 6-Script-Funciones

Este script crea dos funciones:

- calcular_estadisticas_por_estado: La primera función calcula, en relación del estado ingresado, el monto total (sumatoria de importes), y los promedio de patrimonios netos, margen, roa y roe de todas las empresas de ese estado.

Para testear esta función se debe ingresar el siguiente script:

```
USE proyecto_alonsomauricio;

SELECT calcular_estadisticas_por_estado('nombre_estado');

EJEMPLO: SELECT calcular_estadisticas_por_estado('Instrumentación');
```

Los valores almacenados en nombre_estado son: Instrumentación, Amortizando, Cancelado, Desistido, Moroso

- contar_empresas_por_dep: La segunda función calcula, en relación de la dep ingresada, la cantidad de empresas vinculadas a esa dep.

Para testear esta función se debe ingresar el siguiente script:

```
USE proyecto_alonsomauricio;

SELECT contar_empresas_por_dep('nombre_dep');

EJEMPLO : contar_empresas_por_dep SELECT contar_empresas_por_dep('DEP - Buenos Aires');
```

Los valores almacenados en nombre_dep son: DEP - Buenos Aires, DEP - La Plata, DEP - Catamarca, DEP - Resistencia, DEP - Rawson, DEP – Córdoba DEP - Resistencia, DEP - Paraná, DEP - Formosa, DEP - San Salvador de Jujuy, DEP - Santa Rosa DEP - La Rioja, DEP - Mendoza, DEP - Posadas, DEP - Neuquén, DEP - Viedma, DEP - Salta, DEP - San Juan DEP - San Luis, DEP - Río Gallegos, DEP - Santa Fe, DEP - Santiago del Estero, DEP - Ushuaia, DEP - San Miguel de Tucumán.

- 7-Script-StoredProcedures

Este script crea dos stored procedures:

- OrdenarTabla: El primer Stored Procedure ordena una tabla de forma ascendente o descendente.

Lo que se completa en los parámetros nombre de tabla, campos y criterio de ordenamiento (ASC o DESC) es a elección del usuario.

Ejemplo:

```
use proyecto_alonsomauricio;

CALL OrdenarTabla('empresa', 'nombre_soc', 'ASC');

CALL OrdenarTabla('asesor', 'nombre_asesorcalcular_estadisticas_por_estado', 'DESC');

CALL OrdenarTabla('bancos', 'idbanco', 'DESC');
```

- GestionarBancos: El segundo Stored Procedure gestiona la tabla bancos, agregando o eliminando bancos.

Ejemplo para agregar un banco:

```
use proyecto_alonsomauricio;

SET @resultado = '';

CALL GestionarBancos('insertar', 'Banca Almafuerte', NULL, @resultado);

SELECT @resultado;
```

Se debe declarar la variable resultado antes de correr el CALL. La función tiene que llevar el parámetro ´insertar´

Para diferenciar de ´eliminar´: Luego viene el nombre del banco. El campo NULL es porque no se toma un id ya cargado.

Ejemplo para eliminar un banco:

```
use proyecto_alonsomauricio;

SET @resultado = '';

CALL GestionarBancos('eliminar', NULL, 20, @resultado);

SELECT @resultado;
```

Se debe declarar la variable resultado antes de correr el CALL. La función tiene llevar el parámetro ´eliminar´

Para diferenciar de ´insertar´: Luego NULL, que sería el nombre del banco; no se necesita porque ese busca con el id del banco a eliminar.

- 8-Script-DCL

Este script crea dos usuarios:

- coderhouse-subzero@localhost: este usuario tiene la password *coderhouse* y tiene permiso sólo de lectura de datos.
- coderhouse-scorpion@localhost: este usuario tiene la password *coderhouse* y tiene permisos para lectura, inserción y modificación de datos.

Importante, al menos el usuario **coderhouse-scorpion@localhost** con la misma password debe estar creado para testear las herramientas Power BI Desktop y NodeJs. En caso de tener otro usuario, deberán incorporarse las especificaciones de dicho usuario.

- 9-Script-TCL:

Este script es opcional correrlo, tiene como objetivo testear la manipulación de los datos incorporando las palabras claves COMMIT, ROLLBACK y SAVEPOINT.

El script tiene partes de su código comentado para que, al quitarle lo comentado y dejar activas estas partes, se puedan testear otros efectos fal correr el script. Así se puede testear Rollback, commit, la inserción de datos, la eliminación de un savepoint.

- 10-Script-Backup:

El backup se realizó con la opción “Dump Structure and Data”.

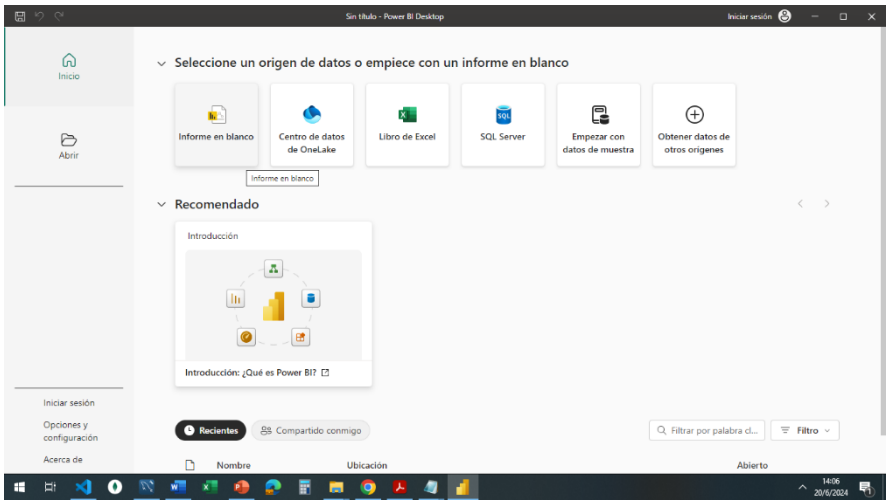
Se eligió “Export to Self-Containded-File”.

No se incorporaron “Dump Stored Procedures and Funcions”, “Dump Events”, “Dump Triggers”.

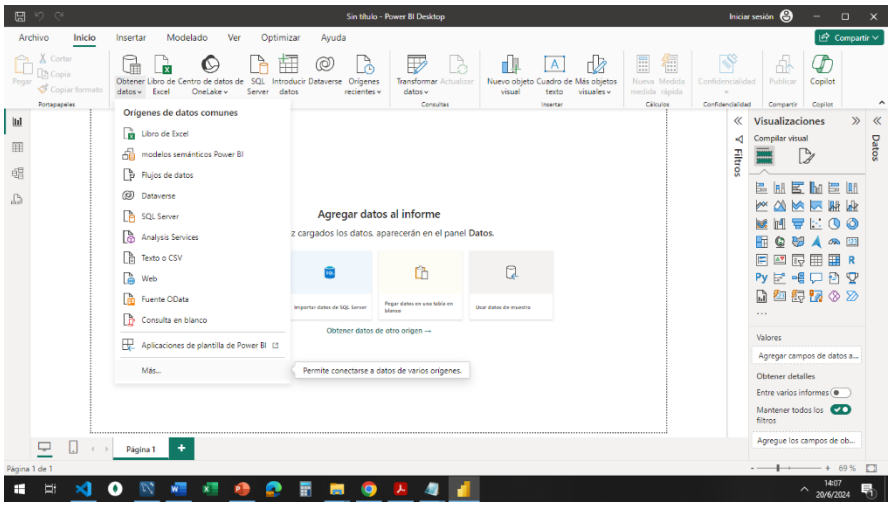
2. Como configurar Power Bi Desktop

Para poder importar los datos en Power Bi, se recomienda seguir estos pasos:

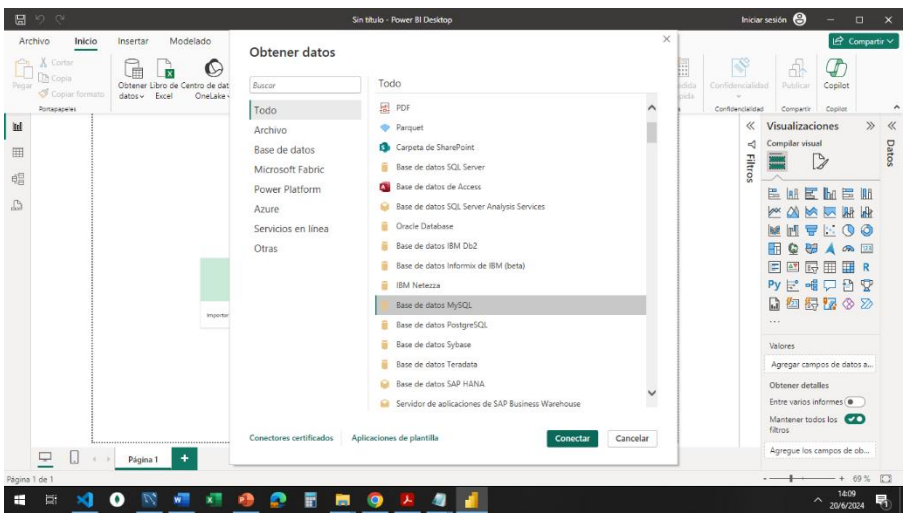
- Abrir Power Bi Desktop.
- Ir a Informe en Blanco.



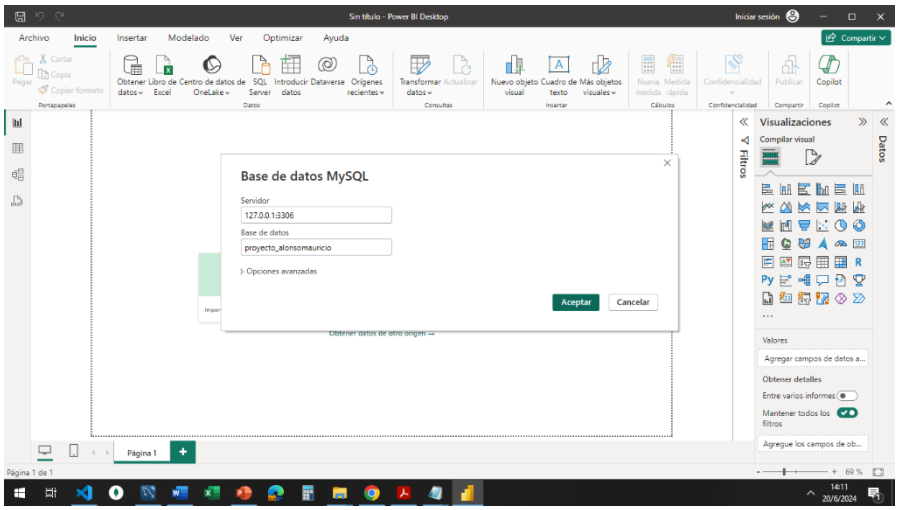
- Ir a Obtener Datos -> Más:



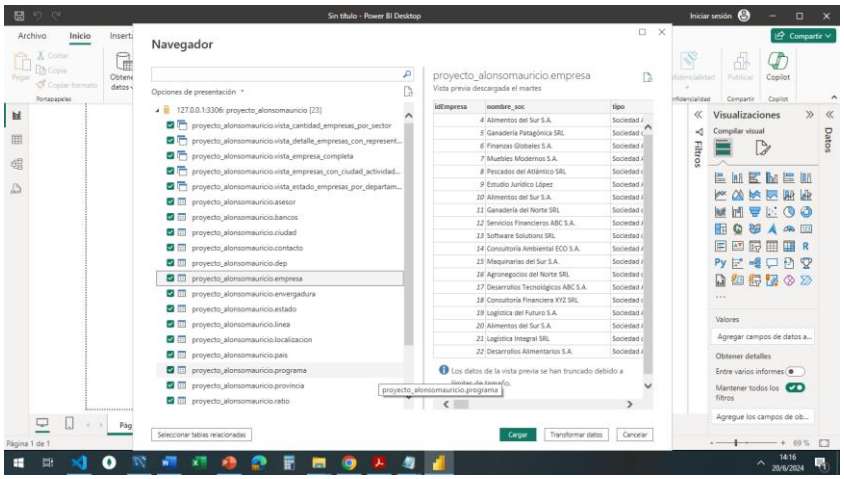
- Seleccionar Base de Datos MySQL:



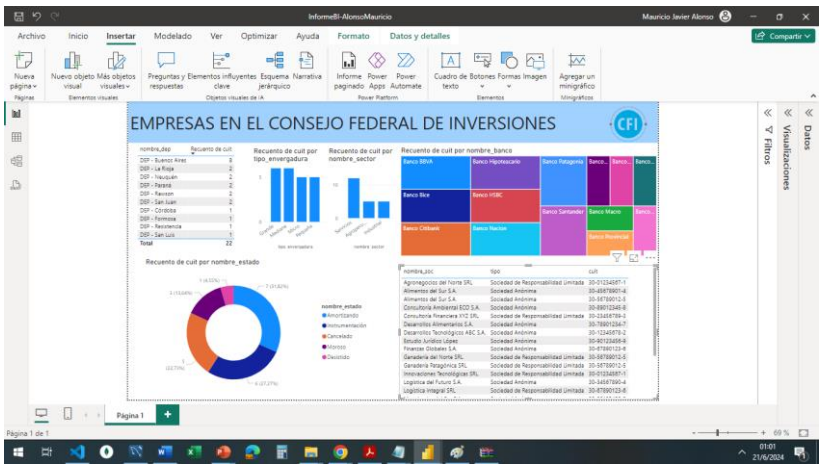
- Ingresar la referencia a la base de datos: en este caso el servidor es **127.0.0.1:3306**



- Si la conexión es exitosa, se podrán importar las tablas:



- Trabajar en el diseño del informe:



3. Como trabajar con NodeJs y Sequelize

Se eligió NodeJs para testear la integración con otras herramientas.

La app creada es muy simple y se desarrolló sólo a los fines de testear la integración entre MySQL y NodeJs. Sólo contiene dos peticiones “GET” para traer los nombres de las empresas y sus cuits o bien buscar empresas por CUIT.

No contiene la posibilidad de insertar datos, cambiar el usuario y otras operaciones con los datos, ya que exceden el límite de este curso. Incluso el usuario está ingresado de modo hardcode.

Para explorar cómo se hizo la integración, se deben seguir los siguientes pasos:

- Bajar el código del repositorio de Github:

La dirección del repositorio es: https://github.com/nunicho/sql-busqueda-empresas.git

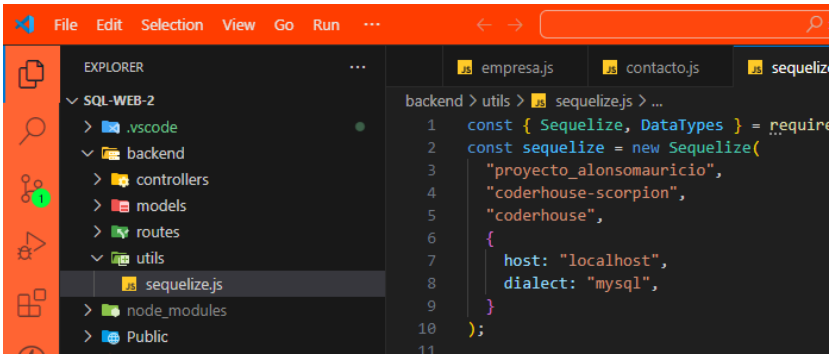
- Abrirlo con el editor de código de preferencia. La recomendación es utilizar Visual Studio Code.

Se debe tener instalado NodeJs en la pc donde se descargue el repositorio.

- Una vez descargado, abrir una terminal y, en la raíz del repositorio, ejecutar: **npm install**

Es importante tener instalados los scripts de mysql y **tener creado el usuario coderhouse-scorpion@localhost** con la password **coderhouse**

En caso de querer utilizar otro usuario y/u otra password, modificar el archivo **sequelize.js**



- Hechos los cambios necesarios, ejecutar el comando **npm run dev**
- Abrir el sitio web en **http://localhost:3000/**