Universidad Autónoma del Estado de Hidalgo

REPORTE DE ÚLTIMA PRÁCTICA

DATA WAREHOUSE

ALUMNO:Israel Campos Vázquez
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En el ámbito de la gestión, manipulación y administración de bases de datos, es fundamental comprender diversas técnicas y procesos que optimizan el almacenamiento, la consulta y la integración de datos. Este documento aborda los siguientes temas clave:

Procesos ETL (Extract, Transform, Load): Un conjunto de procedimientos utilizados para extraer, transformar y cargar datos desde múltiples fuentes hacia un almacenamiento centralizado, como un data warehouse.

DATA WAREHOUSE: Un Data Warehouse (en español, almacén de datos) es un sistema que almacena grandes volúmenes de datos históricos provenientes de diferentes fuentes, con el objetivo de facilitar el análisis, la toma de decisiones y el reporteo empresarial.

SELECT INTO FILE: Una técnica para exportar datos de una tabla a un archivo externo, facilitando la generación de reportes y respaldos.

LOAD: Un comando que permite la importación eficiente de datos desde archivos externos hacia una base de datos.

API: Por sus siglas en inglés Application Programming Interface es una interfaz que permite que dos sistemas o programas se comuniquen entre sí de forma estructurada, segura y controlada.

2. Marco teórico

Procesos ETL (Extract, Transform, Load)

Los procesos ETL (Extracción, Transformación y Carga) son una metodología clave en la gestión de datos, utilizada para integrar datos desde diferentes fuentes en un almacén de datos unificado.

Las siglas ETL significan:

E - Extract (Extracción) Se trata de obtener los datos desde una o varias fuentes de datos. Estas fuentes pueden ser bases de datos, archivos planos (CSV, Excel, txt), APIs, etc.

T - Transform (Transformación) En esta fase se limpian, normalizan, combinan o modifican los datos para que tengan la estructura adecuada. Aquí se puede:
-Corregir errores de formato
-Unificar unidades
-Calcular nuevos campos
-Filtrar datos irrelevantes

L - Load (Carga) Los datos ya transformados se cargan en el sistema destino, que suele ser un almacén de datos (data warehouse), donde pueden ser usados para análisis, reportes, BI (business intelligence), etc.

SELECT INTO OUTFILE

El comando SELECT INTO FILE se usa para exportar datos desde una tabla hacia un archivo externo en sistemas MySQL y MariaDB. Es particularmente útil para generar reportes o respaldos en formatos como CSV.

Listing 1: Sintaxis

```
SELECT * FROM empleados
INTO OUTFILE '/ruta/del/archivo.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Seleccionamos los campos de la tabla que queremos cargar al archivo .CSV o .TXT, después específicamos la ruta de destino donde se encuentra nuestro archivo (en este caso es un archivo .TXT). Enseguida especificamos que los valores de los campos los separe por comas y que los encapsule entre comillas dobles. Por último decimos que cada registro lo termine con un salto de línea.

LOAD

El comando LOAD DATA INFILE permite importar datos desde un archivo externo a una tabla en MySQL o MariaDB. Se utiliza para cargar grandes volúmenes de datos de manera eficiente.

Listing 2: Sintaxis

```
LOAD DATA INFILE '/ruta/del/archivo.csv'
INTO TABLE empleados
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Le especificamos la ruta del archivo del cual queremos obtener la información. Después le indicamos en que tabla cargue la información. Enseguida le especificamos que los valores de los campos van a estar separados por comas, encerrados entre comillas dobles, cada registro termina con un salto de línea y que ignore la primer línea.

API

Una API es como un mesero en un restaurante:
Tú (el cliente) haces una solicitud (por ejemplo, un platillo).
El mesero (API) lleva tu orden a la cocina (el servidor).
Luego regresa con tu comida (la respuesta).

Una API permite que un programa pida información o servicios a otro programa, define cómo deben estructurarse esas solicitudes y respuestas (por ejemplo, usando JSON o XML) y establece reglas de acceso, formatos y métodos (como GET, POST, PUT, DELETE si es una API web REST).

El tipo de API más común es el API REST, ya que es moderna, está basado en HTTP y es común en apps web y móviles.

3. Herramientas empleadas

1. DataGrip: Es un entorno de desarrollo integrado (IDE) para bases de datos creado por JetBrains. Está diseñado para ayudar a los desarrolladores y administradores de bases de datos a gestionar, consultar y optimizar sus bases de datos de manera eficiente.

2. MySQL Server: Es un sistema de gestión de bases de datos relacionales. Utiliza el lenguaje SQL para la administración y manipulación de datos. Se emplea para almacenar, organizar y gestionar información dependiendo el contexto en el que se utilice.

   Se suele utilizar para desarrollar aplicaciones web, gestionar datos de empresas, análisis de datos, entre otros.

4. Desarrollo

Creación de nodos

Listing 3: Profesor

```
CREATE DATABASE profesor;
```

Listing 4: Investigador

```
CREATE DATABASE investigador;
```

Listing 5: Integrante

```
CREATE DATABASE integrante;
```

LCS (Local Conceptual Schema)

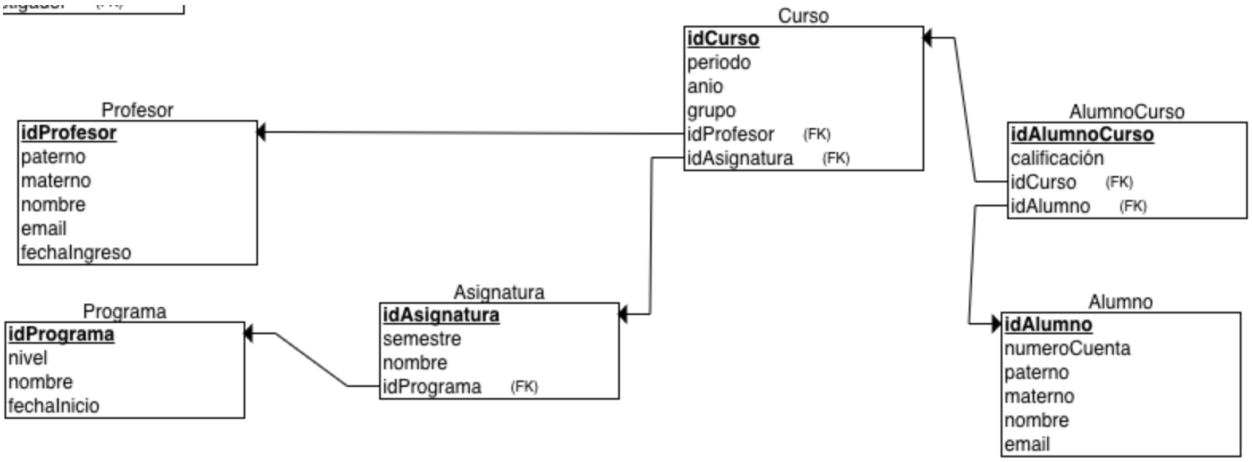

Figure 1: Diagrama Relacional Base de Datos Profesor

Listing 6: Tablas BD Profesor

```
CREATE TABLE Alumno
(
  idAlumno INT NOT NULL,
  numeroCuenta varchar(25) NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(120) NOT NULL,
  PRIMARY KEY (idAlumno)
);

CREATE TABLE Profesor
(
  idProfesor INT NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(250) NOT NULL,
```

```
  fechaIngreso date NOT NULL,
  PRIMARY KEY (idProfesor)
);

CREATE TABLE Programa
(
  idPrograma INT NOT NULL,
  nivel varchar(25) NOT NULL,
  nombre VARCHAR(150) NOT NULL,
  fechaInicio DATE NOT NULL,
  PRIMARY KEY (idPrograma)
);

CREATE TABLE Asignatura
(
  idAsignatura INT NOT NULL,
  semetre INT NOT NULL,
  nombre varchar(120) NOT NULL,
  programa INT NOT NULL,
  PRIMARY KEY (idAsignatura),
  FOREIGN KEY (programa) REFERENCES Programa(idPrograma)
);

CREATE TABLE Curso
(
  idCurso INT NOT NULL,
  periodo VARCHAR(15) NOT NULL,
  anio INT NOT NULL,
  asignatura INT NOT NULL,
  grupo INT NOT NULL,
  profesor INT NOT NULL,
  PRIMARY KEY (idCurso),
  FOREIGN KEY (asignatura) REFERENCES Asignatura(idAsignatura),
  FOREIGN KEY (profesor) REFERENCES Profesor(idProfesor)
);

CREATE TABLE AlumnoCurso
(
  idAlumnoCurso INT NOT NULL,
  alumno INT NOT NULL,
  curso INT NOT NULL,
  calificacion DECIMAL NOT NULL,
  PRIMARY KEY (idAlumnoCurso),
  FOREIGN KEY (alumno) REFERENCES Alumno (idAlumno),
  FOREIGN KEY (curso) REFERENCES Curso (idCurso)
);
```

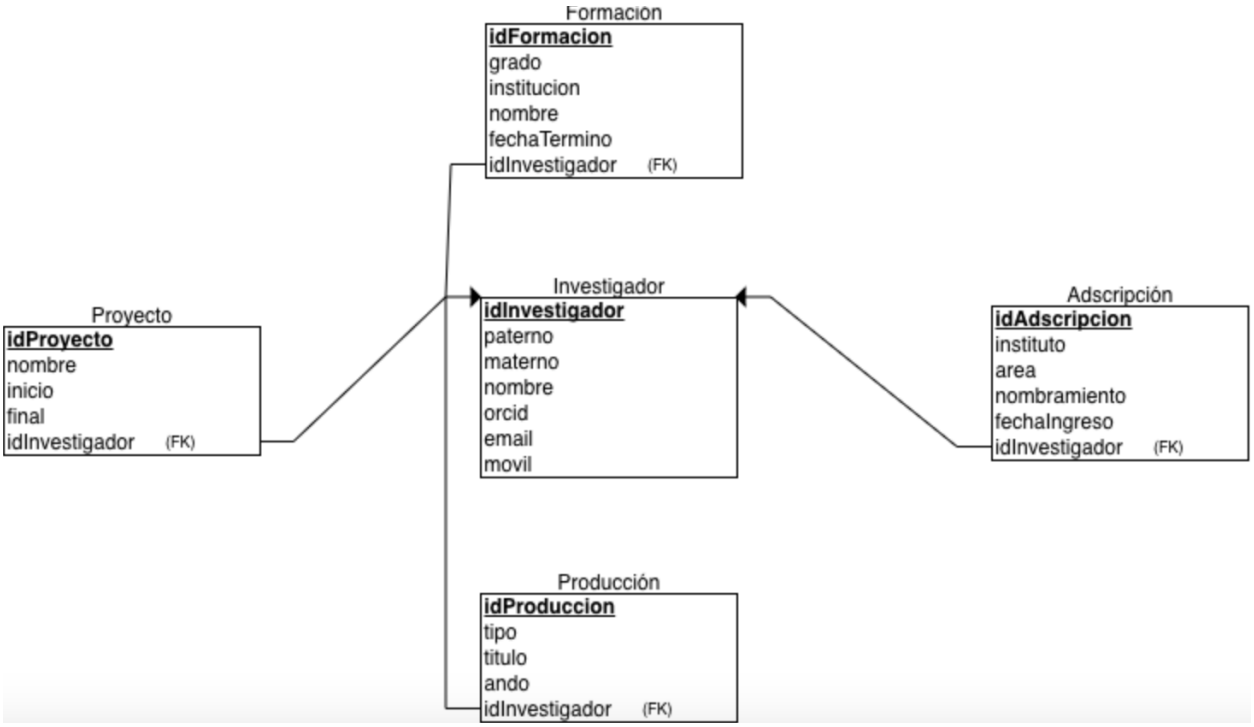

Figure 2: Diagrama Relacional Base de Datos Investigador

Listing 7: Tablas BD Investigador

```
CREATE TABLE Investigador
(
  idInvestigador INT NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(250) NOT NULL,
  orcid varchar(30) NOT NULL,
  movil varchar(15) NOT NULL,
  PRIMARY KEY (idInvestigador)
);

CREATE TABLE Proyecto
(
  idProyecto INT NOT NULL,
  nombre varchar(250) NOT NULL,
  inicio DATE NOT NULL,
  final DATE NOT NULL,
  investigador INT NOT NULL,
  PRIMARY KEY (idProyecto),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
);

CREATE TABLE Produccion
(
  idProduccion INT NOT NULL,
  investigador INT NOT NULL,
  tipo varchar(60) NOT NULL,
  titulo varchar(250) NOT NULL,
```

```
  anio INT NOT NULL,
  PRIMARY KEY (idProduccion),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
);

CREATE TABLE Adscripcion
(
  idAdscripcion INT NOT NULL,
  investigador INT NOT NULL UNIQUE,
  instituto varchar(80) NOT NULL,
  area varchar(150) NOT NULL,
  nombramiento varchar(10) NOT NULL,
  fechaIngreso DATE NOT NULL,
  PRIMARY KEY (idAdscripcion),
  FOREIGN KEY(investigador) REFERENCES Investigador(idInvestigador)
);

CREATE TABLE Formacion
(
  idFormacion INT NOT NULL,
  investigador INT NOT NULL,
  grado varchar(18) NOT NULL,
  institucion varchar(70) NOT NULL,
  nombre varchar(120) NOT NULL,
  fechaTermino DATE NOT NULL,
  PRIMARY KEY (idFormacion),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
);
```

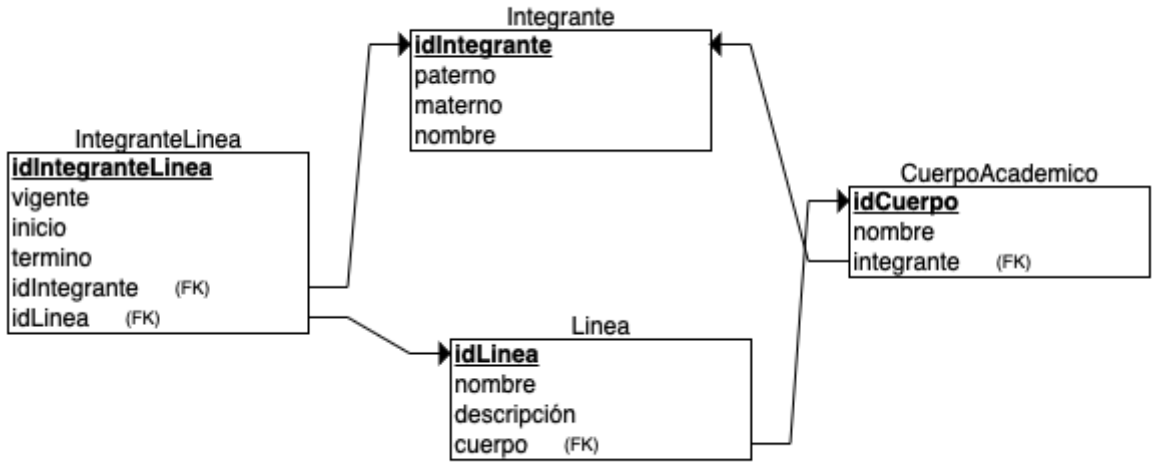

Figure 3: Diagrama Relacional Base de Datos Integrante

Listing 8: Tablas BD Integrante

```
CREATE TABLE Integrante
(
  idIntegrante INT NOT NULL,
  paterno varchar(80) NOT NULL,
```

```
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  PRIMARY KEY (idIntegrante)
);

CREATE TABLE CuerpoAcademico
(
  idCuerpo INT NOT NULL,
  nombre varchar(250) NOT NULL,
  integrante INT NOT NULL UNIQUE,
  PRIMARY KEY (idCuerpo),
  FOREIGN KEY (integrante) REFERENCES Integrante(idIntegrante)
);

CREATE TABLE Linea
(
  idLinea INT NOT NULL,
  nombre varchar(120) NOT NULL,
  descripcion varchar(500) NOT NULL,
  cuerpo INT NOT NULL,
  PRIMARY KEY (idLinea),
  FOREIGN KEY (cuerpo) REFERENCES CuerpoAcademico(idCuerpo)
);

CREATE TABLE IntegranteLinea
(
  idIntegranteLinea INT NOT NULL,
  integrante INT NOT NULL,
  linea INT NOT NULL,
  inicio DATE NOT NULL,
  termino DATE NOT NULL,
  vigente BOOLEAN NOT NULL,
  PRIMARY KEY (idIntegranteLinea),
  FOREIGN KEY (integrante) REFERENCES Integrante(idIntegrante),
  FOREIGN KEY (linea) REFERENCES Linea(idLinea)
);
```

Inserción de registros en los LCS

Listing 9: Registros Tabla Alumno BD Profesor

```
INSERT INTO Alumno (idAlumno, numeroCuenta, paterno, materno, nombre, email) VALUES
(1, 'CU00001', 'Lopez', 'Ramirez', 'Juan', 'juan.lopez1@example.com'),
(2, 'CU00002', 'Hernandez', 'Gomez', 'Ana', 'ana.hernandez2@example.com'),
(3, 'CU00003', 'Martinez', 'Perez', 'Carlos', 'carlos.martinez3@example.com');
```

Listing 10: Registros Tabla Profesor BD Profesor

```
INSERT INTO Profesor (idProfesor, paterno, materno, nombre, email, fechaIngreso) VALUES
(1, 'Lopez', 'Ramirez', 'Carlos', 'carlos.lopez1@univ.edu', '2015-08-01'),
(2, 'Hernandez', 'Gomez', 'Ana', 'ana.hernandez2@univ.edu', '2016-09-12'),
(3, 'Martinez', 'Perez', 'Luis', 'luis.martinez3@univ.edu', '2017-01-25');
```

Listing 11: Registros Tabla Programa BD Profesor

```
INSERT INTO Programa (idPrograma, nivel, nombre, fechaInicio) VALUES
(1, 'Licenciatura', 'Ingenieria␣en␣Sistemas␣Computacionales', '2015-08-01'),
(2, 'Maestria', 'Maestria␣en␣Ciencias␣de␣la␣Computacion', '2016-09-01'),
(3, 'Licenciatura', 'Ingeniera␣Industrial', '2017-01-15');
```

Listing 12: Registros Tabla Asignatura BD Profesor

```
INSERT INTO Asignatura (idAsignatura, semetre, nombre, programa) VALUES
(1, 1, 'Matematicas␣I', 1),
(2, 1, 'Introduccion␣a␣la␣Ingenieria', 2),
(3, 2, 'Programacion␣I', 3);
```

Listing 13: Registros Tabla Curso BD Profesor

```
INSERT INTO Curso (idCurso, periodo, anio, asignatura, grupo, profesor) VALUES
(1, 'Enero-Junio', 2022, 1, 1, 1),
(2, 'Agosto-Diciembre', 2022, 2, 1, 2),
(3, 'Enero-Junio', 2023, 3, 1, 3);
```

Listing 14: Registros Tabla AlumnoCurso BD Profesor

```
INSERT INTO AlumnoCurso (idAlumnoCurso, alumno, curso, calificacion) VALUES
(1, 1, 1, 9.0),
(2, 2, 2, 8.5),
(3, 3, 3, 7.8);
```

Listing 15: Registros Tabla Investigador BD Investigador

```
INSERT INTO Investigador (idInvestigador, paterno, materno, nombre, email, orcid, movil) VALUES
(1, 'Garcia', 'Lopez', 'Ana', 'ana.garcia1@example.com', '0000-0001-0001-0001', '5551001001'),
(2, 'Hernandez', 'Martinez', 'Luis', 'luis2@example.com', '0000-0001-0002-0002', '5551001002'),
(3, 'Martinez', 'Ramirez', 'Elena', 'elena3@example.com', '0000-0001-0003-0003', '5551001003');
```

Listing 16: Registros Tabla Proyecto BD Investigador

```
INSERT INTO Proyecto (idProyecto, nombre, inicio, final, investigador) VALUES
(1, 'Proyecto␣de␣Biotecnologia␣Aplicada', '2021-01-15', '2022-01-14', 1),
(2, 'Estudio␣de␣Microorganismos␣en␣Zonas␣Urbanas', '2020-03-01', '2021-02-28', 2),
(3, 'Nanotecnologia␣en␣Medicina', '2019-06-10', '2021-06-09', 3);
```

Listing 17: Registros Tabla Produccion BD Investigador

```
INSERT INTO Produccion (idProduccion, investigador, tipo, titulo, anio) VALUES
(1, 1, 'Articulo␣Cientifico', 'Impacto␣del␣Cambio␣Climatico␣en␣Zonas␣Aridas', 2020),
(2, 2, 'Libro', 'Fundamentos␣de␣Inteligencia␣Artificial', 2019),
(3, 3, 'Capitulo␣de␣Libro', 'Avances␣en␣Biotecnologia␣Vegetal', 2021);
```

Listing 18: Registros Tabla Adscripcion BD Investigador

```
INSERT INTO Adscripcion(idAdscripcion,investigador,instituto,area,nombramiento,fechaIngreso) VAI
(1, 1, 'Instituto␣de␣Ciencias', 'Fisica␣Cuantica', 'TC', '2015-01-12'),
(2, 2, 'Instituto␣de␣Tecnologia', 'Robotica␣Industrial', 'TC', '2016-03-22'),
(3, 3, 'Facultad␣de␣Biologia', 'Biologia␣Marina', 'MT', '2014-06-15');
```

Listing 19: Registros Tabla Formacion BD Investigador

```
INSERT INTO Formacion(idFormacion,investigador,grado,institucion,nombre,fechaTermino) VALUES
(1, 1, 'Licenciatura', 'Universidad␣Nacional', 'Ingenieria␣en␣Computacion', '2015-12-15'),
(2, 2, 'Maestria', 'Instituto␣Tecnologico', 'Ciencias␣de␣la␣Computacion', '2017-08-21'),
(3, 3, 'Licenciatura', 'Universidad␣de␣Guadalajara', 'Biologia', '2014-06-10');
```

Listing 20: Registros Tabla Integrante BD Integrante

```
INSERT INTO Integrante (idIntegrante, paterno, materno, nombre) VALUES
(1, 'Gonzalez', 'Lopez', 'Carlos'),
(2, 'Hernandez', 'Perez', 'Ana'),
(3, 'Rodriguez', 'Martinez', 'Luis');
```

Listing 21: Registros Tabla CuerpoAcademico BD Integrante

```
INSERT INTO CuerpoAcademico (idCuerpo, nombre, integrante) VALUES
(1, 'Cuerpo Academico de Fisica Teorica', 1),
(2, 'Cuerpo Academico de Matematicas Avanzadas', 2),
(3, 'Cuerpo Academico de Ingenieria de Software', 3);
```

Listing 22: Registros Tabla Linea BD Integrante

```
INSERT INTO Linea (idLinea, nombre, descripcion, cuerpo) VALUES
(1, 'Linea de Investigacion en Fisica', 'Estudio de teorias fundamentales de la fisica.', 1),
(2, 'Lanea de Investigacion en Matematicas Avanzadas', 'Investigacion en algebra abstracta.', 2)
(3, 'Linea de Investigacion en Ingenieria de Software', 'Desarrollo de sistemas complejos.', 3);
```

Listing 23: Registros Tabla IntegranteLinea BD Integrante

```
INSERT INTO IntegranteLinea(idIntegranteLinea,integrante,linea,inicio,termino,vigente) VALUES
(1, 1, 1, '2022-01-01', '2023-01-01', TRUE),
(2, 2, 2, '2022-02-01', '2023-02-01', TRUE),
(3, 3, 3, '2022-03-01', '2023-03-01', TRUE);
```

Creación de DATA WAREHOUSE

Listing 24: dataWarehouse

```
    CREATE DATABASE dataWarehouse;
```

GCS (Global Conceptual Schema)

Listing 25: Creacion de tablas en dataWarehouse

```
CREATE TABLE Alumno
(
  idAlumno INT NOT NULL,
  numeroCuenta varchar(25) NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(120) NOT NULL,
  PRIMARY KEY (idAlumno)
);

CREATE TABLE Profesor
(
  idProfesor INT NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(250) NOT NULL,
  fechaIngreso date NOT NULL,
  PRIMARY KEY (idProfesor)
);

CREATE TABLE Programa
(
  idPrograma INT NOT NULL,
  nivel varchar(25) NOT NULL,
  nombre VARCHAR(150) NOT NULL,
  fechaInicio DATE NOT NULL,
```

```
  PRIMARY KEY (idPrograma)
);

CREATE TABLE Asignatura
(
  idAsignatura INT NOT NULL,
  semetre INT NOT NULL,
  nombre varchar(120) NOT NULL,
  programa INT NOT NULL,
  PRIMARY KEY (idAsignatura),
  FOREIGN KEY (programa) REFERENCES Programa(idPrograma)
);

CREATE TABLE Curso
(
  idCurso INT NOT NULL,
  periodo VARCHAR(20) NOT NULL,
  anio INT NOT NULL,
  asignatura INT NOT NULL,
  grupo INT NOT NULL,
  profesor INT NOT NULL,
  PRIMARY KEY (idCurso),
  FOREIGN KEY (asignatura) REFERENCES Asignatura(idAsignatura),
  FOREIGN KEY (profesor) REFERENCES Profesor(idProfesor)
);

CREATE TABLE AlumnoCurso
(
  idAlumnoCurso INT NOT NULL,
  alumno INT NOT NULL,
  curso INT NOT NULL,
  calificacion DECIMAL NOT NULL,
  PRIMARY KEY (idAlumnoCurso),
  FOREIGN KEY (alumno) REFERENCES Alumno (idAlumno),
  FOREIGN KEY (curso) REFERENCES Curso (idCurso)
);


CREATE TABLE Investigador
(
  idInvestigador INT NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  email varchar(250) NOT NULL,
  orcid varchar(30) NOT NULL,
  movil varchar(15) NOT NULL,
  PRIMARY KEY (idInvestigador)
);

CREATE TABLE Proyecto
(
  idProyecto INT NOT NULL,
  nombre varchar(250) NOT NULL,
  inicio DATE NOT NULL,
  final DATE NOT NULL,
  investigador INT NOT NULL,
  PRIMARY KEY (idProyecto),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
```

```
);

CREATE TABLE Produccion
(
  idProduccion INT NOT NULL,
  investigador INT NOT NULL,
  tipo varchar(60) NOT NULL,
  titulo varchar(250) NOT NULL,
  anio INT NOT NULL,
  PRIMARY KEY (idProduccion),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
);

CREATE TABLE Adscripcion
(
  idAdscripcion INT NOT NULL,
  investigador INT NOT NULL UNIQUE,
  instituto varchar(80) NOT NULL,
  area varchar(150) NOT NULL,
  nombramiento varchar(10) NOT NULL,
  fechaIngreso DATE NOT NULL,
  PRIMARY KEY (idAdscripcion),
  FOREIGN KEY(investigador) REFERENCES Investigador(idInvestigador)
);

CREATE TABLE Formacion
(
  idFormacion INT NOT NULL,
  investigador INT NOT NULL,
  grado varchar(18) NOT NULL,
  institucion varchar(70) NOT NULL,
  nombre varchar(120) NOT NULL,
  fechaTermino DATE NOT NULL,
  PRIMARY KEY (idFormacion),
  FOREIGN KEY (investigador) REFERENCES Investigador(idInvestigador)
);


CREATE TABLE Integrante
(
  idIntegrante INT NOT NULL,
  paterno varchar(80) NOT NULL,
  materno varchar(80) NOT NULL,
  nombre varchar(80) NOT NULL,
  PRIMARY KEY (idIntegrante)
);

CREATE TABLE CuerpoAcademico
(
  idCuerpo INT NOT NULL,
  nombre varchar(250) NOT NULL,
  integrante INT NOT NULL UNIQUE,
  PRIMARY KEY (idCuerpo),
  FOREIGN KEY (integrante) REFERENCES Integrante(idIntegrante)
);

CREATE TABLE Linea
(
  idLinea INT NOT NULL,
```

```
  nombre varchar(120) NOT NULL,
  descripcion varchar(500) NOT NULL,
  cuerpo INT NOT NULL,
  PRIMARY KEY (idLinea),
  FOREIGN KEY (cuerpo) REFERENCES CuerpoAcademico(idCuerpo)
);

CREATE TABLE IntegranteLinea
(
  idIntegranteLinea INT NOT NULL,
  integrante INT NOT NULL,
  linea INT NOT NULL,
  inicio DATE NOT NULL,
  termino DATE NOT NULL,
  vigente BOOLEAN NOT NULL,
  PRIMARY KEY (idIntegranteLinea),
  FOREIGN KEY (integrante) REFERENCES Integrante(idIntegrante),
  FOREIGN KEY (linea) REFERENCES Linea(idLinea)
);
```

Extracción de datos

Listing 26: Extraccion de datos de BD Profesor

```
SELECT *
INTO OUTFILE '/tmp/Alumno.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Alumno;

SELECT *
INTO OUTFILE '/tmp/Profesor.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Profesor;

SELECT *
INTO OUTFILE '/tmp/Programa.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Programa;

SELECT *
INTO OUTFILE '/tmp/Asignatura.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Asignatura;

SELECT *
INTO OUTFILE '/tmp/Curso.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Curso;
```

```
SELECT *
INTO OUTFILE '/tmp/AlumnoCurso.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM AlumnoCurso;
```

Listing 27: Extraccion de datos de BD Investigador

```
SELECT *
INTO OUTFILE '/tmp/Investigador.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Investigador;

SELECT *
INTO OUTFILE '/tmp/Proyecto.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Proyecto;

SELECT *
INTO OUTFILE '/tmp/Produccion.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Produccion;

SELECT *
INTO OUTFILE '/tmp/Adscripcion.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Adscripcion;

SELECT *
INTO OUTFILE '/tmp/Formacion.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Formacion;
```

```
israelcamposvazquez@MacBook-Air-de-Israel ~ % sudo cat /tmp/Alumno.txt

Password:
"1","CU00001","López","Ramírez","Juan","juan.lopez1@example.com"
"2","CU00002","Hernández","Gómez","Ana","ana.hernandez2@example.com"
"3","CU00003","Martínez","Pérez","Carlos","carlos.martinez3@example.com"
"4","CU00004","García","Torres","Lucía","lucia.garcia4@example.com"
"5","CU00005","Sánchez","Morales","Pedro","pedro.sanchez5@example.com"
"6","CU00006","Ruiz","Vargas","Sofía","sofia.ruiz6@example.com"
"7","CU00007","Flores","Jiménez","Diego","diego.flores7@example.com"
"8","CU00008","Díaz","Herrera","María","maria.diaz8@example.com"
"9","CU00009","Cruz","Silva","José","jose.cruz9@example.com"
"10","CU00010","Ramírez","Navarro","Valeria","valeria.ramirez10@example.com"
"11","CU00011","Castro","Romero","Luis","luis.castro11@example.com"
"12","CU00012","Reyes","Mendoza","Paula","paula.reyes12@example.com"
"13","CU00013","Ortega","Rojas","Andrés","andres.ortega13@example.com"
"14","CU00014","Chávez","Luna","Fernanda","fernanda.chavez14@example.com"
"15","CU00015","Aguilar","Cortés","David","david.aguilar15@example.com"
"16","CU00016","Vega","Salazar","Laura","laura.vega16@example.com"
"17","CU00017","Molina","Delgado","Emilio","emilio.molina17@example.com"
"18","CU00018","Ibarra","Carrillo","Diana","diana.ibarra18@example.com"
"19","CU00019","Núñez","Acosta","Marco","marco.nunez19@example.com"
"20","CU00020","León","Guerrero","Isabel","isabel.leon20@example.com"
"21","CU00021","Campos","Rivas","Hugo","hugo.campos21@example.com"
"22","CU00022","Paredes","Peña","Julia","julia.paredes22@example.com"
"23","CU00023","Fuentes","Vega","Raúl","raul.fuentes23@example.com"
"24","CU00024","Valdez","Santos","Natalia","natalia.valdez24@example.com"
"25","CU00025","Miranda","Mejía","Iván","ivan.miranda25@example.com"
"26","CU00026","Herrera","Flores","Elena","elena.herrera26@example.com"
"27","CU00027","Ramos","Guzmán","Sebastián","sebastian.ramos27@example.com"
"28","CU00028","Cervantes","Lozano","Camila","camila.cervantes28@example.com"
"29","CU00029","Moreno","Barajas","Jorge","jorge.moreno29@example.com"
"30","CU00030","Soto","Villanueva","Daniela","daniela.soto30@example.com"
"31","CU00031","Medina","Bravo","Adrián","adrian.medina31@example.com"
"32","CU00032","Arias","Solís","Claudia","claudia.arias32@example.com"
"33","CU00033","Montes","Lara","Erick","erick.montes33@example.com"
"34","CU00034","Salinas","Quintero","Renata","renata.salinas34@example.com"
"35","CU00035","Delgado","Bautista","Oscar","oscar.delgado35@example.com"
"36","CU00036","Zamora","Camacho","Marina","marina.zamora36@example.com"
"37","CU00037","Treviño","Rosales","Alan","alan.trevino37@example.com"
"38","CU00038","Becerra","Avila","Luisa","luisa.becerra38@example.com"
"39","CU00039","Carranza","Valencia","Tomás","tomas.carranza39@example.com"
"40","CU00040","Villalobos","Aguirre","Carla","carla.villalobos40@example.com"
"41","CU00041","Acevedo","Escobar","Mario","mario.acevedo41@example.com"
"42","CU00042","Benítez","Padilla","Alejandra","alejandra.benitez42@example.com"
"43","CU00043","Estrada","Montoya","Héctor","hector.estrada43@example.com"
"44","CU00044","Lara","Saavedra","Patricia","patricia.lara44@example.com"
"45","CU00045","Gallegos","Esquivel","Rodrigo","rodrigo.gallegos45@example.com"
"46","CU00046","Cuevas","Del Río","Sara","sara.cuevas46@example.com"
"47","CU00047","Avendaño","Tapia","Fabián","fabian.avendano47@example.com"
"48","CU00048","Meza","Ponce","Gabriela","gabriela.meza48@example.com"
"49","CU00049","Rocha","Olvera","Esteban","esteban.rocha49@example.com"
"50","CU00050","Navarrete","Alvarado","Brenda","brenda.navarrete50@example.com"
```

Figure 4: Formato de cómo se guardan los registros

Listing 28: Extraccion de datos de BD Integrante

```
SELECT *
INTO OUTFILE '/tmp/Integrante.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Integrante;

SELECT *
INTO OUTFILE '/tmp/CuerpoAcademico.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM CuerpoAcademico;

SELECT *
INTO OUTFILE '/tmp/Linea.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Linea;

SELECT *
INTO OUTFILE '/tmp/IntegranteLinea.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM IntegranteLinea;
```

Carga de datos

Listing 29: Load DataWarehouse

```
LOAD DATA INFILE '/tmp/Alumno.txt'
INTO TABLE Alumno
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Profesor.txt'
INTO TABLE Profesor
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Programa.txt'
INTO TABLE Programa
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Asignatura.txt'
INTO TABLE Asignatura
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Curso.txt'
```

```
INTO TABLE Curso
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/AlumnoCurso.txt'
INTO TABLE AlumnoCurso
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';


LOAD DATA INFILE '/tmp/Investigador.txt'
INTO TABLE Investigador
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Proyecto.txt'
INTO TABLE Proyecto
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Produccion.txt'
INTO TABLE Produccion
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Adscripcion.txt'
INTO TABLE Adscripcion
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Formacion.txt'
INTO TABLE Formacion
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';


LOAD DATA INFILE '/tmp/Integrante.txt'
INTO TABLE Integrante
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/CuerpoAcademico.txt'
INTO TABLE CuerpoAcademico
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/Linea.txt'
INTO TABLE Linea
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
```

```
LINES TERMINATED BY '\n';

LOAD DATA INFILE '/tmp/IntegranteLinea.txt'
INTO TABLE IntegranteLinea
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Consultas a Data Warehouse

Listing 30: Stored Procedure producciones de un profesor

```
DELIMITER //
CREATE PROCEDURE ObtenerProduccionesProfesor(IN aPaterno varchar(80),IN aMaterno varchar(80),
IN pNombre varchar(80),IN pEmail varchar(250))
BEGIN
    SELECT P.nombre, P.paterno, P.materno, PR.titulo, PR.anio FROM Profesor P
    JOIN Investigador I ON
    P.nombre = I.nombre AND P.paterno = I.paterno AND P.materno = I.materno
    AND P.email = I.email
    JOIN Produccion PR ON PR.investigador = I.idInvestigador
    WHERE P.paterno=aPaterno AND P.materno=aMaterno AND P.nombre=pNombre AND P.email=pEmail;
END //
DELIMITER ;

call ObtenerProduccionesProfesor('Gonz lez','Mart nez','Laura','laura.gm@example.com');
```

	idInvestigador	nombre	paterno	materno	titulo	anio
1	51	Laura	González	Martínez	Impacto del Cambio Climático en Zonas Áridas	2020

Figure 5: Respuesta a la consulta

Listing 31: Stored Procedure cuerpos académicos de un Investigador

```
DELIMITER //
CREATE PROCEDURE CuerposAcademicosInvestigador(IN aPaterno varchar(80),IN aMaterno varchar(80),
IN iNombre varchar(80),IN iEmail varchar(250))
BEGIN
    SELECT I.nombre, I.paterno, I.materno, CA.nombre FROM Investigador I
    JOIN Integrante INTE ON
    I.nombre = INTE.nombre AND I.paterno = INTE.paterno AND I.materno = INTE.materno
    AND I.email = INTE.email
    JOIN CuerpoAcademico CA ON CA.integrante = INTE.idIntegrante
    WHERE I.paterno=aPaterno AND I.materno=aMaterno AND I.nombre=iNombre AND I.email=iEmail;
END //
DELIMITER ;

call CuerposAcademicosInvestigador('Gonz lez','Mart nez','Laura','laura.gm@example.com');
```

	nombre	paterno	materno	nombre
1	Laura	González	Martínez	Cuerpo Académico de Sociología y Educación

1 row

Figure 6: Respuesta a la consulta

Listing 32: Stored Procedure obtener alumnos de un integrante de cuerpo académico

```
DELIMITER //
CREATE PROCEDURE ObtenerAlumnosIntegrante(IN aPaterno varchar(80),IN aMaterno varchar(80),IN iNo
BEGIN
    SELECT A.numeroCuenta,A.nombre, A.paterno, A.materno FROM Alumno A
    JOIN AlumnoCurso AC ON A.idAlumno = AC.alumno
    JOIN Curso C ON C.idCurso = AC.curso
    JOIN Profesor P ON P.idProfesor = C.profesor
    JOIN Integrante I ON
    I.nombre = P.nombre AND I.paterno = P.paterno AND I.materno = P.materno AND I.email = P.emai
    WHERE I.paterno=aPaterno AND I.materno=aMaterno AND I.nombre=iNombre AND I.email=iEmail;
END //
DELIMITER ;

call ObtenerAlumnosIntegrante('Gonz lez','Mart nez','Laura','laura.gm@example.com');
```

	numeroCuenta	nombre	paterno	materno
1	CU00050	Brenda	Navarrete	Alvarado
2	CU00045	Rodrigo	Gallegos	Esquivel

2 rows

Figure 7: Respuesta a la consulta

TRIGGERS

Listing 33: Insertar profesor en dataWarehouse al ingresar un profesor nuevo en nodo profesor

```
DELIMITER //
    CREATE TRIGGER insertar_profesor
    AFTER INSERT ON Profesor
    FOR EACH ROW
    BEGIN
        DECLARE idProfesor int;
        DECLARE paterno,materno,nombre,email varchar(250);
        DECLARE fechaIngreso date;

        SET idProfesor = NEW.idProfesor;
        SET paterno = NEW.paterno;
        SET materno = NEW.materno;
        SET nombre = NEW.nombre;
        SET email = NEW.email;
        SET fechaIngreso = NEW.fechaIngreso;

        INSERT INTO dataWarehouse.Profesor
        VALUES(idProfesor,paterno,materno,nombre,email,fechaIngreso);
    end //
DELIMITER ;
```

Listing 34: Actualizar dataWarehouse al actualizar un profesor en nodo profesor

```
DELIMITER //
    CREATE TRIGGER actualizar_profesor
    AFTER UPDATE ON Profesor
    FOR EACH ROW
```

```
    BEGIN
        UPDATE dataWarehouse.Profesor
        SET idProfesor = NEW.idProfesor,
            paterno = NEW.paterno,
            materno = NEW.materno,
            nombre = NEW.nombre,
            email = NEW.email,
            fechaIngreso = NEW.fechaIngreso
        WHERE idProfesor = OLD.idProfesor;
    end //
DELIMITER ;
```

Listing 35: Eliminar profesor en dataWarehouse al eliminar un profesor en nodo profesor

```
DELIMITER //
    CREATE TRIGGER eliminar_profesor
    AFTER DELETE ON Profesor
    FOR EACH ROW
    BEGIN
        DELETE FROM dataWarehouse.Profesor
        WHERE idProfesor = OLD.idProfesor;
    END //
DELIMITER ;
```

Listing 36: Insertar investigador en dataWarehouse al ingresar un investigador nuevo en nodo investigador

```
DELIMITER //
    CREATE TRIGGER insertar_investigador
    AFTER INSERT ON Investigador
    FOR EACH ROW
    BEGIN
        DECLARE idInvestigador int;
        DECLARE paterno,materno,nombre,email,movil varchar(250);

        SET idInvestigador = NEW.idInvestigador;
        SET paterno = NEW.paterno;
        SET materno = NEW.materno;
        SET nombre = NEW.nombre;
        SET email = NEW.email;
        SET movil = NEW.movil;

        INSERT INTO dataWarehouse.Investigador
        VALUES(idInvestigador,paterno,materno,nombre,email,orcid,movil);
    end //
DELIMITER ;
```

Listing 37: Actualizar dataWarehouse al actualizar un investigador en nodo investigador

```
DELIMITER //
    CREATE TRIGGER actualizar_investigador
    AFTER UPDATE ON Investigador
    FOR EACH ROW
    BEGIN
        UPDATE dataWarehouse.Investigador
        SET idInvestigador = NEW.idInvestigador,
            paterno = NEW.paterno,
            materno = NEW.materno,
            nombre = NEW.nombre,
            email = NEW.email,
            movil = NEW.movil
```

```
        WHERE idInvestigador = OLD.idInvestigador;
    end //
DELIMITER ;
```

Listing 38: Eliminar investigador en dataWarehouse al eliminar un investigador en nodo investigador

```
DELIMITER //
    CREATE TRIGGER eliminar_investigador
    AFTER DELETE ON Investigador
    FOR EACH ROW
    BEGIN
        DELETE FROM dataWarehouse.Investigador
        WHERE idInvestigador = OLD.idInvestigador;
    END //
DELIMITER ;
```

Listing 39: Insertar integrante en dataWarehouse al ingresar un integrante nuevo en nodo integrante

```
DELIMITER //
    CREATE TRIGGER insertar_integrante
    AFTER INSERT ON Integrante
    FOR EACH ROW
    BEGIN
        DECLARE idIntegrante int;
        DECLARE paterno,materno,nombre,email varchar(250);

        SET idIntegrante = NEW.idIntegrante;
        SET paterno = NEW.paterno;
        SET materno = NEW.materno;
        SET nombre = NEW.nombre;
        SET email = NEW.email;

        INSERT INTO dataWarehouse.Integrante
        VALUES(idIntegrante,paterno,materno,nombre,email);
    end //
DELIMITER ;
```

Listing 40: Actualizar dataWarehouse al actualizar un integrante en nodo integrante

```
DELIMITER //
    CREATE TRIGGER actualizar_integrante
    AFTER UPDATE ON Integrante
    FOR EACH ROW
    BEGIN
        UPDATE dataWarehouse.Integrante
        SET idIntegrante = NEW.idIntegrante,
            paterno = NEW.paterno,
            materno = NEW.materno,
            nombre = NEW.nombre,
            email = NEW.email
        WHERE idIntegrante = OLD.idIntegrante;
    end //
DELIMITER ;
```

Listing 41: Eliminar integrante en dataWarehouse al eliminar un integrante en nodo integrante

```
DELIMITER //
    CREATE TRIGGER eliminar_integrante
    AFTER DELETE ON Integrante
    FOR EACH ROW
```

```
    BEGIN
        DELETE FROM dataWarehouse.Integrante
        WHERE idIntegrante = OLD.idIntegrante;
    END //
DELIMITER ;
```

API con Flask

El Data Warehouse sólo servirá para realizar consultas, no se podrá usar para hacer operaciones de inserción, actualización o eliminación de registros, ya que esto puede generar incosistencia en los datos en los nodos locales. En tal caso, las operaciones CRUD podrán hacerce en los LCS y ejecutar triggers para actualizar la información en tiempo real en el GCS si es necesario.

Realicé una API muy sencilla para mostrar como funcionarían las consultas al Data Warehouse y cómo se ejecutan los Stored Procedures de MySQL en código de Python Flask.

Listing 42: Configuración inicial

```
from flask import Flask, jsonify, request
from flask_restx import Api,Resource,fields
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import text

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:Avs2020.@localhost
    :3306/dataWarehouse'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy()
db.init_app(app)
api = Api(app, version='1.0', title='Agencia de Viajes API', doc='/swagger')
```

Listing 43: Definimos Espacios de Nombre para la API

```
alumnos_ns = api.namespace('alumnos', description='Operaciones con alumnos')
investigadores_ns = api.namespace('investigadores', description='Operaciones con
    investigadores')
```

Listing 44: Definimos los modelos de MySQL con SQLAlchemy

```
class Alumno(db.Model):
    __tablename__ = 'Alumno'
    idAlumno = db.Column(db.Integer, primary_key=True)
    numeroCuenta = db.Column(db.String(25))
    paterno = db.Column(db.String(80))
    materno = db.Column(db.String(80))
    nombre = db.Column(db.String(80))
    email = db.Column(db.String(120))

    @classmethod
    def serialize_from_row(cls, row):
        return {
            "numeroCuenta": row[0],
            "nombre": row[1],
            "paterno": row[2],
            "materno": row[3]
        }

```

```
    def serialize(self):
        return {
            "idAlumno": self.idAlumno,
            "numeroCuenta": self.numeroCuenta,
            "paterno": self.paterno,
            "materno": self.materno,
            "nombre": self.nombre,
            "email": self.email
        }


class Investigador(db.Model):
    __tablename__ = 'Investigador'
    idInvestigador = db.Column(db.Integer, primary_key=True)
    paterno = db.Column(db.String(80))
    materno = db.Column(db.String(80))
    nombre = db.Column(db.String(80))
    email = db.Column(db.String(250))
    orcid = db.Column(db.String(30))
    movil = db.Column(db.String(15))

    def serialize(self):
        return {
            "idInvestigador": self.idInvestigador,
            "paterno": self.paterno,
            "materno": self.materno,
            "nombre": self.nombre,
            "email": self.email,
            "orcid": self.orcid,
            "movil": self.movil
        }
```

Listing 45: Endpoints relacionados con alumnos

```
@alumnos_ns.route("/<int:id>")
@alumnos_ns.param('id', 'Id del alumno')
class obtenerAlumno(Resource):
    def get(self, id):
        alumno = db.session.query(Alumno).filter_by(idAlumno=id).first()
        if not alumno:
            return {"error": "Alumno no encontrado"}, 404
        return jsonify(alumno.serialize())


@alumnos_ns.route('/alumnosDeIntegrante/<paterno>/<materno>/<nombre>/<email>')
class obtenerAlumnosIntegrante(Resource):
    def get(self, paterno, materno, nombre, email):
        try:
            with db.engine.connect() as connection:
                result = connection.execute(
                    text("CALL ObtenerAlumnosIntegrante(:aPaterno, :aMaterno, :
                        pNombre, :pEmail)"),
                    {
                        "aPaterno": paterno,
                        "aMaterno": materno,
                        "pNombre": nombre,
                        "pEmail": email
                    }
                )
```

```
                alumnos = [Alumno.serialize_from_row(row) for row in result]
                return alumnos, 200
        except Exception as e:
            return {"error": str(e)}, 500


@alumnos_ns.route("/allAlumnos")
class ListarAlumnos(Resource):
    def get(self):
        alumnos = db.session.query(Alumno).all()
        if not alumnos:
            return {"error": "Ningun alumno encontrado"}, 404
        alumnos_serializados = [alumno.serialize() for alumno in alumnos]
        return jsonify({'alumnos':alumnos_serializados})
```

Listing 46: Endpoints relacionados con investigadores

```
@investigadores_ns.route('/producciones/<paterno>/<materno>/<nombre>/<email>')
class obtenerProducciones(Resource):
    def get(self, paterno, materno, nombre, email):
        try:
            with db.engine.connect() as connection:
                result = connection.execute(
                    text("CALL ObtenerProduccionesProfesor(:aPaterno, :aMaterno, :
                        pNombre, :pEmail)"),
                    {
                        "aPaterno": paterno,
                        "aMaterno": materno,
                        "pNombre": nombre,
                        "pEmail": email
                    }
                )
                producciones = [Produccion.serialize_from_row(row) for row in
                    result]
                return producciones, 200
        except Exception as e:
            return {"error": str(e)}, 500

@investigadores_ns.route('/cuerposAcademicos/<paterno>/<materno>/<nombre>/<email>'
    )
class obtenerCuerposAcademicos(Resource):
    def get(self, paterno, materno, nombre, email):
        try:
            with db.engine.connect() as connection:
                result = connection.execute(
                    text("CALL CuerposAcademicosInvestigador(:aPaterno, :aMaterno,
                         :pNombre, :pEmail)"),
                    {
                        "aPaterno": paterno,
                        "aMaterno": materno,
                        "pNombre": nombre,
                        "pEmail": email
                    }
                )
                cuerposAcademicos = [CuerpoAcademico.serialize_from_row(row) for
                    row in result]
                return cuerposAcademicos, 200
        except Exception as e:
            return {"error": str(e)}, 500
```

Listing 47: Inicialización de la aplicación

```
if __name__ == "__main__":
    app.run(debug=True)
```

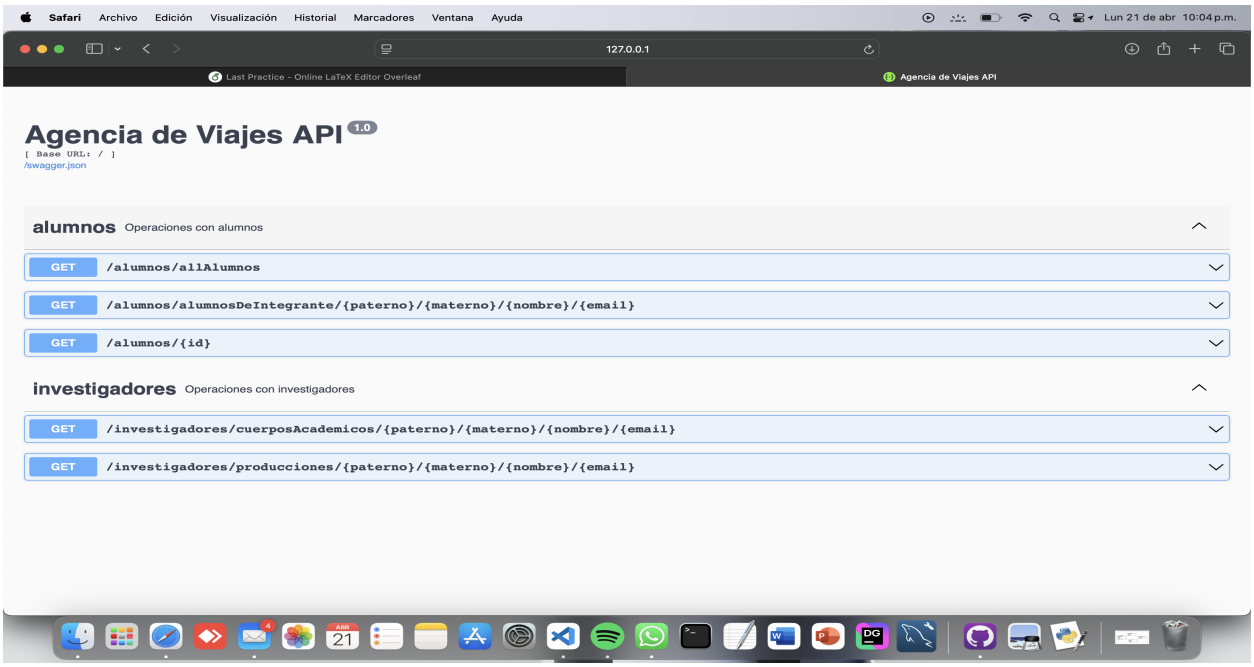

Figure 8: Documentación Swagger

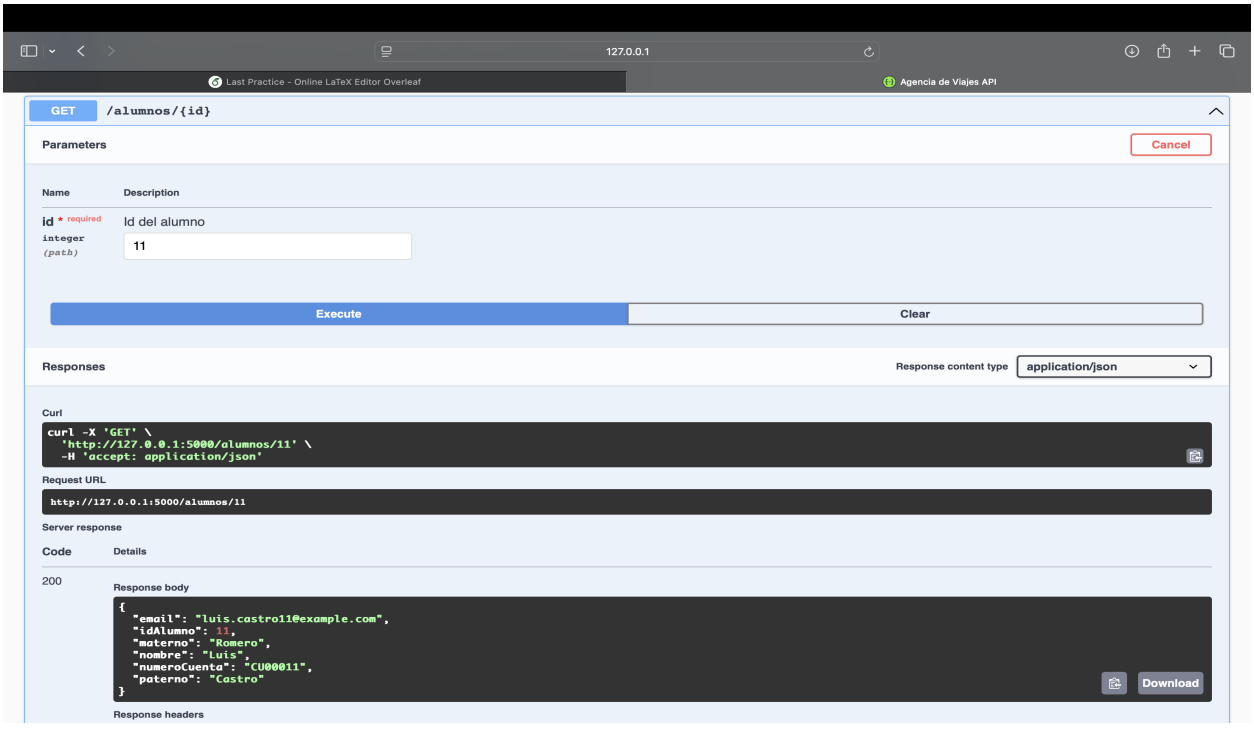

Figure 9: Ejecución endpoint con idAlumno

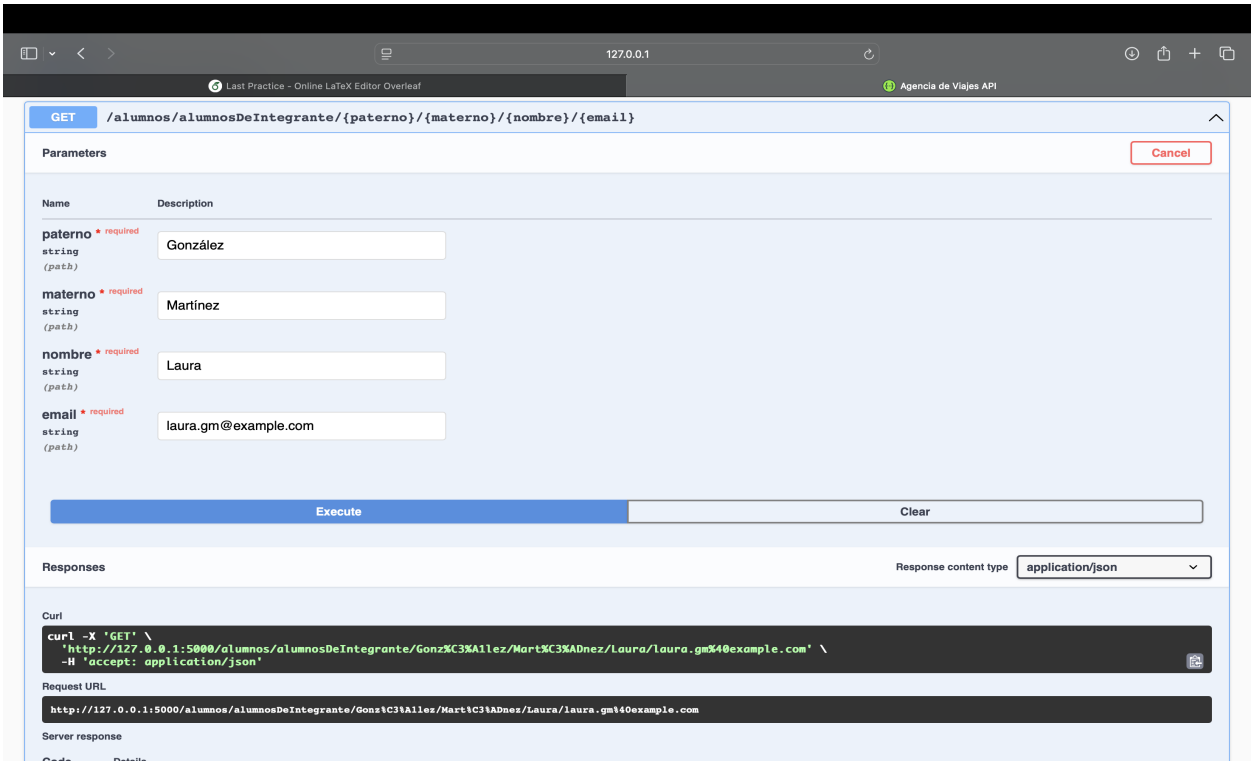

Figure 10: 1ra parte ejecución de Stored Procedure

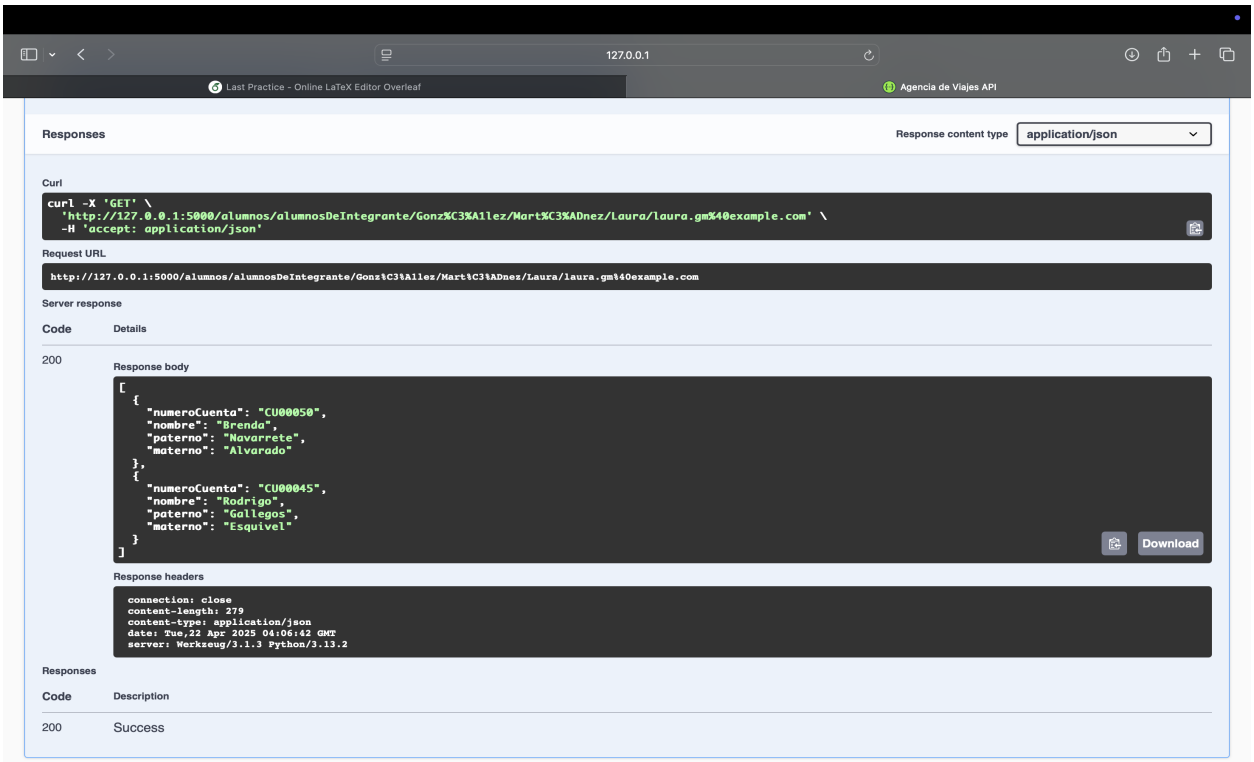

Figure 11: 2da parte ejecución de Stored Procedure

5. Conclusiones

Al realizar esta práctica, aprendí a utilizar la estrategia BOTTOM-UP, es decir, de 3 bases de datos que ya existen creamos una global que reuniera todos los datos de las 3 bases de datos.

Utilizamos procesos ETL para extraer, transformar y cargar la información de los LCS en el GCS.

Usamos select into outfile, load data infile, usamos consultas y de manera adicional agregué el uso de triggers para automatizar inserciones, actualizaciones y eliminaciones que ocurrieran en las Bases de Datos Locales, y replicar esas operaciones en la Base de Datos Global.

Referencias Bibliográficas

References

[1] Elmasri, R., Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson.

[2] Özsu, M. T., Valduriez, P. (2020). Principles of Distributed Database Systems (4th ed.). Springer.