REPORTE DE PRÁCTICA

PRACTICA INTEGRACIÓN CON ESTRATEGIA BUTTOM UP

ALUMNO:

Morales Vázquez Rubén

1. Introducción

En esta practica abordaremos el tema de button up el cual es una metodologia de creacion de bases de datos, ademas de aplicar los procesos ETL a cada base de datos; los procesos ETL (Extract, Transform, Load), esenciales para la integración y transformación de datos desde múltiples fuentes; el uso de SELECT + INTO FILE y LOAD, que facilitan la exportación e importación de datos entre sistemas; y finalmente, cómo realizar consultas con SELECT en tablas de dos bases de datos diferentes, permitiendo un análisis más completo y flexible. Estos conceptos son pilares para cualquier profesional que trabaje con grandes volúmenes de información.

2. Herramientas empleadas

Las herramientas que fueron necesarias para la realización de esta practica fueron las siguientes:

1. Repositorios de Github. Plataforma web donde se encuentran nuestros apuntes donde podemos reciclar información.
2. MySQL Workbench. Es la aplicación donde logramos establecer la base de datos en lenguaje SQL.
3. Online LaTeX Editor Overleaf. Es un sitio web en el cual podemos generar nuestros documentos.

3. Marco teórico

Button up

Es una metodología para diseñar bases de datos donde se parte de los detalles específicos (atributos) para luego construir estructuras más complejas (tablas y relaciones). Básicamente, empiezas desde abajo (los datos individuales) y vas subiendo hasta definir el esquema completo.

Pasos detallados

1. **Identificar los atributos:** Se listan todos los campos o datos necesarios (ej: nombrecliente, fecha-compra, productoid).
2. **Agrupar atributos relacionados:** Se juntan en grupos lógicos (ej: atributos de "clientes", "productos", "ventas")
3. **Normalizar las relaciones:** Se aplican reglas de normalización (1FN, 2FN, 3FN, etc.) para evitar redundancias y dependencias.
4. **Definir claves primarias y foráneas:** Se asignan claves únicas (ej: clienteid) y se vinculan entre tablas.
5. **Integrar en un esquema global:** Todas las tablas normalizadas se unen en un modelo completo (Diagrama Entidad-Relación).

Ventajas de la metología Button up:

1. Ideal cuando ya tienes claros los datos concretos pero no el modelo global.
2. Evita redundancias porque se basa en normalización.
3. Flexible para sistemas con datos bien definidos.

Desventajas de la metología Button up:

1. Puede ser tedioso si no tienes claro cómo agrupar atributos.
2. Menos intuitivo que el Top-Down (que empieza con entidades generales).

Bases de datos Heterogéneas

Son sistemas que combinan diferentes tipos de bases de datos, tecnologías o formatos en un mismo entorno. A diferencia de las bases de datos homogéneas (que usan el mismo sistema, como solo MySQL o solo Oracle), las heterogéneas mezclan varias tecnologías para adaptarse a distintas necesidades.

Características principales:

- **Diversidad de sistemas:** Pueden incluir SQL (MySQL, PostgreSQL), NoSQL (MongoDB, Redis), data warehouses (BigQuery), e incluso archivos planos (CSV, Excel).
- **Datos en distintos formatos:** Estructurados (tablas), semi-estructurados (JSON, XML) o no estructurados (texto, imágenes).
- **Distribución geográfica:** A veces están en diferentes servidores o incluso en la nube.
- **Integración compleja:** Como no son uniformes, a veces se necesitan herramientas extra (middleware, APIs, ETL) para que funcionen juntas.

Ejemplos de uso:

- Una app que guarda usuarios en PostgreSQL, logs en MongoDB y reportes en Excel.
- Un sistema bancario que usa Oracle para transacciones y Hadoop para analítica.

Ventajas:

- **Flexibilidad:** Cada sistema usa lo que mejor le funciona.
- **Escalabilidad:** Puedes añadir nuevas tecnologías según crezca el proyecto.

Desventajas:

- **Complejidad:** Mantener la integración y consistencia de datos es más difícil.
- **Rendimiento:** La comunicación entre sistemas distintos puede ser lenta.

Procesos ETL (Extract, Transform, Load)

Los procesos ETL son un conjunto de operaciones que permiten extraer, transformar y cargar datos desde múltiples fuentes hacia un destino, como un almacén de datos o un sistema de análisis. Son fundamentales en la integración de datos, especialmente en entornos donde se manejan grandes volúmenes de información proveniente de sistemas heterogéneos.

1. **Extracción (Extract)** En esta fase, los datos se extraen desde diversas fuentes, como:
   - Bases de datos relacionales (MySQL, PostgreSQL, Oracle, etc.).
   - Archivos planos (CSV, Excel, JSON, XML).
   - APIs o servicios web.
   - Sistemas legacy o aplicaciones empresariales.

   El objetivo es recolectar datos crudos, sin modificar, para su posterior procesamiento.
2. **Transformación (Transform)** En esta etapa, los datos extraídos se transforman para cumplir con los requisitos del destino. Esto incluye: Limpieza de datos: Eliminar duplicados, corregir errores o manejar valores nulos.
   - Normalización: Ajustar los datos a un formato estándar (por ejemplo, fechas en un mismo formato).
   - Enriquecimiento: Agregar información adicional o calcular nuevos campos.
   - Agregación: Resumir datos para análisis (por ejemplo, sumar ventas por región).
   - Validación: Asegurar que los datos cumplan con reglas de negocio o integridad.
3. **Carga (Load)** Finalmente, los datos transformados se cargan en un sistema de destino, como:
   - Un data warehouse (por ejemplo, Amazon Redshift, Google BigQuery, Snowflake).
   - Una base de datos analítica.
   - Un lago de datos (data lake).
   - La carga puede ser:
   - Completa: Se reemplazan todos los datos existentes.
   - Incremental: Solo se añaden los datos nuevos o modificados.

Herramientas ETL Existen diversas herramientas para implementar procesos ETL, tanto comerciales como de código abierto:

- Comerciales: Informatica, Talend, Microsoft SSIS, IBM DataStage.
- Open Source: Apache NiFi, Pentaho, Airflow.
- Basadas en la nube: AWS Glue, Google Dataflow, Azure Data Factory.

Importancia de los procesos ETL

- Integración de datos: Permiten combinar información de diferentes fuentes en un solo lugar.
- Calidad de datos: Aseguran que los datos sean precisos, consistentes y confiables.
- Análisis eficiente: Facilitan la creación de informes y dashboards para la toma de decisiones.
- Automatización: Reducen el tiempo y esfuerzo manual en la gestión de datos.

SELECT + INTO OUTFILE

El comando SELECT + INTO OUTFILE en MySQL se utiliza para exportar el resultado de una consulta a un archivo externo en el servidor. Es útil cuando necesitas guardar datos en un formato legible, como un archivo CSV o texto plano, para su posterior análisis, compartir con otros sistemas o realizar copias de seguridad.

¿Para qué sirve?
Exportar datos a un archivo CSV o texto plano.

- Generar informes o backups de datos.
- Integrar datos con otras herramientas o sistemas que requieren archivos como entrada.
- Facilitar la migración de datos entre sistemas.

Sintaxis

```
SELECT columnas
INTO OUTFILE 'ruta_del_archivo'
[OPCIONES]
FROM tabla
[WHERE condiciones];

```

Listing 1: Ejemplo de código MySQL

LOAD

El comando LOAD en MySQL se utiliza para importar datos desde un archivo externo (como un CSV o un archivo de texto) hacia una tabla en la base de datos. Es especialmente útil cuando necesitas cargar grandes volúmenes de datos de manera eficiente, como en procesos de migración, integración de sistemas o actualización de bases de datos.

¿Para qué sirve?

- Importar datos desde archivos CSV o de texto.
- Cargar grandes volúmenes de datos de manera rápida y eficiente.
- Migrar datos desde otros sistemas o aplicaciones.
- Actualizar tablas con información externa.

Sintaxis

```
LOAD DATA [LOCAL] INFILE 'ruta_del_archivo'
INTO TABLE nombre_tabla
[FIELDS TERMINATED BY 'delimitador']
[ENCLOSED BY 'caracter']
[LINES TERMINATED BY 'delimitador']
[IGNORE numero_de_lineas]
[(columna1, columna2, ...)];

```

Listing 2: Ejemplo de código MySQL

Script de creación de nodos

```
CREATE DATABASE node1Integrante;
USE node1integrante;

CREATE TABLE investigador
(
  idInvestigador INT NOT NULL,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  orcid VARCHAR(30) NOT NULL,
  email VARCHAR(150) NOT NULL,
  movil VARCHAR(15) NOT NULL,
  PRIMARY KEY (idInvestigador)
);

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

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

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

CREATE TABLE proyecto
(
  idProyecto INT NOT NULL,
  nombre VARCHAR(80) NOT NULL,
  inicio DATE NOT NULL,
  final DATE NOT NULL,
  idInvestigador INT NOT NULL,
  PRIMARY KEY (idProyecto),
  FOREIGN KEY (idInvestigador) REFERENCES investigador(idInvestigador)
);


```

Listing 3: Creacion de node1Integrante

```sql
create database node2Investigador;
use node2investigador;
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,
  fehcaIngreso DATE NOT NULL,
  PRIMARY KEY (idProfesor)
);

CREATE TABLE programa
(
  idPrograma INT NOT NULL,
  nivel VARCHAR(80) NOT NULL,
  nombre VARCHAR(80) NOT NULL,
  fechaInicio DATE NOT NULL,
  PRIMARY KEY (idPrograma)
);

CREATE TABLE asignatura
(
  idAsignaura INT NOT NULL,
  semestre INT NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  idPrograma INT NOT NULL,
  PRIMARY KEY (idAsignaura),
  FOREIGN KEY (idPrograma) REFERENCES programa(idPrograma)
);

CREATE TABLE curso
(
  idCurso INT NOT NULL,
  periodo VARCHAR(15) NOT NULL,
  anio INT NOT NULL,
  grupo INT NOT NULL,
  idAsignaura INT NOT NULL,
  idProfesor INT NOT NULL,
  PRIMARY KEY (idCurso),
  FOREIGN KEY (idAsignaura) REFERENCES asignatura(idAsignaura),
  FOREIGN KEY (idProfesor) REFERENCES profesor(idProfesor)
);

CREATE TABLE alumno
(
  idAlumno INT NOT NULL,
  numeroCuenta VARCHAR(50) 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 alumnoCurso
(
  idAlumnoCurso INT NOT NULL,
  calificacion float NOT NULL,
  idAlumno INT NOT NULL,
  idCurso INT NOT NULL,
  PRIMARY KEY (idAlumnoCurso),
  FOREIGN KEY (idAlumno) REFERENCES alumno(idAlumno),
  FOREIGN KEY (idCurso) REFERENCES curso(idCurso)
);
```

Listing 4: Creacion de node2Investigador

```sql
create database node3Profesor;
use node3profesor;
CREATE TABLE integrante
(
  idIntegrante INT NOT NULL,
  paterno VARCHAR(90) NOT NULL,
  materno VARCHAR(90) NOT NULL,
  nombre VARCHAR(110) NOT NULL,
  PRIMARY KEY (idIntegrante)
);

CREATE TABLE cuerpoAcademico
(
  idCuerpoAcademico INT NOT NULL,
  nombre VARCHAR(250) NOT NULL,
  idIntegrante INT NOT NULL,
  PRIMARY KEY (idCuerpoAcademico),
  FOREIGN KEY (idIntegrante) REFERENCES integrante(idIntegrante)
);

CREATE TABLE linea
(
  idLinea INT NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  descripcion VARCHAR(500) NOT NULL,
  idCuerpoAcademico INT NOT NULL,
  PRIMARY KEY (idLinea),
  FOREIGN KEY (idCuerpoAcademico) REFERENCES cuerpoAcademico(idCuerpoAcademico)
);

CREATE TABLE integranteLinea
(
  idIntegranteLinea INT NOT NULL,
  vigente bool NOT NULL,
  inicio DATE NOT NULL,
  termino DATE NOT NULL,
  idIntegrante INT NOT NULL,
  idLinea INT NOT NULL,
  PRIMARY KEY (idIntegrante),
  FOREIGN KEY (idIntegrante) REFERENCES integrante(idIntegrante),
  FOREIGN KEY (idLinea) REFERENCES linea(idLinea)
);

```

Listing 5: Creacion de node3Profesor

Script de Base de datos Global

```
create database investigacionGlobal;
use investigacionglobal;

CREATE TABLE investigador
(
  idInvestigador INT NOT NULL,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  orcid VARCHAR(30) NOT NULL,
  email VARCHAR(150) NOT NULL,
  movil VARCHAR(15) NOT NULL,
  PRIMARY KEY (idInvestigador)
);

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


CREATE TABLE proyecto
(
  idProyecto INT NOT NULL,
  nombre VARCHAR(80) NOT NULL,
  inicio DATE NOT NULL,
  final DATE NOT NULL,
  idInvestigador INT NOT NULL,
  PRIMARY KEY (idProyecto),
  FOREIGN KEY (idInvestigador) REFERENCES investigador(idInvestigador)
);


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,
  fehcaIngreso DATE NOT NULL,
  PRIMARY KEY (idProfesor)
);


CREATE TABLE curso
(
  idCurso INT NOT NULL,
  periodo VARCHAR(15) NOT NULL,
  anio INT NOT NULL,
  grupo INT NOT NULL,
  idAsignaura INT NOT NULL,
  idProfesor INT NOT NULL,
  PRIMARY KEY (idCurso),
  FOREIGN KEY (idProfesor) REFERENCES profesor(idProfesor)
);
```

```sql
CREATE TABLE alumno
(
  idAlumno INT NOT NULL,
  numeroCuenta VARCHAR(50) 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 integrante
(
  idIntegrante INT NOT NULL,
  paterno VARCHAR(90) NOT NULL,
  materno VARCHAR(90) NOT NULL,
  nombre VARCHAR(110) NOT NULL,
  PRIMARY KEY (idIntegrante)
);

CREATE TABLE cuerpoAcademico
(
  idCuerpoAcademico INT NOT NULL,
  nombre VARCHAR(250) NOT NULL,
  idIntegrante INT NOT NULL,
  PRIMARY KEY (idCuerpoAcademico),
  FOREIGN KEY (idIntegrante) REFERENCES integrante(idIntegrante)
);

CREATE TABLE linea
(
  idLinea INT NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  descripcion VARCHAR(500) NOT NULL,
  idCuerpoAcademico INT NOT NULL,
  PRIMARY KEY (idLinea),
  FOREIGN KEY (idCuerpoAcademico) REFERENCES cuerpoAcademico(idCuerpoAcademico)
);

```

Listing 6: Creacion de base de datos global

Script de extracción de datos

```
USE node1Integrante;

SELECT idInvestigador, paterno, materno, nombre, orcid, email, movil
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/investigadores_node1.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM investigador;

SELECT idProduccion, tipo, titulo, anio, idInvestigador
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/produccion_node1.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM produccion;

SELECT idProyecto, nombre, inicio, final, idInvestigador
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/proyectos_node1.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM proyecto;

```

Listing 7: Extraccion para node1Integrantes

```
USE node2investigador;

SELECT idProfesor, paterno, materno, nombre, email, fehcaIngreso
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/profesores_node2.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM profesor;

SELECT idCurso, periodo, anio, grupo, idAsignaura, idProfesor
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cursos_node2.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM curso;

SELECT idAlumno, numeroCuenta, paterno, materno, nombre, email
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/alumnos_node2.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM alumno;

```

Listing 8: Extraccion para node2Investigador

```
USE node3Profesor;
SELECT idIntegrante, paterno, materno, nombre
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/integrantes_node3.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM integrante;

SELECT idCuerpoAcademico, nombre, idIntegrante
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cuerpos_academicos_node3.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM cuerpoAcademico;

SELECT idLinea, nombre, descripcion, idCuerpoAcademico
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/lineas_node3.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM linea;

```

Listing 9: Extraccion para node3Profesor

Script de carga de datos

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/investigadores_node1.txt'
INTO TABLE investigador
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/produccion_node1.txt'
INTO TABLE produccion
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/proyectos_node1.txt'
INTO TABLE proyecto
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

```

Listing 10: Carga de datos de node1Integrante

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/profesores_node2.txt'
INTO TABLE profesor
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cursos_node2.txt'
INTO TABLE curso
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/alumnos_node2.txt'
INTO TABLE alumno
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

```

Listing 11: Carga de datos node2Investigador

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/integrantes_node3.txt'
INTO TABLE integrante
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cuerpos_academicos_node3.txt
    '
INTO TABLE cuerpoAcademico
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/lineas_node3.txt'
INTO TABLE linea
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

```

Listing 12: Carga de datos node3Profesor

Script de consulta de datos

```sql
SELECT p.tipo, p.titulo, p.anio, i.nombre, i.paterno
FROM produccion p
JOIN investigador inv ON p.idInvestigador = inv.idInvestigador
JOIN integrante i ON CONCAT(inv.nombre, ' ', inv.paterno) = CONCAT(i.nombre, ' ', i.paterno)
LIMIT 10;
```

Listing 13: Ver Producción académica vinculada a integrantes de cuerpos académicos

```sql
SELECT 'Investigadores' AS tipo, COUNT(*) AS total FROM investigador
UNION ALL
SELECT 'Profesores', COUNT(*) FROM profesor
UNION ALL
SELECT 'Integrantes CA', COUNT(*) FROM integrante;
```

Listing 14: Ver Conteo total de registros por cada origen

```sql
SELECT p.nombre, p.inicio, p.final, i.nombre AS investigador
FROM proyecto p
JOIN investigador i ON p.idInvestigador = i.idInvestigador
WHERE p.final > CURDATE();
```

Listing 15: Ver Proyectos de investigación activos

4. Conclusiones

En esta práctica, lo que desarrollamos fue lograr establecer una base de datos distribuidas pequeña con tres nodos en los cuales utilizamos la metodología de Button Up , utilizamos el proceso ETL, en el cual tenemos que utilizar y experimentar con la consulta de SELECT INTO OUTLFILE y LOAD el cual consiste en extraer datos de una base de datos y lograr insertarlos en una nueva base de datos, logrando asi la base de datos distribuida con datos insertados de otras bases de datos.

References

[1] Giménez, J. A. (**2019**). *Buenas prácticas en el diseño de bases de datos.* ARANDU UTIC, *6*(1), 193–210.

[2] Mosquera Palacios, D. J., & Wanumen Silva, L. F. (**2018**). *Arquitectura para la generación de consultas SQL usando lógica de conjuntos.* Visión Electrónica, *2*, 307–318.

[3] Date, C. J. (**2004**). An Introduction to Database Systems *8th ed Addison-Wesley*

[4] Elmasri, R. Navathe, S. B. (**2016**). Fundamentals of Database Systems *11th ed Pearson*