1. Introducción

La práctica tiene como objetivo implementar una arquitectura distribuida de bases de datos mediante la fragmentación vertical utilizando una estrategia Bottom-Up. Esta estrategia consiste en integrar bases de datos especializadas en una base de datos global o completa. Para este ejemplo, se integrarán tres áreas académicas: Investigadores, Profesores e Integrantes de Cuerpos Académicos, en una base de datos universitaria completa.

2. Marco teórico

Fragmentación Vertical

La fragmentación vertical consiste en dividir las tablas de una base de datos en columnas, creando tablas más especializadas. En el enfoque Bottom-Up, primero se crean bases de datos independientes que luego se integran en una base de datos completa.

Estrategia Bottom-Up

La estrategia Bottom-Up para la fragmentación de bases de datos sigue estos pasos:

- Diseñar e implementar bases de datos locales independientes
- Identificar relaciones entre las entidades de diferentes bases de datos
- Crear una base de datos global que integre todas las locales
- Transferir datos desde las bases locales a la global
- Mantener la consistencia entre las bases locales y la global

Procesos ETL en la Estrategia Bottom-Up

Los procesos ETL (Extracción, Transformación y Carga) son fundamentales para una estrategia Bottom-Up:

- **Extracción:** Seleccionar datos de las bases locales
- **Transformación:** Adaptar los datos al esquema de la base global
- **Carga:** Insertar los datos en la base global

3. Esquemas Conceptuales Locales

Base de Datos Investigador

Esta base contiene información sobre investigadores, sus adscripciones, formación académica, proyectos y producción científica.

Base de Datos Profesor

Esta base gestiona información sobre profesores, programas académicos, asignaturas, cursos y alumnos.

Base de Datos Integrante

Esta base almacena información sobre integrantes de cuerpos académicos, cuerpos académicos, líneas de investigación y la participación de integrantes en líneas.

4. Script de Creación de Bases de Datos Locales

Creación de Bases de Datos

Listing 1: Creación de las bases de datos

```
-- Crear bases de datos locales
CREATE DATABASE Investigador;
CREATE DATABASE Profesor;
CREATE DATABASE Integrante;
-- Crear base de datos global
CREATE DATABASE Universidad;
```

Creación de Tablas en cada Base de Datos Local

4.2.1 Base de Datos Investigador

Listing 2: Creación de tablas en Investigador

```
USE Investigador;

CREATE TABLE Investigador (
  idInvestigador INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(80) NOT NULL,
  ORCID VARCHAR(150) NOT NULL,
  email VARCHAR(150) NOT NULL,
  movil VARCHAR(15) NOT NULL
);

CREATE TABLE Adscripcion (
  idAdscripcion INT AUTO_INCREMENT PRIMARY KEY,
  idInvestigador INT NOT NULL,
  instituto VARCHAR(80) NOT NULL,
  area VARCHAR(150) NOT NULL,
  nombramiento VARCHAR(10) NOT NULL,
  fechaIngreso DATETIME,
  FOREIGN KEY (idInvestigador) REFERENCES Investigador (idInvestigador)
);

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

CREATE TABLE Proyecto (
  idProyecto INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(250) NOT NULL,
```

```
  inicion DATE NOT NULL,
  final DATE NOT NULL,
  idInvestigador INT NOT NULL,
  FOREIGN KEY (idInvestigador) REFERENCES Investigador (idInvestigador)
);

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

4.2.2 Base de Datos Profesor

Listing 3: Creación de tablas en Profesor

```
USE Profesor;

CREATE TABLE Profesor (
  idProfesor INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(255) NOT NULL,
  email VARCHAR(120) NOT NULL,
  fechaIngreso DATE NOT NULL
);

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

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

CREATE TABLE Asignatura (
  idAsignatura INT AUTO_INCREMENT PRIMARY KEY,
  semestre INT NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  idPrograma INT NOT NULL,
  FOREIGN KEY (IdPrograma) REFERENCES Programa (IdPrograma)
);
```

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

CREATE TABLE AlumnoCurso (
  idAlumnoCurso INT AUTO_INCREMENT PRIMARY KEY,
  idAlumno INT NOT NULL,
  curso INT NOT NULL,
  calificacion DECIMAL(4, 2) NOT NULL,
  FOREIGN KEY (IdAlumno) REFERENCES Alumno (IdAlumno),
  FOREIGN KEY (Curso) REFERENCES Curso (IdCurso)
);
```

4.2.3 Base de Datos Integrante

Listing 4: Creación de tablas en Integrante

```
USE Integrante;

CREATE TABLE Integrante (
  idIntegrante INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(90) NOT NULL,
  materno VARCHAR(90) NOT NULL,
  nombre VARCHAR(110) NOT NULL
);

CREATE TABLE CuerpoAcademico (
  idCuerpo INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(250) NOT NULL,
  idLider INT NOT NULL,
  FOREIGN KEY (IdLider) REFERENCES Integrante (IdIntegrante)
);

CREATE TABLE Linea (
  idLinea INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(120) NOT NULL,
  descripcion VARCHAR(500) NOT NULL,
  cuerpo INT NOT NULL,
  FOREIGN KEY (Cuerpo) REFERENCES CuerpoAcademico (IdCuerpo)
);

CREATE TABLE IntegranteLinea (
  idIntegranteLinea INT AUTO_INCREMENT PRIMARY KEY,
  integrante INT NOT NULL,
  linea INT NOT NULL,
  vigente BOOLEAN NOT NULL,
  inicio DATE NOT NULL,
```

```
  termino DATE NOT NULL,
  FOREIGN KEY (integrante) REFERENCES Integrante (IdIntegrante),
  FOREIGN KEY (linea) REFERENCES Linea (IdLinea)
);
```

5. Creación de la Base de Datos Global

Creación de Tablas en la Base de Datos Universidad

Listing 5: Creación de tablas en Universidad

```
USE Universidad;

-- Tablas de Investigador
CREATE TABLE Investigador (
  idInvestigador INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(80) NOT NULL,
  ORCID VARCHAR(150) NOT NULL,
  email VARCHAR(150) NOT NULL,
  movil VARCHAR(15) NOT NULL
);

CREATE TABLE Adscripcion (
  idAdscripcion INT AUTO_INCREMENT PRIMARY KEY,
  idInvestigador INT NOT NULL,
  instituto VARCHAR(80) NOT NULL,
  area VARCHAR(150) NOT NULL,
  nombramiento VARCHAR(10) NOT NULL,
  fechaIngreso DATETIME,
  FOREIGN KEY (idInvestigador) REFERENCES Investigador (idInvestigador)
);

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

CREATE TABLE Proyecto (
  idProyecto INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(250) NOT NULL,
  inicion DATE NOT NULL,
  final DATE NOT NULL,
  idInvestigador INT NOT NULL,
  FOREIGN KEY (idInvestigador) REFERENCES Investigador (idInvestigador)
);
```

```sql
CREATE TABLE Produccion (
  idProduccion INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(60) NOT NULL,
  titulo VARCHAR(250) NOT NULL,
  anio INT NOT NULL,
  idInvestigador INT NOT NULL,
  FOREIGN KEY (idInvestigador) REFERENCES Investigador (idInvestigador)
);

-- Tablas de Profesor
CREATE TABLE Profesor (
  idProfesor INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(80) NOT NULL,
  materno VARCHAR(80) NOT NULL,
  nombre VARCHAR(255) NOT NULL,
  email VARCHAR(120) NOT NULL,
  fechaIngreso DATE NOT NULL
);

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

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

CREATE TABLE Asignatura (
  idAsignatura INT AUTO_INCREMENT PRIMARY KEY,
  semestre INT NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  idPrograma INT NOT NULL,
  FOREIGN KEY (IdPrograma) REFERENCES Programa (IdPrograma)
);

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

```
CREATE TABLE AlumnoCurso (
  idAlumnoCurso INT AUTO_INCREMENT PRIMARY KEY,
  idAlumno INT NOT NULL,
  curso INT NOT NULL,
  calificacion DECIMAL(4, 2) NOT NULL,
  FOREIGN KEY (IdAlumno) REFERENCES Alumno (IdAlumno),
  FOREIGN KEY (Curso) REFERENCES Curso (IdCurso)
);

-- Tablas de Integrante
CREATE TABLE Integrante (
  idIntegrante INT AUTO_INCREMENT PRIMARY KEY,
  paterno VARCHAR(90) NOT NULL,
  materno VARCHAR(90) NOT NULL,
  nombre VARCHAR(110) NOT NULL
);

CREATE TABLE CuerpoAcademico (
  idCuerpo INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(250) NOT NULL,
  idLider INT NOT NULL,
  FOREIGN KEY (IdLider) REFERENCES Integrante (IdIntegrante)
);

CREATE TABLE Linea (
  idLinea INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(120) NOT NULL,
  descripcion VARCHAR(500) NOT NULL,
  cuerpo INT NOT NULL,
  FOREIGN KEY (Cuerpo) REFERENCES CuerpoAcademico (IdCuerpo)
);

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

6. Scripts de Extracción y Carga de Datos (ETL)

Transferencia de Datos de Bases Locales a Base Global

Listing 6: Transferencia de datos de Investigador a Universidad

```
-- Transferencia de tabla Investigador
INSERT INTO Universidad.Investigador (idInvestigador, paterno, materno, nombre, ORCID, ema
SELECT idInvestigador, paterno, materno, nombre, ORCID, email, movil
FROM Investigador.Investigador;
```

```
-- Transferencia de tabla Adscripcion
INSERT INTO Universidad.Adscripcion (idAdscripcion, idInvestigador, instituto, area, nombr
SELECT idAdscripcion, idInvestigador, instituto, area, nombramiento, fechaIngreso
FROM Investigador.Adscripcion;

-- Transferencia de tabla Formacion
INSERT INTO Universidad.Formacion (idFormacion, idInvestigador, grado, institucion, nombre
SELECT idFormacion, idInvestigador, grado, institucion, nombre, fechaTermino
FROM Investigador.Formacion;

-- Transferencia de tabla Proyecto
INSERT INTO Universidad.Proyecto (idProyecto, nombre, inicion, final, idInvestigador)
SELECT idProyecto, nombre, inicion, final, idInvestigador
FROM Investigador.Proyecto;

-- Transferencia de tabla Produccion
INSERT INTO Universidad.Produccion (idProduccion, tipo, titulo, anio, idInvestigador)
SELECT idProduccion, tipo, titulo, anio, idInvestigador
FROM Investigador.Produccion;
```

Listing 7: Transferencia de datos de Profesor a Universidad

```
-- Transferencia de tabla Profesor
INSERT INTO Universidad.Profesor (idProfesor, paterno, materno, nombre, email, fechaIngreso
SELECT idProfesor, paterno, materno, nombre, email, fechaIngreso
FROM Profesor.Profesor;

-- Transferencia de tabla Programa
INSERT INTO Universidad.Programa (idPrograma, nivel, nombre, fechaInicio)
SELECT idPrograma, nivel, nombre, fechaInicio
FROM Profesor.Programa;

-- Transferencia de tabla Alumno
INSERT INTO Universidad.Alumno (idAlumno, numeroCuenta, paterno, materno, nombre, email)
SELECT idAlumno, numeroCuenta, paterno, materno, nombre, email
FROM Profesor.Alumno;

-- Transferencia de tabla Asignatura
INSERT INTO Universidad.Asignatura (idAsignatura, semestre, nombre, idPrograma)
SELECT idAsignatura, semestre, nombre, idPrograma
FROM Profesor.Asignatura;

-- Transferencia de tabla Curso
INSERT INTO Universidad.Curso (idCurso, periodo, anio, idAsignatura, grupo, profesor)
SELECT idCurso, periodo, anio, idAsignatura, grupo, profesor
FROM Profesor.Curso;

-- Transferencia de tabla AlumnoCurso
INSERT INTO Universidad.AlumnoCurso (idAlumnoCurso, idAlumno, curso, calificacion)
SELECT idAlumnoCurso, idAlumno, curso, calificacion
FROM Profesor.AlumnoCurso;
```

Listing 8: Transferencia de datos de Integrante a Universidad

```
-- Transferencia de tabla Integrante
```

```sql
INSERT INTO Universidad.Integrante (idIntegrante, paterno, materno, nombre)
SELECT idIntegrante, paterno, materno, nombre
FROM Integrante.Integrante;

-- Transferencia de tabla CuerpoAcademico
INSERT INTO Universidad.CuerpoAcademico (idCuerpo, nombre, idLider)
SELECT idCuerpo, nombre, idLider
FROM Integrante.CuerpoAcademico;

-- Transferencia de tabla Linea
INSERT INTO Universidad.Linea (idLinea, nombre, descripcion, cuerpo)
SELECT idLinea, nombre, descripcion, cuerpo
FROM Integrante.Linea;

-- Transferencia de tabla IntegranteLinea
INSERT INTO Universidad.IntegranteLinea (idIntegranteLinea, integrante, linea, vigente, ini
SELECT idIntegranteLinea, integrante, linea, vigente, inicio, termino
FROM Integrante.IntegranteLinea;
```

7. Consultas Cruzadas entre Entidades

Consulta entre Profesor e Investigador

Listing 9: Consulta de profesores que también son investigadores

```sql
SELECT p.idProfesor, p.paterno, p.materno, p.nombre, p.email,
       i.ORCID, i.movil, a.instituto, a.area, a.nombramiento
FROM Universidad.Profesor p
JOIN Universidad.Investigador i ON p.paterno = i.paterno AND p.materno = i.materno AND p.no
JOIN Universidad.Adscripcion a ON i.idInvestigador = a.idInvestigador;
```

Consulta entre Investigador e Integrante

Listing 10: Consulta de investigadores que son integrantes de cuerpos académicos

```sql
SELECT i.idInvestigador, i.paterno, i.materno, i.nombre, i.ORCID,
       int.idIntegrante, ca.nombre AS CuerpoAcademico, l.nombre AS LineaInvestigacion
FROM Universidad.Investigador i
JOIN Universidad.Integrante int ON i.paterno = int.paterno AND i.materno = int.materno AND
JOIN Universidad.IntegranteLinea il ON int.idIntegrante = il.integrante
JOIN Universidad.Linea l ON il.linea = l.idLinea
JOIN Universidad.CuerpoAcademico ca ON l.cuerpo = ca.idCuerpo
WHERE il.vigente = TRUE;
```

Consulta entre las Tres Entidades

Listing 11: Consulta integrando las tres entidades

```sql
SELECT p.idProfesor, p.paterno, p.materno, p.nombre,
       i.idInvestigador, i.ORCID,
       int.idIntegrante,
       ca.nombre AS CuerpoAcademico,
       COUNT(DISTINCT c.idCurso) AS CursosDictados,
```

```
        COUNT(DISTINCT pr.idProyecto) AS ProyectosDirigidos
FROM Universidad.Profesor p
LEFT JOIN Universidad.Investigador i ON p.paterno = i.paterno AND p.materno = i.materno AN
LEFT JOIN Universidad.Integrante int ON p.paterno = int.paterno AND p.materno = int.materno
LEFT JOIN Universidad.CuerpoAcademico ca ON int.idIntegrante = ca.idLider
LEFT JOIN Universidad.Curso c ON p.idProfesor = c.profesor
LEFT JOIN Universidad.Proyecto pr ON i.idInvestigador = pr.idInvestigador
GROUP BY p.idProfesor, i.idInvestigador, int.idIntegrante, ca.nombre;
```

8. Vistas de Integración Bottom-Up

Vista de Profesores-Investigadores

Listing 12: Vista de profesores-investigadores

```
CREATE VIEW Universidad.Vista_ProfesorInvestigador AS
SELECT
    p.idProfesor,
    i.idInvestigador,
    p.paterno,
    p.materno,
    p.nombre,
    p.email,
    i.ORCID,
    i.movil,
    p.fechaIngreso AS fechaIngresoProfesor,
    a.fechaIngreso AS fechaIngresoInvestigador,
    a.instituto,
    a.area,
    a.nombramiento
FROM Universidad.Profesor p
JOIN Universidad.Investigador i ON p.paterno = i.paterno AND p.materno = i.materno AND p.no
LEFT JOIN Universidad.Adscripcion a ON i.idInvestigador = a.idInvestigador;
```

Vista de Producción Académica Integral

Listing 13: Vista de producción académica integral

```
CREATE VIEW Universidad.Vista_ProduccionAcademica AS
SELECT
    i.idInvestigador,
    i.paterno,
    i.materno,
    i.nombre,
    i.ORCID,
    p.idProfesor,
    int.idIntegrante,
    pr.idProyecto,
    pr.nombre AS NombreProyecto,
    pr.inicion AS InicioProyecto,
    pr.final AS FinProyecto,
    pd.idProduccion,
    pd.tipo AS TipoProduccion,
```

```
    pd.titulo AS TituloProduccion,
    pd.anio AS AnioProduccion,
    l.nombre AS LineaInvestigacion,
    ca.nombre AS CuerpoAcademico
FROM Universidad.Investigador i
LEFT JOIN Universidad.Profesor p ON i.paterno = p.paterno AND i.materno = p.materno AND i.
LEFT JOIN Universidad.Integrante int ON i.paterno = int.paterno AND i.materno = int.materno
LEFT JOIN Universidad.Proyecto pr ON i.idInvestigador = pr.idInvestigador
LEFT JOIN Universidad.Produccion pd ON i.idInvestigador = pd.idInvestigador
LEFT JOIN Universidad.IntegranteLinea il ON int.idIntegrante = il.integrante
LEFT JOIN Universidad.Linea l ON il.linea = l.idLinea
LEFT JOIN Universidad.CuerpoAcademico ca ON l.cuerpo = ca.idCuerpo;
```

9. Recomendaciones de Mantenimiento del Sistema Bottom-Up

Sincronización de Datos

Para mantener la consistencia entre las bases de datos locales y la base de datos global, se recomiendan los siguientes procesos de sincronización:

Listing 14: Procedimiento para sincronización periódica

```
-- Crear procedimiento almacenado para sincronizaci n
DELIMITER //
CREATE PROCEDURE Universidad.SincronizarDatos()
BEGIN
    -- Sincronizar Investigador
    INSERT INTO Universidad.Investigador
    SELECT * FROM Investigador.Investigador i
    WHERE NOT EXISTS (SELECT 1 FROM Universidad.Investigador u WHERE u.idInvestigador = i.

    -- Sincronizar Profesor
    INSERT INTO Universidad.Profesor
    SELECT * FROM Profesor.Profesor p
    WHERE NOT EXISTS (SELECT 1 FROM Universidad.Profesor u WHERE u.idProfesor = p.idProfes

    -- Sincronizar Integrante
    INSERT INTO Universidad.Integrante
    SELECT * FROM Integrante.Integrante i
    WHERE NOT EXISTS (SELECT 1 FROM Universidad.Integrante u WHERE u.idIntegrante = i.idInt

    -- C digo similar para el resto de las tablas
END //
DELIMITER ;

-- Crear evento para ejecuci n peri dica
CREATE EVENT Universidad.evento_sincronizacion
ON SCHEDULE EVERY 1 DAY
DO
    CALL Universidad.SincronizarDatos();
```

Triggers para Consistencia

Para mantener la consistencia en tiempo real, se pueden implementar triggers en las bases de datos locales:

Listing 15: Ejemplo de trigger para mantener consistencia

```
-- Trigger para inserci n en Investigador
DELIMITER //
CREATE TRIGGER Investigador.after_insert_investigador
AFTER INSERT ON Investigador.Investigador
FOR EACH ROW
BEGIN
    INSERT INTO Universidad.Investigador
    VALUES (NEW.idInvestigador, NEW.paterno, NEW.materno, NEW.nombre, NEW.ORCID, NEW.email
END //
DELIMITER ;

-- Trigger para actualizaci n en Investigador
DELIMITER //
CREATE TRIGGER Investigador.after_update_investigador
AFTER UPDATE ON Investigador.Investigador
FOR EACH ROW
BEGIN
    UPDATE Universidad.Investigador
    SET paterno = NEW.paterno,
        materno = NEW.materno,
        nombre = NEW.nombre,
        ORCID = NEW.ORCID,
        email = NEW.email,
        movil = NEW.movil
    WHERE idInvestigador = NEW.idInvestigador;
END //
DELIMITER ;
```

10. Conclusiones

En esta práctica se ha implementado una estrategia Bottom-Up para la fragmentación de una base de datos universitaria que:

- Permite mantener bases de datos especializadas para cada área funcional (Investigador, Profesor, Integrante)
- Facilita la integración de datos en una base de datos global (Universidad)
- Proporciona mecanismos para mantener la consistencia entre las bases locales y la global
- Permite realizar consultas cruzadas entre las diferentes entidades
- Ofrece vistas integradas para facilitar el acceso a información relacionada

La implementación de esta arquitectura distribuida mejora la eficiencia operativa al permitir que cada departamento trabaje con su base de datos local especializada, mientras que la administración central puede acceder a una visión global e integrada de todos los datos.

Referencias Bibliográficas

References

[1] Elmasri, R.; Navathe, S. (2016). *Fundamentos de Sistemas de Bases de Datos*. Pearson Educación.

[2] Silberschatz, A.; Korth, H.; Sudarshan, S. (2011). *Database System Concepts*. McGraw-Hill.

[3] Coronel, C.; Morris, S. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.

[4] Kimball, R.; Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley.