Sesión	*25*
Tema	Implementación de claves principales y foráneas.
Propósito	Fortalecer las capacidades de los participantes respecto a la implementación de claves en el contexto de desarrollo de bases de datos, mediante la resolución de casos propuestos en una sesión práctica.
Fecha	C.01.12.2025
Hora	17:00

1. Marco Teórico (Background)

Claves Primarias (Primary Keys)

- **Definición**: Un atributo o conjunto de atributos que identifican de manera única cada fila en una tabla.
- **Características**:
  - No admite valores nulos.
  - Debe ser único.
  - Generalmente se implementa con índices para optimizar búsquedas.
- **Limitaciones**:
  - Solo puede existir una clave primaria por tabla.
  - No debe cambiar con frecuencia (estabilidad).

Claves Foráneas (Foreign Keys)

- **Definición**: Un atributo en una tabla que hace referencia a la clave primaria de otra tabla, estableciendo una relación.
- **Características**:
  - Garantizan integridad referencial.
  - Pueden aceptar valores nulos (dependiendo del diseño).
  - Permiten definir acciones en cascada (ON DELETE/ON UPDATE).
- **Limitaciones**:
  - Pueden afectar el rendimiento en inserciones/actualizaciones.
  - No todos los SGBD las implementan de la misma forma (ej. MongoDB no usa claves foráneas explícitas).

🛠️ 2. Implementación en distintos SGBD

SGBD	Clave Primaria	Clave Foránea	Notas
MariaDB/MySQL	PRIMARY KEY	FOREIGN KEY ... REFERENCES	Soporta ON DELETE/UPDATE CASCADE
PostgreSQL	Igual que MySQL	Igual que MySQL	Muy robusto en integridad referencial
SQL Server	PRIMARY KEY	FOREIGN KEY	Permite múltiples restricciones en cascada
Oracle	PRIMARY KEY	FOREIGN KEY	Manejo avanzado de constraints
SQLite	PRIMARY KEY	FOREIGN KEY (requiere activar PRAGMA foreign_keys=ON)	Ligero, pero con soporte limitado
MongoDB	No usa claves foráneas	Relaciones se modelan con referencias o documentos embebidos	Integridad referencial manejada a nivel de aplicación

📄 3. Caso de Ejemplo

Enunciado del problema

Una universidad necesita gestionar **estudiantes**, **cursos** y **matrículas**.

- Cada estudiante tiene un código único.
- Cada curso tiene un código único.
- Una matrícula relaciona a un estudiante con un curso.

Requerimientos

- **Funcionales**:
  - Registrar estudiantes y cursos.
  - Registrar matrículas.
  - Consultar qué cursos lleva un estudiante.
- **No funcionales**:
  - Integridad de datos (no permitir matrículas sin estudiante o curso válido).
  - Escalabilidad para miles de registros.

- **Actores**:
  - Administrador académico.
  - Estudiante.
- **Casos de uso**:
  - Registrar estudiante.
  - Registrar curso.
  - Registrar matrícula.
  - Consultar cursos de un estudiante.

📐 4. Diseño Conceptual (Modelo Entidad-Relación)

- **Entidad Estudiante**: (id_estudiante PK, nombre, apellido, email)
- **Entidad Curso**: (id_curso PK, nombre, créditos)
- **Entidad Matrícula**: (id_matricula PK, id_estudiante FK, id_curso FK, fecha)

Relaciones:

- Estudiante 1..* Matrícula
- Curso 1..* Matrícula

🧩 5. Diseño Lógico (Tablas Relacionales)

sql

```
CREATE TABLE Estudiante (
   id_estudiante INT PRIMARY KEY,
   nombre VARCHAR(50),
   apellido VARCHAR(50),
   email VARCHAR(100) UNIQUE
);


CREATE TABLE Curso (
   id_curso INT PRIMARY KEY,
   nombre VARCHAR(100),
```

```
    creditos INT
);

CREATE TABLE Matricula (
    id_matricula INT PRIMARY KEY,
    id_estudiante INT,
    id_curso INT,
    fecha DATE,
    FOREIGN KEY (id_estudiante) REFERENCES Estudiante(id_estudiante)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (id_curso) REFERENCES Curso(id_curso)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

⚙️ **6. Diseño Físico (Implementación con DDL y DML)**

Inserción de datos de prueba

sql

```
-- Estudiantes
INSERT INTO Estudiante VALUES (1, 'Ana', 'Pérez', 'ana.perez@uni.edu');
INSERT INTO Estudiante VALUES (2, 'Luis', 'García', 'luis.garcia@uni.edu');

-- Cursos
INSERT INTO Curso VALUES (101, 'Base de Datos', 4);
INSERT INTO Curso VALUES (102, 'Programación', 5);

-- Matrículas
INSERT INTO Matricula VALUES (1001, 1, 101, '2025-03-01');
INSERT INTO Matricula VALUES (1002, 1, 102, '2025-03-01');
INSERT INTO Matricula VALUES (1003, 2, 101, '2025-03-02');
```

🔍 7. Ejemplos de Consultas

- **Cursos de un estudiante**:

sql

```
SELECT e.nombre, c.nombre
FROM Estudiante e
JOIN Matricula m ON e.id_estudiante = m.id_estudiante
JOIN Curso c ON m.id_curso = c.id_curso
WHERE e.id_estudiante = 1;
```

- **Estudiantes en un curso**:

sql

```
SELECT c.nombre, e.nombre, e.apellido
FROM Curso c
JOIN Matricula m ON c.id_curso = m.id_curso
JOIN Estudiante e ON m.id_estudiante = e.id_estudiante
WHERE c.id_curso = 101;
```

Bibliografía Recomendada

• "Fundamentos de Bases de Datos" de Abraham Silberschatz, Henry F. Korth y S. Sudarshan

• "Sistemas de Bases de Datos: un enfoque práctico" de Thomas M. Connolly y Carolyn Begg

• "Desarrollo de Bases de Datos: casos prácticos desde el análisis a la implementación" de Dolores Cuadra, Elena Castro, Ana M. Iglesias

• "Tecnología y Diseño de Bases de Datos" de Marcos, C. Calero y B. Vela

• https://docs.oracle.com/en/database/