

Sistema de Gestión Hospitalaria

```
CREATE DATABASE ejercicioOne;
```

```
USE ejercicioOne;
```

```
CREATE TABLE Paciente (
```

```
    id_paciente INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    edad INT,  
    genero VARCHAR(10),  
    direccion VARCHAR(100)
```

```
);
```

```
CREATE TABLE Medico (
```

```
    id_medico INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    especialidad VARCHAR(30),  
    telefono VARCHAR(15)
```

```
);
```

```
CREATE TABLE Cita (
```

```
    id_cita INT AUTO_INCREMENT PRIMARY KEY,  
    id_paciente INT,  
    id_medico INT,  
    fecha_cita DATE NOT NULL,  
    motivo TEXT,  
    FOREIGN KEY (id_paciente) REFERENCES Paciente (id_paciente),  
    FOREIGN KEY (id_medico) REFERENCES Medico (id_medico)
```

```
);
```

```
CREATE TABLE Historial_Clinico (
```

```
    id_historial INT AUTO_INCREMENT PRIMARY KEY,  
    id_paciente INT UNIQUE,  
    diagnostico TEXT,  
    tratamiento TEXT,  
    FOREIGN KEY (id_paciente) REFERENCES Paciente (id_paciente),
```

```
);
```

CREATE TABLE Habitacion (

Id_habitacion INT AUTO_INCREMENT PRIMARY KEY,
numero INT,
piso INT,
tipo VARCHAR(20)

)°

CREATE TABLE Medicamento (

Id_medicamento INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(50),
clasis VARCHAR(20),
descripcion TEXT

)°

CREATE TABLE Paciente_Medicamento (

Id_Paciente INT,
Id_medicamento INT,
fecha_receta DATE,
PRIMARY KEY (Id_Paciente, Id_medicamento) REFERENCES Medicamento,
FOREIGN KEY (Id_medicamento) REFERENCES Medicamento,
Cita_medicamento
FOREIGN KEY (Id_Paciente) REFERENCES Paciente(Id_Pacien

)°

CREATE TABLE cita_Habitacion (

Id_cita INT,
Id_habitacion INT,
observacion TEXT,
PRIMARY KEY (Id_cita, Id_habitacion),
FOREIGN KEY (Id_cita) REFERENCES Cita (Id_cita),
FOREIGN KEY (Id_habitacion) REFERENCES Habitacion

)°

INSERT INTO Paciente (Nombre, edad, genero, direccion)
VALUES ('Luis Conde', 17, 'masculino', 'Calle 45 #7-24');
INSERT INTO Medico (Nombre, especialidad, telefono)
VALUES ('Dra. María Rojas', 'pediatría', '3134567890');

-- INNER JOIN: citas con nombres de paciente y medico

SELECT C.id_cita, P.nombre AS paciente, M.nombre
AS medico
FROM cita C
INNER JOIN Paciente P ON C.id_Paciente = P.id_Paciente
INNER JOIN Medico M ON C.id_medico = M.id_medico;

CFT JOIN: Todas las c.tas y habilidades

SELECT C.Id_cita, H.numero
FROM Cita C
CFT JOIN Cita_Habitacion CH ON C.Id_cita = CH.Id_cita
CFT JOIN Habitacion H ON CH.Id_habitacion = H.Id_habitacion

accion;

- RIGHT JOIN: todas las habitaciones y citas que han sido usadas

SELECT H.numero, C.Fecha_cita
FROM Habitacion H
RIGHT JOIN Cita_Habitacion CH ON H.Id_habitacion =
CH.Id_habitacion
RIGHT JOIN Cita_C ON CH.Id_cita = C.Id_cita;

- FULL OUTER JOIN: Cumulo con UNION

SELECT P.nombre, M.nombre AS medicamento
FROM Paciente P
LEFT JOIN paciente_Medicamento PM ON P.Id_paciente =
PM.Id_paciente
LEFT JOIN Medicamento M ON PM.Id_medicamento =
M.Id_medicamento
UNION
SELECT P.nombre, M.nombre
FROM Paciente P
RIGHT JOIN paciente_Medicamento PM ON P.Id_paciente =
PM.Id_paciente
RIGHT JOIN Medicamento M ON PM.Id_medicamento =
M.Id_medicamento

- Subconsultas, Medicamentos del paciente mas joven

SELECT M.nombre
FROM Medicamento M
JOIN paciente_Medicamento PM ON M.Id_medicamento =
PM.Id_medicamento
WHERE PM.Id_paciente = C;

SELECT Id_paciente FROM Paciente

);

- Paciente con mas medicamentos recibidos

SELECT P.nombre FROM Paciente P
WHERE Id_paciente = C

SELECT Id_paciente
FROM paciente_Medicamento
ORDER BY Id_paciente DESC
LIMIT 1

);

```
DELETE FROM Medico  
WHERE idMedico = 1;
```

Sistema de Gestión de Ventas en una Tienda de Tecnología

```
CREATE DATABASE ejercicioFour;  
USE ejercicioFour;  
CREATE TABLE Categoria_producto (  
    id_categoria INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion VARCHAR(100)
```

```
);
```

```
CREATE TABLE proveedor (  
    id_proveedor INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    precio DECIMAL(10,2) NOT NULL,  
    id_categoria INT,  
    id_proveedor INT FOREIGN KEY (id_proveedor) REFERENCES proveedor  
    C_id_proveedor,  
    FOREIGN KEY (id_categoria) REFERENCES categoria  
    C_id_categoria)
```

```
);  
CREATE TABLE Cliente ( "C_id_cliente" INT AUTO_INCREMENT PRIMARY KEY,  
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    correo VARCHAR(100) UNIQUE,  
    );  
CREATE TABLE Empleado ( "C_id_empleado" INT AUTO_INCREMENT PRIMARY KEY,  
    id_empleado INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100),  
    fecha DATE,  
    FOREIGN KEY (C_id_cliente) REFERENCES Cliente (id_cliente),  
    FOREIGN KEY (C_id_empleado) REFERENCES empleado (id_empleado)
```

```
);  
CREATE TABLE Factura ( "C_id_factura" INT AUTO_INCREMENT PRIMARY KEY,  
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,  
    id_empleado INT,  
    fecha DATE,  
    FOREIGN KEY (C_id_cliente) REFERENCES Cliente (id_cliente),  
    FOREIGN KEY (C_id_empleado) REFERENCES empleado (id_empleado)
```

```
);  
CREATE TABLE Factura ( "C_id_factura" INT AUTO_INCREMENT PRIMARY KEY,  
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,  
    id_empleado INT,  
    fecha DATE,  
    FOREIGN KEY (C_id_cliente) REFERENCES Cliente (id_cliente),  
    FOREIGN KEY (C_id_empleado) REFERENCES empleado (id_empleado)
```

Nombre _____
Profesor _____
Institución _____

Materia _____
Curso _____
Nota _____

CREATE TABLE Detalle_Factura (

Id_factura INT,
Id_producto INT,
Cantidad INT
PRIMARY KEY (Id_factura, Id_producto),
FOREIGN KEY (Id_factura) REFERENCES Factura(Id_factura),
FOREIGN KEY (Id_producto) REFERENCES Producto(Id_producto);

)
CREATE TABLE Factura_Sucursal (

Id_factura INT,
Sucursal VARCHAR(50),
Observacion TEXT,
PRIMARY KEY (Id_factura),
FOREIGN KEY (Id_factura) REFERENCES Factura(Id_factura);

)
-- Inserciones

INSERT INTO Proveedor(Nombre) VALUES ('Lenovo'), ('Samsung'),
INSERT INTO Producto(Nombre, precio, Id_Categoría,
(Id_proveedor)) VALUES ('IdeaPad 3', 2000000, 1, 1),
('Galaxy A34', 1200000, 2, 2);
INSERT INTO Cliente(Nombre, correo) VALUES ('Luis Conde', 'luis@gmail.com');
INSERT INTO Empleado(Nombre) VALUES ('Claudia Ruiz');
INSERT INTO Factura(Id_Cliente, Id_empleado, fecha)
VALUES (1, 1, '2025-06-12');
INSERT INTO Detalle_Factura VALUES (1, 2, 2), (1, 2, 1);
INSERT INTO Factura_Sucursal VALUES (1, 'Neiva Centro', 'Venta Presencial');

-- UPDATE: Modificar precio de un Producto

UPDATE Producto
SET precio = 2100000
WHERE Id_Producto = 1;

-- Delete: Eliminar detalle de factura de producto X3 Y

DELETE FROM Detalle_Factura

WHERE Id_factura = 1 AND Id_producto = 2;

-- INNER JOIN: facturas con nombre del cliente y empleado

SELECT F.Id_factura, C.nombre AS cliente, E.nombre AS empleado, F.Fecha
FROM Factura F
INNER JOIN Cliente C ON F.Id_cliente = C.Id_cliente
INNER JOIN Empleado E ON F.Id_empleado = E.Id_empleado;

- LEFT JOIN : productos y contactos vendidos si existen

```
SELECT P.nombre , DF.cantidad  
FROM Producto P  
LEFT JOIN Detalle_Factura DF ON P.id_Producto =  
DF.id_producto;
```

-- RIGHT JOIN : mostrar proveedores incluso si no tienen productos registrados

```
SELECT PR.nombre AS proveedor , P.nombre  
AS Producto  
FROM Producto P  
RIGHT JOIN Proveedor PR ON P.id_Proveedor =  
PR.id_Proveedor;
```

-- Subconsulta producto mas caro vendido

```
SELECT nombre FROM Producto  
WHERE id_producto = (SELECT id_producto  
FROM factura  
WHERE id_factura = 1  
ORDER BY cantidad DESC  
LIMIT 1);
```

-- Subconsulta nombre del cliente que hizo la ultima compra

```
SELECT nombre FROM Cliente  
WHERE id_cliente = (SELECT id_cliente  
FROM factura  
ORDER BY fecha DESC  
LIMIT 1);
```

C:

Sistema de Gestión de Reservas en un Hotel

```
CREATE DATABASE EjercicioFive;
```

```
USE EjercicioFive;
```

```
CREATE TABLE Fijo_Habitacion (
```

```
id_tipo INT AUTO_INCREMENT PRIMARY KEY,
```

```
descCripeen VARCHAR(100),
```

```
);
```

```
CREATE TABLE Servicio (
```

```
id_Servicio INT AUTO_INCREMENT PRIMARY KEY,
```

```
nombre VARCHAR(100) NOT NULL,
```

```
costo DECIMAL(10,2),
```

```
);
```

```
CREATE TABLE Huesped (
```

```
id_Huesped INT AUTO_INCREMENT PRIMARY KEY,
```

```
nombre VARCHAR(100) NOT NULL,
```

```
documento VARCHAR(20) UNIQUE,
```

```
);
```

CREATE TABLE Empleado (

id_empleado INT AUTO_INCREMENT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL

CREATE TABLE Habitacion (

id_habitacion INT AUTO_INCREMENT PRIMARY KEY,

numero VARCHAR(10) NOT NULL

id_tipo INT

FOREIGN KEY (id_tipo) REFERENCES Tipo_Habitacion

(id_tipo)

);

CREATE TABLE Reserva (

id_reserva INT AUTO_INCREMENT PRIMARY KEY,

id_huesped INT,

id_habitacion INT

id_empleado INT

fecha_ingreso DATE NOT NULL

fecha_salida DATE NOT NULL

FOREIGN KEY (id_huesped) REFERENCES Huesped (id_huesped)

FOREIGN KEY (id_habitacion) REFERENCES habitacion

(id_huesped)

FOREIGN KEY (id_empleado) REFERENCES Empleado (id_empleado)

);

CREATE TABLE Reserva_Servicio (

id_reserva INT,

id_servicio INT,

PRIMARY KEY (id_reserva, id_servicio),

FOREIGN KEY (id_reserva) REFERENCES Reserva (id_reserva),

FOREIGN KEY (id_servicio) REFERENCES Servicio (id_servicio),

);

CREATE TABLE Reserva_Sede (

id_reserva INT,

sede VARCHAR(100),

Comentario TEXT,

PRIMARY KEY (id_reserva, sede),

FOREIGN (id_reserva) REFERENCES Reserva (id_reserva),

);

Insertar datos básicos

INSERT INTO Tipo_Habitacion (descripcion) VALUES ('Sencilla'),
(Doble),
INSERT INTO Habitacion (numero, id_tipo) VALUES (101, 1),
(202, 2),
INSERT INTO Servicio (nombre, costo) VALUES ('Desayuno', 25000),
('Spa', 80000),
INSERT INTO Huesped (nombre, documento) VALUES ('Bris Conde', '700200300'),
('Ana Flores', '123456789'),
INSERT INTO Empleado (nombre) VALUES ('Carlos Pérez'),
('Laura Gómez'),
INSERT INTO Reserva (id_huesped, id_habitacion, id_empleado, fecha_ingreso, fecha_salida) VALUES (1, 1, 1, '2025-07-10'),
(2, 2, 2, '2025-07-10'),
INSERT INTO Reserva_Servicio VALUES (1, 1), (1, 2), (2, 1),

INSERT INTO Reserva_Sede VALUES (2, 'Nueva', 'Reserva Presencial'), (2, 'Bogota', 'Hecha por web');

-- UPDATE: Cambiar costo del servicio '8 pa

```
UPDATE Servicio  
SET costo = 85000  
WHERE id_servicio=2;
```

-- DELETE Eliminar la relación entre la reserva 2 y el servicio 'Desa'

```
DELETE FROM Reserva_Servicio  
WHERE id_reserva=2 AND id_servicio=1;
```

-- INNER JOIN Mostrar reservas con nombre del huésped y habitación

```
SELECT R.id_reserva, H.nombre AS huésped, HA.numero  
AS habitación  
FROM Reserva R  
INNER JOIN Huésped H ON R.id_huesped = H.id_Huesped  
INNER JOIN Habitacion HA ON R.id_habitación = HA.id_habitacion;
```

-- LEFT JOIN: Mostrar todas las habitaciones y su tipo

```
SELECT HA.numero, TH.descripcion  
FROM Habitacion HA  
LEFT JOIN Tipo_Habitacion TH ON HA.id_tipo = TH.id_tipo;
```

-- RIGHT JOIN Mostrar servicios, aunque no sean usados

```
SELECT S.nombre, RS.id_reserva  
FROM Reserva_Servicio RS  
RIGHT JOIN Servicio S ON RS.id_servicio = S.id_servicio;
```

-- SUBCONSULTA 1: Mostrar el nombre del huésped de la última reserva

```
SELECT nombre FROM Huésped  
WHERE id_huesped =  
(SELECT id_huesped FROM Reserva  
ORDER BY fecha_ingreso DESC  
LIMIT 1);
```

-- SUBCONSULTA 2: Servicios de la reserva con más días de duración

```
SELECT S.nombre FROM Servicio S  
JOIN Reserva_Servicio RS ON S.id_servicio = RS.id_servicio  
WHERE RS.id_reserva =  
(SELECT id_reserva FROM Reserva  
ORDER BY DATEDIFF(fecha_salida, fecha_ingreso) DESC  
LIMIT 1);
```

Nombre

Profesor

Institución

Materia

Curso

Nota

Sistema de Gestión de Biblioteca Universitaria

```
CREATE DATABASE ejercicioTwo;
```

```
USE ejercicioTwo;
```

```
CREATE TABLE Categoria (
```

```
    id_categoria INT AUTO_INCREMENT PRIMARY KEY,  
    descripción VARCHAR (100)
```

```
);
```

```
CREATE TABLE Editorial (
```

```
    id_editorial INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR (50) NOT NULL
```

```
);
```

```
CREATE TABLE Libro (
```

```
    id_libro INT AUTO_INCREMENT PRIMARY KEY,  
    título VARCHAR (100) NOT NULL,  
    id_categoria INT,  
    id_editorial INT,  
    FOREIGN KEY (id_categoria) REFERENCES Categoria (id_categoria),  
    FOREIGN KEY (id_editorial) REFERENCES Editorial (id_editorial)
```

```
);
```

```
CREATE TABLE Usuario (
```

```
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR (50) NOT NULL,  
    email VARCHAR (100) UNIQUE
```

```
);
```

```
CREATE TABLE Prestamo (
```

```
    id_prestamo INT AUTO_INCREMENT PRIMARY KEY,  
    id_libro INT,  
    id_usuario INT,  
    fecha_prestamo DATE NOT NULL,  
    Fecha_devolucion DATE,  
    FOREIGN KEY (id_libro) REFERENCES Libro (id_libro),  
    FOREIGN KEY (id_usuario) REFERENCES Usuario (id_usuario)
```

```
);
```

```
CREATE TABLE Autor (
```

```
    id_autor INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR (50) NOT NULL
```

```
);
```

```
CREATE TABLE Libro_Autor (
    id_libro INT,
    id_autor INT,
    PRIMARY KEY (id_libro, id_autor),
    FOREIGN KEY (id_libro) REFERENCES Libro (id_libro),
    FOREIGN KEY (id_autor) REFERENCES Autor (id_autor)
);
```

```
;
```

```
CREATE TABLE Libro_Sede (
    id_libro INT,
    sede VARCHAR(50),
    cantidad INT,
    PRIMARY KEY (id_libro, sede),
    FOREIGN KEY (id_libro) REFERENCES Libro (id_libro)
);
```

```
;
```

```
INSERT INTO Categoría (descripción) VALUES ('Literatura'), ('Ingeniería'),
('Historia');
INSERT INTO Editorial (nombre) VALUES ('Alfaomega'), ('McGraw-Hill'),
('Isaac Asimov');
INSERT INTO Autor (nombres) VALUES ('Gabriel García Marquez');
```

```
INSERT INTO Libro (título, id_categoria, id_editorial)
VALUES ('Cien años de soledad', 1, 1), ('Fundación', 2, 2);
```

```
INSERT INTO Usuario (nombre, email, )
VALUES ('Luis Conde', 'luse@gmail.com'), ('Ana Torres', 'ana@gmail.com');
```

```
INSERT INTO Prestamo (id_libro, id_usuario, fecha_prestamo, fecha_devolucion)
VALUES (1, 1, '2025-06-10', '2025-06-20'), (2, 2, '2025-06-11', NULL);
```

```
INSERT INTO Libro_Autor VALUES (1, 1), (2, 2);
```

```
INSERT INTO Libro_Sede (values (1, 'Neiva', 5), (2, 'Bogotá', 3));
```

```
— UPDATE: Cambiar cantidad de copias de un libro en Neiva
```

```
UPDATE Libro_Sede
SET cantidad = ?
```

```
WHERE id_libro = 1 AND sede = 'Neiva';
```

```
— Delete: Eliminar prestamo con id 2
DELETE FROM Prestamo
WHERE id_prestamo = 2;
```

```
— INNER JOIN: Libros prestados con nombre de usuario
```

```
Select L.título, U.nombre, P.fecha_prestamo
FROM Prestamo P
INNER JOIN Libro L ON P.id_libro = L.id_libro
INNER JOIN Usuario U ON P.id_usuario = U.id_usuario;
```

```
— LEFT JOIN: Todos los libros y su cantidad en Neiva (si existe)
```

```
SELECT L.título, LS.cantidad
FROM Libro L
LEFT JOIN Libro_Sede LS ON L.id_libro = LS.id_libro AND LS.sede = 'Neiva';
```

6HT JOIN: Mostrar autores aunque no tengan libros registrados con datos actuales no hay nulls, pero sirve de ejemplo)

SELECT A.nombre AS autor, L.titulo

FROM Libro ALIAS LA

INNER JOIN Autor A ON LA.id-autor = A.id-autor

6HT JOIN libro L ON LA.id-libro = L.id-libro;

SUBCONSULTA 1: Libros prestados por el usuario más reciente

SELECT titulo FROM libro

WHERE id-libro IN C

SELECT id-libro FROM Presbamo

WHERE id-usuario = (SELECT MAX(id-usuario) FROM Usuario)

;

- SUBCONSULTA 2: Mostrar el libro con más copias en Melua

SELECT titulo FROM libro

WHERE id-libro = C

SELECT id-libro FROM libro-sede

WHERE sede = 'Melua'

ORDER BY cantidad DESC

LIMIT 1

;

Sistema de Gestión de Eventos Culturales

CREATE DATABASE ejercicioThree;

USE ejercicioThree;

CREATE TABLE Lugar (

id-lugar INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR (100) NOT NULL,
ciudad VARCHAR (50)

);

CREATE TABLE categoria-Evento (

id_categoria INT AUTO_INCREMENT PRIMARY KEY,
descripcion VARCHAR (100)

);

CREATE TABLE Organizador (

id_organizador INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR (100) NOT NULL

);

CREATE TABLE Evento (

id-evento INT AUTO_INCREMENT PRIMARY KEY,

nombre VARCHAR (100) NOT NULL,

Fecha DATE NOT NULL,

id-lugar INT,

id-categoría INT,

id-organizador INT,

FOREIGN KEY (id-lugar) REFERENCES Lugar (id-lugar),

FOREIGN KEY (id-categoría) REFERENCES categoria-Evento (id-categoría),

FOREIGN KEY (id-organizador) REFERENCES organizador (id-organizador)

');

```
CREATE TABLE Artista (
    id_artista INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    especialidad VARCHAR(50)
```

');

```
CREATE TABLE Evento_Artista (
```

```
    id_evento INT,
    id_artista INT,
    PRIMARY KEY (id_evento, id_artista),
    FOREIGN KEY (id_evento) REFERENCES Evento (id_evento),
    FOREIGN KEY (id_artista) REFERENCES Artista (id_artista)
```

');

```
CREATE TABLE Evento_Sponsor (
```

```
    id_evento INT,
    patrocinador VARCHAR(100),
    aporte DECIMAL(10,2),
    PRIMARY KEY (id_evento, patrocinador),
    FOREIGN KEY (id_evento) REFERENCES Evento (id_evento)
```

');

- Insertar datos básicos

```
INSERT INTO Lugar (Nombre, ciudad) VALUES ('Teatro Nacional', 'Bogotá'),
('Auditorio U', 'Neiva');
```

```
INSERT INTO categoria_Evento (descripcion) VALUES ('Música'), ('Danza'),
('Teatro');
```

```
INSERT INTO Organizador (Nombre) VALUES ('Luis Conde'), ('Ana Ropas');
```

```
INSERT INTO Artista (nombre, especialidad) VALUES ('Carlos Vives', 'Musico'),
('Ballet Neiva', 'Danza');
```

```
INSERT INTO Evento (Nombre, Fecha, id_lugar, id_categoria, id_organizador)
VALUES ('Festival Andino', '2025-07-20', 1, 1, 1), ('Noche de Danza',
'2025-08-15', 2, 2, 2);
```

```
INSERT INTO Evento_Artista VALUES (1, 1), (2, 2);
```

```
INSERT INTO Evento_Sponsor VALUES (1, 'Alcaldía de Bogotá', 5000000.00),
(2, 'Gobernación del Huila', 3000000.00);
```

- UPDATE : Cambiar ciudad del lugar

```
UPDATE Lugar SET ciudad = 'Medellín' WHERE id_lugar = 1;
```

- DELETE : Eliminar artista de evento

```
DELETE FROM Evento_Artista WHERE id_evento = 2 AND id_artista = 2;
```

- INNER JOIN : Eventos con lugar, Organizador y categoría

```
SELECT E.nombre AS evento, L.nombre AS lugar, C.descripcion AS
categoria, O.nombre AS organizador
FROM Evento E
```

```
INNER JOIN Lugar L ON E.id_lugar = L.id_lugar,
```

```
INNER JOIN Categoría_Evento C ON E.id_categoria = C.id_categoria
```

```
INNER JOIN Organizador O ON E.id_organizador = O.id_organizador;
```

— LEFT JOIN: Todos los eventos y sus patrocinadores, (Si tienen)

SELECT E.nombre AS evento, S.patrocinador, S.aporte
 FROM Evento E
 LEFT JOIN Evento-Sponsor S ON E.id-evento;

— RIGHT JOIN: Mostrar todos los artistas aunque no estén en eventos

SELECT A.nombre AS artista, E.nombre AS evento
 FROM Evento-Artista EA
 RIGHT JOIN Artista A ON EA.id-artista = A.id-artista
 RIGHT JOIN Evento E ON EA.id-evento = E.id-evento;

— SUBCONSULTA 1: Evento con mayor aporte total de patrocinadores

SELECT nombre FROM Evento
 WHERE id-evento = C
 SELECT id-evento FROM Evento-Sponsor
 GROUP BY id-evento
 ORDER BY sum(aporte) DESC
 LIMIT 1

);

— SUBCONSULTA 2: Artistas del último evento registrado

SELECT A.nombre
 FROM Artista A
 JOIN Evento-Artista EA ON A.id-artista = EA.id-artista
 WHERE EA.id-evento = (SELECT MAX(id-evento) FROM Evento);

DICCIONARIO DE DATOS ejercicioOne: Sistema de Gestión Hospitalaria

Tabla	Atributo	Tipo MySQL	Descripción	Restricciones
Paciente	id-paciente	INT	ID del paciente	PK, AUTO_INCREMENT
	nombre	VARCHAR(50)	Nombre completo	NOT NULL
	edad	INT	edad	
	genero	VARCHAR(10)	Sexo del paciente	
	direccion	VARCHAR(100)	Dirección	
Medico	id-medico	INT	ID del medico	PK, AUTO_INCREMENT
	nombre	VARCHAR(50)	Nombre completo	NOT NULL
	especialidad	VARCHAR(30)	Especialidad médica	
	telefono	VARCHAR(15)	telefono	
Cita	id-cita	INT	ID de la cita	PK, AUTO_INCREMENT
	id-paciente	INT	FK a Paciente	FK
	id-medico	INT	FK a Medico	FX
	fecha-cita	DATE	Fecha	NOT NULL
	motivo	TEXT	Motivo de la cita	
Historial Clinico	id-historial	INT	ID historial	PK, AUTO_INCREMENT
	id-paciente	INT	FK único a Paciente	UNIQUE, FK
	diagnostico	TEXT	Diagnóstico médico	
	tratamiento	TEXT	Tratamiento	
Habitacion	id-habitacion	INT	ID de habitación	PK, AUTO_INCREMENT
	numero	INT	Número	
	piso	INT	Piso	
	tipo	VARCHAR(20)	UCI, General, etc	

Medicamento	id_medicamento	INT		id del medicamento	PK, Auto.
	nombre	VARCHAR(50)		nombre comercial	
	dosis	VARCHAR(20)		Dosis	
	descripcion	TEXT	-	Detalles	
Paciente-Medico	id_paciente	INT		FK a Paciente	PK, FK
mento	id_medicamento	INT		FK a medicamento	PK, FK
	Fecha-receta	DATE	-	Fecha de Prescripción	
Cita-Habitación	id_cita	INT		FK a cita	PK, FK
	id_habitacion	INT		FK a habitación	PK, FK
	observación	TEXT	-	Notas sobre el uso de habitación	

DICCIONARIO DE DATOS ejercicio Tres: Sistema de Gestión de Biblioteca Universitaria

Tabla	Atributo	Tipo MySQL	Descripción	Restricciones
Libro	id_libro	INT	Identificador libro	PK, Auto-increment
	título	VARCHAR(100)	Título del libro	NO NULL
	id_categoria	INT	FK a Categoría	FX
	id_editorial	INT	FK a Editorial	FX
Usuario	id_usuario	INT	Id del usuario	PK, Auto-increment
	nombre	VARCHAR(50)	Nombre de usuario	NO NULL
	email	VARCHAR(100)	correo electrónico	UNIQUE
Prestamo	id_prestamo	INT	Id del préstamo	PK, Auto-increment
	id_libro	INT	FK a libro	FK
	id_usuario	INT	FK a Usuario	FX
	fecha_prestamo	DATE	Fecha del préstamo	NOT NULL
	fecha_devolucion	DATE	Fecha de devolución	
Autor	id_autor	INT	Id del autor	PK, Auto-increment
	nombre	VARCHAR(50)	Nombre de autor	NO NULL
Categoría	id_categoria	INT	Id de la categoría	PK, Auto-increment
	descripción	VARCHAR(100)	Nombre de categoría	
Editorial	id_editorial	INT	Id de la editorial	PK, Auto-increment
	nombre	VARCHAR(50)	Nombre de editorial	NO NULL
Libro-Autor	id_libro	INT	FK a libro	PK, PK
	id_autor	INT	FK a autor	PK, FK
Libro-Sede	id_libro	INT	FK a libro	PK, PK
	sede	VARCHAR(50)	Nombre de la sede	
	cantidad	INT	Número de copias en esa sede	

Diccionario de Datos 8 GESTIÓN DE EVENTOS CULTURALES

Tabla	Atributo	Tipo MySQL	Descripción	Restricciones
Evento	id_evento	INT	ID del evento	PK, AUTO_INCREMENT
	nombre	VARCHAR(100)	Nombre del evento	NOT NULL
	fecha	DATE	Fecha del evento	NOT NULL
	id_lugar	INT	FK a lugar	FK
	id_categoria	INT	FK a categoría	FK
	id_organizador	INT	FK a organizador	FK
Artista	id_artista	INT	ID del Artista	PK, AUTO_INCREMENT
	nombre	VARCHAR(100)	Nombre Artista	NOT NULL
	especialidad	VARCHAR(100)	Tipo de Arte	
Lugar	id_lugar	INT	ID del lugar	PK, AUTO_INCREMENT
	nombre	VARCHAR(100)	Nombre del lugar	NOT NULL
	ciudad	VARCHAR(50)	Ciudad donde es	
Categoría	id_categoria	INT	ID de categoría	PK, AUTO_INCREMENT
Evento	descripción	VARCHAR(100)	Nombre categoría	
Organizador	id_organizador	INT	ID del organizador	PK, AUTO_INCREMENT
	nombre	VARCHAR(100)	Nombre de organ	

	Atributo	Tipo Dato	Descripción	Restricciones
1.	Id_Evento	INT	FK a evento	PK, FK
ta	Id_artista	INT	FK a Artista	PK, FK
o-	Id_Evento	NT	FK a evento	PK, FK
vor	Patrocinador	VARCHAR(100)	nombre Patrocinador	PK
	aporte	DECIMAL(10,2)	monto aportado	

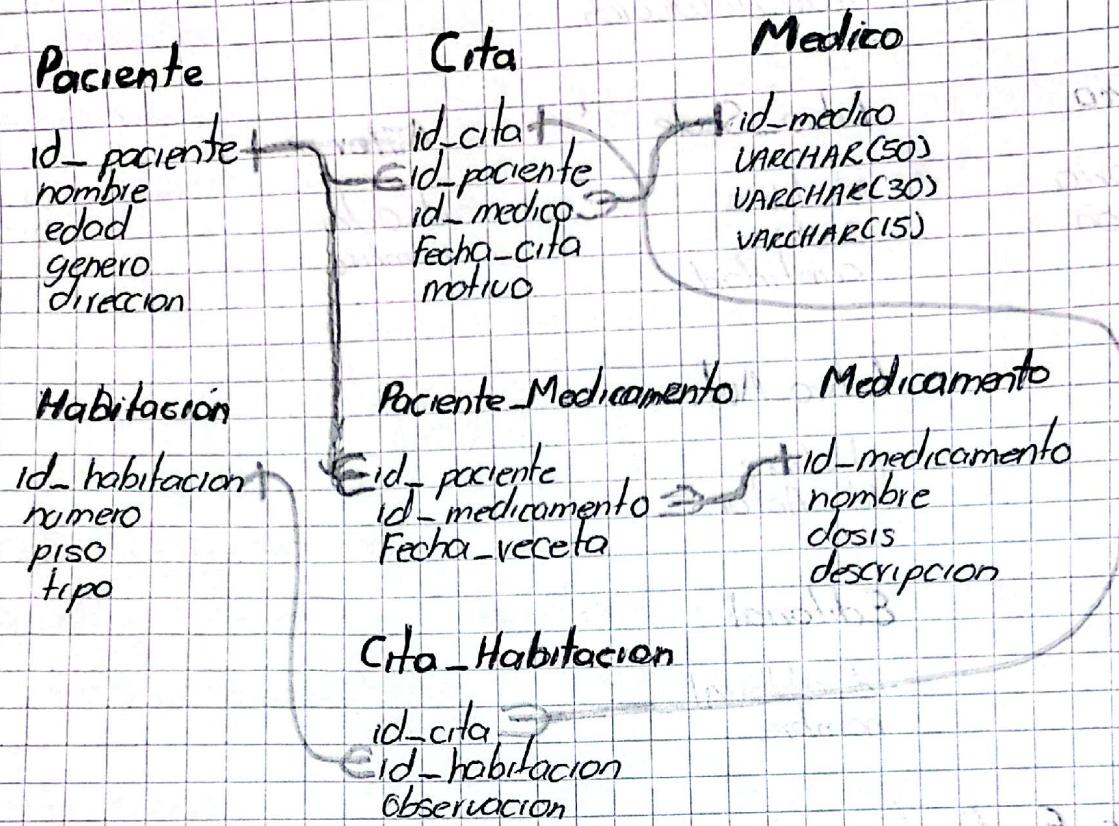
Diccionario De Datos : Gestión de ventas en una tienda de
Tecnología

	Atributo	Tipo Dato	Descripción	Restricciones
dpto	Id_producto	INT	ID Producto	PK AUTO_INCREMENT
	nombre	VARCHAR(100)	nombre Producto	NOT NULL
	Precio	DECIMAL(10,2)	Precio unitario	NOT NULL
	Id_categoria	INT	FK a Categoria P	FK
iente	Id_Proveedor	INT	FK a Proveedor	FK
	Id_Cliente	INT	ID Cliente	PK, AUTO_INCREMENT
	nombre	VARCHAR(100)	nombre Cliente	NOT NULL
	Correo	VARCHAR(100)	Email Cliente	UNIQUE
mpelado	Id_empleado	INT	ID empleado	PK AUTO_INCREMENT
	nombre	VARCHAR(100)	Nombre empleado	NOT NULL
tegoria	Id_categoria	INT	ID Categoria	PK AUTO_INCREMENT
rolsito	descripcion	VARCHAR(100)	nombre Categoria	
roveedor	Id_Proveedor	INT	ID Proveedor	PK AUTO_INCREMENT
	Nombre	VARCHAR(100)	Nombre Proveedor	NOT NULL
actura	Id_factura	INT	FK a factura	PK, AUTO_INCREMENT
	Id_Cliente	INT	FK a cliente	FK
	Id_empleado	INT	FK a empleado	FK
	Fecha	DATE	Fecha adquisición	NOT NULL
Detalle_factura	Id_factura	INT	FK a factura	PK, FK
	Id_producto	INT	FK a Producto	FK
	Cantidad	INT	Cantidad Comprada	
Factura_Sucursal	Id_factura	INT	FK a factura	PK, FK
	Sucursal	VARCHAR(50)	Nombre sucursal	PK
	Observacion	TEXT	Observación sobre la transacción	

Diccionario de Datos : RESERVAS EN UN HOTEL

	Atributo	Tipo Dato	Descripción	Restricciones
Huesped	Id_huesped	INT	ID del Huesped	PK AUTO_INCREMENT
	Nombre	VARCHAR(100)	Nombre del Huesped	NOT NULL
	Documento	VARCHAR(20)	Documento de ID	UNIQUE
Empleado	Id_empleado	INT	ID empleado	PK, AUTO_INCREMENT
	Nombre	VARCHAR(100)	Nombre completo	NOT NULL
Tipo_Habitacion	Id_tipo	INT	ID del tipo Habitación	PK AUTO_INCREMENT
	descripcion	VARCHAR(100)	Descripción del tipo	
Habitacion	Id_habitacion	INT	Id habitación	PK AUTO_INCREMENT
	Número	VARCHAR(10)	Número habitación	NOT NULL
	Id_tipo	INT	FK a Tipo Habitación	FK
Servicio	Id_servicio	INT	ID del servicio	PK AUTO_INCREMENT
	Nombre	VARCHAR(100)	Nombre del servicio	NOT NULL
	Costo	DECIMAL(10,2)	Precio del Servicio	
Reserva	Id_reserva	INT	ID de la reserva	PK AUTO_INCREMENT
	Id_huesped	INT	FK a Huesped	FK
	Id_habitacion	INT	FK a Habitación	FK
	Id_empleado	INT	FK a Empleado	FK
	Fecha_ingreso	DATE	fecha de ingreso	NOT NULL
	Fecha_salida	DATE	fecha de salida	NOT NULL
Reserva-Servicio	Id_reserva	INT	FK a reserva	PK, FK
	Id_servicio	INT	FK a Servicio	PK, FK
Reserva-Sede	Id_reserva	INT	FK a reserva	PK, FK
	Sede	VARCHAR(100)	Nombre de la sede	PK
	Comentario	TEXT	Observaciones sobre la sede	

MER Gestión Hospitalaria



MER Gestión Hospitalaria Explicación

Relación

Paciente → Historial_Clinico	1:N	Un paciente tiene un historial clínico
Paciente → Cita	1:N	Un paciente puede tener varios citas
Medico → Cita	N:M	Un medico atiende muchas citas
Paciente ↔ Medicamento	N:M	Vía Paciente-Medicamento
Cita ↔ Habitacion	N:M	Vía Cita-Habitacion

MER Gestión de Biblioteca Universitaria

Libro

Prestamo

Usuario

Id_libro
 Titulo
 Id_categoria
 Id_editorial
 Foto_deportacion

Id_Prestamo
 Id_libro
 Id_usuario
 Nombre
 Email

Categoría

Id_categoria
 Descripción

Id_libro
 Sede
 Cantidad
 Nombre

Libro-Autor

Id_libro
 Id_Actor

Editorial

Id_editorial
 Nombre

Relaciones Expliados

Relación

$\text{Libro} \leftrightarrow \text{Autor} (\text{Libro-Autor})$

$N:M$

Un libro puede ser escrito por varios autores, y un autor puede escribir varios libros.

$\text{Usuario} \rightarrow \text{Prestamo}$

$1:N$

Un usuario puede hacer muchos préstamos a lo largo del tiempo. Un libro puede ser prestado muchas veces, cada uno es un registro diferente.

$\text{Libro} \rightarrow \text{Prestamo}$

$N:1$

Cada libro pertenece a una sola categoría.

$\text{Libro} \rightarrow \text{Editorial}$

$1:N$

Cada libro es publicado por una sola editorial.

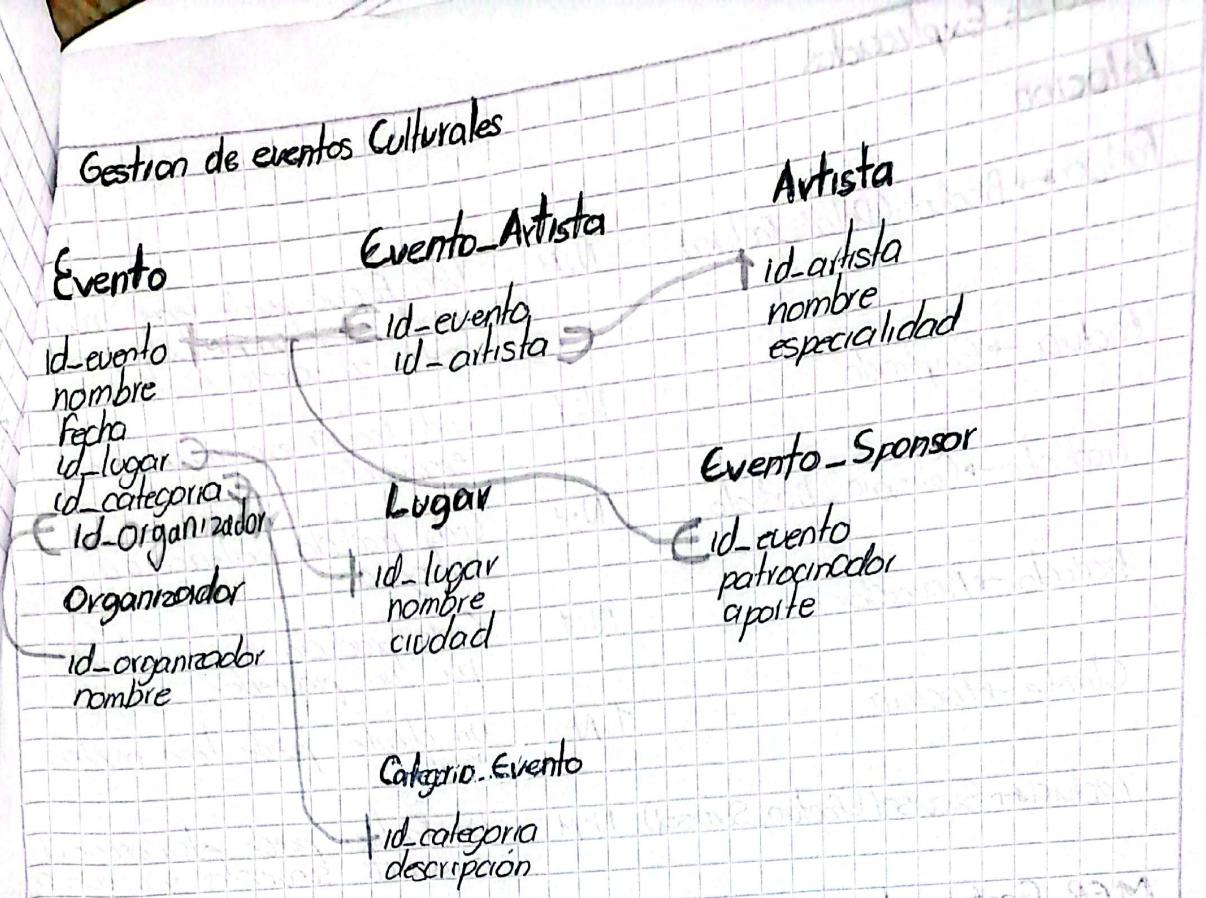
$\text{Libro} \rightarrow \text{Sede} (\text{Libro-Sede})$

$N:H$

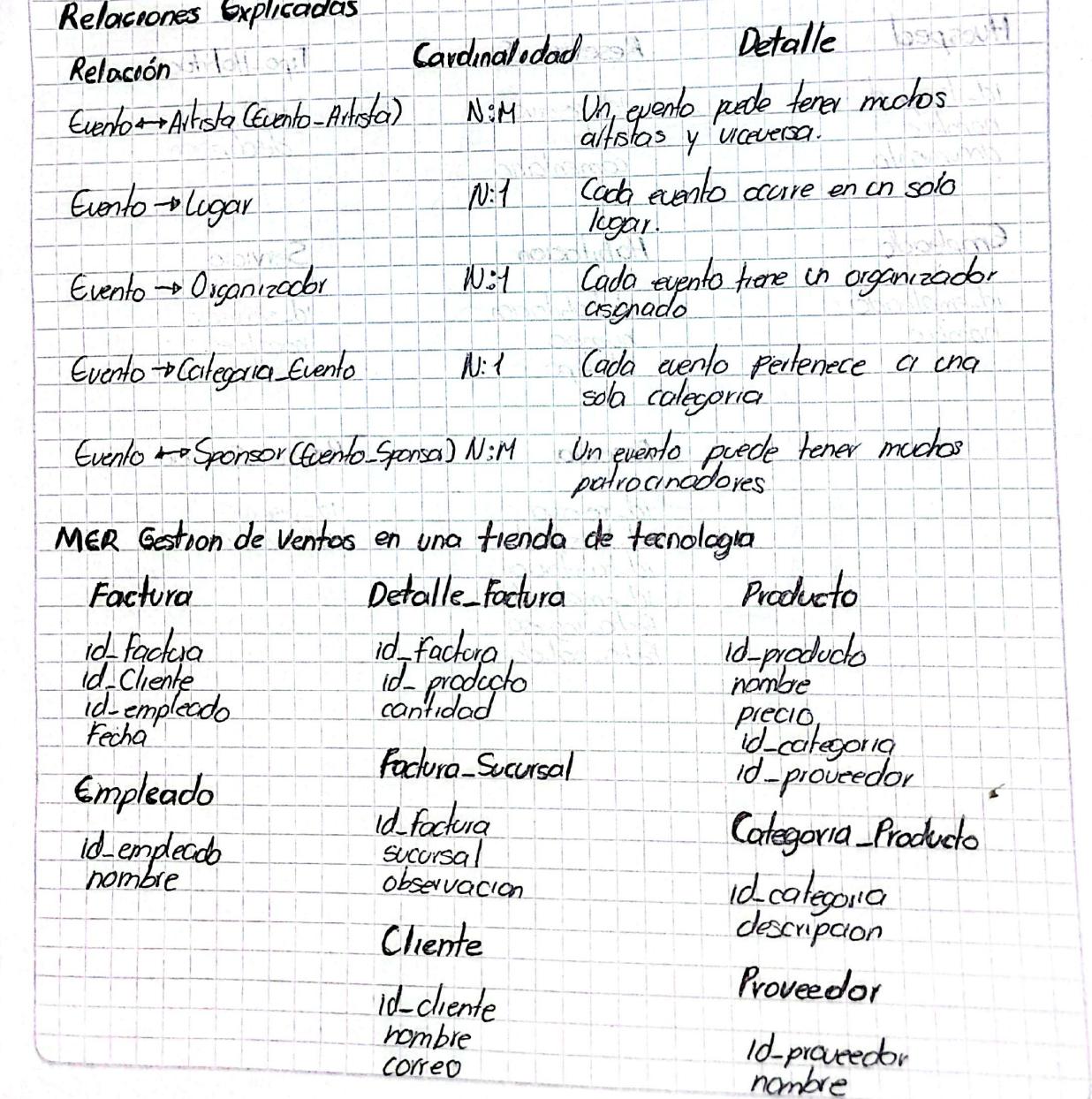
Un libro puede estar disponible en muchas sedes y una sede tiene muchos libros.

Aunque Sede no es una entidad, se puede usar un JOIN entre libro y libro-Sede para saber en que sedes está disponible cada libro, siniendo así una relación N:N primaria.

Gestión de eventos Culturales



MER Gestión de Ventas en una tienda de tecnología



Relaciones Explorados

Relación

Factura → Producto (Detalle_Factura) N:M

Una Factura puede tener muchos productos y un producto puede estar en varias facturas.

Factura → Empleado N:1

Cada factura es atendida por un empleado

Producto → Categoría_Producto N:1

Cada producto pertenece a una categoría.

Producto → Proveedor N:1

Cada producto es suministrado por un proveedor

Cliente → Factura 1:N

Un cliente puede tener muchas facturas.

Factura → Sucursal (Factura_Sucursal) N:M

Una factura puede estar asociada a varias sucursales y viceversa.

MER Gestión de reservas en un Hotel

Huesped

id_huesped
nombre
documento

Reserva_Sede

id_reserva
sede
comentario

Tipo-Habitacion

id_tipo
descripcion

Empleado

id_empleado
nombre

Habitacion

id_habitacion
numero
id_tipo

Servicio

id_servicio
nombre
costo

Reserva

id_reserva
id_huesped
id_habitacion
id_empleado
Fecha_ingreso
Fecha_salida

Reserva_Servicio

id_reserva
id_servicio

Nombre

Fecha

dia

mes

año

Profesor

Materia

Institución

Curso

Nota

Relaciones Explicadas

Relación

Cardinalidad

Descripción

Huesped → Reserva

1:N

Un huésped puede tener muchos reservas.

Reserva → Servicio (Reserva, Servicio) N:M

Una reserva puede incluir muchos servicios, y un servicio puede repetirse.

Reserva → Habitacion

N:1

Cada reserva se asigna a una sola habitación.

Habitacion → Tipo-habitacion

N:1

Cada habitación tiene un tipo asignado.

Reserva → Empleado

N:1

Cada reserva es gestionada por un empleado.

Reserva → Sede (Reserva, Sede) N:M*

Si, Sede no es una entidad formal, la relación es 1:M primitiva.

La relación es técnicamente 1:N desde Reserva hacia Reserva-Sede ya que no existe una entidad formal Sede. Sin embargo, mediante la estructura del código (usando id_reserva + sede como clave compuesta), se intenta simular una relación de muchos a muchos (N:M) de forma primitiva.