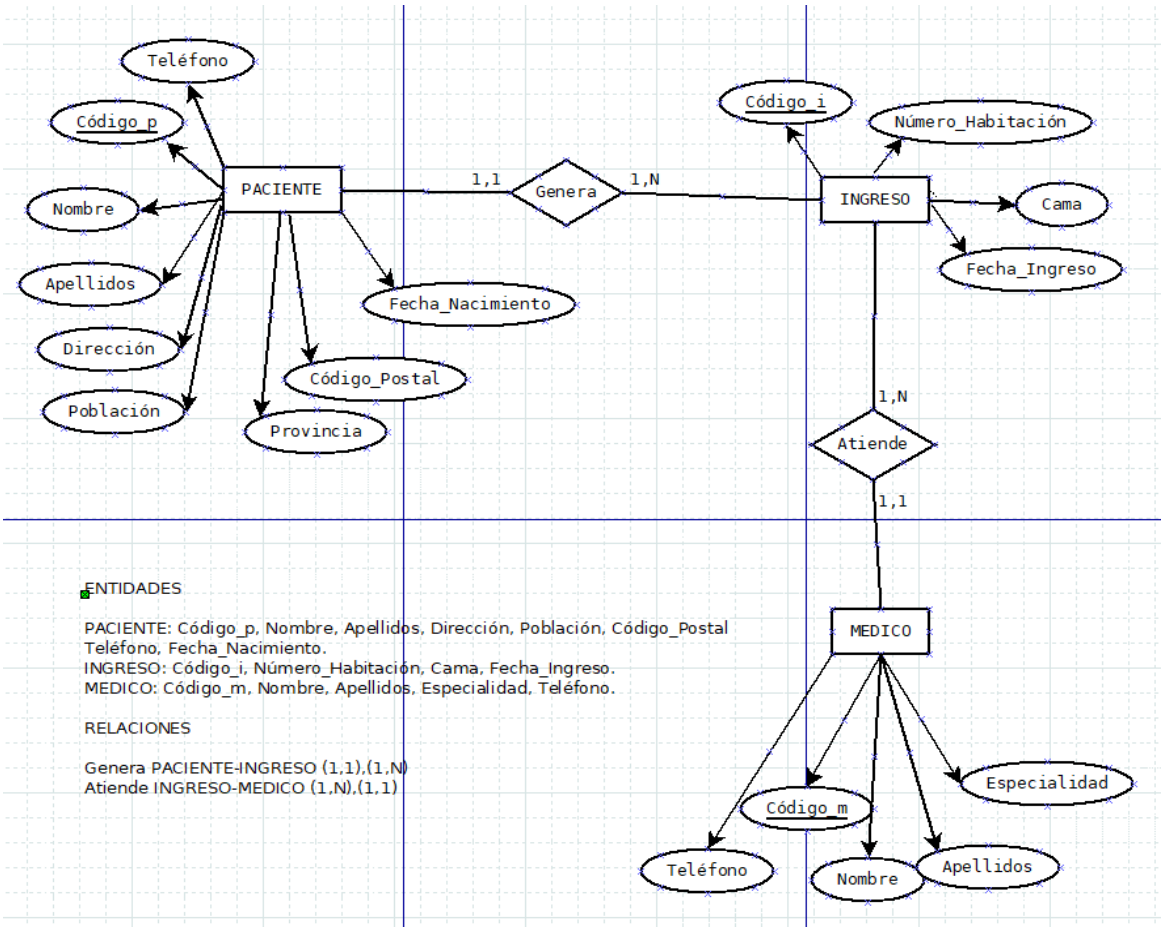
ESCUELAS SALESIANAS MARÍA AUXILIADORA

CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CENTRO SALUD PALMERITAS

Juan Manuel Ramos Villarreal
SEVILLA, 2023

APARTADO A


Teléfono
Código_p
Nombre
Apellidos
Dirección
Población
PACIENTE
Código_Postal
Provincia
Fecha_Nacimiento
1,1
Genera
1,N
Código_i
Número_Habitación
INGRESO
Cama
Fecha_Ingreso
1,N
Atiende
1,1
ENTIDADES
PACIENTE: Código_p, Nombre, Apellidos, Dirección, Población, Código_Postal
Teléfono, Fecha_Nacimiento.
INGRESO: Código_i, Número_Habitación, Cama, Fecha_Ingreso.
MEDICO: Código_m, Nombre, Apellidos, Especialidad, Teléfono.
RELACIONES
Genera PACIENTE-INGRESO (1,1),(1,N)
Atiende INGRESO-MEDICO (1,N),(1,1)
MEDICO
Especialidad
Código_m
Teléfono
Nombre
Apellidos

APARTADO B

CODIGO COMPLETO

```
-- Creación y selección de la base de datos.

DROP DATABASE IF EXISTS palmeritas;
CREATE DATABASE palmeritas;
USE palmeritas;

-- Estructura de tabla para la tabla paciente.
-- codigo_p tiene agregado UNIQUE pese a no ser necesario ya que está definido como clave
-- principal.
CREATE TABLE IF NOT EXISTS paciente(
	codigo_p int UNSIGNED NOT NULL UNIQUE,
	nombre varchar(100) NOT NULL,
	apellidos varchar(200) NOT NULL,
	direccion varchar(255) NOT NULL,
	poblacion varchar(255),
	provincia varchar(255),
	codigoPostal int (10) UNSIGNED NOT NULL,
	telefono int (10) UNSIGNED NOT NULL UNIQUE,
	fechaNacimiento DATE NOT NULL,
	PRIMARY KEY (codigo_p)
);

-- Estructura de tabla para la tabla ingreso.
-- codigo_i tiene agregado UNIQUE pese a no ser necesario ya que está definido como clave
-- principal.
CREATE TABLE IF NOT EXISTS ingreso(
	codigo_i int UNSIGNED NOT NULL  AUTO_INCREMENT,
	numeroHabitacion int (5) UNSIGNED NOT NULL,
	cama varchar(1) NOT NULL,
	CONSTRAINT ck_cama CHECK (cama IN ('A', 'B', 'C')),
	fechaIngreso DATE NOT NULL,
	codigo_p int UNSIGNED NOT NULL,
	codigo_m int UNSIGNED NOT NULL,
	PRIMARY KEY (codigo_i),
	UNIQUE (codigo_i),
	FOREIGN KEY (codigo_p) REFERENCES paciente(codigo_p),
	FOREIGN KEY (codigo_m) REFERENCES medico(codigo_m)
);

-- Estructura de tabla para la tabla medico.
-- codigo_m tiene agregado UNIQUE pese a no ser necesario ya que está definido como clave
-- principal.
CREATE TABLE IF NOT EXISTS medico(
	codigo_m int UNSIGNED NOT NULL UNIQUE,
	nombre varchar(100) NOT NULL,
	apellidos varchar(200) NOT NULL,
	telefono int (10) UNSIGNED NOT NULL,
	especialidad varchar(100) NOT NULL,
	PRIMARY KEY (codigo_m)
);
```

PUNTO 1: DENTRO DE LA TABLA PACIENTE

codigo_p int UNSIGNED NOT NULL UNIQUE,

PUNTO 2: DENTRO DE LA TABLA MÉDICO

codigo_m int UNSIGNED NOT NULL UNIQUE,

PUNTO 3: DENTRO DE LA TABLA INGRESO

codigo_i int UNSIGNED NOT NULL AUTO_INCREMENT,
UNIQUE (codigo_i),

PUNTO 4: FECHA DE INGRESO DENTRO DE LA TABLA INGRESO

fechaIngreso DATE NOT NULL,

PUNTO 5: FECHA DE NACIMIENTO DENTRO DE LA TABLA PACIENTE

fechaNacimiento DATE NOT NULL,

PUNTO 6: TELEFONO DENTRO DE LA TABLA PACIENTE

telefono Tinyint (10) UNSIGNED NOT NULL UNIQUE,

PUNTO 7: CAMA DENTRO DE LA TABLA PACIENTE

cama varchar(1) NOT NULL,
CONSTRAINT ck_cama CHECK (cama IN ('A', 'B', 'C')),

APARTADO C

MODIFICACIÓN 1

ALTER TABLE ingreso
ADD CONSTRAINT ck_cama CHECK (cama IN ('A', 'B', 'C', 'D'));

MODIFICACIÓN 2

ALTER TABLE paciente DROP COLUMN fechaNacimiento;

MODIFICACIÓN 3

ALTER TABLE paciente ADD edad int(3) UNSIGNED;

MODIFICACIÓN 4

CREATE INDEX paciente_nombre_apellidos ON paciente(nombre, apellidos);

MODIFICACIÓN 5

CREATE INDEX fecha_ingreso ON ingreso (fechaIngreso);

MODIFICACIÓN 6

ALTER TABLE paciente RENAME TO enfermo;

MODIFICACIÓN 7

DROP TABLE IF EXISTS medico;

APARTADO D

1. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE PROCEDURE agregar_paciente(
   IN codigo_p_entrada INT UNSIGNED,
   IN nombre_entrada VARCHAR(100),
   IN apellidos_entrada VARCHAR(200),
   IN direccion_entrada VARCHAR(255),
   IN poblacion_entrada VARCHAR(255),
   IN provincia_entrada VARCHAR(255),
   IN codigoPostal_entrada INT(10) UNSIGNED,
   IN telefono_entrada INT(10) UNSIGNED,
   IN fechaNacimiento_entrada DATE)
BEGIN
   INSERT INTO paciente (codigo_p, nombre, apellidos, direccion, poblacion, provincia,
codigoPostal, telefono, fechaNacimiento)
   VALUES (codigo_p_entrada, nombre_entrada, apellidos_entrada, direccion_entrada,
poblacion_entrada, provincia_entrada, codigoPostal_entrada, telefono_entrada,
fechaNacimiento_entrada);
END $$
DELIMITER ;
```

2. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE PROCEDURE agregar_Medico(
   IN codigo_m_entrada INT UNSIGNED,
   IN nombre_entrada VARCHAR(100),
   IN apellidos_entrada VARCHAR(200),
   IN telefono_entrada INT(10) UNSIGNED,
   IN especialidad_entrada VARCHAR(100))
BEGIN
   INSERT INTO medico (codigo_m, nombre, apellidos, telefono, especialidad)
   VALUES (codigo_m_entrada, nombre_entrada, apellidos_entrada, telefono_entrada,
especialidad_entrada);
END $$
DELIMITER ;
```

3. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE PROCEDURE actualizar_cama(
   IN num_ingreso INT,
   IN num_habitacion INT (5),
   IN nueva_cama VARCHAR(1))
BEGIN
   UPDATE ingreso
   SET cama = nueva_cama
   WHERE codigo_i = num_ingreso AND numeroHabitacion = num_habitacion;
END$$
DELIMITER ;
```

4. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE PROCEDURE listar_ingresos_por_paciente(IN codigo INT)
BEGIN
   SELECT p.codigo_p, p.nombre, p.apellidos, i.fechaIngreso
   FROM paciente p
   INNER JOIN ingreso i ON p.codigo_p = i.codigo_p
   WHERE p.codigo_p = codigo;
END$$
DELIMITER ;
```

5. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE PROCEDURE ingresos_medico_atendido_por_anio(IN codigo_m_entrada INT)
BEGIN
        DECLARE nombre_medico VARCHAR(100);
        DECLARE anio INT;
        DECLARE num_ingresos INT;

        SELECT nombre INTO nombre_medico
        FROM medico
        WHERE codigo_m = codigo_m_entrada;

        SELECT YEAR(fechaIngreso), COUNT(*) INTO anio, num_ingresos
        FROM ingreso
        WHERE codigo_m = codigo_m_entrada
        GROUP BY YEAR(fechaIngreso);

        SELECT nombre_medico, anio, num_ingresos;

END $$
DELIMITER ;
```

6. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE FUNCTION ingresos_por_paciente(codigo INT) RETURNS VARCHAR(255)
BEGIN
	DECLARE num_ingresos INT;
	DECLARE nombre_apellidos VARCHAR(300);

	SELECT CONCAT(paciente.nombre, ' ', paciente.apellidos), COUNT(*)
	INTO nombre_apellidos, num_ingresos
	FROM ingreso
	INNER JOIN paciente ON ingreso.codigo_p = paciente.codigo_p
	WHERE paciente.codigo_p = codigo;

	RETURN CONCAT('El paciente ', nombre_apellidos, ' ha tenido ', num_ingresos, '
ingresos.');
END $$
DELIMITER ;
```

7. PROCEDIMIENTO ALMACENADO

```
DELIMITER $$
CREATE FUNCTION num_ingresos_por_medico(codigo_m_entrada INT) RETURNS INT
BEGIN
	DECLARE num_ingresos INT;
	SELECT COUNT(*) INTO num_ingresos
	FROM ingreso
	WHERE codigo_m = codigo_m_entrada;
	RETURN num_ingresos;
END $$
DELIMITER ;
```

8. PROCEDIMIENTO ALMACENADO

```
CREATE VIEW vista_ingresos AS
SELECT paciente.nombre AS nombre_paciente, paciente.apellidos AS apellidos_paciente,
medico.nombre AS nombre_medico, medico.apellidos AS apellidos_medico,
ingreso.fechaIngreso
FROM ingreso
INNER JOIN paciente ON ingreso.codigo_p = paciente.codigo_p
INNER JOIN medico ON ingreso.codigo_m = medico.codigo_m;
```