



GRUPO 6

BASE DE DATOS HOSPITAL

PRESENTADO POR:

Ahumada, Iván

Civiero, Tomás

Arce, Ezequiel

Magliotti, Gian Franco

Cármenes, Santiago

Rasso, Micaela

Índice de Contenidos

01 MER

02 MR y Normalización

03 Diseño físico

MER

Modelo Entidad-Relación

Se realizó a partir de la confección de un DER (Diagrama Entidad-Relación), donde se modeló lo solicitado del TP, se definieron hipótesis y restricciones adicionales

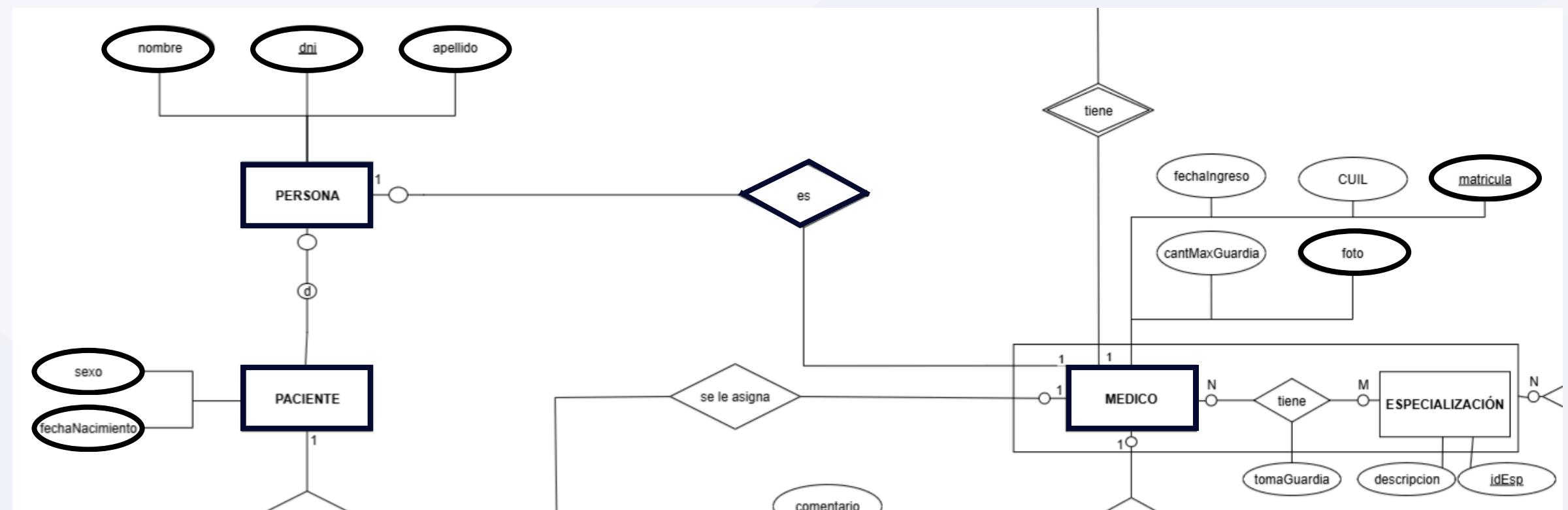
Requerimientos del director del hospital

REQUERIMIENTOS:

- Un **Paciente** se identifica por su número de DNI.
- De un **Paciente** se debe conocer su nombre, apellido, sexo y fecha de nacimiento.
- Un **Médico** se identifica por su número de matrícula.
- De un **Médico** se debe conocer su DNI, nombre, apellido y foto.
- **Todo Médico puede ser también Paciente.**

IMPLEMENTACIÓN:

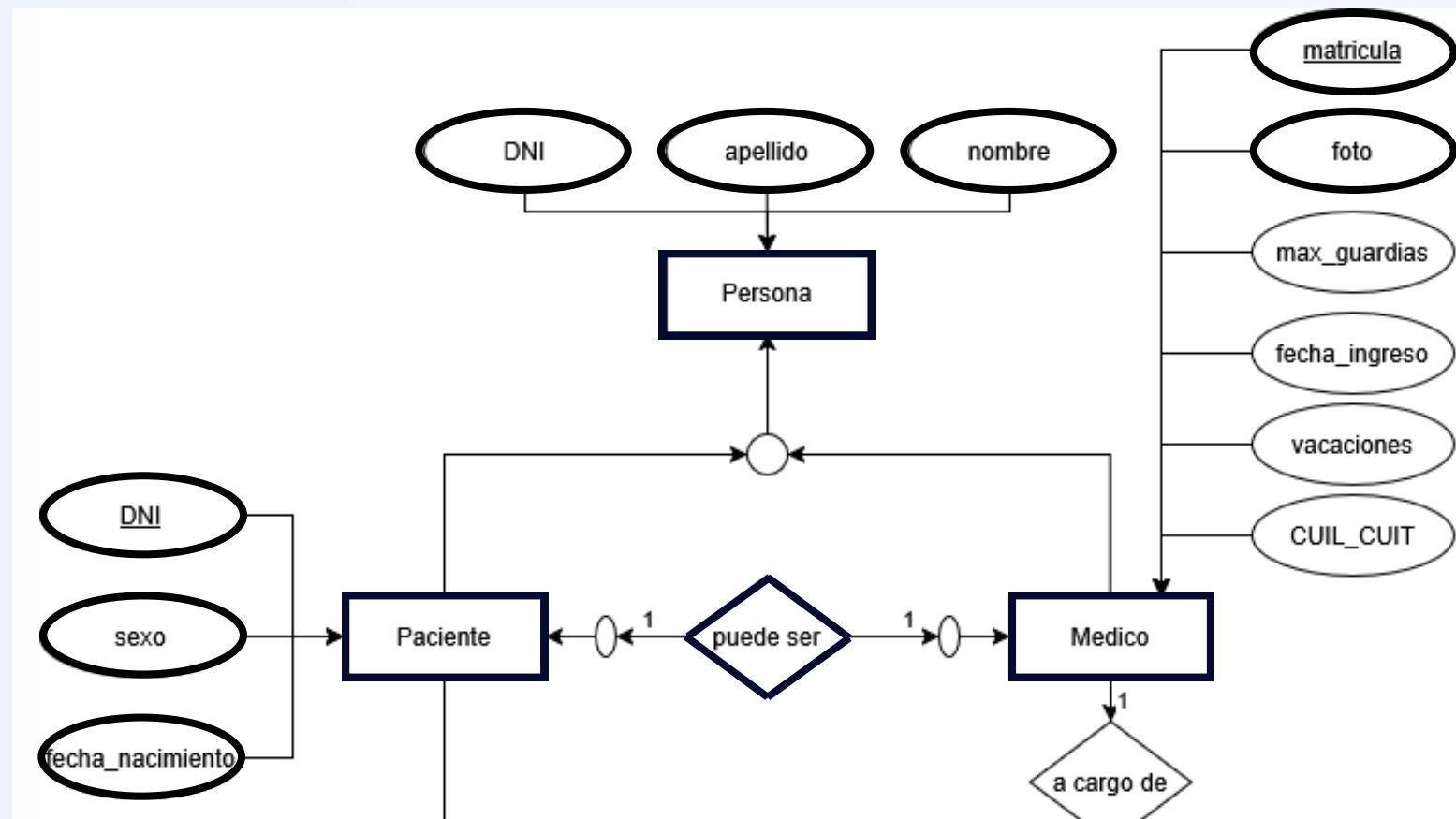
- Entidad **Persona**.
- **Jerarquía pura** de Persona a Paciente.
- **Relación 1:1** entre Persona y Médico.



Requerimientos del director del hospital

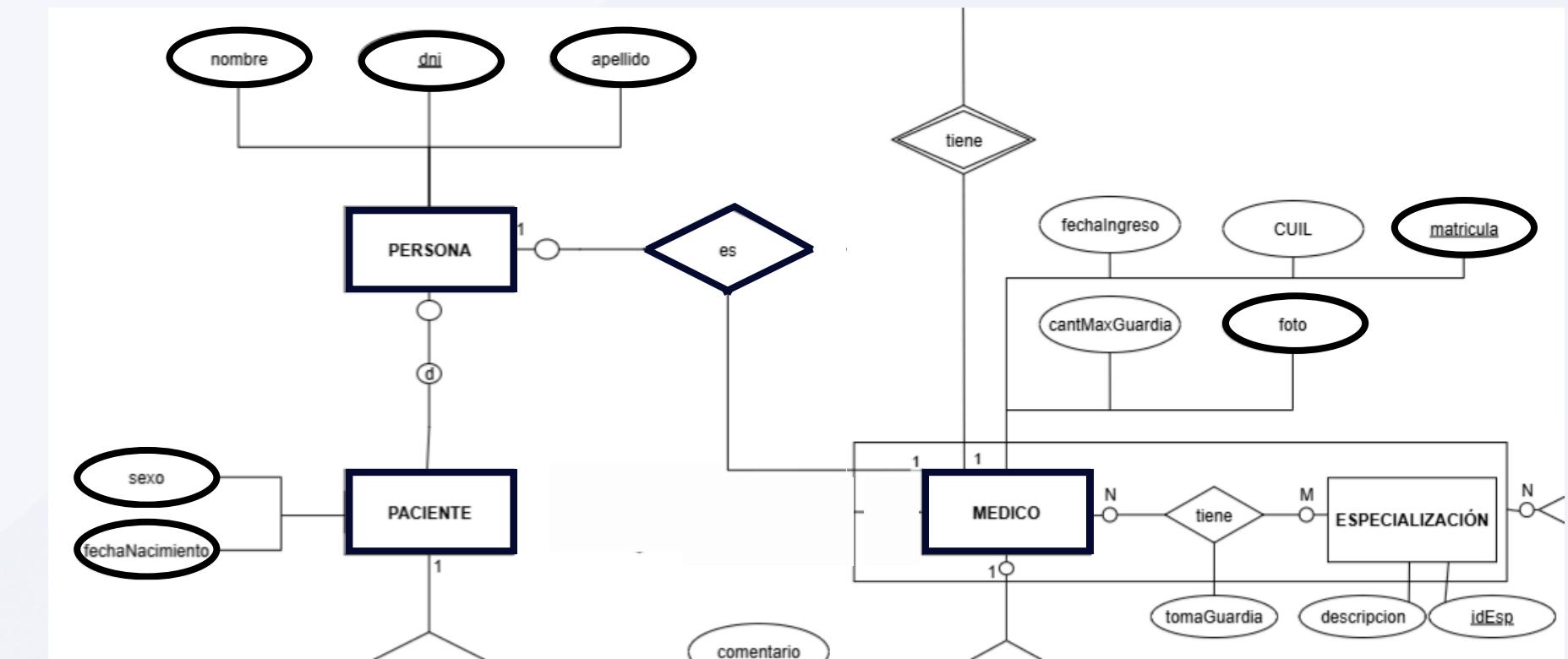
PRIMER DISEÑO:

- Relación de jerarquía, **Paciente** y **Médico** como especializaciones de **Persona**.
- Interrelación entre **Paciente** y **Persona**, con cardinalidad 1:1 y participación parcial de ambas entidades.

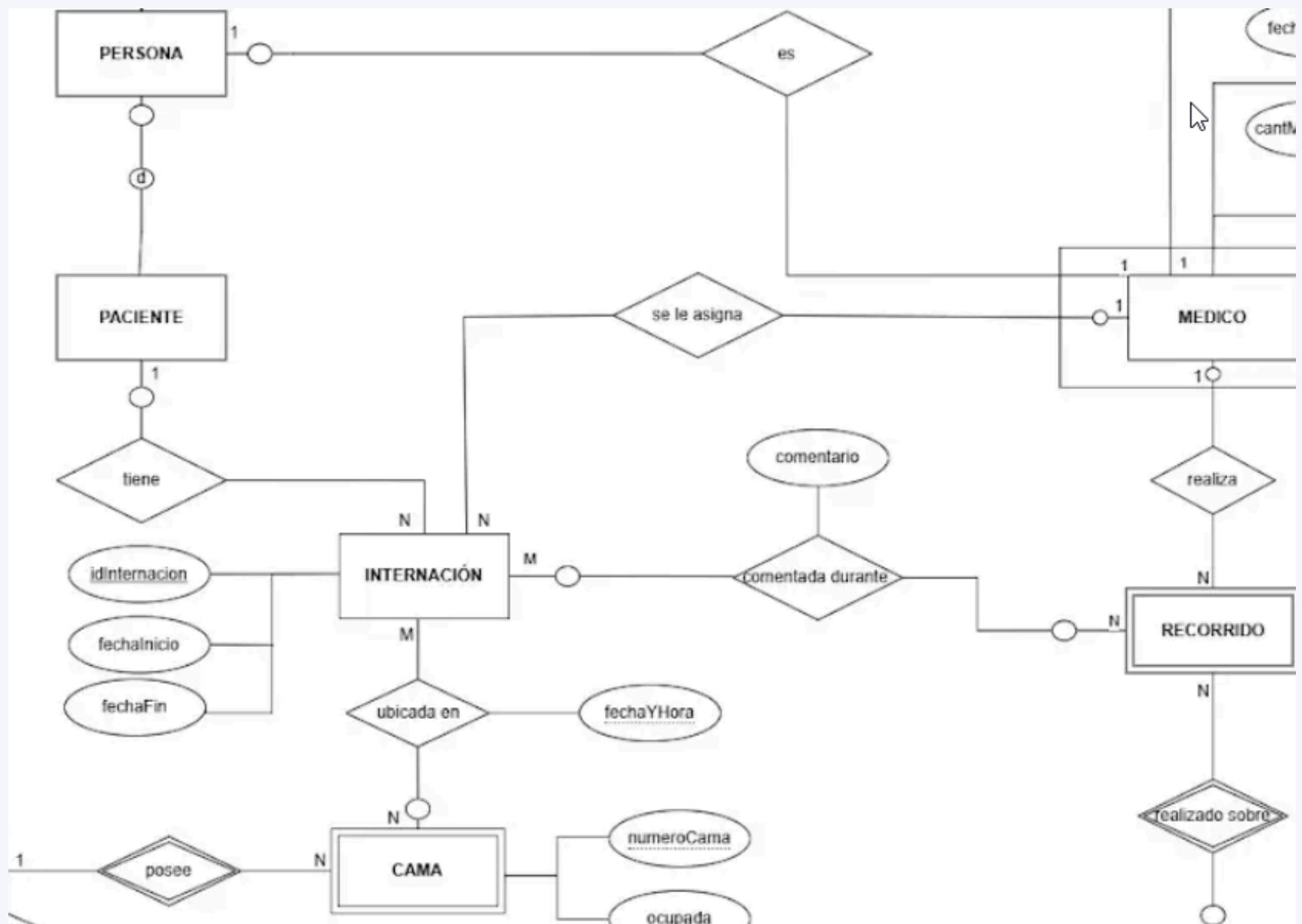


DISEÑO FINAL:

- Relación de jerarquía, **Paciente** como especialización de **Persona**.
- **Médico** tiene una interrelación con **Persona**, con cardinalidad 1:1 y participación parcial de **Persona**.



Área de Internación



CARACTERÍSTICAS:

- Administra la ocupación hospitalaria
- Identificación de las Camas
- Gestión de Estados
- Historial de Movimientos.
- Lógica Temporal
- Asignación de personal
- Cama, Paciente, Recorrido, Médico

Área de Seguimiento médico

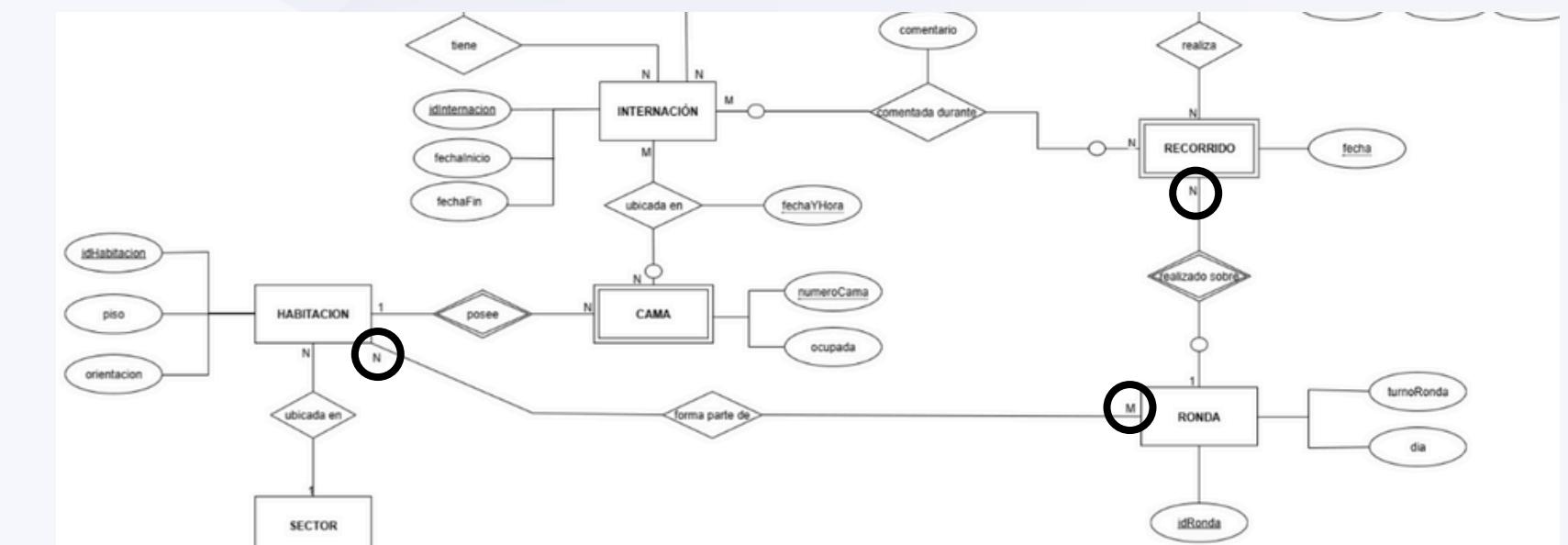
REQUERIMIENTOS:

- Un **Paciente** tiene un **médico** principal por **internación**.
- Un **Médico** puede atender **múltiples internaciones**.
- **Recorridos** sobre **una ronda**, **comentarios**.
- **Rondas**, múltiples habitaciones, con **múltiples recorridos**.



IMPLEMENTACIÓN:

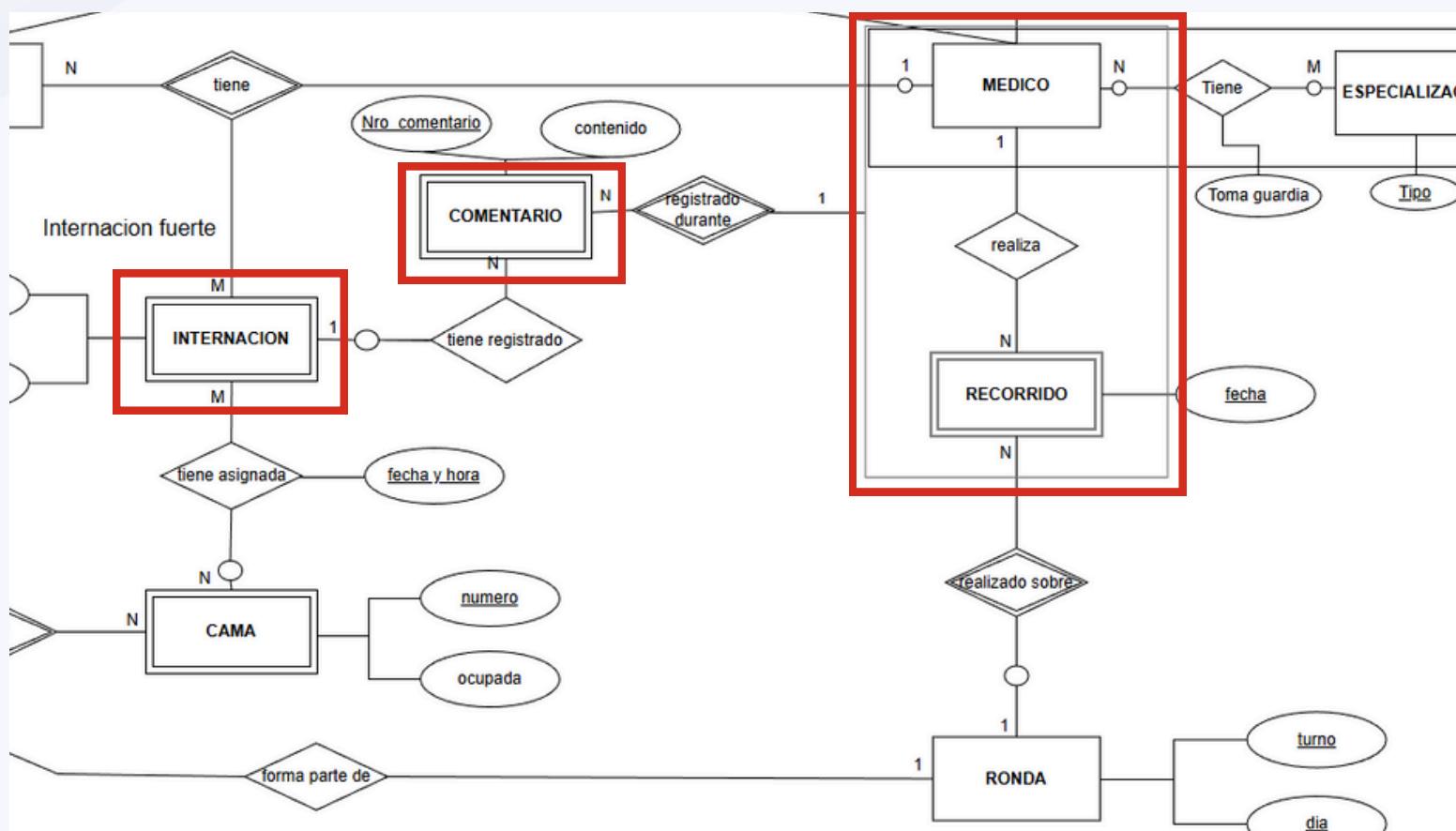
- **Cardinalidades**.
- Como el **recorrido** depende de la **ronda**, **entidad débil**.
- **Comentario** es una **relación** entre **recorrido** e **internación**
- **Restricción adicional**: Un médico no puede ser su propio paciente.



Área de Seguimiento médico

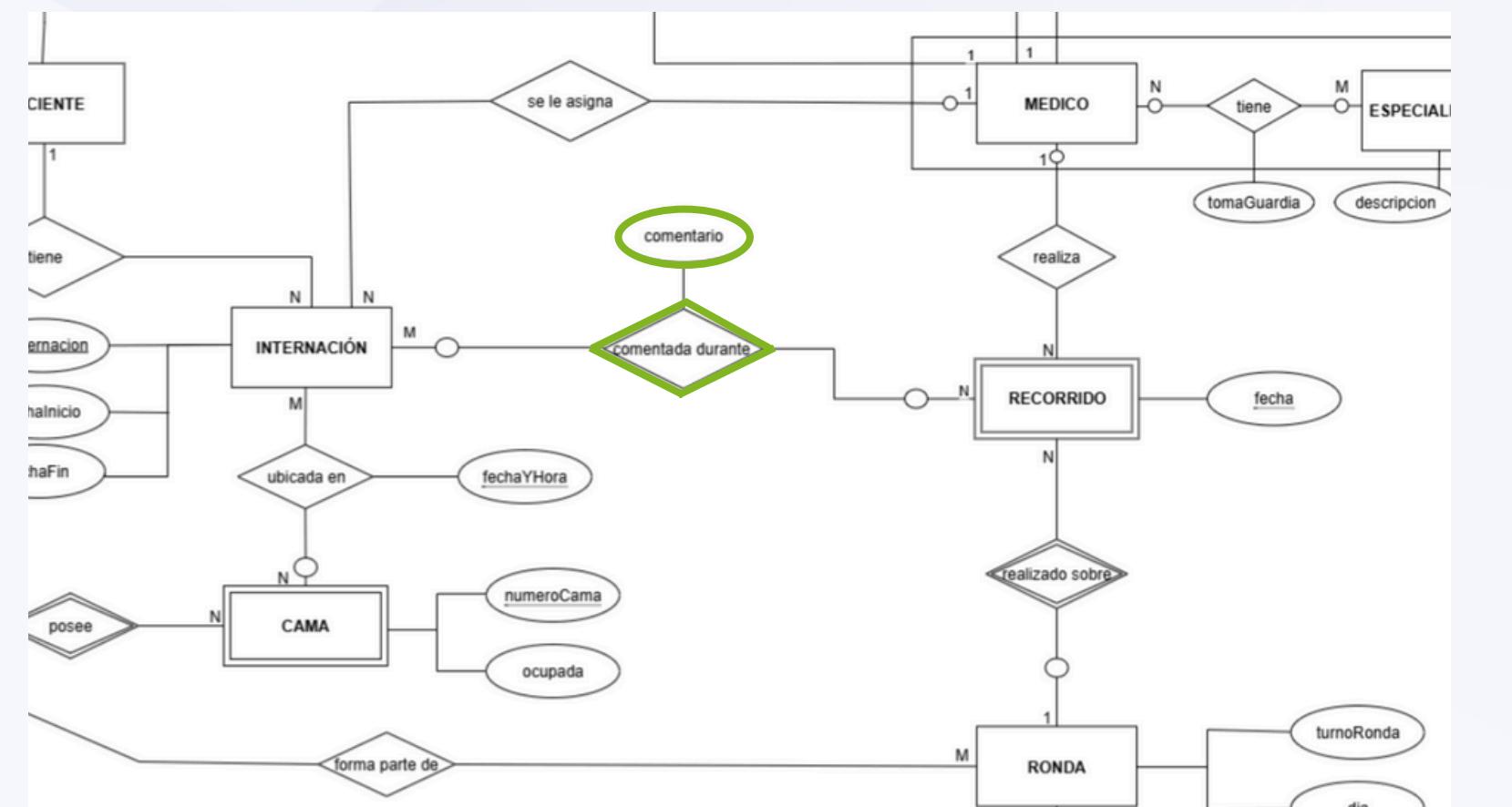
PRIMER DISEÑO:

- **Internación** como **entidad débil**.
- Comentario como entidad.
- **Agregación** Médico-Recorrido.
- Relaciones innecesarias.



DISEÑO FINAL:

- Comentada como una relación entre Recorrido e Internación.
- Recorrido tiene información de Médico por medio de la FK matricula, se saco la agregación.
- Internación es una entidad fuerte.



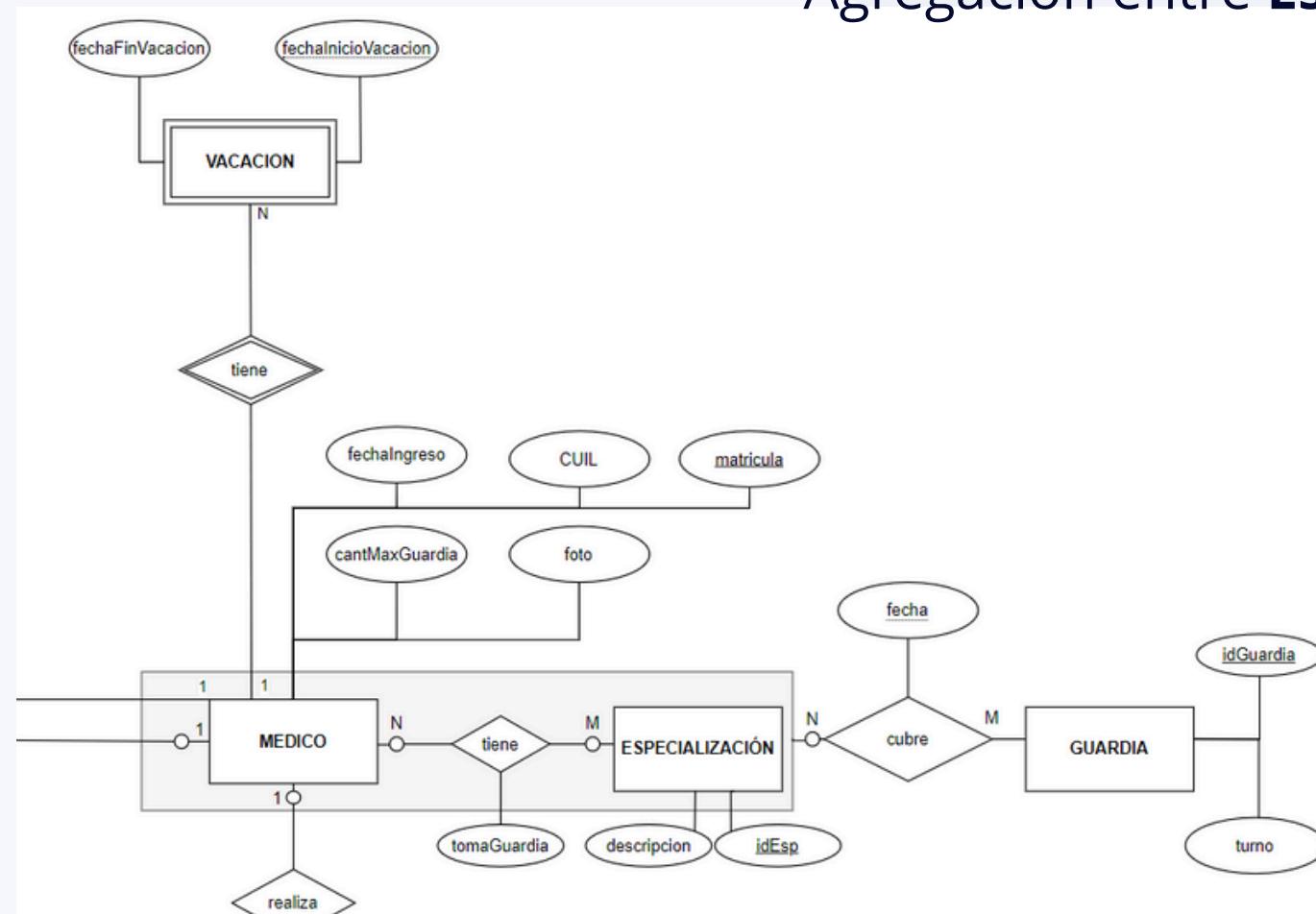
Área de Guardias

REQUERIMIENTOS:

- **3 turnos diarios** para cada especialidad.
- Conocer la **especialidad de cada médico**.
- **Cantidad máxima guardia** por médico.
- CUIT/CUIL, fecha ingreso hospital y períodos de vacaciones del **médico**.

IMPLEMENTACIÓN:

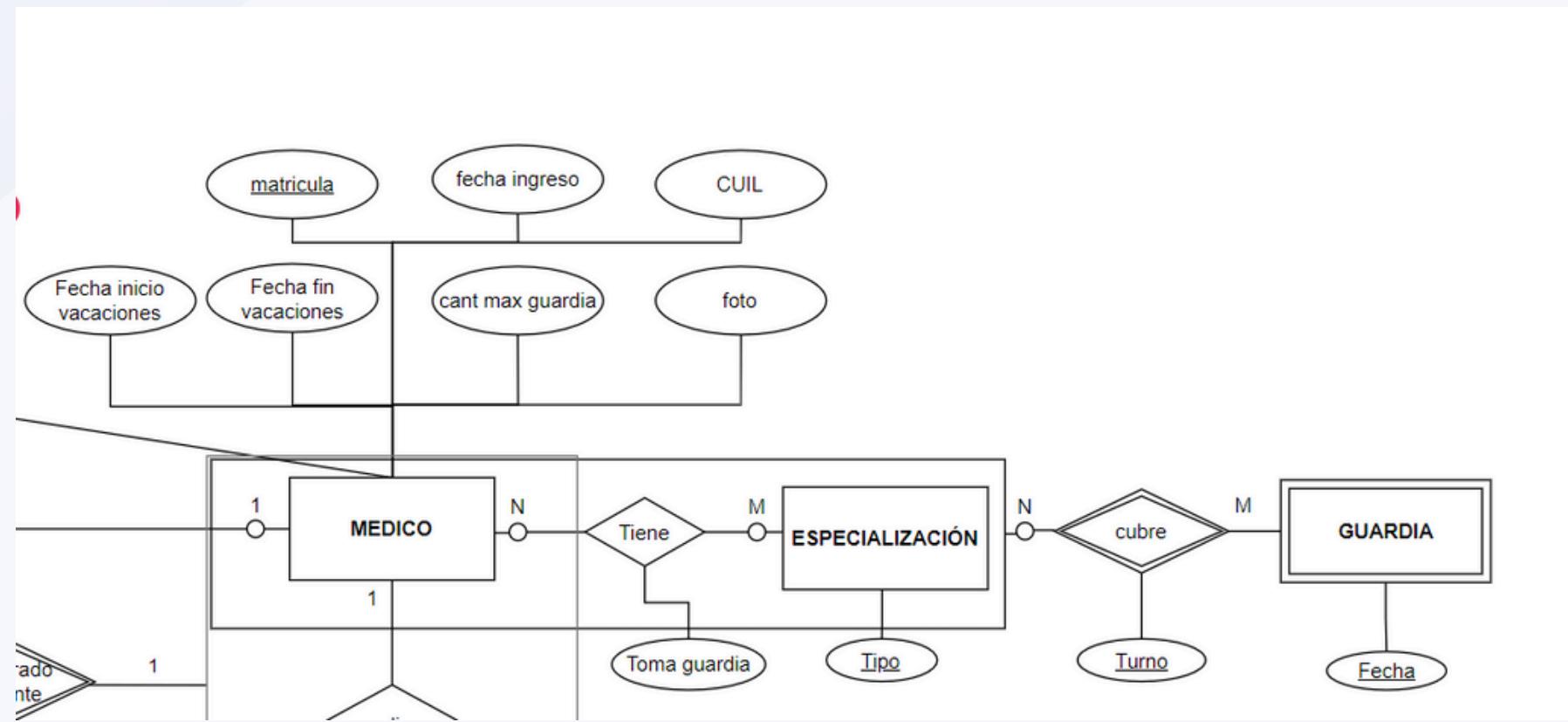
- Relación **Médico-Especialización**
- Relación **Médico-Vacación**
- Atributos CUIT/CUIL sobre **Médico**
- Atributo cantidad máxima de guardias en **Médico**
- Atributo tomaGuardia sobre la relación entre **Médico-Especialización**
- Agregación entre **Espacialización-Médico-Guardia**



Área de Guardias

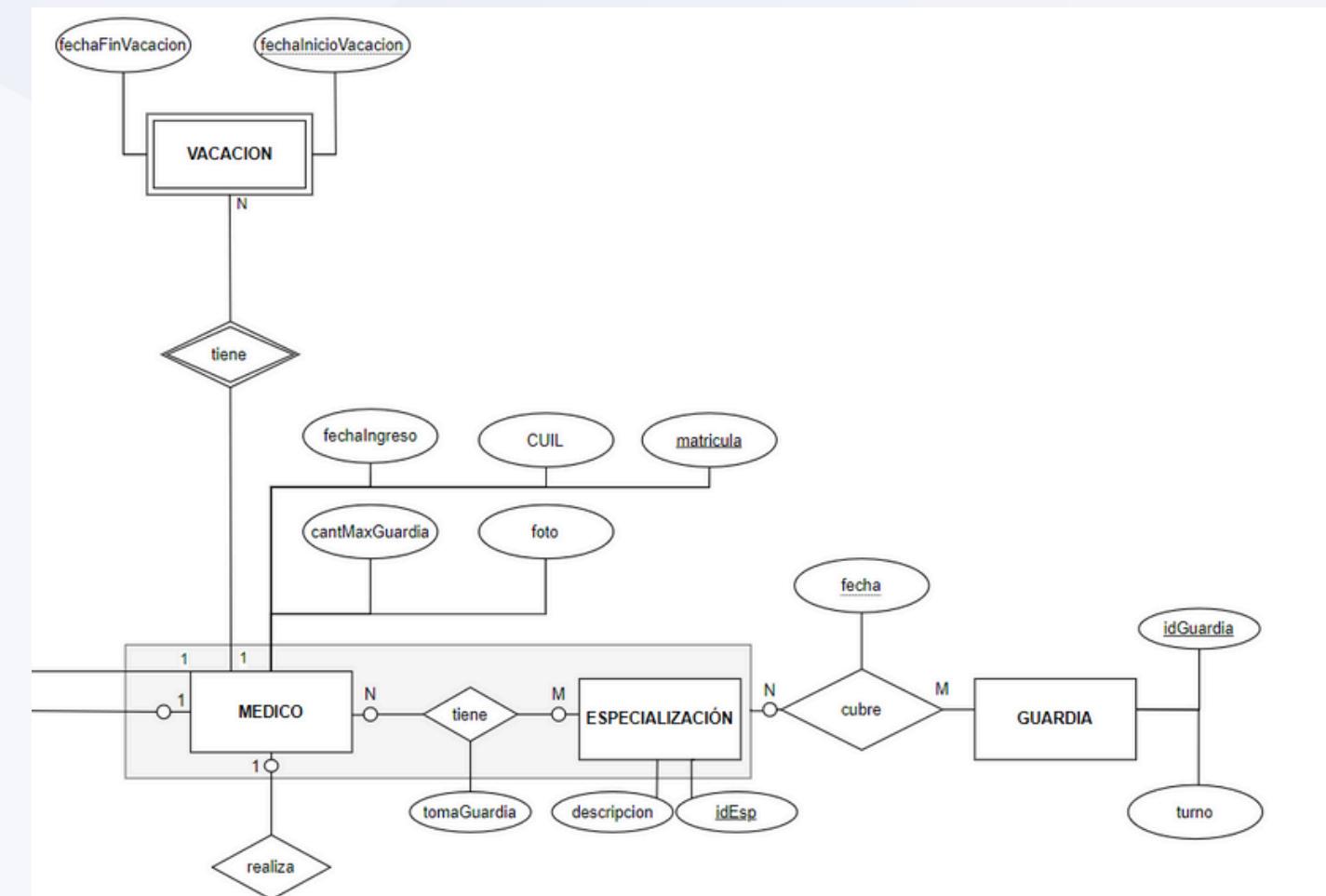
PRIMER DISEÑO:

- **Guardia** como **entidad débil**.
- Período de vacación como atributo, único período de vacaciones.
- Turno como atributo y **PK** en la relación **Cubre**
- Fecha como atributo y **PK** en **Guardia**



DISEÑO FINAL:

- **Guardia** como **entidad fuerte**.
- Vacación como **entidad débil**, permite múltiples períodos de vacaciones.
- Turno como atributo en **Guardia**.
- Fecha como atributo y **PK** en **Cubre**



MR

Modelo lógico relacional y normalización

Se transformó el DER al modelo lógico relacional. Luego se verificó que en forma normal cumplían las relaciones.

Modelo relacional

ESQUEMAS RESULTANTES:

Medico(matricula, cuil, cantMaxGuardia, fechalngreso, foto, dni)

CK = {(matricula), (cuil), (dni)}

PK = {(matrícula)}

FK = {(dni)}

Especialización(idEsp, descripción)

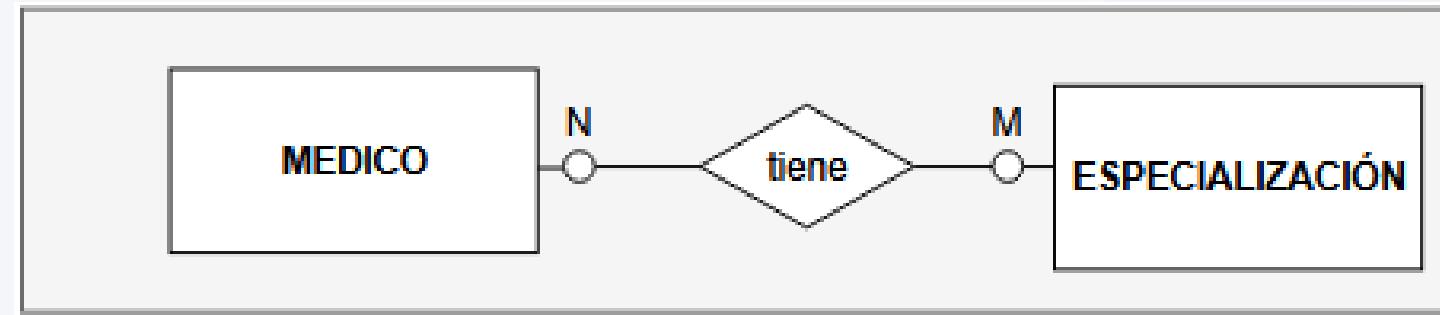
CK = PK = {(idEsp)}

FK = {}

MedicoEspecializado(matrícula, IdEsp, tomaGuardia)

CK = PK = {(matrícula, IdEsp)}

FK = {(matrícula), (IdEsp)}



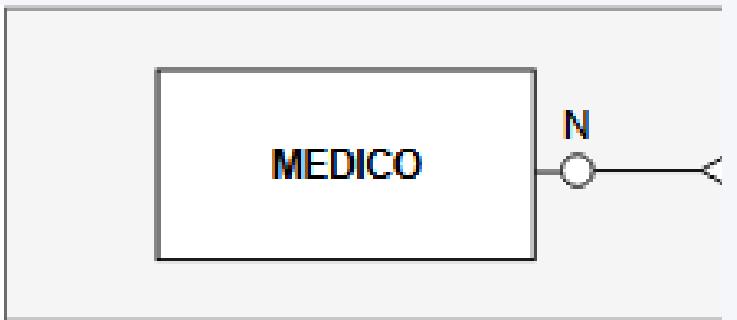
RESTRICCIONES ADICIONALES:

- MedicoEspecializado.matricula debe estar en Medico.matricula
- Medico.matricula puede no estar en MedicoEspecializado.matricula
- MedicoEspecializado.IdEsp debe estar en Especializacion.IdEsp
- Especializacion.IdEsp puede no estar en MedicoEspecializado.IdEsp

Normalización

Verificación FNBC

R está en FNBC si para toda dependencia funcional no trivial $X \rightarrow A$ definida sobre R,
X es superclave de R.



Medico(matricula, cuil, cantMaxGuardia, fechalIngreso, foto, dni)

$F = \{ \text{matricula} \rightarrow \text{cuil}, \text{cantMaxGuardia}, \text{fechalIngreso}, \text{foto}, \text{dni}$
 $\text{cuil} \rightarrow \text{matricula}, \text{cantMaxGuardia}, \text{fechalIngreso}, \text{foto}, \text{dni}$
 $\text{dni} \rightarrow \text{cuil}, \text{cantMaxGuardia}, \text{fechalIngreso}, \text{foto}, \text{matricula} \}$

$CK = \{(\text{matrícula}), (\text{cuil}), (\text{dni})\}$

Atributos primos = { dni, matricula, cuil }

Atributos no primos = { cantMaxGuardia, fechalIngreso, foto }

FNBC: los atributos a la izquierda de las DF son superclaves.

SQL

Implementación en SQL Server

Se implementó lo definido en la etapa de diseño con el MER en el sistema de gestión de bases de datos MSSQL (Microsoft SQL Server), utilizando el lenguaje de consulta T-SQL.

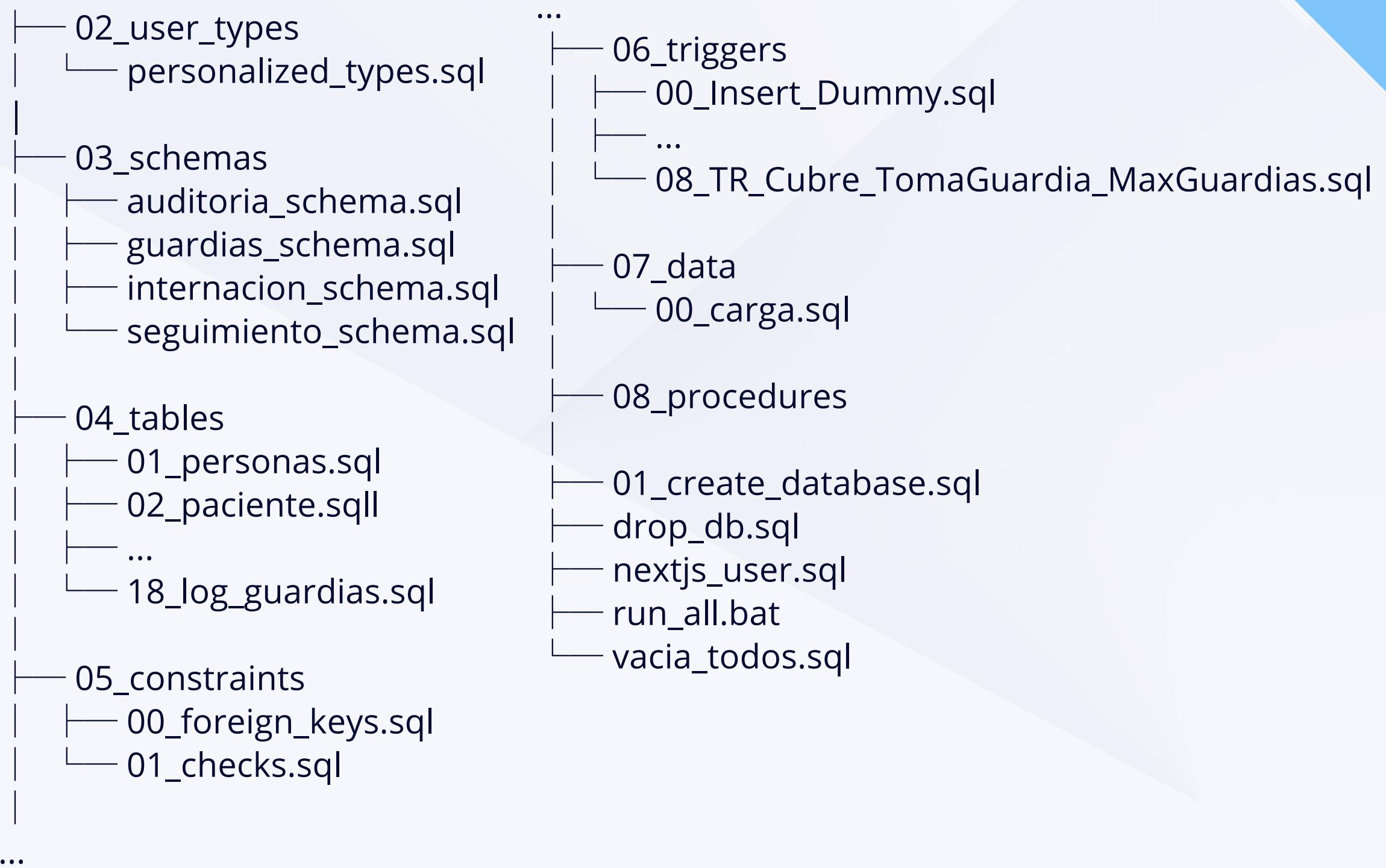
Organización de Archivos

CRITERIOS:

- Separación de tareas.
- Mantenibilidad.
- Orden.

ARCHIVOS NOTABLES:

- run_all.bat
- drop_db.sql
- nextjs_user.sql
- vacia_todos.sql



Definición de datos

DATA TYPES

```
3  
4  -- Data types  
5  
6  CREATE TYPE dni_type FROM [varchar](10);  
7  GO  
8  CREATE TYPE matricula_type FROM [nvarchar](20);  
9  GO  
10 CREATE TYPE imagen_type FROM VARBINARY(MAX);  
11 GO  
12 CREATE TYPE nombre_type FROM NVARCHAR(30)  
13 GO
```

SCHEMAS

```
1  USE [Hospital];  
2  GO  
3  
4  CREATE SCHEMA Auditoria;  
5  GO
```



```
1  USE [Hospital];  
2  GO  
3  
4  CREATE SCHEMA Internacion;  
5  GO
```

```
1  USE [Hospital];  
2  GO  
3  
4  CREATE SCHEMA Guardias;  
5  GO
```



```
1  USE [Hospital];  
2  GO  
3  
4  CREATE SCHEMA Seguimiento;  
5  GO
```

TABLES

```
1  USE [Hospital]  
2  GO  
3  
4  CREATE TABLE Internacion.Habitacion (  
5    idHabitacion INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
6    piso TINYINT NOT NULL,  
7    orientacion NVARCHAR(20) NOT NULL,  
8    idSector INT NOT NULL,  
9  );  
10  
11  GO
```

Limitación TINYINT

Carga de datos y constraints

FOREIGN KEYS

```
23    -- Medico especializado
24
25    ALTER TABLE Guardias.MedicoEspecializado
26        ADD CONSTRAINT FK_MedicoEspecializado_Medico
27            FOREIGN KEY (matricula)
28            REFERENCES Internacion.Medico
29            ON DELETE CASCADE;
30
31    ALTER TABLE Guardias.MedicoEspecializado
32        ADD CONSTRAINT FK_MedicoEspecializado_Especializacion
33            FOREIGN KEY (idEsp)
34            REFERENCES Guardias.Especializacion
35            ON DELETE CASCADE;
36
37    GO
```

CHECKS

```
10    ALTER TABLE Internacion.Vacacion  
11        ADD CONSTRAINT CK_Vacacion_Fechas  
12            CHECK (fechaFinVacacion >= fechaInicioVacacion);  
13    GO
```

INSERCIÓNES

```
186 -- Tabla Guardias.Especializacion
187 INSERT INTO Guardias.Especializacion (Description) VALUES
188 ('Cardiología'), -- id 1
189 ('Pediatría'), -- id 2
190 ('Traumatología'), -- id 3
191 ('Gastroenterología'), -- id 4
192 ('Neurología'), -- id 5
193 ('Dermatología') -- id 6
```

Procedures: Comentarios de una internación

DESCRIPCIÓN:

Listado de los comentarios de las visitas médicas a un paciente en una cierta internación ordenado por la fecha en la que se realizaron.

ESQUEMA PERTENECIENTE:

Seguimiento

PARAMETROS:

@idInternacion INT

TABLAS QUE UTILIZA:

- ComentadaDurante
- Recorrido
- Medico
- Persona

```
4  CREATE OR ALTER PROCEDURE Seguimiento.ListarComentariosInternacion
5  (
6  |   @idInternacion INT
7  )
8  AS
9  BEGIN
10 |   SET NOCOUNT ON;
11 |
12 |   BEGIN TRY
13 |       SELECT
14 |           cd.idInternacion,
15 |           cd.idRonda,
16 |           cd.fecha AS fechaComentario,
17 |           cd.comentario,
18 |           r.matricula AS medicoMatricula,
19 |           p.apellido + ', ' + p.nombre AS medicoNombre
20 |       FROM ComentadaDurante cd
21 |       JOIN Recorrido r
22 |           ON cd.idRonda = r.idRonda AND cd.fecha = r.fecha
23 |       JOIN Internacion.Medico m
24 |           ON m.matricula = r.matricula
25 |       JOIN Internacion.Persona p
26 |           ON p.dni = m.dni
27 |       WHERE cd.idInternacion = @idInternacion
28 |       ORDER BY cd.fecha;
29 |
30 |   END TRY
31 |   BEGIN CATCH
32 |       THROW; -- Reenvía el error al cliente
33 |   END CATCH;
34 |END;
35 |GO
```

Procedures: Consultar Auditoria Guardias

DESCRIPCIÓN:

Obtiene la auditorias realizadas sobre los usuarios que hacen cambios a datos que afectan el proceso de asignación de guardias.

ESQUEMA PERTENECIENTE:

Guardias

PARAMETROS:

- @usuario VARCHAR(100)
- @accion VARCHAR(20)
- @fecha DATETIME

TABLAS QUE UTILIZA:

- LogGuardias

```
CREATE OR ALTER PROCEDURE Guardias.ConsultarLogGuardias
(
    @usuario      VARCHAR(100) = NULL,
    @accion       VARCHAR(20)  = NULL,
    @fecha        DATETIME    = NULL
)
AS
BEGIN
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

    BEGIN TRY
        BEGIN TRAN;

        SELECT
            *
        FROM Auditoria.LogGuardias
        WHERE
            (@usuario IS NULL OR usuario = @usuario)
            AND (@accion IS NULL OR accion = @accion)
            AND (@fecha IS NULL OR fecha = @fecha)
        ORDER BY fecha DESC, idLog DESC;

        COMMIT;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK;

        THROW;
    END CATCH
END;
GO
```

Procedures: Listado de camas disponibles (cantidad)

DESCRIPCIÓN:

Obtiene un listado con la cantidad de camas disponibles en el hospital en el momento de la consulta.

ESQUEMA PERTENECIENTE:

Internación

TABLAS QUE UTILIZA:

- Sector
- Habitación
- Camas

```
USE [Hospital]
GO

CREATE OR ALTER PROCEDURE Internacion.ListadoCantCamasDisponiblesPorSector
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        SELECT
            s.idSector,
            s.descripcion AS Sector,
            COUNT(*) AS CantidadCamasDisponibles
        FROM
            Sector s
            INNER JOIN Habitacion h ON s.idSector = h.idSector
            INNER JOIN Cama c ON h.idHabitacion = c.idHabitacion
        WHERE c.ocupada = 0
        GROUP BY s.idSector, s.descripcion
    END TRY
    BEGIN CATCH
        THROW; --Reenvía el error al cliente
    END CATCH
END;
GO
```

Procedures: Listado de camas disponibles (detalle)

DESCRIPCIÓN:

Obtiene un listado con los detalles de camas disponibles en el hospital. Indica sector, descripción del sector, zona, habitación y número de cama.

ESQUEMA PERTENECIENTE:

Internación

TABLAS QUE UTILIZA:

- Sector
- Habitación
- Camas

```
USE [Hospital]
GO

CREATE OR ALTER PROCEDURE Internacion.ListadoDetalleCamasDisponiblesPorSector
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        SELECT
            s.idSector,
            s.descripcion AS Sector,
            h.idHabitacion,
            h.piso,
            h.orientacion,
            c.nroCama
        FROM
            Sector s
            INNER JOIN Habitacion h ON s.idSector = h.idSector
            INNER JOIN Cama c ON h.idHabitacion = c.idHabitacion
        WHERE c.ocupada = 0
        ORDER BY s.idSector, s.descripcion, h.idHabitacion, c.nroCama;
    END TRY
    BEGIN CATCH
        THROW;
    END CATCH;
END;
GO
```

Triggers: Validacion de reglas complejas

OBJETIVO:

Garantizar la integridad de los datos mas alla de las restricciones estandar.

ESTRATEGIA:

Uso de triggers INSTEAD OF para interceptar las operaciones (INSERT, UPDATE) antes de que impacten en la tabla

LISTADO:

Seguimiento.TR_Recorrido_Fecha_Before

Seguimiento.TR_Recorrido_Fecha_Iol

Internacion.TR_Internacion_FechaFin

Internacion.TR_UbicadaEn_FechaEntrada

Internacion.TR_Internacion_DistintoPacienteYMedico

Guardias.TR_Cubre_TomaGuardia_MaxGuardias

Internacion.TR_Habitacion_Orientacion

Guardias.TR_Guardia_LogGuardias

Triggers: Integridad de recorridos médicos

PROBLEMA:

La entidad Ronda define un día fijo, pero el sistema permitía cargar un recorrido que no respete esta fecha.

SOLUCIÓN:

TR_Recorrido_Fecha_Before.

Compara el día de la semana de la fecha real con la configuración de la ronda, usando DATENAME.

```
-- Validación de consistencia de día

IF EXISTS (
    SELECT 1
    FROM inserted i
    JOIN Seguimiento.Ronda r ON r.idRonda = i.idRonda
    WHERE DATENAME(WEEKDAY, i.fecha) <> r.dia
)
BEGIN
    RAISERROR('La fecha no coincide con el día de la ronda.', 16, 1);
    RETURN;
END
```

Triggers: Restricciones éticas y operativas

REGLA:

Un médico no puede estar a cargo de su propia internación.

SOLUCIÓN:

TR_Internacion_DistintoPacienteYMedico.

Hace un JOIN entre Médico y la tabla insertada usando la matrícula, si el DNI del médico coincide con el del paciente, se rechaza la transacción.

```
IF EXISTS (
    SELECT 1
    FROM inserted i
    INNER JOIN Internacion.Medico m ON m.matricula = i.matricula
    WHERE i.dni = m.dni -- El médico es el mismo paciente
)
BEGIN
    RAISERROR('Error: Un paciente no puede ser su propio médico.', 16, 1);
    RETURN;
END
```

Transacciones: Actualizar fecha recorrido

DESCRIPCIÓN:

Actualiza la fecha asociada a un recorrido existente, validando de que el registro exista y realiza uso de un nivel de aislamiento Serializable

ESQUEMA PERTENECIENTE:

Seguimiento

PARAMETROS:

- @idRonda INT
- @fechaActual DATE
- @nuevaFecha DATE

TABLAS QUE UTILIZA:

- Recorridos

```
1 USE Hospital;
2 GO
3
4 CREATE OR ALTER PROCEDURE Seguimiento.ActualizarFechaRecorrido
5 (
6     @idRonda INT,
7     @fechaActual DATE,
8     @nuevaFecha DATE
9 )
10 AS
11 BEGIN
12     SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
13
14     BEGIN TRY
15         BEGIN TRAN;
16
17         UPDATE Seguimiento.Recorrido
18             SET fecha = @nuevaFecha
19             WHERE idRonda = @idRonda
20             AND fecha = @fechaActual;
21
22         IF @@ROWCOUNT = 0
23             BEGIN
24                 DECLARE @fechaStr VARCHAR(10) = CONVERT(VARCHAR(10), @fechaActual, 120);
25                 RAISERROR(
26                     'No se encontró el recorrido a actualizar (idRonda=%d, fecha=%s).',
27                     16, 1, @idRonda, @fechaStr
28                 );
29                 ROLLBACK TRAN;
30                 RETURN;
31             END
32
33             COMMIT TRAN;
34
35             SELECT *
36             FROM Seguimiento.Recorrido
37             WHERE idRonda = @idRonda
38             AND fecha = @nuevaFecha;
39     END TRY
40     BEGIN CATCH
41         IF @@TRANCOUNT > 0 ROLLBACK TRAN;
42
43         THROW;
44     END CATCH
45 END;
46 GO
```

Transacciones: % camas libres por sector

DESCRIPCIÓN:

Obtiene el porcentaje de camas disponibles por sector. La operación se realiza dentro de una transacción con nivel aislamiento REPEATABLE READ.

ESQUEMA PERTENECIENTE:

Internacion

PARAMETROS:

Tablas temporales en memoria)

- @camasDisponiblesPorSector (idSector, CantidadCamasDisponibles)
- @camasTotalesPorSector (idSector, CantidadCamas)

TABLAS QUE UTILIZA:

- Sector
- Habitacion
- Cama

```
4 CREATE OR ALTER PROCEDURE Internacion.ListadoPorcentajeCamasDisponiblesPorSector
5 AS
6 BEGIN
7     SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; -- Serializable es excesivo para este problema
8
9     BEGIN TRY
10        BEGIN TRAN;
```

```
43    SELECT
44        cd.idSector,
45        s.descripcion,
46        CAST(cd.CantidadCamasDisponibles AS DECIMAL(10,2))
47        / CAST(ct.CantidadCamas AS DECIMAL(10,2)) * 100 AS PorcentajeDisponible
48    FROM @camasDisponiblesPorSector cd
49    JOIN @camasTotalesPorSector ct ON ct.idSector = cd.idSector
50    JOIN Sector s ON s.idSector = cd.idSector
51    ORDER BY
52        CAST(cd.CantidadCamasDisponibles AS DECIMAL(10,2))
53        / CAST(ct.CantidadCamas AS DECIMAL(10,2)) * 100 DESC;
54
55    COMMIT TRAN;
```



UNIVERSIDAD NACIONAL
de MAR DEL PLATA

¡Muchas gracias!

¿Consultas?



Grupo 6

BASE DE DATOS NO SQL

Presentado por:

Ahumada, Iván

Arce, Ezequiel

Cármenes, Santiago

Civiero, Tomás

Magliotti, Gian Franco

Rasso, Micaela

Paradigma NoSQL

¿EN QUE CONSISTE?

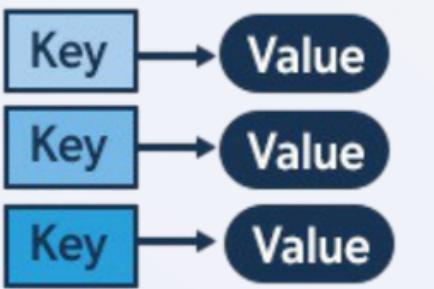
- Estructura flexible
- Naturaleza distribuida
- Orientación a agregados

PROBLEMAS QUE INTENTA RESOLVER

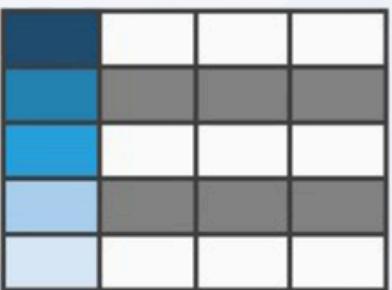
- Problemas de escalabilidad
- Rigidez de los datos
- Teorema CAP

NoSQL

Key-Value



Column-Family



Graph



Document



Bases de datos Clave-Valor

¿EN QUE CONSISTE?

- Estructura basada en **IDs** o **claves**.
- Los agregados son **opacos a la base de datos**.
- Sólo se puede acceder a través del ID.

REDUNDANCIA Y CONSISTENCIA DE DATOS

- Hashing consistente.
- Consistencia ajustable (Quórum N, R, W).
- Reloj vectorial.



Bases de datos de Documentos

¿EN QUE CONSISTE?

- La **estructura interna es conocida**, permite consultas complejas.
- **Schema-less**.
- Estructura basada en **Documentos**.

REDUNDANCIA Y CONSISTENCIA DE DATOS

- Atomicidad a nivel de documento.
- Redundancia lógica (desnormalización)
- Redundancia física mediante conjuntos de réplicas.
- Consistencia eventual en lecturas.



Documentos vs Clave-Valor

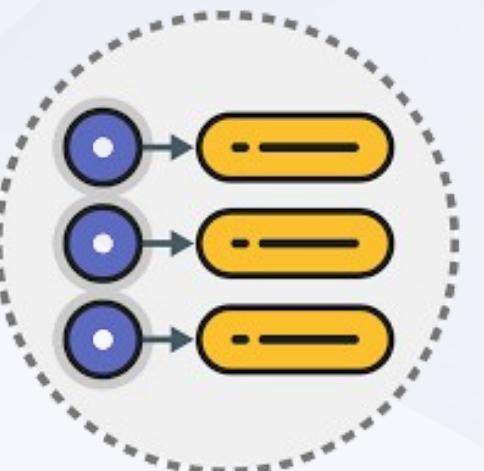
¿CUANDO UTILIZAR BD ORIENTADA A DOCUMENTOS?

- **Datos complejos**, variados o cambian con frecuencia
- Consultas por **campos internos**, búsquedas por contenido, filtrado, consultas **complejas**
- **Ejemplo** : Plataformas con publicaciones y catalogos de productos con atributos muy variados



¿CUANDO UTILIZAR BD CLAVE VALOR?

- **Datos simples**, homogeneos y accedidos solo por una clave
- Se requiere **rendimiento alto** y baja latencia
- Solo consultas **simples**
- **Ejemplo**: Estados en tiempo real y manejo de sesiones de usuario



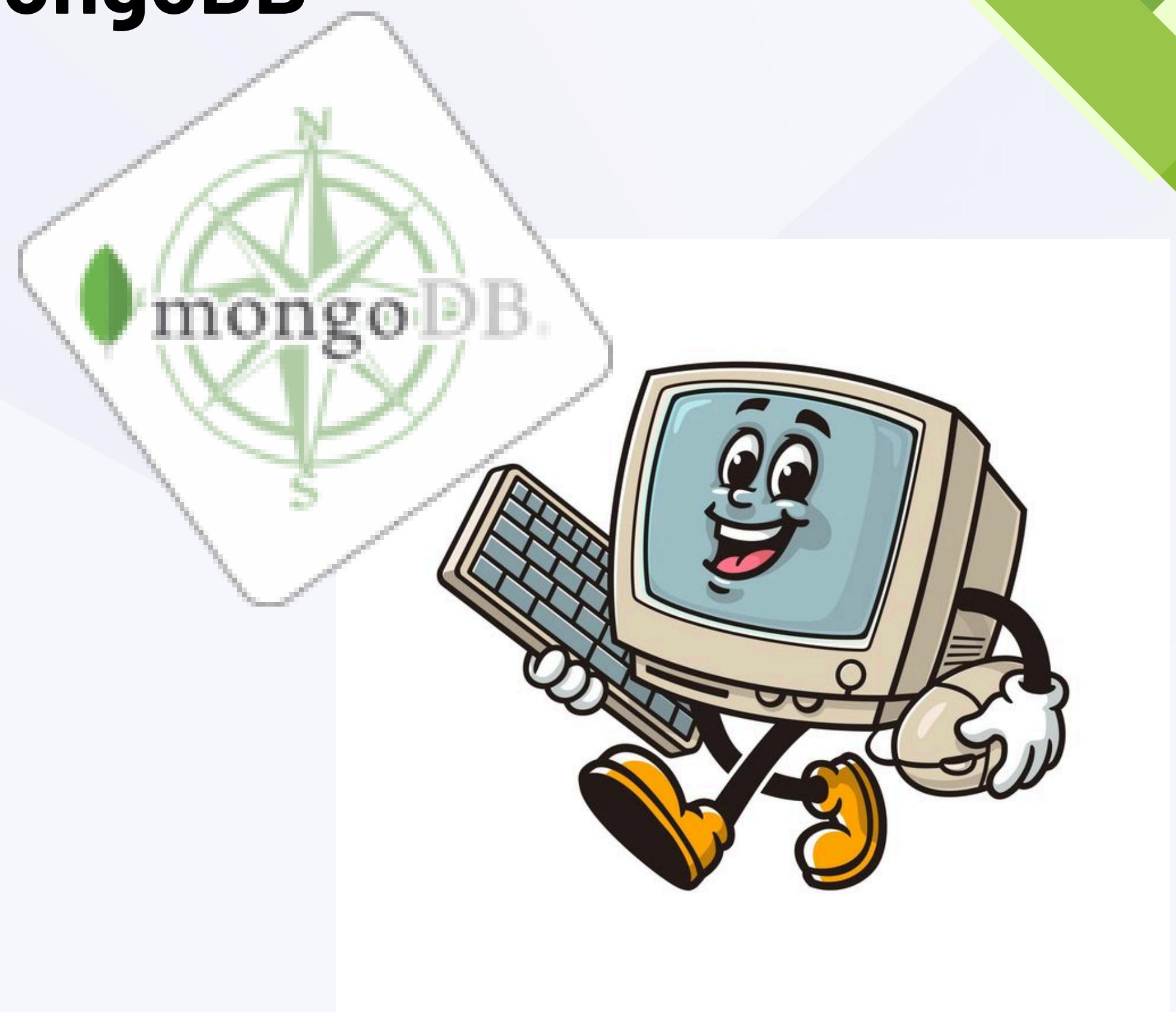
Implementación en MongoDB

OBJETIVO:

Demostrar la flexibilidad del esquema (Schema-less) frente a productos con atributos heterogéneos.

ESTRATEGIA DE DISEÑO:

- Polimorfismo: Almacenamiento de documentos con estructuras distintas en la misma colección.
- Evolución de esquema: Adición de nuevos campos sin necesidad de ALTER TABLE
- Operaciones atómicas: Uso de UPSERT para garantizar consistencia en la carga de datos



Interfaz y estructura JSON (Compass)

```
// 1. Selecciono la base de datos
use TiendaComputacion

// 2. Inserción de datos
db.productos.insertOne({
  sku: "LAP-001",
  nombre: "Laptop Gamer X",
  precio: 1500,
  categoria: "Computadoras",
  stock: 10,
  especificaciones: {
    ram: "16GB",
    procesador: "Intel i7",
    disco: "1TB SSD"
  }
})
```

```
// 3. Inserto datos con estructura diferente (Polimorfismo)
db.productos.insertOne({
  sku: "MOU-001",
  nombre: "Mouse Óptico",
  precio: 20,
  stock: 50,
  categoria: "Accesorios",
  dpi: 12000,
  inalambrico: true
})

// 4. Actualización de datos (Update)
// Actualizar precio
db.productos.updateOne(
  { sku: "LAP-001" },
  { $set: { precio: 1400 } }
)
```

```
// Agrego campo nuevo (Evolución de esquema)
db.productos.updateOne(
  { sku: "MOU-001" },
  { $set: { marca: "Logitech" } }
)
```

```
// 5. Upsert (Insertar si no existe)
db.productos.updateOne(
  { sku: "TEC-003" },
  {
    $set: {
      nombre: "Teclado Mecánico",
      precio: 100,
      categoria: "Periféricos"
    }
  },
  { upsert: true }
)

// 6. Eliminación de datos (Delete)
db.productos.deleteMany({ precio: { $lt: 50 } })

// 7. Consulta final para verificar estado
db.productos.find()
```

```
_id: ObjectId('691f6610bcf27b7fde42cddd')
sku : "LAP-001"
nombre : "Laptop Gamer X"
precio : 1400
categoria : "Computadoras"
stock : 10
- especificaciones : Object
  ram : "16GB"
  procesador : "Intel i7"
  disco : "1TB SSD"

_id: ObjectId('691f668ba73daf4d9895b5ec')
sku : "TEC-003"
categoria : "Periféricos"
nombre : "Teclado Mecánico"
precio : 100
```

Relacional vs Mongo DB

RIGIDEZ FLEXIBILIDAD

CONTEXTO VS VELOCIDAD

SQL VS MQL

JOINS VS AUTOCONTENIDO

UNICIDAD VS REDUNDANCIA

ESTRUCTURA VS FUNCIONALIDAD

DISTINTAS NECESIDADES



UNIVERSIDAD NACIONAL
de MAR DEL PLATA

¡Muchas gracias!

¿Consultas?