Ficha Técnica — Módulo 3: Red Lógica

Propósito del componente

La Red de Frames constituye el **modelo lógico del sistema experto**, donde se formaliza cómo EduDB evalúa las Formas Normales mediante reglas, slots, daemons y criterios.
Mientras que la Red Semántica define los conceptos y relaciones del dominio, este módulo establece **cómo se comporta el sistema**, qué reglas se ejecutan y qué información necesita para determinar si un esquema cumple 1FN, 2FN o 3FN.

Su objetivo es traducir el modelo conceptual en una estructura lógica que permita al sistema:

- Evaluar esquemas de bases de datos,
- Activar reglas automáticas mediante demonios,
- Razonar sobre dependencias funcionales,
- Inferir cumplimiento o violaciones de FN,
- Soportar una posible evaluación difusa (cumplimiento parcial / grado de severidad).

Es el corazón del sistema experto: define **qué reglas se aplican, cuándo se activan y cómo se razonan los criterios de normalización**.

Se plantearon 2 casos de usos principales para guiar el modelado:
CU1 — Sistema evalúa una forma normal deseada (1FN, 2FN, 3FN) e informa su cumplimiento
CU2 — Usuario pide información sobre una forma normal

Entradas

Las entradas que alimentan este módulo son:

- **Esquema a evaluar**, con sus atributos y dependencias funcionales.
- **Forma Normal solicitada** (1FN, 2FN o 3FN).
- **Propiedades del esquema**, evaluadas previamente o calculadas:
    - atributos multivaluados
    - existencia de clave compuesta
    - dependencias parciales
    - dependencias transitivas

Estas entradas activan los **daemons** y reglas dentro de los frames.

Salidas

El módulo produce los resultados lógicos derivados de las reglas definidas:

- Determinación CUMPLE / NO CUMPLE para la forma normal solicitada.
- Motivo del incumplimiento (multivaluados, parciales, transitivas, PK no compuesta).
- Activación explícita de criterios (Criterio1FN, Criterio2FN, Criterio3FN).
- Clasificación automática de dependencias funcionales.
- En modo difuso (planteado teóricamente):
    - Grado de cumplimiento de la Forma Normal (Bajo / Medio / Alto)
    - Prioridad de corrección (Baja / Media / Alta)

Herramientas utilizadas y entorno

El modelo de Frames está construido conceptualmente en Miro y se fundamenta en teoría de sistemas expertos:

- Frames (estructura)
- Slots (propiedades)
- Daemons (disparadores)
- Reglas difusas conceptuales (no implementadas aún en Neo4j)

Arquitectura o funcionamiento interno

La arquitectura está formada por 3 elementos principales:

A. Frames

Representan las entidades lógicas del dominio:

- **EVALUAR_FORMA_NORMAL**
- **1FN**, **2FN**, **3FN**
- **CRITERIO_1FN**, **CRITERIO_2FN**, **CRITERIO_3FN**
- **ESQUEMA**, **ATRIBUTO**, **DEPENDENCIA_FUNCIONAL**

Cada frame contiene **slots** y **demonios** que definen su comportamiento.

B. Slots

Son propiedades de cada frame, por ejemplo:

- forma_normal
- sin_atributos_multivaluados
- pk_compuesta
- sin_dependencias_parciales
- sin_dependencias_transitivas
- dependencias_parciales[]
- dependencias_transitivas[]

Estos valores se completan en la instancia del esquema evaluado.

C. Daemons

Los daemons actúan como *gatillos* que activan la inferencia:

- **if-added(forma_normal)**
  Cuando se selecciona 1FN/2FN/3FN se inicia el análisis correspondiente.
- **if-needed(Criterio)**
  Evalúan automáticamente los criterios asociados a cada FN.
- **if-added(tipo)**
  Clasifica dependencias funcionales como plenarias, parciales o transitivas.

D. Reglas de evaluación por Forma Normal

1FN

El esquema cumple si:

- No tiene atributos multivaluados.

2FN

Cumple si:

- Cumple 1FN
- Tiene clave compuesta
- No existen dependencias parciales

3FN
Cumple si:

- Cumple 2FN
- No existen dependencias transitivas

E. Variables difusas (conceptuales)
Aunque no implementadas el modelo lógico incluye:

- Cantidad de atributos multivaluados (Baja/Media/Alta)
- Cantidad de dependencias parciales
- Cantidad de dependencias transitivas
- Complejidad del esquema
- Grado de cumplimiento FN
- Prioridad de corrección

Estas sirven como ampliación futura del sistema.

Ver anexo: Captura del modelo lógico

Código relevante

Este módulo es conceptual; su implementación concreta se refleja en el metamodelo y reglas de Neo4j.
Código relacionado:
https://github.com/PauRodriguezz/EduDB-ProyectoIA/blob/main/Neo4j/setup.cypher

Ejemplo de instancia dentro del modelo conceptual

Ejemplo: Evaluar 2FN en el esquema "Pedido"
Atributos:

- IDProducto (PK)
- IDPedido (PK)
- NombreProducto
- NroPedido
- Cantidad

Dependencias:

- (IDProducto, IDPedido) → Cantidad (Plena)
- IDProducto → NombreProducto (Parcial)
- IDPedido → NroPedido (Parcial)

Procesamiento en frames:

- Frame 1FN
  - sin_atributos_multivaluados = true → ✓
- Frame 2FN
  - pk_compuesta = true
  - sin_dependencias_parciales = false → ❌
- Resultado lógico: **NO CUMPLE 2FN**
- Daemon activa *NO_CUMPLE → 2FN*

Resultados obtenidos

- Las reglas lógicas funcionan correctamente para múltiples esquemas.

- 1FN, 2FN y 3FN se determinan de forma consistente.
- Los daemons clasifican adecuadamente las dependencias.
- Permite construir relaciones CUMPLE/NO_CUMPLE en Neo4j sin inconsistencias.
- La instancia de ejemplo “Pedido” funciona como caso base y coincide con la teoría.

Observaciones y sugerencias

- Sería deseable implementar la capa difusa en Neo4j o en un motor externo para evaluar esquemas ambiguos o incompletos.
- El uso de demonios podría ampliarse para detectar automáticamente claves candidatas.

Anexo

📌 Captura del modelo lógico