Para un sistema como el **Nexo Sinérgico**, donde combinamos datos relacionales rígidos (usuarios, proyectos, costes) con datos flexibles de IA (prompts, parámetros variables, feedback subjetivo), la mejor opción es **PostgreSQL**.

¿Por qué? Porque PostgreSQL maneja relaciones estrictas (SQL) y tiene un soporte para **JSONB** (NoSQL) de clase mundial. Esto te permite tener lo mejor de los dos mundos en una sola base de datos.

Aquí tienes el diseño del esquema optimizado para el **Aprendizaje Continuo (Continuous Learning)**.

---

1. Diagrama Entidad-Relación (Conceptual)

La estructura central se basa en rastrear la **Genealogía de la Generación**. Si una imagen es "hija" de otra (una regeneración corregida), debemos saberlo para ver si la corrección funcionó.

---

2. El Esquema SQL (PostgreSQL)

Copia y pega esto para crear tu estructura. He utilizado tipos de datos `JSONB` para los campos que cambiarán con el tiempo (para que no tengas que migrar la base de datos cada vez que cambies el prompt).

SQL

```
-- Habilitar extensión para UUIDs (identificadores únicos universales)
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- 1. TABLA DE PROYECTOS (Ya deberías tener algo similar)
CREATE TABLE projects (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(255),
    owner_id UUID, -- Link a tu tabla de usuarios actual
    created_at TIMESTAMP DEFAULT NOW()
);

-- 2. TABLA MAESTRA DE GENERACIONES (El Historial)
-- Guarda cada intento de la IA, sea bueno o malo.
CREATE TABLE ai_generations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    project_id UUID REFERENCES projects(id),
```

```
    -- Genealogía: Si esto es una corrección, ¿de quién viene?
    parent_generation_id UUID REFERENCES ai_generations(id),
    iteration_index INT DEFAULT 1, -- 1 = Original, 2 = Corrección 1, etc.

    -- Contexto de Entrada (Los datos duros)
    -- Ej: { "demand_kw": 385, "cost_eur": 184000, "efficiency": 0.06 }
    telemetry_snapshot JSONB NOT NULL,

    -- Configuración del Motor (Cómo se configuró el "VisualMetaphorEngine")
    -- Ej: { "audience": "INVESTOR", "style_preset": "CYBERPUNK_DARK" }
    engine_config JSONB NOT NULL,

    -- El Prompt Real (Lo que se envió a la API)
    full_prompt_text TEXT NOT NULL,
    negative_prompt_text TEXT,

    -- El Resultado (Lo que devolvió la API)
    output_url TEXT, -- URL de la imagen en S3/Cloudinary
    output_metadata JSONB, -- { "seed": 12345, "model": "dall-e-3" }

    created_at TIMESTAMP DEFAULT NOW()
);

-- 3. TABLA DE FEEDBACK (El Aprendizaje)
-- Solo se llena cuando el usuario interactúa (Vota o Regenera)
CREATE TABLE ai_feedback (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    generation_id UUID REFERENCES ai_generations(id),

    -- El Voto
    sentiment VARCHAR(20) CHECK (sentiment IN ('POSITIVE', 'NEGATIVE', 'NEUTRAL')),

    -- Etiquetas Estructuradas (Para analítica fácil)
    -- Ej: ["TOO_DARK", "TEXT_GLITCH", "WRONG_COLOR"]
    tags TEXT[],

    -- Comentarios libres del usuario
    user_comment TEXT,

    -- Si hubo refinamiento manual, qué pidió el usuario
    manual_instruction TEXT,

    created_at TIMESTAMP DEFAULT NOW()
);
```

```
-- ÍNDICES PARA BÚSQUEDAS RÁPIDAS (Optimización)
-- Permite buscar dentro de los JSONs rápidamente
CREATE INDEX idx_telemetry_cost ON ai_generations USING gin ((telemetry_snapshot ->
'cost_eur'));
CREATE INDEX idx_engine_style ON ai_generations USING gin ((engine_config ->
'style_preset'));
CREATE INDEX idx_feedback_tags ON ai_feedback USING gin (tags);
```

---

3. Cómo usar esto para el "Análisis a Largo Plazo"

La magia de este esquema reside en las consultas que puedes hacer después de un mes de uso. Aquí es donde el sistema se vuelve inteligente.

Caso A: Detectar "Puntos Ciegos" del Modelo

Pregunta: "¿Qué configuración de estilo está fallando más?"

```
SQL
SELECT
    gen.engine_config ->> 'style_preset' as estilo_usado,
    COUNT(*) as total_intentos,
    SUM(CASE WHEN fb.sentiment = 'NEGATIVE' THEN 1 ELSE 0 END) as total_quejas,
    (SUM(CASE WHEN fb.sentiment = 'NEGATIVE' THEN 1 ELSE 0 END)::float / COUNT(*)) *
100 as tasa_fallo
FROM ai_generations gen
LEFT JOIN ai_feedback fb ON gen.id = fb.generation_id
GROUP BY estilo_usado
ORDER BY tasa_fallo DESC;
```

Resultado posible: Descubres que el estilo `CYBERPUNK_DARK` tiene una tasa de rechazo del 60% por ser "demasiado oscuro". *Acción:* Ajustas tu clase `VisualMetaphorEngine` en Python para añadir más luces al prompt de ese estilo.

Caso B: Encontrar Prompts "De Oro" (Golden Assets)

Pregunta: "¿Cuáles son las imágenes con mejores valoraciones para mostrarlas en mi web de marketing?"

```
SQL
SELECT
    gen.output_url,
```

```
    gen.full_prompt_text,
    gen.telemetry_snapshot ->> 'efficiency' as eficiencia
FROM ai_generations gen
JOIN ai_feedback fb ON gen.id = fb.generation_id
WHERE fb.sentiment = 'POSITIVE'
AND (gen.telemetry_snapshot ->> 'efficiency')::float < 0.08 -- Solo casos de alta eficiencia
ORDER BY fb.created_at DESC
LIMIT 10;
```

---

4. Implementación en NoSQL (MongoDB)

Si prefieres usar **MongoDB** (por ejemplo, si usas MERN Stack), el esquema sería una sola colección de documentos embebidos. Es más rápido de escribir, pero un poco más difícil de analizar estadísticamente después.

Colección: `generation_logs`

JSON

```
{
  "_id": "ObjectId(...)",
  "project_id": "xf992",
  "timestamp": "2023-10-27T10:00:00Z",
  "input_context": {
    "kw": 385,
    "cost": 184000
  },
  "prompt_data": {
    "text": "Subject: turbine...",
    "style": "INVESTOR",
    "model": "dall-e-3"
  },
  "output": {
    "url": "https://...",
    "seed": 49230
  },
  "feedback": {  // Este campo se actualiza o añade después
    "rating": "negative",
    "tags": ["TOO_DARK"],
    "refined_in_generation_id": "ObjectId(child_id...)" // Link a la corrección
  }
}
```

Recomendación de Nexo Sinérgico

Dado que tu aplicación **Pyrolysis Hub** ya maneja datos estructurados financieros y de ingeniería (Costes, Caudales), **te recomiendo encarecidamente la opción SQL (PostgreSQL)**.

Te permitirá cruzar datos duros (ej. "Las plantas con más de 500kW de consumo") con datos blandos (ej. "tienden a rechazar imágenes minimalistas"). Ese *insight* es oro puro para refinar tu producto.