



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

DEPARTAMENTO
DE INFORMÁTICA



MAGÍSTER EN
TECNOLOGÍAS DE
LA INFORMACIÓN

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE
INFORMÁTICA VALPARAÍSO - CHILE



**“Neuro RAG SQL: Framework Neuro Simbólico
para Traducción de Lenguaje Natural a SQL en
Bases de Datos Relacionales poco
Documentadas”**

MAURICIO GUSTAVO BELTRÁN BELTRÁN

TESINA PARA OPTAR AL GRADO
DE MAGÍSTER EN TECNOLOGÍAS DE
LA INFORMACIÓN

Profesor Guía: Werner Creixell Fuentes
Director de tesina: Werner Creixell Fuentes



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título ☒ Tesis de Postgrado

Título del trabajo: "Neuro RAG SQL: Framework Neuro Simbólico Para Traducción De Lenguaje Natural A SQL En Bases De Datos Relacionales Poco Documentadas"

Nombre del candidato(a): Mauricio Gustavo Beltrán Beltrán

Carrera / Grado: Magister en Tecnologías de la Información

Campus: San Joaquín

Departamento: Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Werner Creixell Fuentes, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO** contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 09-09-2026

Firma:

Estudiante o Candidato(a):

Fecha: 09-09-2026

Firma:



Neuro RAG SQL: Framework Neuro Simbólico Para Traducción De Lenguaje Natural A SQL En Bases De Datos Relacionales Poco Documentadas

Mauricio Beltrán Beltrán
Instituto de Desarrollo Agropecuario
Agustinas 1465, Santiago, Región Metropolitana

mbeltran@usm.cl

Resumen: Las bases de datos relacionales heredadas y poco documentadas dificultan la traducción de lenguaje natural a SQL, especialmente en entornos del sector público caracterizados por nomenclaturas crípticas, ausencia de claves foráneas físicas y alta dependencia de especialistas. Esta tesina presenta **Neuro RAG SQL**, un *framework* neuro-simbólico que optimiza la generación de consultas SQL mediante la combinación de recuperación semántica, el modelamiento del esquema como un grafo de conocimiento y algoritmos topológicos deterministas para guiar la inferencia de modelos de lenguaje de gran escala. Validado en un entorno operativo real en el Instituto de Desarrollo Agropecuario (INDAP), el *framework* demostró un incremento en la tasa de ejecución exitosa global del 63.0% al 80.0% sobre un corpus de 100 consultas de prueba, elevando la ejecutabilidad del 45.5% al 81.8% en el subconjunto de consultas complejas con uniones ($n = 11$), junto a una reducción de la tasa de error sintáctico y de compilación T-SQL inicial del 54.0% al 21.0% ($n = 100$). En el ámbito cualitativo y exploratorio, un piloto con analistas institucionales caracterizó la viabilidad del autoservicio asistido, registrando un 80.55% (29 de 36) de consultas adoptadas sin requerir edición manual. Este trabajo demuestra la viabilidad de gobernar modelos generativos mediante restricciones simbólicas estructuradas, aportando trazabilidad, robustez y eficiencia en escenarios corporativos hostiles.

Palabras clave: *Text to SQL*, IA neuro-simbólica, grafos de conocimiento, bases de datos relacionales, recuperación semántica.

1 Introducción

El acceso oportuno a los datos relacionales representa un activo crítico para la modernización del Estado. Sin embargo, los sistemas de información del sector público acumulan una severa complejidad estructural. Esta obsolescencia tecnológica debilita la toma de decisiones y sobrecarga a los departamentos técnicos. El desarrollo de interfaces de traducción automática facilita la autonomía de los equipos analíticos frente a estos almacenes de datos.

1.1 Contexto, Motivación y Problemática General

La modernización del Estado requiere una gestión basada en evidencia y datos de calidad. En el contexto de la administración pública chilena, el Instituto de Desarrollo Agropecuario (INDAP) cumple un rol estratégico. Este servicio público administra presupuestos sectoriales para apoyar activamente a más de 160.000 pequeños productores agrícolas. El correcto diseño de sus programas de fomento demanda la consulta permanente de sistemas de información transaccionales históricos.

Los sistemas de información transaccionales de INDAP operan sobre motores de bases de datos de gran escala. No obstante, la extracción de información analítica requiere el dominio del lenguaje estructurado (SQL) y dominio de negocio. Esta barrera técnica genera una alta dependencia de ingenieros de datos y analistas especializados. Dicha centralización operativa ralentiza el flujo de reportes críticos para las autoridades institucionales.

La dependencia técnica se ve agravada por el estado de la documentación del catálogo de datos. Las bases de datos *legacy* de INDAP carecen de diccionarios técnicos y esquemas actualizados. Esta desconexión impide

que los nuevos profesionales comprendan de manera oportuna la estructura del negocio. La automatización del proceso de consulta surge como una alternativa científica para mitigar esta asimetría de información.

1.2 Definición Técnica del Problema

El paradigma de traducción de lenguaje natural a SQL (*Text to SQL*) busca resolver esta brecha de usabilidad [1]. Los avances recientes en modelos de lenguaje de gran escala (LLM) han acelerado el desarrollo de estas herramientas de traducción [1], [2], [3]. Sin embargo, la efectividad de los LLM decrece significativamente en entornos industriales ruidosos [1]. Los esquemas de bases de datos reales presentan dimensiones que superan las capacidades estándar de inferencia [1], [4].

La base de datos de producción de INDAP refleja de manera fidedigna esta hostilidad estructural. El catálogo revisado está compuesto por un volumen masivo de 1.822 tablas activas. El esquema físico carece de restricciones declaradas de integridad referencial (claves foráneas). Asimismo, la nomenclatura de las tablas y columnas responde en ocasiones a convenciones crípticas e inconsistentes en español.

Estas deficiencias estructurales inducen fallos severos de generación en los asistentes generativos comerciales [1]. Al estar expuestos a contextos extensos y ambiguos, los modelos de lenguaje sufren de alucinación relacional [1], [5]. El sistema de inteligencia artificial omite tablas de tránsito e inyecta alias inválidos, produciendo errores de unión (JOIN) y fallos de compilación sintáctica en el motor relacional de destino [1], [5]. El análisis sistemático de estas deficiencias relacionales fundamenta de manera directa la urgencia y relevancia de su resolución.

1.3 Importancia De La Resolución Del Problema

La resolución de este problema técnico es prioritaria para optimizar la eficiencia operativa institucional. La automatización de consultas rutinarias disminuye la carga operativa sobre los ingenieros de datos y analistas especializados [1]. Dicha liberación de capacidad técnica permite redirigir los esfuerzos del personal hacia tareas estratégicas de mayor complejidad. Esta reasignación de funciones es indispensable para acelerar los flujos de reportería crítica y mitigar la dependencia técnica en la toma de decisiones [1].

Al descentralizar el acceso a la información, se minimiza el riesgo asociado a interpretaciones de datos erróneas. La prevención de uniones de tablas incorrectas o productos cartesianos evita interpretaciones sesgadas [5], [6]. Este incremento en la confiabilidad de los datos analizados fortalece la transparencia y la trazabilidad de la gestión pública.

La mitigación de estos riesgos operativos facilita, además, la preservación y formalización de la propiedad intelectual de la organización. Las bases de datos *legacy* albergan dependencias que solo residen en la memoria operativa de los desarrolladores y analistas veteranos. El catálogo automático de relaciones implícitas mitiga la pérdida de conocimientos críticos por rotación de personal [7]. De este modo, la estructura del negocio se vuelve auditable, accesible y documentada para las futuras generaciones de profesionales. Este entorno operativo estructurado y libre de asimetrías de información constituye el fundamento lógico para el diseño de una interfaz neuro-simbólica [8].

1.4 Propuesta de Solución: El Framework Neuro RAG SQL

Como respuesta técnica a esta problemática, se propone el diseño y la implementación de **Neuro RAG SQL**. Esta solución se fundamenta en los principios de la Inteligencia Artificial Neuro-Simbólica [8]. El sistema unifica la flexibilidad semántica de los modelos de lenguaje (Sistema 1) con la rigurosidad deductiva de las estructuras discretas (Sistema 2) [8]. La arquitectura modela formalmente el catálogo físico como un Grafo de Conocimiento orientado a esquemas denominado *Schema KG* [6], [7].

El *Schema KG* consolida la representación lógica del motor transaccional mediante dos familias de aristas. El *framework* utiliza búsqueda semántica vectorial para identificar un conjunto inicial de tablas semilla en el grafo. Posteriormente, se aplican algoritmos topológicos como la búsqueda en anchura (*Breadth-First Search*, BFS) y el cálculo de caminos mínimos de Dijkstra [7]. Estos operadores discretos aíslan de forma determinista un subesquema óptimo con redundancias y ruido textual reducido [4].

El subesquema mínimo resultante es serializado dinámicamente y se inyecta como contexto acotado en la instrucción (*prompt*) de traducción del LLM para guiar la generación del SQL final [4], [7]. Este enrutamiento simbólico previo restringe el espacio de búsqueda del generador probabilístico de código. El modelo de lenguaje actúa únicamente como un transformador sintáctico de precisión, minimizando la latencia y la tasa de errores de compilación física en el servidor relacional [4].

1.5 Objetivos del Proyecto

El objetivo general consiste en diseñar, implementar y evaluar el *framework* Neuro RAG SQL como una solución neuro-simbólica para optimizar la eficiencia y la tasa de ejecución exitosa en la generación de consultas SQL sobre bases de datos relacionales poco documentadas, reduciendo la dependencia técnica y facilitando el autoservicio asistido de los analistas en el Instituto de Desarrollo Agropecuario (INDAP).

Para lograr el objetivo general, se formulan los siguientes objetivos específicos:

1. Diseñar una representación jerárquica y estructurada del catálogo transaccional de INDAP mediante la construcción incremental del *Schema KG*.
2. Implementar un pipeline de procesamiento agéntico que combine de forma asíncrona la recuperación vectorial, la expansión topológica de grafos y la inferencia lingüística guiada.
3. Desarrollar una aplicación web que provea un entorno visual de auditoría de trazas y mecanismos interactivos de retroalimentación (*Human in the Loop*).
4. Evaluar cuantitativamente el rendimiento técnico mediante un estudio de ablación y caracterizar cualitativamente la usabilidad y adopción en un piloto operacional con analistas en INDAP.

1.6 Marco Teórico Adoptado

El marco teórico integra aportes sobre *Text to SQL* y *schema linking* [5], [6], grafos de conocimiento e interoperabilidad semántica [9], [10], RAG aplicado a bases de datos [4], [7] e IA neuro-simbólica [8].

1.7 Hipótesis

Hipótesis: La integración del *Schema KG* y algoritmos de optimización topológica en una arquitectura neuro-simbólica incrementa la tasa de ejecución exitosa y reduce la tasa de error sintáctico en la generación de consultas SQL sobre bases de datos relacionales poco documentadas, en comparación con un método de recuperación semántica vectorial puro.

Hipótesis nula: La integración del *Schema KG* y algoritmos de optimización topológica no produce mejoras medibles en la tasa de ejecución exitosa ni en la reducción de errores sintácticos respecto de la línea base de recuperación semántica vectorial pura.

Para contrastar empíricamente esta hipótesis, se implementa un diseño experimental aplicado en el entorno operacional de INDAP. El método consiste en un estudio de ablación técnica que contrasta el rendimiento del *framework* propuesto (Modo B) frente a la línea base vectorial (Modo A) sobre un *corpus* de control estandarizado de 100 consultas de negocio en SQL Server. De manera complementaria, se instrumenta una fase piloto cualitativa y descriptiva con analistas expertos de la institución para caracterizar la viabilidad del flujo de autoservicio asistido, el grado de intervención humana y la percepción de usabilidad de la interfaz en producción. El reporte y análisis sistemático de esta validación se distribuye de forma estructurada a lo largo del documento.

1.8 Variables de Investigación

La metodología experimental de validación opera controlando e instrumentando las siguientes variables:

1. **Variable Independiente (V_{ind}):** El uso del *framework* neuro-simbólico basado en *Schema KG*, utilizando un LLM generativo constante para la generación de consultas SQL en un entorno real de operación institucional.
2. **Variables Dependientes (V_{dep}):**

- **Eficiencia Técnica:** Tasa de Ejecución Exitosa (EX) y Tasa de Error Sintáctico (ER).
 - **Fricción del Proceso:** Frecuencia de intentos o reintentos necesarios antes de obtener una *query* ejecutable y válida.
 - **Autonomía y Calidad:** Grado de intervención humana (ediciones requeridas), utilidad percibida y flujo de autoservicio asistido.
3. **Variables de Control:** En ambos modos del estudio de ablación se utilizó el modelo fundacional *Google Gemini 3.1 Flash Lite*. La inferencia se configuró de manera estandarizada con una temperatura fija de $T = 0.2$, $Top-p = 0.8$ y un límite de 1.024 tokens de generación.

1.9 Estructura del Informe

Este informe se organiza formalmente en cinco capítulos secuenciales. El Capítulo 2 establece las bases teóricas de la traducción *Text to SQL*, el paradigma de recuperación aumentada y los fundamentos de la Web Semántica. El Capítulo 3 detalla la arquitectura modular, el flujo de estados agenticos, la construcción del *Schema KG* y los algoritmos topológicos de Neuro RAG SQL. El Capítulo 4 sistematiza las pruebas técnicas de ablación y los indicadores de impacto registrados en el piloto operacional de INDAP. Finalmente, el Capítulo 5 presenta las conclusiones generales, limitaciones tecnológicas y proyecciones futuras de la investigación.

2 Marco Teórico y Estado del Arte

Este capítulo describe los fundamentos teóricos y el estado del arte de la traducción relacional y el procesamiento de lenguaje natural. Se definen las bases de la traducción *Text to SQL*, el paradigma de recuperación aumentada y el modelamiento semántico mediante grafos de conocimiento. Asimismo, se analizan las limitaciones de la industria frente a entornos relacionales reales y se delimita el alcance del proyecto.

2.1 Fundamentos de la Traducción *Text to SQL*

La traducción de lenguaje natural a SQL, denominada *Text to SQL*, se define formalmente en el campo de la Inteligencia Artificial y el Procesamiento del Lenguaje Natural (*Natural Language Processing*, NLP) como la obtención de una función de mapeo probabilística f capaz de traducir una consulta formulada en lenguaje natural $Q = \{q_1, q_2, \dots, q_m\}$ a una sentencia equivalente y ejecutable en lenguaje de consulta estructurado $S = \{s_1, s_2, \dots, s_n\}$ [4], [6]. Esta traducción se realiza con respecto a un esquema de base de datos relacional subyacente $\mathcal{E} = \{T, C, FK\}$, donde T es el conjunto de tablas, C el de columnas y FK el de restricciones de clave foránea (*Foreign Keys*), tal como se formaliza en los marcos de evaluación de dominio cruzado modernos [11]. La modelación matemática de esta tarea busca maximizar la probabilidad condicional de la secuencia estructurada S dado el par conformado por la pregunta del usuario Q y la especificación del esquema $\mathcal{E}$ [6], [11]:

$$\hat{S} = \arg \max_S P(S | Q, \mathcal{E})$$

El desarrollo tecnológico para aproximar este cálculo de probabilidad ha experimentado una evolución histórica en la literatura.

Los primeros sistemas de traducción *Text to SQL* empleaban modelos secuencia a secuencia (Seq2Seq) clásicos [2], [11]. Estas arquitecturas utilizaban redes neuronales recurrentes con mecanismos de atención básicos [2]. Los modelos trataban la tarea como una traducción de idiomas tradicional, mapeando directamente tokens lingüísticos a sintaxis SQL. Este enfoque inicial presentaba fallos ante esquemas dinámicos no vistos durante el entrenamiento, debido a su incapacidad para codificar de forma aislada la estructura del esquema relacional [6]. La superación de estas limitaciones secuenciales se logró mediante la introducción de mecanismos de atención estructurados [2].

Los modelos basados en lenguaje preentrenados (PLM) introdujeron codificadores adaptativos del esquema [2]. Arquitecturas como RAT-SQL incorporaron el concepto de autoatención ponderada por relaciones (*relation-aware self-attention*) [6]. Este método mapea explícitamente las interconexiones entre tablas y columnas directamente dentro del espacio de autoatención del codificador. De este modo, el sistema interpreta

la topología física de la base de datos antes de generar el código ejecutable [2],[6]. Esta codificación profunda de relaciones ha sido absorbida y escalada en el paradigma tecnológico dominante actual.

Los modelos de lenguaje de gran escala (LLM) han consolidado el aprendizaje en contexto (*In-Context Learning*, ICL) como estándar de traducción [3], [5]. Los LLM modernos no requieren un reentrenamiento de pesos relacionales para operar sobre esquemas dinámicos [2]. En su lugar, el sistema depende de la ingeniería de *prompts* de alta complejidad que manipula la representación sintáctica del esquema [3]. La efectividad de esta inferencia contextual masiva requiere de métodos eficientes para restringir la información inyectada al modelo [3], [4].

2.2 Schema Linking y Desambiguación Semántica

El *schema linking* constituye una de las tareas y cuellos de botella técnicos principales de cualquier sistema *Text to SQL* [6]. Este proceso se define formalmente como la alineación y emparejamiento de los términos o entidades presentes en la pregunta lingüística con las tablas o columnas del catálogo de la base de datos [6]. En entornos relacionales reales, el proceso se ve dificultado por la presencia de nombres de atributos inconsistentes y abreviaciones crípticas [4]. La superposición de una ontología estructurada permite mitigar estas asimetrías nominales.

La ontología de grafos actúa como un filtro restrictivo determinista sobre el espacio de búsqueda del modelo de lenguaje [7]. Un modelo expuesto a un catálogo físico completo sin filtrar se enfrenta a un espacio combinatorio inmenso de uniones posibles de tablas [4]. Esta dispersión de información induce inevitablemente errores de alucinación y la selección de atributos incorrectos pertenecientes a tablas homónimas [4]. La topología del *Schema KG* bloquea la activación de tablas vecinas inactivas al evaluar la existencia de relaciones lógicas previas [7].

2.3 Generación Aumentada por Recuperación (RAG) Aplicada a Esquemas

El paradigma de Generación aumentada por recuperación (*Retrieval augmented generation*, RAG) optimiza el comportamiento de los modelos lingüísticos mediante contexto externo [12]. El sistema asocia un recuperador de información denso que extrae documentos relevantes de un espacio latente [12]. El generador formula la probabilidad de salida condicionada por los documentos recuperados [12]. La formulación general del modelo probabilístico se define de la siguiente manera:

$$p(y | x) \approx \sum_{z \in \text{top-}k(p(\cdot | X))} p_{\eta}(z | x) p_{\theta}(y | x, z)$$

Este paradigma de recuperación se adapta de forma específica al dominio de las bases de datos relacionales [4]. En el dominio de las bases de datos, el paradigma se denomina *Schema RAG* [4]. El objetivo del recuperador no consiste en extraer texto libre, sino en aislar el subesquema mínimo necesario para resolver la consulta [4]. Este subesquema aislado es serializado de forma dinámica e inyectado como contexto estructurado en el *prompt* del LLM [4]. Sin embargo, los recuperadores vectoriales clásicos fallan al procesar tablas desconectadas de forma textual. El fallo en la captura de estas conexiones físicas constituye la principal deficiencia de las implementaciones vectoriales tradicionales [4].

La recuperación vectorial pura opera mediante la similitud de coseno en el espacio latente de *embeddings* vectoriales [4]. Este método evalúa cada tabla del catálogo de manera aislada y plana [4]. La métrica evalúa el coseno del ángulo entre los vectores que representan la consulta Q y los metadatos de la tabla T_i mediante la función [4], [7]:

$$\text{Sim}(Q, T_i) = \frac{\vec{E}(Q) \cdot \vec{E}(T_i)}{|\vec{E}(Q)| \times |\vec{E}(T_i)|}$$

Al carecer de noción de conectividad topológica, el sistema omite tablas puente necesarias para realizar uniones (*JOINS*) si estas tablas no presentan coincidencia de texto con la consulta del usuario. Esto resulta inevitablemente en la alucinación del LLM y en fallos sintácticos en el código SQL generado [4]. Para superar esta ceguera relacional de los vectores, la literatura propone modelar la estructura de datos mediante grafos.

2.4 Grafos de Conocimiento del Esquema (*Schema KGs*)

Un Grafo de Conocimiento del Esquema (*Schema KG*) representa de manera formal el catálogo físico y conceptual de la base de datos [6], [10]. El catálogo se modela como un grafo dirigido, etiquetado y ponderado $G = (V, E, W)$ [10]. Los vértices V se componen de la unión de subconjuntos de tablas V_T y columnas V_C , definido matemáticamente como [6]:

$$V = V_T \cup V_C$$

El tratamiento lógico y semántico de esta estructura de datos adopta los postulados de la ingeniería ontológica [9], [10]. Las conexiones lógicas entre las entidades de la base de datos se registran mediante el conjunto de aristas dirigidas E [6], [10]. Estas aristas se definen mediante la unión disjunta de dos familias de relaciones implementadas:

$$E = E_{fk} \cup E_{sim}$$

Donde E_{fk} representa los enlaces de claves foráneas y E_{sim} define las aristas de similitud semántica [4]. Para cada par de columnas $(u, v) \in V_C \times V_C$, la afinidad semántica base $s(u, v) \in [0, 1]$ es determinada mediante la similitud de coseno entre sus representaciones vectoriales densas (*embeddings*) [7], como se expresa en la ecuación:

$$s(u, v) = \cos(e_u, e_v) = \frac{e_u \cdot e_v}{\|e_u\| \|e_v\|}$$

Donde $e_u, e_v \in R^d$ corresponden a los vectores obtenidos a partir de la concatenación del nombre del atributo, su tipo de dato SQL y una muestra representativa de los valores más frecuentes observados en producción [4], [6].

Con el propósito de limitar la densidad de aristas en el grafo y evitar la inclusión de relaciones espurias, el subconjunto E_{sim} se construye mediante la aplicación de un umbral de corte determinista $\tau_{sim} = 0.60$ [7], de acuerdo con la ecuación:

$$E_{sim} = \{ (u, v) \in V_C \times V_C \mid u \neq v \wedge s(u, v) \geq 0.60 \}$$

Toda correlación semántica que verifique la condición $s(u, v) < 0.60$ es descartada determinísticamente del grafo, omitiéndose la creación de la arista [7]. Este procedimiento de poda garantiza que solo asociaciones con respaldo vectorial suficiente sean consideradas durante las etapas de navegación topológica [7], [10].

Una vez admitidas las aristas, la función de ponderación $W: E \rightarrow R^+$ asigna un costo de navegación sobre cada enlace [10]. Dado que los algoritmos de caminos mínimos (Dijkstra) minimizan la suma acumulada de pesos en una trayectoria, $W(e)$ cuantifica una penalización de tránsito y no una afinidad directa [10], [13]. La función de costos se formaliza en la ecuación:

$$W(e) = \begin{cases} w_{fk} = 0.10, & \text{si } e \in E_{fk} \\ 1.0 - s(u, v), & \text{si } e = (u, v) \in E_{sim} \end{cases}$$

De acuerdo con la ecuación, se establecen dos criterios de navegación:

1. **Relaciones estructurales (E_{fk}):** reciben un costo fijo preferencial de 0.10, induciendo a los algoritmos de enrutamiento a priorizar las restricciones de clave foránea existentes en el catálogo [13].
2. **Relaciones de similitud (E_{sim}):** imponen un costo complementario a su afinidad ($1.0 - s(u, v)$) [10]. Dado el umbral $\tau_{sim} = 0.60$, los costos de navegación para esta familia varían estrictamente en el intervalo $[0.00, 0.40]$. Por consiguiente, una similitud alta ($s = 0.90$) genera un costo bajo (0.10), mientras que una similitud cercana al límite de corte ($s = 0.60$) impone un costo de penalización de 0.40.

La distribución jerárquica y el acoplamiento de conexiones se representan de forma visual en la Figura 1. Las líneas continuas corresponden a relaciones estructurales E_{fk} [6] (:foreign_key:), tales como el enlace entre las tablas *Explotacion* y *Rubro* mediante el mapeo de los atributos *rubro_id* e *id_rubro*. Por su parte, las líneas segmentadas representan enlaces E_{sim} derivados de similitud semántica (:similarity:). Los valores numéricos exhibidos en el diagrama (0.7, 0.8 y 0.9) constituyen instancias de afinidades calculadas $s(u, v)$ que superaron el umbral $\tau_{sim} = 0.60$. Al ser admitidos en el grafo, dichos enlaces asumen costos de navegación de 0.30, 0.20 y 0.10 respectivamente, permitiendo al algoritmo de Dijkstra identificar tablas puente únicamente cuando las claves foráneas no están presentes en el catálogo físico.

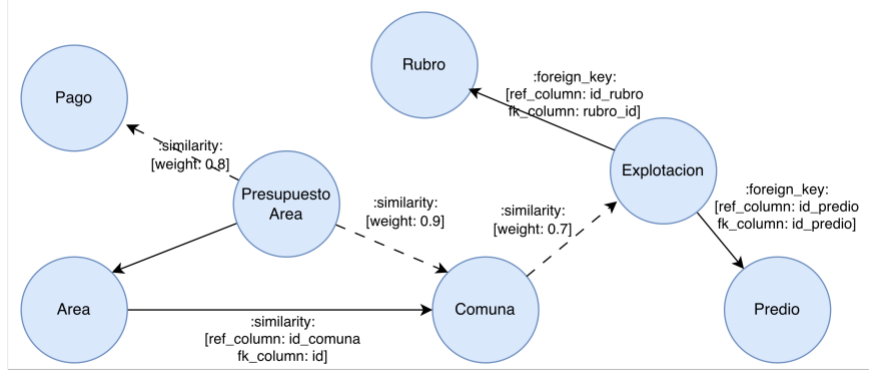


Figura 1. Representación conceptual grafo de conocimiento

Esta hibridación de restricciones duras y blandas permite mantener la conectividad topológica ante catálogos físicos fragmentados, sirviendo de base para la navegación de rutas lógicas en el estándar de la web semántica [9], [10].

La representación en grafos inspirada en el estándar RDF de la W3C adopta la Hipótesis del Mundo Abierto (*Open World Assumption*, OWA) [9], [10]. Bajo la OWA, la ausencia de una clave foránea física registrada en el catálogo de producción no implica la inexistencia de una relación lógica de negocio [9], [10]. El *Schema KG* permite inyectar y fusionar de forma no destructiva nuevas aristas de relación lógica calculadas a partir de la similitud semántica de los datos [7], [10]. La construcción práctica de este grafo se realiza integrando de forma incremental múltiples fuentes de metadatos del motor transaccional [10]. Una vez construido el grafo, se requiere de operadores matemáticos para navegar su topología.

2.5 Algoritmos Topológicos de Recorrido de Grafos

La exploración de la estructura del *Schema KG*, para aislar el contexto relevante, se realiza mediante algoritmos de la teoría de grafos clásica [4], [13]. El algoritmo de Búsqueda en Anchura (BFS) se implementa de manera dirigida a partir de los nodos identificados por similitud textual [7]. Dado un conjunto de nodos semilla $S_0 \subset V$, el BFS explora la vecindad inmediata para arrastrar tablas maestras de primer nivel definiendo un conjunto expandido V_{exp} mediante la regla:

$$V_{exp} = \{v \in V \mid \exists s \in S_0 \text{ t.q. } dist_G(s, v) \leq \delta\}$$

Donde $dist_G(s, v)$ representa la distancia de camino más corto en el grafo G , y δ define el parámetro de profundidad máxima de expansión [7].

La conexión de entidades que se encuentran disjuntas en la consulta del usuario se resuelve mediante el cálculo de distancias mínimas [13]. El algoritmo de Dijkstra computa el camino de menor costo para conectar semillas disjuntas, descubriendo de forma automática tablas puente implícitas [13]. El cálculo del camino óptimo P^* entre dos semillas minimiza de manera determinista la suma acumulada de los pesos de las aristas en el grafo:

$$P^* = \arg \min_P \sum_{e \in P} W(e)$$

Este operador matemático garantiza que el subesquema resultante sea conexo y contenga las llaves lógicas requeridas para formular las cláusulas *JOIN* en SQL [13]. La unión de los nodos expandidos por vecindad y

los caminos mínimos define el subgrafo inducido G_{sub} . La centralidad de los nodos recuperados se calcula mediante algoritmos de propagación de importancia estructural [10].

El algoritmo *PageRank* evalúa la importancia estructural relativa de cada tabla a partir de la conectividad local del subgrafo inducido G_{sub} [10]. La ponderación estructural de *PageRank* se calcula de forma iterativa formalizada por la siguiente definición:

$$PR(u) = \frac{1-d}{|V_{\text{exp}}|} + d \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Donde d representa el factor de amortiguación, B_u denota el conjunto de nodos con enlaces entrantes a u , y $L(v)$ es el grado de salida del nodo v [10]. Esta ponderación estructural se combina con el puntaje de similitud semántica de los *embeddings* [4]. Este filtrado por centralidad estructural prioriza las tablas nucleares del negocio y descarta tablas accesorias de bitácora o configuración [4]. La selección de estos algoritmos discretos se prefiere frente a enfoques neuronales contemporáneos por razones de ingeniería pragmática [4], [13].

La aplicación de algoritmos topológicos clásicos sobre el *Schema KG* ofrece ventajas críticas frente a las Redes de Grafos Neuronales (GNN) [6], [13]. Las GNN requieren de grandes conjuntos de datos de entrenamiento para converger, un recurso ausente en entornos de bases de datos *legacy* y personalizadas [13].

Adicionalmente, los algoritmos clásicos operan con una latencia baja y proveen explicabilidad de caja blanca para auditoría gubernamental [4]. El uso de estos operadores topológicos permite estructurar un flujo de control híbrido que gobierne los modelos generativos [4], [13].

2.6 Integración Neuro-Simbólica

La propuesta técnica de esta investigación se inscribe en la corriente de la Inteligencia Artificial Neuro-Simbólica [8]. Este paradigma unifica las capacidades de aprendizaje probabilístico de las redes neuronales con la rigidez deductiva de los sistemas de representación lógica [8]. El componente de procesamiento asociativo y neuronal se denomina Sistema 1 [8]. Por el contrario, el componente lógico y estructurado se define como Sistema 2 [8]. La integración sinérgica de ambos sistemas mitiga las deficiencias intrínsecas de cada enfoque aislado.

El Sistema 1 es veloz, probabilístico y asociativo [8]. Este componente mapea de forma eficiente la similitud semántica de términos lingüísticos e interpreta la semántica de la consulta mediante *embeddings* vectoriales [4]. El Sistema 2 actúa como una capa de control estructurada, determinista y auditable [8]. Este componente opera sobre la conectividad física de los catálogos lógicos para restringir las uniones algebraicas [8]. La combinación de estas operaciones define un pipeline de procesamiento de tres fases, tal como se representa conceptualmente en la Figura 2.

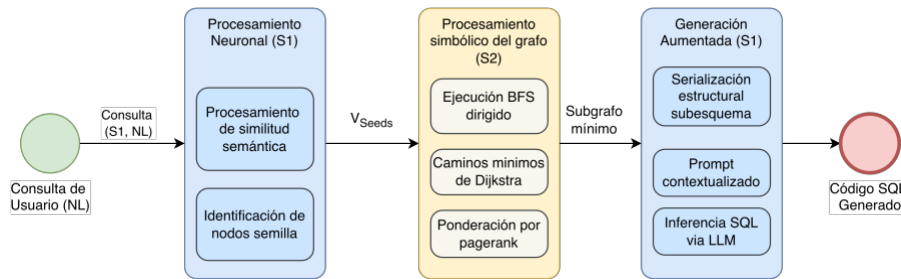


Figura 2. Integración Neuro Simbólica En La Generación *Text to SQL*

Como se detalla en la Figura 2, el flujo de datos transaccional inicia con la activación semántica de los nodos semilla mediante *embeddings* densos (Sistema 1, S1) [4]. Esta fase de procesamiento neuronal asocia los términos lingüísticos de la consulta en lenguaje natural con los metadatos físicos de la base de datos para aislar el vector de semillas (V_{Seeds}). Posteriormente, el explorador de grafos (Sistema 2, S2) ejecuta algoritmos

topológicos discretos para interconectar y filtrar las entidades, aplicando un recorrido BFS dirigido, el cálculo de caminos mínimos de Dijkstra y la ponderación por *PageRank* [13]. Este procesamiento simbólico aísla un subesquema mínimo conexo y coherente. Finalmente, el subesquema aislado es serializado de forma estructural e inyectado como una restricción lógica en el *prompt* del generador (Sistema 1, S1) [4]. De este modo, la inferencia probabilística del LLM se restringe al subgrafo recuperado, garantizando la generación sintáctica de un código SQL preciso y ejecutable. Este acoplamiento híbrido resulta fundamental para guiar de manera determinista la alineación de términos textuales con la estructura relacional de destino [4], [8].

2.7 Estado del Arte y Optimizaciones en *Text to SQL*

La optimización de los LLM para la generación relacional de código se fundamenta en técnicas de aprendizaje en contexto [3]. El estándar metodológico del estado del arte está definido por el *framework* DAIL-SQL [3]. Este sistema organiza el proceso de traducción en tres fases operacionales: la serialización compacta del esquema, la selección de ejemplos demostrativos y la secuenciación de los mismos en el *prompt* de traducción [3]. La optimización de estas instrucciones condicionales se ha estandarizado a través de arquitecturas de selección de ejemplos.

El algoritmo *Skeletal-Similarity Selector* de DAIL-SQL evalúa la similitud estructural de los esqueletos lógicos SQL de las consultas de prueba [3]. Esto asegura que el LLM reciba ejemplos de demostración (*Few-Shot*) con un nivel de complejidad de uniones homólogo al requerido por la consulta actual [3]. Asimismo, se eliminan rodeos e instrucciones de lenguaje natural redundantes para mantener la densidad del *prompt* [5]. La efectividad de estas optimizaciones se evalúa de manera tradicional en conjuntos de datos estandarizados por la academia.

La evaluación científica de sistemas *Text to SQL* utiliza *benchmarks* de dominio cruzado como Spider [11] y BIRD [1]. No obstante, la literatura reciente documenta que estos conjuntos de datos sufren de un severo sesgo de laboratorio [4]. Spider asume de forma poco realista que el esquema completo cabe dentro del contexto del LLM y utiliza esquemas esterilizados donde los nombres de columnas coinciden exactamente con la pregunta [11]. Las deficiencias de estos *benchmarks* de laboratorio se evidencian al contrastar sus métricas con sistemas productivos reales.

2.8 Práctica Industrial y Desafíos en Entornos Reales

La práctica industrial revela una severa brecha de desempeño al aplicar traductores automatizados sobre bases de datos corporativas [4]. Los *benchmarks* académicos operan sobre entornos controlados y altamente normalizados [11]. La probabilidad de que una consulta de negocio involucre más de 4 uniones de tablas supera el 25% en la industria, en contraste con el valor nulo de Spider [4]. Por el contrario, los sistemas en producción presentan una gran complejidad relacional y estructural [4]. Esta desconexión exige caracterizar las métricas físicas de las bases de datos de nivel empresarial para evaluar la viabilidad de los modelos [4].

Las bases de datos corporativas del sector público ejemplifican esta hostilidad estructural. La presencia de catálogos masivos con miles de tablas activas, sumada a una alta dispersión en la densidad de atributos por entidad, sobrecarga la ventana de inferencia de los modelos de lenguaje y dificulta la localización precisa de atributos mediante recuperadores planos [4].

La dificultad de la traducción se incrementa por deficiencias de infraestructura y nomenclaturas crípticas de negocio [1]. Por razones de rendimiento del motor histórico, las restricciones de clave foránea no están necesariamente declaradas físicamente en el catálogo relacional [4]. Los atributos y tablas pueden seguir convenciones de almacenamiento obsoletas y acrónimos complejos como `tbl_prod_run_v1` o `MVDEMOV`.

Esta asimetría nominal inhabilita los analizadores sintácticos clásicos y exige el modelado de relaciones semánticas complementarias en el grafo [4].

Frameworks industriales de integración clásicos, como Vanna.ai, resultan ineficaces ante este nivel de complejidad relacional [4]. Estas herramientas recuperan esquemas mediante similitud vectorial plana, inyectando código DDL desarticulado directamente en el *prompt* [13]. Al ignorar las conexiones lógicas de tránsito, el LLM alucina caminos de unión y genera alias inválidos [4]. Ante la escala de INDAP, estos recuperadores extraen definiciones desarticuladas, omitiendo tablas esenciales de tránsito y provocando fallos de compilación sintáctica en el motor relacional [4]. Esta vulnerabilidad sintáctica fundamenta el desarrollo de una capa de control topológica basada en la integración de conocimiento híbrido [8].

2.9 Estudios de Ablación y Taxonomía de Errores

En la investigación de sistemas de inteligencia artificial, un estudio de ablación consiste en la remoción sistemática de componentes para evaluar su contribución marginal al rendimiento [14]. Este método experimental es indispensable cuando los sistemas operan sobre bases de datos corporativas privadas y personalizadas [4]. Al carecer de conjuntos de prueba estandarizados en la literatura académica, no es posible realizar comparaciones directas contra *benchmarks* de laboratorio [4]. La validez de la hipótesis se formula analizando la utilidad marginal del *Schema KG* [13].

El experimento metodológico contrasta el rendimiento del pipeline completo frente a un *baseline* de control de recuperación de metadatos plano [13]. El *baseline* se configura mediante la desactivación de la navegación y expansión topológica en el grafo [4]. Ambos entornos se ejecutan de manera idéntica utilizando el mismo modelo generativo fundacional de lenguaje [13]. Las métricas de precisión de la ejecución se evalúan formalmente mediante un *set* de control de consultas reales de negocio [4]. Para tipificar la severidad de los fallos de traducción, se asume una taxonomía de errores estructurada.

La taxonomía categoriza los fallos de traducción en cuatro variables de error discretas e independientes para el análisis cualitativo [5]. El error de tabla clave (e_{tab}) ocurre cuando el *query* generado omite entidades indispensables o confunde nombres conceptuales [5]. El error de columna (e_{col}) se presenta ante la proyección de atributos inexistentes o pertenecientes a tablas homónimas equivocadas [5]. El error de *join* (e_{join}) denota fallas en los caminos de uniones lógicas o productos cartesianos inválidos [4]. Finalmente, el error de sintaxis (e_{sin}) agrupa fallas de compilación física en el servidor relacional [5]. El registro sistemático de estos fallos permite contrastar el rendimiento de la solución propuesta frente a tecnologías comerciales alternativas.

2.10 Trabajos Relacionados y Alcance del Framework

La propuesta de Neuro RAG SQL se diferencia de las líneas base existentes mediante la integración de la IA neuro-simbólica (Sistema 1 y Sistema 2) [8]. A diferencia de los sistemas RAG vectoriales puros que recuperan texto sin evaluar conexiones duras, la propuesta restringe de forma determinista el espacio de búsqueda del LLM utilizando el *Schema KG* [4], [13]. Esta limitación en el análisis de flujos estructurales se diferencia también de los enfoques basados en grafos desestructurados.

El *framework* GraphRAG de Microsoft emplea modelos de lenguaje para resumir comunidades de grafos abstractos construidos a partir de texto libre narrativo [15]. Neuro RAG SQL diverge críticamente de dicho enfoque al operar sobre la topología dura, tipada e intrínseca del catálogo y los metadatos de una base de datos relacional [4], [13]. Las diferencias entre los tres enfoques de recuperación de esquemas se sistematizan en la Tabla 1 para delimitar el alcance de la solución.

Tabla 1. Tabla comparativa de enfoques de recuperación aumentados.

Dimensión Técnica	RAG Vectorial Puro	Microsoft GraphRAG	Neuro RAG SQL (Propuesta)
Fuente de Datos	Documentos de texto / DDL planos.	Corpus de texto libre narrativo.	Catálogo físico y metadatos de base de datos relacional.
Construcción del Grafo	No aplica (almacenamiento plano).	Extracción de entidades y relaciones mediante LLM.	Inferencia determinista (FK, nominal, histórico).
Algoritmos de Navegación	Similitud de coseno vectorial.	Resumen de comunidades por LLM.	BFS dirigido, caminos mínimos y <i>PageRank</i> .
Propósito de Salida	Recuperación de fragmentos de texto.	Generación de resúmenes globales.	Aislamiento de subesquema óptimo para código SQL.

Frente a la postura de omitir el *schema linking* inyectando el catálogo completo en modelos con contexto de más de un millón de tokens, este enfoque presenta cuatro limitaciones críticas:

1. **Degradación atencional (*Lost in the Middle*):** En esquemas no documentados y masivos como el de INDAP, la saturación contextual dispersa la atención del LLM [15], incrementando los errores de uniones (e_{join}) y de columnas (e_{col}) [1], [4].

2. **Latencia operacional:** El tiempo de generación del primer token y la latencia global crecen sustancialmente con el volumen de entrada, frenando el flujo interactivo que requiere un analista institucional [5].
3. **Costo de cómputo:** El envío recurrente de catálogos masivos vuelve inviable el costo por tokens en nubes o satura los servidores dedicados locales. Acotar el contexto garantiza sostenibilidad presupuestaria [13].
4. **Trazabilidad y gobernanza:** A diferencia de un volcado plano de caja negra, el subgrafo resultante entrega un linaje explícito y auditable para la administración pública [8], [13].

La propuesta metodológica se circunscribe estrictamente al entorno operacional de INDAP. La solución se diseña para actuar como una capa de consulta de solo lectura, impidiendo la ejecución de sentencias de manipulación de datos o definición de esquemas que comprometan la seguridad transaccional. La validación empírica se restringe al motor de base de datos SQL Server y al conjunto de analistas expertos que participan de la fase piloto. La fundamentación de estas bases teóricas y metodológicas permite iniciar el diseño detallado de la solución de software.

3 Desarrollo de la propuesta

Este capítulo describe el diseño e implementación del *framework* Neuro RAG SQL. El objetivo del sistema es resolver la traducción de lenguaje natural a sentencias estructuradas en entornos relacionales poco documentados. Para lograr este objetivo, se propone una arquitectura modular basada en microservicios independientes.

La arquitectura modular permite el desacoplamiento de los componentes de cómputo intensivo. Los servicios se ejecutan de manera aislada dentro de contenedores de software, esta independencia facilita la escalabilidad horizontal y el mantenimiento del sistema. Las interacciones físicas entre estos servicios distribuidos se estructuran mediante una matriz de comunicaciones jerárquica.

3.1 Arquitectura de la Solución y Matriz de Interacciones

El diseño del *framework* se organiza bajo un esquema de componentes desacoplados que interactúan de forma asíncrona. La distribución lógica de estos elementos y sus fronteras tecnológicas se detallan en la Figura 3.

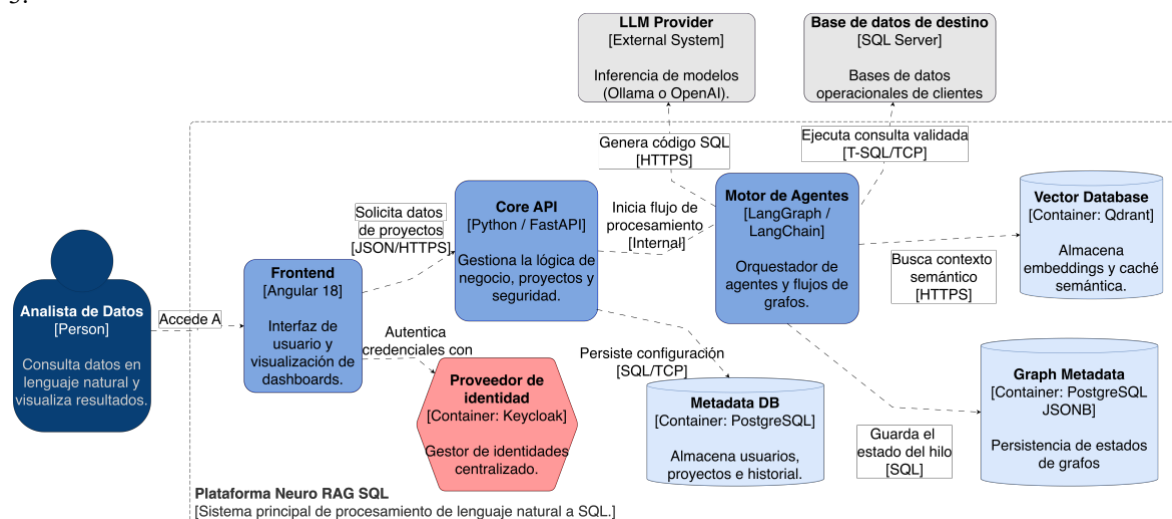


Figura 3. Arquitectura *Framework* Neuro RAG SQL

Los componentes ilustrados en este diagrama de arquitectura operan de forma coordinada para procesar la consulta del analista de datos. El acceso al sistema se realiza a través de una interfaz web interactiva que consulta a una API central. La seguridad y validación de accesos se delegan a un gestor de identidades centralizado. La

inferencia lingüística y la persistencia de estados de grafos se ejecutan directamente en el motor de agentes distribuidos con motores de bases de datos relacionales y base de datos vectoriales. Para asegurar un canal de comunicación robusto entre estas entidades, se establece una matriz de interacciones de red.

La matriz de comunicaciones detallada en la Tabla 2, define los protocolos y canales de intercambio de datos entre la interfaz de usuario, la API lógica así como los sistemas de almacenamiento persistente.

Tabla 2. Matriz de comunicaciones Neuro RAG SQL

Componente Origen	Protocolo / Canal	Componente Destino	Propósito y Tipo de Datos
Cliente Web (Angular 18)	HTTPS / JSON	API Gateway (FastAPI)	Autenticación de sesiones y envío de consultas.
Cliente Web (Angular 18)	Server-Sent Events (SSE)	API Gateway (FastAPI)	Recepción de trazas agenticas y alertas de compilación.
API Gateway (FastAPI)	HTTP / JSON	Embedder Service	Generación asíncrona de <i>embeddings</i> vectoriales.
API Gateway (FastAPI)	gRPC / TCP	Qdrant Vector Store	Búsqueda semántica de esquemas relacionales.
API Gateway (FastAPI)	SQL / TCP	PostgreSQL	Persistencia de metadatos e historial de transacciones.
API Gateway (FastAPI)	T-SQL / TCP	MSSQL DB	Ejecución de consultas de solo lectura en INDAP.
Componentes del Sistema	OpenID Connect (OIDC)	Keycloak	Validación perimetral de tokens de acceso JWT.

3.2 Orquestación del Flujo de Trabajo Agentico

El motor de flujo de trabajo se implementa con la librería especializada *LangGraph*. Esta herramienta permite modelar el proceso de traducción como una máquina de estados finitos. Los ciclos de retroalimentación integrados son necesarios para corregir errores sintácticos del motor relacional. La información procesada a lo largo de estos ciclos se transfiere mediante una estructura de datos compartida y centralizada.

La estructura de datos centralizada se denomina *AgentState*. Este objeto almacena variables de control, la consulta original, el contexto recuperado y la bitácora de errores de compilación. El estado es legible por los agentes de procesamiento pero su modificación directa está restringida. La actualización del estado se realiza exclusivamente a través de los nodos de ejecución.

3.3 Clasificación y Definición de Nodos de Ejecución

Los nodos de ejecución representan tareas u operaciones lógicas delimitadas. La secuencia lógica y el enrutamiento de estas operaciones se definen mediante una máquina de estados dirigida. Este flujo secuencial, desde el estado de inicio hasta el estado de finalización, se detalla en la Figura 4.

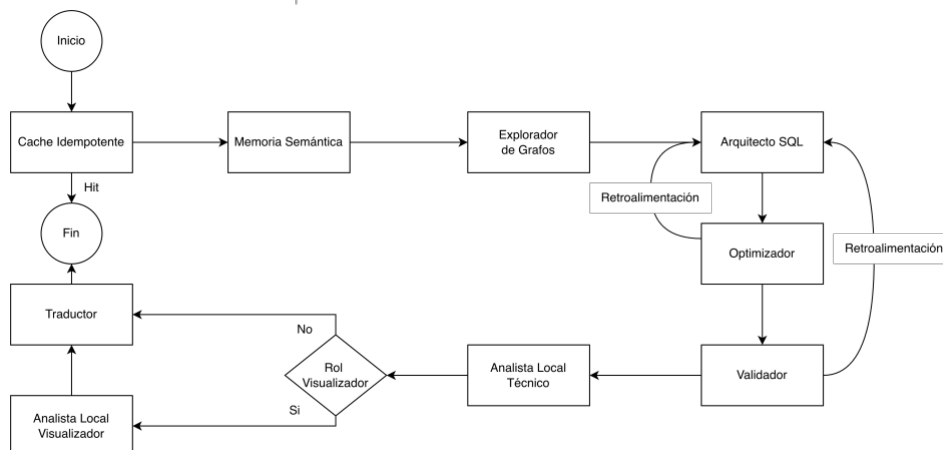


Figura 4. Diagrama de flujo de los nodos de ejecución.

El flujo operativo inicia con una evaluación determinista en el nodo de **caché idempotente**. Si se detecta una coincidencia exacta de la consulta en la caché, se produce un acierto (*hit*) que redirige la ejecución al estado de finalización. En caso contrario, se activan de manera secuencial los componentes de **memoria semántica** y el **explorador de grafos**. Estos elementos de procesamiento procedimental simbólico (Sistema 2) aíslan el subesquema de datos antes de inyectar en el nodo de razonamiento probabilístico (Sistema 1) del **arquitecto SQL**.

La generación del código se somete a ciclos de validación lógica y sintáctica. El nodo **optimizador** evalúa la consulta generada y, si detecta fallos lógicos, activa un bucle de retroalimentación interna hacia el **arquitecto SQL**. De igual manera, el nodo de **validación física** interactúa con el motor relacional y puede gatillar bucles de autorreparación ante excepciones de compilación. Las sentencias SQL que superan exitosamente estas capas de control son transferidas al nodo de **análisis local técnico**.

El análisis de los resultados y su explicación se bifurcan según los privilegios del usuario autenticado. Una compuerta de decisión evalúa el rol de visualizador en el *token* de acceso para enrutar el estado de ejecución. Los funcionarios de negocio con rol consultivo (directivos o administrativos ajenos al área TI) reciben una explicación simplificada y libre de jerga técnica generada por el nodo analista local visualizador. Por el contrario, los analistas de datos omiten este componente y avanzan de forma directa al nodo traductor para consolidar los reportes técnicos en español. La generación del subesquema inyectado en este proceso requiere una representación formal del catálogo en la capa semántica.

3.4 Construcción de la Capa Semántica y Grafo de Conocimiento

La capa semántica organiza el catálogo físico de la base de datos como un Grafo de Conocimiento del Esquema denominado *Schema KG*. El *Schema KG* modela de manera jerárquica las tablas, columnas y relaciones del sistema. Este modelo lógico permite superar la ausencia de integridad referencial física característica de los entornos heredados de INDAP. El flujo de construcción e inferencia de esta estructura lógica se presenta en la Figura 5.

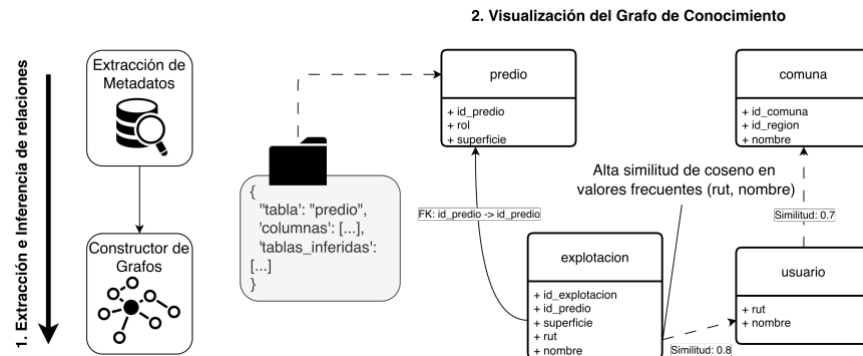


Figura 5. Proceso de extracción e inferencia de relaciones en el *Schema KG*.

El proceso inicia con la fase de extracción de metadatos del motor relacional. El componente de extracción recopila las definiciones de tablas, columnas y tipos de datos del sistema. La información recuperada se almacena en un directorio de caché local en formato JSON. Cada archivo registra atributos de tipado, descripciones y las llaves primarias detectadas en el catálogo. Una vez almacenados los metadatos aislados, el constructor de grafos inicia una fase de inferencia para interconectar los nodos de la red.

El constructor de grafos utiliza las definiciones del catálogo para instanciar los nodos y calcular las aristas estructurales. Estas aristas registran las uniones lógicas explícitas que simulan restricciones de clave foránea.

En la Figura 5, este enlace se representa mediante la arista directa entre las tablas *predio* y *explotacion* asociada a la llave común *id_predio*. Sin embargo, la integridad referencial declarada físicamente suele ser inexistente en sistemas heredados. Para resolver esta desconexión, el sistema ejecuta de manera paralela un análisis probabilístico de relaciones implícitas.

El análisis de relaciones implícitas calcula las aristas de similitud semántica mediante el uso de *embeddings* vectoriales. El sistema evalúa la distancia coseno entre los vectores de *embeddings* de los nombres de campos y los valores más frecuentes de las columnas. En la Figura 5, este procedimiento genera enlaces dirigidos con pesos ponderados, como la arista de valor 0.8 que conecta la tabla *explotacion* con *usuario*. Estas conexiones descubiertas abordan la brecha del catálogo físico agregando semántica de negocio al grafo. El grafo de conocimiento consolidado resultante actúa como el soporte topológico para la fase de recuperación de contexto.

3.5 Recuperación Contextual y Algoritmos Topológicos

La recuperación contextual busca aislar la porción mínima del esquema relacional necesaria para resolver cada pregunta del usuario. Este aislamiento es indispensable para evitar la saturación de la ventana de contexto del modelo de lenguaje. El flujo general de esta extracción se organiza en tres etapas consecutivas, tal como se detalla en la Figura 6.

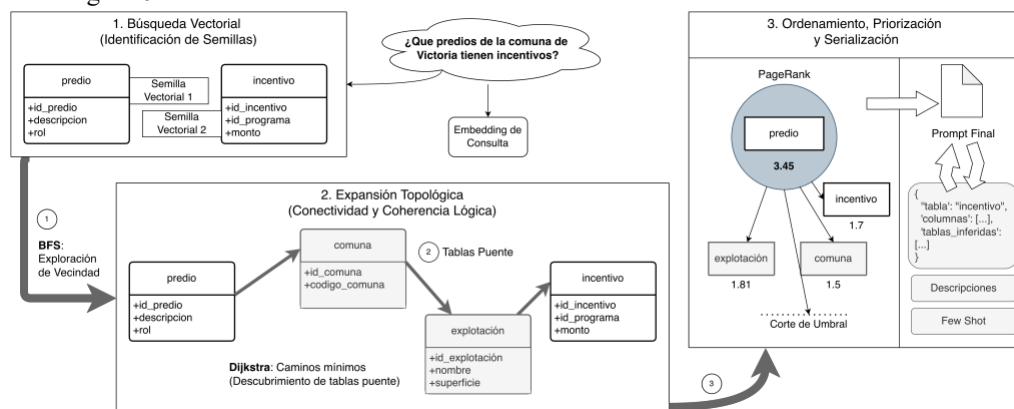


Figura 6. Camino de recuperación contextual ante una pregunta de usuario

La primera etapa de este proceso consiste en la búsqueda semántica vectorial para identificar nodos semilla. El sistema genera el *embedding* de la consulta del usuario y recupera las $k = 8$ tablas con mayor coincidencia textual. No obstante, estas entidades iniciales frecuentemente carecen de conexiones físicas declaradas en el catálogo de datos. Para restablecer su conectividad y coherencia lógica, se ejecuta una fase de expansión topológica en el *Schema KG*.

La segunda etapa realiza la expansión topológica sobre el *Schema KG* combinando algoritmos de búsqueda en anchura (BFS) y cálculo de caminos mínimos mediante Dijkstra. En primer lugar, la búsqueda BFS explora la vecindad inmediata de las semillas para capturar entidades estructuralmente conexas, parametrizada con una profundidad de retroceso de $\delta_{\text{backward}} = 2$ saltos para incorporar tablas maestras referenciadas y un avance de $\delta_{\text{forward}} = 1$ salto hacia tablas transaccionales dependientes, imponiendo un límite superior de 15 nodos para contener la dispersión combinatoria en zonas densas del catálogo. De forma complementaria, el algoritmo de Dijkstra computa la trayectoria de menor penalización acumulada para conectar semillas disjuntas, descubriendo de forma determinista tablas puente implícitas. Esta exploración se acota a una distancia máxima de $\delta_{\text{path}} \leq 8$ saltos entre pares de semillas utilizando la función de costos $W(e)$ formalizada en la Sección 2.4, la cual asigna un costo fijo preferencial de $w_{fk} = 0.10$ a los enlaces de clave foránea en E_{fk} y penalizaciones complementarias $1.0 - s(u, v) \in [0.00, 0.40]$ para las aristas de similitud semántica en E_{sim} que superaron el umbral $\tau_{\text{sim}} = 0.60$. Las entidades consolidadas por ambos procedimientos deben ser posteriormente ordenadas y filtradas para evitar ruido estructural en la ventana de contexto.

La tercera etapa ejecuta el ordenamiento y priorización de las tablas recolectadas mediante el algoritmo *PageRank*, evaluando la centralidad y relevancia estructural de cada nodo dentro de la malla activa inducida. Para este cómputo se configura un factor de amortiguación de $d = 0.85$ sobre la proyección del subgrafo en *NetworkX*, priorizando la influencia de aquellas tablas que concentran una mayor conectividad referencial y funcional de negocio. Finalmente, el subgrafo resultante se poda reteniendo los nodos con mayor centralidad relativa hasta satisfacer una capacidad máxima de contexto de $K = 50$ nodos, obteniendo un subesquema optimizado que es formateado y serializado para su inyección contextual en el prompt final.

El *prompt* final contiene la serialización del subesquema acotado, descripciones de columnas y ejemplos de traducción históricos. El modelo de lenguaje recibe estas instrucciones lógicas estructuradas como restricciones operativas inviolables. El enrutamiento simbólico previo mitiga la propensión a la alucinación del modelo y garantiza la correcta generación de uniones (JOIN). El control y supervisión de este proceso de inferencia está disponible para los usuarios en la interfaz de la aplicación web.

3.6 Interfaz de Usuario y Ciclo de Retroalimentación

La aplicación web de interfaz de usuario, organiza la persistencia y los accesos lógicos bajo la jerarquía de proyectos independientes. Cada proyecto asegura el aislamiento de los datos y restringe la navegación del *Schema KG* según el rol asignado al funcionario. El control perimetral y los permisos de acceso se validan mediante la integración nativa con gestor de identidades. Tras iniciar sesión, el usuario puede ejecutar consultas lingüísticas y auditar de forma visual las trazas de ejecución en tiempo real, tal como se detalla en la Figura 7.

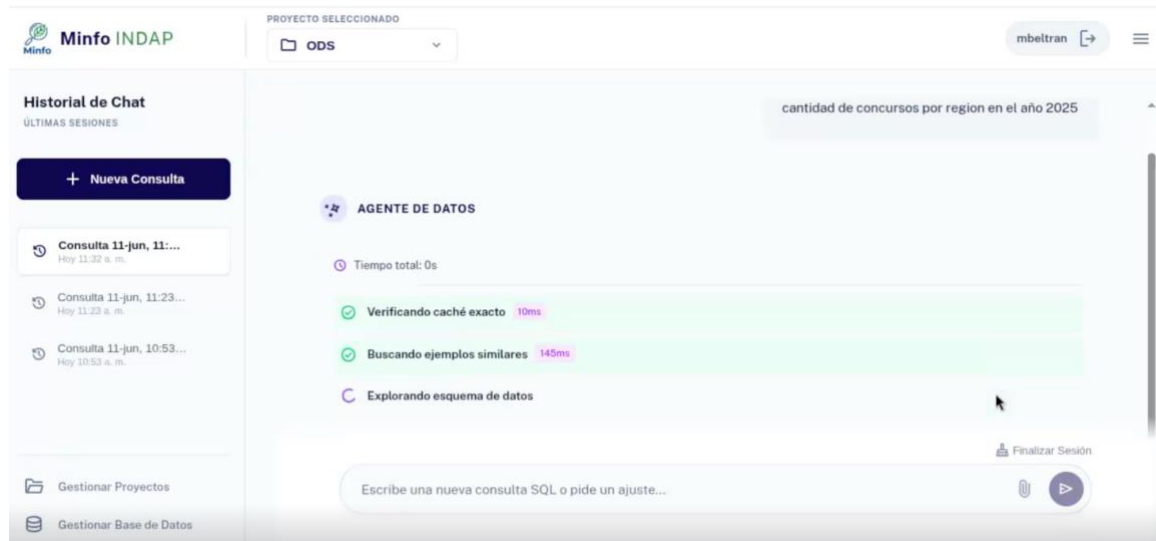


Figura 7. Trazas de ejecución del agente de datos en tiempo real.

En esta primera fase de interacción, el analista de datos ingresa la consulta lingüística en el campo de texto ubicado en la interfaz central. El sistema inicia la ejecución agentica de forma asíncrona y transmite el estado del procesamiento mediante *Server-Sent Events* (SSE). El panel central despliega secuencialmente las marcas de tiempo y el estado del pipeline, registrando latencias para la validación de caché y para la búsqueda de ejemplos semánticos dinámicos. Una vez completada la exploración del esquema, se genera el código SQL final, el cual se expone en la Figura 8.

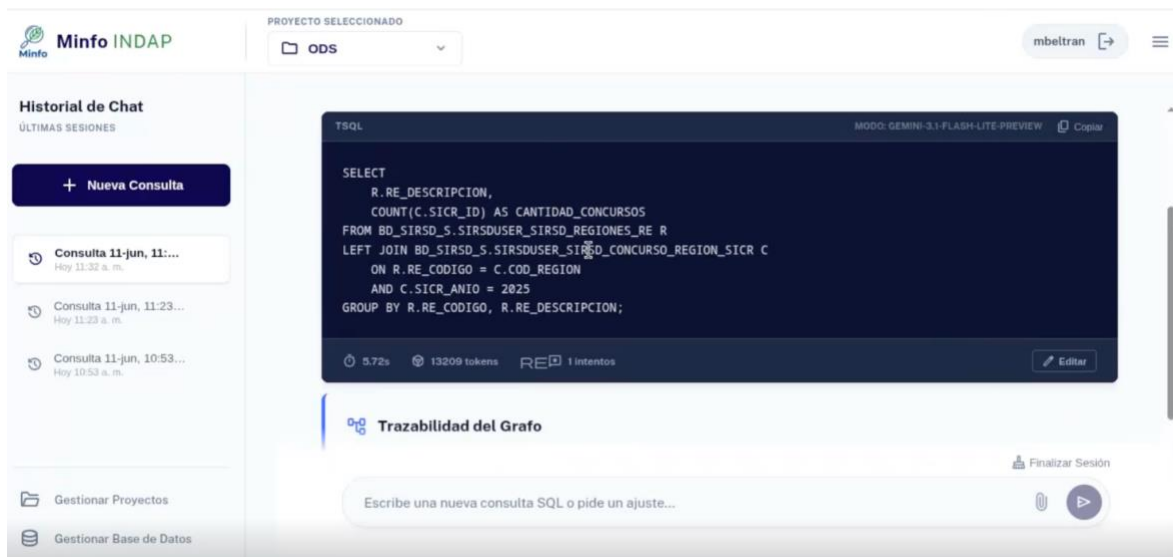


Figura 8. Sentencia SQL generada y métricas de procesamiento asociadas.

El resultado de la traducción relacional se visualiza en la consola estructurada de *Transact-SQL*. El sistema proporciona la *query* de solo lectura compatible con el motor de base de datos de destino, implementando de manera precisa la cláusula de unión (*LEFT JOIN*) entre las tablas de regiones (*SIRS_REGIONES_RE*) y de concursos territoriales (*SIRS_CONCURSO_REGION_SICR*). Debajo del editor de código, se exponen de forma transparente las métricas operativas de auditoría: un tiempo de generación, *tokens* inyectados en el *prompt* y un historial de intento de ejecución. El linaje físico de los datos que soportan esta consulta se expone en la sección de trazabilidad, ilustrado en la Figura 9.

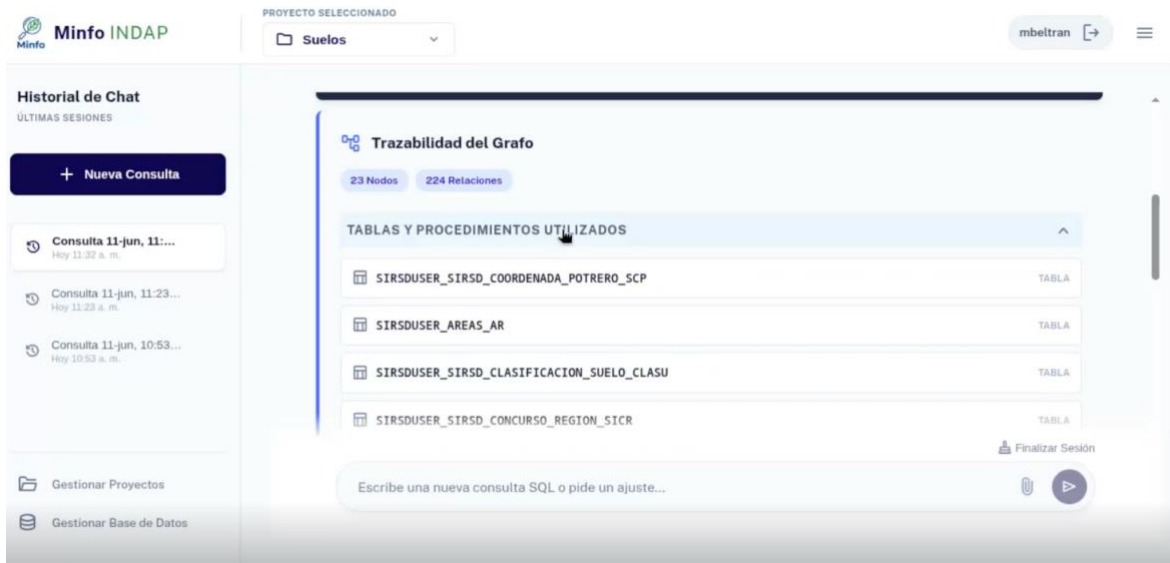


Figura 9. Panel de trazabilidad de grafo y objetos lógicos del subesquema.

La justificación procedimental se despliega en el panel de trazabilidad del grafo. El sistema expone la porción activa del catálogo de datos que fue aislada del esquema completo, registrando un subgrafo de contexto compuesto por 23 nodos y 224 relaciones lógicas activas. Bajo este resumen conceptual, se listan los metadatos de las tablas y procedimientos almacenados involucrados en la búsqueda, tales como `SIRS_COORDENADA_POTRERO_SCP`, `SIRS_AREAS_AR`, entre otras. Si se detectan inconsistencias conceptuales en el código propuesto, el usuario puede presionar el botón de editar visible en la Figura 8 para interactuar directamente con el sistema.

El mecanismo de retroalimentación (*Human in the Loop*) se activa de forma asíncrona tras la aplicación de una corrección manual por parte del analista experto en la consola de edición. El par conformado por la consulta lingüística original y la sentencia SQL modificada se envía inmediatamente a la API lógica. El servicio de *embedding* vectorial calcula el vector correspondiente a esta estructura sintáctica corregida. El nuevo vector se indexa directamente en la colección de la base de datos vectorial, sirviendo como ejemplo (*Few Shot*) de máxima prioridad para futuras inferencias de consultas de estructura análoga en el servicio de INDAP.

3.7 Diseño y Métodos de Investigación Aplicados para la Validación

La evaluación del *framework* se estructura bajo un enfoque metodológico aplicado y descriptivo. Este diseño prioriza la observación sistemática de la plataforma de software en el entorno operativo real del Instituto de Desarrollo Agropecuario (INDAP). La recolección de métricas empíricas en producción permite constatar de manera objetiva el desempeño práctico de la solución. Este comportamiento operacional es condicionado de forma directa por un conjunto de factores y variables de estudio.

3.7.1 Definición de las Variables de Investigación

El comportamiento del sistema y su impacto en la organización se evalúan mediante la interacción de tres conjuntos de variables metodológicas. Estas variables operan sobre el catálogo transaccional del sistema.

La **variable independiente** (V_{ind}) se define como el modo de operación del *framework* de traducción en el entorno de producción. Esta variable es de naturaleza discreta y binaria, adoptando dos estados excluyentes durante el estudio de ablación:

- **Modo A (Línea Base):** Operación por recuperación semántica vectorial tradicional y generación directa vía modelo de lenguaje, inhabilitando la capa del grafo de conocimiento.
- **Modo B (Propuesto):** Operación neuro-simbólica completa, activando la recuperación semántica, el modelamiento en el Grafo de Conocimiento del Esquema (*Schema KG*) y la optimización topológica.

Las **variables dependientes** (V_{dep}) registran los efectos cuantitativos y operativos derivados de la manipulación de la variable independiente. Se clasifican en cuatro dimensiones métricas principales:

- **Tasa de Ejecución Exitosa (EX):** Cuantifica la proporción de consultas SQL generadas que compilan y se ejecutan satisfactoriamente en el motor relacional de destino sin arrojar excepciones y retornando un conjunto de datos válido.
- **Tasa de Error Sintáctico T-SQL y de Compilación (T-SQL Error Rate, ER):** Proporción de consultas generadas que arrojan excepciones de sintaxis, fallos gramaticales de dialecto o errores de resolución de entidades en el motor relacional durante el primer intento de inferencia, previo a los ciclos agénticos de autorreparación.
- **Fricción Operativa:** Registra el número de intentos o iteraciones en la interfaz que el usuario debe realizar antes de consolidar una consulta ejecutable.
- **Eficiencia Temporal y de Recursos:** Evalúa el costo computacional de cada consulta a través de dos dimensiones:
 - **Latencia (t_{total}):** Desglosada en recuperación de contexto (t_{ret}), inferencia del LLM (t_{gen}) y ejecución SQL (t_{exec}).
 - **Consumo de tokens:** Medición de tokens de entrada (*prompt*) y salida (*completion*). Ambas métricas se contrastan entre el Modo A y el Modo B para analizar el compromiso costo-efectividad de la capa de grafos.

Las **variables de Control** se implementan sobre parámetros estabilizados para mitigar factores exógenos y garantizar la reproducibilidad experimental:

- **Corpus Estandarizado de Prueba:** Conjunto fijo de 100 consultas de negocio que cubren de manera representativa los niveles de complejidad y los patrones analíticos de la institución para el estudio de ablación.
- **Infraestructura de Cómputo y Motor de Base de Datos:** Instancia local dedicada con asignación fija de memoria y CPU para el motor SQL Server, suprimiendo variaciones causadas por congestión de red o concurrencia externa.
- **Perfil de Usuarios en Piloto Operacional:** Convocatoria restringida a analistas institucionales con conocimiento técnico homogéneo sobre el modelo de negocio para la caracterización cualitativa de usabilidad en producción.

Para contrastar la significancia estadística de las diferencias de rendimiento entre la línea base y la propuesta sobre el mismo conjunto de prueba ($N = 100$), se utiliza la prueba de McNemar con corrección por continuidad de Edwards. Este test no paramétrico es el estándar formal para diseños experimentales pareados con variables de respuesta dicotómica (éxito o fallo de ejecución), evaluando la asimetría en la distribución de los pares discordantes frente a la hipótesis nula de equivalencia.

3.7.2 Configuración del Estudio de Ablación

El estudio de ablación se diseñó para contrastar de manera directa el rendimiento del Modo A frente al Modo B sobre el *corpus* de control. Este procedimiento técnico permite aislar la ganancia algorítmica provista por la capa simbólica determinista. Las trazas de ejecución y los errores generados en ambos modos se almacenan en un componente de logs del servidor lógico para su posterior auditoría.

Para el análisis cualitativo de las fallas detectadas en la fase de ablación, se estructuró una taxonomía formal de errores basada en variables discretas. Cada consulta fallida se clasifica de forma mutuamente excluyente dentro de las siguientes categorías:

- **Error de Unión (e_{join}):** El sistema alucina uniones lógicas o caminos de tránsito inválidos entre las tablas [2].
- **Error de Columna (e_{col}):** Selección de atributos inexistentes o prefijos asimétricos incorrectos dentro de la sentencia [4].
- **Error de Tabla (e_{tab}):** Omisión de entidades críticas o invocación de catálogos ajenos a la consulta [4].

- **Error Sintáctico (e_{sin}):** Fallos de escritura nativa del dialecto T-SQL que impiden la compilación en el motor.

3.7.3 Instrumentación del Piloto

La validación e impacto organizacional se realiza mediante una fase de prueba cerrada con los analistas institucionales seleccionados. Los expertos interactúan con la interfaz funcional del *framework* procesando requerimientos reales vinculados al apoyo de los más de 160.000 pequeños productores agrícolas. El sistema registra de forma transparente la tasa de edición del código y los mecanismos de asistencia e interacción *Human in the Loop* aplicados.

La evaluación de usabilidad, percepción de autonomía y satisfacción del usuario se instrumenta de forma nativa a través de la propia interfaz web del sistema. La plataforma captura la conformidad del usuario mediante un módulo integrado de retroalimentación que registra valoraciones (positivas o negativas) y comentarios cualitativos opcionales al momento de valorar negativamente la consulta. Estos eventos de telemetría de interfaz se procesan de forma asíncrona junto con la bitácora de logs y las métricas operativas del servidor, permitiendo validar la hipótesis de investigación bajo una perspectiva integral de ingeniería, utilidad y usabilidad práctica basada en datos de interacción reales, continuos y no intrusivos.

4 Resultados y Evaluación de Desempeño

En este capítulo se presenta el análisis empírico derivado del monitoreo operacional en producción y de las pruebas experimentales del *framework* Neuro RAG SQL en el Instituto de Desarrollo Agropecuario (INDAP). La evaluación cuantifica la Tasa de Ejecución Exitosa (EX), la capacidad de autorreparación agéntica y el compromiso de costos en latencia y tokens, integrando los resultados de ablación validados estadísticamente sobre el corpus de control y analizando la viabilidad operativa del autoservicio asistido en la gestión analítica institucional.

4.1 Entorno de Operación y Monitoreo de Uso Real en Producción

El despliegue y monitoreo del *framework* se ejecutó directamente sobre la infraestructura y bases de datos transaccionales de producción en INDAP. Este catálogo corporativo se caracteriza por ser un entorno relacional indocumentado, con ausencia de restricciones físicas de integridad referencial (claves foráneas) y convenciones de nomenclatura asimétricas acumuladas históricamente.

Para evaluar el comportamiento del sistema en condiciones reales de uso operacional, se instrumentó un mecanismo de telemetría asíncrona durante un piloto cerrado con un panel de 4 analistas de datos. Esta medición en producción auditó un total de 36 interacciones en tiempo real de los usuarios a través de la interfaz web, midiendo las latencias (mediana y media) del pipeline agéntico, la tasa de intervención humana (*Human in the Loop*), la efectividad del bucle de autorreparación ante errores sintácticos del motor T-SQL y la reducción en los tiempos de atención de solicitudes analíticas.

4.2 Resultados del Piloto Operacional: Eficiencia Técnica y Autorreparación Agéntica

El análisis de telemetría sobre la muestra de 36 transacciones reales en el entorno productivo de INDAP registró una mediana global de respuesta de 16.45 s por consulta para el pipeline completo del Modo B (con una media aritmética global de 58.59 s, condicionada por la asimetría ante consultas complejas y latencias de red externa):

1. **Efectividad al primer intento:** El 33.3% de las consultas (12 de 36) se ejecutó correctamente en el servidor relacional en su primera iteración, sin requerir ajustes lógicos o sintácticos por parte del nodo optimizador.
2. **Capacidad de autorreparación agéntica:** De las consultas que presentaron fallas sintácticas en el primer intento, el 83.33% fue corregido de forma autónoma en el segundo ciclo de validación mediante la inyección de la traza de error del motor relacional en el orquestador de *LangGraph*.

3. **Fallos no resueltos y latencia por reintentos:** Un 11.1% de las solicitudes en producción (4 de 36) no logró una compilación exitosa tras agotar el límite de tres ciclos. El tiempo de procesamiento exhibió una marcada asimetría condicionada por los ciclos de autorreparación: en el primer intento, las consultas registraron una mediana de 15.66 s (media de 29.87 s, con un mínimo de 5.66 s y un máximo de 80.28 s); al requerirse un segundo ciclo de validación, el tiempo se incrementó a una mediana de 18.22 s (media de 54.88 s); mientras que los casos marginales que demandaron un tercer ciclo alcanzaron una media de 101.50 s (mediana de 83.41 s). Esta divergencia confirma que, si bien la experiencia operativa modal se sitúa bajo los 20 segundos, la acumulación secuencial de trazas de error y llamadas consecutivas a la API en consultas multirrelacionales complejas introduce casos atípicos que elevan el promedio aritmético.
4. **Consumo de ventana de contexto:** El tamaño medio de los *prompts* inyectados al modelo de lenguaje se registró en 12.404 tokens (con un rango oscilante entre 2.952 y 31.339 tokens). El volumen del *prompt* se incrementó progresivamente en cada iteración del agente, pasando de un promedio de 7.738 tokens en la primera revisión a 19.596 tokens en la tercera, debido a la acumulación recursiva de subgrafos relacionales y trazas de error de compilación. Por su parte, las respuestas generadas requirieron una media de 223 tokens de salida.
5. **Comportamiento de la caché semántica:** Durante las 36 transacciones auditadas en esta fase inicial no se registraron aciertos en la caché semántica (0 hits), comportamiento atribuido a la alta variabilidad en los parámetros de entrada formulados por los analistas durante el periodo de pilotaje.

4.3 Caracterización Descriptiva y Usabilidad del Piloto Operacional

Para evaluar la interacción de los usuarios en condiciones reales de trabajo, se instrumentó un piloto operacional cerrado con 4 analistas sénior de INDAP, auditando un total de 36 transacciones mediante telemetría en producción. Dada la acotada escala muestral de esta fase, los hallazgos reportados en esta sección se formulan con carácter descriptivo y exploratorio, complementando la validación cuantitativa formal sustentada en el estudio de ablación.

1. **Tiempos de Atención y Autoservicio Asistido:** En el esquema de trabajo histórico, la resolución de requerimientos analíticos complejos a través de la cola centralizada de tickets de TI registraba demoras de hasta 13 días hábiles en casos críticos, según los registros de gestión de soporte institucional. Durante el piloto, la modalidad de autoservicio asistido permitió a los analistas formular y validar consultas de forma directa. Los analistas participantes estimaron que este proceso tomó menos de 30 minutos por requerimiento analítico. Esta reducción temporal representa una percepción cualitativa levantada en entrevistas técnicas, condicionada por el contexto exploratorio de la muestra.
2. **Intervención del Usuario (*Human in the Loop*):** Los analistas intervinieron para editar el código SQL generado en el 19.44% de los casos observados (7 de 36 consultas auditadas). En 29 de las 36 transacciones (80.55%), la sentencia provista por el pipeline fue adoptada sin requerir modificación alguna. Las intervenciones manuales se aplicaron principalmente sobre consultas que ya eran ejecutables en el motor, actuando como un ajuste de reglas de negocio específicas y no como corrección de fallas sintácticas T-SQL.
3. **Percepción de Usabilidad y Trazabilidad:** El módulo integrado de retroalimentación registró 6 valoraciones voluntarias por parte de los usuarios, resultando el 100% de ellas favorables (6 positivas y 0 negativas). En las entrevistas cualitativas, los analistas destacaron la visualización del panel de trazabilidad del subgrafo como un componente que aporta explicabilidad al origen de los datos y permite auditar las tablas puente empleadas en la resolución de las consultas.

4.4 Estudio de Ablación y Evaluación del Corpus de Control

Para la cuantificación estandarizada y reproducible del rendimiento técnico previa al análisis de hipótesis, se construyó un corpus de control de 100 consultas en lenguaje natural, clasificadas en 3 niveles de complejidad relacional (Simple, Medio y Complejo).

Sobre este conjunto de pruebas se contrastó el rendimiento de 2 configuraciones de arquitectura operando sobre la misma base de datos de producción y modelo de lenguaje fundacional:

1. **Modo A (Línea Base):** Operación por recuperación semántica vectorial pura (recuperación plana de metadatos DDL) e inferencia directa por el LLM, inhabilitando la capa del Grafo de Conocimiento del Esquema (*Schema KG*).
2. **Modo B (Propuesto):** Operación neuro-simbólica completa, integrando la recuperación vectorial, la expansión topológica sobre el *Schema KG* (algoritmos BFS, Dijkstra y *PageRank*) y la inferencia guiada por restricciones lógicas.

La comparación exploratoria de las métricas de ejecutabilidad técnica con y sin el componente simbólico (*Schema KG*) sobre el *dataset* de control de 100 consultas arrojó el comportamiento de rendimiento sistematizado en la Tabla 3.

Tabla 3. Resultados consolidados de la prueba de ablación.

Métrica Técnica	RAG Vectorial Puro (Ablación - Modo A)	Neuro RAG SQL (Propuesta - Modo B)	Diferencial
Tasa de Ejecución Exitosa (EX) - Global	63.0% (63 / 100)	80.0% (80 / 100)	+17.0 pp
EX en Dificultad Fácil (Easy)	72.7% (24/33)	84.8% (28/33)	+12.1 pp
EX en Dificultad Media (Medium)	59.1% (26/44)	79.5% (35/44)	+20.4 pp
EX en Dificultad Compleja (Hard)	56.5% (13/23)	73.9% (17/23)	+17.4 pp
EX en Consultas de Agregación	62.2% (23/37)	70.3% (26/37)	+8.1 pp
EX en Consultas de Conteo (Count)	73.7% (14/19)	89.5% (17/19)	+15.8 pp
EX en Consultas de Filtrado (Filter)	60.9% (14/23)	82.6% (19/23)	+21.7 pp
EX en Consultas de Unión (Join)	45.5% (5/11)	81.8% (9/11)	+36.3 pp
EX en Consultas de Ranking	70.0% (7/10)	90.0% (9/10)	+20.0 pp
Tasa de Error T-SQL Inicial (ER, pre-reparación)	54.0% (54 / 100)	21.0% (21 / 100)	-33.0 pp
Fallos Residuales No Resueltos (Post-autorreparación)	37.0% (37 / 100)	20.0% (20 / 100)	-17.0 pp
Errores de Unión o Conexión (join_error)	30	16	-14 fallos
Otros Errores (Sintaxis / Atributos / Motor)	7	4	-3 fallos

El indicador *T-SQL Error Rate* (ER) se midió en la inferencia inicial previa a la autorreparación, registrando 54 fallos en el Modo A (54.0%) frente a 21 en el Modo B (21.0%), lo que evidencia una reducción inicial de 33.0 puntos porcentuales atribuible al *Schema KG*. En la fase posterior de autorreparación agéntica, el Modo A subsanó 17 consultas y consolidó 37 fallos residuales irre recuperables (tasa de ejecución exitosa final del 63.0%), distribuidos en 30 errores de unión y 7 de sintaxis o atributos. Por su parte, el Modo B redujo los fallos residuales a 20 consultas (tasa de ejecución exitosa final del 80.0%), compuesta por 16 errores de unión y 4 de sintaxis.

La mayor diferencia se observó en las consultas con uniones ($n = 11$), donde el aporte conjunto de la capa topológica sobre el *Schema KG* elevó el éxito del 45.5% (5/11) al 81.8% (9/11). Para contrastar la significancia estadística sobre las 100 consultas pareadas (59 aciertos compartidos, 16 fallos compartidos, 21 aciertos exclusivos de Modo B y 4 de Modo A), la prueba de McNemar con corrección por continuidad de Edwards arrojó un estadístico chi-cuadrado de 10.24 ($gl = 1$, $p = 0.00137$, test binomial exacto $p = 0.00091$). El *Odds Ratio* resultante (OR) de 5.25 constata que por cada consulta degradada el sistema resolvió 5.25 consultas adicionales, confirmando una superioridad estadísticamente significativa ($p < 0.01$).

Asimismo, habiéndose controlado las condiciones de inferencia con el modelo *Google Gemini 3.1 Flash Lite* bajo hiperparámetros homogéneos ($T = 0.2$, $Top - p = 0.8$), la magnitud del incremento observado (+17.0 pp en EX y -33.0 pp en ER) excede con holgura el ruido de decodificación típico documentado para dicho régimen térmico ($\pm 2\%$), confirmando que la brecha responde al aporte de la capa simbólica y no a variabilidad estocástica en la corrida de evaluación.

Para contrastar la eficiencia temporal y el consumo de recursos declarada en la metodología, la Tabla 4 sintetiza el compromiso operativo entre ambas configuraciones sobre el corpus de 100 consultas:

Tabla 4. Eficiencia temporal, consumo de tokens y ejecutabilidad

Dimensión Operativa	RAG Vectorial Puro (Ablación - Modo A)	Neuro RAG SQL (Propuesta - Modo B)	Diferencial	Impacto Relativo
Latencia Total Promedio	4.88 s	6.02 s	+1.14 s	+23.4%
Recuperación Contexto (t_{ret})	0.34 s	0.92 s	+0.58 s	+170.6%
Inferencia Generativa (t_{gen})	4.37 s	4.93 s	+0.56 s	+12.8%
Ejecución en Motor SQL (t_{exec})	0.17 s	0.17 s	+0.00 s	0.0%
Tokens Totales Consumidos	590.2 tok	713.6 tok	+123.4 tok	+20.9%
Tokens de Entrada (Prompt)	499.1 tok	607.0 tok	+107.9 tok	+21.6%
Tokens de Salida (Completion SQL)	91.1 tok	106.6 tok	+15.5 tok	+17.0%
Tasa de Ejecución Exitosa (EX)	63.0% (63 / 100)	80.0% (80 / 100)	+17.0 pp	+27.0%

El contraste evidencia un balance computacional favorable, la navegación y poda topológica en el *Schema KG* añade una sobrecarga media de 1.14 s de latencia (+23.4%) y 123.4 tokens (+20.9%). Esta penalización marginal mantiene la respuesta global en 6.02 s (plenamente dentro del rango de usabilidad interactiva) y compensa la inversión computacional al elevar la Tasa de Ejecución Exitosa en +17.0 puntos porcentuales (de 63.0% a 80.0%) y reducir los fallos residuales irrecuperables de 37 a 20 consultas.

4.5 Síntesis y Validación de Hipótesis

El contraste experimental sobre el corpus de 100 consultas de negocio pareadas en SQL Server demostró que la arquitectura neuro-simbólica incrementó la tasa de ejecución exitosa en 17.0 puntos porcentuales (del 63.0% al 80.0%) y redujo la tasa de error sintáctico inicial en 33.0 puntos porcentuales (del 54.0% al 21.0%). La significancia estadística obtenida mediante la prueba de McNemar ($\chi^2 = 10.24$, $p = 0.00137$) confirma que estas variaciones son atribuibles al framework propuesto y no a fluctuaciones muestrales.

Por consiguiente, **se rechaza la hipótesis nula (H_0) y se acepta la hipótesis de investigación (H_1)**. La evidencia valida que la incorporación de restricciones topológicas deterministas sobre el *Schema KG* resuelve eficazmente la conectividad relacional y mitiga las alucinaciones estructurales en esquemas poco documentados.

5 Conclusiones, Limitaciones y Trabajo Futuro

El presente capítulo consolida los hallazgos y contribuciones científicas derivadas del diseño, implementación y evaluación del *framework* Neuro RAG SQL. Se presenta un análisis de la validación empírica de la hipótesis de investigación a partir de los resultados experimentales. Asimismo, se discuten las restricciones de diseño identificadas en el entorno real de producción. Finalmente, se proponen las directrices técnicas para orientar los trabajos futuros de desarrollo en la disciplina.

5.1 Conclusiones Generales

La presente investigación abordó el problema de la traducción de lenguaje natural a sentencias SQL estructuradas en entornos corporativos poco documentados. El desarrollo del *framework* neuro-simbólico permitió estructurar un canal de acceso directo a los datos transaccionales de la institución. Dicha interfaz mitigó la brecha de asimetría de información que afecta a los equipos de análisis de negocio. La viabilidad operativa de esta solución se fundamentó en la validación empírica de su hipótesis de trabajo.

Los resultados obtenidos en el estudio de ablación permitieron rechazar formalmente la Hipótesis Nula (H_0). La incorporación de la capa semántica de grafos y algoritmos topológicos produjo mejoras estadísticamente observables en el desempeño. La tasa de ejecución exitosa global se incrementó de un 63% en la línea base

vectorial hasta un 80% con el pipeline propuesto, diferencia validada estadísticamente mediante la prueba de McNemar ($\chi^2 = 10.24$, $gl = 1$, $p = 0.00137$, $OR = 5.25$). Este salto demuestra que la estructuración del contexto físico es indispensable para guiar a los generadores probabilísticos.

El análisis cuantitativo demostró que la ceguera relacional de los recuperadores vectoriales clásicos es el origen de los fallos sintácticos. En el Modo A, el 81.1% de las excepciones (30 de 37) se debieron a errores de unión (*join_error*) al alucinar columnas de tránsito. Por el contrario, la inyección determinista del subesquema optimizado en el Modo B redujo la tasa de error sintáctico inicial a un 21.0% (consolidando un 20.0% de fallos residuales post-autorreparación). Esta contención de las alucinaciones estructurales valida el uso del *Schema KG* como un método viable de *schema linking* en la industria.

En el ámbito cualitativo y de uso operacional, el seguimiento en producción junto a 4 analistas expertos sobre una muestra de 36 transacciones caracterizó descriptivamente la viabilidad del flujo de trabajo. Se observó una tasa de adopción sin modificaciones manuales en el 80.55% de los casos (29 de 36 transacciones), circunscribiendo la intervención humana (*Human in the Loop*) a ajustes contextuales de reglas de negocio en el 19.44% restante. Asimismo, la totalidad de las valoraciones capturadas por el módulo de telemetría ($N = 6$) resultaron favorables, destacando la explicabilidad del subgrafo activo como mecanismo de auditoría de linaje. Estos hallazgos validan la premisa de que gobernar modelos generativos mediante restricciones topológicas discretas constituye una alternativa viable para desplegar asistentes *Text-to-SQL* en bases de datos institucionales complejas. No obstante, la implementación en un entorno operativo de producción evidenció limitaciones críticas de infraestructura.

5.2 Limitaciones del *Framework*

La principal limitación tecnológica del sistema se asocia a la dependencia del servicio de *embeddings* para la localización de los nodos semilla (Sistema 1). Si la consulta en lenguaje natural utiliza términos que carecen por completo de homología semántica con los metadatos físicos, el recuperador denso omite la activación de las semillas correctas. Al no activarse los nodos nucleares en el *Schema KG*, la navegación topológica posterior hereda este sesgo espacial de búsqueda. Dicha desviación inicial impide que el algoritmo de Dijkstra establezca caminos lógicos conexos de negocio. Esta dependencia del vector denso se ve complementada por restricciones asociadas a la latencia del pipeline.

La generación de sentencias SQL requiere transferir dinámicamente la serialización de tablas, columnas y reglas de negocio del subesquema recuperado hacia una interfaz de programación de aplicaciones externa (API de Google Gemini). Aunque el *framework* implementa mecanismos perimetrales que garantizan que los registros de datos sensibles no salgan de la infraestructura de INDAP, la exposición de los metadatos estructurales a proveedores en la nube introduce una dependencia tecnológica y condiciona el cumplimiento estricto de las directrices de gobernanza, privacidad y soberanía de la información del sector público.

El flujo de orquestación agentica cíclica (*LangGraph*) incide negativamente en la latencia de respuesta global. La ejecución del *framework* completo registró una mediana de respuesta de 16.45 s en producción (con una media de 58.59 s condicionada por los ciclos de autorreparación). Esta latencia computacional es condicionada de manera directa por la cantidad de bucles de autorreparación que deba realizar el nodo de validación física. Aunque este tiempo es sustancialmente menor que la demora de resolución manual por analistas de TI, limita la interacción directa en interfaces conversacionales de tiempo real. Esta demora operativa restringe el uso del sistema a operaciones exclusivamente asíncronas.

La caracterización cualitativa del sistema en producción se efectuó sobre una muestra exploratoria de 4 analistas y 36 transacciones. Si bien estos datos permitieron observar la usabilidad de la herramienta y la interacción del usuario, la escala muestral acotada no permite derivar inferencias estadísticas formales ni generalizaciones censales respecto al impacto organizacional global y la disminución de tiempos de tramitación a nivel corporativo.

Finalmente, el *framework* se circunscribe estrictamente a un entorno operacional de solo lectura. Por razones de seguridad institucional, la plataforma bloquea de manera perimetral cualquier sentencia que intente modificar datos o alterar el esquema físico de la base de datos. Esta demarcación de seguridad, indispensable para la viabilidad de la puesta en producción, delimita las fronteras de las futuras extensiones del *software*.

5.3 Proyecciones de Trabajo Futuro

Como proyección científica e ingenieril derivada de los hallazgos y limitaciones de este trabajo, se proponen las siguientes líneas de investigación prioritarias:

1. **Migración a Modelos de Lenguaje Locales y Especializados:** Se proyecta la migración de la capa de inferencia desde APIs comerciales externas hacia modelos de código abierto especializados, tales como *DeepSeek Coder* o *Llama3-sqlcoder*. La dependencia actual de servicios en la nube eleva los costos de procesamiento por volumen de *tokens* y condiciona la estabilidad operativa a la conectividad de la red externa. La adopción de modelos locales mediante entrenamiento adaptativo (*Fine Tuning*) mitigará esta dependencia económica. Asimismo, esta transición garantizará la soberanía tecnológica y el cumplimiento estricto de las normativas de seguridad de datos en el sector público.
2. **Actualización Incremental Automatizada del *Schema KG*:** Se propone el diseño y desarrollo de un agente colector asíncrono para automatizar la vigencia del grafo de conocimiento del esquema. En la implementación actual, la inserción y el ajuste de pesos en las aristas lógicas requiere de intervención directa por parte del administrador. El colector propuesto evaluará permanentemente el historial de consultas reales ejecutadas en los servidores de producción. Dicho componente inyectará de forma autónoma aristas dinámicas basadas en los aciertos de los analistas humanos, asimilando orgánicamente las modificaciones físicas y evoluciones lógicas del catálogo de datos.
3. **Inferencia Semántica del Linaje de Aplicación Cliente:** Se propone el desarrollo de conectores agenticos capaces de realizar análisis estático sobre el código fuente de aplicaciones externas (por ejemplo, repositorios legados en Python, ASP o PHP). El objetivo es extraer las sentencias SQL embebidas en las aplicaciones e inyectar de manera automática relaciones procedimentales en el *Schema KG*. Esta expansión ampliará la cobertura de linaje más allá de los límites del motor físico de la base de datos, resolviendo la ceguera del sistema ante lógicas de negocio acopladas en la capa de *software* cliente.
4. **Evolución hacia un Marco Integral de Gestión de Datos Activo:** La viabilidad a largo plazo de las interfaces conversacionales de lenguaje natural a SQL no depende exclusivamente del modelo lingüístico, sino de un soporte estructural robusto en la calidad de las bases de datos y la vigencia del gobierno de datos. En este sentido, se proyecta investigar metodologías de autogeneración y diseño de conjuntos o vistas de datos sintetizados y resumidos. Estas estructuras lógicas intermedias serán optimizadas específicamente para su consumo por agentes inteligentes, garantizando la máxima precisión semántica y minimizando de manera determinista la ambigüedad conceptual en consultas de alta complejidad relacional.
5. **Estudio de Ablación Factorial por Componente Algorítmico:** El presente trabajo evaluó el impacto de la capa simbólica como un bloque integrado frente a la línea base vectorial. Se propone como trabajo futuro instrumentar un estudio de ablación granular que aísle experimentalmente la contribución marginal de cada operador topológico de forma independiente (evaluando variantes del pipeline que activen exclusivamente BFS, únicamente caminos mínimos mediante Dijkstra, o solo la poda estructural con *PageRank*). Dicha desagregación permitirá cuantificar el peso específico y la sensibilidad de cada algoritmo en la contención de alucinaciones relacionales y sintácticas.

Agradecimientos. Expreso mi sincero agradecimiento a la Agencia Nacional de Investigación y Desarrollo (ANID) / Subdirección de Capital Humano / Beca Magister Funcionarios Públicos 79240035 por el financiamiento y respaldo otorgado para la realización de este programa de postgrado. Asimismo, agradezco profundamente a Gloria por su apoyo incondicional, la comprensión y el aliento constante que hicieron posible culminar con éxito este importante desafío académico y profesional.

Referencias

- [1] J. Li, «Can LLM Already Serve as a Database Interface? A BIG Bench for Large-Scale Database Grounding», en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024, pp. 43101-43122.

- [2] C. D. Bui, «A systematic survey of LLM-based text-to-SQL: methodologies, security vulnerabilities, and future challenges», *PeerJ Comput Sci*, vol. 12, p. 3773, 2026, doi: 10.7717/peerj-cs.3773.
- [3] D. Gao, «Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation». en *Proceedings of the VLDB Endowment*, vol. 17, n.º 5, 2024, pp. 1132-1145, doi: 10.14778/3641204.3641221
- [4] R. Singh, N. Boškov, L. Drabeck, A. Gudal, y M. A. Khan, «CSR-RAG: An Efficient Retrieval System for Text-to-SQL on the Enterprise Scale». [En línea]. Disponible en: <https://arxiv.org/abs/2601.06564>
- [5] M. Pourreza y D. Rafiei, «DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction», en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 11647-11654.
- [6] B. Wang, R. Shin, X. Liu, O. Polozov, y M. Richardson, «RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers», en *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7567-7578. doi: 10.18653/v1/2020.acl-main.677.
- [7] C. Ma, S. Chakrabarti, A. Khan, y B. Molnár, «Knowledge Graph-based Retrieval-Augmented Generation for Schema Matching», 2025, *arXiv*. doi: 10.48550/ARXIV.2501.08686.
- [8] A. S. Garcez y L. C. Lamb, «Neurosymbolic AI: The 3rd Wave», *Artif Intell Rev*, vol. 56, n.º 11, pp. 12387-12406, dic. 2023, doi: 10.1007/s10462-023-10448-w.
- [9] J. E. L. Gayo, *Web Semántica: Comprendiendo el cambio hacia la Web 3.0. A Coruña*. España: Editorial NetBiblo, 2012.
- [10] A. Hogan, «Knowledge Graphs». Morgan & Claypool Publishers (ACM Books, San Rafael, CA, USA, 2021. [En línea]. Disponible en: <https://kgbook.org/>
- [11] T. Yu, «Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task», en *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3911-3921. doi: 10.18653/v1/d18-1425.
- [12] P. Lewis, «Retrieval-augmented generation for knowledge-intensive NLP tasks», en *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, y H. Lin, Eds., Curran Associates, Inc, 2020, pp. 9459-9474.
- [13] A. Safdarian, M. Mohammadi, E. J. Bashirloo, M. S. Naderi, and H. Faili, «SchemaGraphSQL: Efficient Schema Linking with Pathfinding Graph Algorithms for Text-to-SQL on Large-Scale Databases,» in *Findings of the Association for Computational Linguistics: EACL 2026*, Rabat, Morocco, 2026, pp. 2585–2599, doi: 10.18653/v1/2026.findings-eacl.134.
- [14] S. Sheikholeslami, H. Ghasemirahni, A. H. Payberah, T. Wang, J. Dowling, y V. Vlassov, «Utilizing Large Language Models for Ablation Studies in Machine Learning and Deep Learning», en *Proceedings of the 5th Workshop on Machine Learning and Systems*, World Trade Center Rotterdam Netherlands: ACM, mar. 2025, pp. 230-237. doi: 10.1145/3721146.3721957.
- [15] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness y J. Larson, «From local to global: A graph RAG approach to query-focused summarization», 2025, *arXiv*. doi: 10.48550/ARXIV.2404.16130.