

# Studio e Valutazione di Large Language Models per la generazione di Query Ibride SQL-Cypher

---

**Tirocinante:** Mattia Rocchi

**Tutor didattico:** Golfarelli Matteo

**Laboratorio:** Business Intelligence Lab, Stanza 4136

**Periodo di svolgimento:** 01/09/2025 – 03/11/2025

# 1. Introduzione

Il presente tirocinio si è svolto all'interno di un contesto di ricerca, focalizzato sull'uso di modelli di linguaggio avanzati (Large Language Models, o LLM) per facilitare l'interrogazione di database NoSQL complessi.

I database a grafo, in particolare, rappresentano un'architettura dati efficace per modellare relazioni complesse, trovando applicazione in numerosi domini, incluso quello agricolo oggetto di questo studio. Tuttavia, l'accesso a questi dati richiede la conoscenza di linguaggi di query specifici come **Cypher**.

L'obiettivo di questo tirocinio è stato valutare lo "state-of-the-art" dei LLM nel tradurre in modo efficace le richieste da linguaggio naturale in query valide.

La sfida è resa più complessa dal contesto tecnologico, che spesso include non solo database a grafo puri come **Neo4j**, ma anche ambienti ibridi come **PostgreSQL** con l'estensione **AGE** (A Graph Extension) per i grafi e **TimescaleDB** per le serie temporali. Questo richiede la gestione di query che combinano **SQL** (per interrogare le serie temporali) e **Cypher** (per il grafo).

La fase di studio iniziale ha analizzato lo stato dell'arte dell'interazione tra LLM e dati strutturati. Questo progetto si concentra sull'approccio "Direct Reasoner": si valuta la capacità del LLM di agire come interfaccia "Natural Language-to-Query" fornendogli un contesto strutturato (la struttura del grafo) e una richiesta utente. L'analisi è stata condotta testando quattro modelli di punta su un set di query di benchmark appositamente definite per un sottografo agricolo.

## 2. Tecnologie

Per raggiungere il nostro obiettivo, sono state usate diverse tecnologie.

- **PostgreSQL con estensioni:** Lo stack tecnologico è basato su PostgreSQL con due estensioni principali: **Apache AGE** (A Graph Extension) per la modellazione e l'interrogazione orientata ai grafi, e **TimescaleDB** per la gestione di serie temporali.
- **Linguaggi di query:**
  - **Cypher:** Linguaggio di query per database a grafo, oggetto centrale della generazione da parte dei LLM.
  - **SQL (Structured Query Language):** Linguaggio standard per database relazionali, considerato nel contesto ibrido abilitato da AGE.
- **Large Language Models (LLM):** Sono stati valutati quattro modelli principali tramite le rispettive interfacce web: ChatGPT, Gemini, Claude e DeepSeek.
- **Strumenti di sviluppo e analisi:**
  - **Linguaggio di programmazione:** **Python** è stato il linguaggio principale utilizzato per lo sviluppo degli script di analisi e valutazione.
  - **Formati dati:** **JSON** (per la definizione delle strutture di grafo) e **YAML** (per la definizione dei prompt e delle query di benchmark).
  - **Piattaforme:** **GitHub** per il versionamento del codice e **Google Workspace** (Docs) per la documentazione della ricerca.

### 3. Studio e Approfondimenti Teorici

Una parte significativa del tirocinio è stata dedicata all'analisi dello stato dell'arte per costruire una solida base teorica prima della sperimentazione.

- **Database NoSQL e grafi:** È stata condotta un'analisi delle diverse categorie di database NoSQL (Document, Key-Value, Column-family, Graph), con un focus sui vantaggi e sui casi d'uso dei database a grafo. Sono state approfondite le caratteristiche di Neo4j e PostgreSQL/AGE e le peculiarità del linguaggio Cypher.
- **LLM per dati strutturati:** È stata studiata la letteratura scientifica per comprendere le sfide (es. ambiguità terminologica, allineamento tra linguaggio naturale e strutture dati) e gli approcci esistenti (es. Direct Reasoning, fine-tuning specializzato) nell'utilizzare i LLM per generare query valide per interrogazioni su DBMS.
- **Prompt engineering:** È stato approfondito lo studio delle tecniche per costruire prompt efficaci, come zero-shot e few-shot. Si è compreso che fornire una descrizione dettagliata della struttura del database e un template di risposta è una tecnica cruciale per contestualizzare la richiesta e standardizzare l'output.

## 4. Metodologia

La parte pratica si è concentrata sulla progettazione e l'esecuzione di un benchmark per valutare quantitativamente i modelli LLM.

### 4.1 Progettazione del benchmark

1. **Definizione progressiva della struttura del grafo:** Per valutare come il livello di dettaglio nel prompt influenzi la capacità dei LLM, sono state definite tre versioni progressive della struttura:

- **Versione 1:** Solo la struttura base del grafo (nodi e relazioni) senza dati esemplificativi
- **Versione 2:** Struttura arricchita con tutte le proprietà, presenti nei nodi
- **Versione 3:** Struttura completa che includeva anche la descrizione della tabella relazionale "measurements", una Hypertable di Timescale, con al suo interno le varie misurazioni dei Device e la componente SQL delle query ibride serve per interrogare questa tabella

Questo approccio graduale ha permesso di isolare l'impatto dell'arricchimento semantico sulla qualità delle query generate.

2. **Definizione delle query di benchmark:** È stato creato un set di 10 query in linguaggio naturale di complessità crescente. Le query spaziavano da semplici interrogazioni strutturali a richieste semantiche e inferenziali più complesse, fino a query che coinvolgevano sia il grafo che la tabella "measurements".

3. **Creazione del ground truth:** Per ogni richiesta, è stata scritta manualmente la query Cypher corretta, che funge da riferimento per la valutazione.

4. **Progettazione del prompt:** È stata definita una struttura di prompt modulare, che forniva al LLM la descrizione della struttura del grafo, istruzioni chiare su come formulare la risposta e la specifica richiesta in linguaggio naturale. L'obiettivo era guidare il modello a generare la query basandosi esclusivamente sul contesto fornito.

### 4.2 Raccolta e valutazione delle risposte

La generazione delle query è stata condotta in modo manuale per simulare fedelmente l'interazione di un utente finale con le interfacce web dei modelli. Per ciascuno dei quattro LLM, è stato fornito un prompt contenente la descrizione della

struttura del dato e le istruzioni per la risposta. Le query Cypher generate da ciascun modello sono state quindi raccolte e salvate per l'analisi successiva.

La valutazione è stata automatizzata tramite uno script Python appositamente sviluppato. I criteri di valutazione si basavano sul confronto tra il risultato della query generata dal LLM e il ground truth, considerando il numero di nodi e archi restituiti. La metrica era simmetrica: sia i dati mancanti che quelli in eccesso rispetto al ground truth contribuivano negativamente al punteggio.

I risultati sono stati classificati in tre categorie:

- **Corretto:** La query è sintatticamente corretta e il risultato coincide esattamente con il ground truth.
- **Mismatch (incompleto/inesatto):** La query è sintatticamente corretta ma il risultato è incompleto (mancano dati) o impreciso (contiene dati extra).
- **Errore (sintassi):** La query non è valida e fallisce l'esecuzione.

## 5. Risultati

### 5.1 Analisi preliminare: impatto dell'arricchimento del Contesto

Per determinare la configurazione ottimale per il benchmark finale, è stata condotta un'analisi preliminare sulle tre versioni progressive della struttura del grafo. La Tabella 1 mostra l'evoluzione delle performance per Gemini attraverso le diverse versioni:

Versione	Query Corrette	Mismatch	Errori	Note Principali
V1: Struttura base	1	7	2	Elevate hallucination e dati extra
V2: Con property	2	6	2	Miglioramento stabilità sintattica
V3: Completa (+measurements)	2	7	1	Miglioramento stabilità sintattica

Tabella 1: Evoluzione delle performance di Gemini

L'analisi comparativa rivela che l'arricchimento del contesto migliora principalmente la stabilità sintattica, ma non elimina completamente i problemi di accuratezza semantica:

- Nella **Versione 1**, la mancanza di dati esemplificativi ha portato a un elevato numero di hallucination, con il modello che tendeva a "inventare" proprietà o assumere strutture dati non presenti
- La **Versione 2** ha significativamente ridotto gli errori di interpretazione, fornendo al modello riferimenti concreti su cui basare la generazione delle query
- La **Versione 3** ha dimostrato il miglioramento nella gestione della tabella "measurements", sebbene con risultati variabili tra i diversi modelli testati

Questo pattern è stato osservato in tutti i modelli testati, sebbene con valori diversi, confermando che la completezza del contesto è un fattore critico per la generazione affidabile di query.

5.2 Benchmark finale

Sulla base dei risultati dell'analisi preliminare, il benchmark finale è stato condotto utilizzando la **Versione 3**, con la nostra metrica di valutazione.

Modello LLM	Corretto	Mismatch	Errore
ChatGPT	0	4	6
Claude	2	2	6
Gemini	2	7	1
DeepSeek	3	4	3

Tabella 2: Performance dei modelli LLM sul benchmark finale

I risultati rivelano che DeepSeek ottiene il numero più alto di query "Corrette" (3 su 10), sebbene con errori distribuiti. Gemini mantiene la migliore robustezza sintattica (solo 1 errore) ma presenta il maggior numero di mismatch (7). ChatGPT mostra le maggiori difficoltà, senza query completamente corrette e con 6 errori di sintassi.

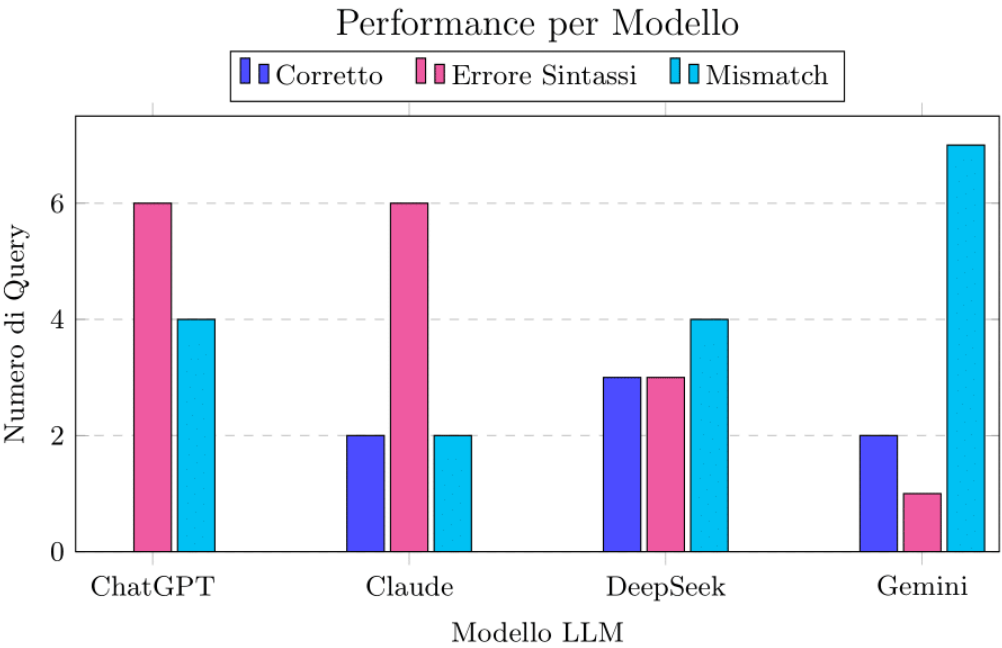


Figura 1: Confronto delle performance dei quattro LLM

Sull'intero set di 40 tentativi (4 modelli  $\times$  10 query):

- Solo il **17.5%** (7/40) delle query risulta completamente corretta
- Il **42.5%** (17/40) produce risultati parziali o imprecisi (mismatch)
- Il **40.0%** (16/40) fallisce per errori di sintassi



Distribuzione Aggregata (Totale 40 query)

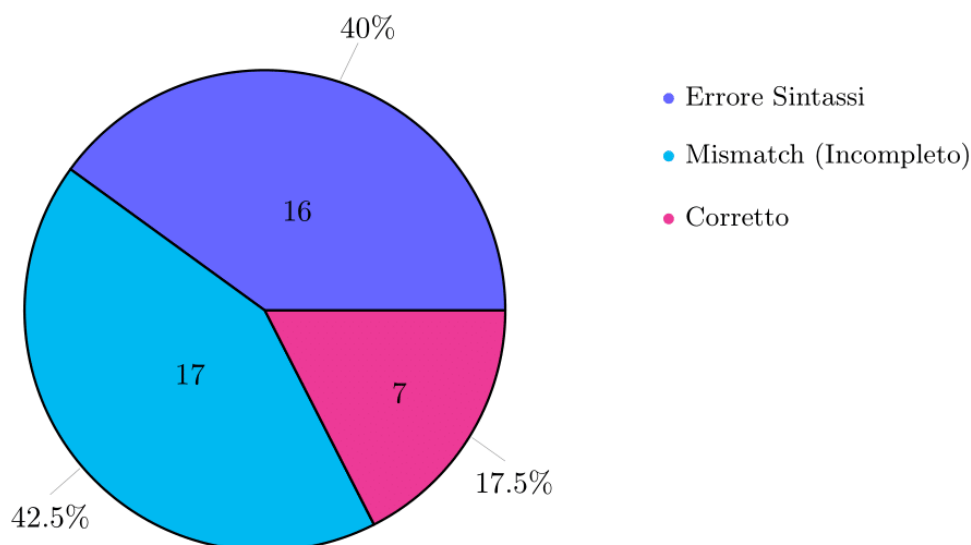


Figura 2: Distribuzione aggregata con metrica simmetrica

### 5.3 Analisi dettagliata dei risultati

#### DeepSeek - Il maggior numero di query corrette ma con errori distribuiti

- **Punti di forza:** 3 query perfettamente corrette, il miglior risultato assoluto
- **Punti critici:** 4 query con mismatch e 3 errori di sintassi
- **Problemi ricorrenti:** Errori di casting ("cannot cast type json to agtype") e problemi con alias nelle sottoquery

#### Gemini - Robusto sintatticamente ma con precisione variabile

- **Punti di forza:** Solo 1 errore di sintassi, dimostra la migliore comprensione della sintassi Cypher
- **Punti critici:** 7 query con mismatch, principalmente per dati extra o mancanti
- **Problemi ricorrenti:** Gestione eccessivamente inclusiva delle relazioni "hasDevice"

#### Claude - Buona precisione quando funziona, ma fragile sintatticamente

- **Punti di forza:** 2 query perfettamente corrette
- **Problema principale:** 6 errori di sintassi per uso di funzioni non standard ("function ANY/distance does not exist")

#### ChatGPT - Le maggiori difficoltà complessive

- **Criticità:** Nessuna query completamente corretta, 6 errori di sintassi

- **Pattern errori:** Accesso improprio a proprietà JSON ("invalid input syntax") e gestione alias ("could not find rte")

## 5.4 Analisi degli errori nel benchmark finale

L'analisi degli errori di sintassi nel benchmark finale rivela che i modelli commettono errori specifici e ricorrenti:

- Claude ha spesso tentato di usare funzioni non standard o non supportate.
- ChatGPT ha mostrato difficoltà nell'accesso a proprietà nidificate e nella gestione degli alias nelle query.
- Un errore comune a tutti i modelli (tranne Claude in un caso) è stata la gestione incorretta degli alias nelle sotto-query o nelle operazioni di aggregazione.

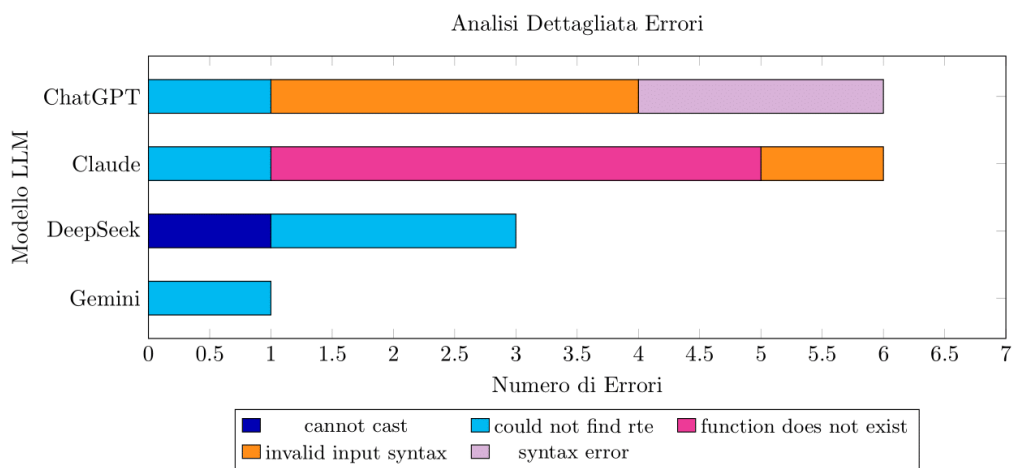


Figura 3: Analisi dei tipi di errori di sintassi per modello nel benchmark finale

L'analisi complessiva conferma che l'arricchimento progressivo del contesto ha un impatto positivo misurabile su tutti i modelli. Tuttavia, non elimina completamente le criticità: anche nella versione più completa (V3), persistono errori di sintassi e mismatch, indicando che la qualità del contesto è necessario ma non sufficiente per una generazione perfetta di query.

## 6. Conclusioni

L'attività di tirocinio ha rappresentato un'opportunità fondamentale per acquisire competenze nell'utilizzo contemporaneo di database a grafo e intelligenza artificiale generativa.

L'attività sperimentale ha dimostrato che, sebbene i LLM mostrino capacità impressionanti, la loro affidabilità nella generazione end-to-end di query Cypher in contesti ibridi è ancora limitata. L'analisi condotta ha rivelato diversi punti chiave:

1. **Robustezza sintattica differenziata:** I modelli mostrano una variabilità significativa nella capacità di generare query sintatticamente corrette. Gemini si è distinto per la minore propensione agli errori di sintassi.
2. **Il problema dell'incompletezza:** Un numero elevato di query, sebbene sintatticamente valide, ha restituito risultati incompleti o imprecisi. Questo rappresenta un vero problema, in quanto l'utente finale potrebbe ricevere dati errati senza un evidente messaggio di errore.
3. **Pattern di errore ricorrenti:** L'analisi ha identificato pattern di errore specifici per modello (es. uso di funzioni non standard, gestione degli alias), utili per indirizzare future ricerche sul miglioramento di questi strumenti.
4. **Impatto progressivo del contesto:** L'arricchimento graduale della struttura del grafo (dalla versione base a quella completa con la tabella measurements) ha mostrato un **miglioramento incrementale delle performance** su tutti i modelli. In particolare, si è osservata una riduzione significativa delle hallucination e degli errori di interpretazione, dimostrando che la completezza del contesto fornito è un fattore determinante per l'accuratezza delle query generate.

Prima di iniziare non avevo mai svolto attività di ricerca e sviluppo; questa è stata la mia prima esperienza in tale ambito, il che ha reso il percorso particolarmente stimolante.

Pur avendo conoscenze di base sui linguaggi di interrogazione come SQL e Python, non avevo alcuna familiarità con i database NoSQL, in particolare con i database a grafo e linguaggi come Cypher. Questo ha comportato un processo di apprendimento intenso e progressivo, che ha richiesto studio, ricerca e applicazione pratica.

Questa esperienza mi ha permesso di sviluppare:

- Capacità di ricerca autonoma e di approfondimento di tecnologie innovative.

- Competenze tecniche nuove, soprattutto nell'interazione tra AI e database strutturati.
- Analisi critica dei risultati, integrando aspetti quantitativi e qualitativi.
- Consapevolezza dell'importanza del prompt engineering e della modellazione dei dati per migliorare le performance dei LLM.

## 7. Bibliografia

Durante la fase di studio sono state consultate le seguenti risorse:

- [LLM-powered GraphQL Generator for Data Retrieval](#)
- [DBSM non relazionali NoSQL](#)
- [Graph DataBase Neo4j](#)
- [MongoDB a document-oriented DataBase](#)
- [Augmented Knowledge Graph Querying leveraging LLMs SparqLLM](#)
- [A survey of large language models for data challenges in graphs](#)
- [A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models](#)
- [CypherBench: Towards Precise Retrieval over Full-scale Modern Knowledge Graphs in the LLM Era](#)
- [Hybrid-LLM-GNN: integrating large language models and graph neural networks for enhanced materials property prediction](#)
- [Injecting Structured Knowledge into LLMs via Graph Neural Networks](#)
- [Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL](#)