

Article

GeoIR-Compiler: A Geospatial Intermediate Representation and Compilation Framework for Chinese Urban Spatial Question Answering

Chaolin Zhang ¹ , Jiqui Deng ^{1,*} , Hui Zhang ^{2,3}, Longbo Li ⁴, Liji Sun ^{2,5} and Xiao Ma ^{2,5} 

¹ School of Geosciences and Info-Physics, Central South University, Changsha 410083, China

² Hunan Center of Natural Resources Affairs, Changsha 410004, China

³ Observation and Research Station of Lengshuijiang Mining Ecological Environmental Monitoring, Ministry of Natural Resources, Changsha 410004, China

⁴ Zhaojin Mining Industry Co., Ltd., 118 Wenquan Road, Zhaoyuan 265400, China

⁵ Technology Innovation Center for Ecological Conservation and Restoration in Dongting Lake Basin, Ministry of Natural Resources, Changsha 410007, China

* Correspondence: csugis@csu.edu.cn

Abstract

Natural-language access to spatial databases requires relation interpretation, entity grounding, metric normalization, and database-specific execution semantics. Direct generation of Structured Query Language (SQL) by large language models (LLMs) can therefore return executable but spatially wrong SQL, especially for Chinese urban questions with aliases, abbreviated place names, and geometry-dependent predicates. This paper presents GeoIR-Compiler, a spatially specialized framework that maps a Chinese question to a typed geospatial intermediate representation (GeoIR), grounds mentions and attributes to database objects, and deterministically compiles the grounded representation into SQL for PostGIS, a spatial database extension for PostgreSQL. The contribution is the specialization of intermediate representations for Chinese urban spatial question answering through explicit spatial relations, metric constraints, grounding records, and PostGIS execution templates. We construct two controlled executable benchmarks, NJ-GeoIR-700 and WH-GeoIR-700, covering retrieval, topology, distance, nearest-neighbor, aggregation, compositional, and alias/noisy-mention queries. Across seven locally served backbones, GeoIR-Full reaches mean execution accuracies of 0.7271 on Nanjing and 0.7363 on Wuhan, outperforming Direct-SQL, Data-Augmented In-Context Learning (DAIL)-SQL-style, and Linking-SQL under the fixed evaluation protocol. Ablations are consistent with grounding contributing strongly to the observed gains, while verification mainly trades coverage for answer reliability.

Keywords: spatial database; natural language interface; Text-to-SQL; geospatial question answering; PostGIS; large language model; intermediate representation



Academic Editors: Eliseo Clementini and Wolfgang Kainz

Received: 12 May 2026

Revised: 25 June 2026

Accepted: 6 July 2026

Published: 8 July 2026

Copyright: © 2026 by the authors.

Published by MDPI on behalf of the International Society for Photogrammetry and Remote Sensing.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

1.1. Motivation

Natural language interfaces to databases (NLIDBs) aim to let users retrieve data without writing formal query languages. This objective is particularly important in urban geographic information system (GIS) settings, where non-expert users may need to search public services, compare administrative areas, or ask distance and topology questions

while the underlying data remain accessible only through Structured Query Language (SQL), GeoSPARQL—a geospatial query standard built on SPARQL (SPARQL Protocol and Resource Description Framework (RDF) Query Language)—or spatial database functions. Comparative NLIDB surveys trace the transition from rule-based interfaces to modern data-driven systems [1]. Deep-learning Text-to-SQL surveys identify schema understanding as a persistent bottleneck [2]. Broader Text-to-SQL surveys further summarize recent progress and open problems in executable semantic parsing [3].

Spatial databases add commitments that ordinary relational Text-to-SQL does not make explicit. A system must identify point, line, and polygon entities; interpret containment, intersection, crossing, proximity, and nearest-neighbor relations; normalize metric expressions; resolve place-name aliases; and select database-specific spatial operations. For example, answering “Which Grade-A tertiary hospitals are within one kilometer of Zhujiang Road subway station?” requires a hospital-level filter, a grounded subway-station anchor, a distance normalized to meters, and a metric PostGIS predicate. A directly generated SQL statement may be executable while still containing the wrong entity, relation, or distance semantics.

Prior spatial database-interface studies emphasize spatial comprehension, database access, and explicit spatial-relation handling as core requirements for natural-language access to spatial data. SpatialNLI formalized natural-language access to spatial-domain databases and highlighted the need for spatial comprehension [4]. Its multilingual extension showed that the same challenge persists across languages and geographic contexts [5]. NALSpatial represents spatial query types and operations in a transformation framework for spatial data access [6]. GS-SQL models spatial semantics for spatial Text-to-SQL, including topological and metric predicates [7]. Recent geospatial large language model (LLM) studies further examine prompt-driven spatial SQL generation, spatial schema descriptions, and geographic disambiguation. Jiang and Yang evaluated ChatGPT for geospatial SQL analysis inside a spatial database [8]. Wang et al. studied GPT-based Text-to-SQL for spatial databases with prompt and schema-description design [9]. Monkuu explored dynamic schema mapping and human-in-the-loop geographic disambiguation for geospatial databases [10]. However, many systems still place semantic interpretation, database binding, and executable query construction inside one generation process. This coupling makes failures difficult to prevent and diagnose: an incorrect answer may originate from the interpretation of the question, the grounding of a mention, the choice of a spatial function, or the generated SQL text, but the final output does not expose which decision failed.

This study addresses that gap with GeoIR-Compiler, a staged framework for Chinese urban spatial question answering (QA) over databases managed with PostGIS, a spatial database extension for PostgreSQL. The language model predicts a typed Geospatial Intermediate Representation (GeoIR) that expresses query intent, target entities, attributes, spatial constraints, aggregation, ranking, and output form. A separate grounding stage binds logical mentions and fields to database objects, after which deterministic templates compile the grounded GeoIR into PostGIS SQL. The design does not claim novelty for intermediate representations or compilation in Text-to-SQL generally. Its narrower contribution is to specialize this separation for spatial semantics, Chinese urban aliases, metric constraints, explicit grounding records, and inspectable PostGIS execution.

1.2. Contributions

The contributions of this paper are as follows:

- We propose GeoIR-Compiler, a spatially specialized GeoIR-grounding-PostGIS framework for Chinese urban spatial question answering. Its main contribution is to make

spatial relations, metric constraints, entity grounding, and PostGIS compilation explicit within a typed intermediate representation pipeline.

- We construct two controlled executable benchmarks: NJ-GeoIR-700 over a cleaned Nanjing spatial database covering eight districts, and WH-GeoIR-700 over an aligned compact Wuhan spatial database. Both provide annotations for GeoIR, grounding, SQL, executable answers, duplicate checks, and reproducible structural quality-control scripts.
- We conduct controlled experiments on seven local backbones from the Qwen2.5, DeepSeek-Coder-V2-Lite, Mistral Small 3.2, and GLM4 families and show that, across both Nanjing and Wuhan, GeoIR-Compiler improves over Direct-SQL, DAIL-SQL-style, and Linking-SQL under a fixed deployment protocol. The ablation pattern is consistent with grounding and staged execution control contributing to the observed reliability gains, without isolating a single component as the sole causal factor.

1.3. Organization

The remainder of the paper is organized as follows. Section 2 reviews Text-to-SQL, spatial database interfaces, and geospatial LLM research. Section 3 defines the framework, GeoIR representation, grounding process, and deterministic compilation procedure. Section 4 describes the two controlled benchmarks, baselines, model settings, and evaluation metrics. Section 5 reports the main, task-level, cross-city, ablation, and error analyses. Section 6 concludes the paper and discusses its limitations.

2. Related Work

2.1. Text-to-SQL and NLDBs

Early NLDB research focused on mapping natural-language questions to database queries through rule-based grammars, semantic parsing, statistical models, and later neural models. Affolter et al. summarize recent natural language database interfaces from a comparative perspective [1]. Katsogiannis-Meimarakis and Koutrika review deep-learning approaches for Text-to-SQL [2]. Deng et al. organize recent Text-to-SQL advances and open expectations for future work [3]. Early semantic parsing work on GeoQuery [11] and SQL-oriented corpora [12] established important foundations for mapping natural language to executable formal queries.

The rise of large-scale Text-to-SQL datasets changed the research focus from small grammar-driven interfaces to neural query generation. Seq2SQL introduced reinforcement learning for generating structured SQL over WikiSQL-style tables [13], and SQLNet proposed a sketch-based method that avoided reinforcement learning [14]. Spider then shifted the field toward complex cross-domain and multi-table semantic parsing [15], while SyntaxSQLNet incorporated syntax-tree structures for the same setting [16]. These benchmarks and models made execution accuracy a central evaluation criterion and clarified that schema linking and query-structure prediction are key bottlenecks.

Chinese Text-to-SQL research further complicates the problem because Chinese questions often contain implicit segmentation, flexible word order, abbreviated entity names, and context-dependent attribute expressions. CSpider provided an early Chinese SQL semantic parsing resource [17]. DuSQL introduced a large-scale pragmatic Chinese Text-to-SQL dataset [18]. TableQA contributed table-aware Chinese SQL generation data [19]. These resources show that Chinese database interfaces require not only SQL generation, but also robust language normalization and entity linking.

On the modeling side, intermediate representations (IRs) and schema-aware encoders have become common. IRNet introduced a SemQL-style intermediate representation to reduce the mismatch between natural language and SQL [20]. TaBERT learned representations that bridge textual and tabular data [21]. RAT-SQL strengthened relation-aware

schema encoding and linking [22]. SmBoP explored bottom-up semi-autoregressive parsing [23]. PICARD constrained autoregressive decoding through incremental SQL parsing [24]. RESDSQL decoupled schema linking from skeleton parsing [25], while later work re-examined the role of schema linking in Text-to-SQL [26]. Data-Augmented In-Context Learning (DAIL)-SQL studied LLM prompting for Text-to-SQL [27]. BIRD (Big Bench for Large-Scale Database Grounded Text-to-SQL Evaluation) further demonstrated the potential and limits of using LLMs as database interfaces at the benchmark scale [28].

For our purposes, the key lesson from this literature is the usefulness of intermediate representations and schema linking. At the same time, most of these methods target ordinary relational databases. Spatial query generation adds geometry, topological predicates, metric reasoning, and reference-object semantics, so reusing a generic SQL representation is often not enough. A spatial interface benefits from treating those spatial commitments as first-class intermediate objects rather than burying them inside free-form SQL generation.

2.2. Spatial Database Interfaces and Geospatial LLMs

Natural language interfaces for spatial databases have emphasized spatial comprehension for a long time. Compared with ordinary relational questions, spatial questions often contain prepositions, implicit spatial frames, topology, distance, direction, scale, and geographic entity names. SpatialNLI formalized the problem of natural-language access to spatial-domain databases and highlighted the need for spatial comprehension [4]. Its multilingual extension showed that the challenge remains important across languages and that spatial database interfaces must handle linguistic and geographic variation simultaneously [5].

Transformation-based spatial interfaces have also been studied. NALSD developed a natural language interface for spatial databases [29]. NALSpatial first appeared as a transformation framework for queries over spatial data [30] and was later extended into a journal-level interface study [6]. These systems demonstrate that spatial query generation benefits from explicit handling of spatial relations and database operations, rather than treating spatial predicates as ordinary textual tokens.

Another line of work focuses on spatial query representations. Xiao and Li studied an improved abstract syntax tree for spatial Text-to-SQL in the geography domain [31]. GS-SQL modeled spatial semantics in spatial Text-to-SQL and explicitly emphasized predicates such as containment, intersection, crossing, and metric distance [7]. These methods are close in spirit to the present study because they recognize that spatial relations should be represented before SQL generation.

The present work differs in two aspects. First, it is designed around a Chinese urban spatial database, where entity aliases, district names, subway stations, and point-of-interest (POI)-style layers create a strong grounding requirement. Second, it treats semantic parsing, grounding, and compilation as separate modules and evaluates the system primarily through grounding accuracy, execution accuracy, and accuracy under coverage. Table 1 therefore compares named recent studies using design features rather than accuracy values from incompatible datasets.

Large language models have also entered this space. Jiang and Yang evaluated ChatGPT for geospatial SQL analysis inside a spatial database [8]. Wang et al. examined GPT-based Text-to-SQL for spatial databases and highlighted the role of prompt design and spatial schema descriptions [9]. Monkuu explored an LLM-based natural language interface for geospatial databases with dynamic schema mapping and human-in-the-loop geographic disambiguation [10]. Taken together, these studies show that LLMs contribute useful language understanding, but they also make clear that spatial database access does not become reliable by prompt-driven free-form generation alone.

Table 1. Feature/design comparison of representative recent spatial database and geospatial natural-language-interface studies. The table compares reported design properties and evaluation settings; it is not a numerical ranking or leaderboard.

Study	Target/Backend	Spatial Representation	Grounding or Schema Handling	Execution and Evaluation Scope
NALSD (2023) [29]	Spatial database interface	Spatial query transformation	Explicit grounding record not reported	Transformation system; deterministic compiler details not explicit; cross-city urban benchmark not reported
NALSpatial (2025) [6]	Spatial database queries	Spatial query types and operations represented in the framework	Key-entity extraction and translation; explicit grounding record not reported	Translates natural language to executable database queries; cross-city urban benchmark not reported
GS-SQL (2024) [7]	Spatial Text-to-SQL	Spatial semantics modeled for SQL generation	Explicit grounding record not reported	SQL generation with spatial-semantic modeling; cross-city urban benchmark not reported
ChatGPT for geospatial SQL (2024) [8]	Spatial database SQL	No explicit spatial IR reported	Not explicit	LLM-generated SQL for geospatial database analysis; no cross-dataset ranking here
GPT-based spatial Text-to-SQL (2025) [9]	Spatial database SQL	Prompted spatial database knowledge; explicit IR not reported	Schema information in prompts; explicit grounding record not reported	GPT-based SQL generation; no cross-dataset ranking here
Monkuu (2026) [10]	Geospatial database SQL	LLM-based natural-language-to-SQL (NL-to-SQL) interface; explicit IR not reported	Dynamic schema mapping and geographic disambiguation reported	LLM-generated geospatial SQL; cross-city urban benchmark not reported
Text-to-OverpassQL (2024) [32]	OpenStreetMap (OSM)/OverpassQL	OverpassQL target query form	OSM target entities are expressed through generated OverpassQL; explicit grounding record not reported	Generates and executes OverpassQL queries in an OSM setting
ChatGeoAI (2024) [33]	PyQGIS geospatial analysis code	Geospatial task interpretation with ontology support	Entity recognition and ontology mapping; explicit database grounding record not reported	Generates executable PyQGIS scripts; not a SQL benchmark
GeoIR-Compiler (this study)	Chinese urban PostGIS QA	Typed semantic GeoIR with spatial constraints, aggregation, ranking, and output form	Explicit grounding records for database objects and fields	Grounded GeoIR deterministically compiled to PostGIS SQL; two controlled city benchmarks

Work outside relational SQL points in the same direction. NLmaps introduced a natural language interface for querying OpenStreetMap [34]. Text-to-OverpassQL extended that idea to more complex OpenStreetMap geodata querying [32]. Potnis et al. studied natural-language question answering over earth observation linked data [35]. ChatGeoAI examined how LLMs may support public geospatial analysis through natural language [33]. These systems broaden the landscape of geospatial interfaces, but they also reinforce a common lesson: domain-specific representations and controlled execution matter.

Despite these advances, most LLM-based geospatial systems remain close to a direct-generation paradigm. The model is expected to generate a final SQL, OverpassQL, or analysis script from a prompt. This design is simple, but it makes execution-critical choices difficult to inspect: an error in entity grounding, unit normalization, spatial-function choice, or output formatting may only be discovered after execution. By contrast, GeoIR-Compiler exposes these decisions as explicit intermediate artifacts.

The present work sits at the intersection of Text-to-SQL, spatial database interfaces, and LLM-based geospatial querying. From Text-to-SQL, it adopts execution accuracy as the main evaluation target and takes seriously the role of intermediate representations. From spatial database interface research, it inherits the view that spatial relations and database grounding are core components rather than incidental prompt details. It also uses modern LLMs, but limits their role mainly to semantic decomposition.

Accordingly, the contribution is not a new large model or another prompt-engineering variant for direct SQL generation, and it does not claim novelty for intermediate representations or compilation in Text-to-SQL as a whole. The contribution is narrower and more concrete: a spatially specialized intermediate pipeline in which natural language is mapped to a typed GeoIR with explicit spatial relations and metric constraints, grounded to concrete database objects, and compiled into PostGIS SQL. That design lets us compare direct SQL generation with a structured interpretation–grounding–execution pipeline under the same backbones and database conditions.

3. Materials and Methods

3.1. Overview and Design Rationale

The preceding discussion points to a specific design problem. In Chinese urban spatial QA, a system must decide what spatial relation is being asked, identify the target and anchor entities, normalize aliases and distance expressions, choose the correct geometry operation, and still produce executable database code. A Direct-SQL system asks the language model to make all of these choices in a single generation step. That is a convenient interface, but it entangles semantic interpretation, database binding, and spatial execution in a way that makes errors hard to prevent and hard to diagnose.

GeoIR-Compiler therefore adopts a staged interpretation–grounding–execution design. The LLM is used for the part where it is most useful: mapping the Chinese question to a typed logical representation. Database-dependent decisions are delayed to local modules that can use fixed schema mappings, alias inventories, spatial templates, and execution checks. The overall framework is shown in Figure 1. Given a Chinese natural-language question, the LLM first produces a semantic GeoIR object. A grounding module then resolves entity mentions and logical attributes to database objects. The compiler maps the grounded representation into PostGIS SQL, and a verifier checks basic execution-oriented constraints before the query is executed.

Figure 1 separates the workflow into six layers. The user/application layer supplies the Chinese question and receives either a formatted answer or a recorded rejection. The semantic parsing layer contains the LLM parser, which maps the question to semantic GeoIR using the model-visible logical schema card. The GeoIR/type-checking layer applies specification-level validity and compatibility conditions before any database binding. The grounding and schema-binding layer uses system-owned alias, logical-to-physical, and physical-schema resources to produce grounded GeoIR records or to stop on missing or ambiguous bindings. The compilation and verification layer applies GeoIR-to-PostGIS templates and conservative execution-oriented checks. The final layer executes accepted SQL against the frozen PostGIS snapshot and formats the returned scalar, Boolean, or entity-list answer. For the benchmark protocol, the supporting resources have fixed origins: the logical schema card comes from the benchmark/task schema specification, the alias index from curated entity aliases and normalization resources, the physical schema from the frozen PostGIS database snapshot, and the GeoIR-to-PostGIS templates from system-owned compiler templates.

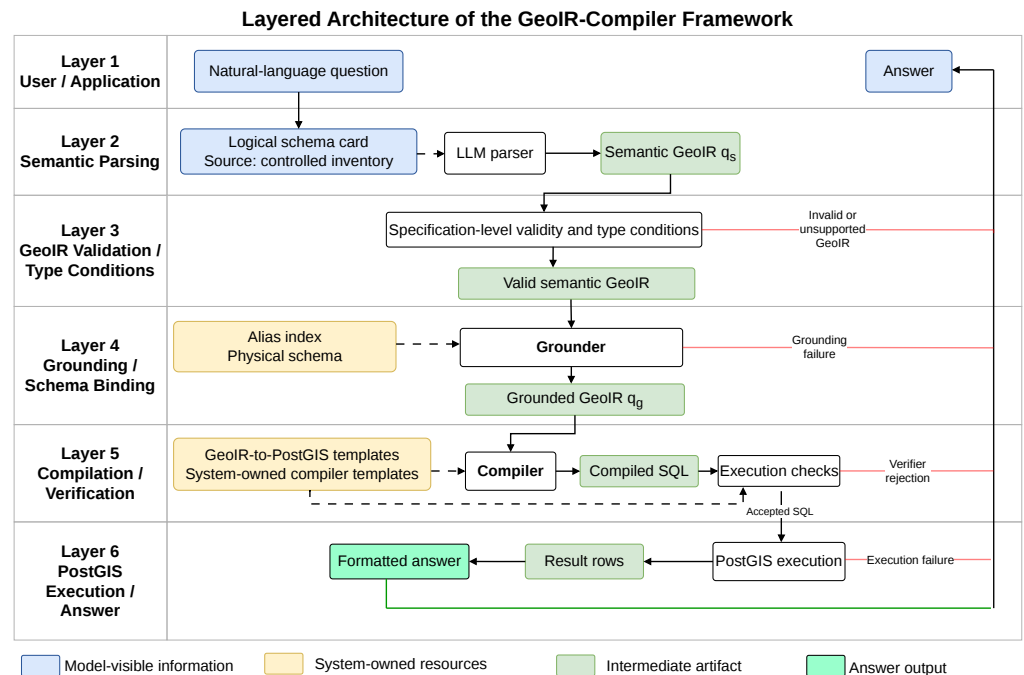


Figure 1. Layered architecture of GeoIR-Compiler. Solid arrows show the main artifact flow from a natural-language question to semantic GeoIR, grounded GeoIR, compiled SQL, and answer output. Dashed arrows show supporting information. The LLM parser receives only model-visible inputs: the user question and the logical schema card. System-owned resources include the alias index, logical-to-physical schema mappings, frozen physical schema, GeoIR-to-PostGIS templates, and execution checks. The side exits distinguish invalid or unsupported GeoIR, grounding failure, verifier rejection, and execution failure; these operational checks do not prove semantic correctness.

3.1.1. Use-Case Sequence

Figure 2 presents the operational use case at two levels. Figure 2a summarizes the user-facing workflow from question submission through parsing, validation, grounding, compilation/verification, and database execution. Figure 2b shows the corresponding message sequence among the user/application, LLM parser, GeoIR validator, grounder, compiler/verifier, PostGIS executor, and response formatter. The solid path passes a Chinese spatial question through semantic GeoIR q_s , valid semantic GeoIR, grounded GeoIR q_g , accepted SQL, result rows, and a formatted answer. Dashed alternatives record invalid or unsupported GeoIR, missing or ambiguous grounding, verifier rejection, execution failure, and the resulting failure/rejection response. These are operational exits rather than a proof that an accepted answer matches the user's latent intent. The hospital-near-subway instance and its concrete representations are shown in Figure 3.

3.1.2. Design Rationale

The framework is built around four design principles. First, the representation should be typed. Spatial relations, entity types, query intents, and output forms are selected from predefined inventories rather than expressed as arbitrary strings. This design reduces the space of possible model outputs and allows the compiler to reject unsupported combinations before execution.

Second, semantic parsing should be separated from physical schema binding. In the full pipeline, the model is not required to know that hospitals are stored in a table named `nanjing_spatial.hospitals`, that the geometry column is `geom`, or that subway stations are identified by `station_id`. These details are handled by local mappings. This is different

from Direct-SQL, where the model must simultaneously understand the question, select the table, select columns, decide the spatial function, and produce valid SQL.

Third, entity grounding should be explicit. Chinese urban questions frequently contain mentions that are not identical to database names. District suffixes may be omitted, subway stations may be referred to with the suffix “站”, and some entities may have shortened or normalized forms. If the system directly inserts raw mentions into SQL, these surface differences become execution errors or empty results. Therefore, grounding is treated as a separate module with its own evaluation metric.

Fourth, compilation should be deterministic and inspectable. Once a grounded GeoIR is produced, the compiler applies predefined mappings from spatial relations to PostGIS operators. This makes the final SQL easier to audit and makes failures easier to attribute. If an answer is wrong, the error can be traced to semantic parsing, grounding, compilation, or execution, rather than being treated as an opaque model-generation failure.

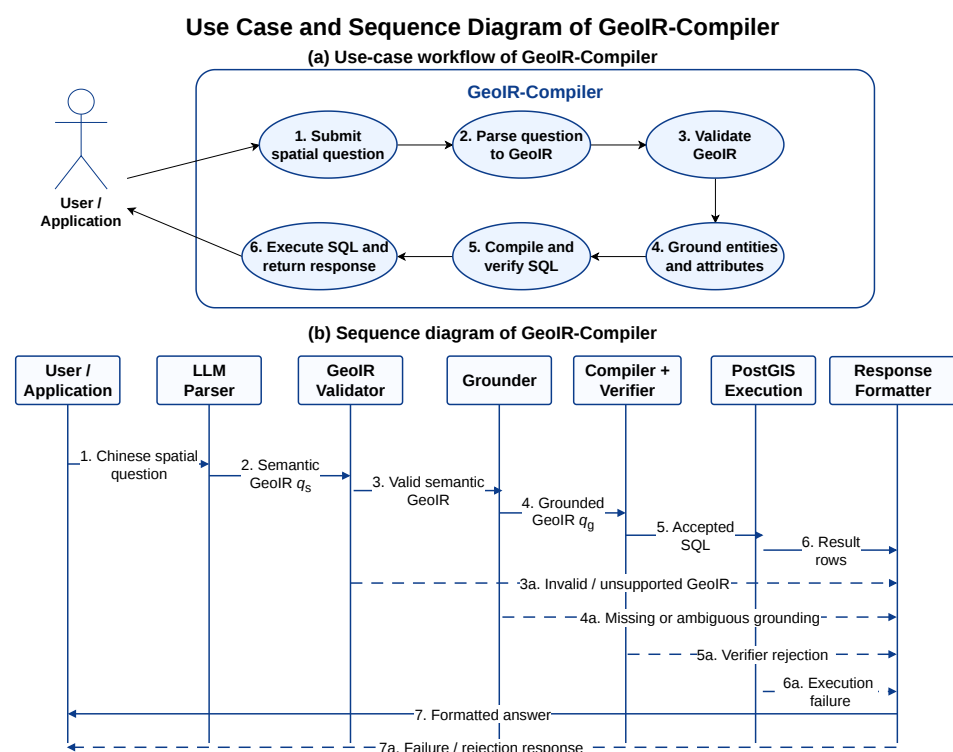


Figure 2. Use-case workflow and interaction sequence of GeoIR-Compiler. (a) gives the user-facing workflow; (b) shows the component-level message sequence. Solid arrows denote the successful path from a Chinese spatial question to a formatted answer, whereas dashed arrows denote invalid or unsupported GeoIR, missing or ambiguous grounding, verifier rejection, execution failure, and the returned failure/rejection response. The diagram summarizes operational control flow and does not claim semantic-correctness proof.

3.1.3. Schema Exposure

For experimental fairness, this paper distinguishes between logical schema information and physical schema information. The logical schema describes the task world: supported entity types such as hospital, park, district, river, road, school, and subway station; supported attributes such as hospital level or subway line; and supported relations such as within, intersects, near, and nearest. The physical schema describes the actual database implementation, including table names, primary keys, geometry columns, and PostGIS functions.

All model-based methods receive the same logical schema card. Direct-SQL additionally receives a physical schema summary because it must generate executable SQL by itself. In contrast, the GeoIR pipeline does not expose physical schema details to the model; grounding and compilation modules own this information locally. The resulting protocol equalizes model-visible logical knowledge, but it should be interpreted as a system-level comparison under a fixed deployment setting rather than a claim that the two methods rely on identical internal knowledge sources.

3.2. GeoIR Representation and Grounding

3.2.1. Semantic GeoIR Syntax

The parser produces a semantic GeoIR object q_s , defined as a typed intermediate representation rather than as an empirically derived equation:

$$q_s = \langle I, T, A, S, G, R, O \rangle \in \mathcal{Q}_s,$$

where I is the query intent, T is the target entity specification, A is a list of attribute constraints, S is a list of spatial constraints, G is an optional aggregation specification, R is an optional ranking specification, and O is the output specification. The equality sign defines the component structure of the representation; it is not a fitted model, empirical relationship, or equation requiring numerical derivation. The controlled entity-type inventory is

$$\mathcal{E} = \{\text{district, hospital, park, river, road, school, subway_station}\},$$

and the query-intent inventory is

$$\mathcal{I} = \{\text{select, count, exists, argmin, argmax}\}.$$

The abstract syntax used by the parser is:

$$\begin{aligned} q_s &::= \langle i, t, A, S, g, r, o \rangle, & i &\in \mathcal{I}, \\ t &::= \text{entity}(\tau, m), & \tau &\in \mathcal{E}, \quad m \in \Sigma^* \cup \{\perp\}, \\ A &::= [a_1, \dots, a_p], & a &::= \text{attr}(f, op, v), \\ S &::= [s_1, \dots, s_q], & s &::= \text{spatial}(\rho, t_a, d), \\ g &::= \perp \mid \text{aggregate}(\alpha, u, h), \\ r &::= \perp \mid \text{rank}(\mu, order, k), \\ o &::= \text{output}(return_type, limit). \end{aligned} \tag{1}$$

Here, Σ^* is the set of mention strings and $\perp$ denotes an absent optional field. The attribute field f , comparison operator op , aggregation operator α , grouping field h , ranking metric μ , and output values are selected from the finite logical-schema inventories supplied to the parser. The spatial-relation inventory used in the benchmarks is $\{\text{within, intersects, crosses, near}\}$; nearest and farthest requests are represented through R , with an anchor in S .

Table 2 summarizes the controlled GeoIR schema used in the experiments. The schema is intentionally narrower than SQL: it represents the spatial meaning required by the benchmark, while leaving physical table names, geometry columns, and PostGIS functions to grounding and compilation.

Table 2. Core GeoIR schema and controlled value domains used by the parser. Values are logical and model-visible; physical database bindings are added only after grounding.

Field	Supported Values or Structure	Role and Validity Note
query_type/ <i>i</i>	select, count, exists, argmin, argmax	Query intent; must be compatible with aggregation, ranking, and output form
target/ <i>t</i>	entity(τ, m), $\tau \in \mathcal{E}$, $m \in \Sigma^* \cup \{\perp\}$	Main entity set to retrieve, count, or rank; mention resolution is deferred to grounding
attribute_constraints/ <i>A</i>	attr(f, op, v), $f \in \mathcal{F}(\tau)$; operators include equality, inequality, ordering, contains, and in where type-valid	Non-spatial filters; values v must match the field type or controlled category inventory
spatial_constraints/ <i>S</i>	spatial(ρ, t_a, d); ρ is within, intersects, crosses, or near	Topological or metric constraints; near requires finite $d \geq 0$ meters, whereas topological relations require $d = \perp$
aggregation/ <i>g</i>	$\perp$ or aggregate(α, u, h), $\alpha \in \{\text{count}\}$, $h = \perp$ or a valid grouping field	Scalar counts and group-level counts; grouping is used for district-level comparison tasks
ranking/ <i>r</i>	$\perp$ or rank($\mu, order, k$); μ is distance or count; $order$ is asc or desc; $k \geq 1$	Nearest/farthest retrieval and aggregate comparison; requires a compatible metric expression
output/ <i>o</i>	output($return_type, limit$); return type is entity_list, scalar, or boolean	Answer shaping; select/argmin/argmax return entities, count returns a scalar, and exists returns a Boolean

$\mathcal{F}(\tau)$ denotes the logical field inventory available for entity type τ . In the current benchmark it includes identifiers and display names for all entity types, entity-specific attributes such as hospital level, school type, park type, road type, subway line, interchange flag, district label, and geometry-derived attributes used by the compiler. The value domain of v is determined by the selected field: categorical fields accept only values from the corresponding controlled inventory, numeric fields accept numeric thresholds with their normalized units, and name-like fields accept mention strings for later grounding.

The geometry signature of an entity type determines which relations are legal. hospital, school, and subway_station are point-like service or reference entities; road and river are line entities; district and park are polygon entities. In the implemented templates, within is accepted for point-like or line targets inside polygon anchors and for polygon targets inside polygon anchors; intersects is accepted for line–polygon, polygon–polygon, and point-like–polygon combinations; crosses is accepted for line–polygon and line–line combinations; and near and distance ranking are accepted for supported target–anchor pairs whose geometries can be cast to geography. These are compatibility rules for the controlled benchmark templates, not a complete inventory of all possible PostGIS predicates.

Aggregation and ranking are also constrained. Plain count queries use $g = \text{aggregate}(\text{count}, u, \perp)$ and scalar output. Grouped comparison queries use h , typically a district grouping field, and may add $r = \text{rank}(\text{count}, order, k)$ to express largest or smallest grouped counts. For example, a T5 question asking which district has the most parks is represented as a count aggregation over parks grouped by district, ranked by count in descending order with $k = 1$, and returned as an entity-list or name-plus-count

output according to O . The compiler rejects aggregation, ranking, grouping, or output combinations outside these controlled patterns.

3.2.2. Validity Conditions

A semantic GeoIR object is accepted only if it satisfies the controlled type rules in Equation (1) and Table 2. First, each attribute field must belong to $\mathcal{F}(\tau)$, and its operator and value must be valid for that field. Second, the target and anchor types must be supported by the geometry signature of the selected spatial relation. Third, near requires a finite distance $d \geq 0$, stored in meters after unit normalization, whereas non-metric topological relations do not accept a distance. Fourth, intent, aggregation, ranking, grouping, and output form must agree with one of the controlled combinations described above. Combinations that violate these conditions are semantic-validation failures: they are outside $\mathcal{Q}_s$ and therefore have no valid grounded or compiled form under this specification. Mention resolution is deliberately not part of semantic validity: a structurally valid mention may still fail or remain ambiguous when matched against a particular database.

The semantic representation contains only logical concepts. For example, a question such as “珠江路地铁站1公里内的三级甲等医院有哪些” is represented as a hospital retrieval query with a distance constraint around a subway station and an attribute constraint on hospital level. A simplified valid example is shown below:

```
{
  "query_type": "select",
  "target": {"entity_type": "hospital", "mention": null},
  "attribute_constraints": [
    {"field": "hospital_level", "op": "=", "value": "三级甲等"}
  ],
  "spatial_constraints": [
    {
      "relation": "near",
      "anchor": {"entity_type": "subway_station", "mention": "珠江路地铁站"},
      "distance_m": 1000
    }
  ],
  "aggregation": null,
  "ranking": null,
  "output": {"return_type": "entity_list", "limit": null}
}
```

For contrast, a near constraint with $\text{distance_m} = -500$ does not satisfy the semantic GeoIR validity conditions because metric thresholds must be non-negative. Its invalidity is structural and does not depend on whether the anchor mention exists in a database.

Figure 3 instantiates the representation and compilation stages for the hospital-near-subway query. The semantic object q_s contains only the logical intent, target, attribute constraint, spatial relation, metric threshold, and output form. Grounding resolves the station mention to the canonical Nanjing station “珠江路” and its snapshot-specific identifier 13, while binding the hospital and subway-station fields. Deterministic templates then introduce the PostGIS distance predicate and produce the illustrated SQL. For visual compactness, the SQL panel shortens the table labels after the physical binding context has been established. The identifier is traceable to the frozen Nanjing database snapshot and is not a city-independent constant.

Example of GeoIR Grounding and PostGIS Compilation

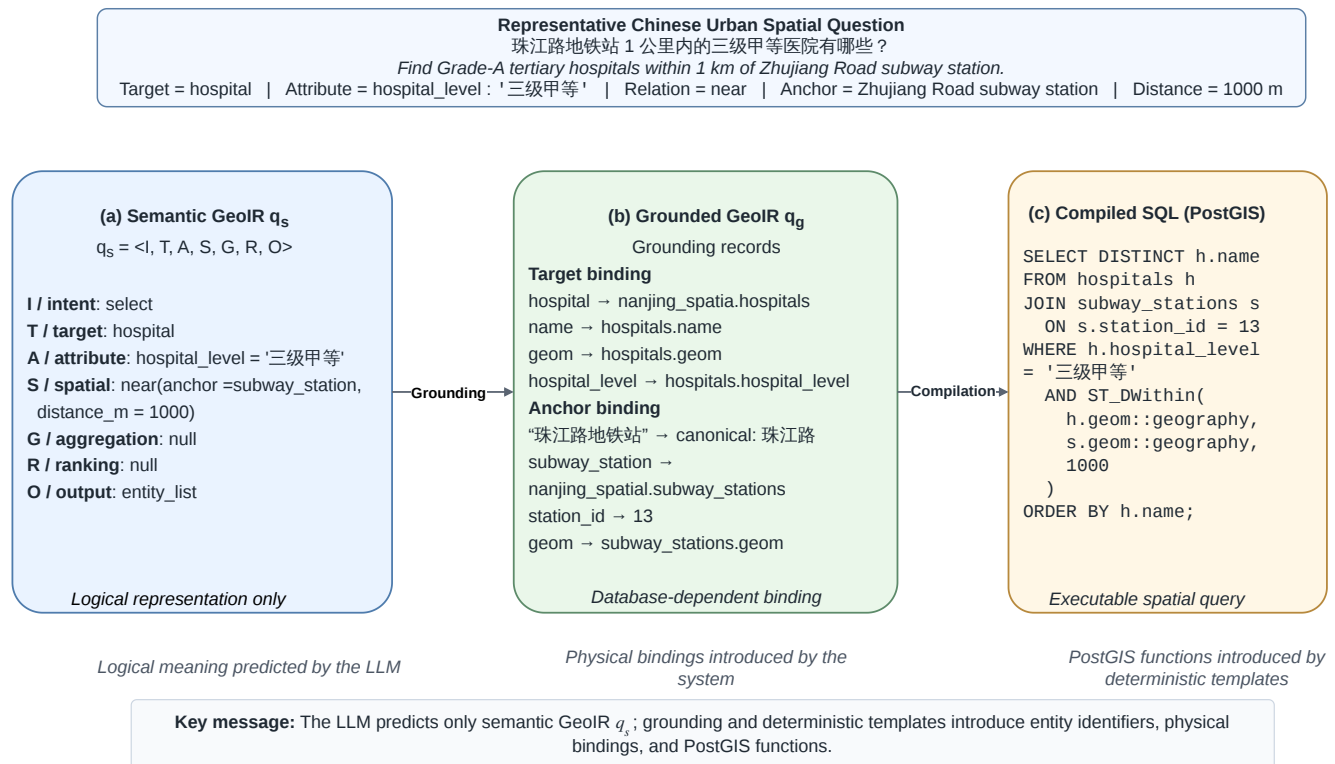


Figure 3. Worked example of semantic GeoIR, grounded GeoIR, and deterministic PostGIS compilation. Panel (a) contains the model-predicted logical representation q_s ; panel (b) adds database-dependent target and anchor bindings, including the snapshot-specific canonical station identifier; panel (c) shows the resulting spatial SQL with the hospital-level predicate and a 1000 m ST_DWithin constraint. The concrete identifier belongs to the frozen Nanjing snapshot and is not a transferable constant.

3.2.3. Grounding

Grounding is a database-dependent partial mapping

$$\Gamma_D : \mathcal{Q}_s \rightarrow \mathcal{Q}_g,$$

where D is a frozen database snapshot and $\mathcal{Q}_g$ is the set of grounded GeoIR objects. It maps logical entity types to tables, logical fields to columns, and each non-null mention to a binding record

$$b = \langle m, \tau, \text{table}, \text{id}, \text{canonical}, \text{strategy} \rangle,$$

where m is the original mention, τ is the expected and resolved entity type, table and id identify the physical database object, canonical is its stored canonical name, and strategy records whether the binding came from an exact, alias, or normalized match.

The resulting grounded object $q_g = \Gamma_D(q_s)$ retains the semantic structure of q_s but adds the physical bindings required for execution. Grounding follows a fixed precedence. It first performs canonical-name matching within the entity type specified by the semantic slot. If no unique exact match is available, it consults the alias index under the same type constraint. It then applies the configured mention normalizations, such as optional district and station suffix handling, and repeats the type-constrained lookup. A single remaining candidate supplies the binding record; no candidate is a missing-binding failure, and multiple candidates are an ambiguity failure. In either failure case, Γ_D is undefined and no grounded GeoIR exists for that input.

This matters because Chinese spatial questions often use abbreviated or non-canonical names. For example, “玄武” should be grounded to the district “玄武区”, and “珠江路站” should be grounded to the subway station “珠江路”. Without an explicit type-constrained grounding stage, the compiler relies more heavily on raw or fallback mentions, which is consistent with the lower final answer accuracy observed in the ablation results in Section 5.

Candidate entities remain constrained by the expected entity type throughout this procedure; for example, a mention inside a subway-station anchor slot cannot resolve to a road even if the surface forms overlap. Grounding failure is distinct from semantic invalidity: an unsupported relation, field, or output combination is outside $\mathcal{Q}_s$, whereas a structurally valid q_s may still have no image under Γ_D for a particular database snapshot.

These explicit records feed both the compiler and the evaluation module. As a result, we can separate a correct semantic GeoIR structure with wrong entity grounding from an incorrect semantic structure that nonetheless yields syntactically valid SQL.

3.2.4. Compiled SQL

Compilation is defined only over grounded representations:

$$\mathcal{C}_D : \mathcal{Q}_g \rightarrow \text{SQL}.$$

For a valid grounded object q_g , $\mathcal{C}_D(q_g)$ is the SQL text produced by the database-specific templates. The mapping is partial because an otherwise grounded object may still contain an unsupported relation–geometry or output combination. Semantic GeoIR is therefore not compiled directly: the execution path is $q_s \xrightarrow{\Gamma_D} q_g \xrightarrow{\mathcal{C}_D} \text{SQL}$. These definitions specify the interfaces and rejection conditions of the stages; they are not presented as a proof that every natural-language question has a valid or correct compilation.

3.3. Deterministic Compilation and Verification

After grounding, the compiler maps GeoIR constraints to deterministic SQL templates. The benchmark templates use the following relation mappings:

`within` $\rightarrow$ `ST_Within`,
`intersects` $\rightarrow$ `ST_Intersects`,
`crosses` $\rightarrow$ `ST_Crosses`,
`near` $\rightarrow$ `ST_DWithin`,
`nearest/farthest` $\rightarrow$ `ST_Distance` with ascending/descending ordering.

For `within`, `intersects`, and `crosses`, the target and anchor geometry columns remain geometry values and must have compatible spatial reference identifiers in the database snapshot. For `near` and distance ranking, both operands are cast to geography; the normalized distance d is then interpreted in meters by `ST_DWithin` or `ST_Distance`. The target–anchor argument order follows the semantic relation: for example, `within` compiles as a target geometry within an anchor geometry.

Non-spatial attribute constraints become typed `WHERE` predicates over grounded columns. A count intent applies the aggregation after its attribute and spatial filters, and an optional grouping field becomes a `GROUP BY` key. Ranking compiles the metric expression into `ORDER BY`, with ascending order for `nearest` and descending order for `farthest`; k and an explicit output limit become `LIMIT` clauses. Output shaping selects the projection required by `O`: an entity-name list, a scalar aggregate, or a Boolean existence result. If a grounded object requests a relation–geometry, aggregation, ranking, or output combination for which no template is defined, $\mathcal{C}_D$ is undefined and no SQL is produced.

This unsupported-compilation condition is separate from both grounding failure and verifier rejection.

Verifier

The verifier is applied only after a grounded object has been compiled. It checks execution-oriented preconditions recorded by the pipeline, including complete required bindings, a supported relation/template selection, finite non-negative metric parameters, and consistency between the compiled projection and the requested output form. A verifier rejection therefore means that compiled SQL is withheld from covered execution because these conservative checks are not satisfied. By contrast, grounding failure produces no q_g , and an unsupported GeoIR combination produces no SQL under $\mathcal{C}_D$. The verifier does not prove that accepted SQL is semantically correct, nor is it a complete answerability detector. In the experiments, its observed role is conservative quality control: it reduces coverage while increasing reliability among answered questions.

4. Experiments

4.1. Study Area and Benchmark

The experiments use a cleaned urban spatial database stored in PostGIS and covering eight districts of Nanjing rather than the full municipal area: Xuanwu, Qinhuai, Gulou, Jianye, Yuhuatai, Qixia, Pukou, and Jiangning. The database contains 8 district polygons, 244 roads, 60 rivers, 50 parks, 66 schools, 24 hospitals, 33 subway stations, and 521 entity-alias records. Each layer stores geometry information and task-relevant attributes such as district name, hospital level, park type, road type, school type, subway line, and interchange status. The entity alias table supports canonical-name resolution and normalized Chinese mention matching.

Figure 4 visualizes the spatial distribution of the Nanjing and Wuhan benchmark database layers. The maps are data-coverage illustrations rather than external cartographic base maps or comprehensive city maps. The displayed administrative district polygons, roads, rivers, parks, schools, hospitals, subway stations, and alias-supported entities provide the layer inventory used by the T1–T7 benchmark tasks: district retrieval and aggregation, topological predicates over points/lines/polygons, metric distance and nearest/farthest queries, compositional constraints, and alias or noisy-mention grounding. The figure should not be used to infer city scale, service level, spatial density, or completeness beyond the retained experimental snapshots.

Table 3 lists the corresponding logical entity types, retained database counts, representative attributes, and spatial roles used by the experiments.

Table 3. Main logical entity types, database counts, and representative attributes used in the experiments.

Logical Entity Type	Count	Representative Attributes	Spatial Role
District	8	name, display name, area	administrative polygon
Hospital	24	name, hospital level, hospital type, district	point-like service entity
Park	50	name, park type, area, district	polygon/area entity
River	60	name, waterway type, length, district	line entity
Road	244	name, road type, lanes, length, district	line entity
School	66	name, school type, district	point-like service entity
Subway station	33	name, line name, interchange flag, district	point-like reference entity

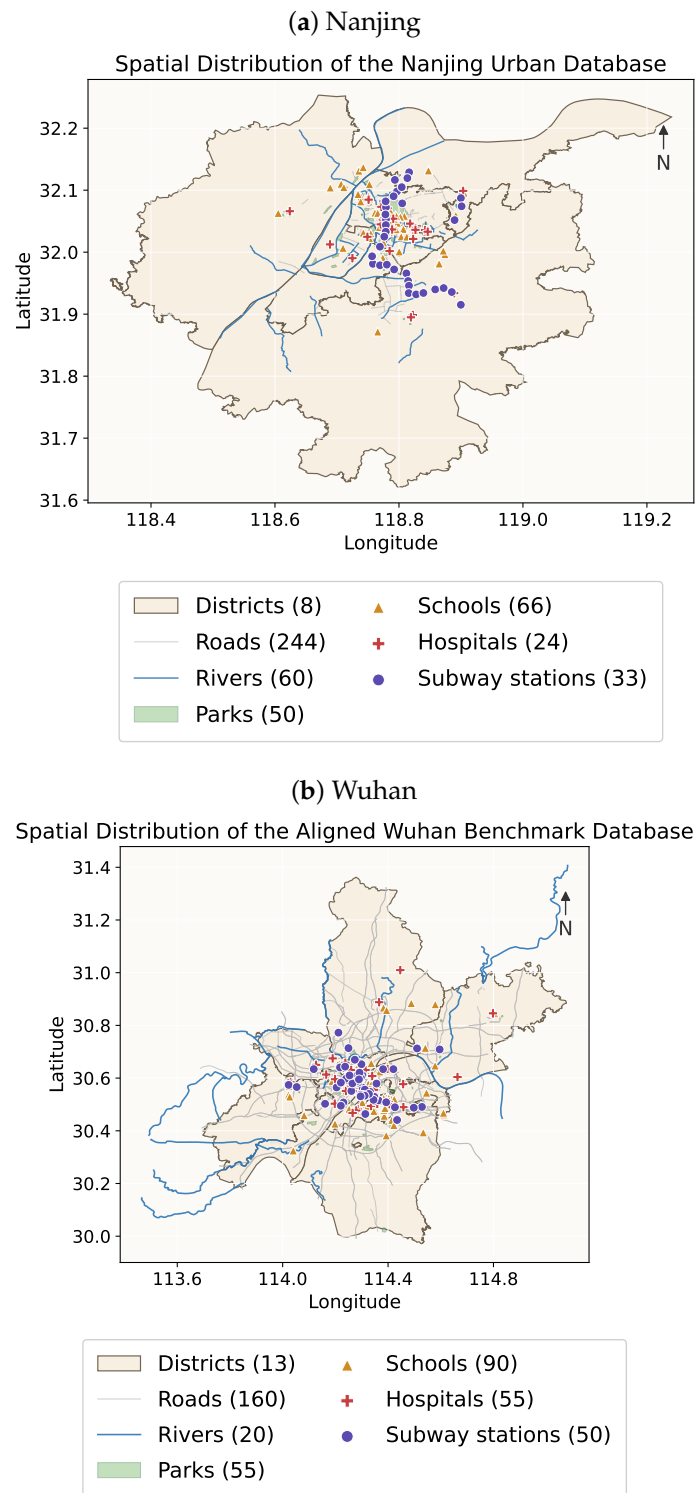


Figure 4. Benchmark database coverage for the two city snapshots. Panel (a) shows the retained Nanjing layers used in NJ-GeoIR-700 and panel (b) shows the aligned Wuhan layers used in WH-GeoIR-700. The panels are stacked and displayed larger than the previous side-by-side layout to improve the readability of the embedded legends and dense urban symbols without changing the underlying figure data. Both panels display the same benchmark layer types—administrative district polygons, roads, rivers, parks, schools, hospitals, and subway stations—which support T1–T7 by supplying administrative containment, topological relations, distance and ranking anchors, aggregation groups, compositional constraints, and alias-grounded entities. The panels describe experimental database coverage only; they are not comprehensive city cartography and should not be interpreted as evidence of city scale, service level, spatial density, or spatial completeness.

Table 4 separates the characteristics of the two real cities from the experimental database snapshots and the controlled benchmark design. According to official 2024 city statistics, Nanjing had 9.5770 million permanent residents, an administrative area of 6587.04 km², and 11 municipal districts, whereas Wuhan had 13.8091 million permanent residents, an administrative area of 8569.19 km², and 13 administrative districts [36,37]. Wuhan is therefore larger than Nanjing in both resident population and administrative area. These real-city differences provide case-study context only: the experimental databases are cleaned, retained snapshots rather than complete municipal inventories, so the official city statistics cannot be used to infer benchmark completeness or difficulty. Nanjing is used as the primary benchmark because the initial database cleaning, schema design, alias normalization, GeoIR annotation, and main system comparison were developed around this snapshot. Wuhan is used as an aligned cross-city validation city: it changes the city-specific database content and aliases while retaining the same task inventory and evaluation protocol. The equal 700-question design controls task composition; it does not imply that the two real cities, database snapshots, or covered areas have the same scale or completeness.

Table 4. Real-city context and roles of the two city benchmarks. Official 2024 population, administrative area, and district counts are reported separately from the retained experimental database snapshots and controlled benchmark design.

Aspect	Nanjing	Wuhan
Role in this paper	Primary benchmark used for the main schema, annotation, and system comparison	Controlled aligned cross-city validation benchmark
Year-end resident population (2024)	9.5770 million	13.8091 million
Administrative area	6587.04 km ²	8569.19 km ²
Municipal districts (2024)	11	13
Experimental snapshot	Cleaned eight-district PostGIS database covering Xuanwu, Qinhuai, Gulou, Jianye, Yuhuatai, Qixia, Pukou, and Jiangning	Benchmark-oriented compact PostGIS database derived from cleaned OSM layers and local normalization
Database counts used here	8 district polygons, 244 roads, 60 rivers, 50 parks, 66 schools, 24 hospitals, 33 subway stations, and 521 entity-alias records	13 districts, 160 roads, 20 rivers, 55 parks, 90 schools, 55 hospitals, and 50 subway stations, for 443 core entities
Benchmark design	700 questions, 175/525 development/test split, and T1–T7 task inventory	Same 700-question split and T1–T7 task inventory to control task composition
Scope caveat	Covers the retained database snapshot, not the full municipality or production query traffic	Tests one controlled city change, not broad geographic generalization

The logical type *District* is used only for administrative district polygons in the experimental snapshots. It does not represent neighborhoods, residential or commercial districts, mixed-use areas, shopping zones, parcels, building footprints, or complete land-use classes. Likewise, schools and hospitals are represented as service entities with selected attributes such as school type, hospital level, and district label; they are not treated as a complete building ontology. Building-level layers, land-use classifications, and urban-functional-zone semantics are therefore outside the current benchmark and are left as future extensions.

4.1.1. Public Benchmark Context

Before constructing the benchmark, we examined whether existing public datasets could serve as the main experimental testbed. They are useful references, but none instantiates the same executable Chinese urban PostGIS setting. GeoQuery is an important geography semantic-parsing benchmark, but it is small, English-centered, and not a modern urban spatial database benchmark [11]. General Text-to-SQL datasets such as Spider and BIRD provide broad SQL evaluation, but they do not systematically test geometry columns, spatial predicates, distance ranking, or city-entity grounding [15,28]. GeoQuestions1089 provides public question–GeoSPARQL/SPARQL–answer triples, including noisy questions, yet it targets linked geospatial data rather than relational PostGIS execution [38]. The Earth Observation GeoSQL Query Set is now publicly downloadable and is closer in execution form because it provides natural-language questions and expected PostGIS SQL [39]. However, it targets an Earth-observation simulation schema rather than Chinese urban entities, aliases, and the city-layer grounding problem studied here. We therefore retain it as a contextual public resource rather than present it as a directly comparable testbed or add a post hoc experiment. Table 5 summarizes this positioning.

Table 5. Relation between public datasets and the benchmark setting in this paper. The table explains why the paper uses newly constructed controlled benchmarks instead of treating an existing resource as a directly comparable main testbed.

Dataset or Resource	Language/ Query Form	Execution Backend	Matches Chinese Urban Grounding?	Use in This Paper
GeoQuery	English/ logical forms	Small geography database	No	Related benchmark reference
Spider/BIRD	Mostly general NL-to-SQL	Relational SQL	No spatial grounding focus	General Text-to-SQL reference
GeoQuestions1089	Natural language/ GeoSPARQL	Linked geospatial data	Partial noisy-question relevance	Auxiliary reference, not main benchmark
Earth Observation GeoSQL Query Set	English/spatial SQL	PostGIS simulation schema	No; different domain and schema	Public contextual reference; not a direct testbed
NJ-GeoIR-700/ WH-GeoIR-700	Chinese/ GeoIR and SQL	Frozen PostGIS city databases	Yes	Main controlled benchmarks

The new benchmarks are therefore not intended to replace broad Text-to-SQL or GeoSPARQL resources. Their role is narrower: they provide a controlled, executable setting in which Chinese urban aliases, POI layers, topological predicates, metric constraints, nearest/farthest queries, and deterministic PostGIS compilation can be evaluated together. The benchmark artifacts needed to reproduce the reported evaluation, including questions, task labels, gold GeoIR annotations, grounding records, gold SQL, gold answers, schema snapshots, prompt/configuration files, and evaluation scripts, will be made available for peer review on reasonable request and are intended for public release through a project repository or archival record, subject to data-licensing constraints for the underlying spatial layers.

4.1.2. NJ-GeoIR-700 Benchmark

We constructed NJ-GeoIR-700, a benchmark with 700 Chinese spatial questions. It contains 175 development questions and 525 test questions. The dataset is not used to fine-tune model parameters; the development set is used only to determine prompts, GeoIR schema, and local rules. The final evaluation is performed on the test set. The benchmark intentionally focuses on database-answerable questions within the supported spatial

coverage, layers, and attributes; out-of-scope, unsupported, or severely underspecified inputs are treated as a deployment concern rather than part of the current evaluation. It should therefore be read as a controlled, execution-grounded benchmark for mechanism evaluation rather than as a large-scale open-domain language benchmark.

The benchmark covers seven task types:

- **T1 Entity retrieval:** listing or counting entities within a district.
- **T2 Topological relation:** relations such as within, intersects, and crosses.
- **T3 Distance query:** entities within a distance threshold from an anchor.
- **T4 Nearest/farthest:** nearest-neighbor and distance ranking queries.
- **T5 Aggregation:** counting or comparing entities across districts.
- **T6 Compositional constraint:** multiple spatial and attribute constraints.
- **T7 Alias/noisy mention:** abbreviated, normalized, or alias-based mentions.

Table 6 gives representative Chinese spatial query cases selected from the NJ-GeoIR-700 test split for the seven task types. English translations are provided only for readability; the benchmark inputs and evaluation remain in Chinese. These benchmark records explain the intended relation between user wording, GeoIR intent, required layers, and executable spatial operations, rather than serving as additional case-study experiments or production-query logs.

Table 6. Representative Chinese spatial query cases selected from the NJ-GeoIR-700 test split, with English translations provided for readability. The evaluated inputs are the original Chinese questions. The examples show the intended GeoIR relation or intent, required layer types, and expected database operation; they are not additional experimental results.

Task	Representative Question (Chinese/English)	GeoIR Relation/ Intent	Required Layers	Expected Operation
T1	“秦淮区的河流数量是多少” <i>How many rivers are in Qinhuai District?</i>	District-based count	River layer with district attribute	Count retained river records whose district attribute is “秦淮区”
T2	“天妃宫小学位于哪个区” <i>Which district is Tianfeigong Primary School located in?</i>	Containment relation	District, school	Resolve the school anchor and apply a polygon–point containment predicate
T3	“距离鼓楼不超过1500 米的公园有哪些” <i>Which parks are within 1500 m of Gulou?</i>	Metric distance threshold	Park, subway station	Normalize the distance to meters and apply a distance-within predicate
T4	“在玄武区范围内，离九华山公园最近的地铁站是什么” <i>Within Xuanwu District, which subway station is nearest to Jiuhuashan Park?</i>	Nearest-neighbor ranking with district filter	Subway station, park	Filter candidate stations by district, compute distances to the park anchor, and return the top-1 entity
T5	“哪个区的河流最多” <i>Which district has the most rivers?</i>	Grouped aggregation and ranking	River layer with district attribute	Count rivers by district, sort counts descending, and return the top district
T6	“玄武区范围内，离玄武门1500米内的区域公园数量是多少” <i>Within Xuanwu District, how many regional parks are within 1500 m of Xuanwumen?</i>	Attribute and spatial conjunction	Park, subway station	Combine district and park-type filters with a distance-within predicate from the station anchor
T7	“秦淮有哪些学校” <i>What schools are there in Qinhuai?</i>	Alias grounding with entity retrieval	Alias records, school layer with district attribute	Normalize the district alias to “秦淮区” and retrieve matching school records

Table 7 maps these task families to practical urban GIS information needs and their current limitations. The mapping is intended to explain what each controlled task approximates, not to claim that the benchmark is sampled from real production query logs or represents real query-frequency distributions. The benchmark is balanced by design so that mechanisms can be compared across retrieval, topology, distance, ranking, aggregation, composition, and grounding conditions.

Table 7. Practical interpretation and limitations of the controlled benchmark tasks. The table maps each balanced task family to an urban GIS information need; it is not evidence of production-query frequency or open-domain coverage.

Task	Representative Urban GIS Need	Spatial/Database Operation	Included Layers	Current Limitation
T1	Find or count mapped facilities and features within an administrative district	Attribute filtering and district membership	District, hospital, park, school, road, river, subway station	Limited to retained layers and administrative district polygons
T2	Ask whether mapped features relate spatially, such as crossing or containment	Topological predicates over point, line, and polygon geometries	District, road, river, park, and service/reference entities	Does not cover informal place regions or land-use semantics
T3	Retrieve entities within a specified distance from an anchor	Metric threshold query after unit normalization	Hospitals, schools, parks, subway stations, roads, rivers, districts	Euclidean/geometric proximity only; no travel-time or route-network reasoning
T4	Identify nearest or farthest mapped entities from an anchor	Distance ranking and top- <i>k</i> selection	Service/reference entities and supported anchors	Does not model route choice, accessibility, or temporal congestion
T5	Summarize counts or compare quantities across districts	Aggregation and grouping over retained entities	District plus retained entity layers	Reports snapshot-level counts, not demographic or official service-capacity statistics
T6	Combine attribute and spatial requirements in one query	Conjunctive filtering with spatial constraints	Supported attributes and retained spatial layers	Limited to predefined attributes and supported relation combinations
T7	Resolve abbreviated, noisy, or alias-based place mentions	Entity grounding and normalization	Alias records for retained entities	Tests curated aliases only; does not cover unrestricted colloquial or multi-turn references

The benchmark is balanced across the seven task types, with 25 development examples and 75 test examples for each task. Each question is annotated with gold GeoIR, grounding information, executable SQL, and the answer obtained from the frozen database.

Because of this design, the benchmark is best interpreted as a controlled, executable, balanced-by-design testbed for mechanism-level comparison. It underrepresents unconstrained open-domain questions, multi-turn clarification, temporal queries, route planning, raster and remote-sensing operations, building-level reasoning, land-use or urban-functional-zone semantics, and severely underspecified questions that would require abstention or user clarification. No additional human-authored query audit is introduced in this revision because the repository does not contain a separate, independently authored, fully annotated, and execution-verified query set.

4.1.3. Question Construction and Annotation

The benchmark is constructed from query skeletons and database-driven instantiation. We first define canonical spatial query patterns, such as district-based entity retrieval, entities within a distance threshold from an anchor, roads or rivers crossing a district,

nearest entities to an anchor, district-level aggregation, and multi-constraint composition. Database entities are then sampled from the Nanjing spatial database to instantiate these skeletons. This approach ensures that each question is executable against the frozen database and that the gold answer can be obtained from SQL execution.

To avoid making the benchmark purely template-like, questions are rewritten into multiple Chinese surface forms. For example, distance queries may be expressed with “距离 A 不超过 1000 米”, “A 周边 1 公里”, or “离 A 1 公里内”. The alias/noisy-mention subset deliberately includes abbreviated district names, station suffix variants, and normalized mentions. These examples test whether the grounding module can bridge natural-language mentions and database canonical names.

Each example contains a natural-language question, task type, gold semantic GeoIR, gold grounding records, gold SQL, and gold answer. The gold answer is not manually invented; it is produced by executing the gold SQL over the frozen database. This avoids inconsistencies between annotations and the database state. The development split is used to tune prompts, schema descriptions, and local rules, while the test split is kept fixed for final reporting.

The benchmark skeletons, gold GeoIR/SQL annotations, and compiler templates were developed for the same controlled executable setting. They share the same logical entity inventory, relation inventory, schema mapping assumptions, and supported PostGIS operation families. This relationship is useful for ensuring that every benchmark question can be executed and audited, but it also means that the benchmark should not be treated as an independent sample of production user questions or as evidence that the compiler templates cover unconstrained spatial query language. We therefore interpret the results as system comparisons within a controlled, mechanism-oriented benchmark, not as a claim of real-world query-distribution representativeness.

4.1.4. Automatic Quality Control

The automatic quality-control procedure is part of benchmark construction, not the runtime verifier. Its purpose is to detect structural, binding-consistency, and execution problems before evaluation. Structural validation checks unique identifiers, valid split and task labels, and required non-empty semantic-representation and answer fields. Schema-consistency checks cover answer types, entity–table agreement, relation inventories, and agreement between the semantic target entity type and the grounded execution object. The latter agreement is also the available grounding-validity check for the recorded gold grounding information; it does not establish that the natural-language interpretation is semantically correct. Execution validation requires the gold SQL to run against the frozen database snapshot, from which the gold answer is derived.

The same procedure enforces the existing balanced split: 25 development examples and 75 test examples for each of the seven task types, for 700 questions in total. Duplicate diagnostics report no exact or normalized duplicate question groups. The evaluation script also retains a no-overlap guard: if a development question exactly overlaps with the target test split, it is removed from the few-shot demonstration pool before prompts are constructed. Answer-equivalence checking has a separate role during evaluation: it compares predicted and gold outputs using the task-aware rules defined in the evaluation metrics, rather than validating benchmark construction or certifying an annotation.

The dataset validation script regenerates a Markdown/JavaScript Object Notation (JSON) quality report and a stratified manual-audit spreadsheet artifact. The report includes the existing split-balance, required-field, answer-type, entity–table, relation-inventory, mention-form, surface-marker, and duplicate diagnostics. The spreadsheet supports targeted manual review; its presence does not imply that every sample received exhaustive human adjudication. These automatic checks reduce detectable construction and execution

inconsistencies, but they do not prove semantic correctness, rule out all annotation errors, certify the benchmark, or establish open-domain linguistic quality. Table 8 therefore reports diagnostics for the controlled, executable benchmark rather than evidence that it matches production user traffic.

Table 8. Automatic dataset-validity diagnostics for the controlled benchmarks.

Diagnostic	NJ-GeoIR-700	WH-GeoIR-700
Total examples	700	700
Development/test split	175/525	175/525
Test examples per task type	75	75
Structural validation errors	0	0
Exact duplicate groups	0	0
Normalized duplicate groups	0	0
Manual audit pack	10 examples per task	10 examples per task

To test whether the conclusions survive one controlled change of city and database snapshot, we further construct WH-GeoIR-700 over a benchmark-oriented Wuhan database derived from cleaned OSM layers and additional local normalization. The Wuhan database contains 13 districts, 55 hospitals, 55 parks, 20 rivers, 160 roads, 90 schools, and 50 subway stations, for 443 core entities in total. Unlike a full-scale municipal OSM snapshot, this aligned benchmark is intentionally compact: road, river, and park names are de-duplicated, long-tail geometry noise is reduced, and the final target-entity and relation quotas are rebalanced to match the Nanjing benchmark structure. The resulting WH-GeoIR-700 keeps the same 700-question, 175/525 split design and the same T1–T7 task inventory, so it can be used as a controlled cross-city validation benchmark rather than as a new large-scale dataset with a different task definition.

The Wuhan panel in Figure 4 visualizes the aligned benchmark database in the same layer grammar as the Nanjing panel. It is not intended to introduce a second primary study area; its role is to show that the controlled cross-city validation benchmark retains heterogeneous point, line, and polygon layers after benchmark-oriented compaction and de-duplication.

4.2. Settings, Baselines, and Metrics

4.2.1. Models and Settings

All seven backbones are served through the same local inference interface. They cover four model families:

- Qwen2.5: qwen2.5:7b, qwen2.5:14b, qwen2.5-coder:7b, and qwen2.5-coder:14b;
- DeepSeek-Coder-V2-Lite: deepseek-coder-v2:16b;
- Mistral: mistral-small3.2;
- GLM4: glm4:9b-chat-fp16.

The model set is intentionally not a balanced model-family benchmark. Qwen2.5 is over-represented because it provides a convenient paired control over two factors that are relevant to this task: general-purpose versus code-specialized variants, and 7B versus 14B parameter scales. DeepSeek-Coder-V2-Lite, Mistral Small 3.2, and GLM4 are included to check whether the architectural pattern also appears outside the Qwen family. Therefore, the model comparison should be read as a fixed local-backbone system evaluation, not as a ranking of model families. All experiments use deterministic few-shot prompting and no model fine-tuning. We report Nanjing as the primary benchmark and Wuhan as a cross-city aligned benchmark with the same task inventory but a different database snapshot.

All model calls use deterministic decoding with temperature 0. The GeoIR parser uses a maximum generation length of 256 tokens, while SQL-first baselines use 512 tokens

because they must emit complete executable SQL. The fixed few-shot setting uses one development-set example per task type after excluding development examples whose question text exactly overlaps with the target test split. For Nanjing, the DAIL-style baseline retrieves up to three in-domain demonstrations under the same no-overlap rule.

For Wuhan, preliminary development-set checks showed that unconstrained retrieval frequently returned surface-similar but semantically mismatched examples, especially for topological and aggregation queries. We therefore use a guarded DAIL configuration on Wuhan: retrieval is enabled only for nearest/farthest and near-count patterns, and low-confidence retrieval falls back to the same fixed few-shot prompt as Direct-SQL. This guarded policy was fixed before final test scoring and was not selected by inspecting test answers. All runs use the same prompt files, logical schema card, endpoint interface, and database snapshot for the corresponding city. The experiments were executed through local Linux endpoints with a server-local PostGIS database. They are therefore few-shot system evaluations rather than supervised model training. This setting is appropriate for the paper’s goal: evaluating whether spatially explicit intermediate representation, grounding, and PostGIS compilation improve spatial database access when the underlying LLM is kept fixed.

Table 9 reports only the reproducibility settings included in the manuscript. Exact public artifact identifiers, model repository revisions, serving-stack versions beyond the listed local endpoint description, random seeds beyond deterministic decoding, run manifests, and benchmark-package access paths require author confirmation before they can be reported in a release or response package.

Table 9. Experimental environment and reproducibility settings.

Item	Setting
Operating system	CentOS Linux 7 (Core), Linux 3.10 x86_64
Graphics processing unit (GPU)	NVIDIA RTX A5000, 24 gigabytes (GB) of memory
Database engine	PostgreSQL 13.16 with PostGIS 3.2.5
GPU driver	NVIDIA driver 550.163.01
Database snapshots	Frozen server-local PostGIS snapshots for Nanjing and aligned Wuhan
Runtime	Python 3.8.8; local Hypertext Transfer Protocol (HTTP) model-serving endpoints
Decoding	Temperature 0; 256 tokens for GeoIR, 512 tokens for direct SQL baselines
Evaluation splits	525 test questions per city, balanced across seven task types
Shared artifacts	Same prompt files, logical schema card, execution scripts, and city-specific frozen snapshots

4.2.2. Compared Methods

We compare the proposed full pipeline against three SQL-first baselines:

- **Direct-SQL:** the model receives the question, a logical schema card, and a physical schema summary and directly generates PostGIS SQL.
- **DAIL-SQL-style:** a retrieval-augmented Direct-SQL baseline with in-context demonstrations inspired by DAIL-SQL. We use it as a stronger prompt-based baseline rather than as a strict reproduction of the original DAIL-SQL system.
- **Linking-SQL:** the model still generates SQL directly, but it additionally receives a compact text-level schema-linking card containing normalized entity mentions and canonical-name candidates, without access to internal resolved identifiers.
- **GeoIR-Full:** the model generates GeoIR; the system then performs grounding, compilation, verification, and execution.

Figure 5 makes the resulting experimental contrast explicit. Direct-SQL maps the question and schema context directly to SQL. DAIL-SQL-style retains this SQL-first path but retrieves in-domain demonstrations before generation. Linking-SQL inserts a text-level schema-linking stage and passes the resulting linking card to the SQL generator. In contrast,

GeoIR-Full asks the model only for semantic GeoIR and delegates database binding, SQL construction, verification, and execution to separate system modules. The figure also distinguishes model-visible inputs from system-owned resources and intermediate artifacts.

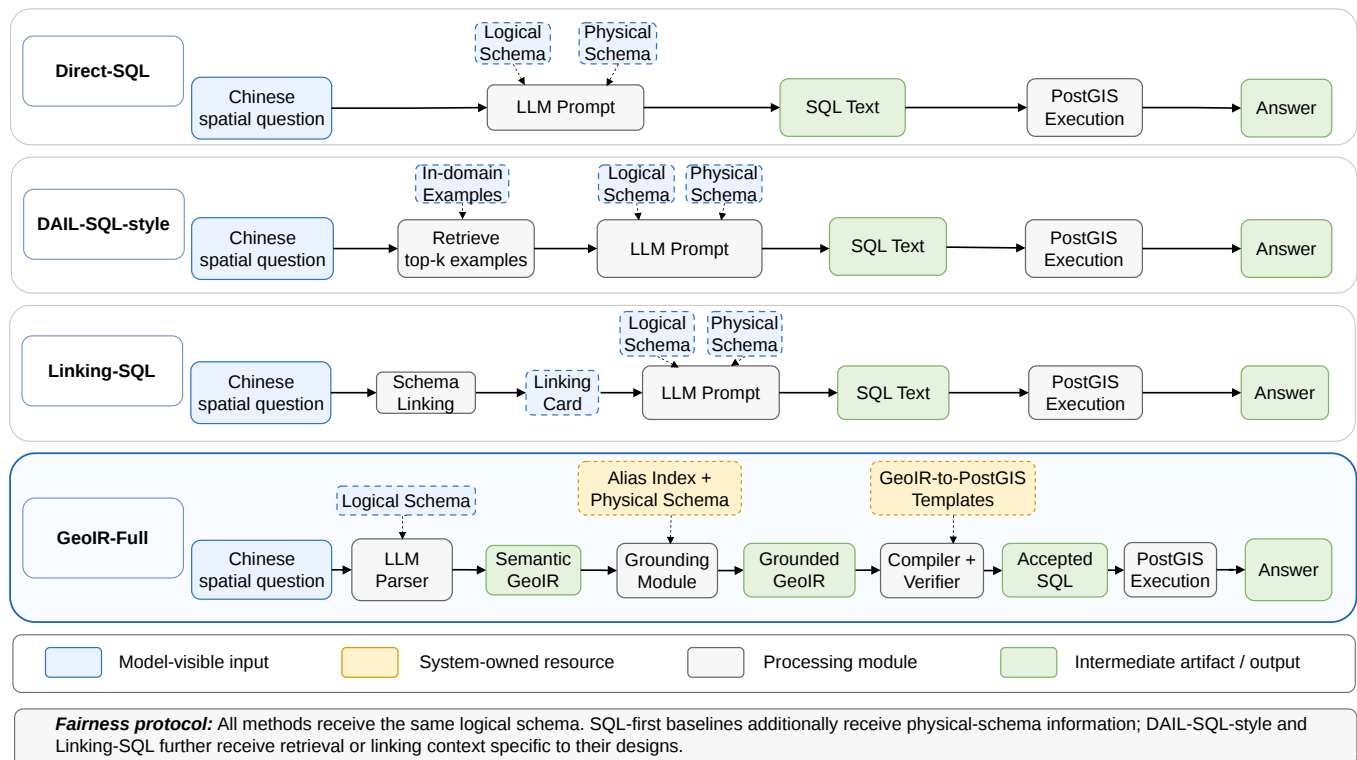


Figure 5. Comparison of the four evaluated methods under the fixed experimental protocol. All methods receive the same Chinese spatial question and logical schema. Direct-SQL generates SQL in one step; DAIL-SQL-style augments direct generation with retrieved in-domain examples; Linking-SQL supplies a text-level linking card; and GeoIR-Full separates semantic parsing, grounding, deterministic compilation, verification, and execution. Dashed arrows indicate supporting inputs or system-owned resources, while solid arrows show the main processing and artifact flow.

The SQL-first baselines are intentionally strong in terms of information exposure: they receive both logical schema information and a physical schema summary, giving them access to table names, important columns, geometry fields, and typical PostGIS functions. DAIL-SQL-style and Linking-SQL further receive retrieval or linking context specific to their designs. The model component of GeoIR-Full, by contrast, receives the logical schema but not the physical schema; physical database knowledge remains inside the grounding and compilation modules. The comparison should therefore be read as a system-level evaluation under a fixed deployment protocol rather than as a claim that all methods use identical internal resources.

We also conduct a focused ablation study:

- **w/o explicit grounding:** disables the type-constrained grounding module and relies on fallback name matching. This is therefore not a no-name-matching baseline.
- **w/o verifier:** disables the conservative verifier and executes more compiled SQL queries.

Alias-table and type-check variants are outside the current reported ablation table, so the discussion below is limited to the two displayed component removals.

4.2.3. Evaluation Metrics

We report execution accuracy, grounding accuracy, coverage, accuracy under coverage (Acc.@Cov.), and executable SQL rate. Following common Text-to-SQL evaluation practice in benchmarks such as Spider [15] and BIRD [28], execution accuracy in Equation (2) is used as the main end-task metric and is labeled Exec. Acc. in compact tables. Grounding accuracy in Equation (4) is the key mechanism-level diagnostic for the proposed architecture, and Acc.@Cov. in Equation (3) measures answer reliability among answered questions. In the main text, we emphasize only these three core metrics. SQL Rate in Appendix A is retained as the reported coverage/executability diagnostic under the fixed protocol; no standalone latency table or latency comparison is reported.

Let D denote the test set, $N = |D|$, and $i \in \{1, \dots, N\}$ index a question. Let a_i be the predicted answer, with $a_i = \perp$ when no answer is returned, and let a_i^* be the gold answer. Define $e_i = \text{Eq}(a_i, a_i^*) \in \{0, 1\}$, where Eq is the task-aware answer-equivalence predicate. Boolean and non-numeric scalar answers require exact equality. Numeric scalars use `math.isclose` with relative and absolute tolerances of 10^{-9} . Entity-list outputs are converted to names, de-duplicated, lexically sorted, and compared as order-insensitive normalized collections. For nearest/farthest queries, ordering and limiting determine which rows enter the answer before this collection normalization; database row order is not scored as a separate metric. An unanswered case has $e_i = 0$. Execution accuracy is

$$\text{ExecAcc} = \frac{1}{N} \sum_{i=1}^N e_i. \quad (2)$$

Let $c_i \in \{0, 1\}$ indicate whether the system returns an executable covered answer and let $M = \sum_{i=1}^N c_i$ be the number of covered questions. Coverage is the covered fraction, and accuracy under coverage is correctness conditional on coverage:

$$\text{Coverage} = \frac{1}{N} \sum_{i=1}^N c_i, \quad \text{Acc@Cov} = \begin{cases} \frac{\sum_{i=1}^N c_i e_i}{M}, & M > 0, \\ \text{undefined (not reported)}, & M = 0. \end{cases} \quad (3)$$

For GeoIR-based methods, grounding accuracy is computed as strict sample-level matching over database entity bindings. For question i , let $\mathcal{P}_i$ and $\mathcal{G}_i$ denote the sets of (entity type, resolved identifier) pairs extracted from the predicted and gold grounding records, respectively. Records without a resolved database identifier contribute no pair. Thus, when the gold record contains a required resolved pair, an unresolved prediction omits that pair and fails the set-equality test. Then

$$\text{Grounding} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\mathcal{P}_i = \mathcal{G}_i). \quad (4)$$

Thus, a question is counted as grounding-correct only when the complete predicted binding set equals the gold binding set; partial entity matches are not credited, and extra bindings also cause failure. Direct-SQL baselines do not expose explicit grounding records, so this metric is reported only for GeoIR-based methods. The executable SQL rate is retained only as an auxiliary executability diagnostic. In the fixed evaluation protocol used here, SQL Rate is the same final covered-execution event as Coverage: it indicates whether the system produced SQL or a final executable answer that entered answer-equivalence scoring, but not whether that answer preserves the intended spatial meaning. We do not emphasize GeoIR slot F1 score (the harmonic mean of precision and recall) in the main results because the paper's central claim concerns database-level correctness and reliability rather than parser-slot similarity alone.

5. Results and Discussion

We organize the analysis around three research questions (RQs). RQ1 asks whether a spatially structured GeoIR-grounding-compilation pipeline improves final execution accuracy over direct SQL generation. RQ2 asks whether a retrieval-enhanced direct-SQL prompt can close this gap when it receives both schema information and in-domain demonstrations. RQ3 asks which components of the proposed pipeline account for the observed reliability gains. This organization keeps the empirical discussion aligned with the paper’s core claim: the main difficulty is preserving spatial meaning through database grounding and execution, not merely producing syntactically runnable SQL.

5.1. Main Results

Table 10 presents the primary Nanjing comparison across all seven backbones and all four methods. These results test whether a spatially structured pipeline can reduce semantic–execution mismatch under the fixed benchmark protocol. Across all seven backbones, GeoIR-Full reaches a mean execution accuracy of 0.7271, compared with 0.4196 for Direct-SQL, 0.5399 for DAIL-SQL-style, and 0.5135 for Linking-SQL. The same pattern holds for reliability: mean Acc.@Cov. rises from 0.5008, 0.5598, and 0.5664 to 0.8950. GeoIR-Full beats all three SQL-first baselines on all evaluated backbones. Within the Qwen2.5 paired control, the gain appears for both 7B and 14B models and for both general-purpose and code-specialized variants; on the three non-Qwen backbones, the same qualitative advantage remains. This pattern is consistent with the staged pipeline explanation, while still not making the model pool a balanced model-family benchmark or isolating a single component as the only cause.

Table 10. Compact main results on the primary Nanjing benchmark across seven backbones. For space, we show only the three core metrics emphasized in the paper: end-task correctness (Exec. Acc.), mechanism-level alignment (Grounding), and answer reliability (Acc.@Cov.). Auxiliary diagnostics appear in Appendix A. Bold values mark the best value within each column group.

Model	Exec. Acc.				GeoIR Grounding	Acc.@Cov.			
	Direct	DAIL	Link	GeoIR		Direct	DAIL	Link	GeoIR
qwen2.5:7b	0.4210	0.5352	0.4533	0.6705	0.7257	0.4966	0.5575	0.5085	0.8934
qwen2.5:14b	0.4324	0.5371	0.4743	0.7714	0.7867	0.4935	0.5476	0.4931	0.9353
qwen2.5-coder:7b	0.4000	0.5257	0.4876	0.7905	0.7981	0.4646	0.5349	0.5151	0.9161
qwen2.5-coder:14b	0.4000	0.5410	0.5029	0.7505	0.7943	0.4497	0.5504	0.5432	0.9314
deepseek-coder-v2:16b	0.4152	0.5429	0.5867	0.6800	0.7848	0.5782	0.5610	0.6754	0.7727
mistral-small13.2	0.4819	0.5733	0.5600	0.7524	0.7543	0.4932	0.5845	0.5611	0.9611
glm4:9b-chat-fp16	0.3867	0.5238	0.5295	0.6743	0.6990	0.5300	0.5826	0.6683	0.8551

Figure 6 visualizes the same execution-accuracy comparison. The consistent separation between GeoIR-Full and the three SQL-first baselines indicates that the observed gain appears across the tested backbones rather than being driven by a single favorable model.

Among the SQL-first baselines, the best Direct-SQL and DAIL-SQL-style runs both use mistral-small13.2, reaching 0.4819 and 0.5733, respectively. Linking-SQL performs best with deepseek-coder-v2:16b, reaching 0.5867. All three values remain below the best GeoIR-Full run, 0.7905 with qwen2.5-coder:7b. This matters because the strongest SQL-first baselines come from non-Qwen backbones, while the strongest full pipeline comes from a Qwen code-specialized backbone. The comparison should therefore focus on the method gap within each backbone rather than on which family produces the single best score. The SQL-first baselines are already strong in terms of prompt exposure: they see both logical and physical schema information, DAIL-style prompting receives retrieved

demonstrations, and Linking-SQL additionally receives compact entity-normalization hints. The remaining gap is therefore not simply a comparison against under-informed prompts. It is consistent with the value of changing the system structure: the LLM predicts a constrained spatial meaning, while database binding and spatial-function realization are handled by explicit local modules. At the same time, the best execution accuracy remains below 0.8 on the primary benchmark, so the result should not be interpreted as solving Chinese urban spatial QA. Its significance is the consistent reduction in semantic-execution mismatch across the evaluated local backbones and baselines.

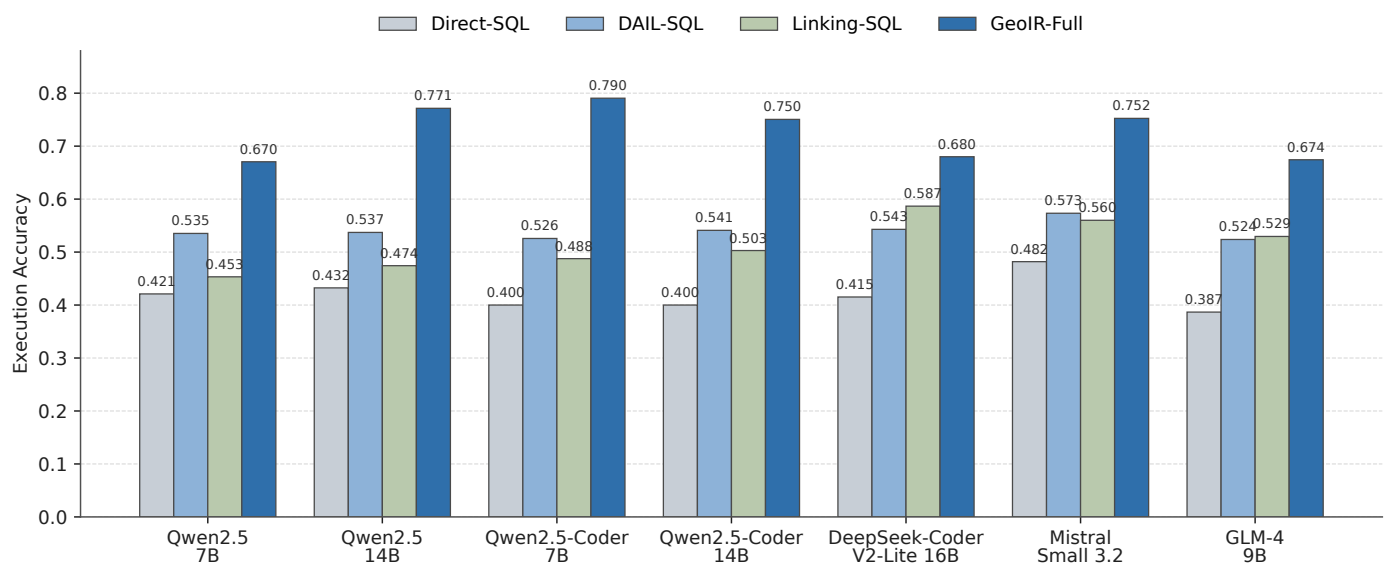


Figure 6. Execution-accuracy comparison on the primary Nanjing benchmark across seven backbones. GeoIR-Full consistently outperforms Direct-SQL, DAIL-SQL-style, and Linking-SQL under the fixed protocol, a pattern consistent with a stable system-level advantage from separating semantic parsing, grounding, and deterministic compilation.

What stands out here is not executability but semantic control. Table A1 in Appendix A shows that the SQL-first baselines often still achieve executable SQL rates above 0.90, yet their execution accuracy and Acc.@Cov. remain far below those of GeoIR-Full. In other words, direct generation often produces SQL that runs while missing the intended spatial meaning. In GeoIR-Full, the failure path is easier to inspect because semantic parsing, grounding, and SQL construction are separate stages; errors can therefore be traced more directly to grounding or compilation decisions rather than to an opaque end-to-end generation step.

5.2. Per-Task Analysis on Nanjing

Table 11 reports a per-task breakdown on the primary Nanjing benchmark. The values average the seven paper backbones for all four compared methods and are taken from the consolidated task-level summary report. The table is intended as a mechanism diagnostic rather than as an additional benchmark headline.

The per-task breakdown also makes the system pattern more concrete. T1 and T5 are already manageable for SQL-first methods because they often collapse to district filters or scalar aggregation. Linking-SQL narrows the gap on topological relations, distance queries, and alias-heavy mentions, which suggests that explicit normalization hints help direct SQL generation. Even so, the largest gains still appear on nearest/farthest queries and noisy references. Those are exactly the settings in which direct SQL generation must infer a spatial operation, bind a reference entity, choose the right distance or ranking template, and still emit executable SQL in one pass. GeoIR-Full spreads those decisions across stages: the model

predicts the logical relation and entity type, grounding resolves the mention, and the compiler applies the deterministic PostGIS template. Topological and distance reasoning remain the weakest areas, however, so the improvement is substantial rather than complete.

Table 11. Per-task execution accuracy on NJ-GeoIR-700 averaged over the seven evaluated backbones. Values are taken from the consolidated task-level summary report. Bold values mark the best method for each task.

Task	Description	Direct-SQL	DAIL-SQL	Linking-SQL	GeoIR-Full
T1	District entity retrieval	0.9867	0.9867	0.9733	0.9943
T2	Topological relation	0.1657	0.2857	0.4133	0.4400
T3	Distance query	0.2914	0.3886	0.4800	0.5200
T4	Nearest/farthest	0.0857	0.2152	0.1581	0.7200
T5	Aggregation	0.8210	0.8648	0.8095	0.8743
T6	Compositional constraint	0.2381	0.4971	0.2590	0.6133
T7	Alias/noisy mention	0.3486	0.5410	0.5010	0.9276

5.3. Cross-City Validation on Wuhan

To test whether the main conclusion survives a change in city and database snapshot, we evaluate the same four methods on the aligned Wuhan benchmark WH-GeoIR-700. Table 12 reports the method-level averages and best runs across the same seven backbones. The headline result is straightforward: GeoIR-Full remains the strongest method, with a mean execution accuracy of 0.7363 and a mean Acc.@Cov. of 0.8981. Its best Wuhan run, qwen2.5-coder:7b at 0.8267, is slightly higher than the best Nanjing run.

The cross-city comparison is useful for two reasons. First, GeoIR-Full is relatively stable in this aligned validation setting: the mean moves from 0.7271 on Nanjing to 0.7363 on Wuhan, and mean Acc.@Cov. shifts from 0.8950 to 0.8981. This supports only a controlled aligned Nanjing–Wuhan validation claim, while remaining far from broad open-domain geographic generalization. Second, the SQL-first baselines respond differently to the city change. Direct-SQL drops from 0.4196 to 0.3954, and Linking-SQL drops from 0.5135 to 0.4590, but both remain interpretable baselines. DAIL-SQL-style is the least stable of the three: on Nanjing, ordinary retrieval prompting transfers reasonably well, whereas on Wuhan ungated retrieval often returns examples that are surface-similar but semantically mismatched. The guarded Wuhan DAIL configuration removes the clearest failure mode, but its seven-model mean still reaches only 0.4343, below both Linking-SQL and GeoIR-Full.

The Wuhan results also sharpen the interpretation of the tested configurations. Linking-SQL shows that explicit entity-normalization hints help SQL-first generation, especially on models such as glm4:9b-chat-fp16; however, this help is still limited because Linking-SQL does not resolve the harder question of how spatial meaning should be structurally represented and compiled. Guarded DAIL-SQL shows that a retrieval baseline is configuration-dependent: demonstration use must be restricted to patterns with genuinely transferable SQL structure in this aligned Wuhan setting. Nevertheless, neither approach closes the gap to GeoIR-Full. The cross-city experiment therefore supports the paper’s main argument only within a controlled two-city setting: the tested staged pipeline remains stronger after the database content and alias inventory are changed and revalidated.

Table 12. Cross-city validation summary on the aligned Wuhan benchmark. DAIL-SQL-style uses the guarded Wuhan configuration described in Section 4. Bold values mark the best method within each city block.

City	Method	Mean Exec. Acc.	Mean Acc.@Cov.	Best Model	Best Exec. Acc.
Nanjing	Direct-SQL	0.4196	0.5008	mistral-small13.2	0.4819
Nanjing	DAIL-SQL-style	0.5399	0.5598	mistral-small13.2	0.5733
Nanjing	Linking-SQL	0.5135	0.5664	deepseek-coder-v2:16b	0.5867
Nanjing	GeoIR-Full	0.7271	0.8950	qwen2.5-coder:7b	0.7905
Wuhan	Direct-SQL	0.3954	0.4358	qwen2.5:14b	0.4438
Wuhan	DAIL-SQL-style	0.4343	0.4532	qwen2.5:7b	0.4648
Wuhan	Linking-SQL	0.4590	0.4991	glm4:9b-chat-fp16	0.5829
Wuhan	GeoIR-Full	0.7363	0.8981	qwen2.5-coder:7b	0.8267

Transferability Boundaries

The Wuhan experiment should be read as controlled aligned cross-city validation, not as proof of broad geographic generalization. The reusable and city-specific parts of the approach differ by transfer setting. For another city, the GeoIR ontology, compiler structure, and evaluation protocol are expected to remain useful, but the database snapshot, aliases, entity identifiers, logical-to-physical schema bindings, and quality checks must be rebuilt or revalidated. For another language, the typed GeoIR and deterministic compiler can remain the target interface, but parser prompts, tokenization behavior, aliases, normalization rules, and multilingual grounding resources would need adaptation. For another database schema, the semantic representation can still separate intent, targets, spatial relations, aggregation, ranking, and output form. However, the geometry columns, identifiers, schema mappings, SQL templates, and backend-specific functions must be adapted. For another geospatial domain, the current evidence is limited to vector urban data; cadastral, environmental, mobility, temporal, network, raster, or Earth-observation tasks would require new entity inventories, relation types, and compiler operations before comparable claims could be made.

A stronger future evaluation could retain Nanjing as the primary city and select multiple cross-city cases through a predefined urban typology. Candidate stratification dimensions include resident population and administrative area, administrative structure, spatial morphology, and dominant urban functions. Each cross-city benchmark should retain the same T1–T7 task inventory, question quotas, and evaluation metrics, while rebuilding or revalidating its database snapshot, alias resources, and schema bindings. Reporting variation both within and across typology groups would help distinguish sensitivity to city type from differences introduced by benchmark construction. The present Wuhan benchmark is only one aligned cross-city case and does not constitute such a typology-stratified multi-city evaluation.

Table 13 gives the corresponding per-task breakdown on WH-GeoIR-700. The Wuhan pattern is consistent with the Nanjing diagnostic but should still be read within the controlled aligned benchmark. SQL-first methods remain strong on simple district retrieval and aggregation, while their accuracy drops on distance, nearest-neighbor, compositional, and alias-heavy queries. GeoIR-Full preserves high accuracy on T4 and T7 and improves substantially on T3 and T6, which is consistent with the interpretation that separating spatial meaning, grounding, and database binding is useful when these choices cannot be safely collapsed into one SQL-generation step.

Table 13. Per-task execution accuracy on WH-GeoIR-700 averaged over the seven evaluated backbones. DAIL-SQL uses the guarded Wuhan configuration reported in Table 12. Bold values mark the best method for each task.

Task	Description	Direct-SQL	DAIL-SQL	Linking-SQL	GeoIR-Full
T1	District entity retrieval	0.9752	0.9714	0.9790	0.9943
T2	Topological relation	0.3429	0.3390	0.3810	0.4095
T3	Distance query	0.2019	0.1829	0.3524	0.6171
T4	Nearest/farthest	0.0933	0.2324	0.1219	0.7010
T5	Aggregation	0.7962	0.8076	0.7714	0.8248
T6	Compositional constraint	0.0971	0.2438	0.1448	0.6610
T7	Alias/noisy mention	0.2610	0.2629	0.4629	0.9467

5.4. Focused Ablation

Table 14 reports a deliberately focused four-model ablation. The first block pairs qwen2.5:14b with qwen2.5-coder:14b; matching the nominal parameter scale controls model size while varying general-purpose versus code-specialized training. The second block uses mistral-small13.2 and deepseek-coder-v2:16b as non-Qwen robustness checks, reducing the risk that the component pattern is specific to one model family. Repeating the same component removals across every backbone would add redundant runs without changing the mechanism question addressed here. Accordingly, this table supports interpretation of grounding and verification effects; it is not a comprehensive ranking of model families and does not imply that a complete seven-model ablation matrix is available.

Table 14. Focused four-model ablation for mechanism interpretation rather than model-family ranking. The 14B Qwen pair controls nominal parameter scale while varying general-purpose versus code specialization; Mistral and DeepSeek provide non-Qwen robustness checks. Deltas are relative to GeoIR-Full for the same model.

Model	Variant	Exec. Acc.	Δ Acc.	Grounding	Acc.@Cov.	Δ Acc.@Cov.
qwen2.5:14b	GeoIR-Full	0.7714	+0.0000	0.7867	0.9353	+0.0000
	w/o Explicit Grounding	0.6914	−0.0800	0.0210	0.7146	−0.2208
	w/o Verifier	0.7867	+0.0152	0.7867	0.8130	−0.1223
qwen2.5-coder:14b	GeoIR-Full	0.7505	+0.0000	0.7943	0.9314	+0.0000
	w/o Explicit Grounding	0.6514	−0.0990	0.0210	0.6590	−0.2725
	w/o Verifier	0.7581	+0.0076	0.7962	0.7669	−0.1646
<i>Cross-backbone robustness check</i>						
mistral-small13.2	GeoIR-Full	0.7524	+0.0000	0.7543	0.9611	+0.0000
	w/o Explicit Grounding	0.6800	−0.0724	0.0210	0.6932	−0.2679
	w/o Verifier	0.7695	+0.0171	0.7562	0.7845	−0.1766
deepseek-coder-v2:16b	GeoIR-Full	0.6800	+0.0000	0.7848	0.7727	+0.0000
	w/o Explicit Grounding	0.6076	−0.0724	0.0210	0.6280	−0.1448
	w/o Verifier	0.6819	+0.0019	0.7848	0.7047	−0.0680

The largest observed degradation in this focused ablation occurs when explicit grounding is removed. Removing the type-constrained grounding module reduces execution accuracy by 8.00 percentage points for qwen2.5:14b and 9.90 percentage points for qwen2.5-coder:14b. More importantly, grounding accuracy collapses from approximately 0.79 to 0.021 for both backbones, and Acc.@Cov. drops by 0.2208 and 0.2725, respectively. This is consistent with the interpretation that the contribution is not merely the existence of an intermediate representation, but also the spatial database binding that connects GeoIR to concrete urban entities and fields before PostGIS compilation. In the w/o Explicit Grounding variant, fallback name matching remains available, so the result should be read

as evidence about removing the explicit type-constrained grounding module rather than about removing every possible name-matching signal.

The verifier shows a different pattern. Removing it does not consistently hurt raw execution accuracy; in fact, it slightly increases Exec. Acc. on both 14B backbones. However, Acc.@Cov. decreases markedly, from 0.9353 to 0.8130 on qwen2.5:14b and from 0.9314 to 0.7669 on qwen2.5-coder:14b. The verifier should therefore not be described as a module that directly maximizes end-task accuracy. Its role is better understood as conservative quality control: it filters uncertain executions and makes answered queries more reliable.

The non-Qwen block points in the same direction. On mistral-small13.2, removing explicit grounding lowers Exec. Acc. from 0.7524 to 0.6800, while removing the verifier slightly raises raw Exec. Acc. to 0.7695 but drops Acc.@Cov. from 0.9611 to 0.7845. On deepseek-coder-v2:16b, removing explicit grounding lowers Exec. Acc. from 0.6800 to 0.6076, and removing the verifier leaves raw accuracy nearly unchanged at 0.6819 while reducing Acc.@Cov. from 0.7727 to 0.7047. Together, the two blocks support the same interpretation under the tested configurations: explicit grounding contributes strongly to the observed gains, and the verifier mainly trades coverage for reliability rather than directly maximizing end-task accuracy.

5.5. Error Diagnostics

Table 15 summarizes automatic error diagnostics for the strongest Nanjing full pipeline. The categories are assigned from logged execution fields rather than from manual adjudication, so they should be read as operational diagnostics. The taxonomy is a first-match rule over wrong answers rather than an exhaustive causal proof: each failed example is counted once under the earliest applicable pipeline state. The rule is applied in this order: unparsable GeoIR; missing or ambiguous grounding that prevents q_g from existing; mismatch between predicted and gold grounding records when q_g exists; compiled-SQL absence, verifier rejection, or execution failure that prevents a covered answer; semantic GeoIR mismatch; and residual answer mismatch. This ordering keeps undefined $\Gamma_D(q_s)$ cases separate from grounding-record mismatches, because no predicted binding record exists when grounding stops on missing or ambiguous candidates.

Table 15. Automatic error diagnostics for qwen2.5-coder:7b GeoIR-Full on NJ-GeoIR-700 (525 test questions).

Diagnostic Category	Count	Share of All Test Questions
Grounding mismatch	97	0.1848
Semantic GeoIR mismatch	11	0.0210
Non-executable/withheld answer	2	0.0038
Total wrong answers	110	0.2095

The diagnostics are consistent with the ablation study: grounding-related failures remain the largest residual category in the best full pipeline. The two cases in the “Non-executable/withheld answer” row have recorded reasons of `ambiguous_grounding` and `execution_error`, respectively. They are grouped by the observed absence of an executable or covered answer among wrong cases and should not be read as both being verifier rejections. The `ambiguous_grounding` case is placed in this row because the grounder produced no q_g , whereas the “Grounding mismatch” row is reserved for cases where predicted and gold binding records can be compared. The pipeline returned no covered answer for 72 queries; this coverage-level count includes upstream non-executable cases, such as ambiguous grounding, as well as post-compilation verifier rejections. It is not an additional mutually exclusive category in Table 15. This separation also explains why removing the

verifier can slightly increase raw execution accuracy while lowering Acc.@Cov.: the verifier governs whether compiled but uncertain cases are answered, whereas grounding failures never produce a grounded representation. Because these categories are derived from execution logs, they should be treated as diagnostic evidence rather than a substitute for human error adjudication. The release version of the benchmark should therefore include a manual audit layer, especially to distinguish true grounding failures from semantically close aliases or underspecified mentions.

6. Conclusions

This study addressed the difficulty of translating Chinese urban spatial questions into executable spatial-database queries when relation interpretation, entity grounding, and database-specific operations are collapsed into one generation step. GeoIR-Compiler instead separates semantic representation, grounding, deterministic compilation, and execution-oriented verification. Semantic GeoIR records the intended target, spatial relation, metric constraint, aggregation, ranking, and output form; grounding binds these elements to the experimental database snapshot; and deterministic templates compile the grounded representation into PostGIS SQL.

Across seven locally deployable backbones, the full pipeline achieved mean execution accuracies of 0.7271 on the primary Nanjing benchmark and 0.7363 on the aligned Wuhan benchmark. The consistent advantage over the evaluated SQL-first comparison approaches supports the staged architecture under the controlled experimental setting. The focused ablation is consistent with explicit grounding contributing strongly to the improvement because it resolves aliases and binds semantic targets to database objects before compilation, while other system resources such as schema mappings, compiler templates, and rejection policies may also contribute. The verifier has a narrower role: its conservative execution-oriented checks improve reliability among answered queries and make rejection or failure analysis more explicit. It neither establishes semantic correctness nor constitutes a proof mechanism.

These findings support controlled urban GIS database access, auditable spatial query generation, and mechanism-level analysis of where interpretation, binding, compilation, or execution fails. They do not show that the framework solves general urban spatial question answering or is ready for unrestricted deployment. The practical contribution is a structured and inspectable route from a Chinese spatial question to an executable query within the schemas, entity layers, relation inventory, and database snapshots evaluated here.

The evidence remains bounded by the benchmark design. The two 700-question benchmarks are controlled, executable, and balanced by task rather than sampled from production query logs, so they do not represent real query-frequency distributions. Nanjing is the primary benchmark, while Wuhan provides controlled and aligned cross-city validation; the Wuhan result is not proof of broad geographic generalization. Although Wuhan has a larger official resident population and administrative area than Nanjing, the experiments evaluate aligned retained database snapshots rather than city-scale strata and therefore do not establish how performance varies with real-city size. The logical type *District* denotes administrative district polygons only. Neighborhood typology, building footprints, residential, commercial, or mixed-use functions, shopping zones, and complete land-use classes are outside the current scope, and schools and hospitals are represented only as service entities with selected attributes. The evaluated model set is also limited to the reported locally deployable backbones and is not evidence of a universal model-family ranking.

Transfer to additional cities requires new or revalidated database snapshots, aliases, identifiers, and schema bindings. Multilingual use requires adapted parsing and grounding resources; cross-schema use requires new physical mappings and compiler templates; and

broader geospatial domains require richer entity, relation, and operation inventories. Future work can therefore examine richer schemas, additional cities, multilingual grounding, and broader geospatial domains. In particular, a typology-stratified design with one primary city and multiple cross-city cases could test generalization across differences in city scale, administrative structure, spatial morphology, and urban function. These directions require separate implementation and evaluation and are not implied by the present two-city results.

Author Contributions: Conceptualization, Chaolin Zhang and Jiqui Deng; methodology, Jiqui Deng; software, Jiqui Deng; validation, Jiqui Deng, Hui Zhang, Liji Sun and Xiao Ma; formal analysis, Jiqui Deng; investigation, Jiqui Deng, Hui Zhang, Liji Sun and Xiao Ma; resources, Chaolin Zhang, Hui Zhang, Longbo Li, Liji Sun and Xiao Ma; data curation, Jiqui Deng and Hui Zhang; writing—original draft preparation, Jiqui Deng; writing—review and editing, Chaolin Zhang, Jiqui Deng, Hui Zhang, Longbo Li, Liji Sun and Xiao Ma; visualization, Jiqui Deng; supervision, Chaolin Zhang; project administration, Chaolin Zhang and Longbo Li; funding acquisition, Chaolin Zhang and Longbo Li. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Science and Technology Program of Hunan Province, grant number 2026AQ2031, and the Natural Resources Science and Technology Project of Hunan Province, grant number 20240131DZ.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The peer-review package is available through an anonymized OSF view-only link provided during review. It includes benchmark questions and task labels, gold GeoIR annotations, grounding records, gold SQL queries, gold answers, schema files, prompts, configurations, evaluation scripts, README files, data dictionaries, and compact database bundles for confidential peer review. Upon acceptance, the releasable benchmark artifacts, scripts, schema files, and documentation will be deposited in a public repository or archival record. Raw spatial layers and restricted database-snapshot attributes will be publicly released only where permitted by the applicable data licenses; otherwise, reconstruction instructions will be provided and accompanied by validation scripts where possible.

Conflicts of Interest: Author Longbo Li was employed by the company Zhaojin Mining Industry Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

Acc.@Cov.	Accuracy under coverage
BIRD	Big Bench for Large-Scale Database Grounded Text-to-SQL Evaluation
DAIL	Data-Augmented In-Context Learning
Exec. Acc.	Execution accuracy
F1	Harmonic mean of precision and recall
GB	Gigabyte
GeoIR	Geospatial Intermediate Representation
GeoSPARQL	Geospatial query standard built on SPARQL
GIS	Geographic Information System
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
IR	Intermediate Representation
JSON	JavaScript Object Notation

LLM	Large Language Model
NL-to-SQL	Natural-language-to-SQL
NLIDB	Natural Language Interface to Database
OSM	OpenStreetMap
POI	Point of interest
PostGIS	Spatial database extension for PostgreSQL
QA	Question Answering
RDF	Resource Description Framework
RQ	Research Question
SPARQL	SPARQL Protocol and Resource Description Framework Query Language
SQL	Structured Query Language

Appendix A. Supplementary Results

Tables A1 and A2 report the full seven-backbone comparisons for Nanjing and Wuhan together with the auxiliary SQL executability diagnostic. The same qualitative pattern as in the main text persists when SQL rate is added: all three SQL-first baselines remain below GeoIR-Full on both execution accuracy and answer reliability. We retain SQL rate in the appendix because SQL executability is a useful diagnostic signal, but it is not the primary success criterion of the paper.

Table A1. Full seven-backbone comparison with auxiliary diagnostics. Bold values mark the best primary metrics within each model block; SQL Rate reports the same covered-execution event as Coverage under the fixed protocol and is not used for best-value emphasis.

Model	Method	Exec. Acc.	Grounding	Acc.@Cov.	SQL Rate
qwen2.5:7b	Direct-SQL	0.4210	–	0.4966	0.8476
qwen2.5:7b	DAIL-SQL-style	0.5352	–	0.5575	0.9600
qwen2.5:7b	Linking-SQL	0.4533	–	0.5085	0.8914
qwen2.5:7b	GeoIR-Full	0.6705	0.7257	0.8934	0.7505
qwen2.5:14b	Direct-SQL	0.4324	–	0.4935	0.8762
qwen2.5:14b	DAIL-SQL-style	0.5371	–	0.5476	0.9810
qwen2.5:14b	Linking-SQL	0.4743	–	0.4931	0.9619
qwen2.5:14b	GeoIR-Full	0.7714	0.7867	0.9353	0.8248
qwen2.5-coder:7b	Direct-SQL	0.4000	–	0.4646	0.8610
qwen2.5-coder:7b	DAIL-SQL-style	0.5257	–	0.5349	0.9829
qwen2.5-coder:7b	Linking-SQL	0.4876	–	0.5151	0.9467
qwen2.5-coder:7b	GeoIR-Full	0.7905	0.7981	0.9161	0.8629
qwen2.5-coder:14b	Direct-SQL	0.4000	–	0.4497	0.8895
qwen2.5-coder:14b	DAIL-SQL-style	0.5410	–	0.5504	0.9829
qwen2.5-coder:14b	Linking-SQL	0.5029	–	0.5432	0.9257
qwen2.5-coder:14b	GeoIR-Full	0.7505	0.7943	0.9314	0.8057
deepseek-coder-v2:16b	Direct-SQL	0.4152	–	0.5782	0.7181
deepseek-coder-v2:16b	DAIL-SQL-style	0.5429	–	0.5610	0.9676
deepseek-coder-v2:16b	Linking-SQL	0.5867	–	0.6754	0.8686
deepseek-coder-v2:16b	GeoIR-Full	0.6800	0.7848	0.7727	0.8800
mistral-small13.2	Direct-SQL	0.4819	–	0.4932	0.9771
mistral-small13.2	DAIL-SQL-style	0.5733	–	0.5845	0.9810
mistral-small13.2	Linking-SQL	0.5600	–	0.5611	0.9981
mistral-small13.2	GeoIR-Full	0.7524	0.7543	0.9611	0.7829
glm4:9b-chat-fp16	Direct-SQL	0.3867	–	0.5300	0.7295
glm4:9b-chat-fp16	DAIL-SQL-style	0.5238	–	0.5826	0.8990
glm4:9b-chat-fp16	Linking-SQL	0.5295	–	0.6683	0.7924
glm4:9b-chat-fp16	GeoIR-Full	0.6743	0.6990	0.8551	0.7886

Table A2. Full seven-backbone comparison on the aligned Wuhan benchmark with auxiliary diagnostics. DAIL-SQL-style uses the guarded Wuhan configuration described in Section 4. Bold values mark the best primary metrics within each model block; SQL Rate reports the same covered-execution event as Coverage under the fixed protocol and is not used for best-value emphasis.

Model	Method	Exec. Acc.	Grounding	Acc.@Cov.	SQL Rate
qwen2.5:7b	Direct-SQL	0.4362	–	0.4673	0.9333
qwen2.5:7b	DAIL-SQL-style	0.4648	–	0.4841	0.9600
qwen2.5:7b	Linking-SQL	0.4857	–	0.5010	0.9695
qwen2.5:7b	GeoIR-Full	0.6895	0.7581	0.9476	0.7276
qwen2.5:14b	Direct-SQL	0.4438	–	0.4968	0.8933
qwen2.5:14b	DAIL-SQL-style	0.4629	–	0.4755	0.9733
qwen2.5:14b	Linking-SQL	0.4419	–	0.4926	0.8971
qwen2.5:14b	GeoIR-Full	0.7790	0.8076	0.9274	0.8400
qwen2.5-coder:7b	Direct-SQL	0.3429	–	0.3822	0.8971
qwen2.5-coder:7b	DAIL-SQL-style	0.3619	–	0.3885	0.9314
qwen2.5-coder:7b	Linking-SQL	0.3810	–	0.4202	0.9067
qwen2.5-coder:7b	GeoIR-Full	0.8267	0.8819	0.9195	0.8990
qwen2.5-coder:14b	Direct-SQL	0.3048	–	0.3493	0.8724
qwen2.5-coder:14b	DAIL-SQL-style	0.4038	–	0.4198	0.9619
qwen2.5-coder:14b	Linking-SQL	0.4000	–	0.4384	0.9124
qwen2.5-coder:14b	GeoIR-Full	0.7714	0.8000	0.9463	0.8152
deepseek-coder-v2:16b	Direct-SQL	0.4133	–	0.4357	0.9486
deepseek-coder-v2:16b	DAIL-SQL-style	0.4552	–	0.4632	0.9829
deepseek-coder-v2:16b	Linking-SQL	0.4305	–	0.4829	0.8914
deepseek-coder-v2:16b	GeoIR-Full	0.7010	0.8248	0.7797	0.8990
mistral-small3.2	Direct-SQL	0.4305	–	0.4388	0.9810
mistral-small3.2	DAIL-SQL-style	0.4514	–	0.4611	0.9790
mistral-small3.2	Linking-SQL	0.4914	–	0.5019	0.9790
mistral-small3.2	GeoIR-Full	0.7981	0.8057	0.9332	0.8552
glm4:9b-chat-fp16	Direct-SQL	0.3962	–	0.4804	0.8248
glm4:9b-chat-fp16	DAIL-SQL-style	0.4400	–	0.4802	0.9162
glm4:9b-chat-fp16	Linking-SQL	0.5829	–	0.6567	0.8876
glm4:9b-chat-fp16	GeoIR-Full	0.5886	0.6610	0.8329	0.7067

References

1. Affolter, K.; Stockinger, K.; Bernstein, A. A comparative survey of recent natural language interfaces for databases. *VLDB J.* **2019**, *28*, 793–819. [\[CrossRef\]](#)
2. Katsogiannis-Meimarakis, G.; Koutrika, G. A survey on deep learning approaches for Text-to-SQL. *VLDB J.* **2023**, *32*, 905–936. [\[CrossRef\]](#)
3. Deng, N.; Chen, Y.; Zhang, Y. Recent advances in Text-to-SQL: A survey of what we have and what we expect. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING 2022), Gyeongju, Republic of Korea, 12–17 October 2022*; International Committee on Computational Linguistics: Gyeongju, Republic of Korea, 2022; pp. 2166–2187. Available online: <https://aclanthology.org/2022.coling-1.190/> (accessed on 5 July 2026).
4. Li, J.; Wang, W.; Ku, W.-S.; Tian, Y.; Wang, H. SpatialNLI: A spatial domain natural language interface to databases using spatial comprehension. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 5–8 November 2019*; pp. 339–348. [\[CrossRef\]](#)
5. Wang, W.; Li, J.; Ku, W.-S.; Wang, H. Multilingual spatial domain natural language interface to databases. *GeoInformatica* **2024**, *28*, 29–52. [\[CrossRef\]](#)
6. Liu, M.; Wang, X.; Xu, J.; Lu, H.; Tong, Y. NALSpatial: A natural language interface for spatial databases. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 2056–2070. [\[CrossRef\]](#)
7. Zhang, X.; Xiao, F.; Yan, L.; Zhang, Z. GS-SQL: Modeling spatial semantics in spatial Text-to-SQL. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2024), Yokohama, Japan, 30 June–5 July 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–7. [\[CrossRef\]](#)
8. Jiang, Y.; Yang, C. Is ChatGPT a good geospatial data analyst? Exploring the integration of natural language into Structured Query Language within a spatial database. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 26. [\[CrossRef\]](#)

9. Wang, H.; Guo, L.; Liang, Y.; Liu, L.; Huang, J. GPT-based Text-to-SQL for spatial databases. *ISPRS Int. J. Geo-Inf.* **2025**, *14*, 288. [CrossRef]
10. Yu, C.; Yao, Y.; Zhang, X.; Zhu, G.; Guo, Y.; Shao, X.; Shibasaki, M.; Hu, Z.; Dai, L.; Guan, Q.; et al. Monkuu: A LLM-powered natural language interface for geospatial databases with dynamic schema mapping. *Int. J. Geogr. Inf. Sci.* **2026**, *40*, 588–609. [CrossRef]
11. Zelle, J.M.; Mooney, R.J. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, OR, USA, 4–8 August 1996*; AAAI Press: Menlo Park, CA, USA, 1996; pp. 1050–1055. Available online: <https://aaai.org/papers/156-aaai96-156-learning-to-parse-database-queries-using-inductive-logic-programming/> (accessed on 5 July 2026).
12. Giordani, A.; Moschitti, A. Corpora for automatically learning to map natural language questions into SQL queries. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010), Valletta, Malta, 17–23 May 2010*; European Language Resources Association (ELRA): Paris, France, 2010; pp. 2690–2697. Available online: <https://aclanthology.org/L10-1502/> (accessed on 5 July 2026).
13. Zhong, V.; Xiong, C.; Socher, R. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv* **2017**, arXiv:1709.00103. [CrossRef]
14. Xu, X.; Liu, C.; Song, D. SQLNet: Generating structured queries from natural language without reinforcement learning. *arXiv* **2017**, arXiv:1711.04436. [CrossRef]
15. Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), Brussels, Belgium, 31 October–4 November 2018*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 3911–3921. [CrossRef]
16. Yu, T.; Yasunaga, M.; Yang, K.; Zhang, R.; Wang, D.; Li, Z.; Radev, D. SyntaxSQLNet: Syntax tree networks for complex and cross-domain Text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), Brussels, Belgium, 31 October–4 November 2018*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 1653–1663. [CrossRef]
17. Min, Q.; Shi, Y.; Zhang, Y. A pilot study for Chinese SQL semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, 3–7 November 2019*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 3652–3658. [CrossRef]
18. Wang, L.; Zhang, A.; Wu, K.; Sun, K.; Li, Z.; Wu, H.; Zhang, M.; Wang, H. DuSQL: A large-scale and pragmatic Chinese Text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020), Online, 16–20 November 2020*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 6923–6935. [CrossRef]
19. Sun, N.; Yang, X.; Liu, Y. TableQA: A large-scale Chinese Text-to-SQL dataset for table-aware SQL generation. *arXiv* **2020**, arXiv:2006.06434. [CrossRef]
20. Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; Zhang, D. Towards complex Text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Florence, Italy, 28 July–2 August 2019*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 4524–4535. [CrossRef]
21. Lin, X.V.; Socher, R.; Xiong, C. Bridging textual and tabular data for cross-domain Text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 4870–4888. [CrossRef]
22. Wang, B.; Shin, R.; Liu, X.; Polozov, O.; Richardson, M. RAT-SQL: Relation-aware schema encoding and linking for Text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, 5–10 July 2020*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 7567–7578. [CrossRef]
23. Rubin, O.; Berant, J. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2021), Online, 6–11 June 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 311–324. [CrossRef]
24. Scholak, T.; Schucher, N.; Bahdanau, D. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021), Punta Cana, Dominican Republic, 7–11 November 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 9895–9901. [CrossRef]
25. Li, H.; Zhang, J.; Li, C.; Chen, H. RESDSQL: Decoupling schema linking and skeleton parsing for Text-to-SQL. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Palo Alto, CA, USA, 2023; Volume 37, pp. 13067–13075. [CrossRef]
26. Gan, Y.; Chen, X.; Purver, M. Re-appraising the schema linking for Text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, ON, Canada, 9–14 July 2023*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 835–852. [CrossRef]

27. Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; Zhou, J. Text-to-SQL empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.* **2024**, *17*, 1132–1145. [CrossRef]
28. Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. Can LLM already serve as a database interface? A big bench for large-scale database grounded Text-to-SQLs. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 42330–42357. [CrossRef]
29. Liu, M.; Wang, X.; Xu, J. NALSD: A natural language interface for spatial databases. In Proceedings of the 18th International Symposium on Spatial and Temporal Data (SSTD 2023), Calgary, AB, Canada, 23–25 August 2023; pp. 175–179. [CrossRef]
30. Liu, M.; Wang, X.; Xu, J.; Lu, H. NALSpatial: An effective natural language transformation framework for queries over spatial data. In Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, Hamburg, Germany, 13–16 November 2023; pp. 1–4. [CrossRef]
31. Xiao, F.; Li, J. An improved abstract syntax tree for spatial Text-to-SQL in geography domain. In *Proceedings of the 9th International Conference on Next Generation Computing (ICNGC 2023)*, Da Nang, Vietnam, 20–23 December 2023; Korean Institute of Next Generation Computing: Seoul, Republic of Korea, 2023; pp. 104–106. Available online: <https://www.earticle.net/Article/A448128> (accessed on 5 July 2026).
32. Staniek, M.; Schumann, R.; Züfle, M.; Riezler, S. Text-to-OverpassQL: A natural language interface for complex geodata querying of OpenStreetMap. *Trans. Assoc. Comput. Linguist.* **2024**, *12*, 562–575. [CrossRef]
33. Mansourian, A.; Oucheikh, R. ChatGeoAI: Enabling geospatial analysis for public through natural language, with large language models. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 348. [CrossRef]
34. Lawrence, C.; Riezler, S. NLmaps: A natural language interface to query OpenStreetMap. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, Osaka, Japan, 11–16 December 2016; The COLING 2016 Organizing Committee: Osaka, Japan, 2016; pp. 6–10. Available online: <https://aclanthology.org/C16-2002/> (accessed on 22 June 2026).
35. Potnis, A.V.; Shinde, R.C.; Durbha, S.S. Towards natural language question answering over earth observation linked data using attention-based neural machine translation. *arXiv* **2021**, arXiv:2101.09427. [CrossRef]
36. Nanjing Municipal People’s Government. City Overview. Available online: https://www.nanjing.gov.cn/zgnjsjb/csgk/202509/t20250919_5652865.html (accessed on 22 June 2026).
37. Wuhan Municipal People’s Government. Wuhan City Overview. Available online: https://www.wuhan.gov.cn/zjwh/whgk/202004/t20200414_999422.shtml (accessed on 22 June 2026).
38. Kefalidis, S.-A.; Punjani, D.; Tsalapati, E.; Plas, K.; Pollali, M.; Mitsios, M.; Tsokanaridou, M.; Koubarakis, M.; Maret, P. Benchmarking geospatial question answering engines using the dataset GeoQuestions1089. In *The Semantic Web—ISWC 2023; Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2023; Volume 14266, pp. 266–284. [CrossRef]
39. Piccolo, G. *Earth Observation GeoSQL Query Set for Natural Language to PostGIS Agentic Evaluation*, version v1; Zenodo: London, UK, 2026. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.