

Eugene Database Stats — End-to-End Flow

Developer walkthrough: FastAPI router → DI factory → Neo4j adapter → per-label COUNT Cypher → DatabaseStats response

Audience

Engineers who need to understand how Eugene reports its top-level graph health numbers (total nodes, total relationships, per-label counts for drug, disease, gene_protein, Patent_Application, ClinicalTrial, Research, Investigators, Organization) via the single read-only GET /stats endpoint. Every reference uses the form path/to/file.py:line.

Reference endpoint

- GET /stats — returns a single DatabaseStats JSON payload. No query parameters, no path parameters, no body.

Caveat

The router carries a # todo: add daily caching comment — every call today fans out to ~12 Cypher queries that scan the entire graph (MATCH (n), MATCH (n)-[r]-(n2), and one MATCH (n:`Label`) per foundational label). On a large graph this can be slow; treat the endpoint as a coarse health probe, not a hot path. The DI is also resolved at module load (stats_adapter = neo4j_database_stats_adapter()), so the Neo4j driver is created once at import time.

0. Schema — what the endpoint exposes

The response is a flat dataclass of pre-formatted count strings (123 -> "123", 1500 -> "1.5k", 2300000 -> "2.3m"). It mixes three kinds of metric:

- **Global graph totals** — node_count, relationship_count.
- **Per-label node counts** — one MATCH (n:`Label`) RETURN count(n) per foundational label: drug, disease, gene_protein, Patent_Application, ClinicalTrial, Research, Investigators, Organization.
- **Domain rollups** — distinct count of organization_id on :Organization (disambiguated orgs), distinct therapeutic_area and therapeutic_subgroup on :ClinicalTrial, plus min/max filing_date on :Patent_Application.
- drug_synonym_count is wired in the model but the adapter hard-codes it to 0 — the route does NOT actually count :drug_synonym nodes (see neo4j_database_stats_adapter.py:62).
- There are no Neo4j CREATE CONSTRAINTs in Eugene; these COUNTs are full label scans, no index assist.

```
DatabaseStats(  
    node_count, relationship_count,  
    drug_count, drug_synonym_count,          # synonym always 0  
    disease_count, gene_protein_count,  
    uspto_count, uspto_filing_date_min, uspto_filing_date_max,  
    clinical_trial_count,  
    therapeutic_area_count, therapeutic_area_subgroup_count,  
    pubmed_count, pubmed_researcher_count,  
    organization_count, disambiguated_organization_count,  
)
```

1. HTTP entry — router, params, DI

[src/stats/router/database_stats_router.py](#)

- Router prefix `/stats`, tag `stats`.
- Single handler `fetch_database_stats` (line 20) bound to `GET ""` — the full path is `GET /stats`.
- `response_model=DatabaseStats` with `response_model_exclude_none=True`.
- No query, path, or body params; no validators; no auth dependency at the route level.
- DI at module load (line 16): `stats_adapter = neo4j_database_stats_adapter()` — the adapter (and its Neo4j driver) is constructed once at import time, not per request.
- Registered in `src/eugene_ws.py:34, 60` via `app.include_router(database_stats_router.router)`.

```
router = APIRouter(prefix="/stats", tags=["stats"])
stats_adapter = neo4j_database_stats_adapter()
```

```
@router.get("", response_model=DatabaseStats,
             response_model_exclude_none=True)
async def fetch_database_stats():
    # todo: add daily caching
    return stats_adapter.query_system_stats()
```

Factory

[src/stats/conf/conf.py](#)

```
def _neo4j_driver() -> Driver:
    return GraphDbConnectionFactory.remote_neo4j_instance_from_env()

def neo4j_database_stats_adapter() -> Neo4jDatabaseStatsAdapter:
    return Neo4jDatabaseStatsAdapter(driver=_neo4j_driver())
```

The driver comes from `GraphDbConnectionFactory` in `src/graph/infra/db/graph_db_connection_factory.py`, which reads Neo4j URI / user / password from the environment. The adapter pins `database="neo4j"`.

2. Query adapter — verbatim Cypher

`src/stats/infra/db/adapter/neo4j_database_stats_adapter.py`

- `query_system_stats()` (line 26) is the only public method. It calls `ensure_connection(self.driver)` once, then runs each Cypher below in sequence and assembles `DatabaseStats`.
- Each helper uses `driver.execute_query(query=..., result_transformer=neo4j.Result.single)` and catches `DriverError / Neo4jError` (re-raises after logging).
- Each helper is wrapped with `@log_time` (`annotation/timer_annotation.py`) — expect one timing log line per query in the response trace.

Total node count (line 137)

```
MATCH (n)
RETURN count(n) as node_count
```

Total relationship count (line 117)

```
MATCH (n)-[r]-(n2)
RETURN count(r) as relationship_count
```

Note the undirected `-[r]-` pattern: it returns **twice** the stored edge count because each relationship is traversed in both directions. The number reported in `relationship_count` is therefore 2x the underlying edge count — this is a known artifact of the current query, not a bug in the count formatter.

Per-label COUNT template (line 94)

```
MATCH (n:`%s`)
RETURN count(n) as count
```

Invoked once per label via `_count_all_by_label(label)` with these label strings (from `FoundationalNodeEnum`, `src/foundation/model/foundational_node_enum.py`): `drug`, `disease`, `gene_protein`, `Patent_Application`, `ClinicalTrial`, `Research`, `Investigators`, `Organization`.

2 (cont.) — rollup Cypher

Disambiguated organizations (line 157)

```
MATCH (n:`Organization`)
RETURN count(distinct(n.organization_id)) as disambiguated_org_count
```

Counts distinct values of the `organization_id` property — one canonical entity may have several `:Organization` nodes sharing an id, so this number is $\leq$ `organization_count`.

Therapeutic areas (line 182)

```
MATCH (n:`ClinicalTrial`)
RETURN COUNT(DISTINCT(n.therapeutic_area)) as therapeutc_area_count,
       COUNT(DISTINCT(n.therapeutic_subgroup)) as therapeutc_area_group_count
```

One query returns both numbers. Property keys `therapeutic_area` and `therapeutic_subgroup` live on `:ClinicalTrial` nodes. (The Cypher aliases misspell 'therapeutic' as `therapeutc_*` — this only matters if you copy the query into Neo4j Browser.)

USPTO filing-date range (line 208)

```
MATCH (n:`Patent_Application`)
RETURN toStringOrNull(min(n.filing_date)) as min_uspto_filing_date,
       toStringOrNull(max(n.filing_date)) as max_uspto_filing_date
```

`toStringOrNull` guarantees JSON-safe strings (or `null`) even when the property is a Neo4j temporal type or absent.

Drug synonyms — not queried

`drug_synonym_count` is hard-coded to 0 on line 62 of the adapter. There is no Cypher for it. If you need the real number, add a `_count_all_by_label(FoundationalNodeEnum.DRUG_SYNONYM.value[1])` call.

3. Response model — DatabaseStats

`src/stats/model/system_stats.py`

- Plain `@dataclass(frozen=True, unsafe_hash=True)` — no Pydantic, but FastAPI is happy because the dataclass is JSON-serializable and used as `response_model`.
- There is no separate mapper class; the adapter builds the dataclass inline.
- Every count field is typed `str` because the static `DatabaseStats.format_count(num: int)` helper (line 28) converts integers to human-friendly suffixes (`<1k -> raw int`, `>=1k -> "1.2k"`, `>=1m -> "3.4m"`, `>=1b -> "1.0b"`). The two date fields are passed through verbatim from `toStringOrNull`.
- `response_model_exclude_none=True` drops any `null` field from the wire payload (e.g. an empty USPTO date range).

JSON sample

```
{
  "node_count": "4.2m",
  "relationship_count": "18.7m",
  "drug_count": "14.3k",
  "drug_synonym_count": "0",
  "disease_count": "31.5k",
  "gene_protein_count": "22.1k",
  "uspto_count": "612.4k",
  "uspto_filing_date_min": "1976-01-06",
  "uspto_filing_date_max": "2026-03-21",
  "clinical_trial_count": "487.2k",
  "therapeutic_area_count": "412",
  "therapeutic_area_subgroup_count": "1.8k",
  "pubmed_count": "2.1m",
  "pubmed_researcher_count": "948.0k",
  "organization_count": "126.7k",
  "disambiguated_organization_count": "58.4k"
}
```

Note the field name `uspto_filing_date_max` — a typo ('filing') baked into the public contract. UI clients (Next.js dashboard, agent UI) depend on this exact spelling; do not silently rename.

4. Data source — whole-graph dependency

Unlike per-entity routes (drug alias, patent search, etc.) this endpoint has no narrow data dependency — every Cypher query above is an unfiltered label or graph scan. The answers reflect the current state of the **entire** Neo4j graph at call time.

- **Nodes by label** come from whichever ingest populated that label: e.g. `:drug` from `bin/seed/*.cypher + src/ingest_drug_aliases.py`; `:Patent_Application` from the USPTO ingest under `src/foundation/load/`; `:ClinicalTrial` from the CT.gov ingest; `:Research + :Investigators` from the PubMed ingest; `:Organization` from the organization disambiguation pipeline.
- `relationship_count` sums *all* edges of every type (`HAS_DRUG_ALIAS`, `MENTIONS`, `CITES`, `ASSIGNED_TO`, ...), counted in both directions.
- **Therapeutic-area rollups** depend on the `therapeutic_area / therapeutic_subgroup` properties being set on `:ClinicalTrial` nodes by the CT.gov ingest. Missing properties simply do not contribute to the distinct count.
- **USPTO date range** depends on the `filing_date` property on `:Patent_Application` — if no patents exist, both bounds come back as `null` and are dropped from the response by `exclude_none`.
- `drug_synonym_count` is independent of the graph — always 0 (see Section 2b).
- Because there is no caching, each call materializes the latest numbers. Expect >100ms on a non-trivial graph (twelve full scans / aggregations).

5. Suggested debugging walk

- Spin Neo4j + the API: `docker-compose up eugene_neo4j eugene_ws` and wait for `GET /health` to return 200.
- Smoke-test the route: `curl -s http://localhost:8000/stats | jq`. Confirm all sixteen fields are present and formatted with k/m/b suffixes.
- Breakpoint at `src/stats/router/database_stats_router.py:22`; step into `stats_adapter.query_system_stats()` and watch each `_count_*` helper fire its Cypher in order.
- Tail the server log: each helper is wrapped with `@log_time` and also emits `logger.info(f"count all: {{query}}")` — you should see ~12 timing lines plus their Cypher payloads per request.
- Sanity-check a single label by hand in Neo4j Browser: paste `MATCH (n:`drug`) RETURN count(n) AS count` and compare with `drug_count` (un-format the suffix to compare).
- To verify the relationship double-count: run `MATCH (n)-[r]->(n2) RETURN count(r)` (directed) and compare with the route's undirected `MATCH (n)-[r]-(n2)` — you should see exactly 2x.
- To investigate slowness, time the helpers individually via the `@log_time` output; the per-label COUNTs scale linearly with label cardinality (no index).

6. Reference index

Schema

src/stats/model/system_stats.py	# DatabaseStats dataclass + format_count
src/foundation/model/foundational_node_enum.py	# label strings consumed by the adapter

HTTP entry

src/stats/router/database_stats_router.py	# GET /stats, module-load DI
src/eugene_ws.py	# import L34, register L60

DI / factory

src/stats/conf/conf.py	# neo4j_database_stats_adapter()
src/graph/infra/db/graph_db_connection_factory.py	# remote_neo4j_instance_from_env()

Query adapter

src/stats/infra/db/adapter/neo4j_database_stats_adapter.py	
L26 query_system_stats	# orchestrates all counts
L80 _count_all_by_label	# per-label COUNT
L94 _build_count_all_by_label	# MATCH (n:`%s`) RETURN count(n)
L103 _count_all_relationships	# MATCH (n)-[r]-(n2) (undirected, 2x)
L123 _count_all_nodes	# MATCH (n) RETURN count(n)
L143 _count_disambiguated_orgs	# DISTINCT n.organization_id
L165 _count_therapeutic_areas	# DISTINCT therapeutic_area + _subgroup
L191 _fetch_min_max_uspto_dates	# min/max n.filing_date

Cross-cutting

src/graph/util/util.py	# ensure_connection
src/annotation/timer_annotation.py	# @log_time decorator