

# OSINT Intelligence Graph

SQLite ops.db vs Triple Store vs Oxigraph (SPARQL)

38 entidades | 48 relaciones | 3 backends | 24 LLM queries

## RESULTADOS CLAVE

Benchmark (6 queries, avg 50 runs):

operations.db:	queries=0.2ms	cold_start=0.8ms
triple store:	queries=0.1ms	cold_start=0.4ms
oxigraph:	queries=0.3ms	cold_start=37.2ms

LLM Query Generation (claude -p haiku):

ops_sql (operations.db):	8/8 ejecutan sin error (100%)
triple_sql:	7/8 ejecutan sin error (87.5%)
sparql:	8/8 generadas con sintaxis valida*

(\*SPARQL no evaluado por lock de Oxigraph en kernel)

Assertions: 38 creadas, 38 pass (100%)

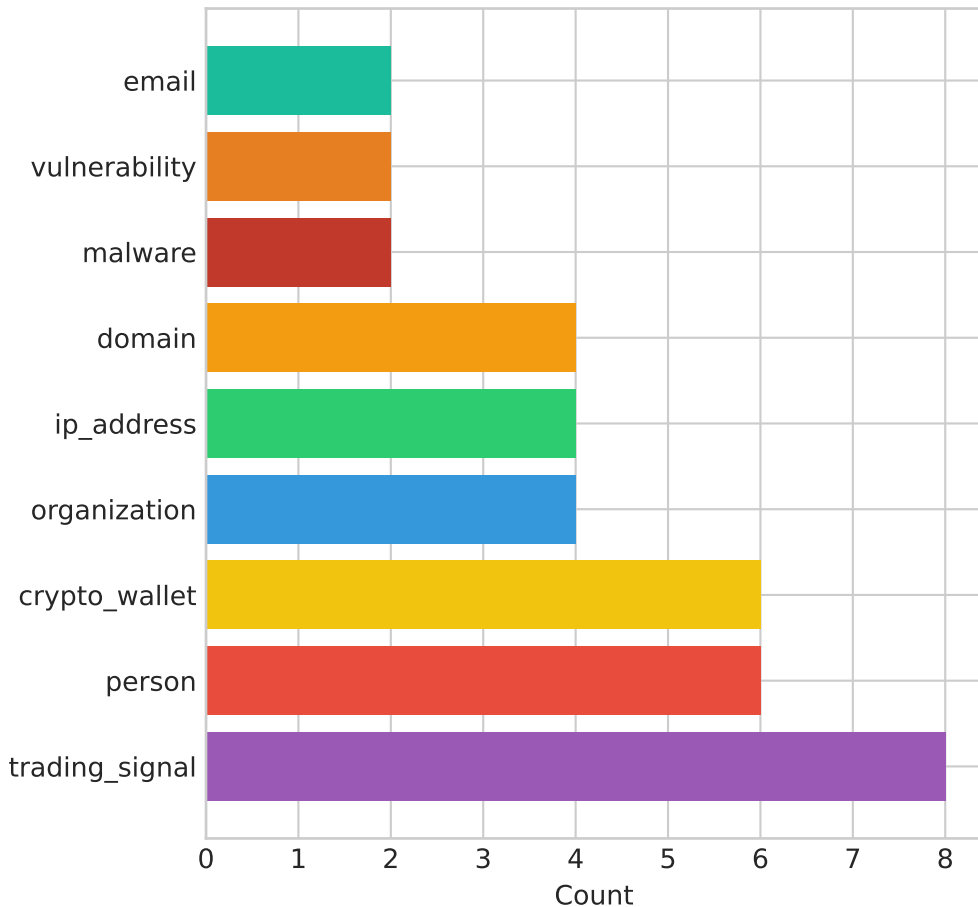
Sigma.js: HTML interactivo con 38 nodos, 48 edges

RECOMENDACION: operations.db

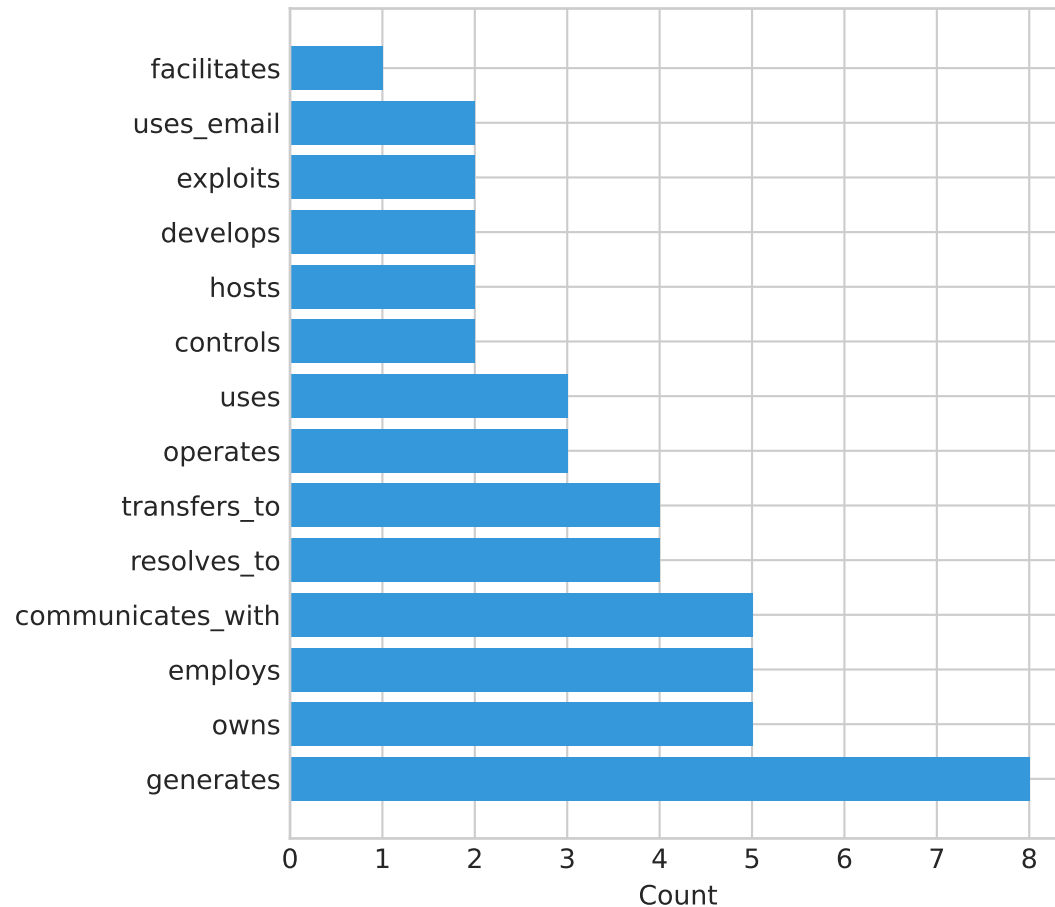
- 100% LLM query success
- Schema rico: entities + relations + assertions + executions
- json\_extract para metadata flexible
- Integrado en fn\_registry (fn ops CLI, reactive loop)
- Assertions evaluan calidad de inteligencia automaticamente

# Dataset OSINT: 2 narrativas interconectadas

## Entidades por tipo

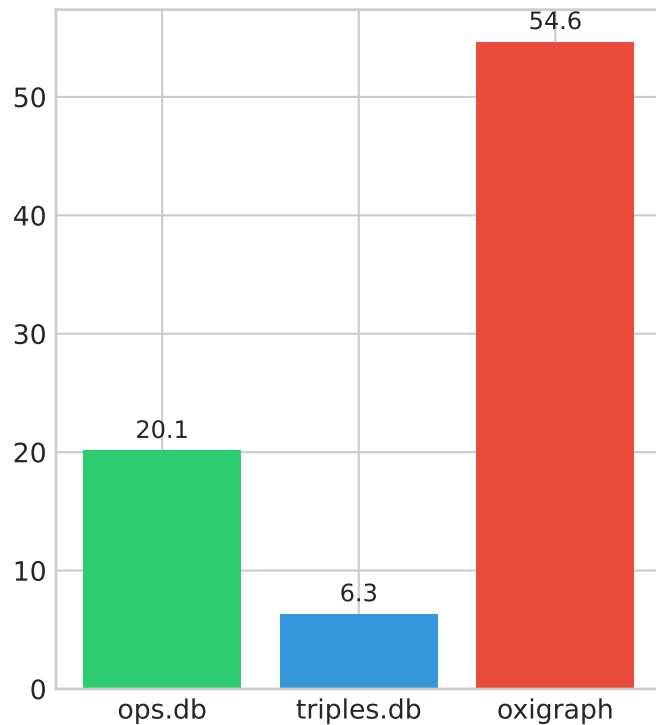


## Relaciones por tipo

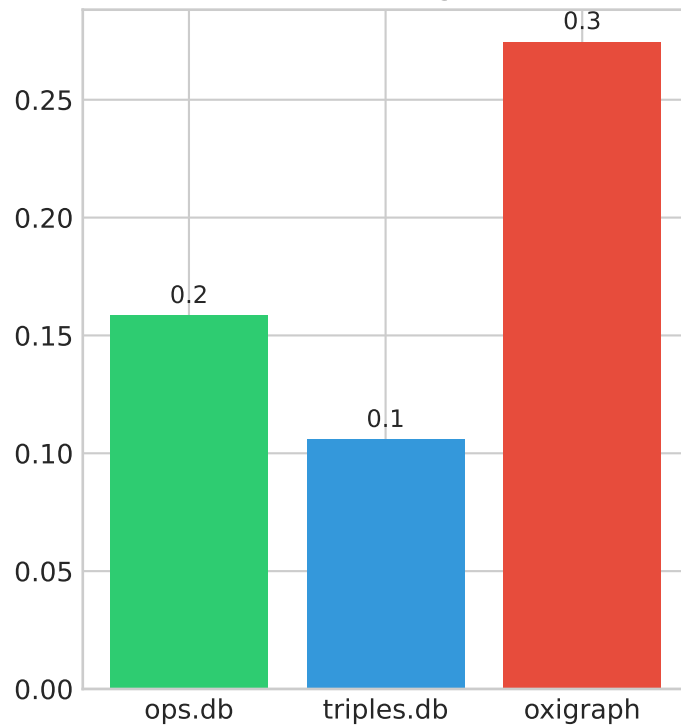


# Benchmark: 3 Backends

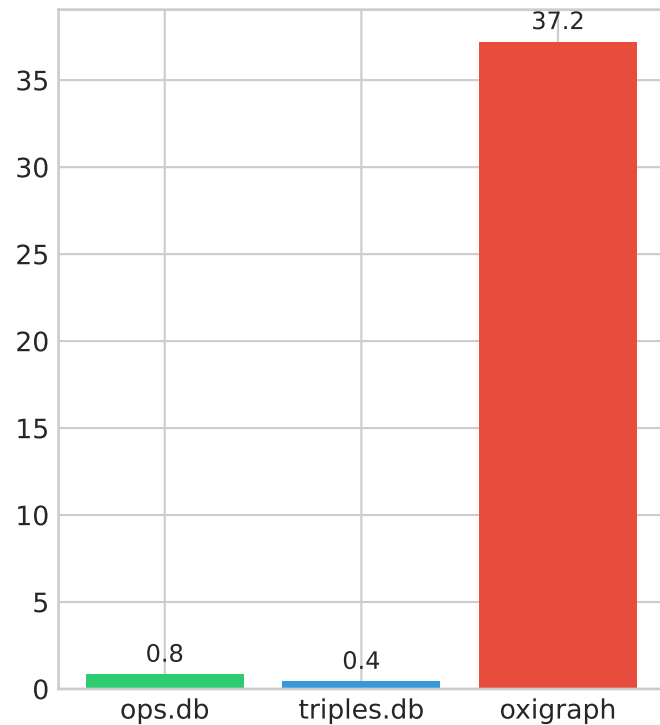
Insert Time (ms)



6 Queries avg (ms)

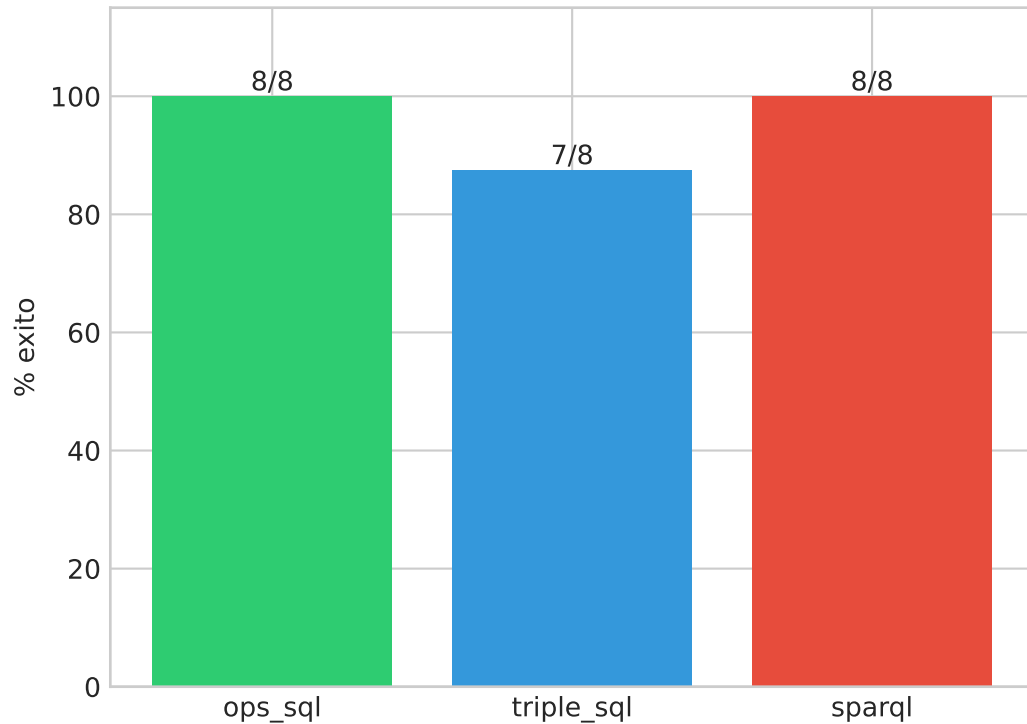


Cold Start (ms)

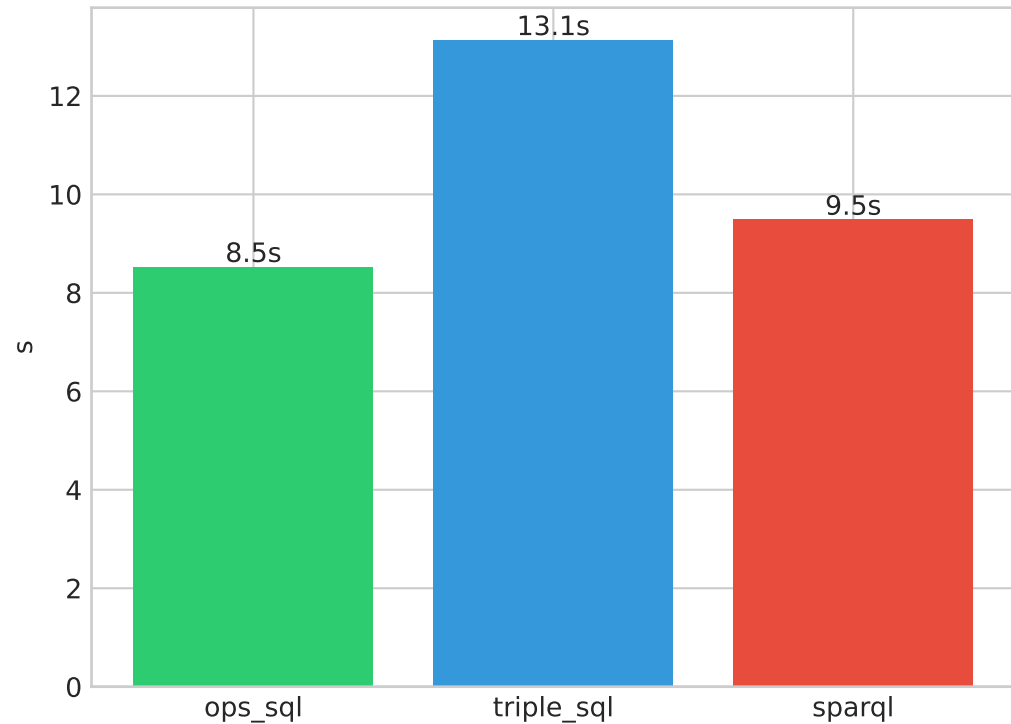


# LLM Query Generation: claude -p haiku

Queries ejecutables



Tiempo generacion promedio



# Assertions y Validacion Cruzada

## ASSERTIONS OSINT (operations.db)

=====

Total: 38 assertions evaluadas

Pass: 38 | Fail: 0 | Skip: 0

### Por severity:

critical : 10 total, 10 pass, 0 fail

warning : 20 total, 20 pass, 0 fail

info : 8 total, 8 pass, 0 fail

## CROSS-VALIDATION (backends devuelven mismos resultados)

=====

q1: ops= 1 triple= 1 [OK]

q2: ops= 4 triple= 4 [OK]

q3: ops= 8 triple= 8 [OK]

q4: ops= 0 triple= 0 [OK]

q5: ops=20 triple=20 [OK]

q7: ops= 3 triple= 3 [OK]

## GAP ANALYSIS: Funciones propuestas para OSINT

=====

validate\_cve\_id – Verificar formato CVE-YYYY-NNNNN

validate\_crypto\_addr – Checksum BTC/ETH addresses

geoip\_lookup – Enriquecer IPs con geolocalizacion

whois\_enrichment – Wrapper Python de lookup\_whois

threat\_score\_calc – Risk score ponderado multi-signal

# Recomendaciones para OSINT + AI Retrieval

## RANKING PARA SISTEMA OSINT CON AI RETRIEVAL

1. operations.db (SQL) [RECOMENDADO]
  - + 100% LLM query success rate
  - + Schema rico: entities + relations + assertions + executions + logs
  - + json\_extract(metadata, '\$.campo') para datos flexibles
  - + Assertions evalúan calidad de inteligencia automáticamente
  - + Reactive loop: assertions -> proposals -> mejoras al registry
  - + fn ops CLI para gestión desde terminal
  - + Sigma.js se alimenta directo del mismo backend
  - + Cold start: 0.8ms
  - No tiene inferencia semántica ni ontologías
2. SQLite Triple Store
  - + 87.5% LLM query success (7/8)
  - + Insert rápido (6.3ms vs 20ms)
  - + CTEs recursivos para traversal multi-hop
  - Pierde riqueza de operations.db (no assertions, no executions)
  - Property filters requieren JOINS verbosos (2+ tablas)
  - No aporta nada que operations.db no haga mejor
3. Oxigraph (SPARQL)
  - + Property paths nativos (+, \*) para traversal
  - + Estandar W3C, interoperable con linked data
  - + Queries SPARQL generadas con buena sintaxis por el LLM
  - Cold start lento (37ms vs 0.8ms)
  - Lock de archivo impide acceso concurrente
  - xsd: casting frágil para comparaciones numéricas
  - 390KB disco vs 4KB triple store

## ARQUITECTURA RECOMENDADA:

```
operations.db (almacenamiento principal)
|
+--> fn ops CLI (gestión de entities/reasons)
+--> assertions (calidad de inteligencia)
+--> reactive loop (proposals automáticas)
+--> sigma.js HTML (visualización)
+--> claude -p + SQL (AI retrieval)
+--> embeddings (búsqueda semántica futura)
```

## PROXIMOS PASOS:

1. Crear app en apps/osint/ con operations.db real
2. Pipeline de ingestión desde fuentes OSINT (APIs, feeds)
3. Funciones: validate\_cve, validate\_crypto, geoip\_lookup
4. Embeddings sobre entity descriptions para búsqueda semántica
5. Dashboard sigma.js auto-generado desde operations.db