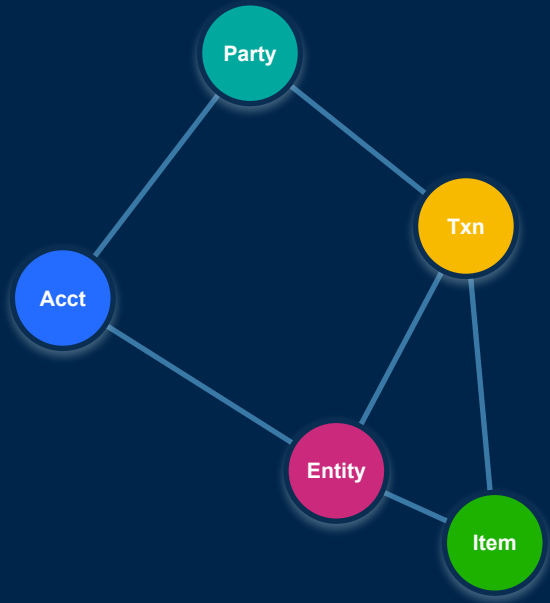
INTUIT turbotax creditkarma quickbooks mailchimp

KNOWLEDGE GRAPHS
using Neo4j

A shared model of meaning for connected data



Harshavardhan Srinivasan
Staff Software Engineer
[date]

AGENDA

What we'll cover

01

The problem

Enterprise data models — and why connecting them is so hard

02

How an ontology helps

A shared layer of meaning over your many models

03

How an ontology is created

Derive it from what you own; from schema (TBox) to data (ABox)

04

Why a graph — not RDF/SPARQL

The storage & query trade-off, and where Neo4j fits

05

Ontology in action

Enrichment, and ontologies that agents use directly

06

The payoff

Real-world use cases, and the takeaways

01

The problem

Your data is connected — your systems aren't. Customers and AI feel the gap.

THE PROBLEM

Every team models the world a little differently

A large organization doesn't have one data model — it has dozens. Each service, team, and system owns its own slice, in its own shape, with its own names.

Billing service

```
customer_id
acct_no  →
cust_name
status
```

CRM system

```
ClientID
Party
AccountRef
segment
```

Ledger

```
entityId
owner
gl_account
balance
```

Same real-world things, three vocabularies. Is `customer_id` the same as `ClientID` the same as `owner`? Nothing in the data says so — the meaning lives in people's heads and tribal knowledge.

THE PROBLEM

The cost of not connecting them

When the connections are missing, your customers feel it — and your AI can't reason.

A disjointed customer experience

No single view of the customer, so they repeat themselves and get service that doesn't add up across touchpoints.

Answers take weeks, not moments

The questions that matter are about how things connect — and those are exactly the ones no system can answer quickly.

Value hidden in the gaps

The signal lives between systems — the relevant offer, the early risk, the fraud ring — so it goes unseen.

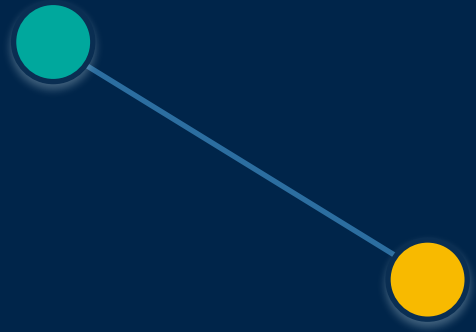
AI that guesses instead of knowing

Without connected context, assistants can't reason about how things relate — so they hallucinate, and you can't ship them.

02

How an ontology helps

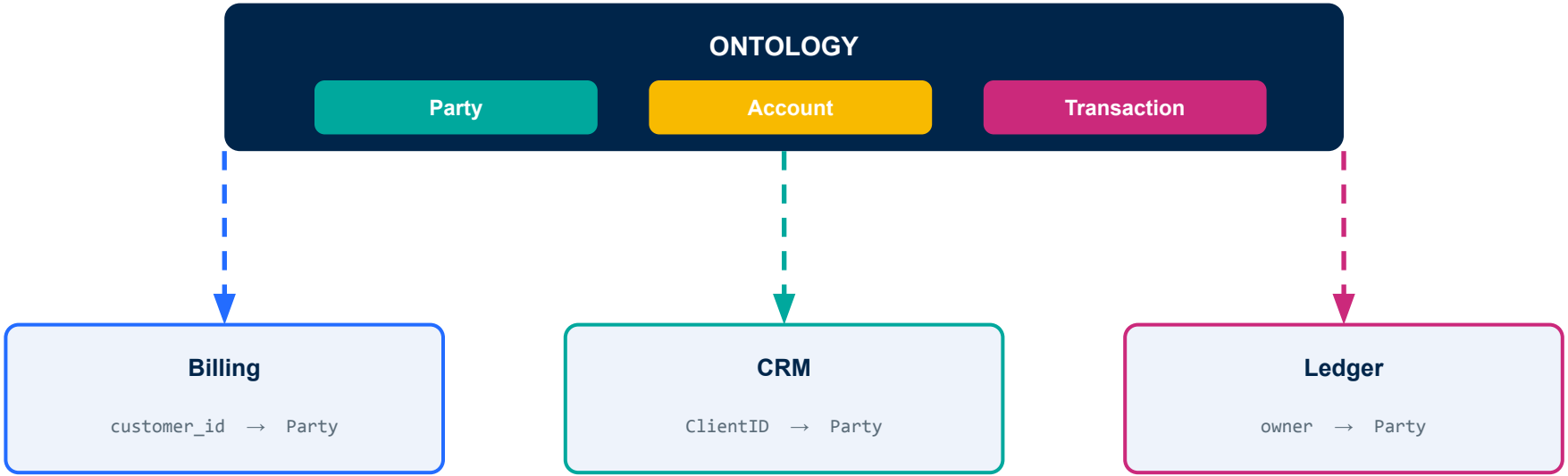
A shared layer of meaning over your many models.



HOW IT HELPS

An ontology is a shared layer of meaning

An ontology is an explicit, shared model of **what things are** and **how they relate** — one agreed vocabulary that sits above the many systems and maps them to common meaning.



Each system maps its local terms up to the shared model — so “party” means one thing everywhere.

CORE CONCEPT

Anatomy of a property graph

Nodes

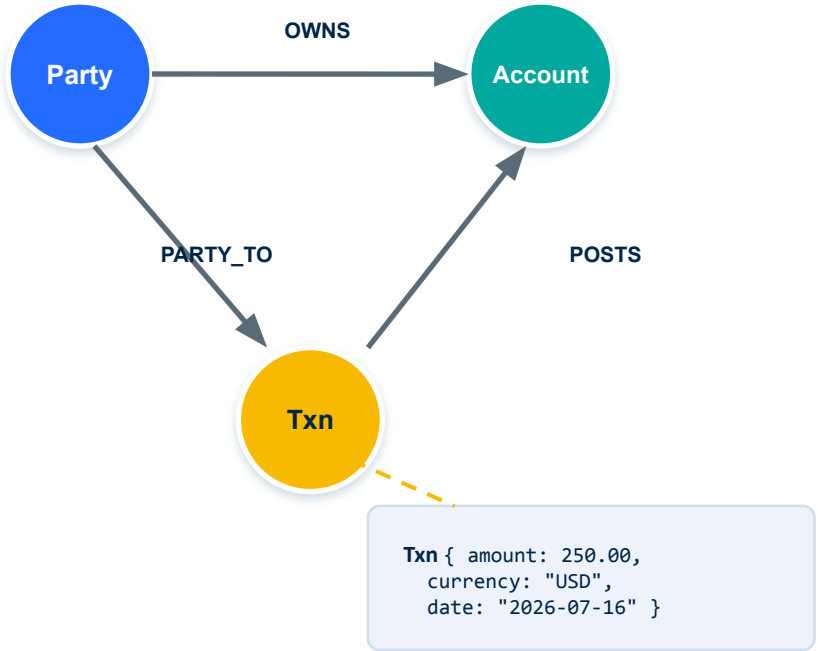
Entities — a Party, an Account, a Transaction

Relationships

Typed, directed edges — OWNS, PARTY_TO

Properties

Key/value data on both nodes and edges



Ontology vs. the graph

Ontology = the schema

- The classes of things that can exist (Party, Account, Txn)
- How they may relate (a Party OWNS an Account)
- Rules, constraints, and cardinality
- The vocabulary of meaning — stable, reusable

"What kinds of things exist and how they connect."

Graph = the data

- The actual instances (Party #42, Account #90)
- The concrete edges between them
- Millions of nodes and relationships
- Populated from your systems of record

"The specific things that actually exist right now."

Analogy: the ontology is the `class`; the graph is the `objects`.

03

How an ontology is created

Derive it from what you already own — then from schema to data.

DERIVATION PATTERN

Inferring the ontology you already have

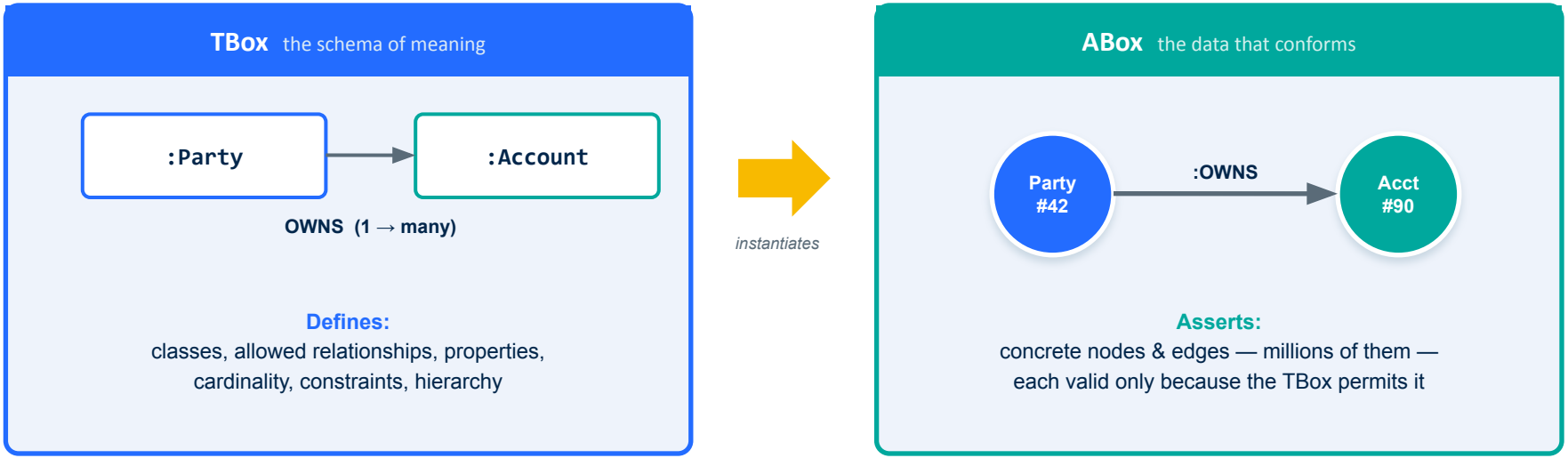
A repeatable technique: read the structures you already own and derive a first-cut graph shape from them.

`billing.acct_no → account.id` becomes `(:Billing)-[:FOR_ACCOUNT]->(:Account)`

Caveat worth stating out loud: inference gives you a strong first draft, not a finished ontology. Many “keys” are soft refs or lookups, not real relationships — inference proposes, a human disposes. Automate the tedium; keep judgment in the loop.

TBox informs ABox: schema instantiates data

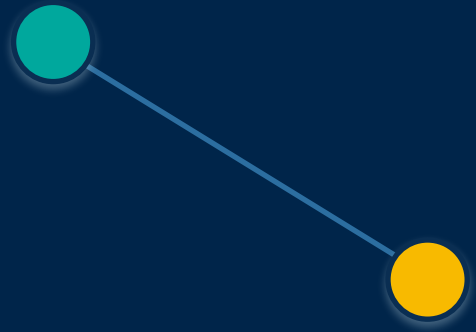
An ontology has two layers. The **TBox** (terminology) defines the *types* of things and how they may relate. The **ABox** (assertions) is the *actual data* — instances that must conform to the TBox.



04

Why a graph — not RDF/SPARQL

Same ontology, two technologies. Here's the trade-off.



Two ways to store an ontology

Ontologies are often associated with **RDF** — the W3C semantic-web stack. A **labeled property graph (LPG)** like Neo4j is the pragmatic alternative. Both model meaning; they trade off differently.

RDF + SPARQL + OWL the semantic-web standard

Triples: everything is subject-predicate-object statements

SPARQL: the standard query language for triples

OWL: rich formal logic + automated reasoning / inference

Strength: open standards, portability, federation across orgs

Labeled property graph Neo4j

Nodes & edges: typed, directed, with properties on both

Cypher: pattern-matching query that reads like a diagram

Model: matches how engineers already think & whiteboard

Strength: fast traversals, developer ergonomics, easy adoption

Our call: for building products fast, the LPG wins on **ergonomics and traversal performance** — and you can add reasoning where you need it. RDF shines when open standards and cross-org federation are the priority.

Why a native graph database

Index-free adjacency

Each node holds direct pointers to its neighbors. Traversal is a local pointer-hop — its cost doesn't grow with total data size.

Constant-time hops

A relationship traversal is O(1), not a join. Deep and variable-length paths stay fast where SQL degrades.

ACID + Cypher

A real transactional database with an expressive, declarative query language purpose-built for patterns.

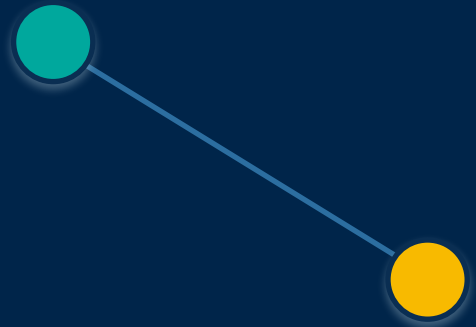
Model = mental model

You query it by drawing the pattern — (:Party)-[:OWNS]->(:Account) — it reads like the diagram itself.

05

Ontology in action

Enrichment, and ontologies that agents can use directly.



SEMANTICS

Enrichment: adding meaning to nodes

A raw graph tells you what connects to what. Enrichment adds what things mean — so queries can reason at the level of concepts, not just records.

Classification

Tag a node with a semantic type or category so it can be grouped and reasoned about.

```
Txn { kind:'transfer' }
```

Derived labels

Add labels from rules — e.g. a Party that owns > N accounts is also :HighValue.

```
:Party:HighValue
```

Hierarchies

Link concepts to a taxonomy so children inherit meaning from parents.

```
(:Txn)-[:IS_A]->(:Cash)
```

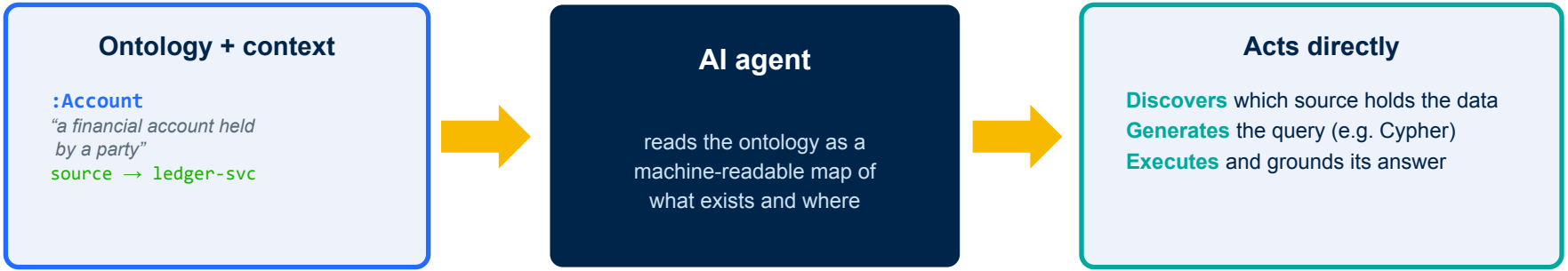
Embeddings

Attach vectors for similarity search alongside structural traversal.

```
n.embedding = [...]
```

A context-rich ontology becomes an agent's map

Enrich the ontology with **semantic context** — human-readable descriptions, synonyms, and links from each class to the **data source** that holds it. That context turns the ontology into something an AI agent can read and act on.

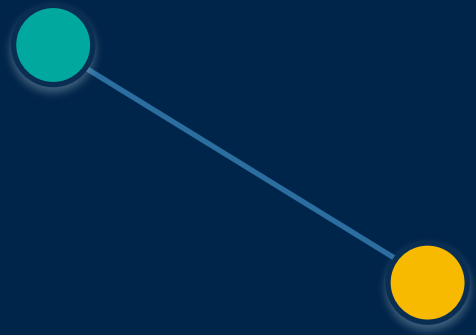


Why it matters: with meaning attached, the agent doesn't need a hard-coded integration per source. The ontology tells it **what things mean, where they live, and how they connect** — so it can discover and query new sources on its own.

06

The payoff

Where knowledge graphs earn their keep.



USE CASES

Where knowledge graphs shine

360° customer view

Unify fragments across systems into one connected picture — so service finally adds up.

Answers in real time

Multi-hop relationship questions that used to take weeks become a single traversal.

Recommendations

Surface the relevant next offer through paths of shared connections.

Fraud & risk

Ring detection: spot the hidden links between entities that look unrelated.

One source of truth

Every system’s data resolves to the same entities — no more conflicting answers.

AI you can trust

Ground assistants in structured, connected context — fewer hallucinations.

Three things to take home

1

Your data is already connected

Microservices and normalized tables didn't remove the relationships — they just made them invisible. A graph makes them usable again.

2

Store relationships as first-class

When the relationship is the unit of value, a native graph turns multi-hop questions from expensive joins into cheap traversals.

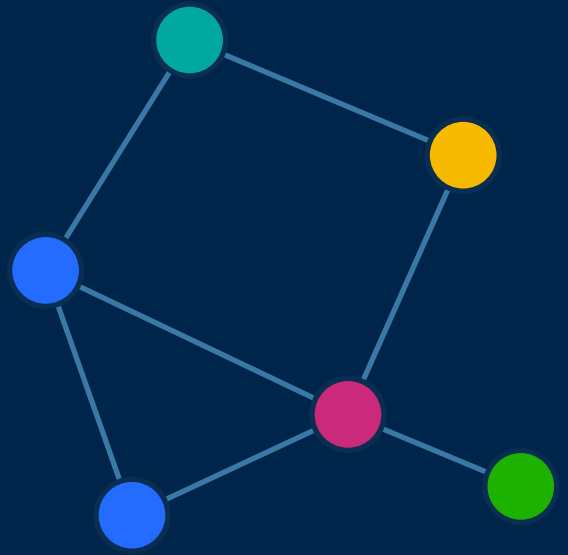
3

You likely already have an ontology

Derive a first draft from the schemas and object models you own — then keep human judgment in the loop to finish it.

Thank you

Questions & discussion



Harshavardhan Srinivasan
Staff Software Engineer