



# A knowledge graph is more than just a graph database

**Ora Lassila & Charles Ivie**

Amazon Neptune



# Who we are

## Ora Lassila

Principal Graph Technologist

- long history with graphs and ontologies
- co-author of the original RDF specification and the seminal article on the Semantic Web

## Charles Ivie

Senior Graph Architect

- long history with graphs and data-driven solutions
- founder of two knowledge graph startups, including Data Lens

# Game plan

Knowledge graphs as a “use case”



Knowledge graph architectures



Software patterns and abstractions



Next steps?

# So what is this talk really about...?

We regularly help our customers build systems that leverage knowledge graphs

- customers come to us because they need a graph database (or have been told that they need one)
- the database is not really the challenge, however

To actually build a knowledge graph system, you need many other tools as well

- (and you may also need to write some code of your own...)

This talk reflects our experiences helping customers as well as our long history in building knowledge graph and Semantic Web software

# Knowledge graphs as a “use case”

(Not a single, narrowly defined use case...)

Building a knowledge graph is usually a project that anticipates to be followed by several specific use cases

Knowledge graphs are typically the first step towards large-scale data integration and general “cleanup” of an organization’s data

Acknowledging that a “knowledge graph system” is really a platform on top of which many applications (i.e., use cases) are implemented, access interfaces and abstractions are of paramount importance

# Graph databases

Graph storage is needed for most knowledge graph projects

Since one seldom uses just a graph database (and builds everything else from scratch), open (or *de facto*) standards help with tooling

- W3C RDF & SPARQL
- Apache Tinkerpop & Gremlin
- Amazon Neptune supports all of the above, and lets customers choose

Nascent market, customers still “kicking the tires”

- having two separate graph models – RDF and property graphs – adds to the confusion; lack of standards for property graphs does not help either
- many customers struggle with the choice between the two models

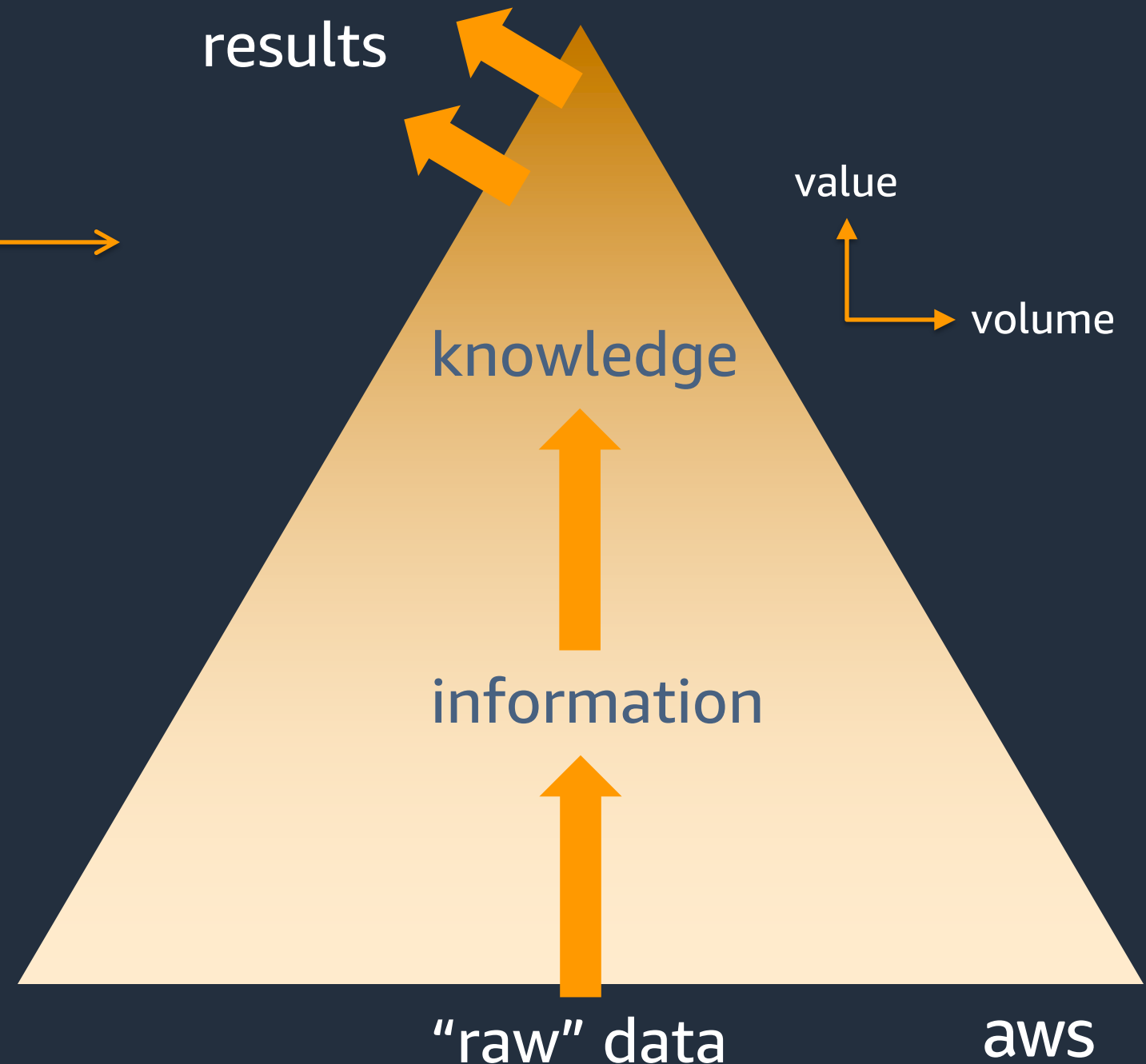
*(Note: this talk may be preferential to RDF...)*

# Knowledge graph architectures: value chain vs. data flow

We have to understand the “value chain” for data

This model fits most of our projects

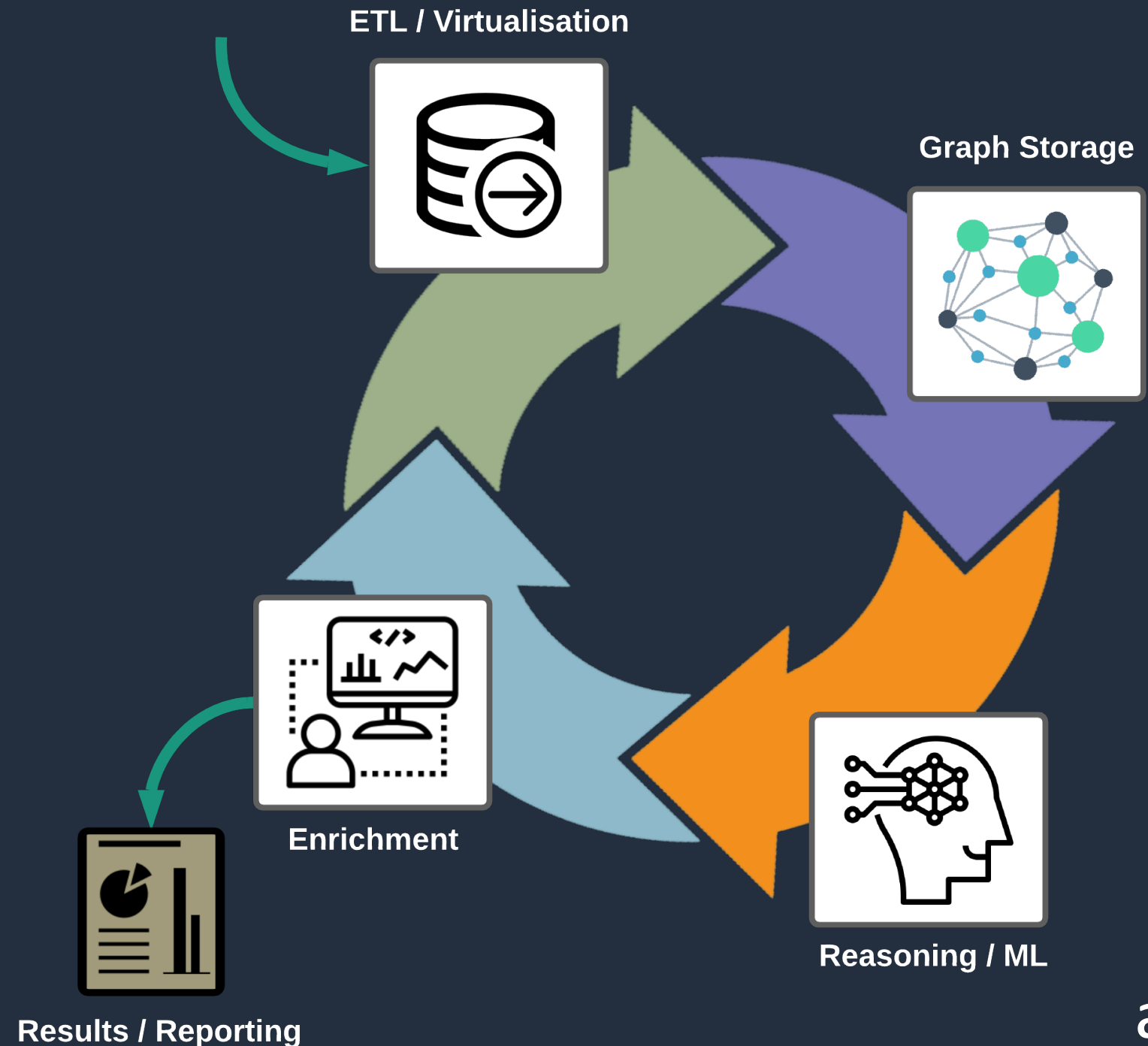
Different technologies apply at different levels of this pyramid, from simple data cleansing to sophisticated symbolic reasoning



# Knowledge graph architectures: value chain vs. data flow

In reality, the value chain does not have to be a straight flow

It can be more like a “virtuous cycle” where results of various types of enrichment are fed back to the graph



# Amazon Neptune and friends

We complete the architecture using Neptune & other tooling:

**APIs:** RDF4J, Jena, RDFLib, Data Lens

**ETL:** Data Lens, data.world, AWS DMS, Amazon Comprehend

**Reasoning:** RDFox, various RDF libraries

**UI:** Metaphactory, Pool Party, Rhizomer, LinkedDataHub, Neptune workbench (Jupyter), Amazon Lex, Tom Sawyer

**Editing:** Gra.fo, Metaphactory, Pool Party

**ML:** Amazon Sagemaker, AWS DGL

The above are examples. We are always interested in discovering new tools and services that work with knowledge graphs.

# Amazon Neptune and friends

We complete the architecture using Neptune & other tooling:

**APIs:** RDF4J, Jena, RDFLib, Data Lens

**ETL:** Data Lens, data.world, AWS DMS, Amazon Comprehend

**Reasoning:** RDFox, various RDF libraries

**UI:** Metaphactory, Pool Party, Rhizomer, LinkedDataHub, Neptune workbench (Jupyter), Amazon Lex, Tom Sawyer

**Editing:** Gra.fo, Metaphactory, Pool Party

**ML:** Amazon Sagemaker, AWS DGL

**Let's talk about these...**

The above are examples. We are always interested in discovering new tools and services that work with knowledge graphs.

# Software patterns and abstractions (or: Triples...? Who cares about triples...)

Many libraries and APIs present a very low-level abstraction of graphs

Examples:

- RDF libraries mostly deal with triples (aka “*assembly language for graphs*”)
- SPARQL SELECT results are just tuples (yet another data model)

How do you bridge the gap between queries, query results, and your code?

- queries present yet another data model
- why is it all about projections: query results always tend to be tuples (true of other kinds of databases too)
- ORM: tuples → objects, is this a good model? (short answer: no)

# Software patterns and abstractions: objects

What about graph → objects?

- in the past we have tried graph → trees (e.g., RDF/XML, JSON-LD), that often leads to confusion
- metamodel mismatch
- (also note: syntax does not matter)

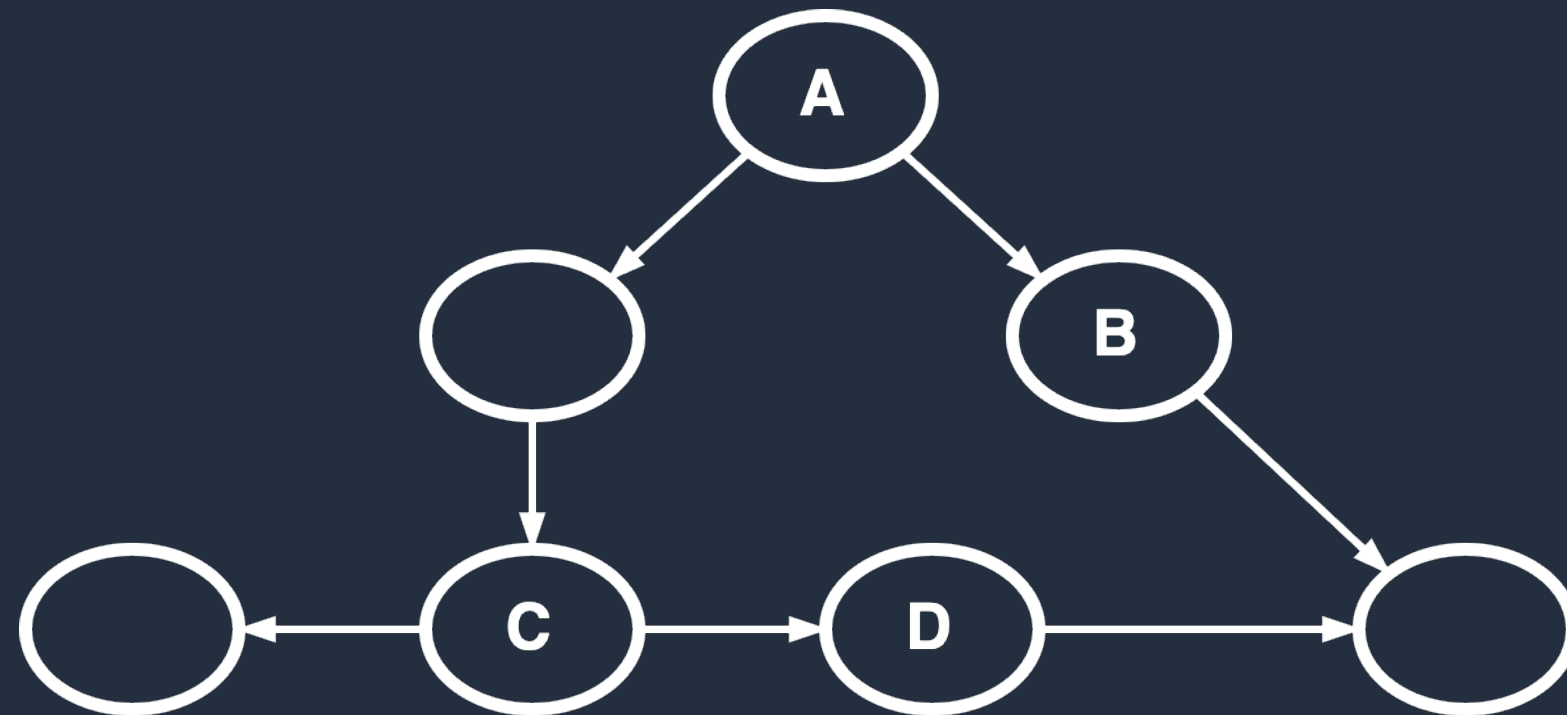
How would you decide how to map a graph onto a set of objects?

What, exactly, constitutes an object?

- RDF Concise Bounded Description is a good candidate
- (there are others, too)
- blank nodes present a problem (at least) when updating

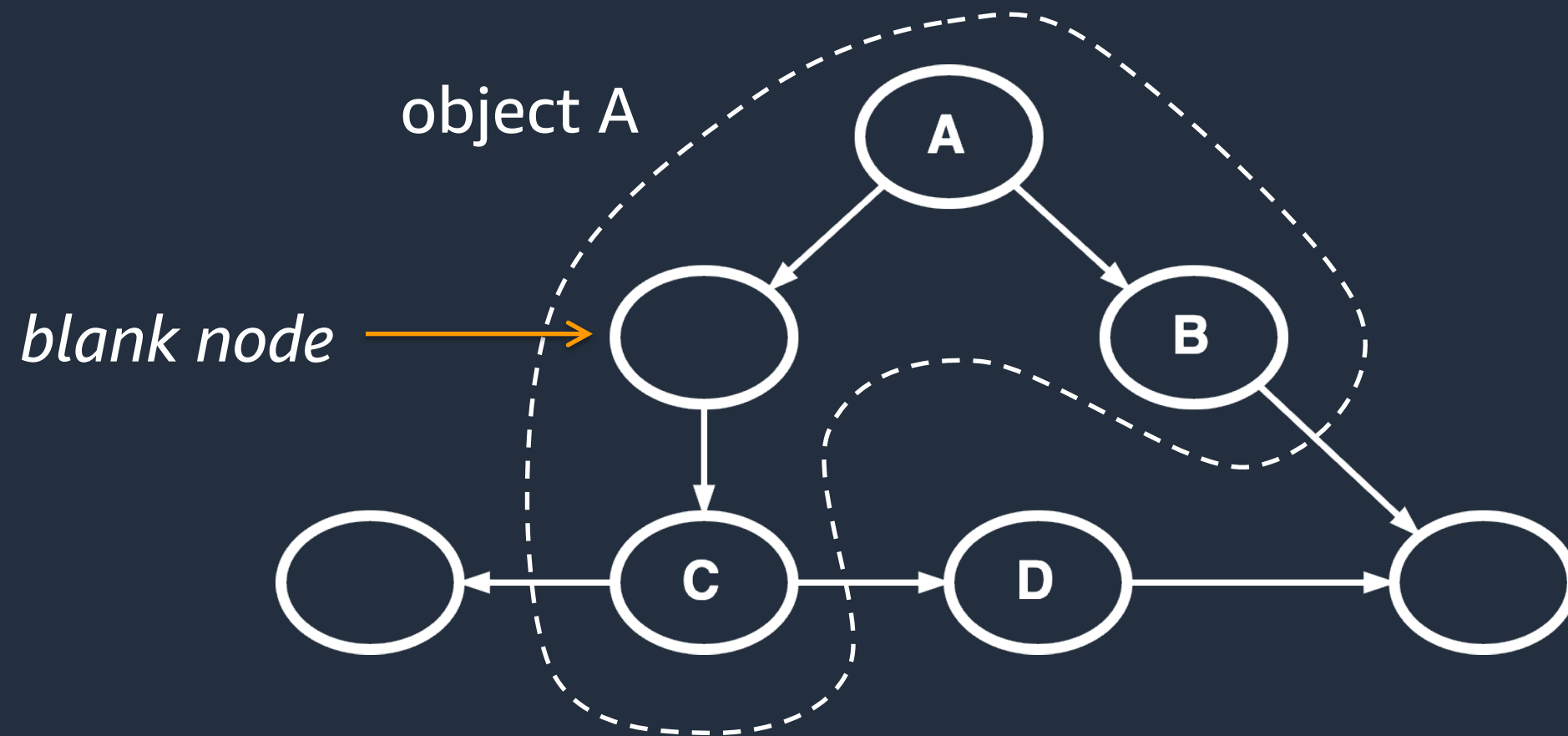
# Software patterns and abstractions: objects

Illustrative example of graph → objects



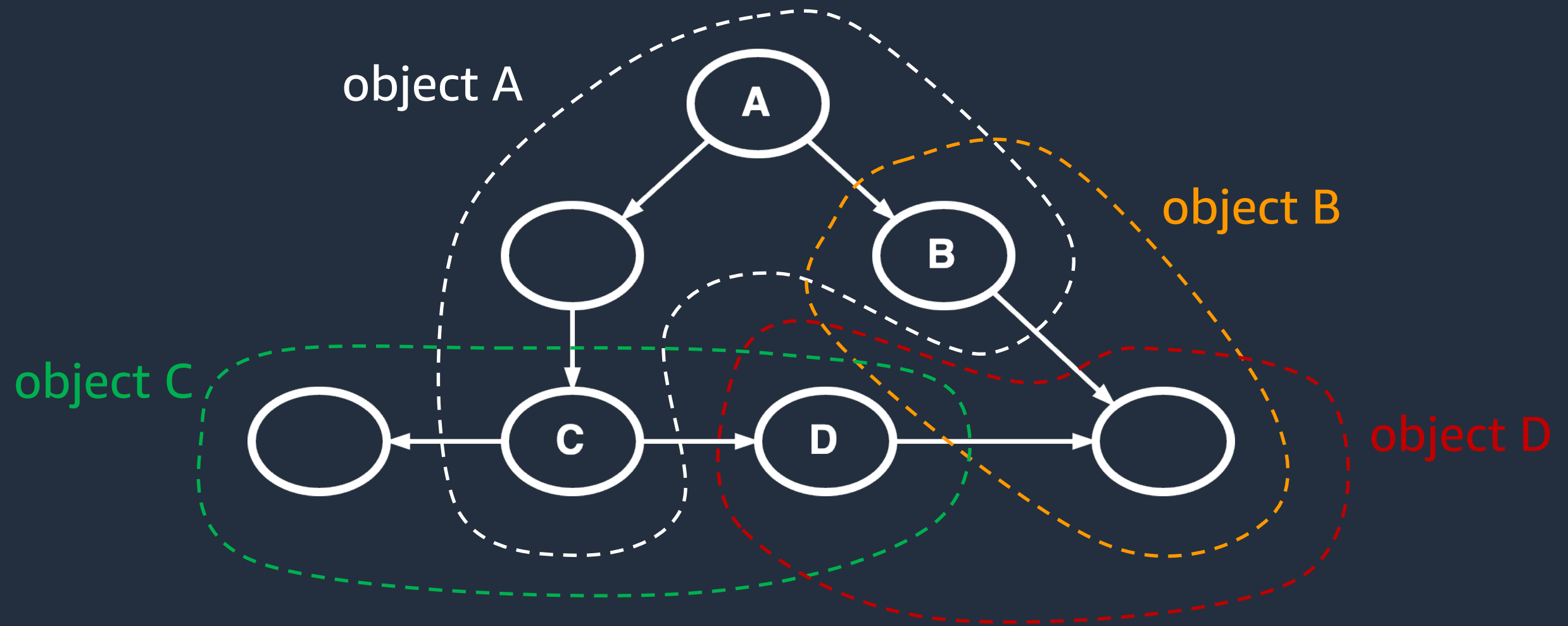
# Software patterns and abstractions: objects

Illustrative example of graph  $\rightarrow$  objects



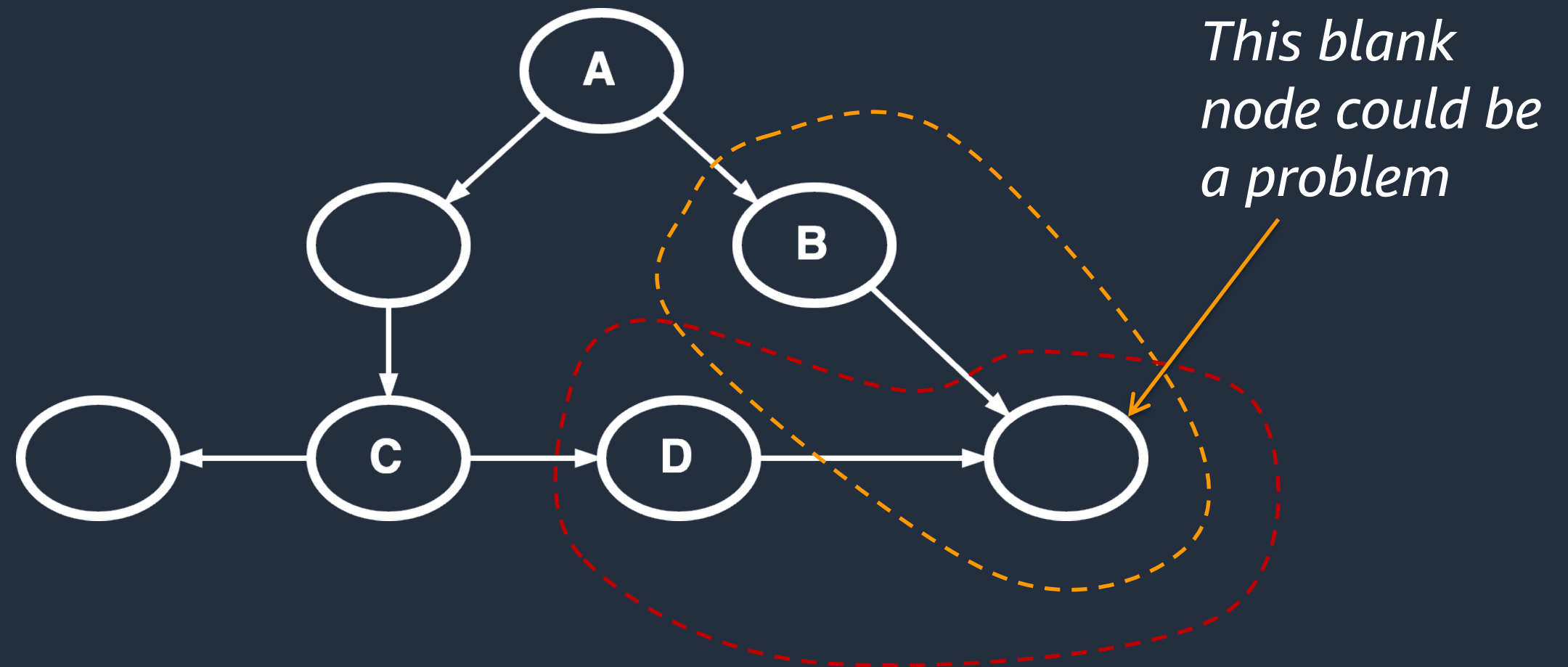
# Software patterns and abstractions: objects

Illustrative example of graph  $\rightarrow$  objects



# Software patterns and abstractions: objects

Illustrative example of graph → objects



# Software patterns and abstractions: APIs

Is GraphQL the answer to the “object access problem”?

- results are still trees; object references, blank nodes can be an issue

We may need simpler APIs:

1. discovery (search)
2. list of filtered things (akin to faceted search)
3. object access: *“give me everything you know about X”*
4. path reachability or *“where can you get from here?”* [Lassila 2007]

# Next steps (or: what are the real pain points?)

How do I write software that leverages a knowledge graph?

- bridging the “graph  $\longleftrightarrow$  code gap”
- low-level interfaces (lots of boilerplate code, etc.)

How do I let users interact with a knowledge graph?

- now we have very expressive data, but UI “expressiveness” lags behind

I have mapped my data to an ontology... now what?

(or: what does it mean to “support ontologies?”)

- role of reasoning
- “hiding” reasoning works well
- transitive taxonomies & owl:sameAs (including inverse functional properties)

# Summary

1. Knowledge graphs are not a single, narrow use case
2. Knowledge graph architectures reflect a data "value chain"
3. To build a knowledge graph solution, you need a graph database and many other components and tools
4. Software abstractions for graphs tend to be too low level
5. We may need to rethink how we want to access graph data

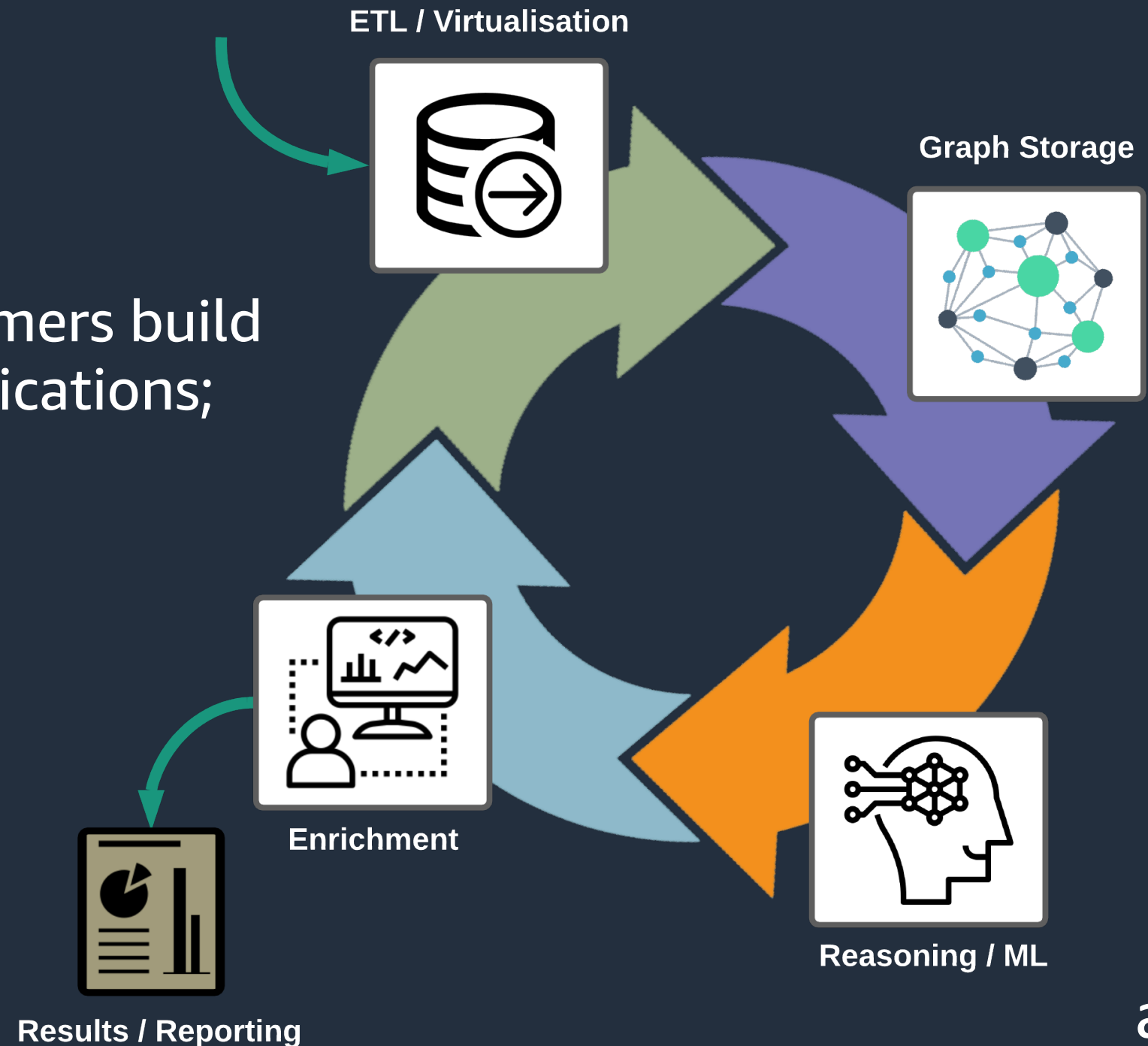
*but there is hope...*

# Summary

The virtuous cycle really works

There is a lot of tooling available

We have helped numerous customers build successful knowledge graph applications; it can be done!



# Questions?

**Contact:**      ora@amazon.com  
                     charivie@amazon.com

**More info:**    <https://aws.amazon.com/neptune/knowledge-graphs-on-aws/>  
                     <https://aws.amazon.com/marketplace>