



# All the World's Data as RDF?

## Towards a Unified Language for Data

*Keynote Address, US2TS 2022, September 2022*

**Dr. Ora Lassila**

Principal Technologist  
Amazon **Neptune**

# This is an **aspirational** talk

Think of it as a “call to action”...

I recently gave talk where I reminded people that

- the “data industry” has **serious issues**
- the Semantic Web technologies are **mature**
- we need a “**unified language for data**” (and we know what the best candidate is)

*(The above seem to be a recurring theme in my life.)*

# I have spoken about this before 🤯

aws

## Will **knowledge graphs** save us from the mess of modern data practice?

Dr. Ora Lassila  
Principal Technologist  
Amazon **Neptune**

© 2022, Amazon Web Services, Inc. or its Affiliates.

aws

## On the broad applicability of **Semantic Web** technologies (a personal journey)

Dr. Ora Lassila  
Principal Technologist  
Amazon **Neptune**

© 2021, Amazon Web Services, Inc. or its Affiliates.

aws

## Graph? Yes! Which one? Help!

Ora Lassila, Michael Schmidt, Brad Bebee, Dave Bechberger, Willem Broekema, Ankesh Khandelwal, Kelvin Lawrence, Ronak Sharda, and Bryan Thompson

## Love Thy Data (or: Apps Considered Harmful)

Dr. Ora Lassila  
Principal Technologist  
Cloud Analytics Team  
Nokia Location & Commerce

2012-06-11  
Elected Member  
Advisory Board  
World Wide Web

CIDOC2012  
HELSINKI • FINLAND

## Digital Permanence (or: Apps Considered Harmful – Again)

Dr. Ora Lassila  
Technology Architect  
Pegasystems, Inc. (Cambridge, MA, USA)

2015-11-23

## Size does not matter (if your data is in a silo)

Dr. Ora Lassila  
Principal Technologist  
Cloud Analytics Team  
Nokia Location & Commerce

2012-11-11  
Elected Member  
Advisory Board  
World Wide Web Consortium (W3C)

**ISWC** 2012  
The 11th International Semantic Web Conference

Semantic Technologies meet Recommender Systems & Big Data (SeRSy 2012)

NOKIA  
Connecting People

# THE SEMANTIC WEB

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by  
TIM BERNERS-LEE,  
JAMES HENDLER and  
ORA LASSILA

PHOTOILLUSTRATIONS BY MIGUEL SALMERON

SCIENTIFIC AMERICAN 35

# Game plan



# Issue #1: Semantics

(or lack thereof...)

# Issues with modern data practice

Old approach:

- applications as guardians of data; silos everywhere
- moving and sharing data difficult

So-called “modern” approach:

- physical data moves better, but semantics doesn't

Why do we think that it is a good idea to separate data from its meaning?

- most folks don't really understand the concept of **semantics**
- confusion between formal semantics and human interpretation of the meaning of data

# Human interpretation vs. formal semantics (demonstrated in the context of JSON)

“JSON is easy to understand”

- no, it isn't (you just think it is)

“My data model is JSON”

- JSON is **not a data model**, and mind you, JSON has **no semantics**

“My data is just JSON”

- your data is never “just JSON”, you **always impose external semantics**

# State of data modeling today

Does not seem to be a priority (although I see some positive developments)

Data models are often seen just as a means to be able to “program with data”

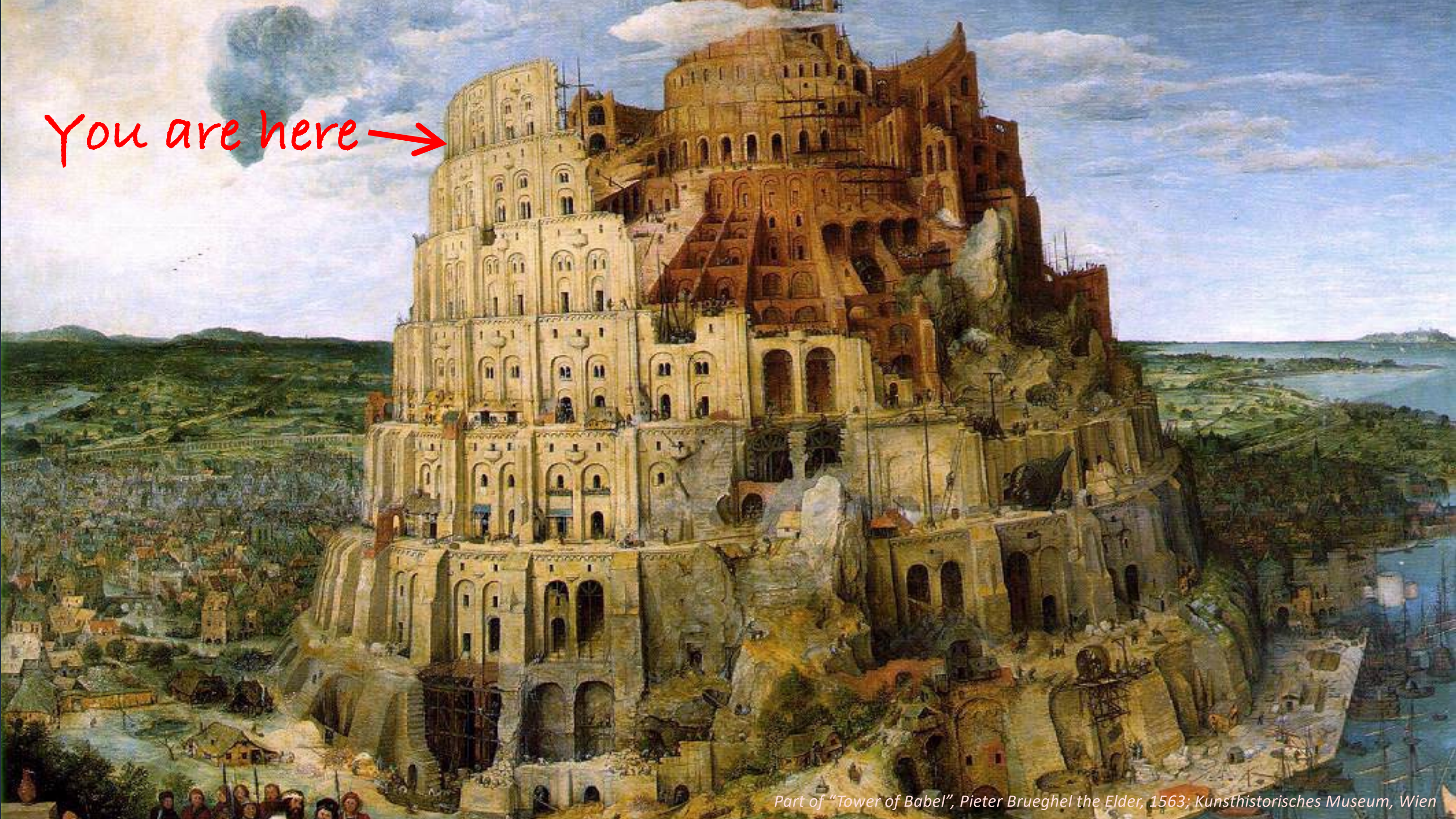
I am not convinced proper modeling is taught enough today

# APIs...?

# APIs...?

# No

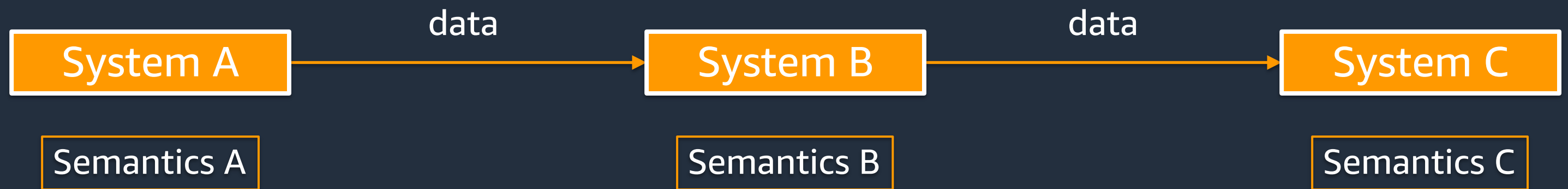
You are here →



Part of "Tower of Babel", Pieter Bruegel the Elder, 1563; Kunsthistorisches Museum, Wien

# This is what we have

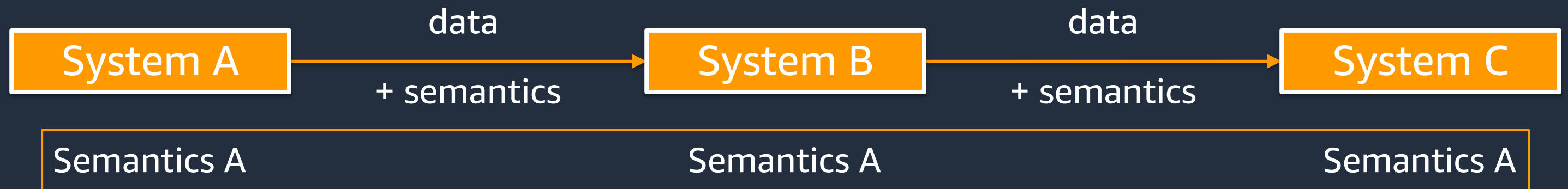
Data moves, but every system has its own idea of semantics



Redefining or “re-articulating” semantics over and over not only is error-prone, it is a lot more work!

# This is what we should have!

Data moves **with** semantics!



What do we need to get there?

# Issue #2: The new <sup>graphs</sup> ~~kids~~

# “Rumors of my demise are grossly exaggerated”

With the advent of Labeled Property Graphs (LPGs), some people might think that the Semantic Web is dead

But LPGs are **not a replacement** for RDF

- Use RDF to build logical models/representations of your data
- LPGs are a literal data structure your program manipulates
- Some folks treat RDF as a mere data structure

good!

uh... OK

yeah, NO!

Note: RDF's design is very deliberate **from the KR standpoint**

# RDF can be hard to “program with”

Triples are a very low-level abstraction → we need something better

Composite types are cumbersome

Developers do not want to learn SPARQL

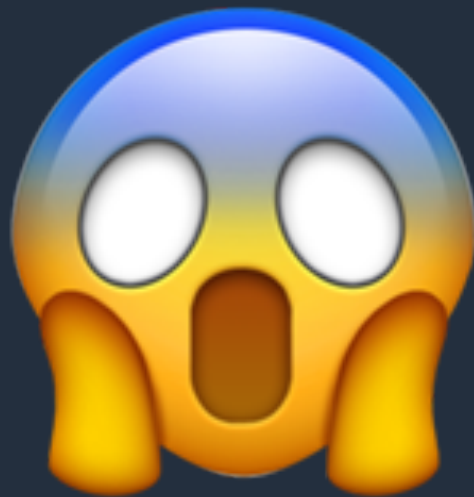
LPGs

- are close to the “developer mindset”
- have good programming abstractions and interfaces
- support “path finding” (nonexistent in SPARQL)

What about **user interfaces**? (Problem for LPGs and RDF alike)

# Code example:

Python code for modifying the items of a (potentially existing) RDF container



```
def setContainerItems(graph, node, predicate, values, newtype=RDF.Seq):
    if values:
        statements = getContainerStatements(graph, node, predicate)
        if statements:
            container = statements[0][0]
            for statement in statements:
                graph.remove(statement)
        else:
            container = BNode()
            graph.add((node, predicate, container))
            graph.add((container, RDF.type, newtype))
        i = 1
        for value in values:
            graph.add((container, URIRef(makeContainerItemPredicate(i)), value))
            i += 1
    else:
        container = getvalue(graph, node, predicate)
        if container:
            graph.remove((node, predicate, container))
            graph.remove((container, None, None))

def getContainerStatements(graph, node, predicate):
    containers = list(graph.objects(node, predicate))
    n = len(containers)
    if n == 1:
        return sorted([statement for statement
                       in graph.triples((containers[0], None, None))
                       if isContainerItemPredicate(statement[1])],
                      key=lambda tr: tr[1])
    elif n == 0:
        return None
    else:
        raise ValueError("Expected only one value for {0}".format(predicate))
```

# Amazon Neptune

Amazon Neptune is a **fast, reliable, fully managed graph database service**

Neptune supports **both RDF and LPGs**

- SPARQL (1.1 with update, federated queries, graph store HTTP protocol)
- Gremlin (Tinkerpop v3.5.2)
- openCypher (v9)

Customers can (and today have to) choose between the two graph models

<https://aws.amazon.com/neptune/>

# The rift: RDF vs. LPGs

Confuses users (when given the choice)

Should not matter (but today it does)

The rift hurts the industry and community as a whole

# Solutions?

# A graph is a graph is a graph



The Amazon Neptune team is working to mitigate the rift: **Project OneGraph**

*See this for more info:*

[semantic-web-journal.net/system/files/swj3273.pdf](https://semantic-web-journal.net/system/files/swj3273.pdf)

The solution is not easy

- RDF-star is a significant step in the right direction

If we succeed (and we will), there are benefits "on both sides":

- use all of the good features of RDF (and SPARQL) with LPGs, without having to reinvent them
- no more complaints that RDF does not have "edge properties" \*
- Gremlin queries over RDF!
- etc.

\* Caveat: we still need a semantic theory for these

# A graph is a graph is a graph



The real issue, however, is this:

See this for more info:  
[semantic-web-journal.net/system/files/swj3273.pdf](https://semantic-web-journal.net/system/files/swj3273.pdf)

Graph as a **logical representation** vs. graph as a **data structure**



# I want a unifying **logical language** for data!

We need a language to

1. represent data
  2. talk about data
- } **thinking about your data as a graph is natural and intuitive**

We need to be able to define and communicate semantics

Syntax does not matter

RDF & OWL are currently our **best candidate** for something like this

# Call to action

You may have figured out  
by now that this is a talk  
about the Semantic Web

# Why are Semantic Web technologies attractive?

1. Self-describing data with accessible semantics
  - remember: **accessible data = physical bits + semantics**
2. Data semantics not defined by code & applications
3. Procedural → declarative
4. Serendipity
5. Graph-based representation is intuitive
6. Based on well understood technologies and infrastructure

*NB: Outside popular non-symbolic AI methods, the Semantic Web technologies are the embodiment of what we wanted to do before the “AI winter”*

# How do we get there?

Small steps...

# Today: mapping existing data to RDF

Non-RDF  
models

RDF & OWL  
models

Logic in  
code

Logic in  
data

Why? Because now you could integrate various data sources and write queries across all of them

Mapping to RDF

*all\**

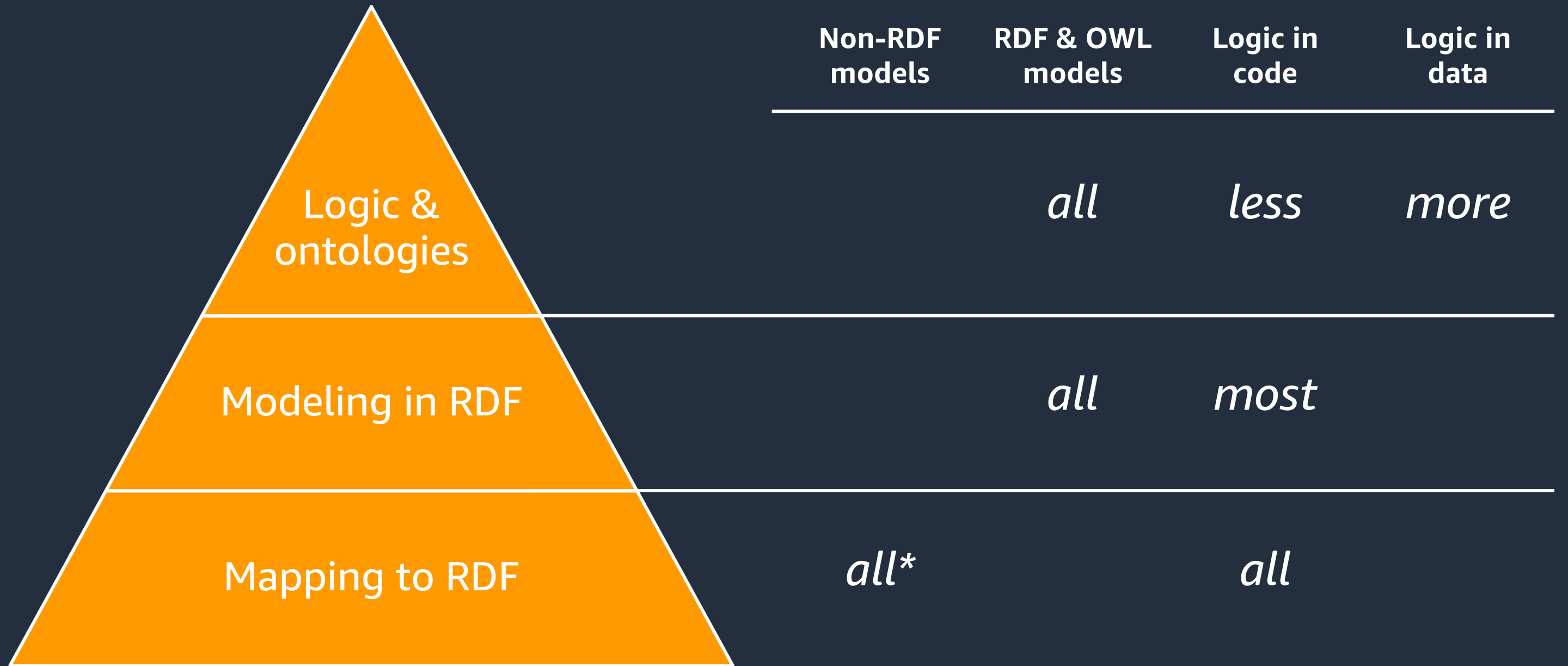
*all*

# Tomorrow: using RDF “natively”

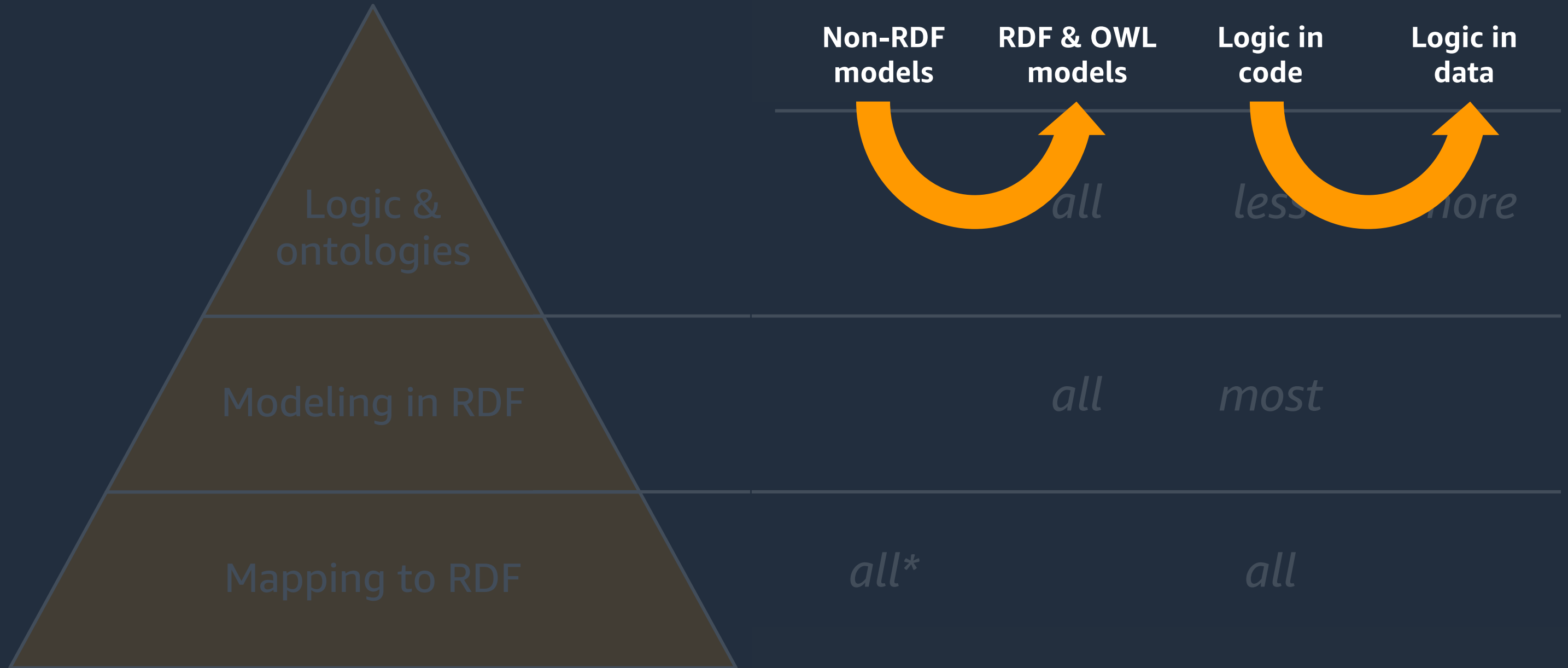
Why? Because now you could start sharing not just your data but also models (semantics)

	Non-RDF models	RDF & OWL models	Logic in code	Logic in data
Modeling in RDF		<i>all</i>	<i>most</i>	
Mapping to RDF	<i>all*</i>		<i>all</i>	

# Some day: more of logic moves to ontologies



# I want to see these trends



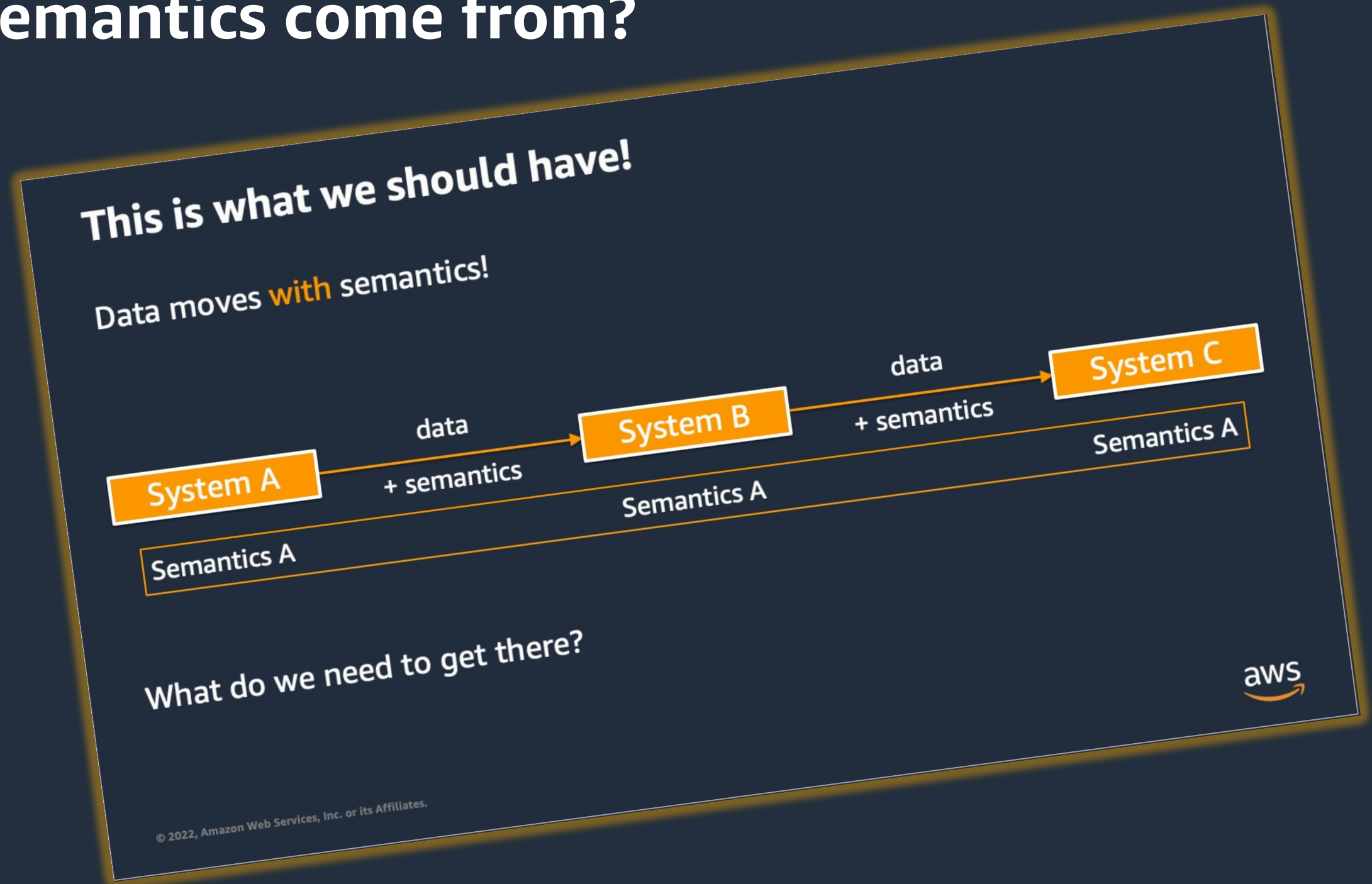
# I want to see these trends

...but they will not manifest unless we

- teach people how to do it
- make it easier to use RDF and OWL in software development
- stop people from wondering whether LPGs would still be a better (and that they really don't need RDF at all...)

# Where does semantics come from?

Remember this?



# Where does semantics come from?

We need a practical, not a philosophical view of semantics and ontologies

## 1. Semantics:

defines how data “behaves” and how software can interpret data

## 2. Ontologies:

means of communicating #1

# Where does semantics come from?

1. Relationships between data and their definitions (including constraints)
2. Relationships within data

But today, semantics is hard-wired in code, inside a black box

# Where does semantics come from?

1. Relationships between data and their definitions (including constraints)

Ontologies & logic

2. Relationships within data

Graph representation

~~But today, semantics is hard-wired in code, inside a black box~~

**procedural → declarative**

# Ontologies (my pragmatic view)

Ontologies exist for at least these reasons:

- operationalizing your data
- formally documenting how you understand your data

Different approaches:

1. Ontology patterns
2. Foundational ontologies (e.g. BFO)
3. Upper ontologies (but not #2)

*sometimes useful*

*no (re: operationalization)*

*yes! (esp. for reusable data)*

What does it mean for your tools to **support ontologies**?

# Make it easier to use RDF and OWL

Higher-level abstractions that better match programming constructs

Better support for composite data types

- OneGraph will give us this

Fix SPARQL

- Support for path finding (we may need composite datatypes for this)

I really want the open source community to invest some energy in reusable vocabularies and ontologies

# Stop people from thinking that LPGs would be better

Actually, if we succeed with OneGraph, we don't need to stop them

# To summarize...

## Semantics

- nobody knows what “semantics” means
- it needs to travel with data

## RDF vs. LPGs

- we are going to fix this
- real issue: logical models vs. concrete data structures

## Unified language for data

- procedural → declarative
- make RDF & SPARQL better for programmers

**This is now up to you**

# Thank you!

Contact:

- [ora@amazon.com](mailto:ora@amazon.com)
- Twitter: [@oralassila](https://twitter.com/oralassila)