Semantic Technologies Primer

Timothy "TK" Keanini, CTO

Introduction

- Tim “TK” Keanini, CTO   nCircle Inc.
  - Why I became interested in Interoperability and then the W3C Semantic Stack?
    - Supply-side problem
      - IT Security market is too fragmented
      - Companies will acquire or be acquired
        - » Same time to market problem
    - Demand-side problem
      - Customers all share a common requirement for multivendor interoperability (every vendor on the floor of RSA interoperating)
      - Syntax-level interoperability will not be sufficient
- Disclaimer and Objective
  - Nothing in my presentation is vendor specific
  - This session is a W3C Semantic Technologies Primer
    - Aspects that are essential to multi-vendor interoperability

SCAP Definitions and Goals

- NIST SP800-117 – "Guide to Adopting and Using Security Content Automation Protocol"
  - "Comprehensive & Standardized Approach"
    - …**organizing and expressing security-related information**…
  - Demonstrate compliance with security requirements
    - Provenance
  - **Content Interoperability across automated tools**
- NIST SP800-126 – "Specification for the Security Content Automation Protocol"
  - "…describes the basics of the SCAP **components** specification and **interrelationships**."
  - …**characteristics of SCAP content** and **SCAP level requirements** not defined in individual component specifications
  - ".. to achieve **security automation**…"

We are off to a great start!

- We have common Names and Languages (syntax level)
- We have a common method of ranking vulnerabilities (members within that set)
- We have a call-to-action for software developers to provide benchmarks for their “platform”
- Today we have a common repository for content (NVD)

Interoperability challenges to address (IMHO)

- Syntax Interoperability versus Semantic Interoperability
  - Heterogeneous IT && Heterogeneous Viewpoint
  - Regex’able versus Inference
- Semantic Interoperability intra and inter SCAP
  - SP-800-126 does provide some semantic framework, it would benefit greatly from the machine-readability of RDF/RDFS/OWL
  - The horizontal nature of security and compliance demands support for heterogeneous viewpoints.
- Knowledge Representation Problem/Opportunity
  - XML/XML-Schema/XSLT are useful and stable
  - RDF/RDFS/OWL/SPARQL
    - Leverages what we already know
    - Use only what you need
    - W3C technologies interoperability

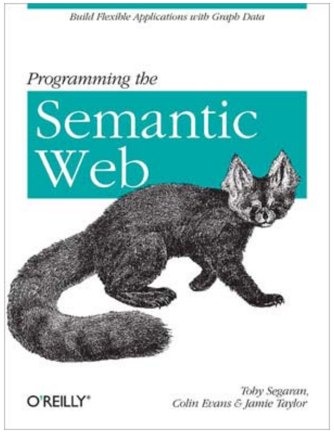
What is an Ontology?

"An ontology provides a **precise vocabulary** with which knowledge can be represented"

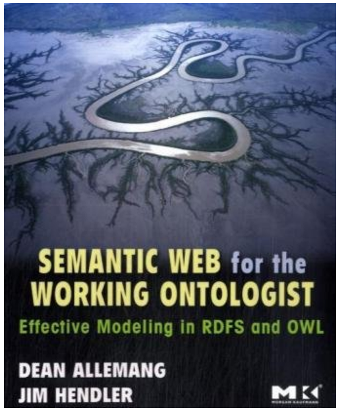
"This vocabulary allows us to specify which **entities** will be represented, how they can be **grouped**, and what **relationship** connect them together"

"…a **social contract** between a data provider and a data consumer"

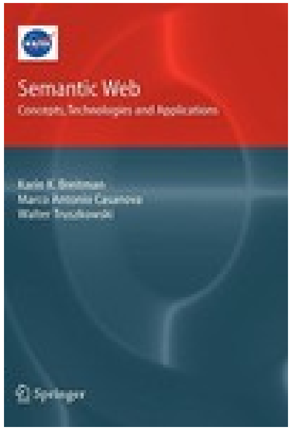
For the programmer



For the model design



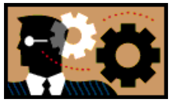
For the techie



nCircle

Quick FAQ

- Do I need to become an Ontologist?
  - Too late, you are already one.
  - We have all done some degree of data modeling
  - We have all had to choose a way to represent our knowledge
- Correctness?



  - Unlike syntax correctness, an ontology is judged by the usefulness of the inferences that can be drawn from the model
  - We know an ontology is right when the communities using the system are able to interoperate at the semantic level
- Can’t I just use XML
  - Semantic Data can be encapsulated in XML
  - XML is based on a document model, Semantic Data is based on a graph

Meaning and what may be inferred

- External agreement on meaning of annotations
  - E.g., Original XML Dublin Core
    - Agree on the meaning of a set of annotation tags
  - Challenges
    - Inflexible
    - Limited number of things can be expressed
- Use Ontologies to specify meaning of annotations
  - Ontologies provide a vocabulary of terms
  - New terms can be formed by combining existing ones
  - Meaning (semantics) of such terms is formally specified
  - Can also specify relationships between terms in multiple Ontologies (relationships intra and inter SCAP)

Myths about Ontologies and Semantic Web

- Semantic Technologies are only about the Web
  - False: Semantic Modeling is about Knowledge Representation
- Semantic Technologies are unrelated to XML
  - False: It pick up where XML leaves off.
- Ontologies are too complex to understand or use
  - False: It can be only as complex as it needs to be. Use what you need.
- Ontologies and Taxonomies are the same
  - False: Taxonomies only allow for parent-child relationships
    - Ontologies are much more expressive and dynamic than Taxonomies
- Ontologies are difficult to create and change all the time
  - False: It is just another language to help model your domain
  - False: change happens! It offers a robust language for versioning
- W3C standards are the only way to perform semantic modeling
  - False: but the interoperability goals of the W3C show great potential

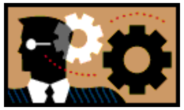
Absurdities

- Machine understanding on par with human understanding
- Describe all of the aspects of the observable world
- Create sentient machines
- It is the silver bullet for all of SCAP

nCircle

Value Proposition

- The value of W3C standards
  - We already experience this through XML
  - Namespaces, Identifiers, Expressivity, Extensibility
- The value of a graph-based model
  - Interoperability
  - Federation and Merging are “free”
  - Multi-perspective and viewpoints without compromise

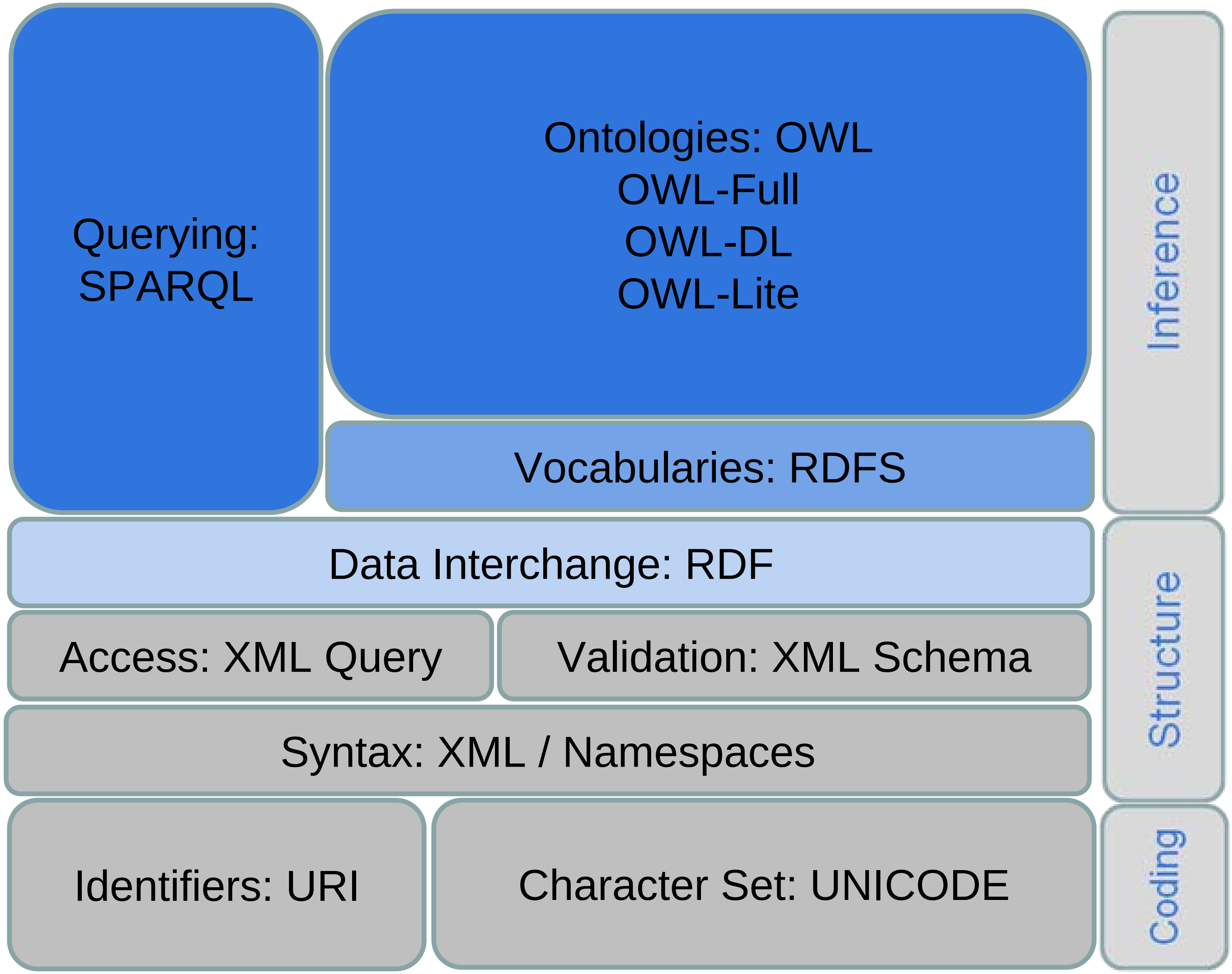


- The value of inference
  - Given some initial information, the following new information can be derived
  - Provides new insight
  - Able to draw useful inferences specific to your domain
  - Ability to draw inferences from another domains viewpoint

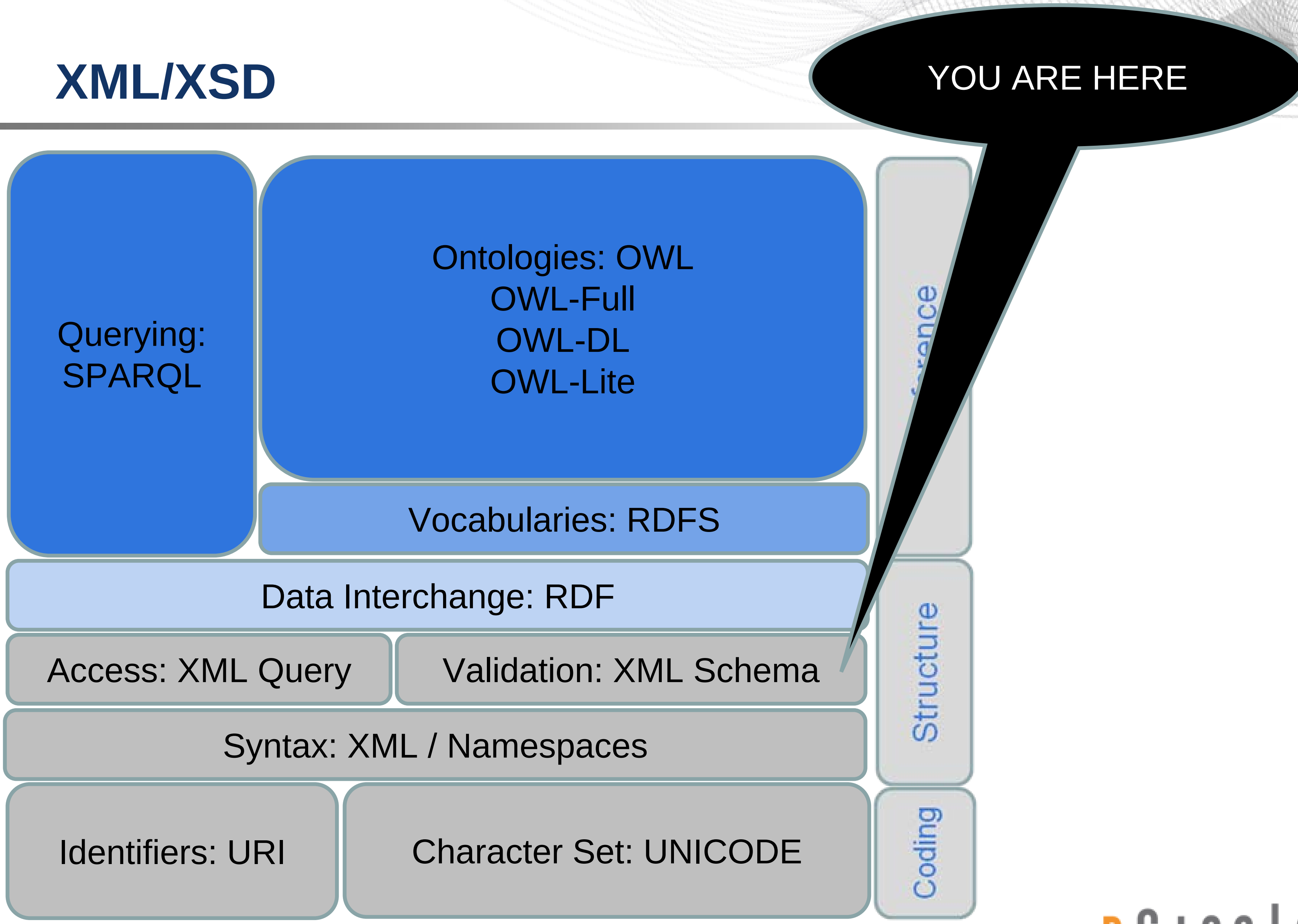
nCircle

W3C Semantic Technologies

W3C Semantic Technology Stack



XML/XSD



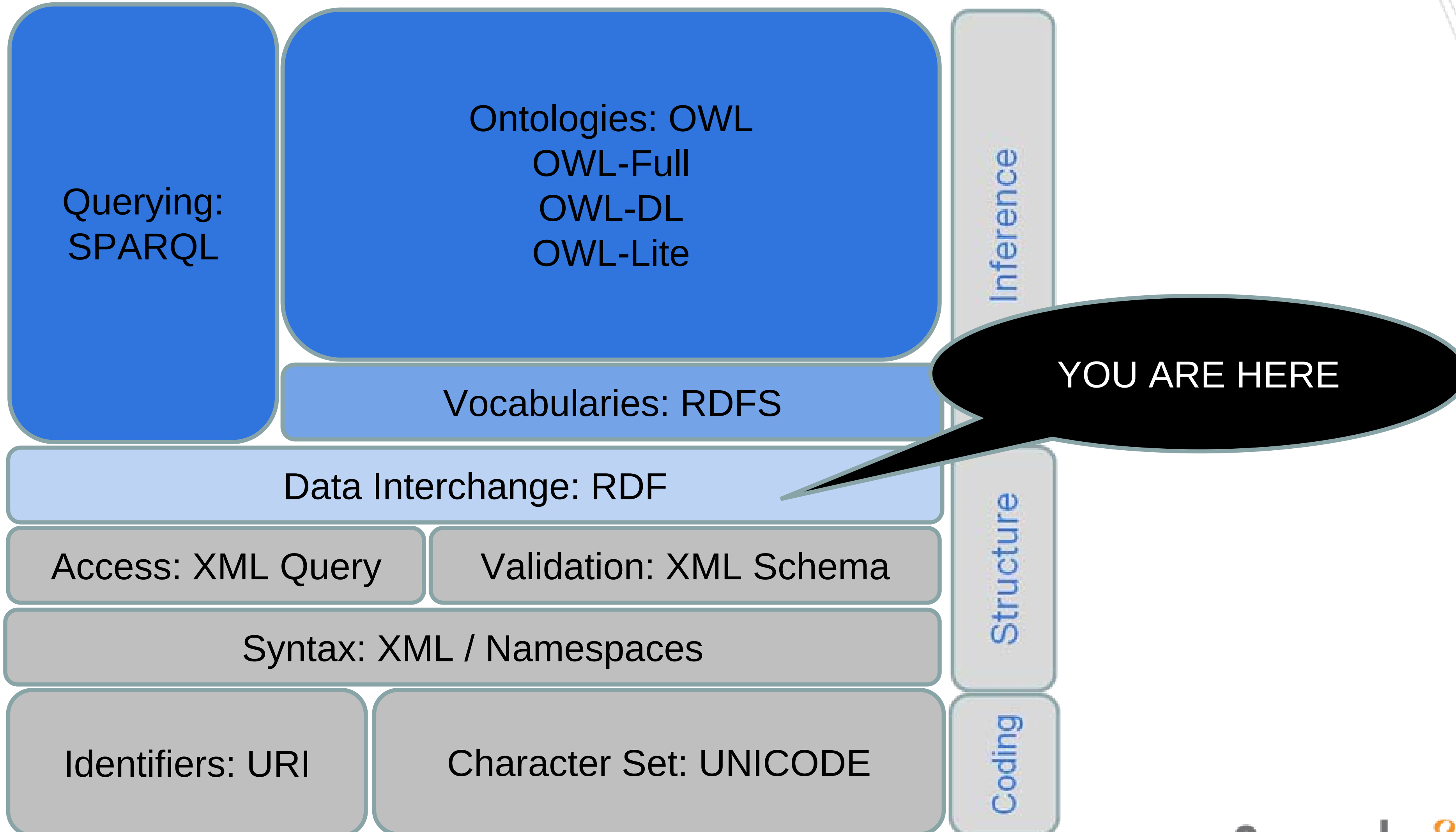
YOU ARE HERE
Querying: SPARQL
Ontologies: OWL
OWL-Full
OWL-DL
OWL-Lite
Vocabularies: RDFS
Data Interchange: RDF
Access: XML Query
Validation: XML Schema
Syntax: XML / Namespaces
Identifiers: URI
Character Set: UNICODE
Structure
Coding

Where are we today? XML/XML Schema

- General purpose markup-language to describe structured documents
- Tree-like syntax for tree-structured data
- Like a taxonomy, terms are classiffied hierarchically
  - Limited to generalization, is-a, type-of, parent-child, etc
  - From general to more specific concepts
- XML Schemas support explicit *application-specific* structures, cardinality, and datatyping constraints. Example:
  - "title is mandatory"
  - "date must be after 1980"
  - "title must be a string"
  - "there can be no more than three titles“
- Infrastructure for serialization and data-level policy

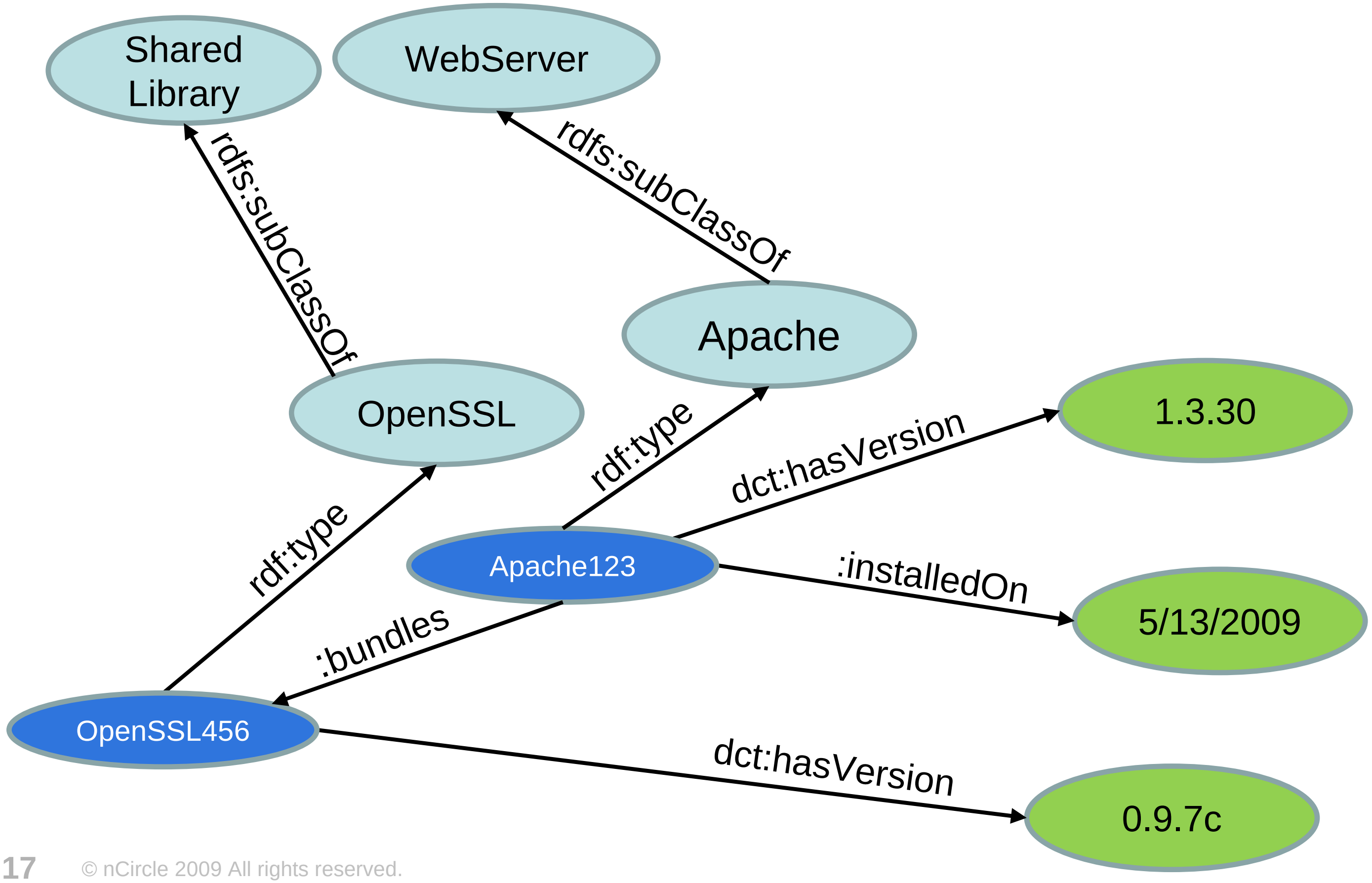
nCircle

RDF – Resource Description Framework



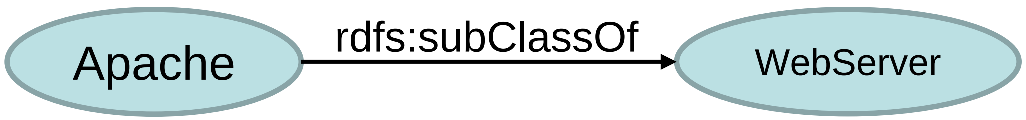
RDF – Labeled-Directed Graph

- Data Model is a 'labeled-directed graph'
  - All nodes and arcs have some type of label (identifier)
  - Arcs point only in one direction



RDF – Statements in the form of a *triple*

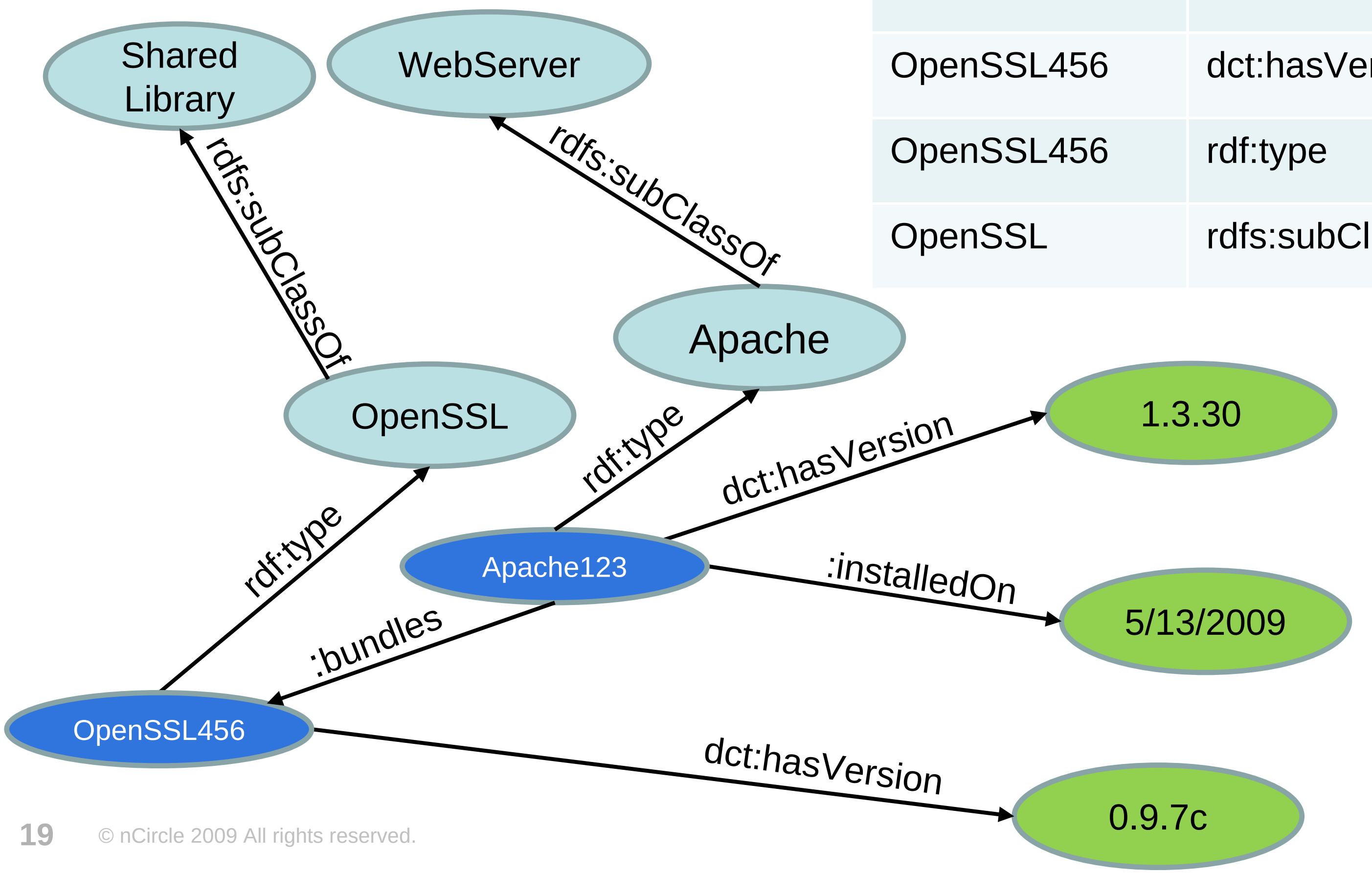
- All statements in the form of a triple
  - Subject-Predicate-Object (S,P,O)
  - Set of these triples begin to model a domain in the form of a graph



Subject (S)	Predicate (P)	Object (O)
Apache	rdfs:subClassOf	WebServer
Apache123	rdf:type	Apache
Apache123	dct:hasVersion	1.3.30
Apache123	:installedOn	05/13/2009
Apache123	:bundles	OpenSSL456
OpenSSL456	dct:hasVersion	0.9.7c
OpenSSL456	rdf:type	OpenSSL
OpenSSL	rdfs:subClassOf	SharedLibrary

RDF – Graph Model

Subject (S)	Predicate (P)	Object (O)
Apache	rdfs:subClassOf	WebServer
Apache123	rdf:type	Apache
Apache123	dct:hasVersion	1.3.30
Apache123	:installedOn	05/13/2009
Apache123	:bundles	OpenSSL456
OpenSSL456	dct:hasVersion	0.9.7c
OpenSSL456	rdf:type	OpenSSL
OpenSSL	rdfs:subClassOf	SharedLibrary



RDF – Different Syntax

- How one would express:
  - Apache is a member of the set Webserver
- RDF/XML

```
<rdf:Description rdf:about="#Apache">
   <rdf:type rdf:resource="#Webserver"/>
 </rdf:Description>
```

- N3

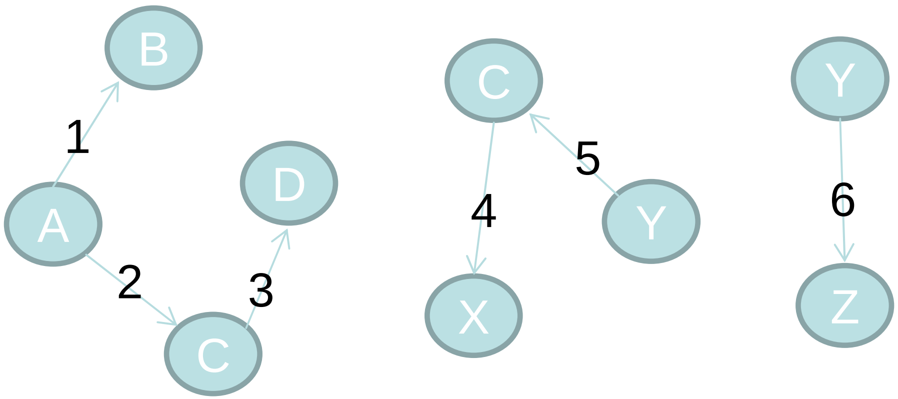
```
:Apache   rdf:type    :Webserver .
:Apache   a   :Webserver .
```

- RDF/XML-ABBREV

```
<Webserver rdf:ID="Apache"/>
```

- SeeAlso: TURTLE and N-TRIPLE

RDF – Merging and Federation



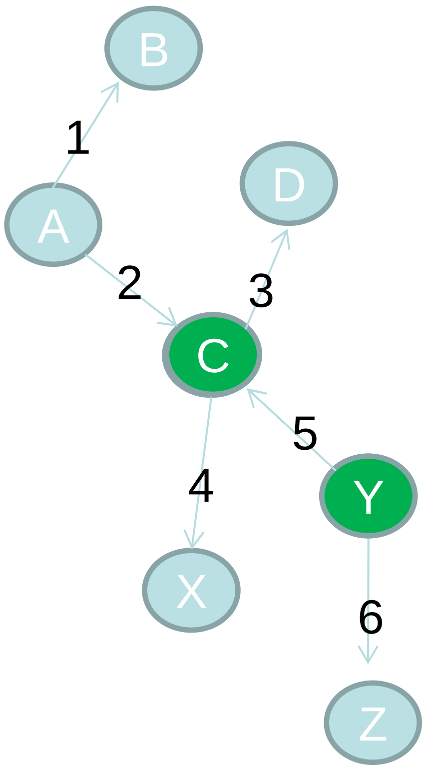
S	P	O
A	1	B
A	2	C
C	3	D

S	P	O
C	4	X
Y	5	C

S	P	O
Y	6	Z

=

S	P	O
A	1	B
A	2	C
C	3	D
C	4	X
Y	5	C
Y	6	Z

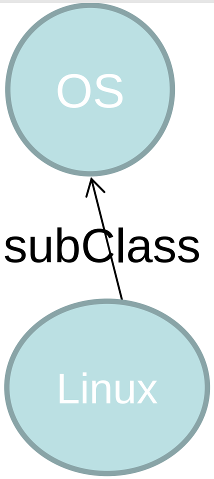


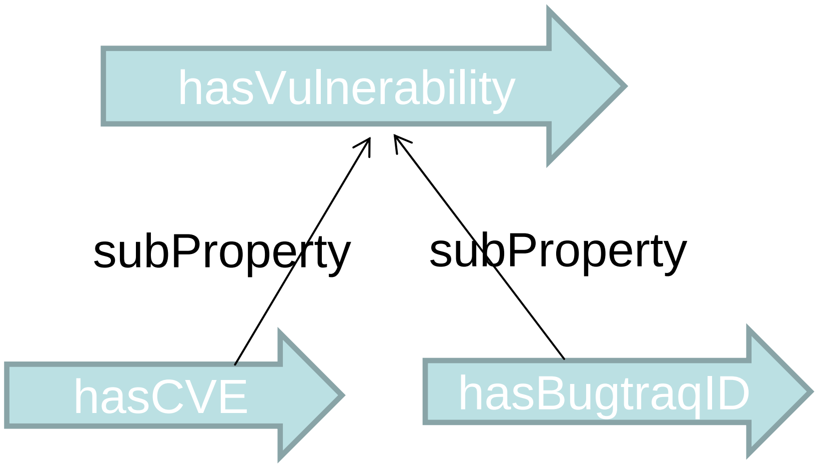
nCircle

RDF - Nodes and Arcs are first-class entities

If X is a member of the Set Linux;
Then X is a member of the Set OS;

If A hasCVE B;
Then A hasVulnerability B;





Assertion: RedHat rdf:type Linux

Inference: RedHat rdf:type OS

Assertion: OpenSSL_0.9.7c hasCVE CVE-2004-0112

Inference: OpenSSL_0.9.7c hasVulnerability CVE-2004-0112

RDF – Common Terms

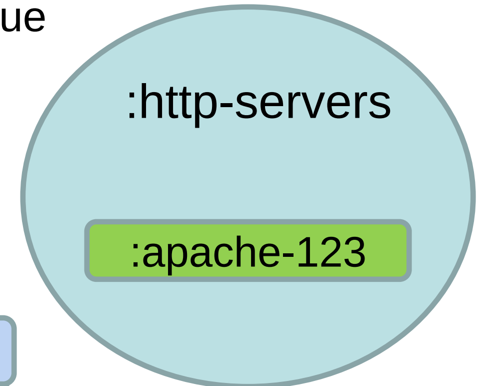
Common Term	Synonyms
Resource	Subject, Object
Resource identifier	Name, URI, ID, identifier, URL, label
Statement	Triple, statement, assertion
Subject (S)	Source, resource, “row”, node
Predicate (P)	Property, “column”, arc, relationship
Object (O)	“value”, resource, literal, node
RDF Store	Triple Store, Graph Database

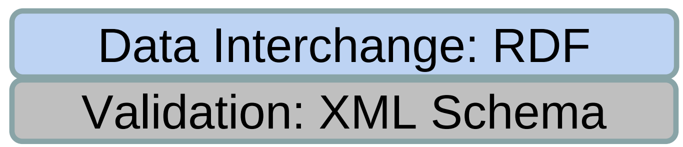
RDF - Vocabulary

rdf:type rdf:Property rdf:XMLLiteral rdf:nil rdf:List rdf:Statement rdf:subject rdf:predicate rdf:object rdf:first rdf:rest rdf:Seq rdf:Bag rdf:Alt rdf:_1 rdf:_2 ... rdf:value

Membership Assertion

```
:http-servers rdf:type rdfs:Class
:apache-123 rdf:type  :http-servers
```





Externally Defined Datatypes

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
```

<http://www.labor.gov/> baf:hasName "US Department of Labor"^^xsd:string .

:theUniversalQuestion :hasAnswer “42”^^xsd:int .

XSD datatypes

xsd:string, xsd:boolean, xsd:decimal, xsd:float, xsd:double, xsd:dateTime, xsd:time, xsd:date, xsd:gYearMonth, xsd:gYear, xsd:gMonthDay, xsd:gDay, xsd:gMonth, xsd:hexBinary, xsd:base64Binary, xsd:anyURI, xsd:normalizedString, xsd:token, xsd:language, xsd:NMTOKEN, xsd:Name, xsd:NCName, xsd:integer, xsd:nonPositiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:nonNegativeInteger, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:positiveInteger

RDF – Resource Description Framework

- Reification: Reifying Relationships
  - A Statement about a Statement (S, P, (S,P,O))
  - When a statement was made, who made the statement, some start of authority, chain of custody, etc
  - Sometimes referred to as *Provenance*
- Statement and Reification

```
– :App123 cve:isVulnerableTo cve:CVE-1999-0067 .
Reification
    – foo:AppVuln456  rdf:type  rdf:Statement .
    – foo:AppVuln456  rdf:subject :App123  .
    – foo:AppVuln456  rdf:predicate cve:isVulnerableTo  .
    – foo:AppVuln456  rdf:object cve:CVE-1999-0067  .
– vendor:scanner22  cve:discovered  foo:AppVuln456  .
```

Table versus Graph Data Model

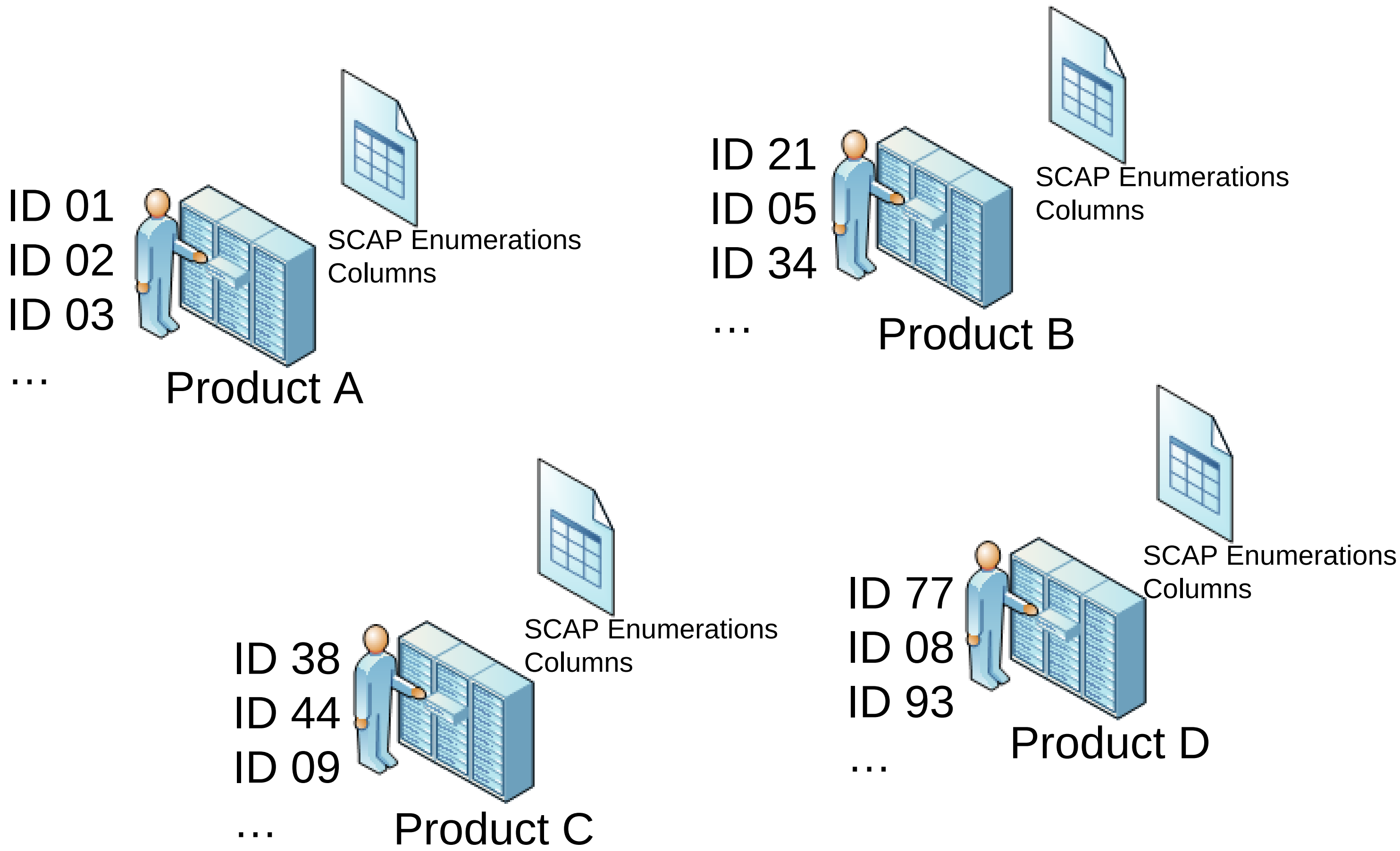
Subject | Predicate | Object

Row, Column, Value == S,P,O

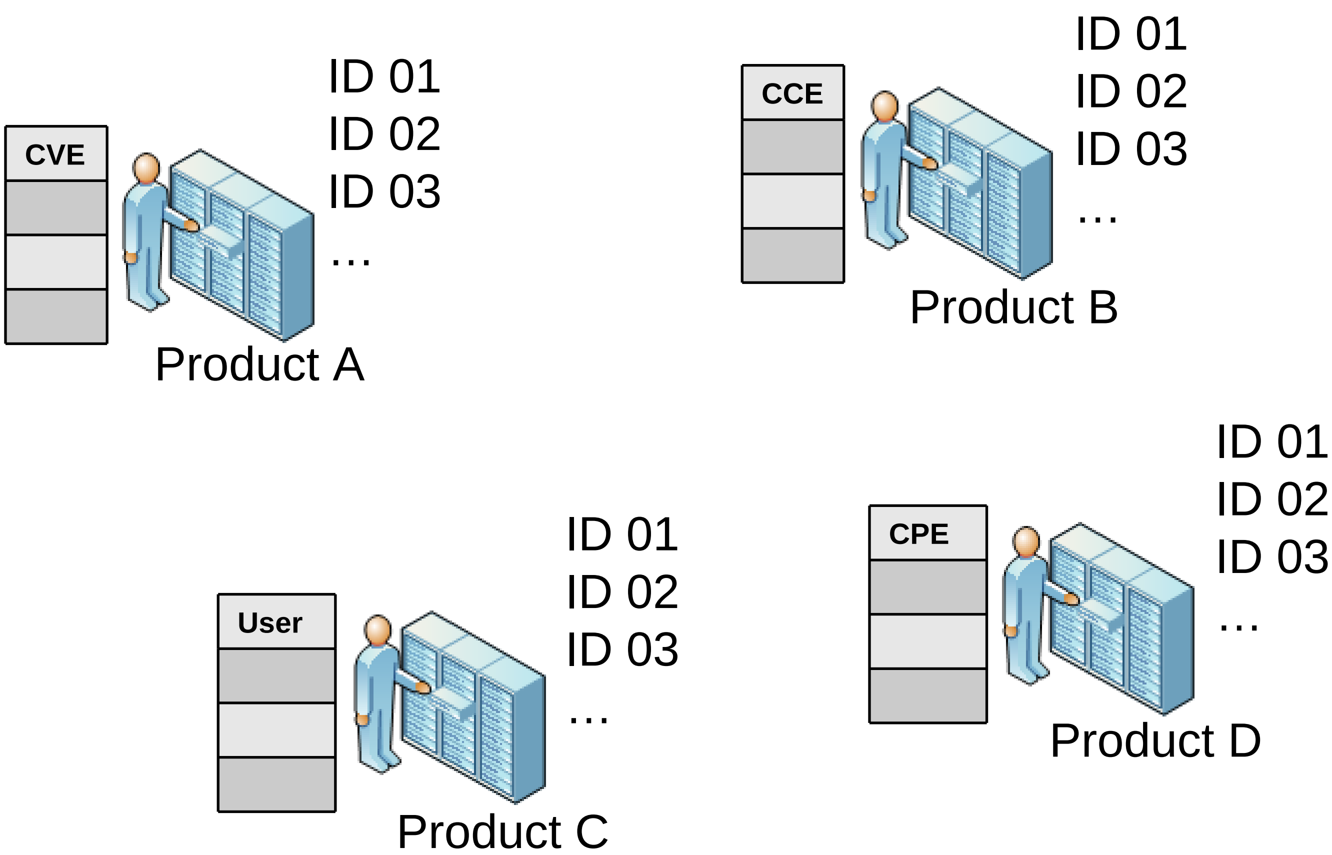
ID	hasOnline Account	End-Point_Address	hasCVE	hasCCE	Business Function
01	Alice	10.20.10.11/32	CVE-1999-888	CCE-2002-787	eCommerse
02	Alice	10.20.10.10/32	CVE-2001-234	CCE-2005-345	Supply Chain
03	Bob	10.20.10.11/32	CVE-2002-444	CCE-2006-666	Supply Chain
04	Bob	10.20.10.12/32	CVE-2004-555	CCE-2002-222	Supply Chain
05	Bob	10.30.10.10/32	CVE-2006-111	CCE-2002-322	Back Office
06	Carol	10.40.10.10/32	CVE-2006-234	CCE-2007-999	Back Office
06	Carol	10.50.10.10/32	CVE-2007-777	CCE-2007-111	HR

nCircle

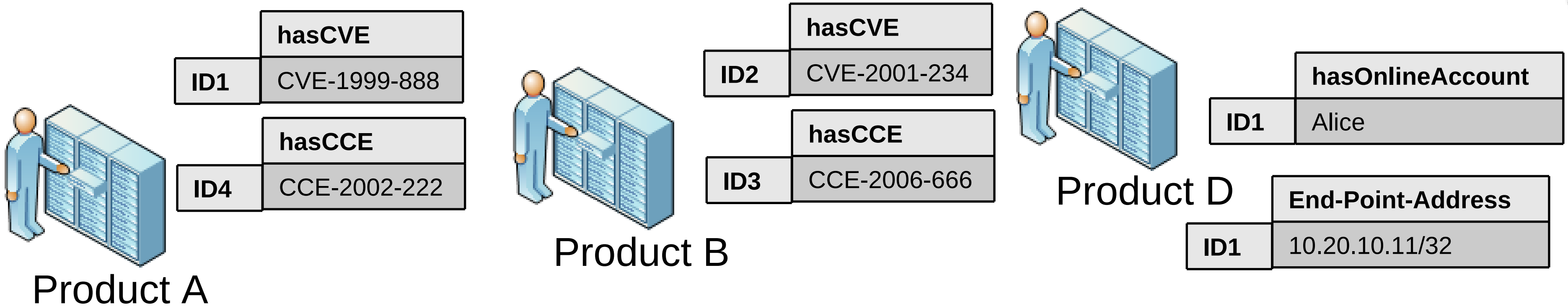
Interoperability: Row Based Multi-Vendor Architecture

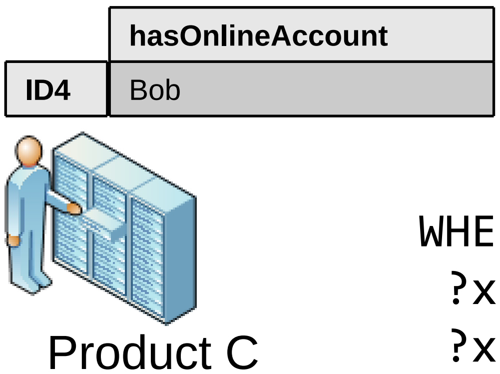


Interoperability: Column Based Multi-Vendor Architecture



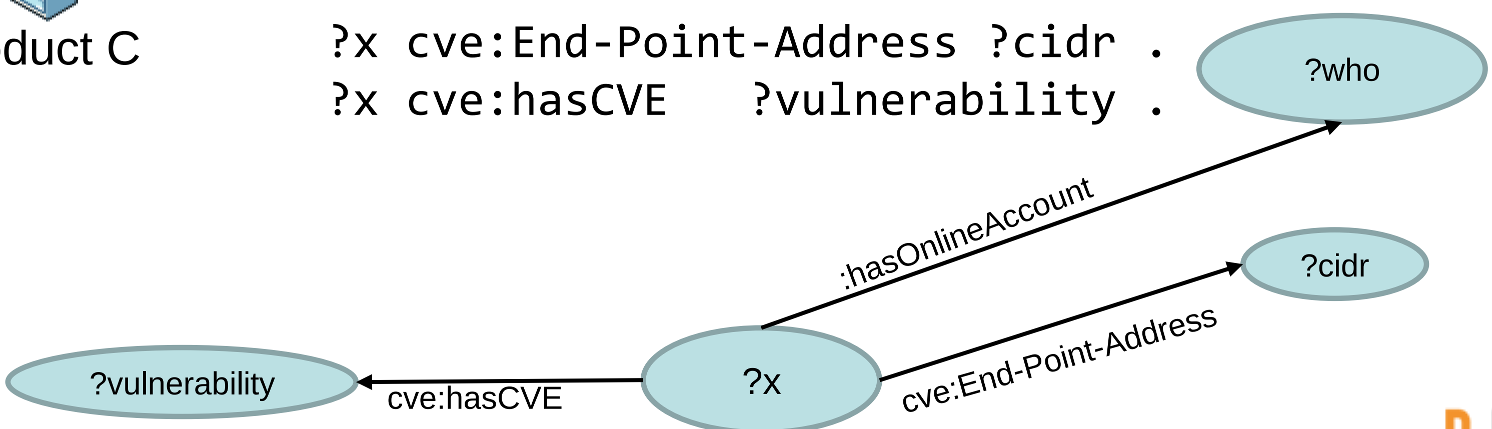
RDF Based Architecture and SPARQL Query





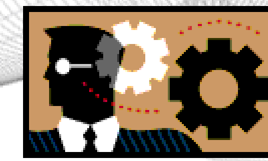
SPARQL allows you to describe a graph pattern

```
WHERE {
  ?x :hasOnlineAccount       ?who .
  ?x cve:End-Point-Address ?cidr .
  ?x cve:hasCVE    ?vulnerability .
```



nCircle

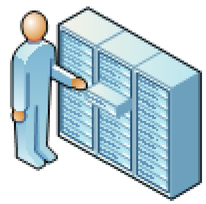
RDF Based Architecture and SPARQL Query



Product A

	hasCVE
ID1	CVE-1999-888

	hasCCE
ID4	CCE-2002-222



Product B

	hasCVE
ID2	CVE-2001-234

	hasCCE
ID3	CCE-2006-666

Product D

	hasOnlineAccount
ID1	Alice

	End-Point-Address
ID1	10.20.10.11/32

	hasOnlineAccount
ID4	Bob

Product C

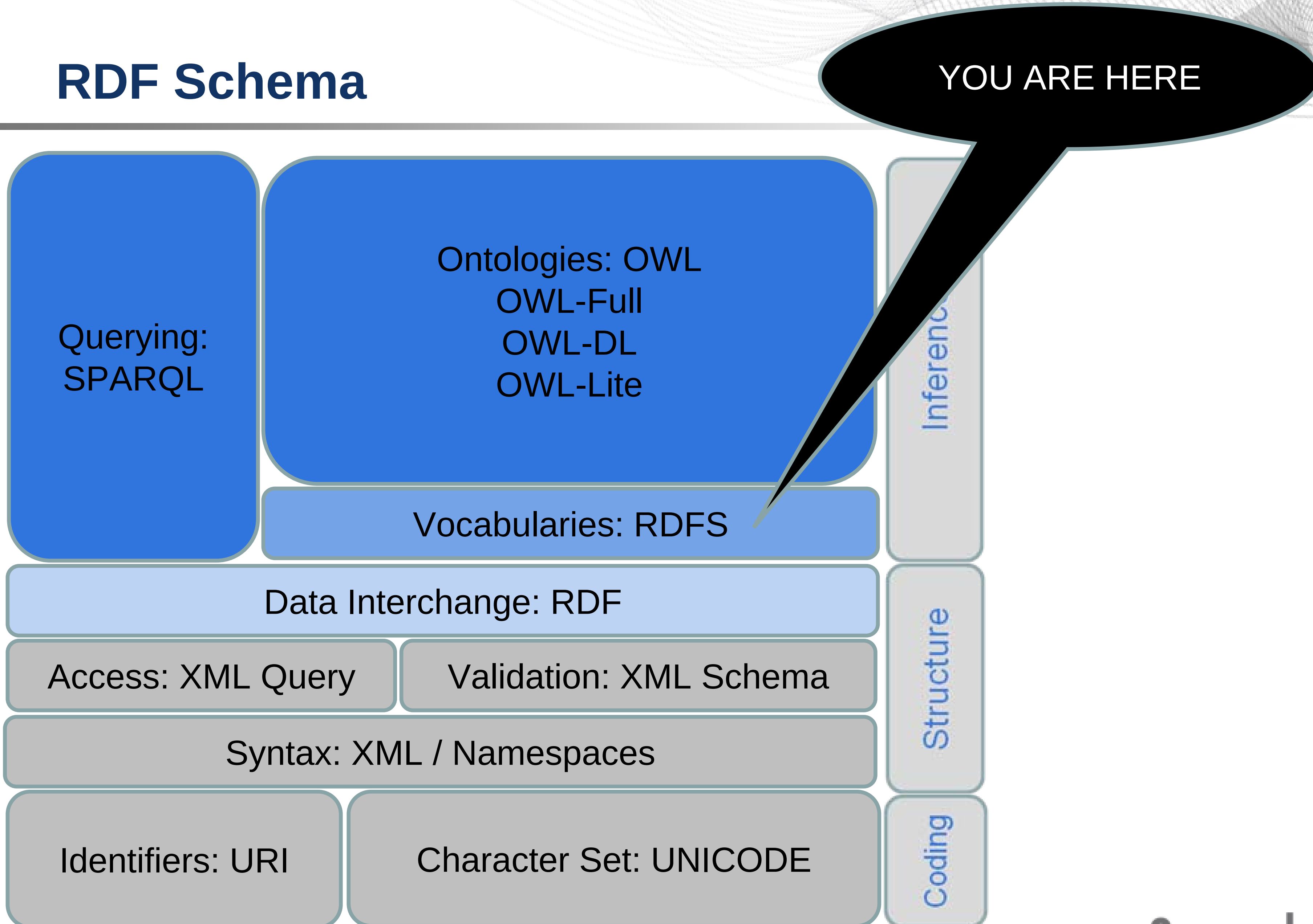
```
PREFIX cve: <http://nvd.nist.gov/cve/1.1/>
SELECT  ?who ?cidr ?vulnerability
FROM <ProductA>
FROM <ProductB>
FROM <ProductC>
FROM <ProductD>
WHERE {
 ?x :hasOnlineAccount      ?who .
 ?x cve:End-Point-Address ?cidr .
 ?x cve:hasCVE   ?vulnerability .
```

?who	?cidr	?vulnerability
Alice	10.20.10.11/32	CVE-1999-888

Quick Review

- RDF is a Labeled-Directed Graph
- An RDF statement is made up of a Subject-Predicate-Object sometimes called a “Triple”
- Federation and Merging is a build-in feature
- Both nodes and arcs are first-class
- Support for Reification
- SPARQL is an access language for Triple Stores
- Next Stop: The Power of Inference

RDF Schema



YOU ARE HERE
Querying: SPARQL
Ontologies: OWL
OWL-Full
OWL-DL
OWL-Lite
Vocabularies: RDFS
Data Interchange: RDF
Access: XML Query
Validation: XML Schema
Syntax: XML / Namespaces
Identifiers: URI
Character Set: UNICODE
Inferenc
Structure
Coding

RDF Schema (RDF-S)

- RDF Vocabulary Description Language 1.0: RDF Schema
  - Vocabulary defined with RDF statements (triples)
- RDF-S Vocabulary is small
  - Relation between classes (Class , subClassOf)
  - Relation between properties (Property, subPropertyOf)
  - Class membership of individuals via properties (domain, range)
- Provides some sense of “meaning” to the RDF data
  - Meaning = what we can explicitly **infer** from the data
  - Axioms that express exactly **what inference can be drawn**
  - Semantics expressed through **the mechanism of inference**
  - Lets explore in the next slides how this works

Type Propagation

AXIOM

IF

A rdfs:subClassOf B .

r rdf:type A .

THEN

r rdf:type B .

- rdfs:Class

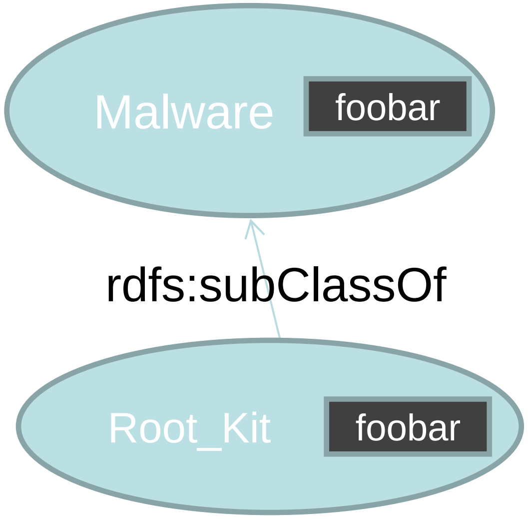
```
:Root_Kit rdf:type  rdfs:Class .
:Malware  rdf:type  rdfs:Class .
```

- rdfs:subClassOf

```
:Root_Kit rdfs:subClassOf  :Malware  .
:foobar   rdf:type         :Root_Kit .
```

we can then infer the triple

```
:foobar   rdf:type         :Malware  .
```



Relationship Propagation

- rdfs:Property

```
:hasBrother rdf:type rdfs:Property .
:hasSibling rdf:type rdfs:Property .
```

- rdfs:subPropertyOf

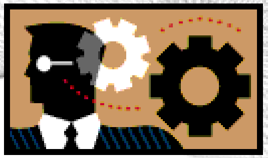
```
:hasBrother rdfs:subPropertyOf :hasSibling .
:alice          :hasBrother     :bob         .
```

we can infer the triple

```
:alice  :hasSibling  :bob .
```

AXIOM
IF
P rdfs:subPropertyOf R .
A P B .
THEN
A R B .

Property-Oriented versus Object-Oriented



- Semantic data is focused on the relationship between entities and thus Property-Oriented
- In Object-Oriented models, an entity is understood to be a member of a class because the class acts as a "template" for its birth
- In Property-Oriented models, an entity is understood to be a member of a class because of its relationships
- <DOMAIN> property_P <RANGE>
  - The domain is the collection of types that use the property
  - The range is the types of values this property describes
  - Example: domain:CPE :hasVulnerability range:CVE

Class Membership through Relationships

- Similar to domain and range in math

```
:property_P rdfs:domain D-class .
:property_P rdfs:range  R-class .
Domain applies to the Subject
Range applies to the Object
```

- Example:

```
:usesSharedLib rdfs:domain :Application .
:usesSharedLib rdfs:range  :SharedLib   .
```

  - Assertion

```
:Apache :usesSharedLib :OpenSSL .
```

  - Inference

```
:Apache  rdf:type :Application .
:OpenSSL rdf:type :SharedLib   .
```

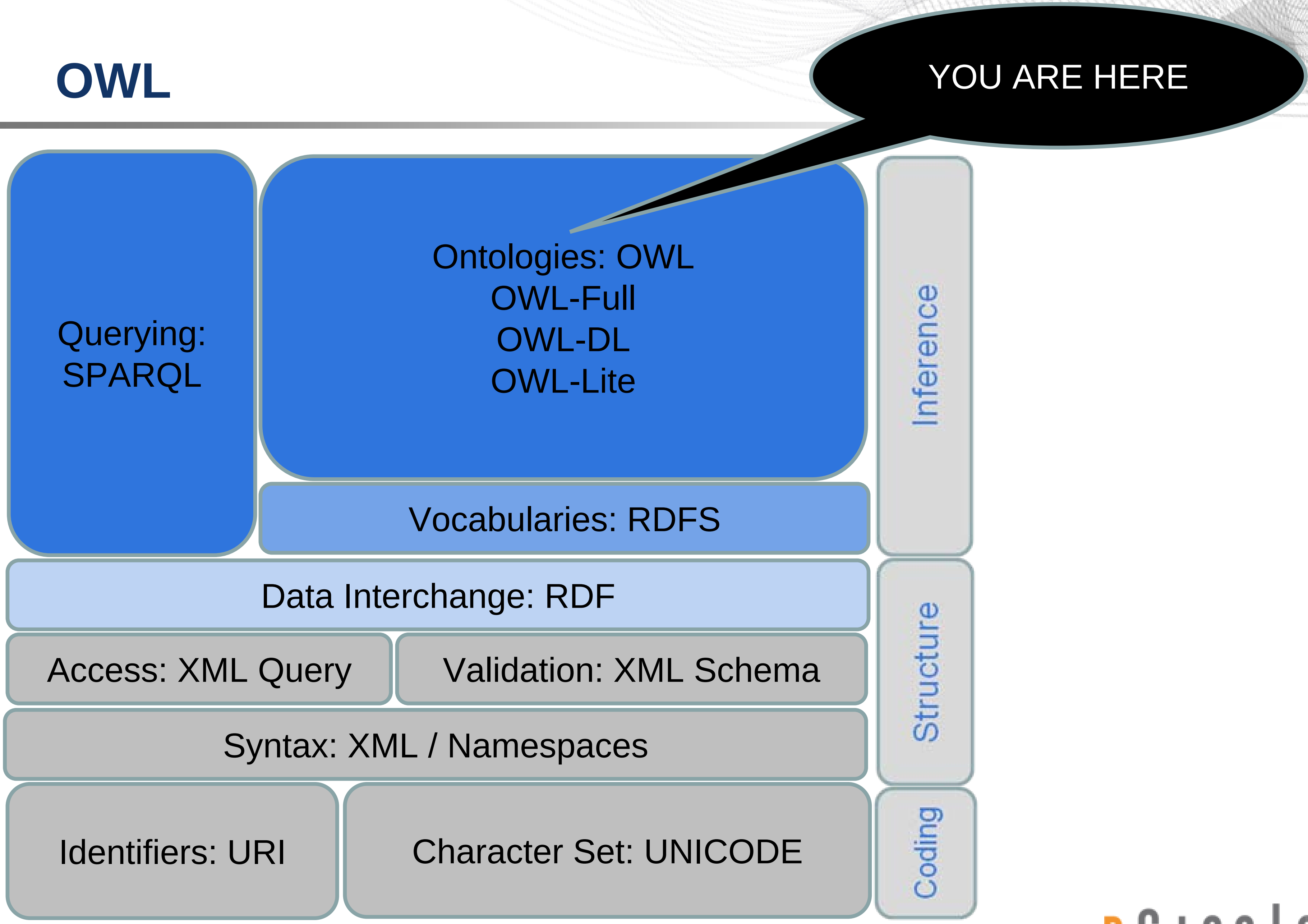
AXIOM (subject)
IF
P rdfs:domain D-class .
and
x P y .
THEN
x rdf:type D-class .

AXIOM (object)
IF
P rdfs:range R-class .
and
x P y .
THEN
y rdf:type R-class .

What are the limits to RDFS?

- RDFS may not have enough detail for your modeling
  - No localized range and domain constraints
    - Can't say that "the domain of hasParent is Child when applied to Human and Calf when applied to Elephants"
  - No existence/cardinality constraints
    - Can't say that "all *instances* of person have a mother that is also a person", or that persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties
    - Can't say that *isAncestorOf* is a transitive property
    - Can't say that *bundles* is the inverse of *isBundledBy*
    - Can't say that *isMarriedTo* or *isPeeredWith* is symmetrical

OWL


YOU ARE HERE
Querying: SPARQL
Ontologies: OWL
OWL-Full
OWL-DL
OWL-Lite
Vocabularies: RDFS
Data Interchange: RDF
Access: XML Query
Validation: XML Schema
Syntax: XML / Namespaces
Identifiers: URI
Character Set: UNICODE
Inference
Structure
Coding

OWL - Capability

- What are _SOME_ things you do with OWL?
  - Anything that can be done with RDFS
  - Addresses the shortcomings of RDFS (if you need it):
    - Classification – richer, more expressive than RDFS
      - Localized domain and range
    - Schema validation and constrains checking
      - Existence/Cardinality
    - Exploring network of relationships
      - Transitive/Inverse/Symmetrical
- Don't use OWL unless your model requires it.
  - Simplicity whenever possible

Computing Equivalence

- owl:functionalProperty
- owl:inverseFuntionalProperty
- Both functional and inverseFuntional

AXIOM	AXIOM
IF	IF
P rdf:type owl:FunctionalProperty .	P rdf:type owl:InverseFunctionalProperty .
x P a .	a P x .
x P b .	b P x .
THEN	THEN
a owl:sameAs b .	a owl:sameAs b .

A way to model when resources from multiple sources are the same individual

Note: ‘a’ and ‘b’ could be from two different Ontologies

Computing Equivalence

:alice :hasBioFather :bob_smith .
:alice :hasBioFather :robert_p_s .

:bob_smith owl:sameAs :robert_p_s

:foo :hasCVEid :CVE-2004-0112 .
:bar :hasCVEid :CVE-2004-0112 .

:foo :owl:sameAs :bar .

AXIOM
IF
P rdf:type owl:FunctionalProperty .
x P a .
x P b .
THEN
a owl:sameAs b .

AXIOM
IF
P rdf:type owl:InverseFunctionalProperty .
a P x .
b P x .
THEN
a owl:sameAs b .

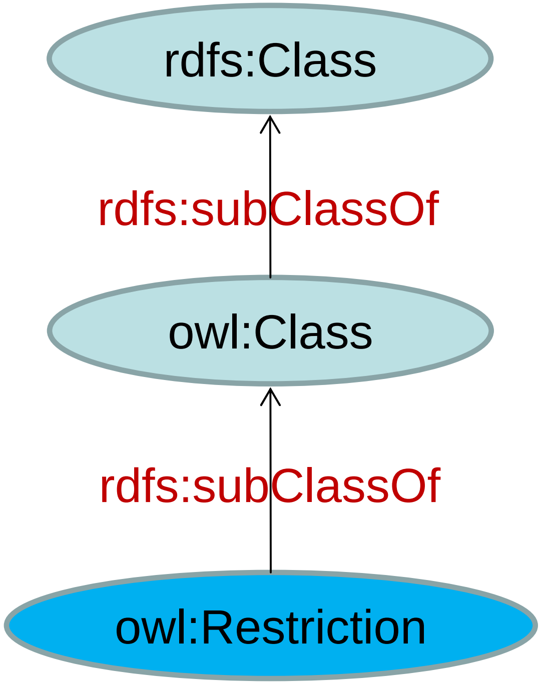
Functional and Inverse Functional like the Property :hasSocialSecurityNumber
One person to have exactly one SS#
one SS# to have exactly one person.

nCircle

OWL extends RDFS which extends RDF…

- RDF is used to define RDFS
- RDFS is used to define OWL and so on and so on

Subject	Predicate	Object
rdf:type	rdfs:domain	rdfs:Resource
rdf:type	rdfs:range	rdfs:Class
rdfs:domain	rdfs:domain	rdf:Property
rdfs:domain	rdfs:range	rdfs:Class
rdfs:range	rdfs:domain	rdf:Property
rdfs:range	rdfs:range	rdfs:Class
rdfs:subClassOf	rdfs:domain	rdfs:Class
rdfs:subClassOf	rdfs:range	rdfs:Class



OWL Restriction Example

- owl:restrictions allows you to describe a class in terms of other things we have already modeled
- Concept of a Child == a thing that has a parent who is human.

Human joe

:joe rdf:type :Human.

Child

```
:Child owl:equivalentClass
          [ a owl:Restriction;
               owl:onProperty hasParent
               owl:someValuesFrom  Human]
.
```

Assert

:jack :hasParent :joe .

Infer

:jack rdf:type :Child .

OWL Restriction – Localized Domain Example

– RDFS has no localised range and domain constraints
  - Can't say that "the domain of hasParent is Child when applied to Human and Calf when applied to Elephant"

Human joe

:joe rdf:type :human.

Elephant joe

:joe rdf:type :Elephant .

Child

```
:Child owl:equivalentClass
   [ a owl:Restriction;
       owl:onProperty hasParent
       owl:someValuesFrom  Human] .
```

Calf

```
:Elephants owl:equivalentClass
     [ a owl:Restriction;
         owl:onProperty hasParent
         owl:someValuesFrom  Elephant ] .
```

Assert

:jack :hasParent :joe .

Infer

:jack rdf:type :Child .

Assert

:jack :hasParent :joe .

Infer

:jack rdf:type :Calf .

nCircle

A partial account of the OWL vocabulary

OWL	Example	Concept
someValuesFrom	hasParent someValuesFrom Human	Child
allValuesFrom	eats allValuesFrom VegetarianFood	Vegetarian
hasValue	hasCountryOfOrigin hasValue USA	American
minCardinality	hasNetInterface min 2	Multihomed
cardinality	hasWheel exactly 1	Unicycle
maxCardinality	hasNetInterface max 0	notNetworked
intersectionOf	Doctor and Female	Female Doctor
unionOf	Man or Woman	Person
complementOf	not Client	Server
equivalentClass	WindowsXP equivalentClass WinXP	Equivalency
equivalentProperty	hasVuln equivalentProperty hasVulnerability	Equivalency

OWL dialects

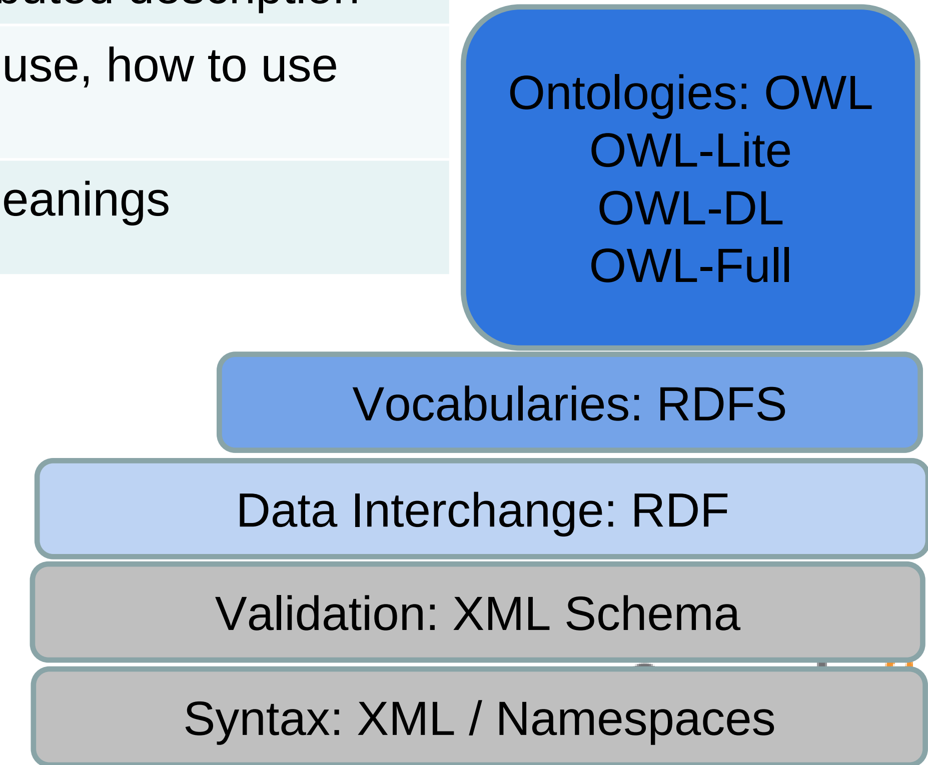
- OWL-Lite, OWL-DL, and OWL-FULL
  - OWL-Lite is a Subset of OWL-DL
  - OWL-DL's Objective is Decidability
  - DL stands for Description Logics which is a First Order Logic
  - OWL-FULL's Objective is Executability
- Use only what you need!

OWL: Managing Ontologies

OWL	Brief Description
DeprecatedClass	Specifies that the class is deprecated in a particular version (and should not be used)
DeprecatedProperty	Specifies that the property is deprecated in a particular version (and should not be used)
versionInfo	Annotation property for version info
priorVersion	Refer one ontology to another ontology that is a prior version
backwardCompatibleWith	Like *priorVersion* but further states the new ontology is backward compatible with the previous one
inCompatibleWith	Like *priorVersion* but further states the new ontology is incompatible with the previous one
Imports	Allows one ontology to refer explicitly to another

Summary - W3C Semantic Technology Stack

Solution	Issue
OWL	Define logical constraints for entities and relationships
RDFS	Provide inference about types and inclusion
RDF	Identify items for distributed description
XML Schema	Describe what tags to use, how to use them (syntax)
XML Namespaces	Same word has two meanings



Thank You

tk@ncircle.com

Thanks to these people for my education on this domain:
Dean Allemang, Jim Hendler, Holger Knublauch, Jans Aasman, Irene Polikoff, Ralph Hodgson, Scott Henninger, Jeremy Carroll, Peter F. Patel-Scheider, Dan Brickley and everyone else…