Semantic Web

OWL e SPARQL

Requisiti per un Web Ontology Language

Caratteristiche desiderabili individuabili per un Web Ontology Language:

- Estende gli standard Web esistenti
  - Come XML, RDF(S)
- Semplice da capire e da utilizzare
  - Dovrebbe essere basato su formalismi KR noti
- Dotato di una specifica formale
- Di potere espressivo “adeguato”
- In grado di fornire un supporto al ragionamento automatizzato

Il Linguaggio OWL

- OWL è basato sul formalismo di rappresentazione della conoscenza noto come Logiche Descrittive (DL)

- OWL (DL) beneficia di numerosi anni di ricerca sulle DL:
  - Semantica ben definita
  - Proprietà formali ben conosciute (complessità, decidibilità)
  - Algoritmi di ragionamento noti
  - Sistemi implementati (reasoner ottimizzati)

- Nasce come standardizzazione del linguaggio DAML+OIL, integrazione dei risultati di ricerche americane (DAML) e europee (OIL)

- Diverse specie in OWL 1 (2004): OWL Full, OWL DL, OWL Lite e altre in OWL 2 (2009): OWL EL, OWL RL, OWL QL

Perché OWL (=gufo)?

- OWL = Web Ontology Language
- Owl's superior intelligence is known throughout the Hundred Acre Wood, as are his talents for Writing, Spelling, other Educated and Special tasks.
- *"My spelling is Wobbly. It's good spelling, but it Wobbles, and* ***the letters get in the wrong places.****"*

Ontologia: Origini e Storia

- Una disciplina filosofica
- Un ramo della filosofia che si occupa della natura e dell'organizzazione della realtà
- Scienza dell'Essere
  (Aristotele, Metafisica, IV, 1)
- Tenta di rispondere alle domande:
  - *Cosa caratterizza l'essere?*
  - *Cos'è in definitiva l'essere?*

Definizione sintetica:

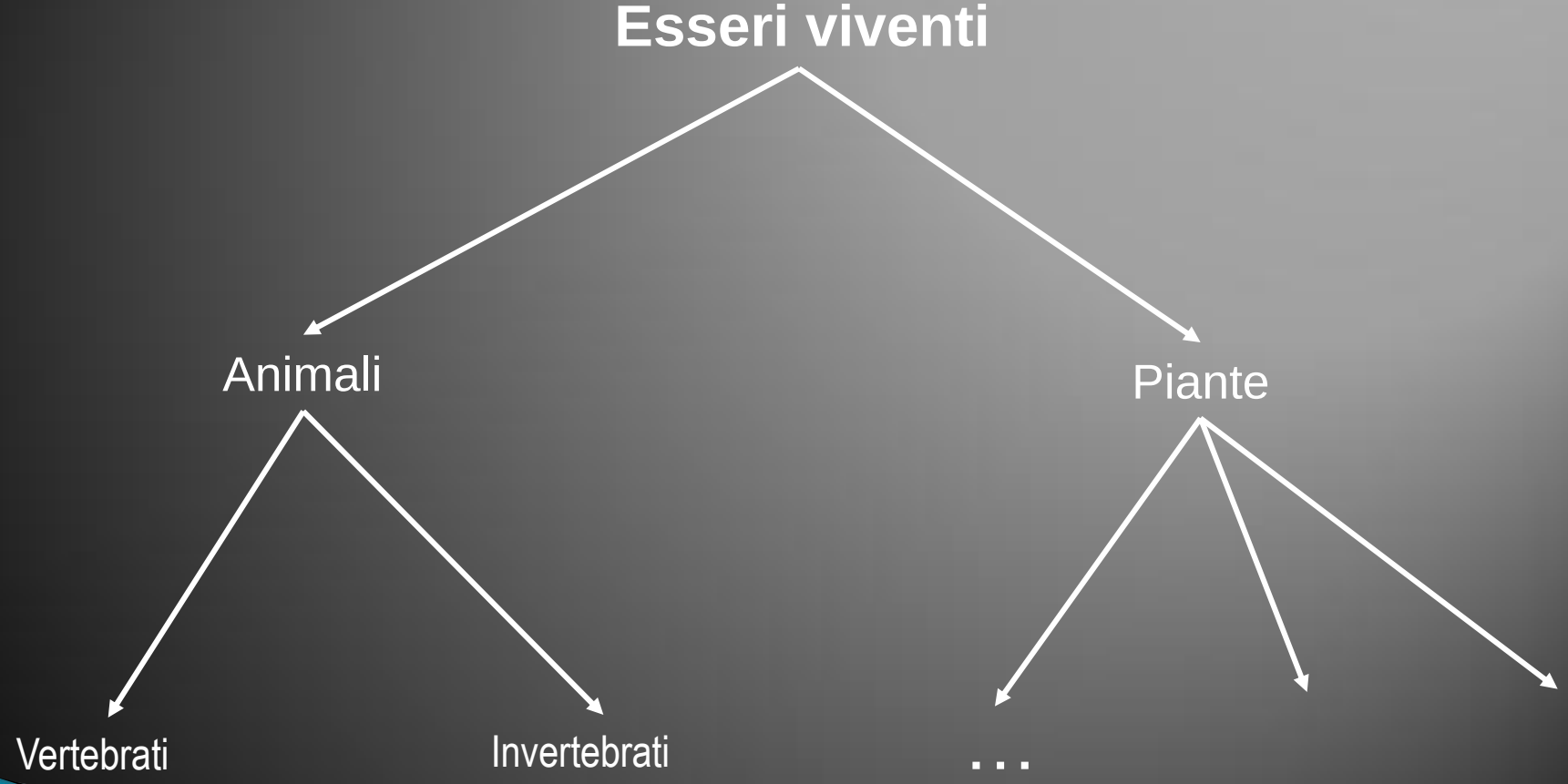
Specifica formale, esplicita di una concettualizzazione condivisa

- **formale**: **Comprensibile alle macchine**
- **esplicita**: **Rende esplicite le assunzioni di dominio**
- **concettualizzazione**: **Modello concettuale di un qualche aspetto della realtà**
- **condivisa**: **Diverse persone concordano sul fatto che tale descrizione della realtà è adeguata**

8

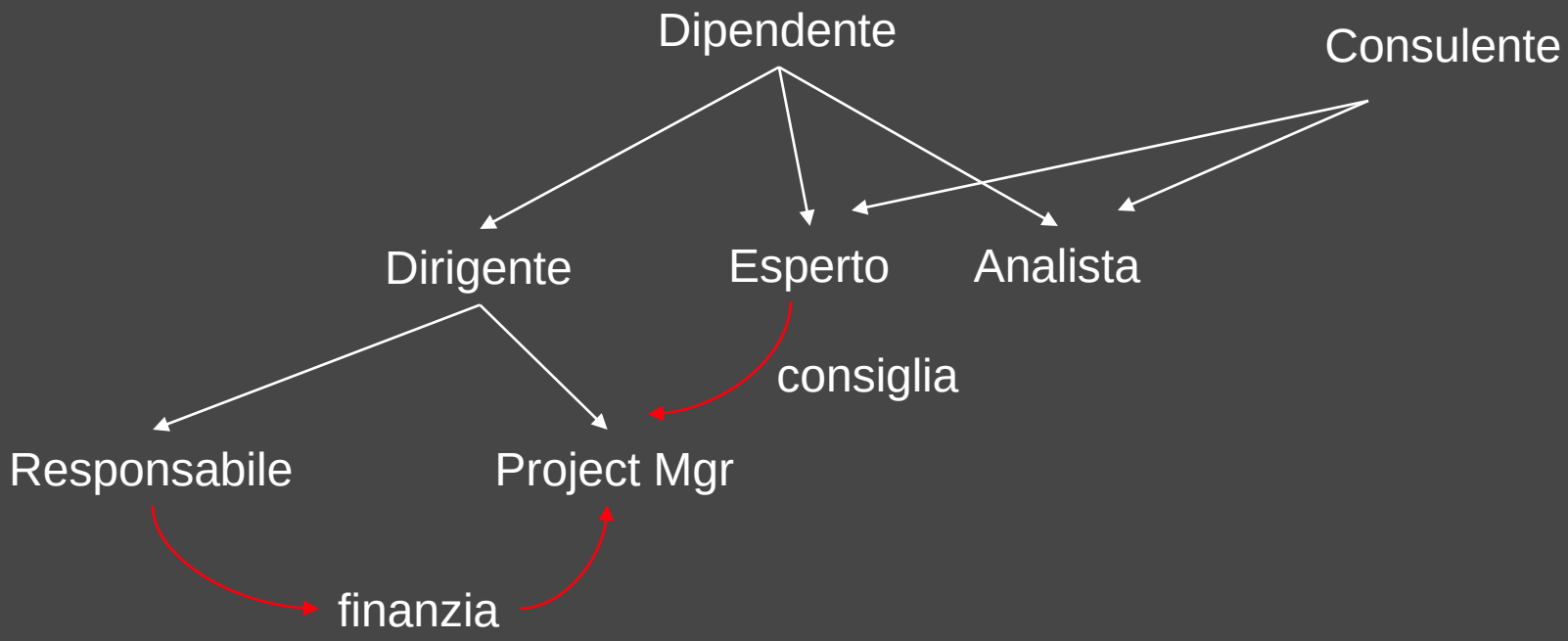
Ontologia come Tassonomia

Una Tassonomia è un sistema di classificazione in cui ciascun nodo ha soltanto un genitore – ontologia semplice (gerarchia)



Ontologia per persone e loro ruoli

Tipicamente, vogliamo un'Ontologia più ricca,
con più relazioni (non solo gerarchiche) fra concetti:



Struttura di un'Ontologia

Le Ontologie hanno tipicamente due componenti:

- Nomi per i concetti e le relazioni importanti nel dominio, es.
  - Elefante è un concetto i cui membri sono un tipo di animale
  - Erbivoro è un concetto i cui membri sono esattamente quegli animali che mangiano solamente piante o parti di piante

- Conoscenza di fondo/vincoli sul dominio, es:
  - Gli Elefanti Adulti pesano almeno 2000 kg
  - Nessun individuo può essere contemporaneamente un Erbivoro e un Carnivoro

Perché sviluppare un ontologia?

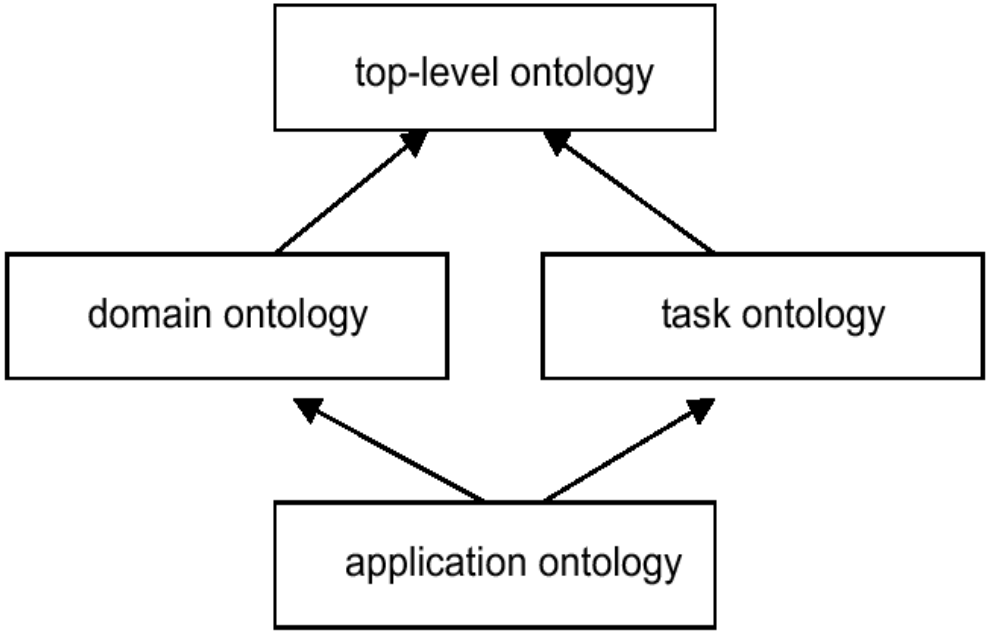
- *Per definire le risorse web più precisamente e renderle più adatte ad un'elaborazione automatica*
- Per rendere esplicite le assunzioni di dominio
  - Più facile cambiare le assunzioni di dominio
  - Più facile comprendere e aggiornare dati legacy
- Per separare la conoscenza di dominio dalla conoscenza operativa
  - Ri-uso delle due conoscenze separatamente
- Per creare una comunità di riferimento per le applicazioni (che sviluppa e manutiene l'ontologia)
- Per far condividere una comprensione consistente di ciò che l'informazione significa

Tipi di Ontologie

[Guarino, 98]

Descrve **concetti molto generali** come spazio, tempo, evento, che sono indipendenti da un particolare problema o dominio. Sembra ragionevole avere ontologie top-level unificate per grandi comunità di utenti.

Descrive il vocabolario relativo ad un **dominio generico** specializzando i concetti introdotti nell'ontologia top-level.



Descrive il vocabolario relativo ad un **generico compito o attività** specializzando l'ontologia top-level.

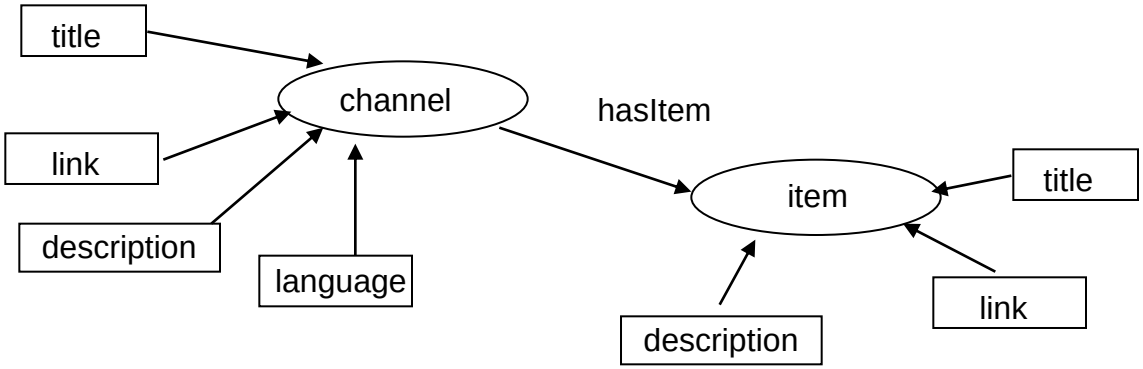
Sono le ontologie più specifiche. I concetti nelle ontogie applicative spesso corrispondono ai **ruoli svolti dalle entità di dominio nell'eseguire una determinata attività**.

Ontologie – Esempi

- **Ontologie general-purpose:**
  - WordNet / EuroWordNet, http://www.cogsci.princeton.edu/~wn
  - The Upper Cyc Ontology, http://www.cyc.com/cyc-2-1/index.html
  - IEEE Standard Upper Ontology, http://suo.ieee.org/
- **Ontologie specifiche per domini o applicazioni:**
  - RDF Site Summary RSS, http://groups.yahoo.com/group/rss-dev/files/schema.rdf
  - RETSINA Calendering Agent, http://ilrt.org/discovery/2001/06/schemas/ical-full/hybrid.rdf
  - AIFB Web Page Ontology, http://ontobroker.semanticweb.org/ontos/aifb.html
  - Dublin Core, http://dublincore.org/
  - UMLS, http://www.nlm.nih.gov/research/umls/
  - Open Biological Ontologies: http://obo.sourceforge.net/
- **Ontologie in un senso più ampio**
  - Agrovoc, http://www.fao.org/agrovoc/
  - Art and Architecture, http://www.getty.edu/research/tools/vocabulary/aat/
  - UNSPSC, http://eccma.org/unspsc/
- **DAML.org biblioteca con i più svariati tipi di ontologie!**

RSS (RDF Site Summary)

- RDF Site Summary (RSS) is a lightweight multipurpose extensible metadata description and syndication format.
- The underlying RDF(S) ontology is extremely simple, mainly consisting of:



UMLS (Unified Medical Language System) (I)

- provided by the US National Library of Medicine (NLM), a database of medical terminology
- terms from several medical databases (MEDLINE, SNOMED International, Read Codes, etc.) are unified so that different terms are identified as the same medical concept
- access at http://umlsks.nlm.nih.gov/

UMLS (Unified Medical Language System) (II)

- **UMLS Knowledge Sources:**
  - Metathesaurus provides the concordance of medical concepts:
    - 730,000 concepts
    - 1.5 million concept names in different source vocabularies

Dublin Core

- The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models
- Ontology includes elements like
  - TITLE
  - CREATOR
  - SUBJECT
  - DESCRIPTION
  - PUBLISHER
  - DATE…

Simple set of elements that people can agree on!

see http://dublincore.org/

Open Biological Ontologies

- Various ontologies in the biological domain
- obo.sourceforge.net
- e.g. Gene Ontology (www.geneontology.org)

"Biologists currently waste a lot of time and effort in searching for all of the available information about each small area of research. This is hampered further by the wide variations in terminology that may be common usage at any given time, and that inhibit effective searching by computers as well as people. For example, if you were searching for new targets for antibiotics, you might want to find all the gene products that are involved in bacterial protein synthesis, and that have significantly different sequences or structures from those in humans. But if one database describes these molecules as being involved in 'translation', whereas another uses the phrase 'protein synthesis', it will be difficult for you — and even harder for a computer — to find functionally equivalent terms. The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases."

- Hundreds of classes

Ontologie e Logica

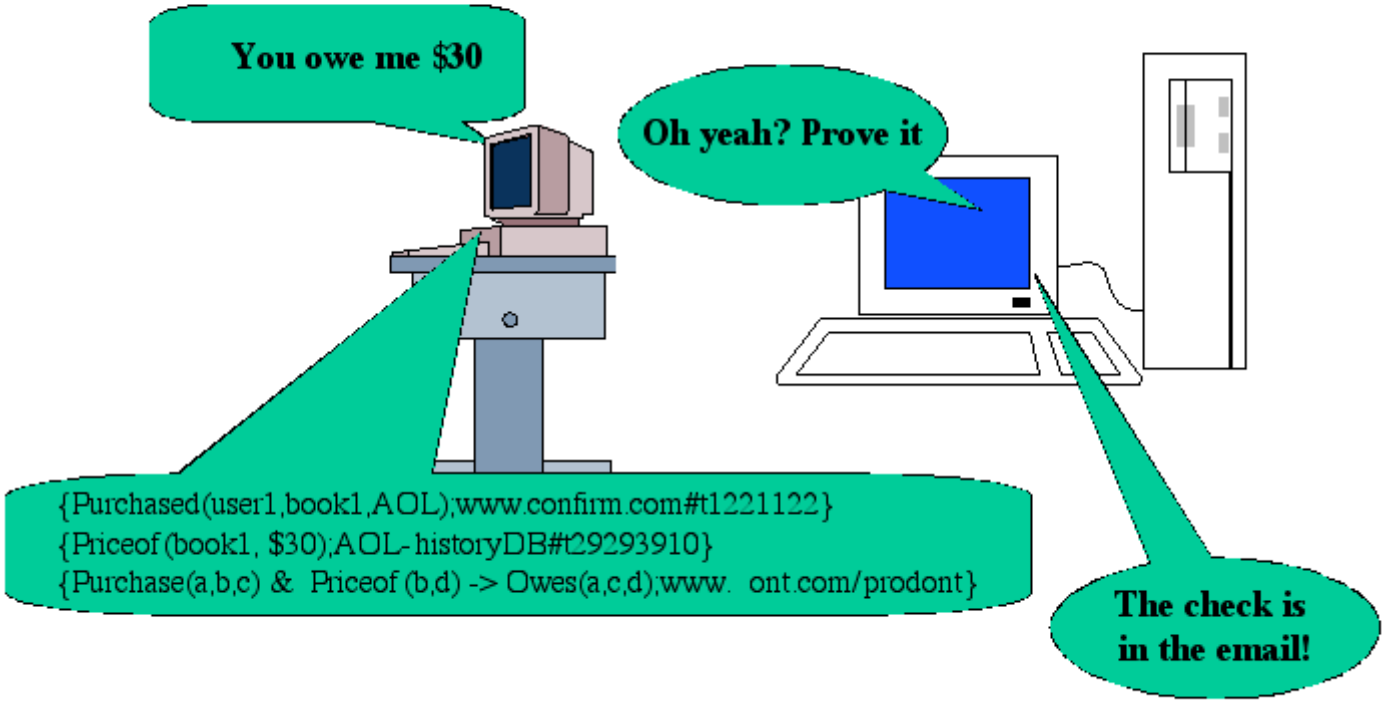
- Ragionamento automatico su ontologie
- Possibilità di inferenze

X is author of Y
⇨ Y is written by X

X is supplier to Y; Y is supplier to Z
⇨ X and Z are part of the same supply chain

Cars are a kind of vehicle;
Vehicles have 2 or more wheels
⇨ Cars have 2 or more wheels

Supporto ai livelli di Proof e Trust



Web Ontology Language (OWL)

- RDFS cattura le relazioni semantiche di base (corrispondenti ad un object-oriented type system)
- Sfumature aggiuntive di significato sono necessarie per un'effettiva rappresentazione della conoscenza (KR)
- OWL introduce quindi un proprio namespace, con costrutti addizionali standard per consentire di rappresentare tali sfumature di significato
  - Costruito sopra RDF(S), serializzabile in XML
  - Assegna una specifica semantica ai nuovi termini

Definizione di Ontologie in OWL

- Le definizioni OWL arricchiscono o raffinano le specifiche di concetti RDFS o mettono in relazione tra loro oggetti

```
<owl:Thing rdf:ID="Mary"/>
<owl:Thing rdf:ID="John">
    <hasParent rdf:resource="#Mary"/>
</owl:Thing>

<owl:Class rdf:ID="Mammal">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <owl:disjointWith rdf:resource="#Reptile"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="#Animal"/>
</owl:ObjectProperty>
```

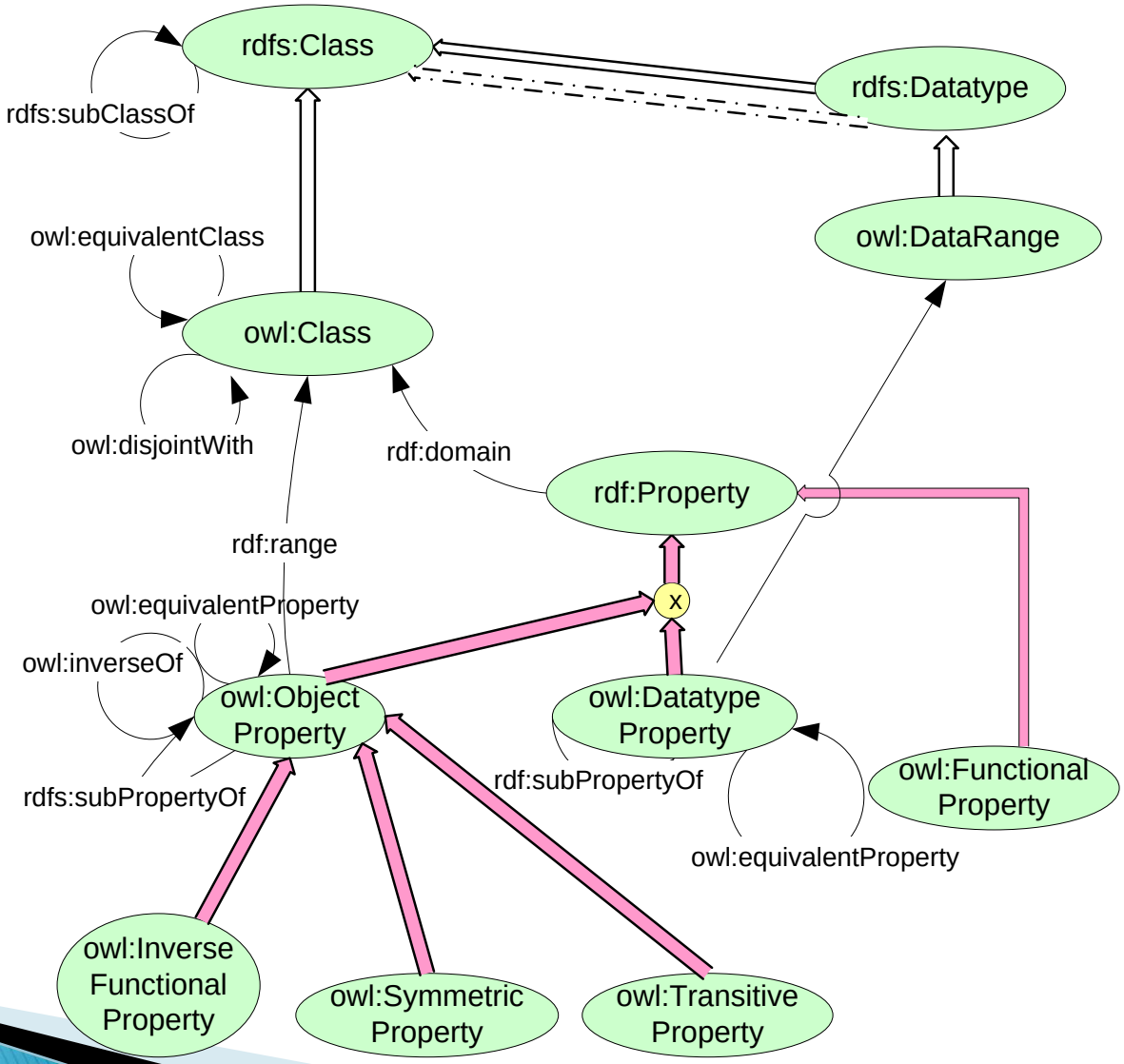
Semplice Inferenza

- Data la definizione della proprietà hasParent e il frammento:

```
<owl:Thing rdf:ID="Fido">
  <hasParent rdf:resource="#Rover"/>
</owl:Thing>
```

Possiamo inferire che Fido è un animale

Entità e relazioni in OWL



Definizione di Classi in OWL

- OWL rispetto a RDFS opera una distinzione concettuale più forte fra classi e individui
  - Per fare la differenza, c'è un termine separato per le classi owl:Class
  - Gli individui sono separati in una classe speciale chiamata owl:Thing
- OWL separa anche le proprietà i cui valori sono oggetti oppure dati (una datatype property ha come range literal tipizzati)

Costruzione di Classi OWL

- Esplicita (come negli esempi precedenti) oppure:
- Anonima, usando:
  - Restrizioni (prossime slide)
  - Operatori insiemistici: intersectionOf, unionOf, complementOf

  Esempio:

```
<owl:Class rdf:ID='SugaryBread'>
  <owl:intersectionOf rdf:parseType='Collection'>
    <owl:Class rdf:about='#Bread'/>
    <owl:Class rdf:about='#SweetFood'/>
  </owl:intersectionOf>
</owl:Class>
```

Altri esempi di definizione di classi

- Classe definita come unione di altre classi

```
<owl:Class rdf:ID="Literature">
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Novel"/>
       <owl:Class rdf:about="#Short_Story"/>
       <owl:Class rdf:about="#Poetry"/>
       …
   </owl:unionOf>
</owl:Class>
```

- Il possibile contenuto può essere elencato esplicitamente (classe “enumerata”):

```
<owl:Class rdf:ID="Currency">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:ID="£"/>
    <owl:Thing rdf:ID="€"/>
    <owl:Thing rdf:ID="$"/>
    …
  </owl:oneOf>
</owl:Class>
```

Restrizioni (1)

- Una caratteristica tipica delle Logiche Descrittive
- “Simile” alla divisione in aritmetica: definisce classi nei termini di una restrizione che devono soddisfare rispetto ad una data proprietà
- Anonima: tipicamente inclusa in una definizione di classe per consentire di fare ad essa riferimento
- Le primitive base sono:
  - someValuesFrom *una classe specificata*
  - allValuesFrom *una classe specificata*
  - hasValue *uguale ad un individuo specifico o ad un data type*
  - minCardinality *un numero intero positivo*
  - maxCardinality *un numero intero positivo*
  - Cardinality (se maxCardinality = minCardinality)

Restrizioni (2)

```
<owl:Class rdf:ID="Wine">
    <rdfs:subClassOf
           rdf:resource="&food;PotableLiquid" />
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasMaker" />
            <owl:allValuesFrom rdf:resource="#Winery" />
        </owl:Restriction>
    </rdfs:subClassOf>
    ...
</owl:Class>
```

- Il produttore di un vino (Wine), se definito, deve essere una cantina (Winery)
  - La restrizione allValuesFrom è sulla proprietà hasMaker della classe Wine qui definita
  - (produttori di altri prodotti, es. di formaggio – Cheese, non sono vincolati da questa restrizione locale)

Restrizioni (3)

Altri esempi di restrizioni:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasFather"/>
  <owl:maxCardinality
     rdf:datatype="xsd:nonNegativeInteger">
    1
  </owl:maxCardinality>
</owl:Restriction>

<owl:Restriction>
    <owl:onProperty rdf:resource='#bakes'/>
    <owl:someValuesFrom rdf:resource='#Bread'/>
</owl:Restriction>
```

Assiomi (1)

- Asserzioni che vengono date per vere
- Possono rivelarsi paricolarmente potenti in combinazione con altri assiomi (che possono anche provenire da documenti differenti)
- Alcune primitive:
  - rdfs:subClassOf
  - owl:equivalentClass

Assiomi (2)

```
<owl:AllDifferent>
<!- in pratica diseguaglianza a coppie ->
  <owl:distinctMembers rdf:parseType='Collection'>
   <ex:Country rdf:ID='Russia'/>
   <ex:Country rdf:ID='India'/>
   <ex:Country rdf:ID='USA'/>
  <owl:distinctMembers/>
</owl:AllDifferent>

<ex:Country rdf:ID='Iran'/>
<ex:Country rdf:ID='Persia'>
  <owl:sameIndividualAs rdf:resource='#Iran'/>
</ex:Country>
```

Restrizioni *vs* Assiomi

- Gli assiomi sono asserzioni globali che possono essere usati come base per derivare inferenze
- Le restrizioni sono costruttori
  - Quando stabiliamo che hasFather ha una maxCardinality di 1:
    - Definiamo la classe di animali che hanno zero o un padre: tale classe può avere istanze o meno
    - Non stabiliamo che tutti gli animali hanno zero o un padre
- Spesso per ottenere il risultato desiderato dobbiamo combinare restrizioni con assiomi (come quelli basati su equivalentClass)

Equivalenza di Termini

- Per le classi:
  - owl:equivalentClass: due classi hanno gli stessi individui
  - owl:disjointWith: non ci sono individui in comune
- Per le proprietà:
  - owl:equivalentProperty
    - Nell'esempio iniziale: a:author – f:auteur
  - owl:propertyDisjointWith
- Per gli individui:
  - owl:sameAs: due URI fanno riferimento allo stesso concetto ("individuale")
  - owl:differentFrom: negazione di owl:sameAs

Ma OWL è complesso!

- La combinazione dei costruttori di classe con le varie restrizione è estremamente potente
- Ciò che avremmo voluto fare è in definitiva:
  - Estendere le possibilità di RDF(S) con nuovi costrutti
  - Definire la loro semantica, ovvero cosa "significano" in termini di relazioni logiche
  - *Inferire nuove relazioni sulla base di quelle specificate nella definizione dell'ontologia*
- Tuttavia… una procedura globale di inferenza è "difficile"
  - Es. Non implementabile con un "rule engine"

Specie o Profili OWL

- Per questo motivo di OWL sono state definite diverse “specie” o profili:
  - I profili impongono limitazioni su quali costrutti possono essere usati e in quali circostanze
  - Se ci si attiene a tali limitazioni, allora possono essere usati dei ragionatori più semplici per l’inferenza
- Riflettono il compromesso fra espressività e implementabilità

Inferenza

- OWL descrive i contenuti, non la sintassi
- Affermazioni riguardanti la stessa URI provenienti da diversi documenti vengono congiunte
- OWL può apparire poco intuitivo ai non esperti
  - Dichiarare che nessuno può avere più di una madre
  - Dichiarare che Mary è la madre di John
  - Dichiarare che Jane è la madre di John
- Un DBMS segnalerebbe una violazione di integrità
- Un reasoner OWL conlcluderebbe: Mary = Jane

Confronto fra i Profili

- *OWL Full:* praticamente RDF(S)+tutta la sintassi OWL; estremamente generale; potenzialmente intrattabile (ragionamento indecidibile); incluso solo per esigenze di espressività estremamente particolari
- *OWL DL:* il dialetto base (frammento FOL di OWL Full), include le primitive delle DL; non necessariamente trattabile (ma in pratica spesso lo è)
- *OWL Lite:* aggiunge limitazioni d'uso ai costruttori di OWL DL per renderlo trattabile

Nuovi Profili in OWL 2

- *OWL EL:* limitazioni (non ammessi vincoli di cardinalità, vincoli sulle proprietà, disgiunzione classi…) finalizzate ad avere un ragionamento polinomiale
- *OWL RL:* finalizzato al ragionamento polinomiale tramite "rule engines"
- *OWL QL:* essenzialmente traducibile in query SQL eseguibili su un DB di istanze

Conclusioni su OWL

- OWL si aggiunge a RDF(S) per fornire un vocabolario ricco come necessario alla rappresentazione della conoscenza
  - E' sintesi di una gran quantità di eccellente lavoro sulla rappresentazione discreta, tassonomica della conoscenza
  - Si adatta bene alla descrizione di risorse informative – una base per descrivere vocabolari di metadati
  - Critico per descrivere servizi in modo non ambiguo, in modo che possano essere scelti e mandati opportunamente in esecuzione

Interrogazioni su grafi RDF

- Anche ai fini dell'inferenza è spesso necessario eseguire interrogazioni sui grafi RDF
  - Qualcosa tipo: "dammi la coppia (a,b) di risorse, per cui c'è un x tale che valgono le relazioni (x parent a) e (b brother x)" (ovvero trova gli zii)
    - Regole di questo tipo in alcuni casi possono diventare estremamente complesse
- A tal fine è stato definito un linguaggio di interrogazione standard:
  SPARQL (Query Language for RDF)

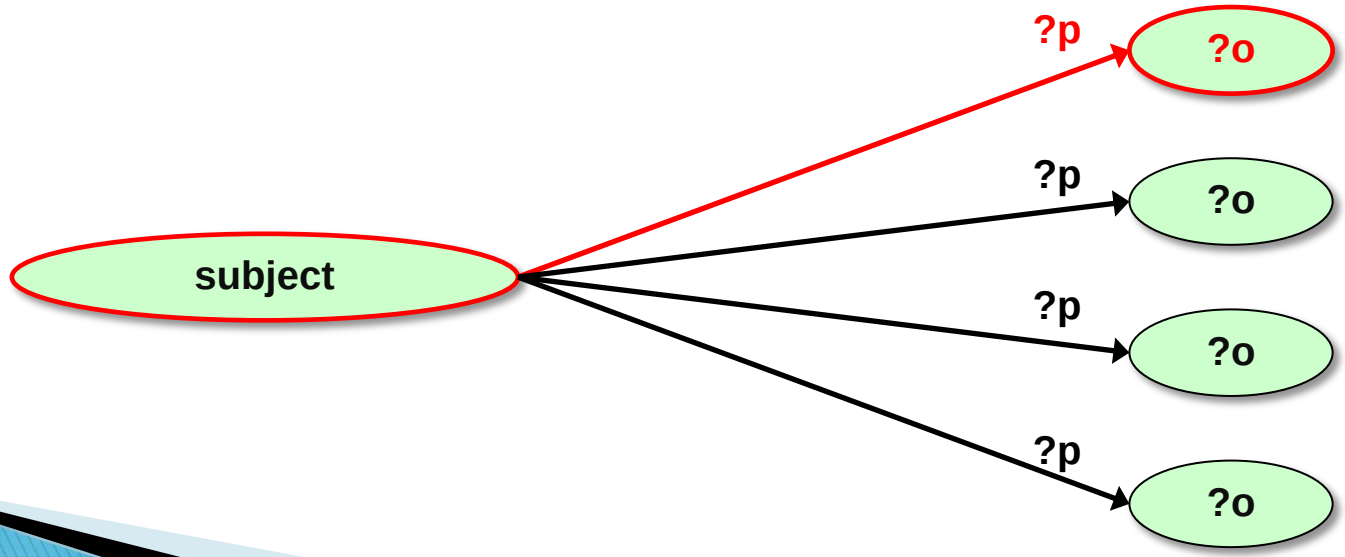
Uso di graph patterns

- L'idea fondamentale sottostante alla definizione di SPARQL è l'uso di "graph patterns"
  - Il pattern contiene simboli (?x, ?y…) non legati
  - Effetuando il collegamento dei simboli, si selezionano sottografi del grafo RDF
  - Se una tale selezione esiste, allora la query restituisce le risorse che si sono legate ai simboli
- Utilizza una sintassi SELECT-WHERE simile a quella di SQL

Una semplice query SPARQL

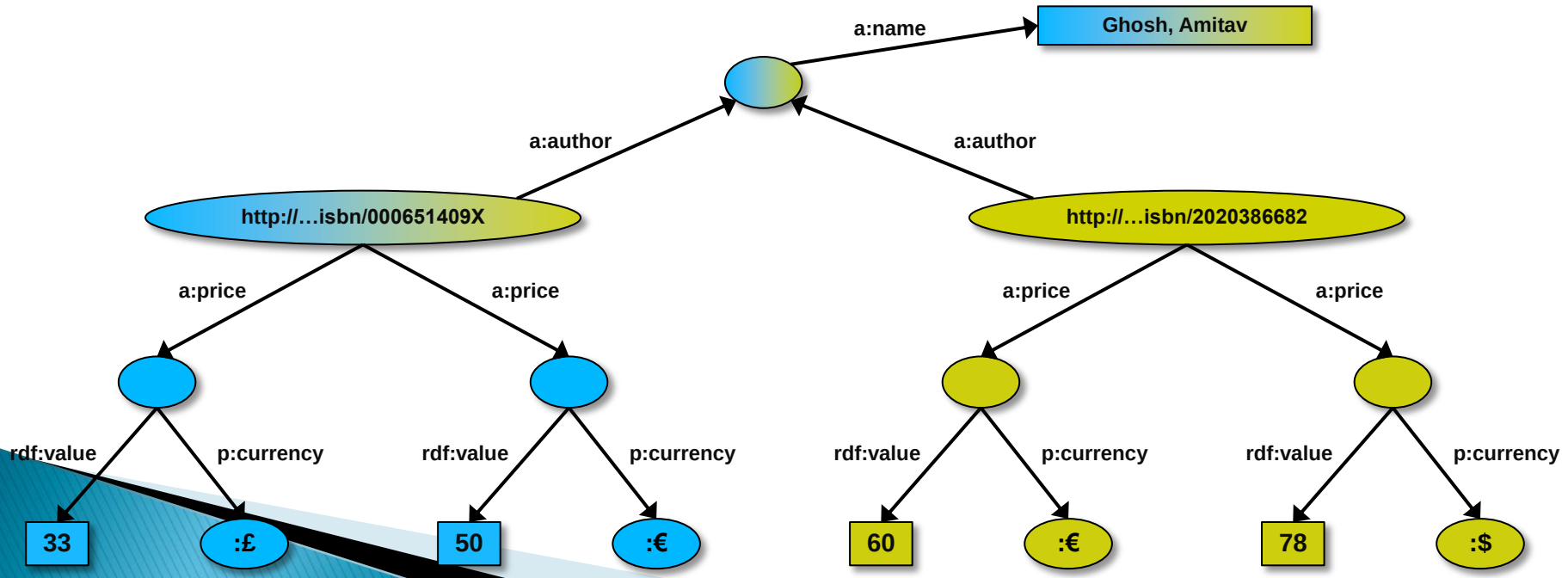
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

- Le triple nella clausola WHERE definiscono il graph pattern, con ?p e ?o simboli non legati
- La query restituisce tutte le coppie p,o



Un semplice esempio SPARQL

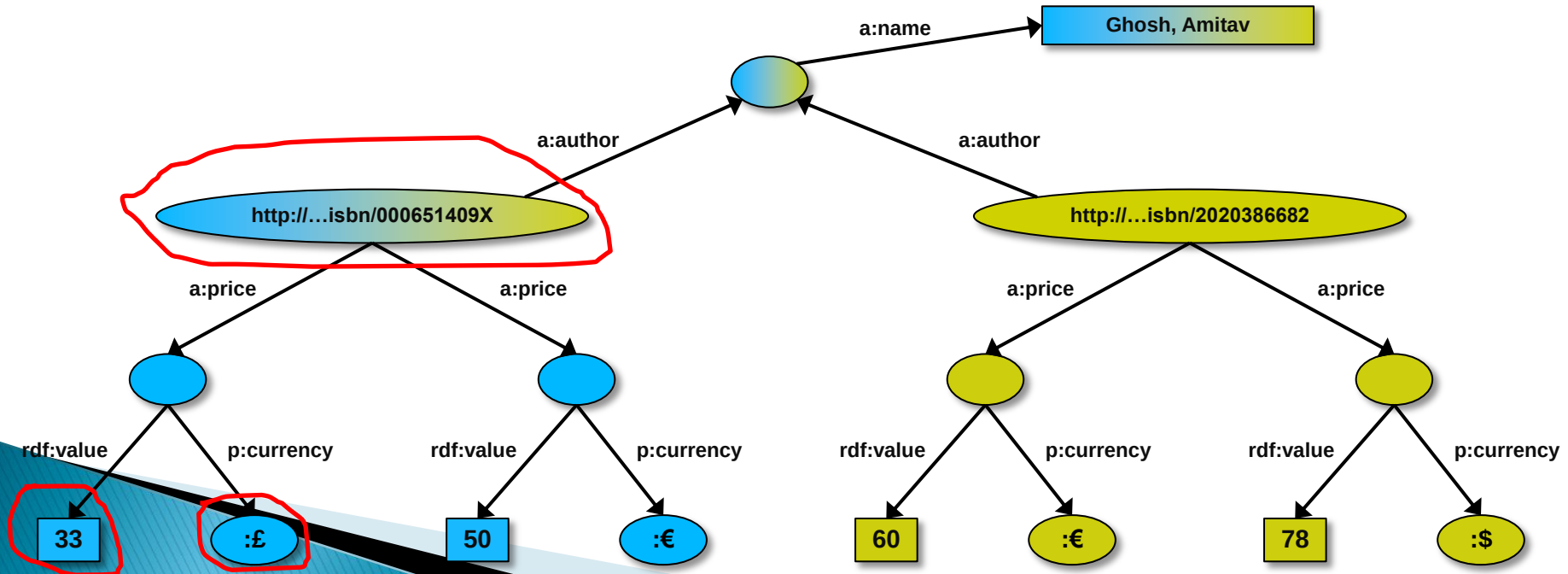
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Un semplice esempio SPARQL

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

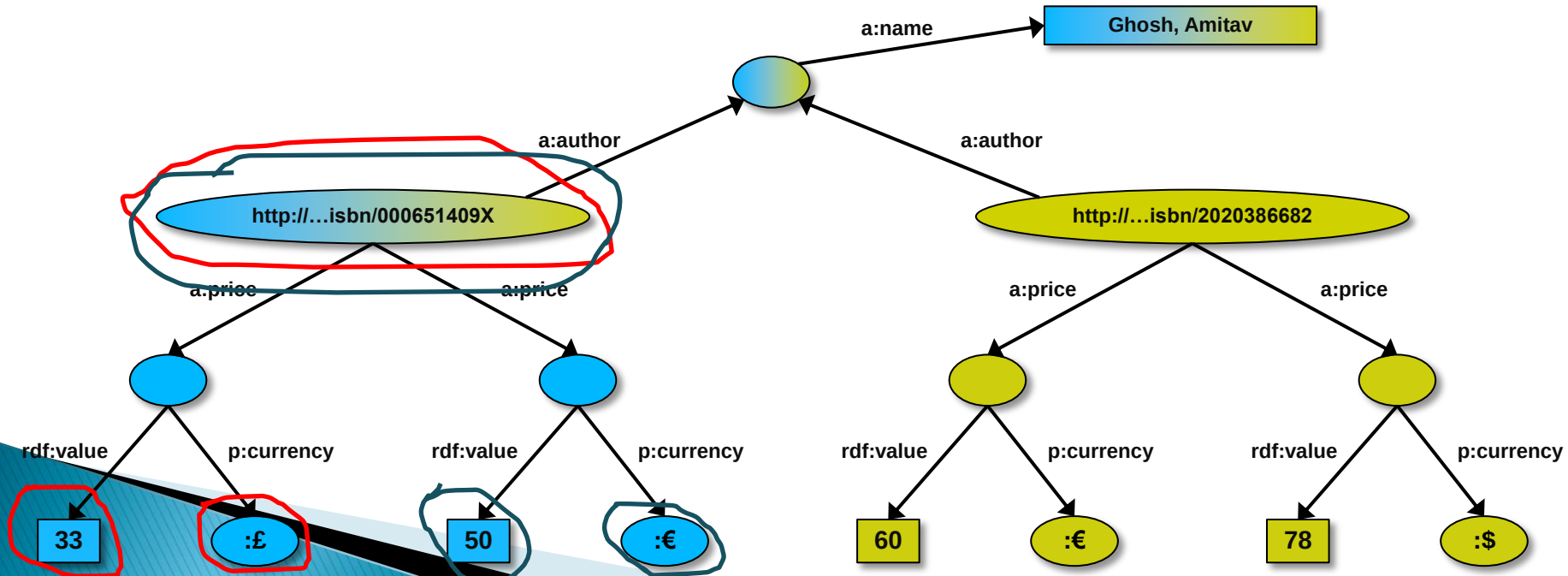
Ritorna: [<…409X>,33,:£]



Un semplice esempio SPARQL

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

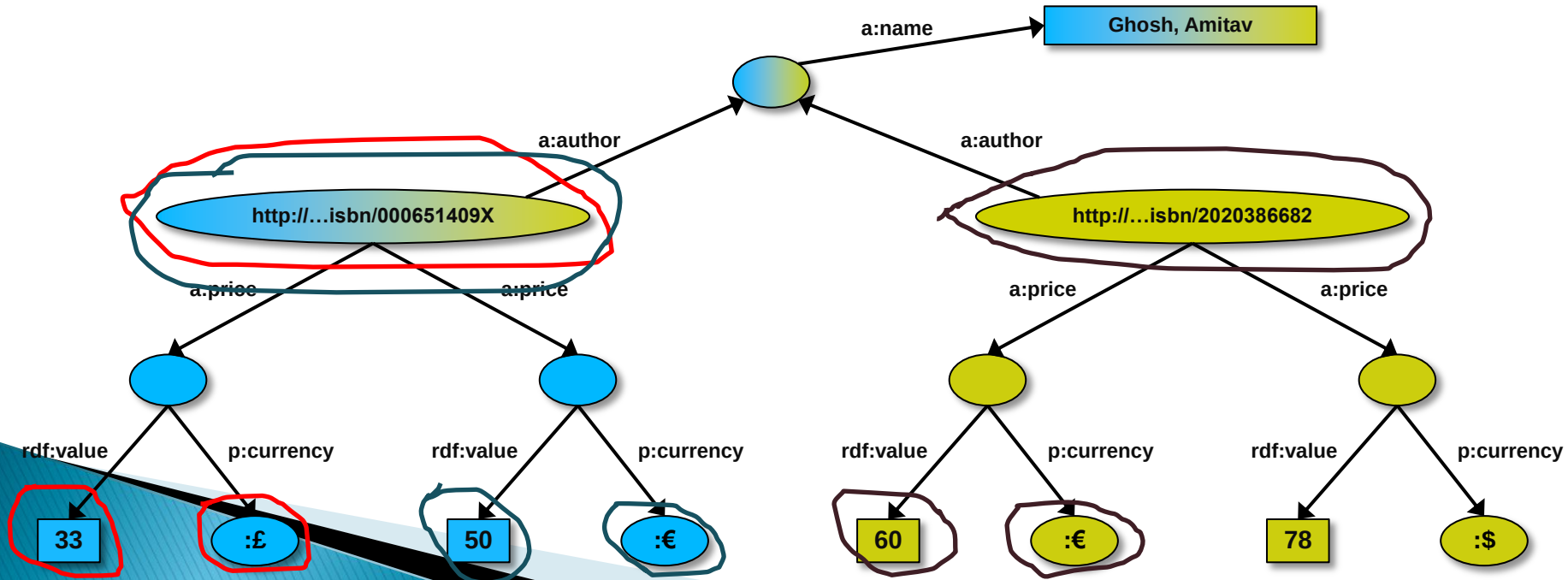
Ritorna: [<…409X>,33,:£], [<…409X>,50,:€]



Un semplice esempio SPARQL

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

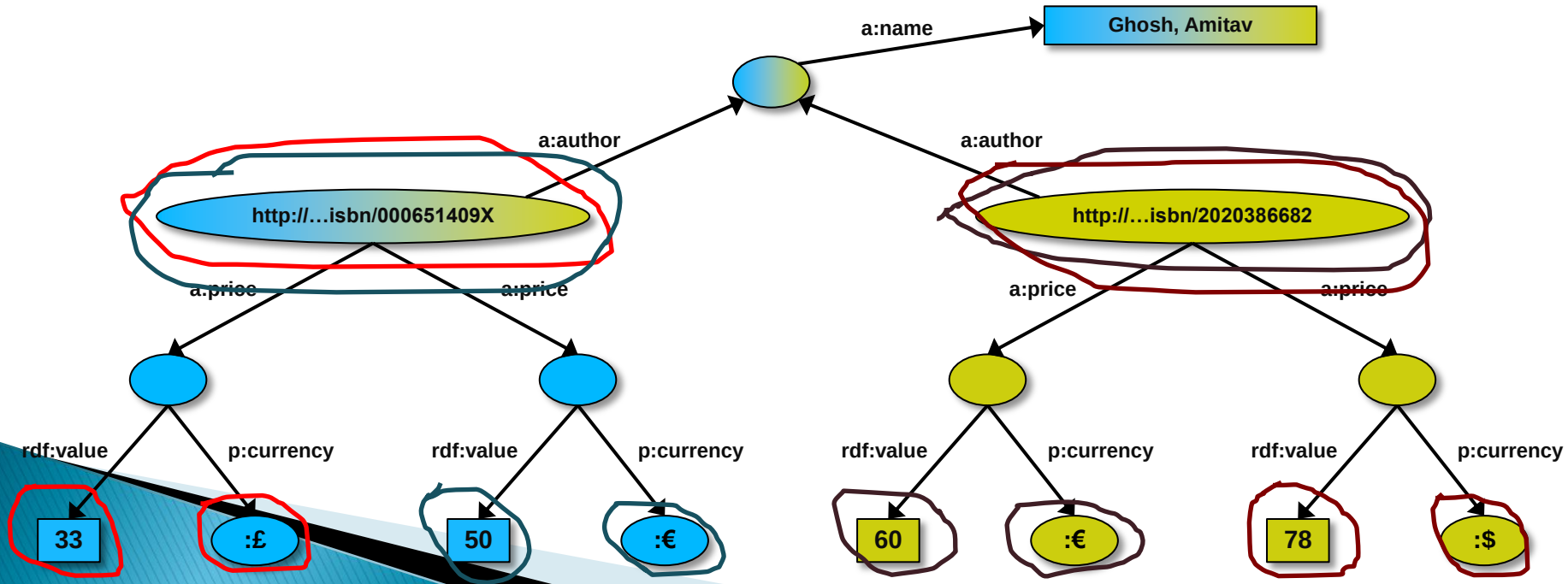
Ritorna: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€]



Un semplice esempio SPARQL

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

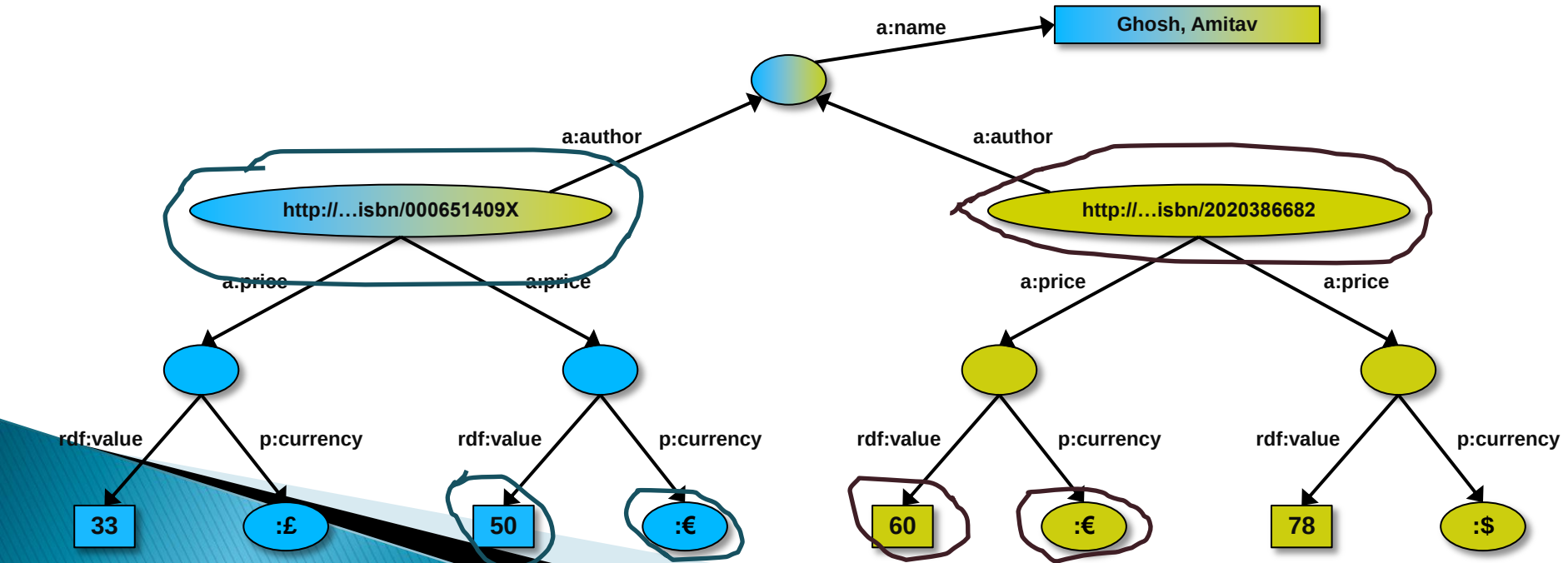
Ritorna: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€], [<…6682>,78,:$]



Vincoli sui Pattern

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == :€) }
```

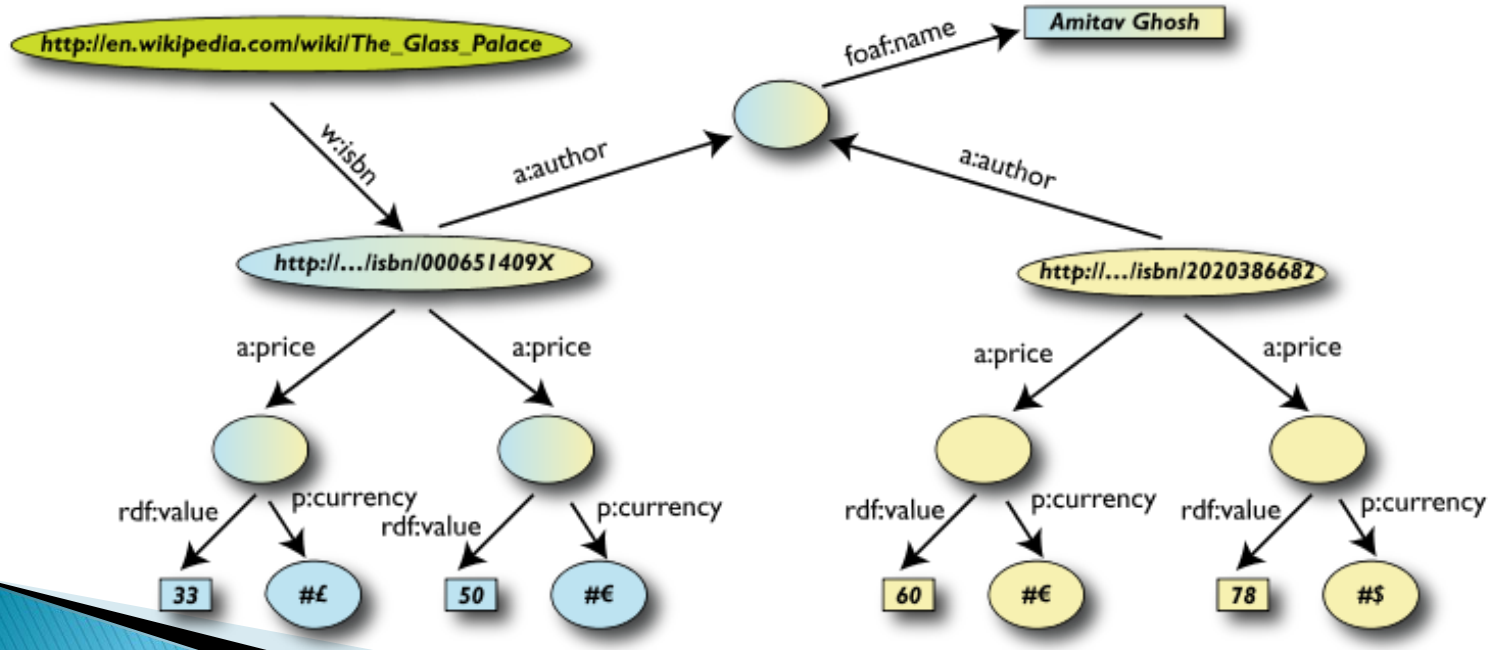
Ritorna: [<…409X>,50,:€], [<…6682>,60,:€]



Pattern Opzionale

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

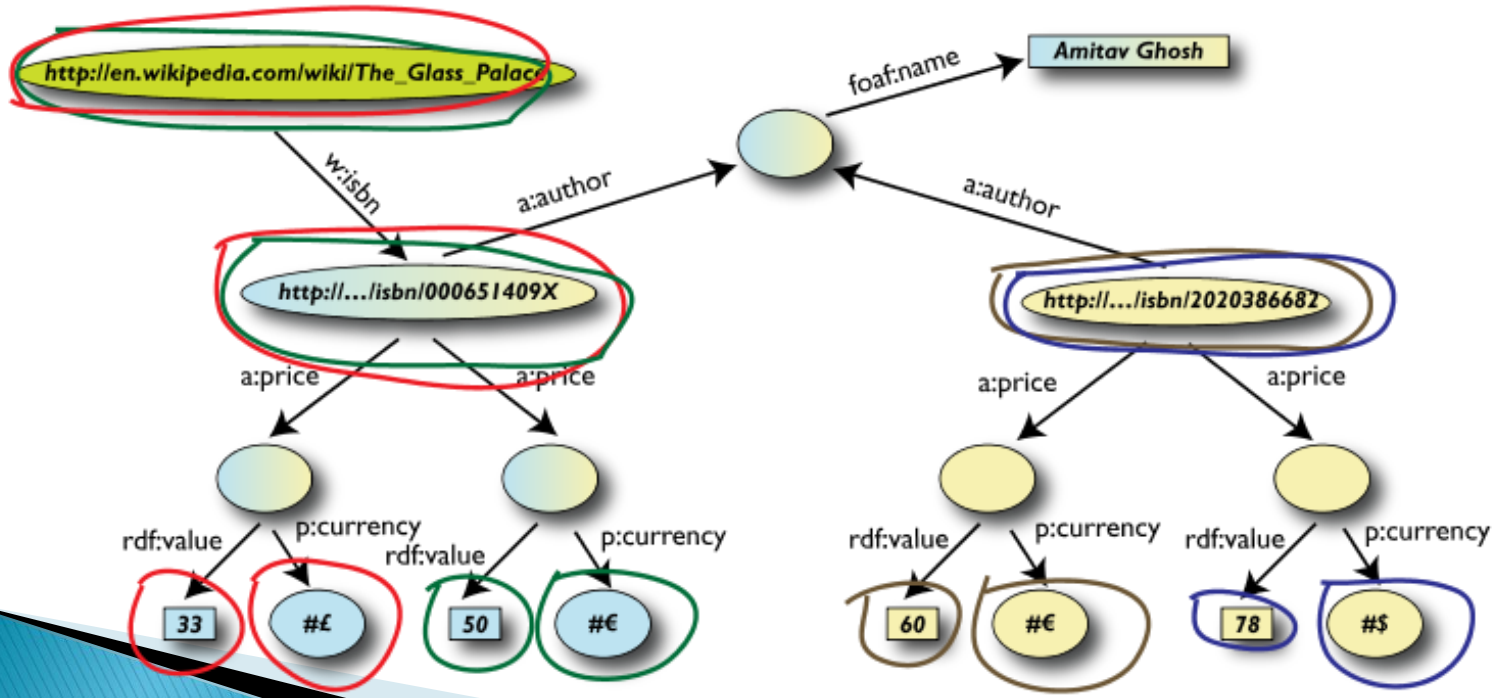
Ritorna: [[<..49X>,33,:£,<…Palace>], … ,
[<..6682>,78,:$,]]



Pattern Opzionale

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

Ritorna: [[<..49X>,33,:£,<…Palace>], … ,
[<..6682>,78,:$,]]



Altre carattristiche di SPARQL

- Si può limitare il numero di risultati restituiti, oridnarlo, rimuovere i duplicati …
- Si possono specificare diverse sorgenti dati (via URI-s) all'interno della query (in pratica viene effettuato un merge delle sorgenti!)
- Si può costruire un grafo combinando un pattern con i risultati di una query
- Si possono usare datatypes e/o tag di lingua nel match con un pattern

Utilizzo di SPARQL in pratica

- SPARQL è normalmente usato sulla rete
  - Documenti separati dello standard definiscono il protocollo di interrogazione e il formato del risultato:
    - SPARQL Protocol for RDF with HTTP and SOAP bindings
    - SPARQL results in XML or JSON formats
- (l'acronimo SPARQL sta per:
  SPARQL Protocol And RDF Query Language)
- I grandi dataset solitamente offrono uno "SPARQL endpoint" usando tale protocollo
  - Esempio tipico: endpoint SPARQL in Dbpedia

Esempio di remote query/reply

```
GET /qps?&query=SELECT+:…+WHERE:+… HTTP/1.1
User-Agent: my-sparql-client/0.0
Host: my.example

HTTP/1.1 200 OK
Server: my-sparql-server/0.0
Content-Type: application/sparql-results+xml

<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#>
  <head>
    <variable name="a"/>
    ...
  </head>
  <results>
    <result ordered="false" distinct="false">
      <binding name="a"><uri>http:…</uri></binding>
      ...
    </result>
    <result> ... </result>
  </results>
</sparql>
```

Uso di CONSTRUCT: per “concatenare” più query

```
CONSTRUCT {
    <http://dbpedia.org/resource/Amitav_Ghosh> ?p1 ?o1.
    ?s2 ?p2 <http://dbpedia.org/resource/Amitav_Ghosh>.
}
WHERE {
    <http://dbpedia.org/resource/Amitav_Ghosh> ?p1 ?o1.
    ?s2 ?p2 <http://dbpedia.org/resource/Amitav_Ghosh>.
}
```

- SPARQL endpoin
- ritorna RDF/XML

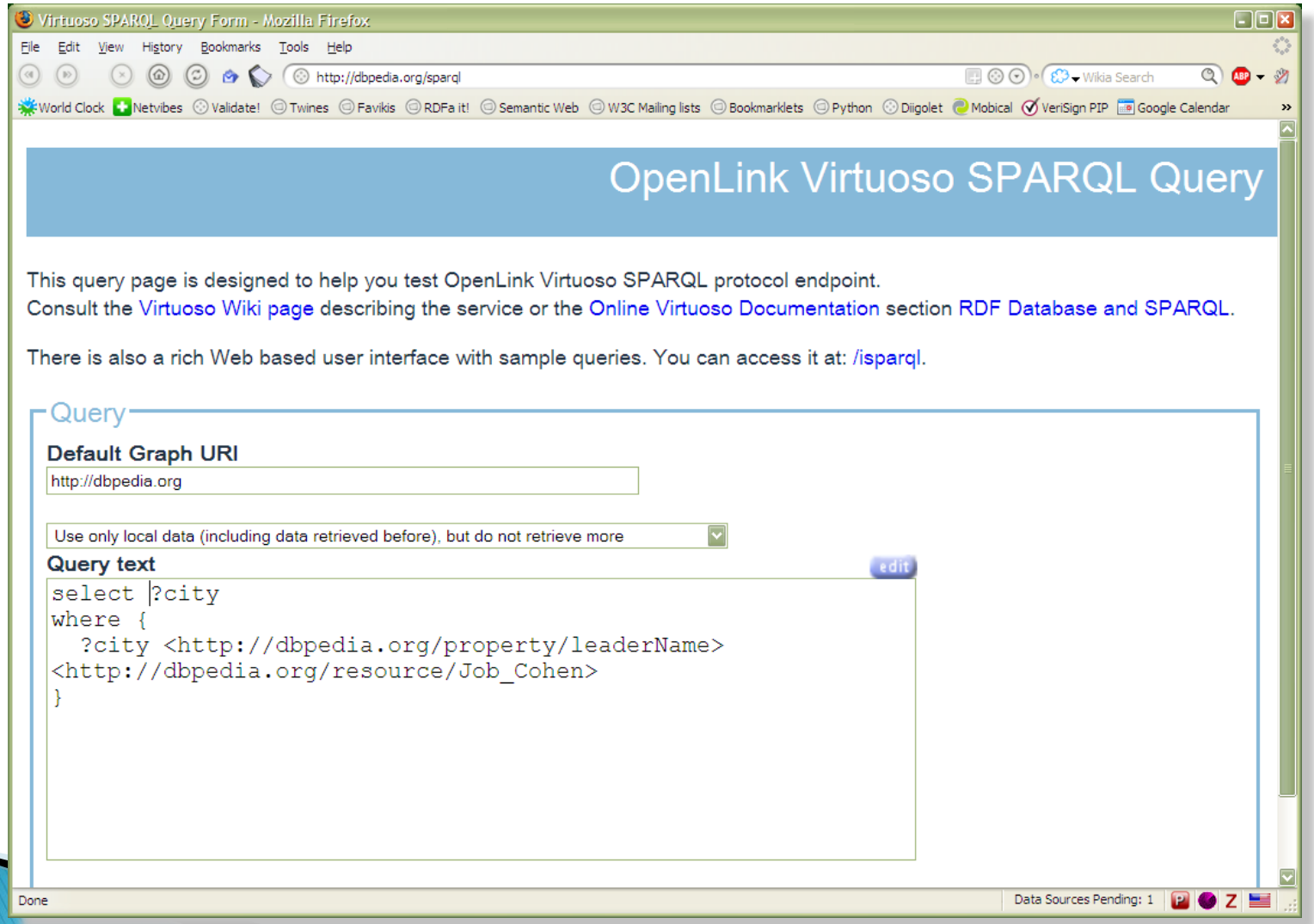
```
SELECT *
FROM <http://dbpedia.org/sparql/?query=CONSTRUCT+%7B++…>
WHERE {
  ?author_of dbpedia:author res:Amitav_Ghosh.
  res:Amitav_Ghosh  dbpedia:reference ?homepage;
                    rdf:type          ?type;
                    foaf:name         ?foaf_name.
  FILTER regex(str(?type),"foaf")
}
```

- Dati riusati in un’altra query…

Nuove caratteristiche in arrivo con la nuova versione SPARQL 1.1

- Alcuni aspetti sono ancora in lavorazione
  - Istruzioni di modifica di un triple store
  - Funzioni aggregate (sum, average, min, max, …)
  - Abbreviazioni per path expressions complesse
    - Simili alle espressioni regolari
  - controllo e/o descrizione del tipo di implicazione usato nel triple store (es. RDFS …)
  - …
- Arrivo previsto nel 2011

SPARQL-ing DBpedia

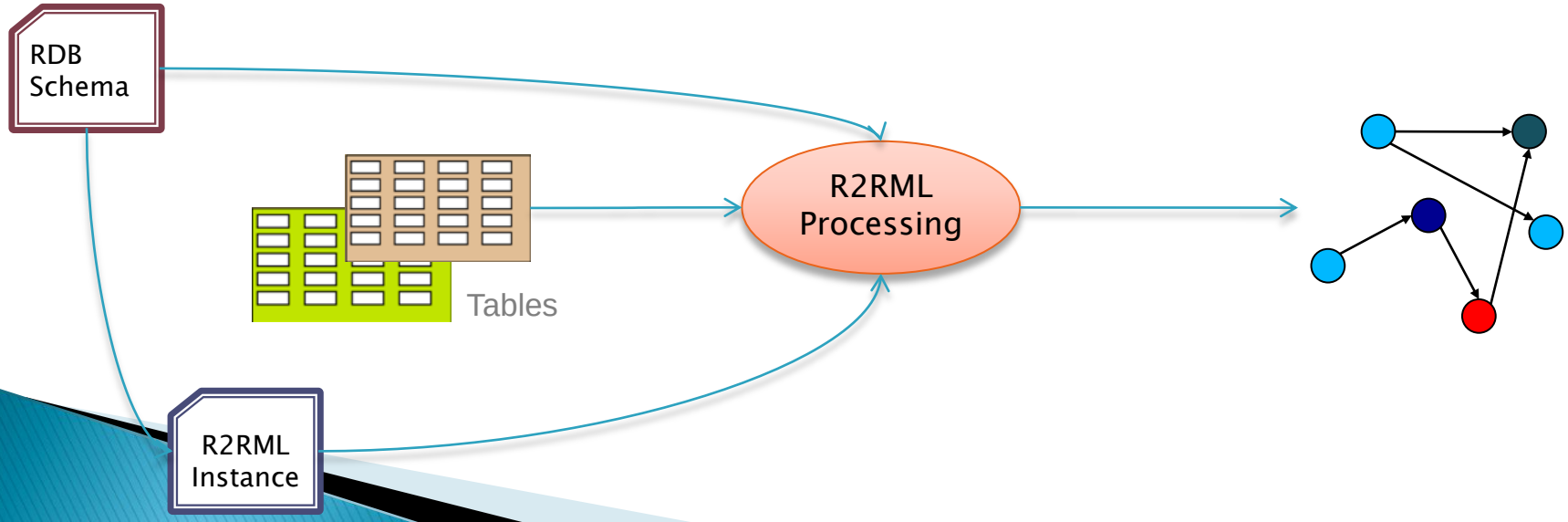


Accesso a DB relazionali con R2RML

- Definisce come una tabella relazionale è mappata in RDF
  - Definito in termini di uno schema RDB
    - Ogni riga è mappata su un subject comune
    - Le intestazioni delle colonne corrispondono ai predicate
    - Ciascuna cella è mappata su di un object
  - differenti tabelle *all'interno del medesimo database* sono inolte collegate all'interno del grafo

Cosa fa il processore R2RML

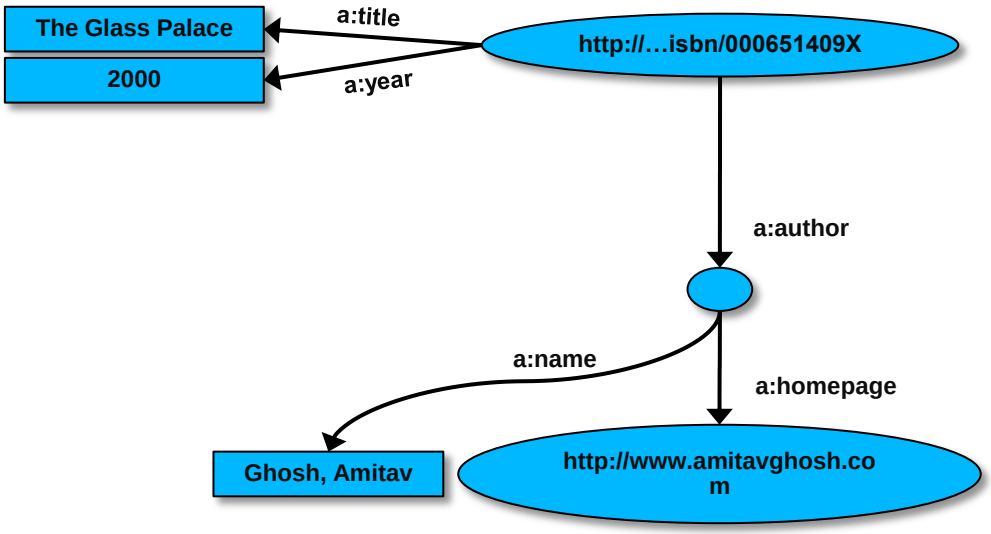
- Un processore R2RML ha accesso a:
  - Uno schema di RDB
  - Una istanza R2RML
  - un database governato dallo schema
- … e produce un grafo RDF



Tornando all'esempio del bookshop

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

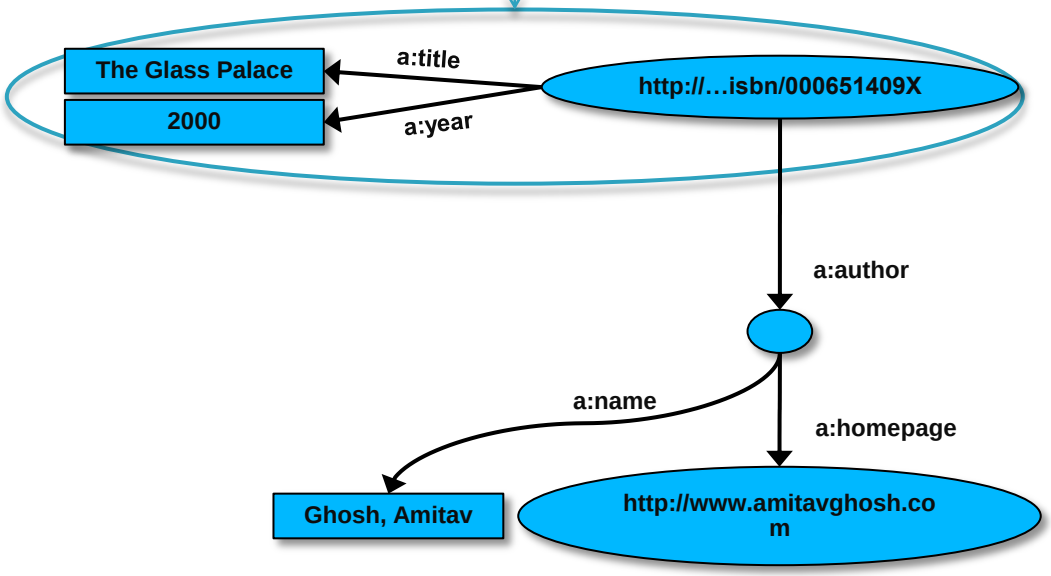
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Tornando all'esempio del bookshop

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

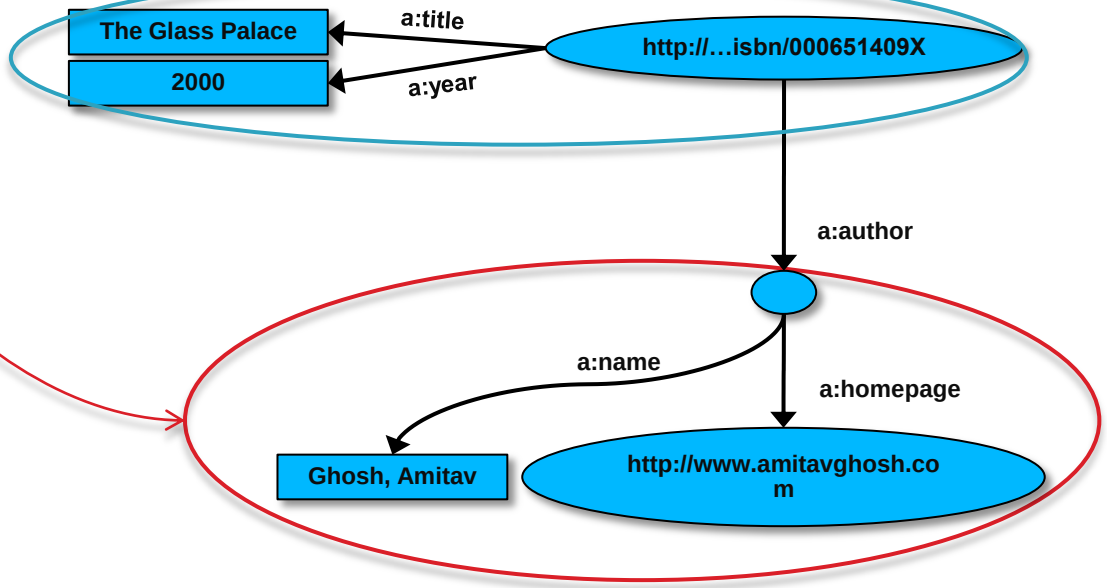
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



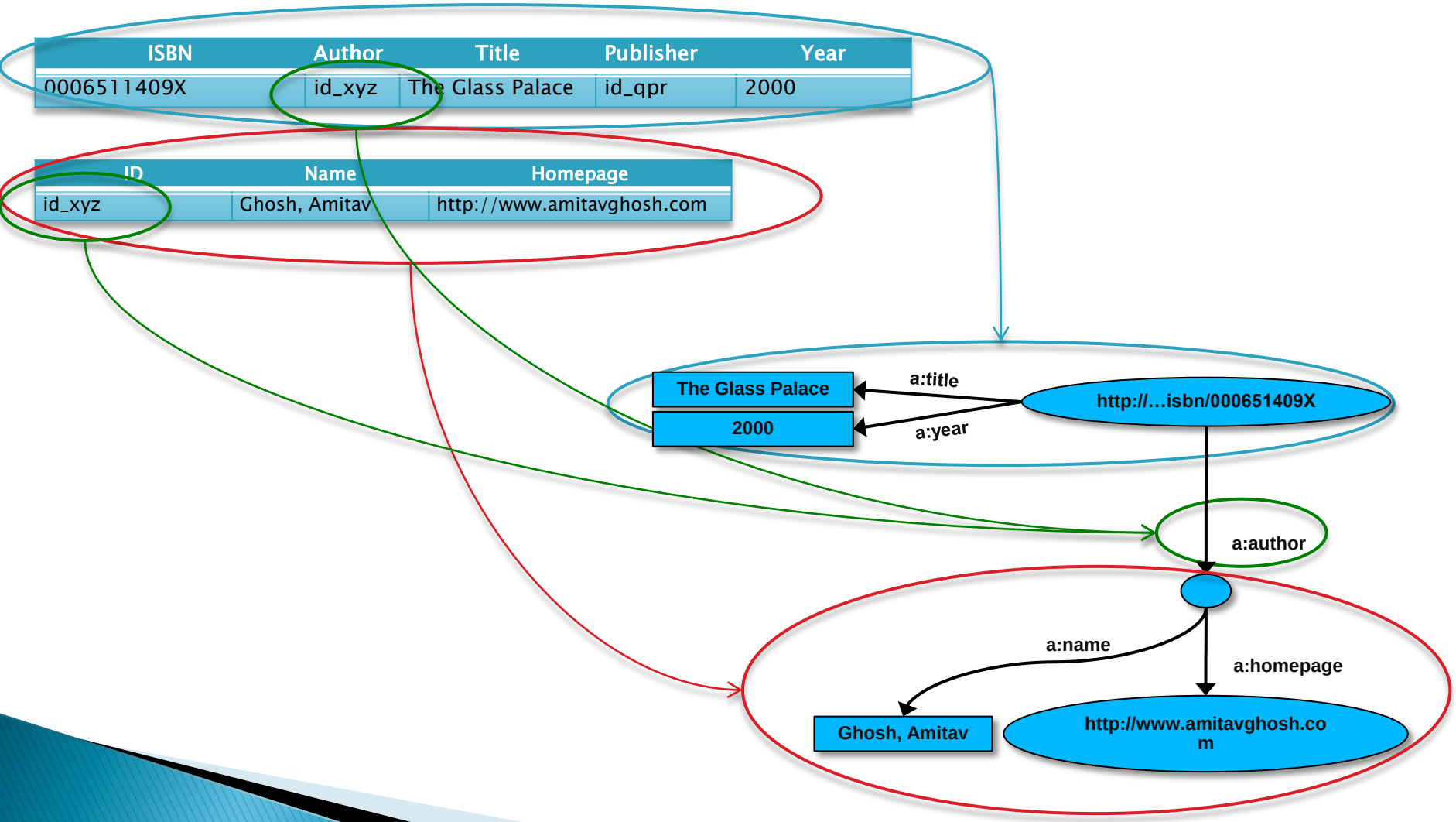
Tornando all'esempio del bookshop

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Tornando all'esempio del bookshop

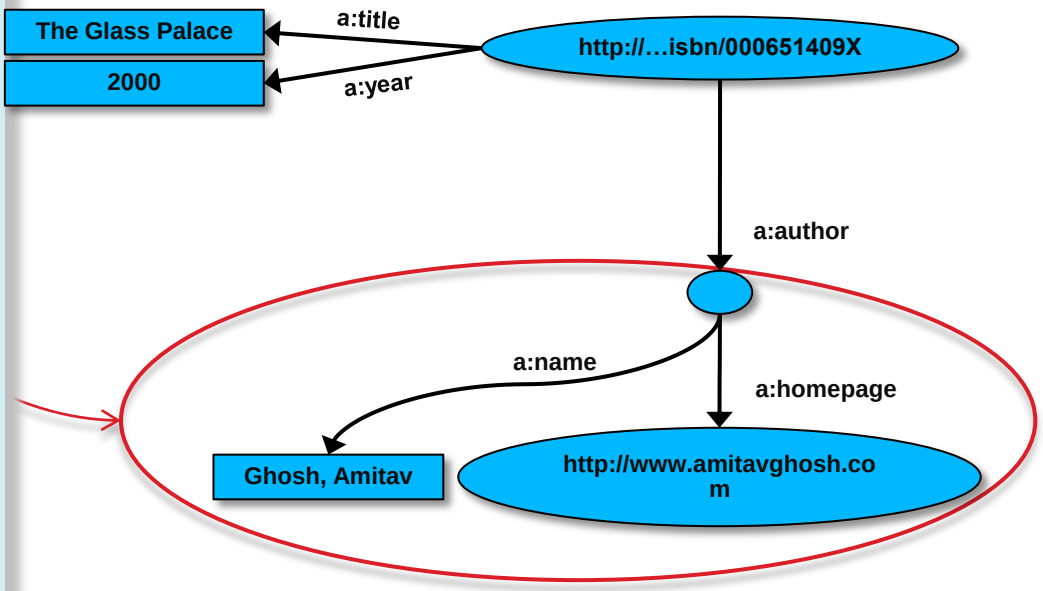


Step 1: trasforma la “Person Table”

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

```
:P_Table rdf:type rr:TriplesMap ;
  rr:subjectMap [
    rr:termtype "BlankNode" ;
    rr:column "ID" ;
  ] ;
  rr:predicateObjectMap [
    rr:predicateMap [
      rr:predicate a:name
    ];
    rr:objectMap [
      rr:column "Name"
    ]
  ] ;
  rr:predicateObjectMap [
    rr:predicateMap [
      rr:predicate a:homepage
    ];
    rr:objectMap [
      rr:column "Homepage" ;
      rr:termtype "IRI"
    ]
  ] ;
```

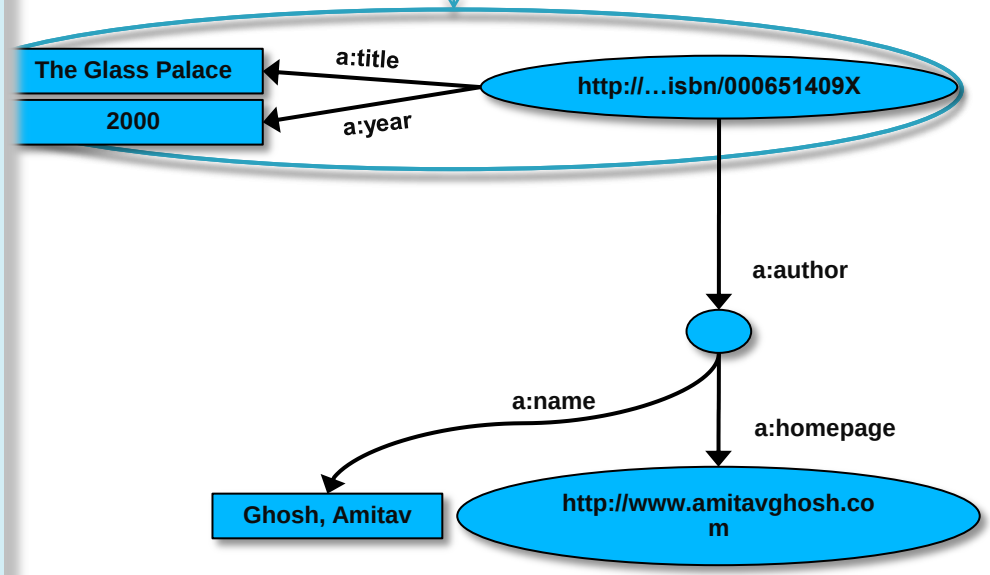


Step 2: trasforma la “Book Table”

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

```
:B_Table rdf:type rr:TriplesMap ;
  rr:subjectMap [
    rr:value [
      rr:prefix "http://...isbn/" ;
      rr:column "ISBN" ;
    ]
  ];
  rr:predicateObjectMap [
    rr:predicateMap [
      rr:predicate a:title
    ];
    rr:objectMap [
      rr:column "Title"
    ]
  ] ;
  rr:predicateObjectMap [
    rr:predicateMap [
      rr:predicate a:year
    ];
    rr:objectMap [
      rr:column "Year" ;
    ]
  ] ;
```

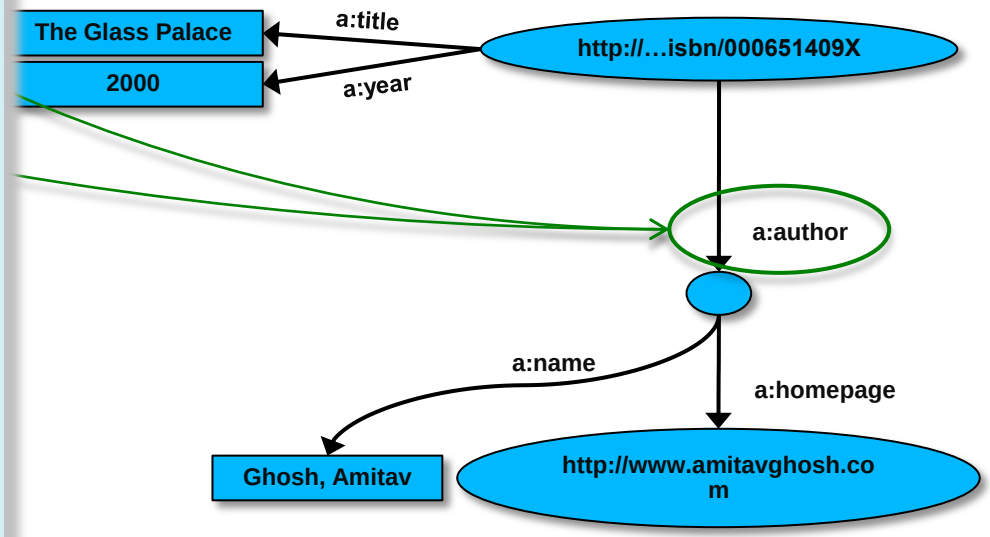


Step 3: “lega” le due tabelle

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

```
:B_Table a rr:TriplesMap ;
  ...
  rr:refPredicateObjectMap [
    rr:refPredicateMap [
      rr:predicate a:author
    ];
    rr:refObjectMap [
      rr:parentTriplesMap :P_Table ;
      rr:joinCondition
        "{child}.Author = {parent}.ID"
      ]
    ]
  ].
```



Ulteriori caratteristiche R2RML

- Ci sono ulteriori caratteristiche:
  - Assegnamento di un datatype ad un literal object
  - Assegnamenti di oggetto più complicati (es., per una specifica colonna l'oggetto assegnato è una cella di un'altra colonna)

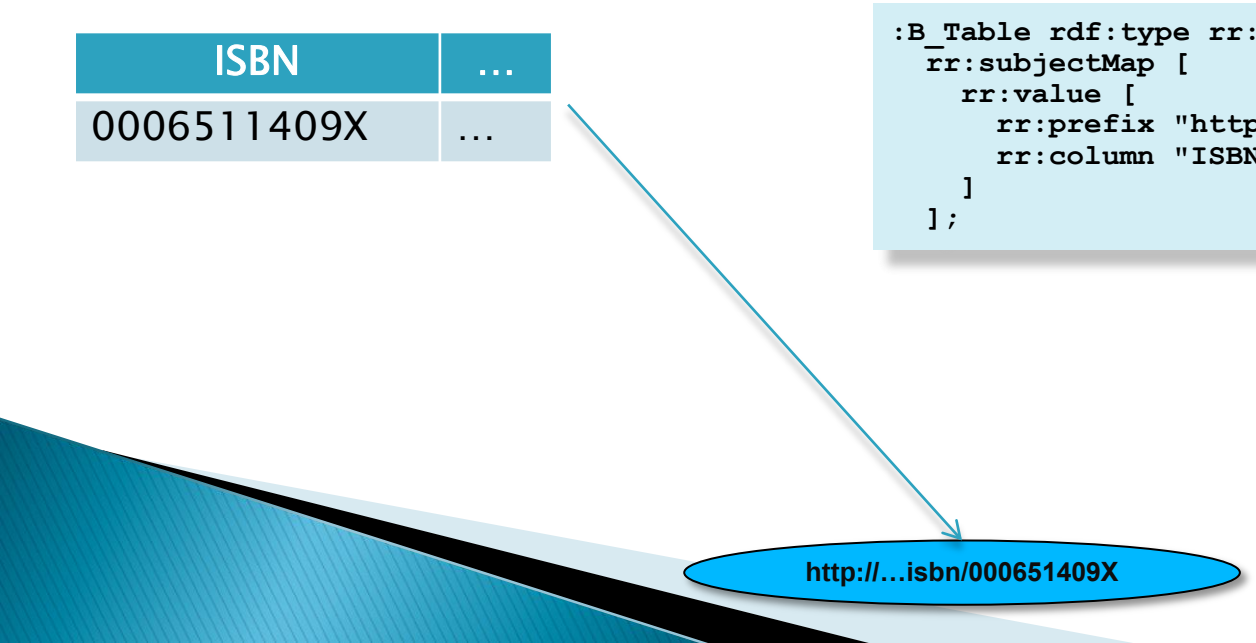
```
:B_Table rdf:type rr:TriplesMap ;
  ...
  rr:predicateObjectMap [
    ...
    rr:objectMap [
      rr:column "Year" ;
      rr:datatype xsd:year
    ]
  ] ;
```

Ulteriori caratteristiche R2RML: logical table

- Tornando all'esempio:

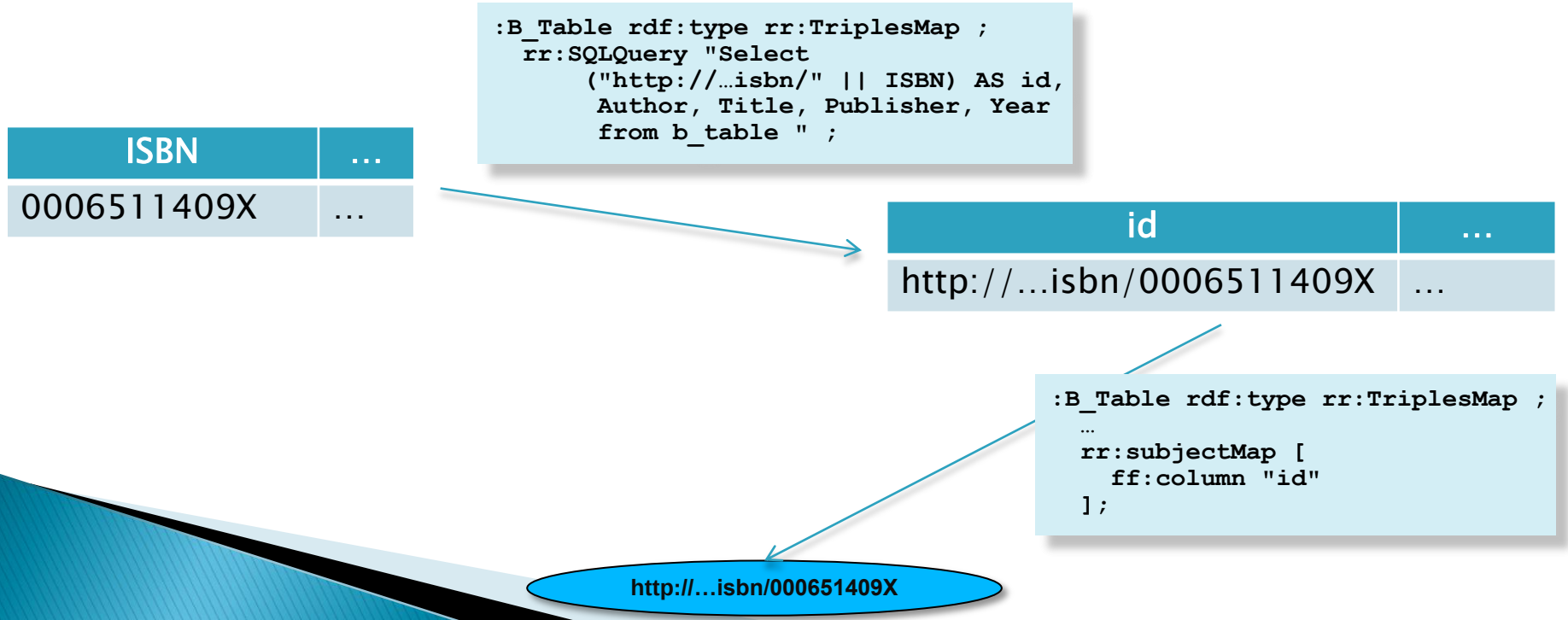
ISBN	…
0006511409X	…

```
:B_Table rdf:type rr:TriplesMap ;
  rr:subjectMap [
    rr:value [
      rr:prefix "http://...isbn/" ;
      rr:column "ISBN" ;
    ]
  ];
```



Ulteriori caratteristiche R2RML: logical table

- In alternativa si sarebbe potuto usare SQL
  - SI genera una “logical table”
  - Ogni altra definizione è relativa a tale logical table
- Nell’esempio è un’inutile complicazione, ma è uno strumento potente in casi complicati!



R2RML Direct Mapping

- Un processore R2RML può essere complesso
  - Per esempio, include un processore SQL
  - Tutto OK quando ci si connette a grandi RDB
- Un'alternativa: generare un puro grafo RDF
  - no transformazioni, solo link fra le tabelle, etc.
  - è l'equivalente di un R2RML "Nullo", ossia non c'è bisogno di specificarne uno
- Tale grafo può poi essere processato con RDFS, rule engines, etc.
- Questo metodo è propriamnete definito come "Direct Mapping"

Direct mapping delle tabelle bookstore

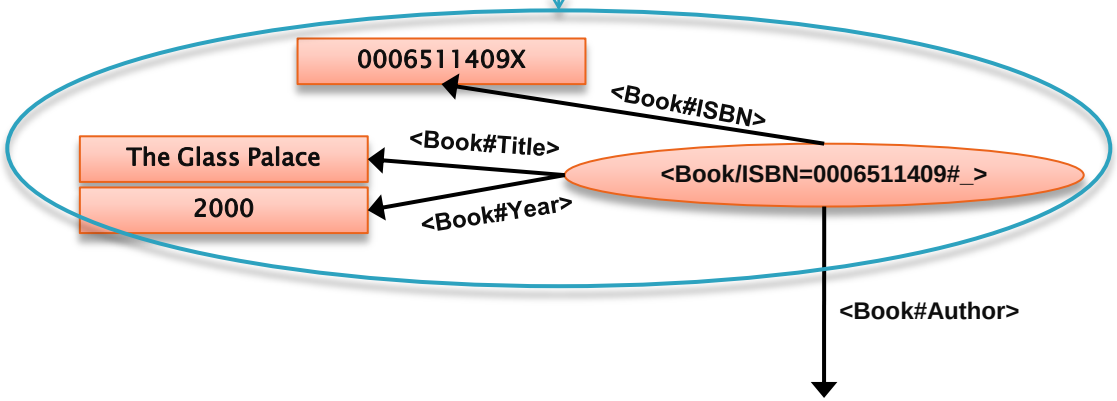
ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

Direct mapping delle tabelle bookstore

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

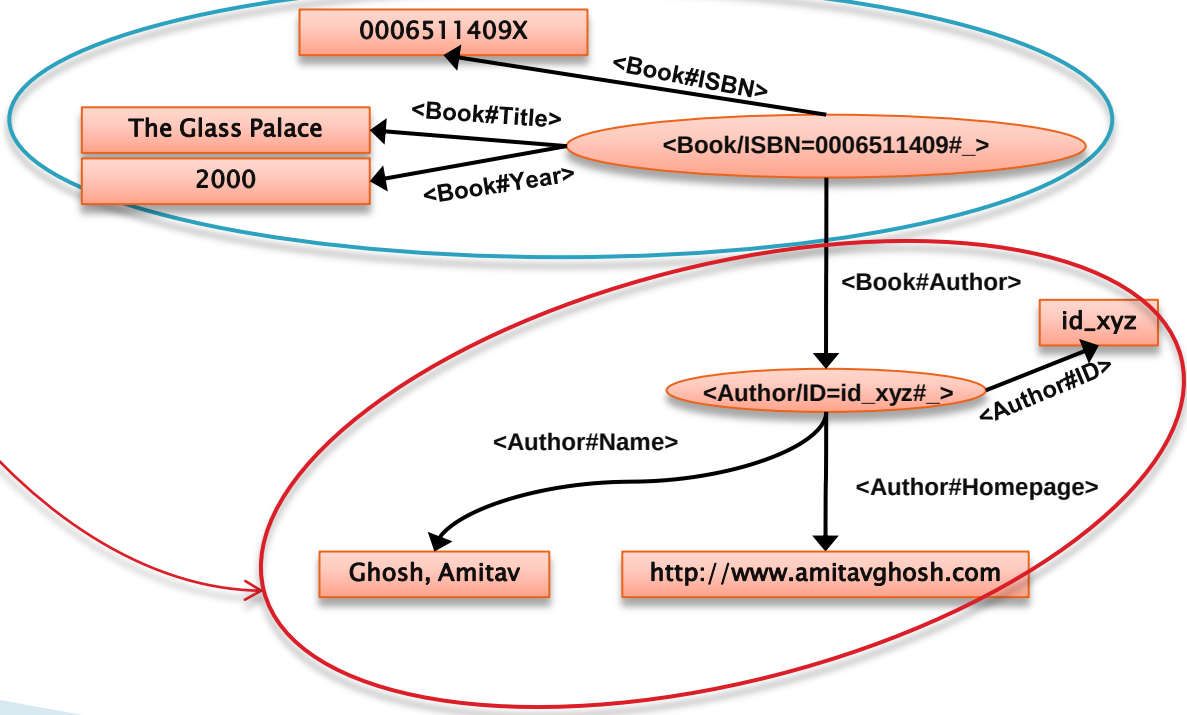
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



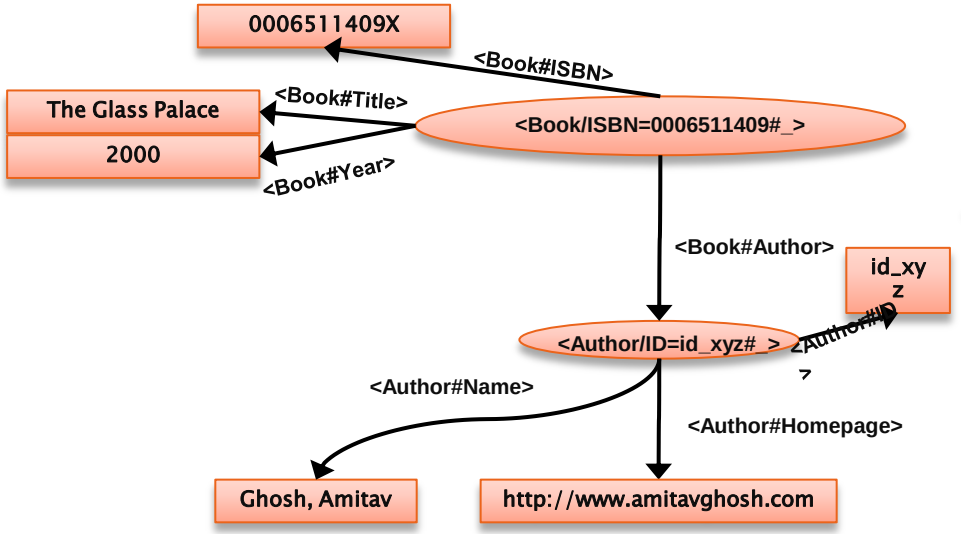
Direct mapping delle tabelle bookstore

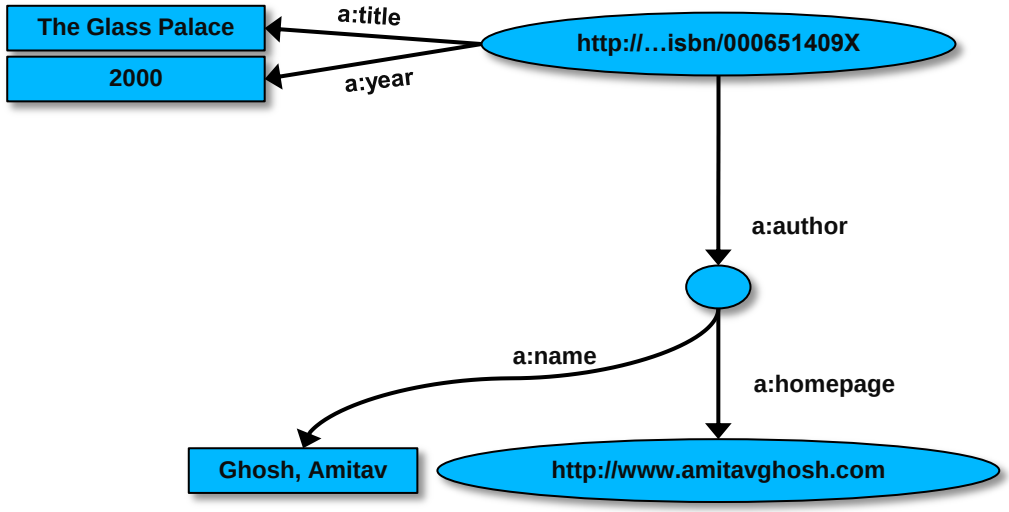
ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



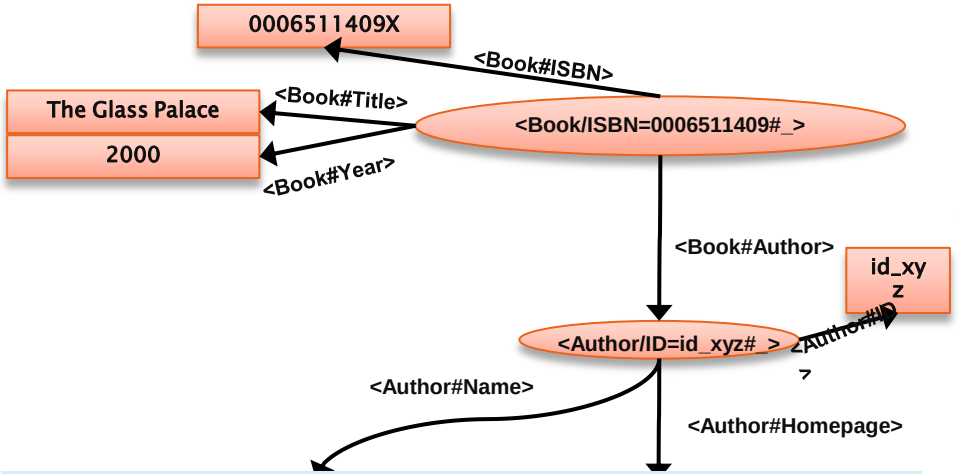
Trasformazione del grafo direzionato



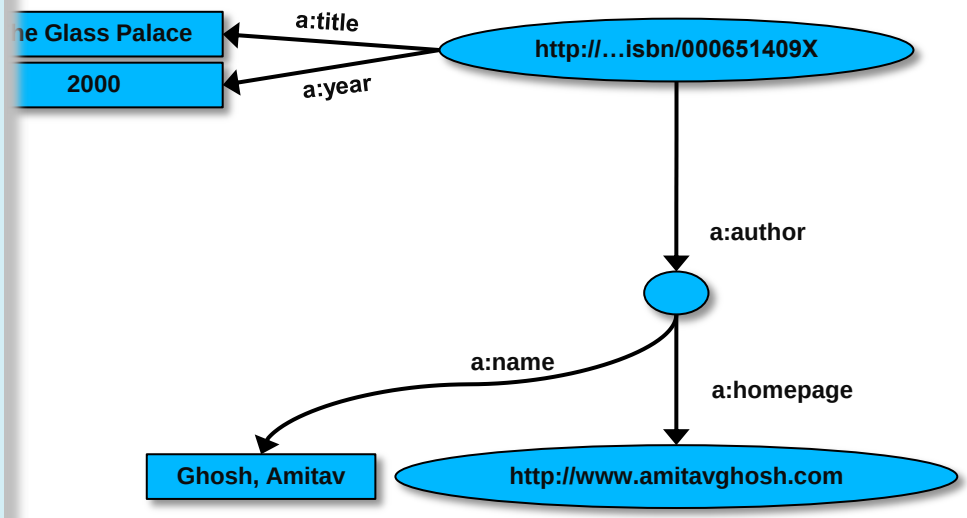


- Nomi di proprietà devono essere mappati
- Nuovi URI devono essere coniati
- Literal devono essere rimpiazzati da URI

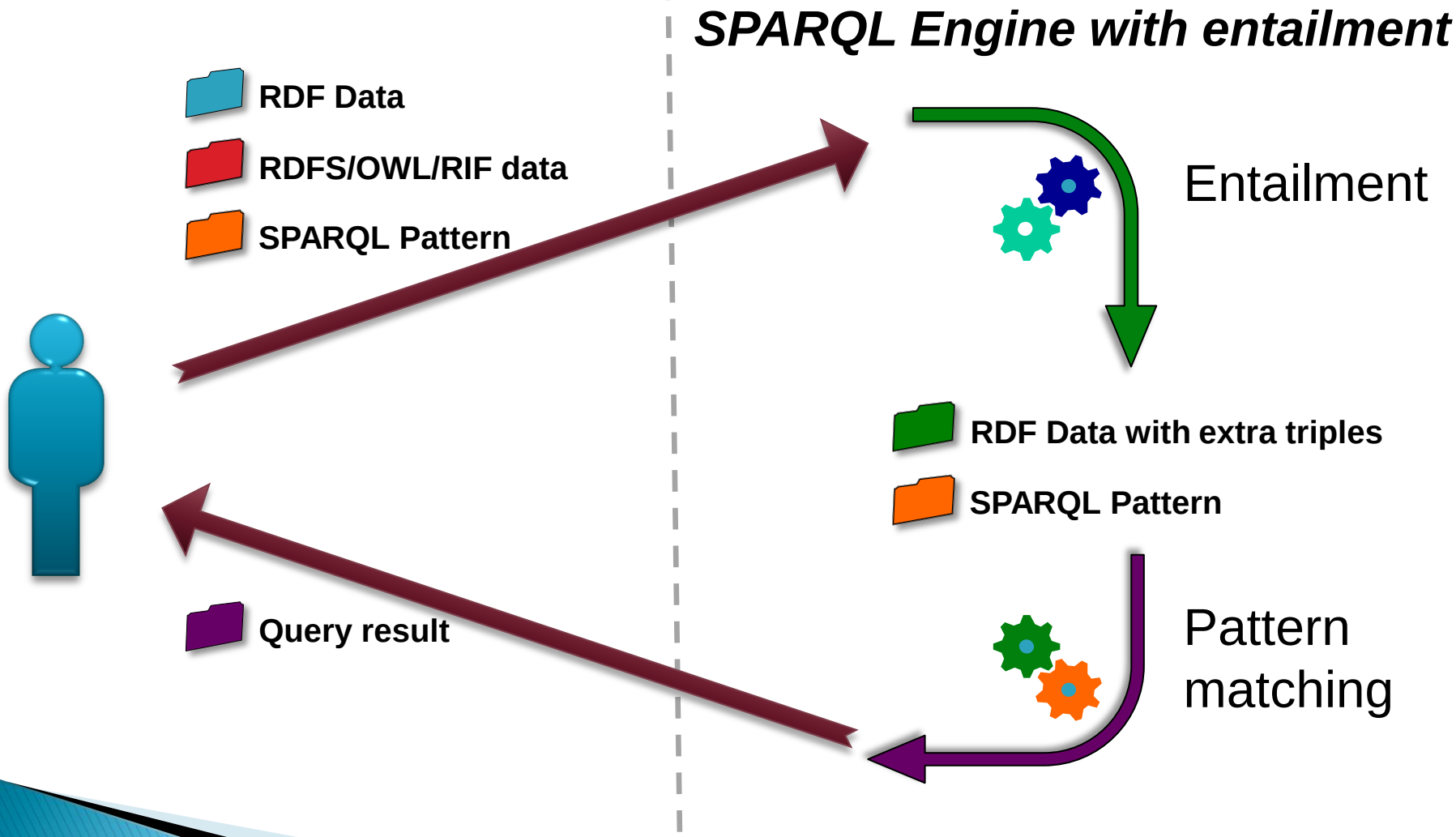
Transformazione con SPARQL 1.1



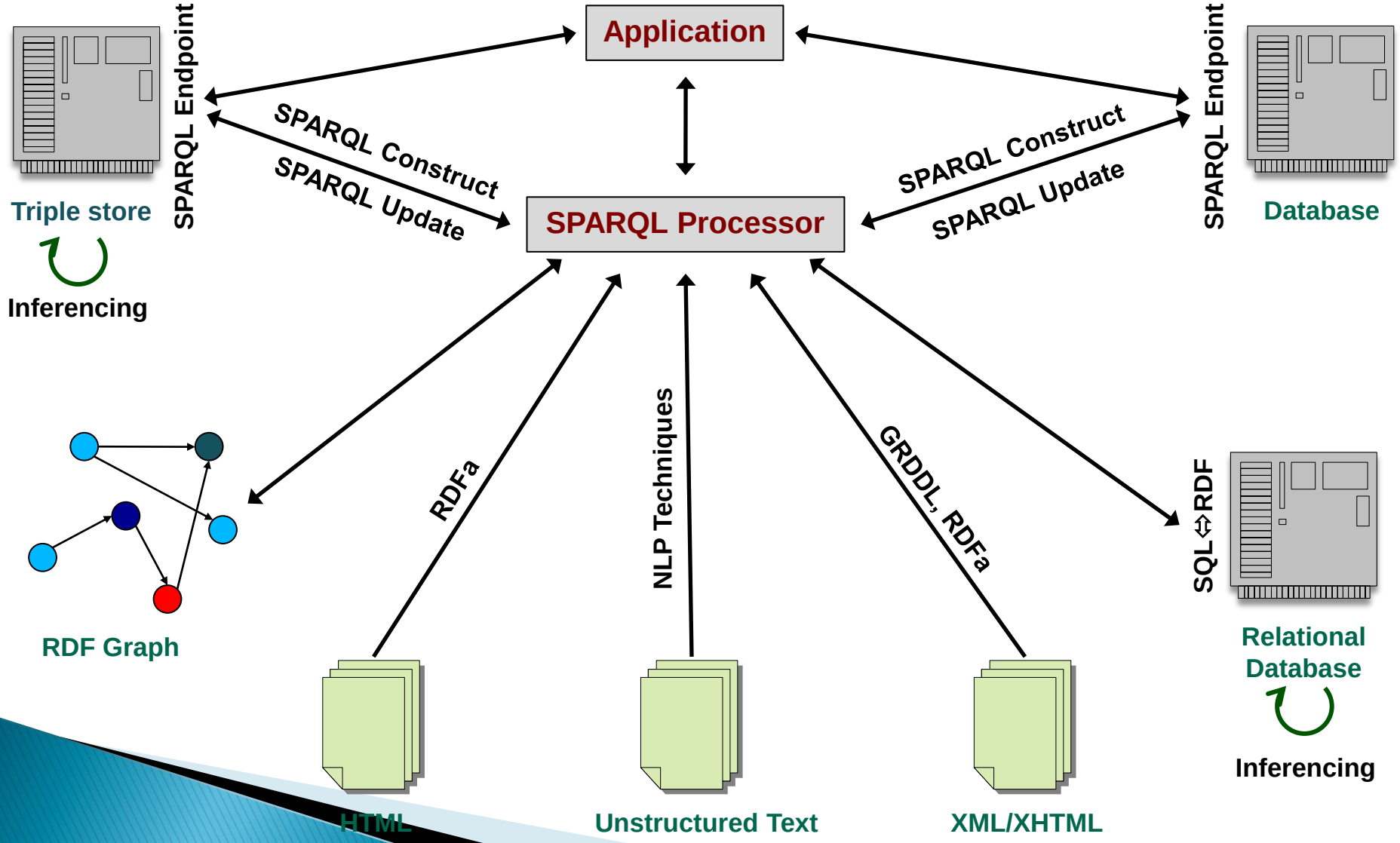

```
CONSTRUCT {
  ?id a:title  ?title ;
      a:year   ?year  ;
      a:author _:x    .
  _:x a:name ?name    ;
      a:homepage ?hp  .
}
WHERE {
  ?book
    <Book#ISBN>   ?isbn   ;
    <Book#Title>  ?title  ;
    <Book#Year>   ?year   ;
    <Book#Author> ?author .
  ?author
    <Author#Name>     ?name     ;
    <Author#Homepage> ?homepage .
  BIND (IRI(fn:concat("http://...",?isbn)) AS ?id)
  BIND (IRI(?homepage) AS ?hp)
}
```



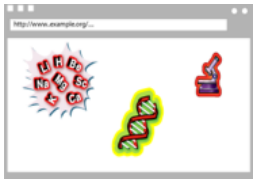
SPARQL 1.1 e RDFS/OWL/RIF



SPARQL 1.1 come punto di unificazione

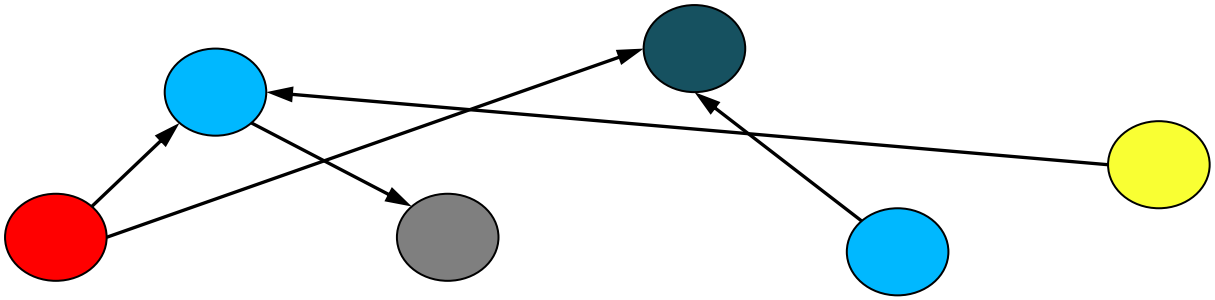


Ricordate l'esempio di integrazione?



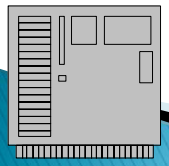
Applications

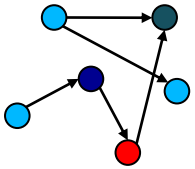
Manipulate
Query
…

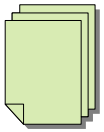


Data represented in abstract format

Map,
Expose,
…

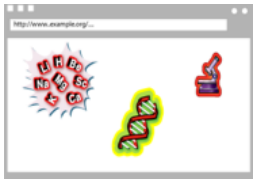






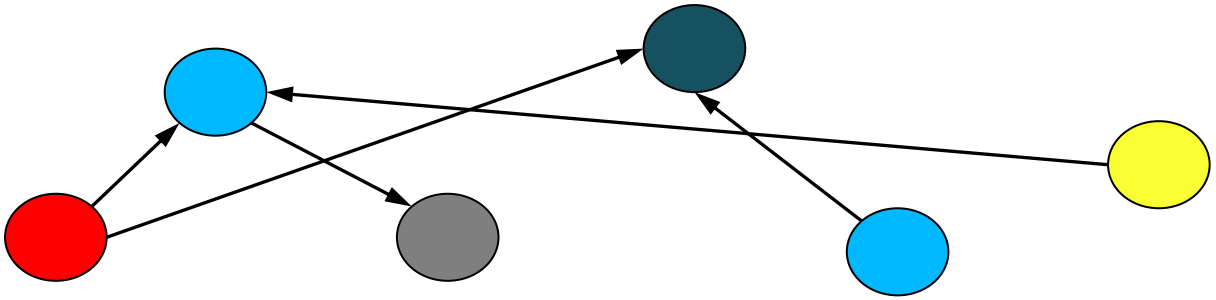
Data in various formats

Alla luce dei nuovi strumenti diventa:



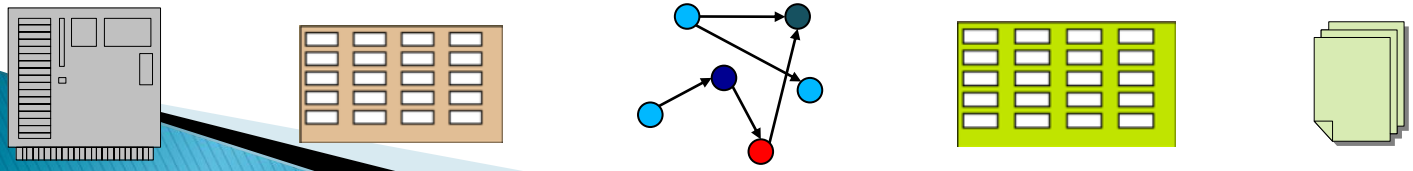
Applications

SPARQL, Inferences …



Data represented in RDF with extra knowledge (RDFS, SKOS, RIF, OWL,…)

R2RML, GRDDL, RDFa, …



Data in various formats

- Il Semantic Web è qui per realizzare l'integrazione di informazioni presenti sul Web
- Lo scopo finale è la creazione di una Web of Data