Summer School LDA

Libraries in the digital age: linked data technologies for a global knowledge sharing

Pula (Cagliari), 29 agosto - 1° settembre 2016

Querying RDF - SPARQL

Oreste Signore

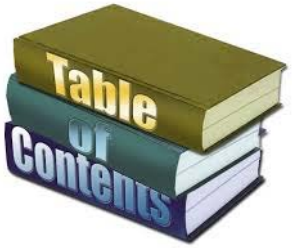
(W3C Italy)

Semantic Web

W3C SKOS

Slide a: http://www.orestesignore.eu/education/lda/slides/sparql.pdf

W3C WORLD WIDE WEB consortium Ufficio Italiano



Ringraziamenti

- Questa presentazione è basata in gran parte sul materiale di presentazioni tenute da Ivan Herman, già W3C Semantic Web Activity Lead
- Alcune parti relative ai database noSQL sono riprese da:
  - ✓ NoSQL An Introduction (http://www.w3resource.com/slides/introduction-to-the-nosql-world.php)
  - ✓ Materiale preso anche dal corso di basi si dati tenuto da Marco Di Felice: http://www.cs.unibo.it/%7Edifelice/dbsi/slides/pdf/20.pdf
- Il materiale di questa presentazione può essere riutilizzato nel rispetto delle leggi sul copyright e delle regole del W3C
- Un particolare ringraziamento agli organizzatori di Summer School LDA per avermi invitato a tenere questo seminario

Contenuto

- I database NoSQL
- Graph database
- Interrogare RDF: SPARQL
- Conclusioni

Database NoSQL

- **NoSQL**
  - ✓ **Movimento che promuove l'adozione di DMBS non basati sul modello relazionale**
- **Il termine NOSQL appare per la prima volta in una pubblicazione di Carlo Strozzi nel 1998.**
- **Oggi, il termine NOSQL viene usato per lo più nell'accezione NoT Only SQL**
  - ✓ ***"Next generation databases mostly addressing some of the points: being non-relational, distributed, open source and horizontally scalable"***
    (definizione da http://nosql-database.org/)

Motivazioni per database NoSQL

- Database distribuiti
- Strumenti generalmente open-source
- NON dispongono di schema
- NON supportano operazioni di join
- NON implementano le proprietà ACID delle transazioni
- Sono scalabili orizzontalmente
- Sono in grado di gestire grandi moli di dati
- Supportano le repliche dei dati

ACID

- **Atomicity, Consistency, Isolation, e Durability**
- **Perché le transazioni operino in modo corretto sui dati è necessario che i meccanismi che le implementano soddisfino queste quattro proprietà:**
  - ✓ **atomicità: la transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere o totale o nulla, non sono ammesse esecuzioni parziali;**
  - ✓ **coerenza: quando inizia una transazione il database si trova in uno stato coerente e quando la transazione termina il database deve essere in un altro stato coerente, ovvero non deve violare eventuali vincoli di integrità, quindi non devono verificarsi contraddizioni (*inconsistenza*) tra i dati archiviati nel DB;**
  - ✓ **isolamento: ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione;**
  - ✓ **durabilità: detta anche persistenza, si riferisce al fatto che una volta che una transazione abbia richiesto un *commit work*, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log dove sono annotate tutte le operazioni sul DB.**

Perché i database NoSQL

- Gestione dei Big-data
- Limitazioni del modello relazionale
- Teorema CAP

Big data

- **Moli di dati eterogenee, destrutturate, difficili da gestire mediante tecnologie tradizionali (RDBMS)**
- **Big-data denota sia le tipologie di dati che le tecnologie e i tool relativi**
- **Aspetti essenziali:**
  - ✓ **Volume (es. esperimenti di fisica delle alte energie)**
  - ✓ **Velocità (es. Stream di dati per monitoraggio nel settore della salute)**
  - ✓ **Varietà (dati eterogenei, multi-sorgente)**

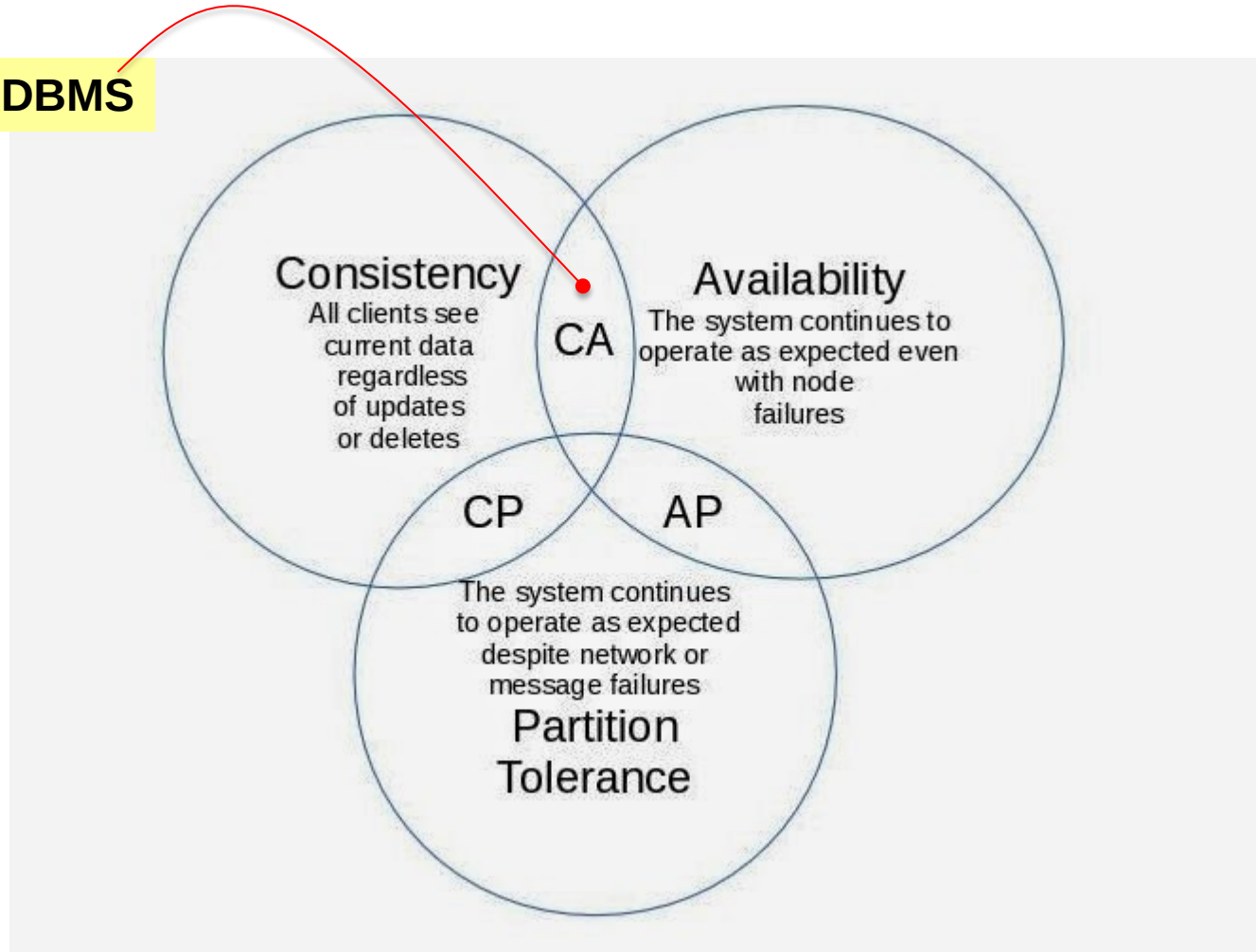
Limitazioni del modello relazionale

- **Il modello relazionale presuppone di modellare la realtà sotto forma di relazioni (rappresentate come tabelle)**
  - ✓ **Come gestire dati che non sono adatti a questo tipo di rappresentazione? (es. una pagina HTML)**
- **Alcune operazioni non possono essere implementate in SQL**
  - ✓ **(es. Memorizzazione di un grafo, percorso minimo tra due punti)**
- **Scalabilità orizzontale dei DBMS relazionali**
  - ✓ **Scalabilità: capacità di un sistema di migliorare le proprie prestazioniper un certo carico di lavoro, quando vengono aggiuinte nuove risorse al sistema**
    - ***Scalabilità verticale*: aggiungere più potenza di calcolo ai nodi che gestiscono il DBMS**
    - ***Scalabilità orizzontale*: aggiungere più nodi al cluster**

Il teorema CAP

- **In informatica teorica, il teorema CAP, noto anche come teorema di Brewer, afferma che un sistema informatico distribuito può soddisfare al massimo solo due delle tre proprietà:**
  - ✓ ***Coerenza***
    - **tutti i nodi vedono gli stessi dati nello stesso momento**
  - ✓ ***Disponibilità***
    - **il servizio è sempre disponibile**
  - ✓ ***Tolleranza di partizione***
    - **il sistema continua a funzionare correttamente anche in presenza di perdita di messaggi o di partizionamenti della rete**

Il teorema CAP (cont)



Proprietà base dei DB NoSQL

- **Basically Available**
  - ✓ **I nodi del sistema distribuito possono essere soggetti a guasti, ma il servizio è sempre disponibile**
- **Soft State**
  - ✓ **Non è garantita in ogni istante la coerenza dei dati**
- **Eventually Consistent**
  - ✓ **Il sistema diventa coerente dopo un certo intervallo di tempo, se cessano le attività di modifica**

Modelli logici dei database NoSQL

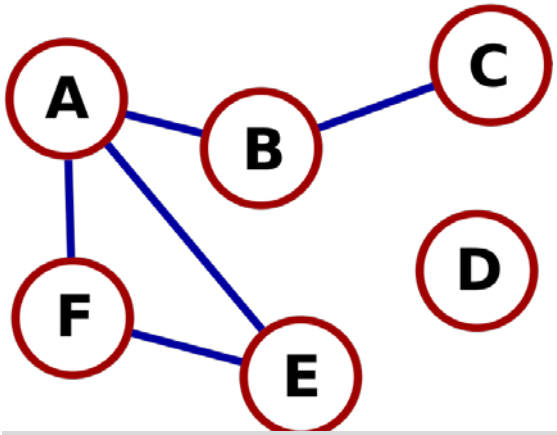
- **Database chiave/valore**
  - ✓ Dati come liste di coppie chiave/valore
    - Chiave: valore univoco
    - Valore: Qualsisi cosa
- **Database document-oriented**
  - ✓ Documento: coppie chiave/valore
- **Database column-oriented**
  - ✓ Dati organizzati su colonne invece che su righe
- **Database graph-oriented**
  - ✓ Dati strutturati sotto forma di grafi

Chi li usa?

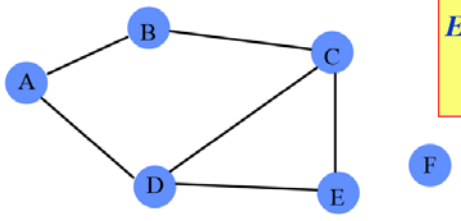
- Google
- Facebook
- Mozilla
- Adobe
- Foursquare
- Linkedin
- ...

Un grafo è...

- **Un insieme di elementi detti *nodi* o *vertici* collegati fra loro da *archi* o *lati*.**
- **Un insieme di nodi e di relazioni (relationship) che li connettono**



Di Pac72 - Opera propria, Pubblico dominio, https://commons.wikimedia.org/w/index.php?curid=2346492



$V = \{A, B, C, D, E, F\}$
$E = \{(A,B), (A,D), (B,C), (C,D), (C,E), (D,E)\}$

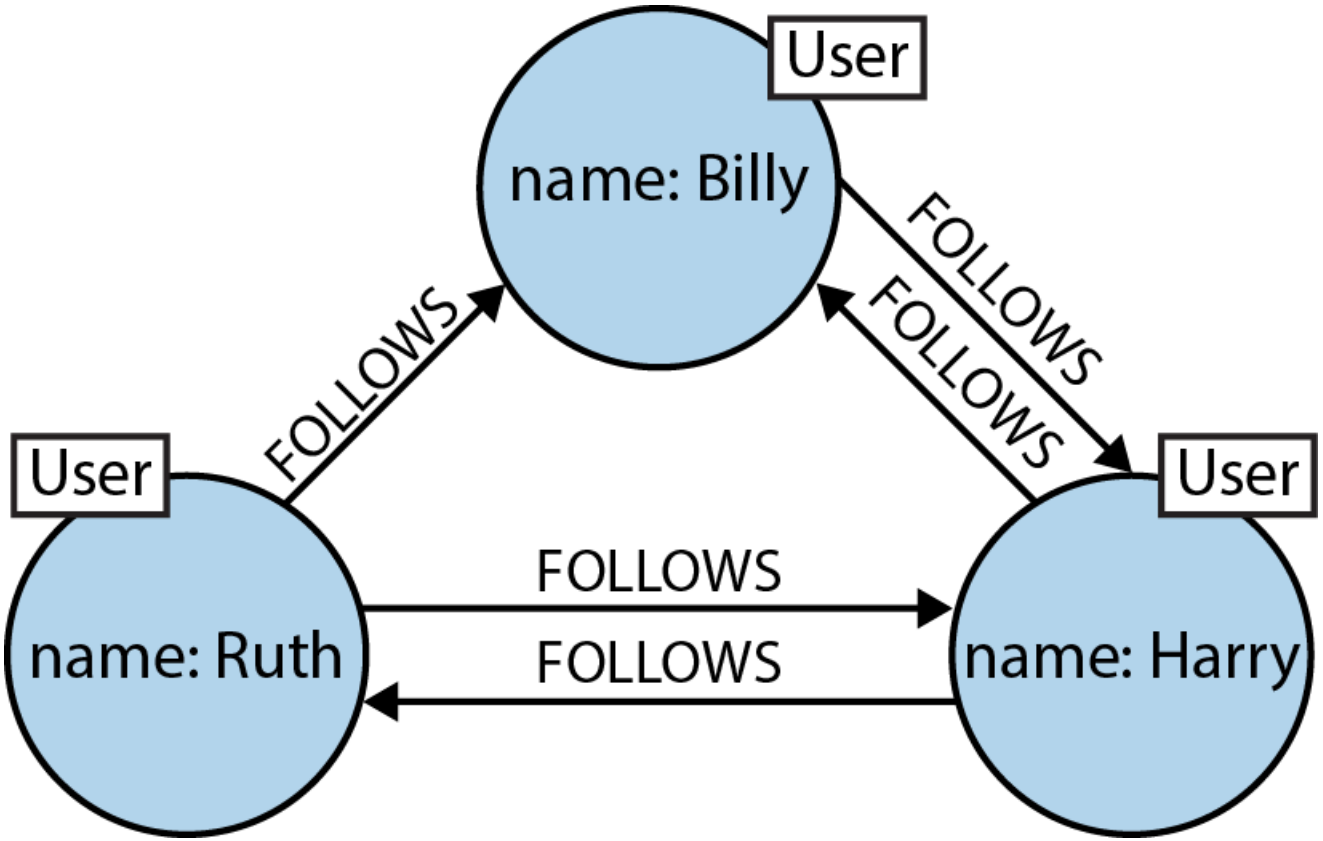
Un grafo non orientato è...

- **Un grafo non orientato G=(V,E) è una raccolta di elementi distinti chiamati vertici v∈V o nodi e coppie di vertici distinte e non ordinate (u,v) ∈ E=VxV (relazione simmetrica), dette archi**
- **Graficamente rappresentiamo i vertici come punti o cerchi e gli archi con linee che collegano coppie di vertici**

Un grafo orientato è...

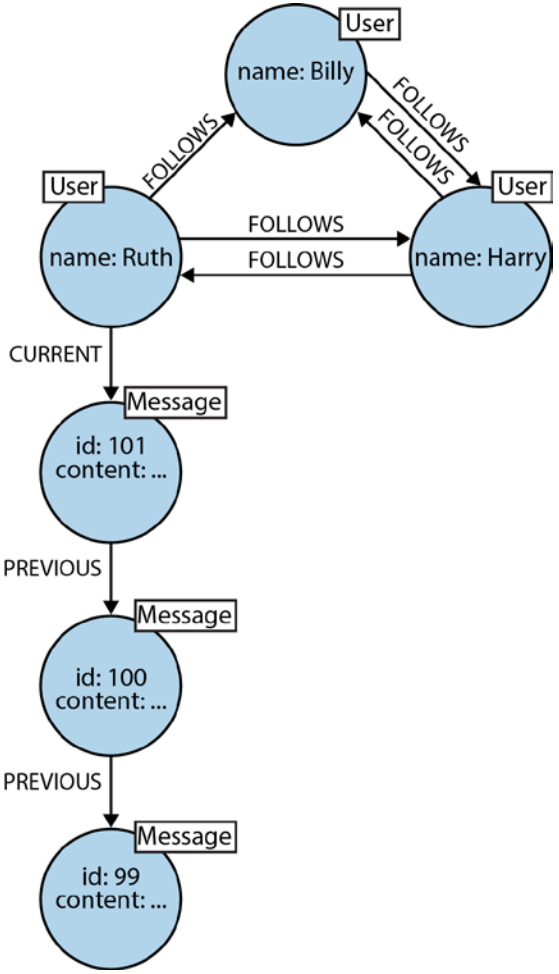
- Un grafo orientato o diretto G=(V,E) (digrafo) è una raccolta di vertici (o nodi) v∈V e archi <u,v> ∈E=VxV (relazione non simmetrica), in cui gli archi sono coppie ordinate di nodi
- Rappresentiamo graficamente gli archi con frecce che vanno dal primo al secondo nodo

Un piccolo social graph



Potenza espressiva

- Ruth ha pubblicato una serie di messaggi
- Il messaggio più recente si trova seguendo la relationship CURRENT
- I messaggi precedenti si trovano seguendo la relationship PREVIOUS



Labeled property graph

- Contiene nodi e relationship
- I nodi contengono proprietà (key-value pairs)
- I nodi possono essere etichettati con una o più etichette (label)
- Le relazioni (relationship)
  - ✓ hanno un nome e un verso
  - ✓ hanno sempre un nodo di partenza e di arrivo (start e end node)
- Le relazioni possono contenere delle proprietà

Graph database

- **Un DBMS online con metodi Create, Read, Update e Delete (CRUD) che espone un data model a grafo**
- **Proprietà di un graph database:**
  - ✓ **Underlying storage**
    - **native graph storage**
    - **grafi serializzati su DB relazionali o Object Oriented o altri sistemi general purpose**
  - ✓ **Processing engine**
    - ***index-free adiacency* (puntatori diretti) – native graph processing**
    - ***esposizione di un modello a grafo* con operazioni CRUD**

Interrogare RDF

RDF data access

- Per applicazioni semplici l'approccio HTTP/RESTful è più che sufficiente
- Per insiemi di dati più rilevanti, aumenta la complessità delle relazioni
  - ✓ diventa necessario poter formulare query più sofisticate
- Come interrogare i dati RDF?

Querying RDF graphs

- **In pratica, dobbiamo poter rispondere a domande complesse sui dati RDF.**
  Per esempio:
  - ✓ **“dammi le coppie di risorse (a,b) per le quali esiste una risorsa x per cui valgono le proprietà (a figlioDi x) e (b germanoDi x)” (quindi le coppie (persona,zio/a))**
    - **le regole possono diventare piuttosto coplesse**
- **Questo è l’obiettivo di SPARQL (Query Language for RDF)**
  - ✓ **SPARQL = SPARQL Protocol and RDF Query Language**

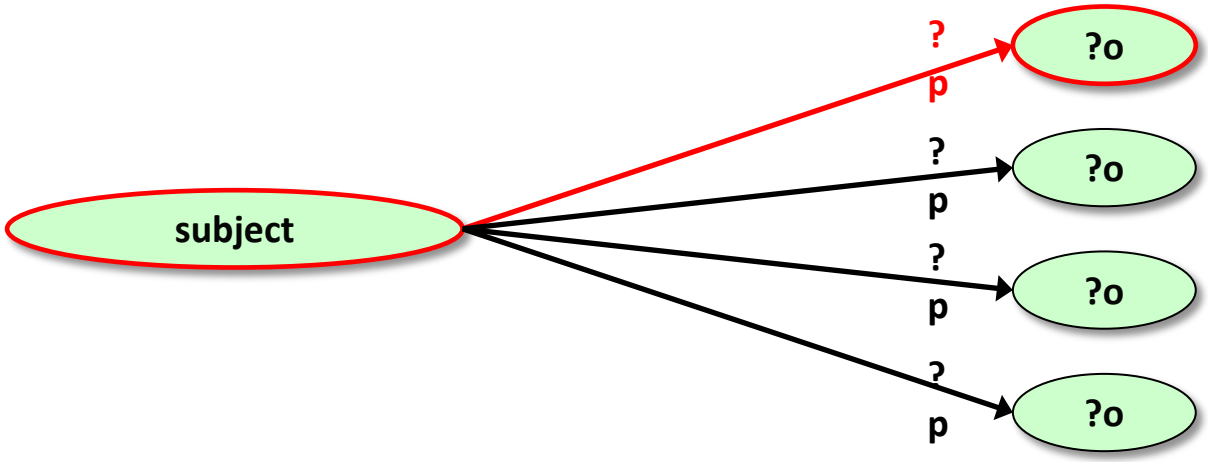
Generalità: graph patterns

- **Idea di base: usare i graph pattern (modelli)**
  - ✓ **il pattern contiene simboli “unbound” (variabili libere)**
  - ✓ **assegnando un valore alle variabili, vengono selezionati dei sottografi del grafo**
  - ✓ **se vengono selezionati dei sottografi, la query restituisce le risorse selezionate**

In SPARQL

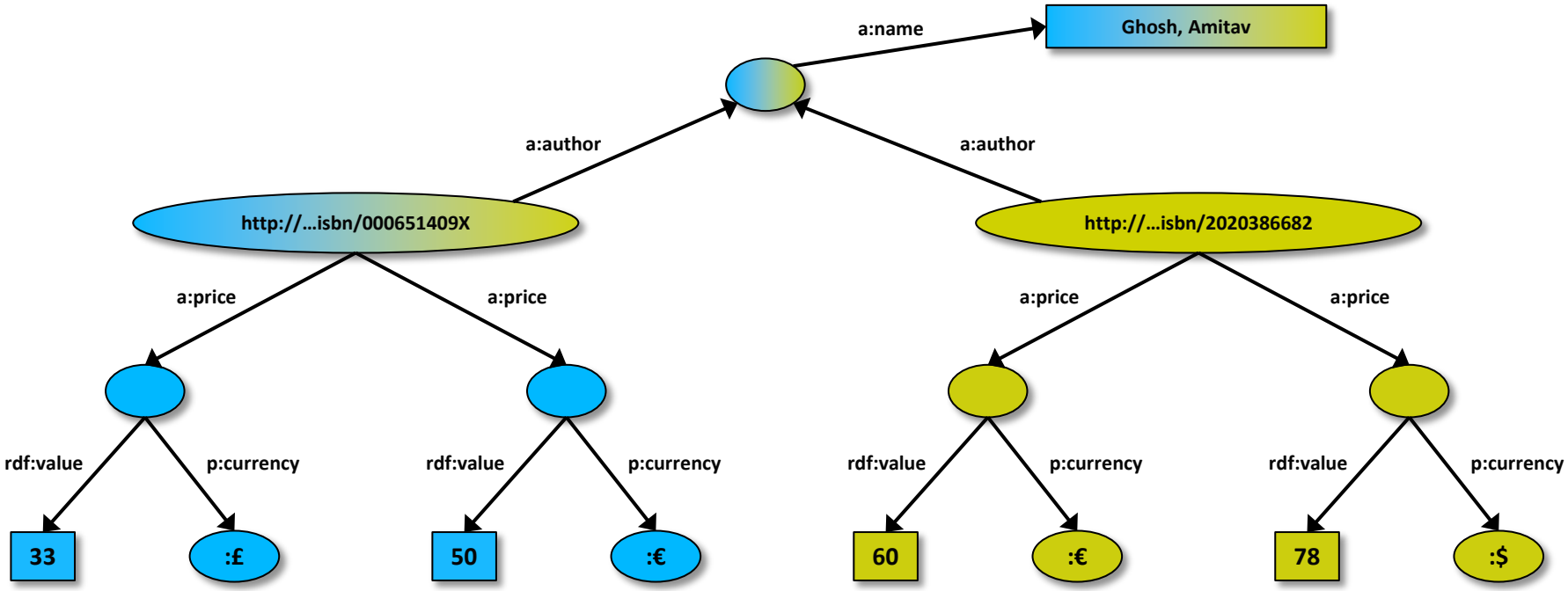
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

- Le triple specificate nella clausola **WHERE** definiscono il graph pattern, con **?p** e **?o** variabili libere (“unbound symbols”)
- La query restituisce **tutte** le coppie **(p,o)**



SPARQL: un esempio semplice (1)

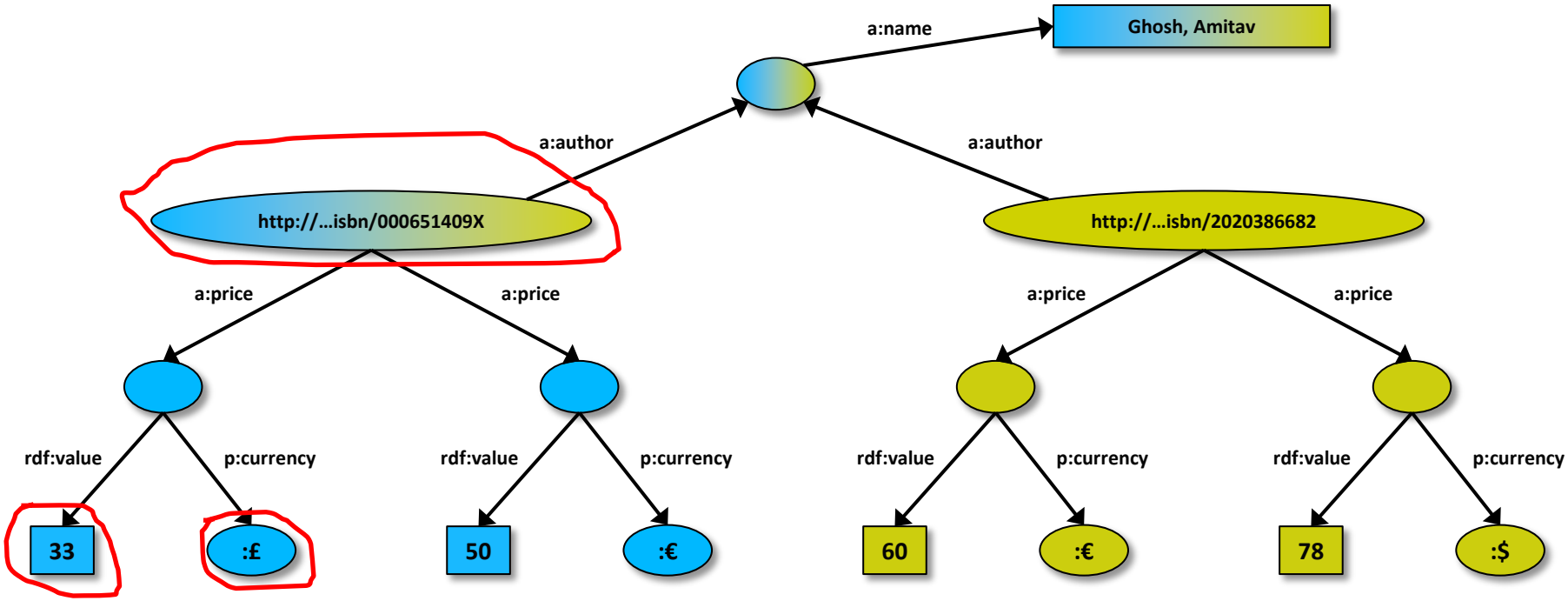
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



SPARQL: un esempio semplice (2)

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

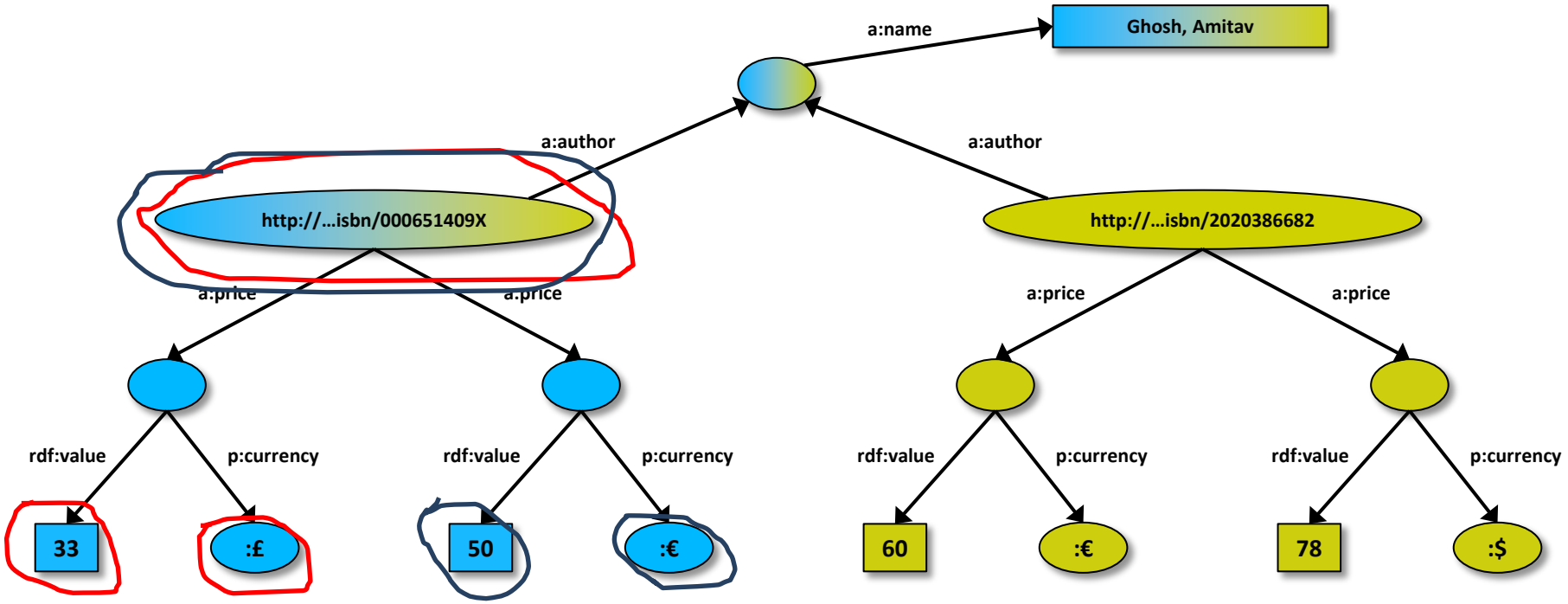
Returns: [<…409X>,33,:£]



SPARQL: un esempio semplice (3)

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

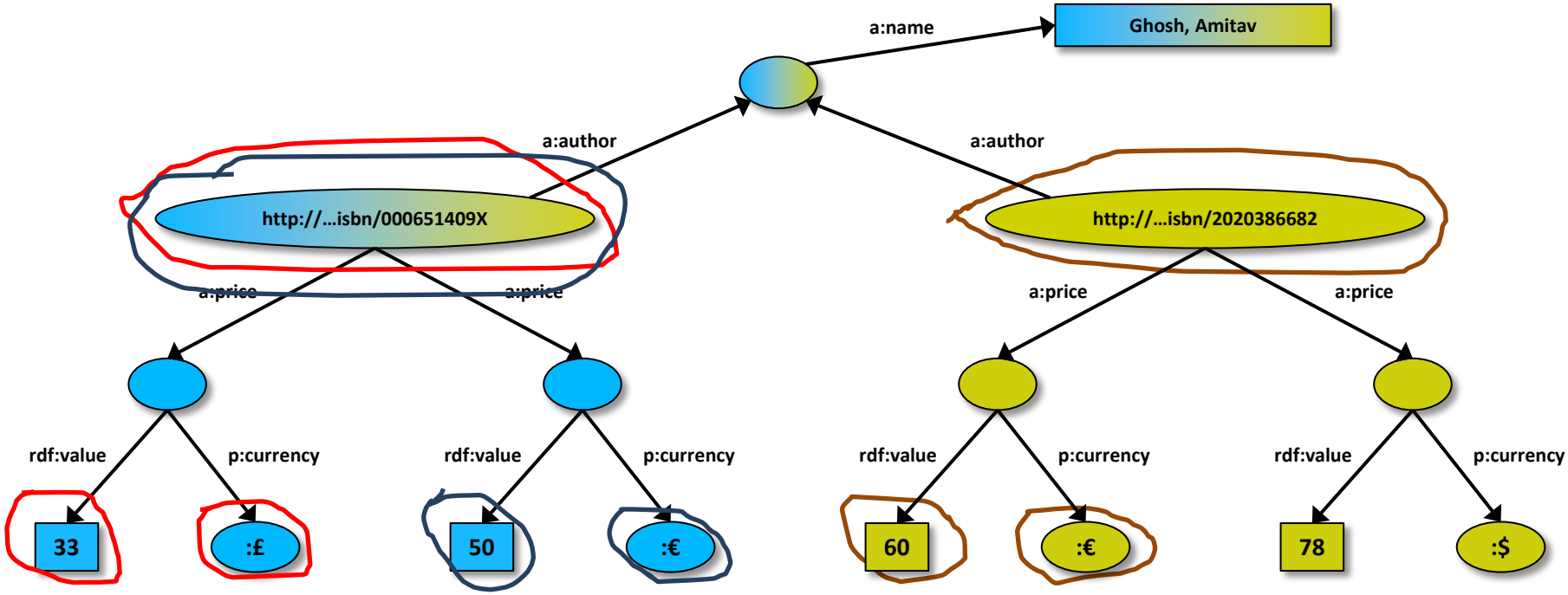
Returns: [<…409X>,33,:£], [<…409X>,50,:€]



SPARQL: un esempio semplice (4)

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

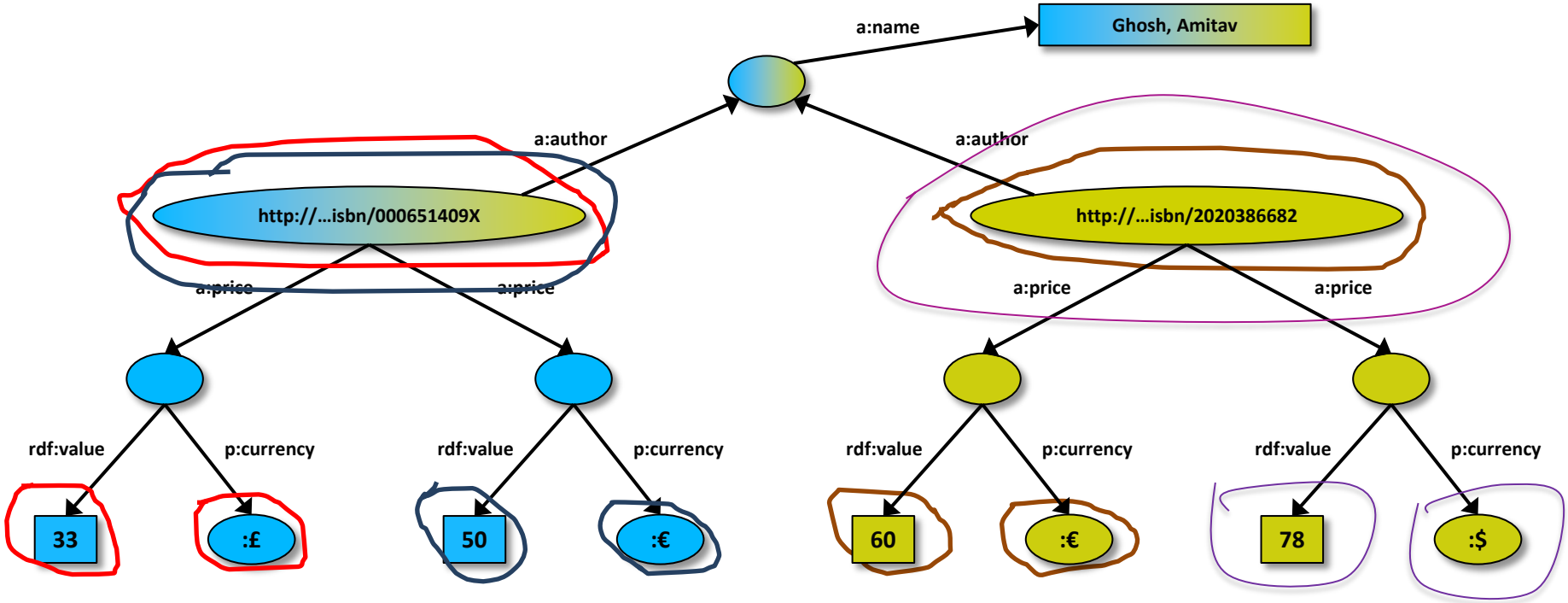
Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€]



SPARQL: un esempio semplice (5)

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

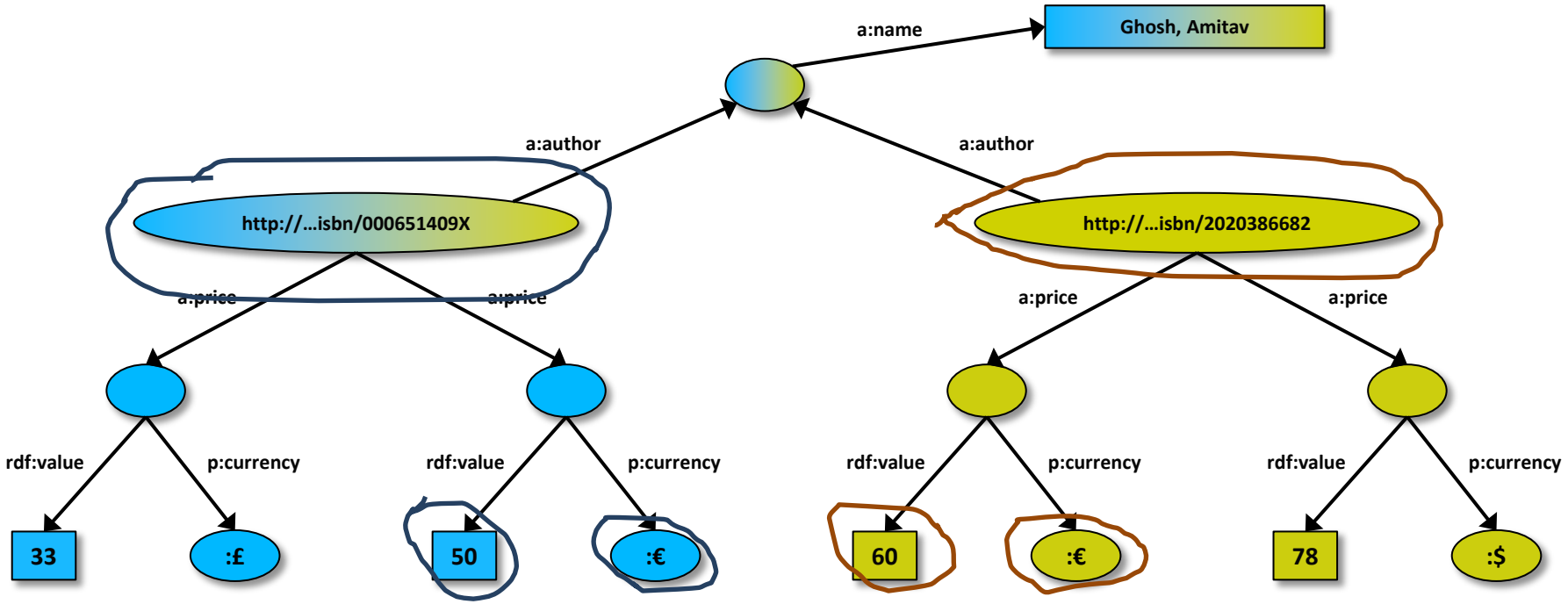
Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€], [<…6682>,78,:$]



Pattern constraints

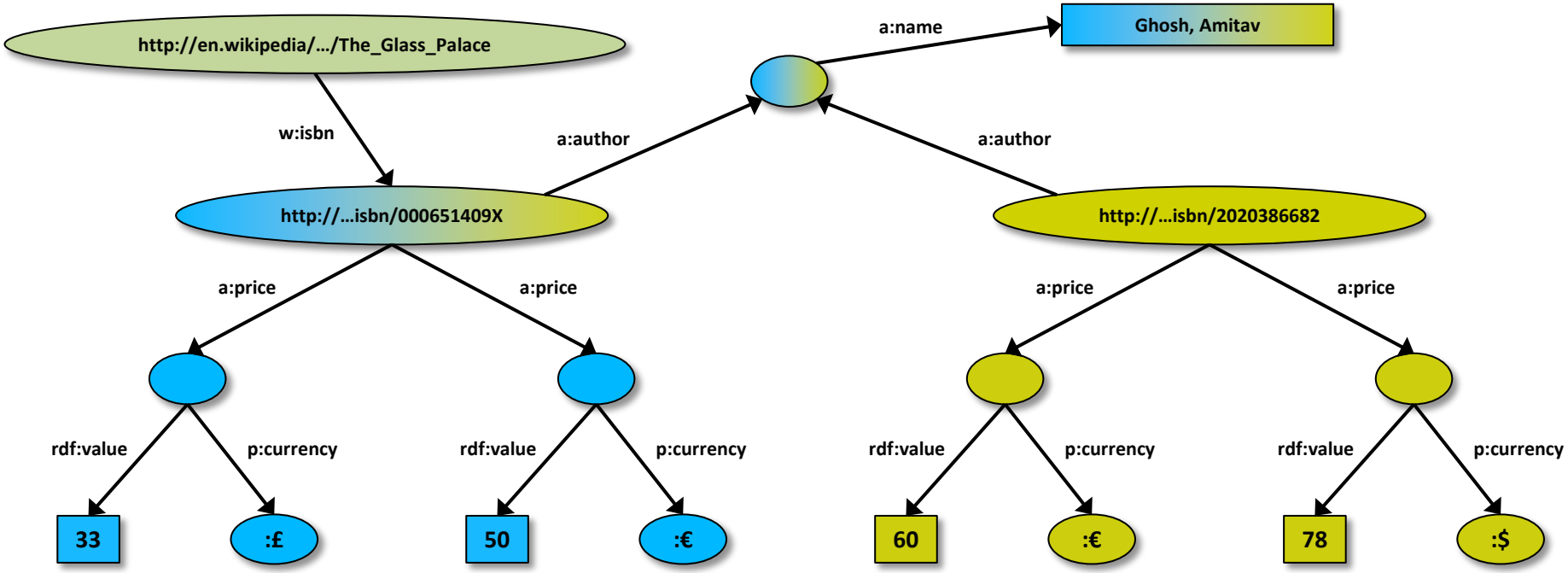
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == :€) }
```

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



OPTIONAL pattern (1)

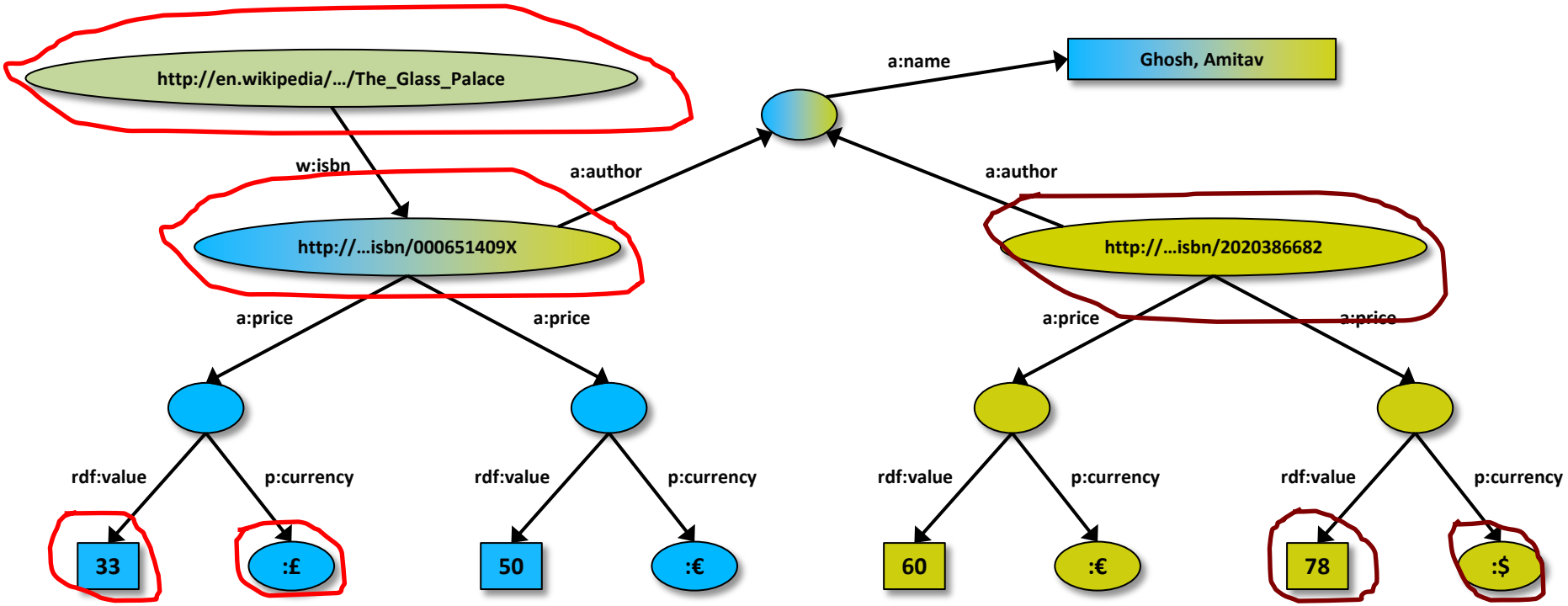
```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```



OPTIONAL pattern (2)

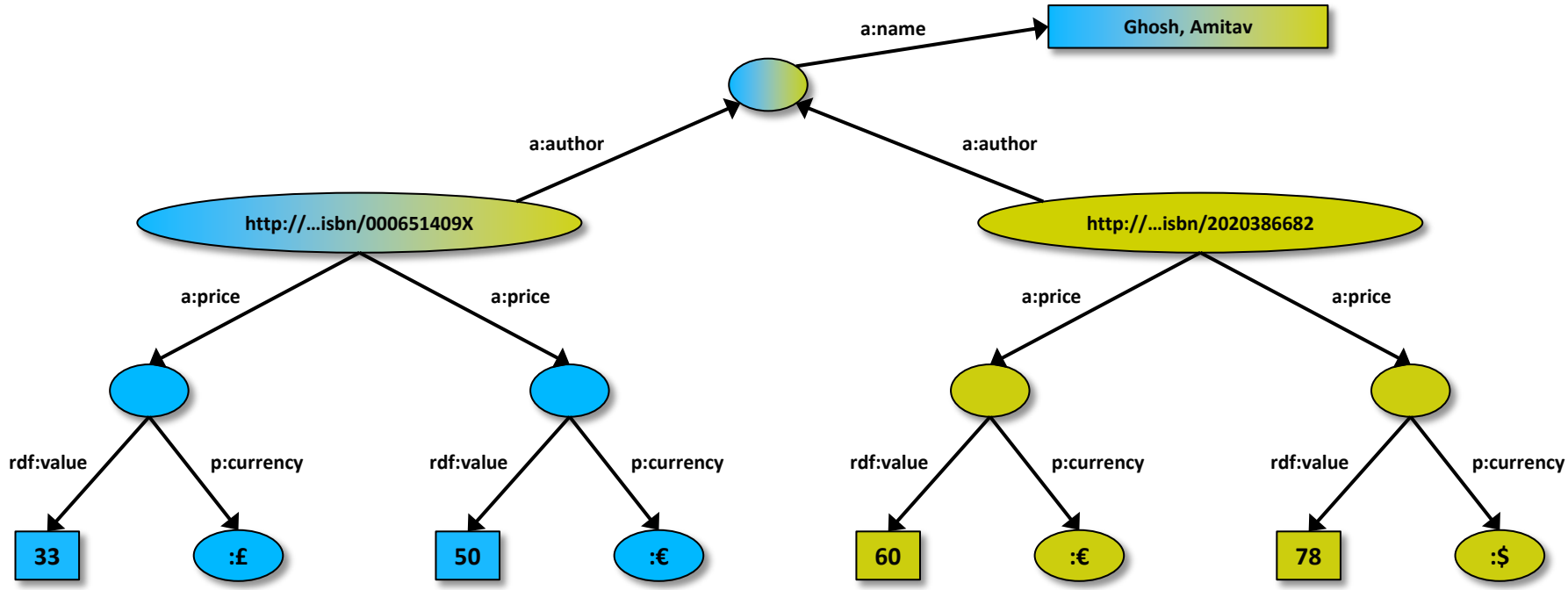
```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

Returns: [[<..09X>,33,:£,<…Palace>], … , [<..6682>,78,:$,]]



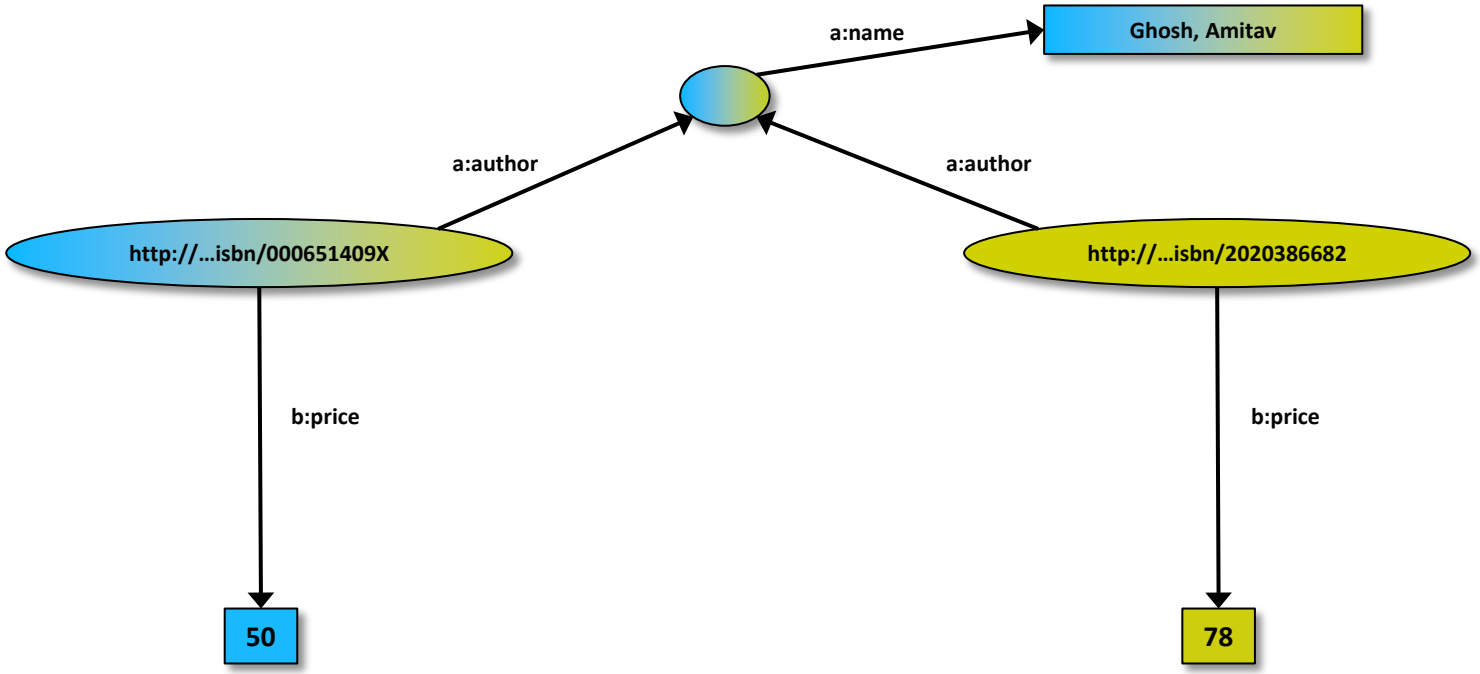
CONSTRUCT: crea nuovo grafo (1)

```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```



CONSTRUCT: crea nuovo grafo (2)

```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```

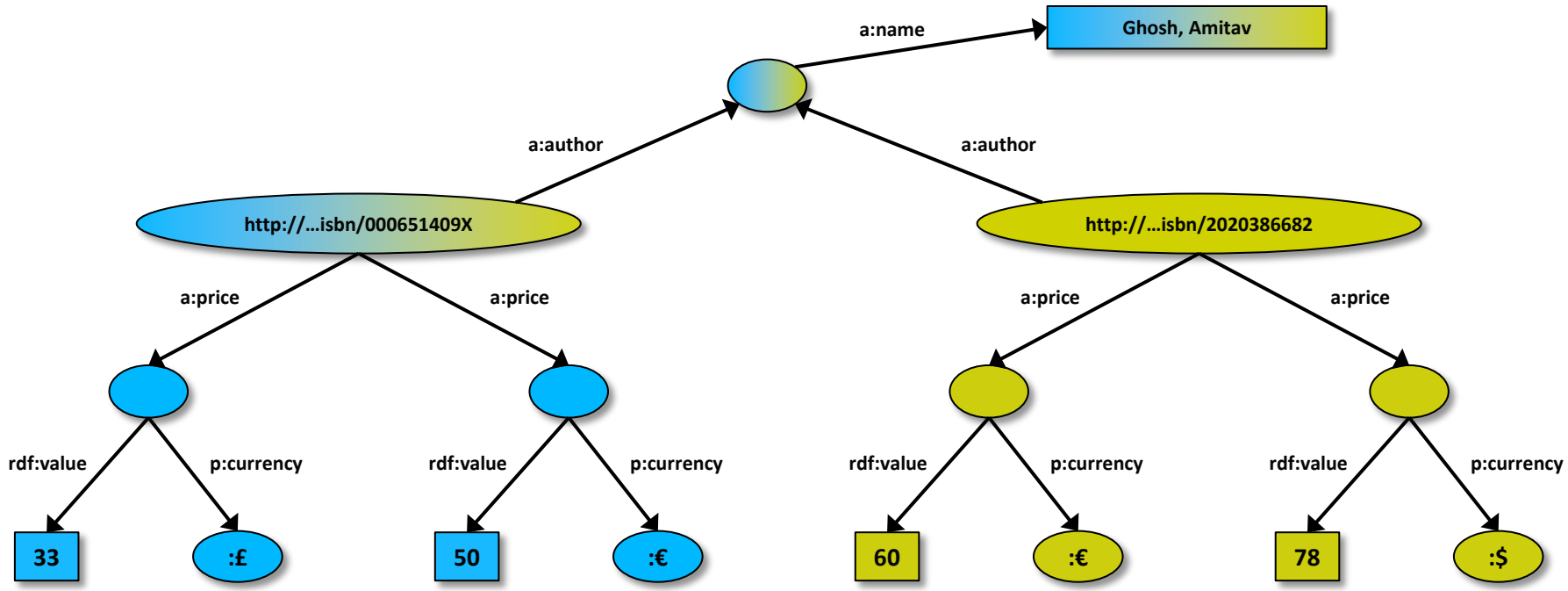


Altre caratteristiche di SPARQL

- Limitare il numero di risultati restituiti, eliminare i duplicati, ordinarli, …
- Specificare nella query varie sorgenti di dati (data source) mediante gli URI
- Costruire un grafo combinando vari pattern e risultati di query
- Usare datatype e/o tag di lingua nel processo di pattern matching
- Aggregazione dei risultati (min, max, average, etc.)
- Path expressions (qualcosa di simile alle espressioni regolari)

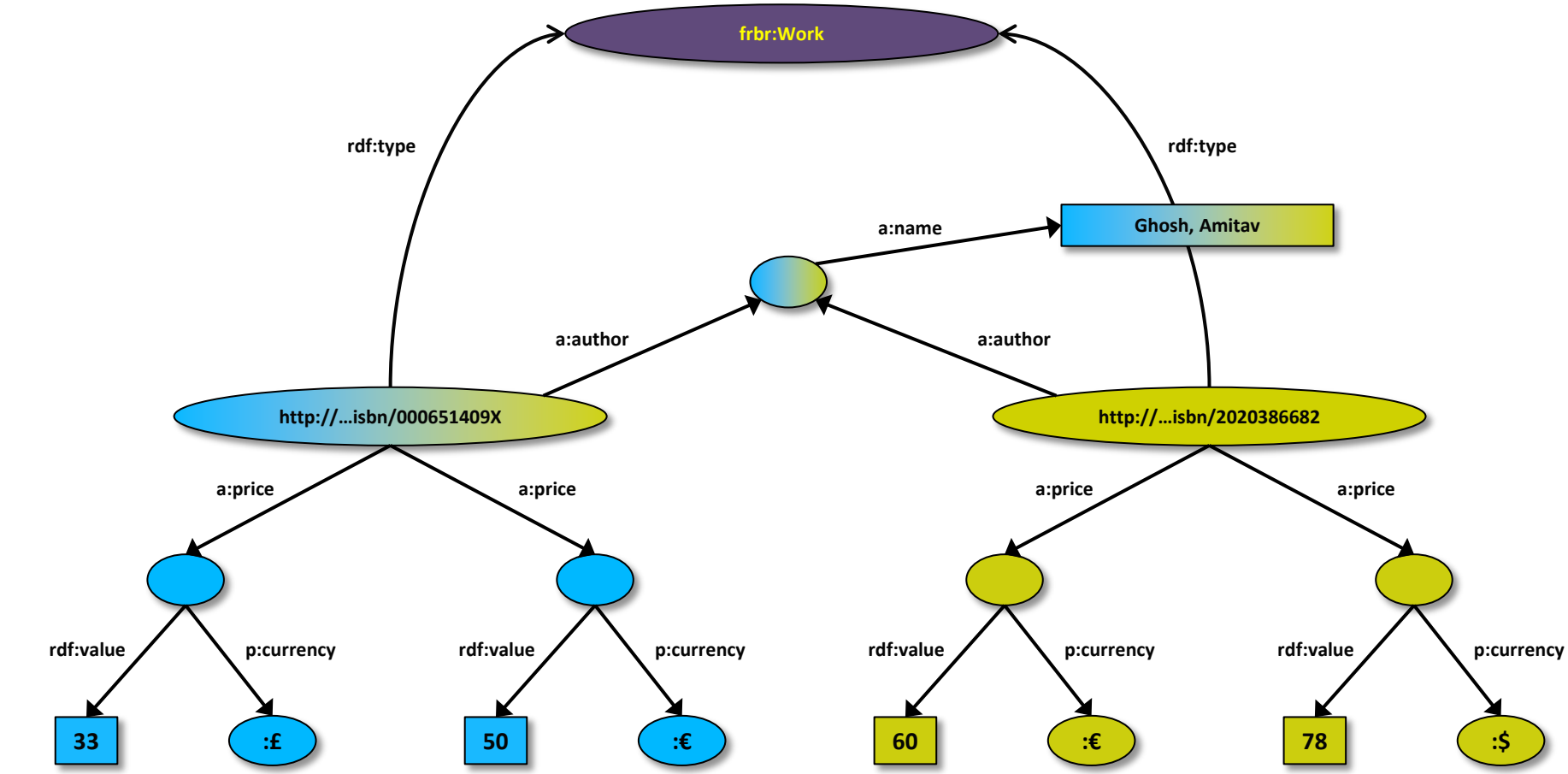
Update: insert (prima)

```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



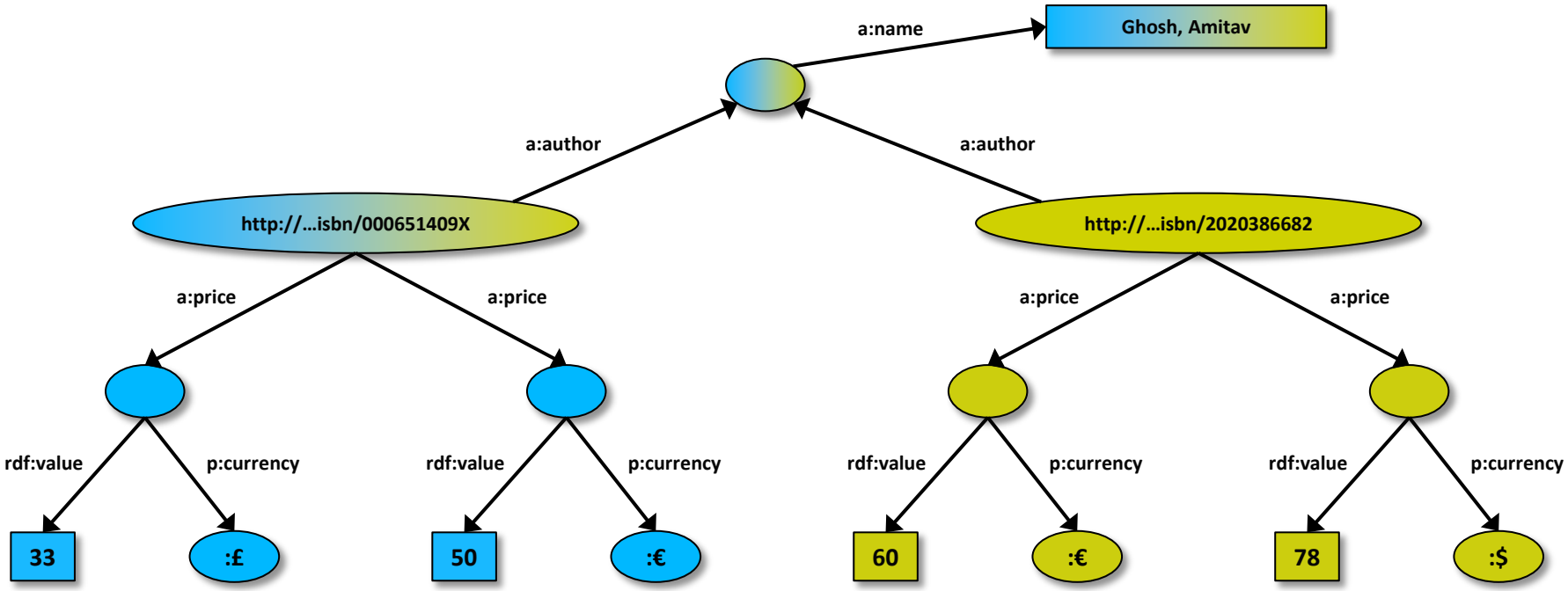
Update: insert (dopo)

```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



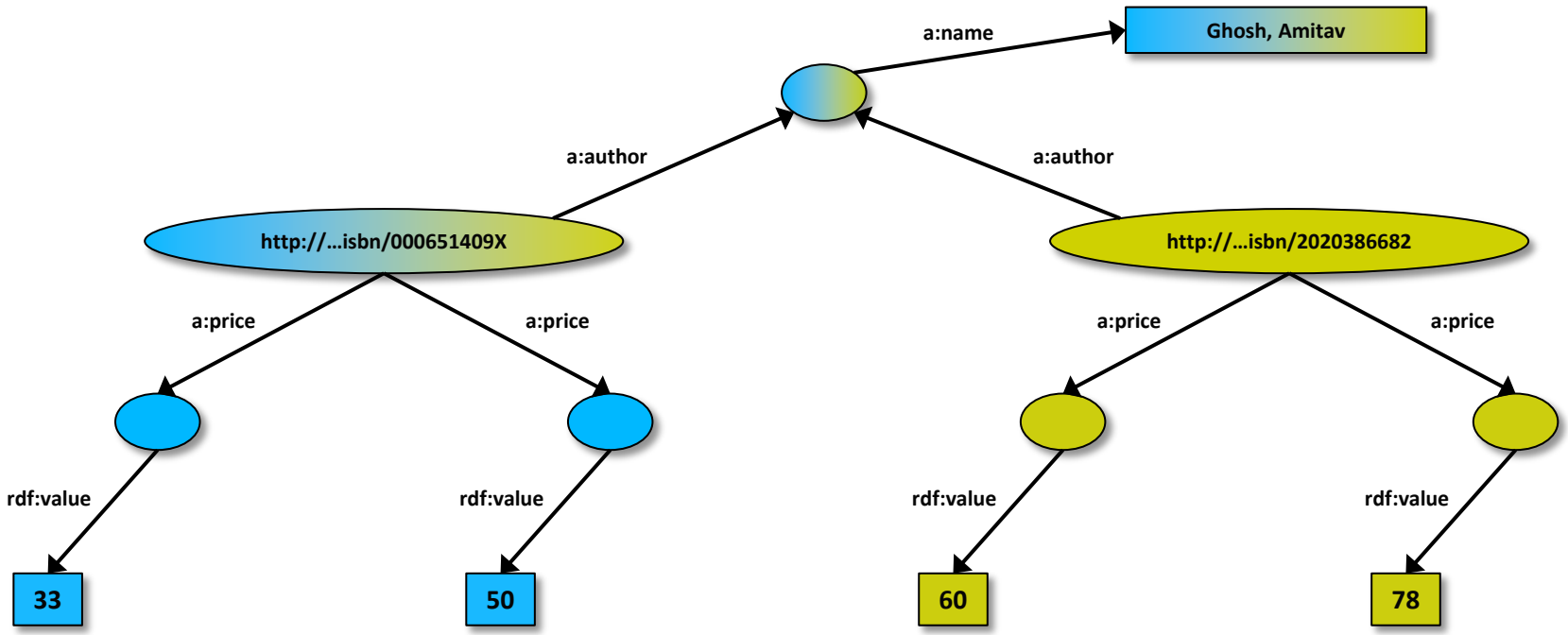
Update: delete (prima)

```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Update: delete (dopo)

```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Conclusioni

- I database NoSQL nascono per far fronte ad alcune nuove esigenze
- SPARQL è il linguaggio di interrogazione per i grafi RDF

Grazie per l'attenzione!

(Nobody's perfect!)

?

Domande

Slide a: http://www.orestesignore.eu/education/lda/slides/sparql.pdf