Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution

Olaf Hartig
http://olafhartig.de/foaf.rdf#olaf
@olafhartig

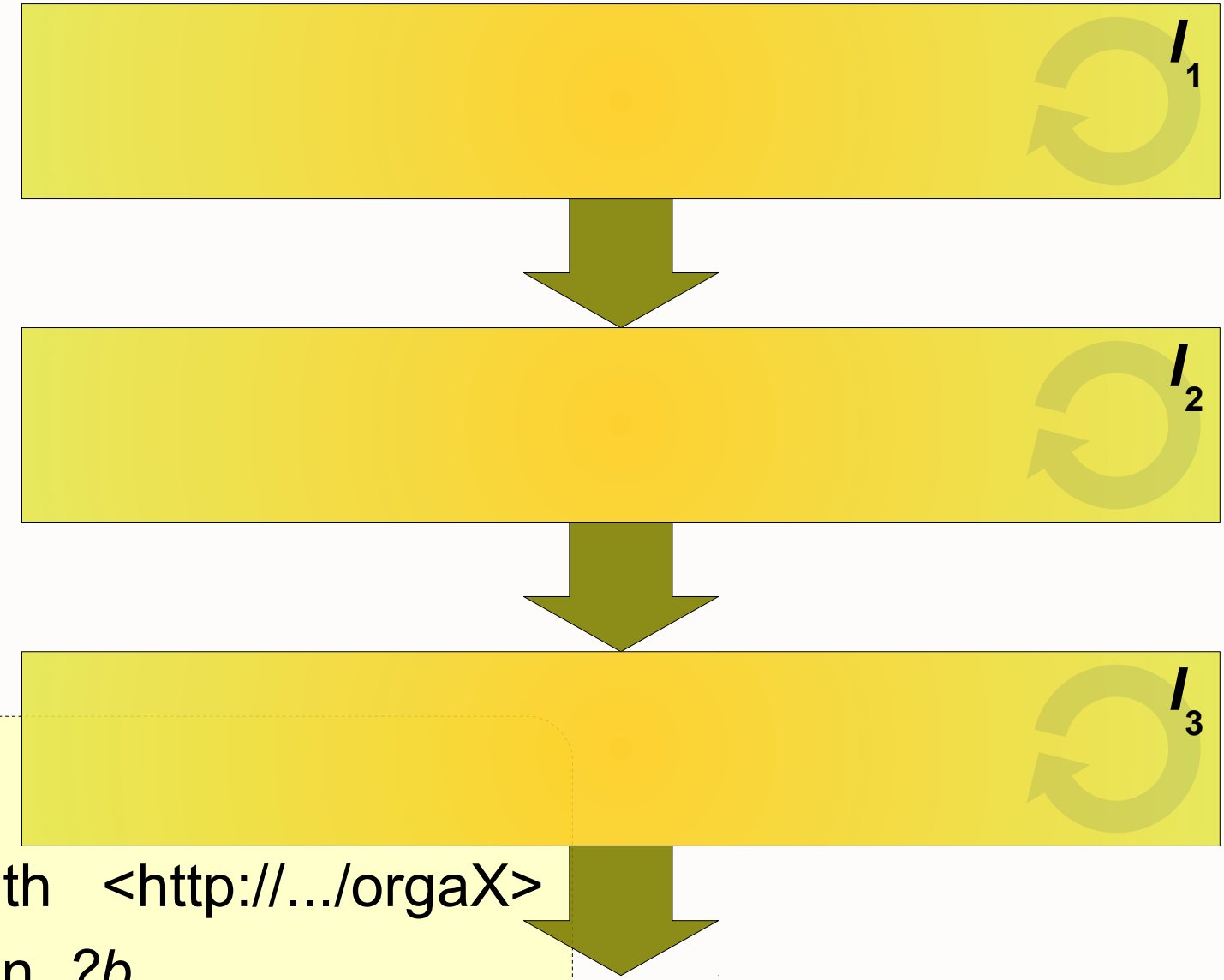
Database and Information Systems Research Group
Humboldt-Universität zu Berlin

Example Query

Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

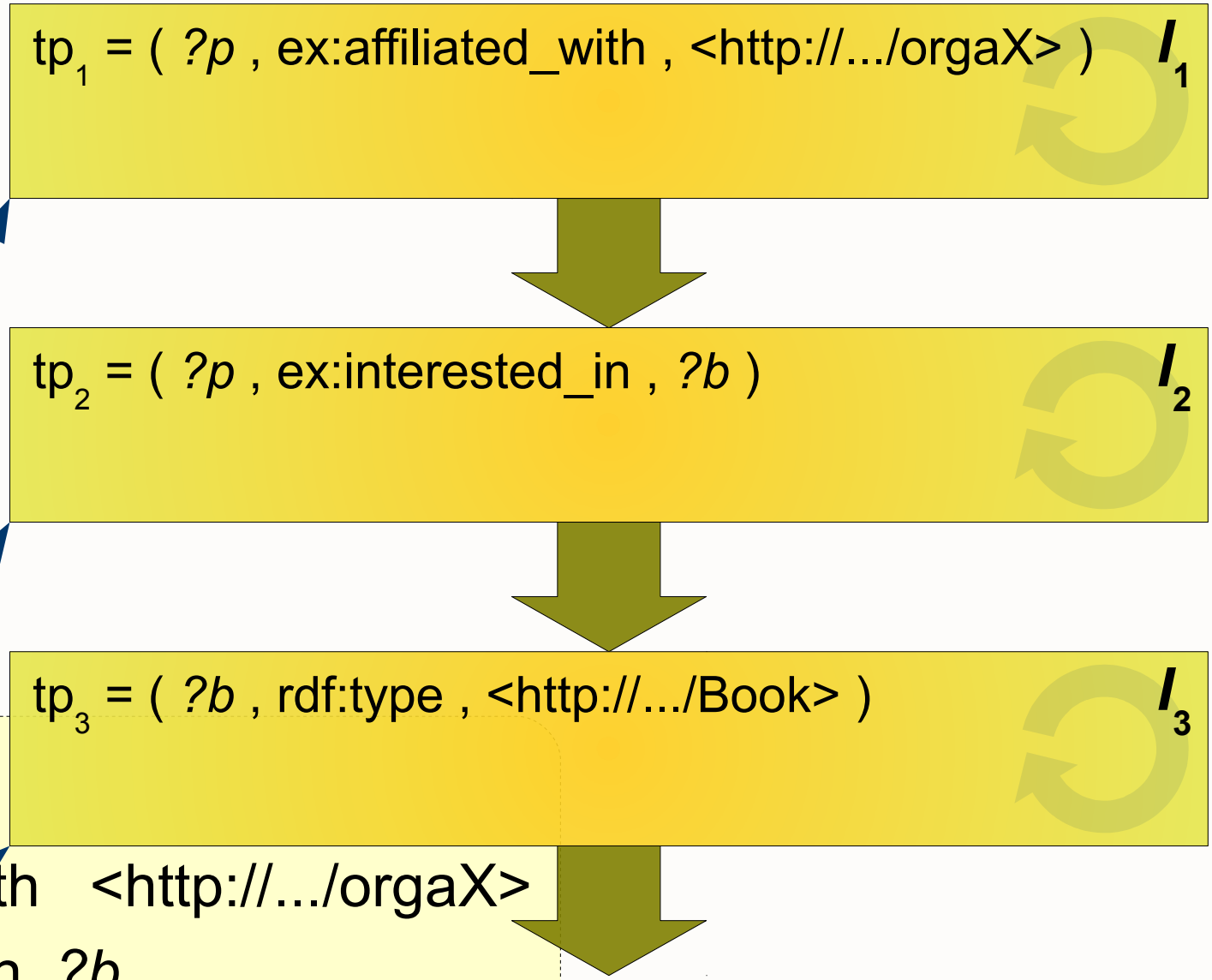
Iterator Based Execution Plan



Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

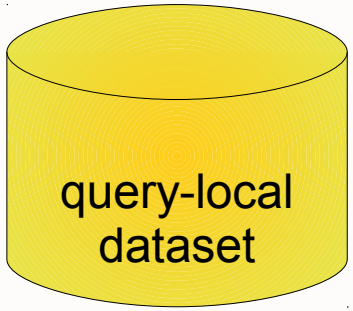
Iterator Based Execution Plan



Query

```
?p  ex:affiliated_with  <http://.../orgaX>
?p  ex:interested_in  ?b
?b  rdf:type  <http://.../Book>
```

Iterator Based Execution Plan

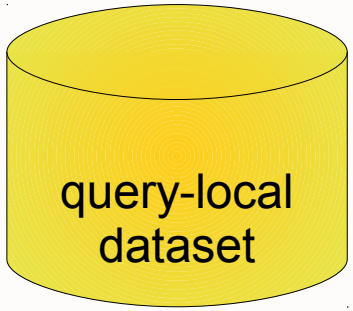


tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Iterator Based Execution Plan



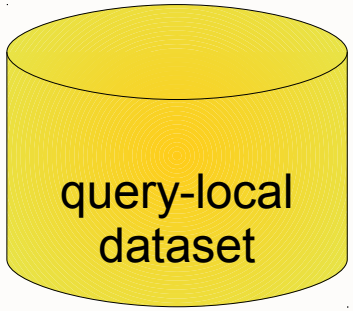
tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

Iterator Based Execution Plan



tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

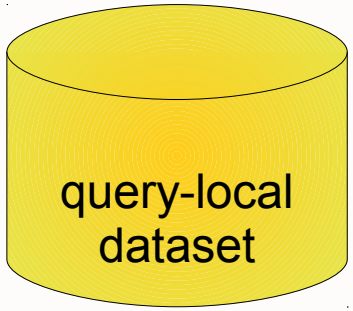
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

Next?

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

Iterator Based Execution Plan



tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

Next?

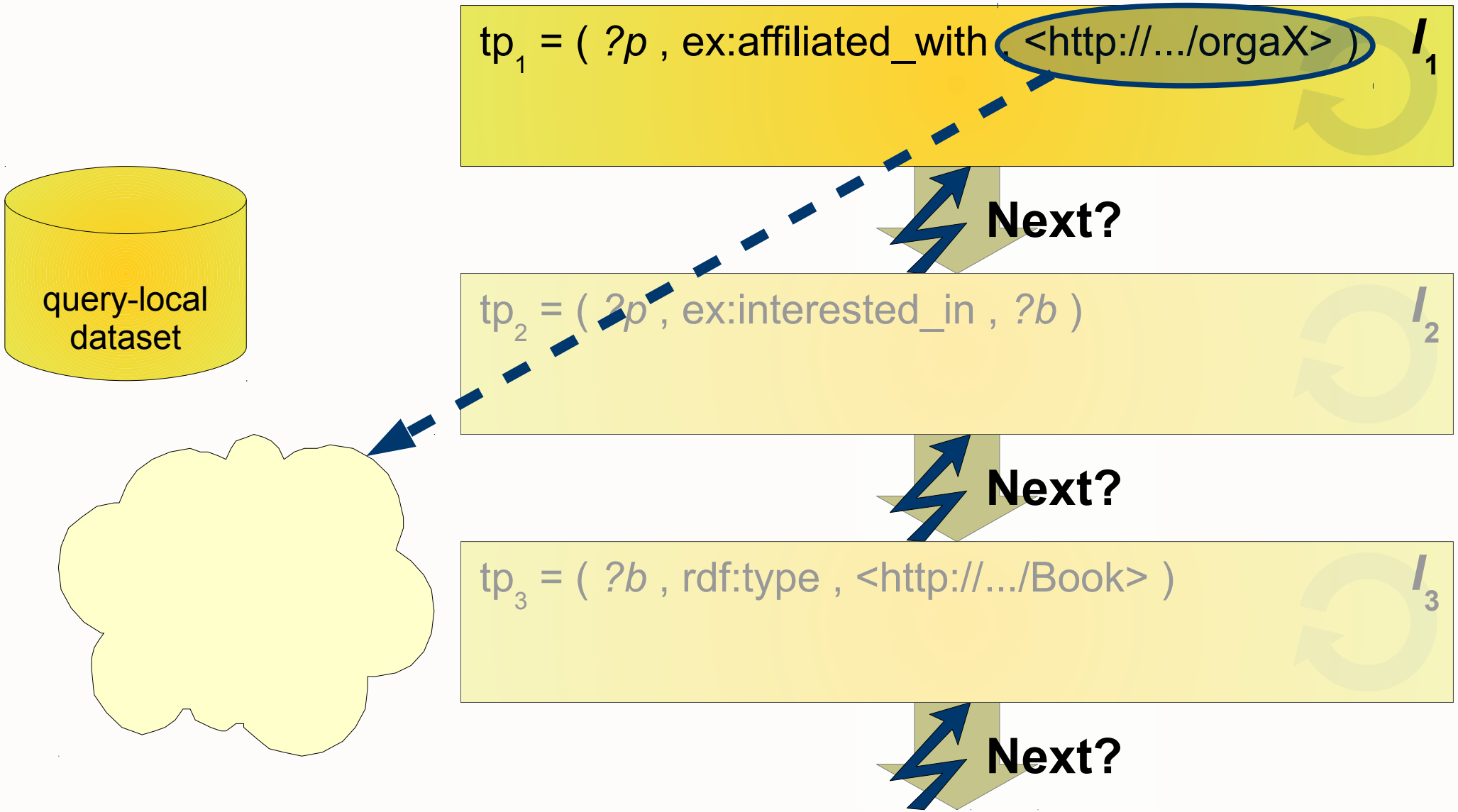
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

Next?

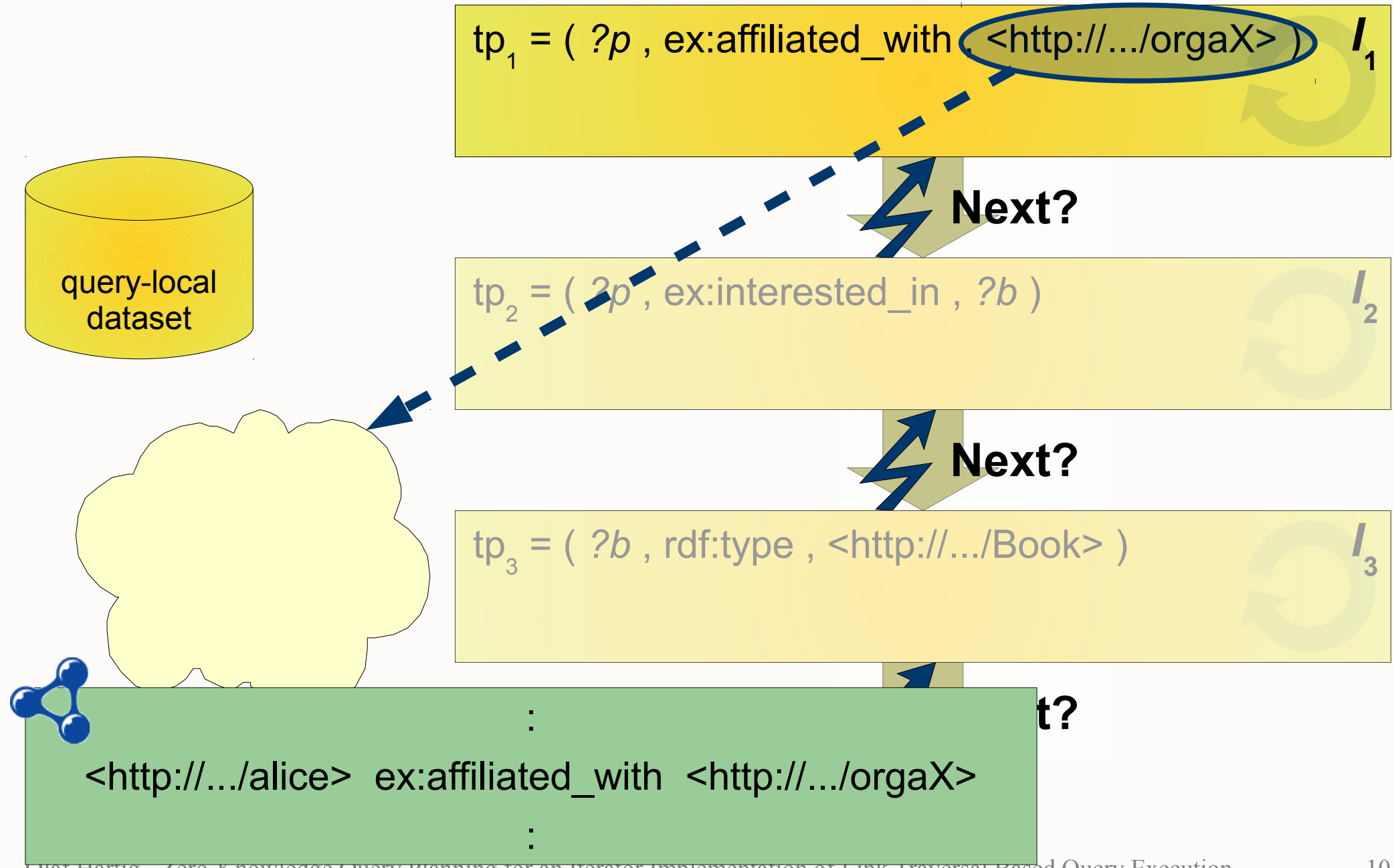
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

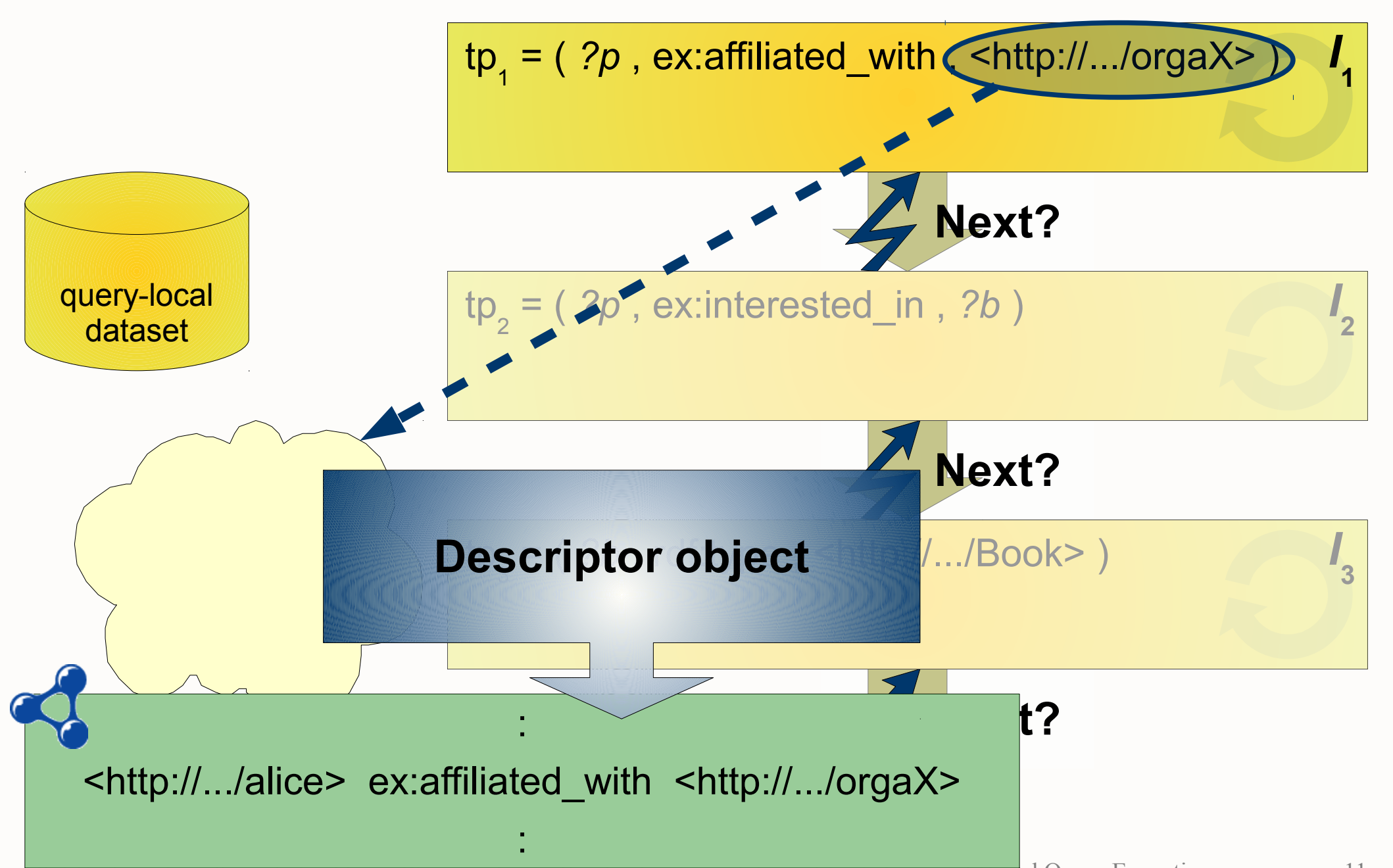
Iterator Based Execution Plan



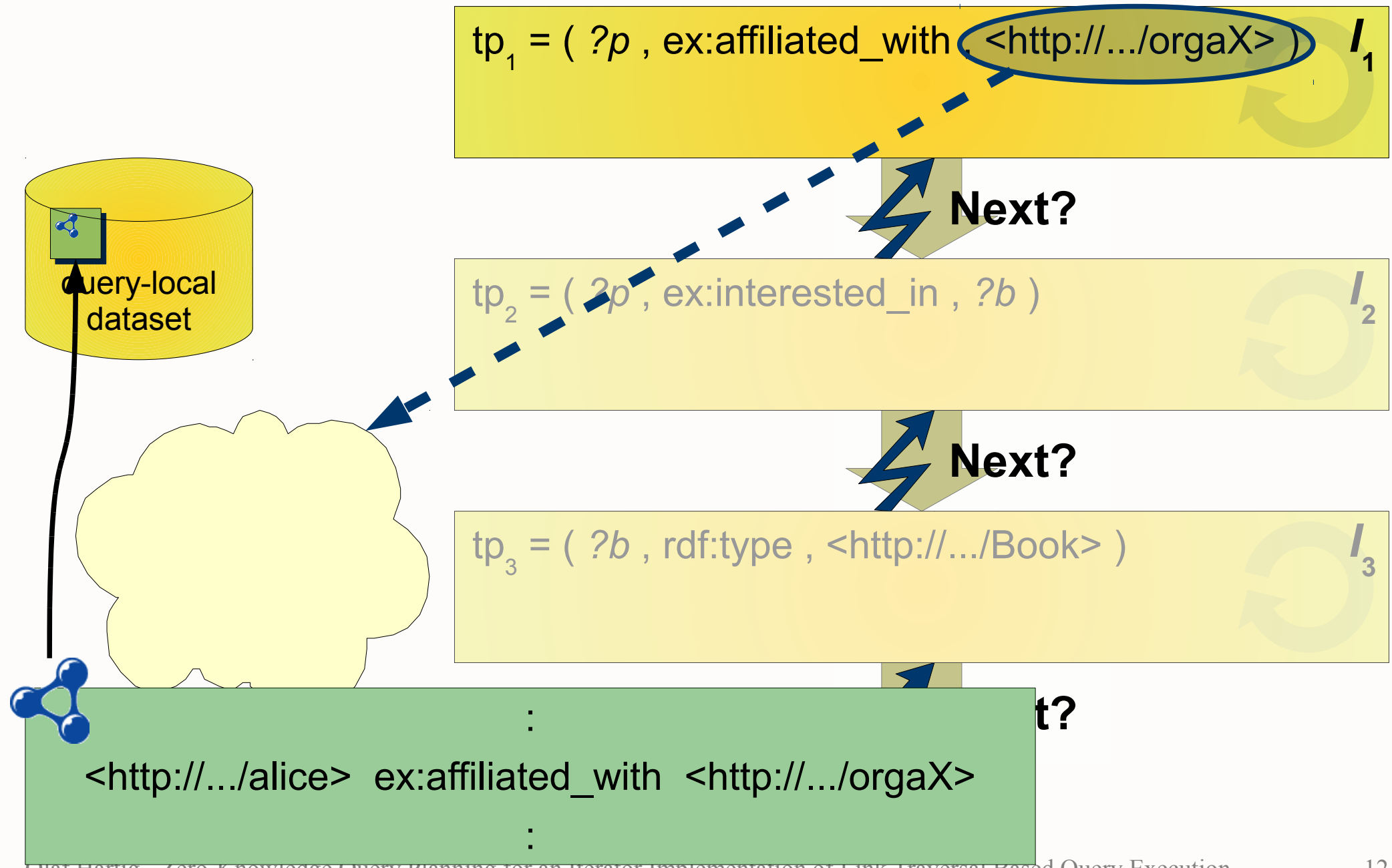
Iterator Based Execution Plan



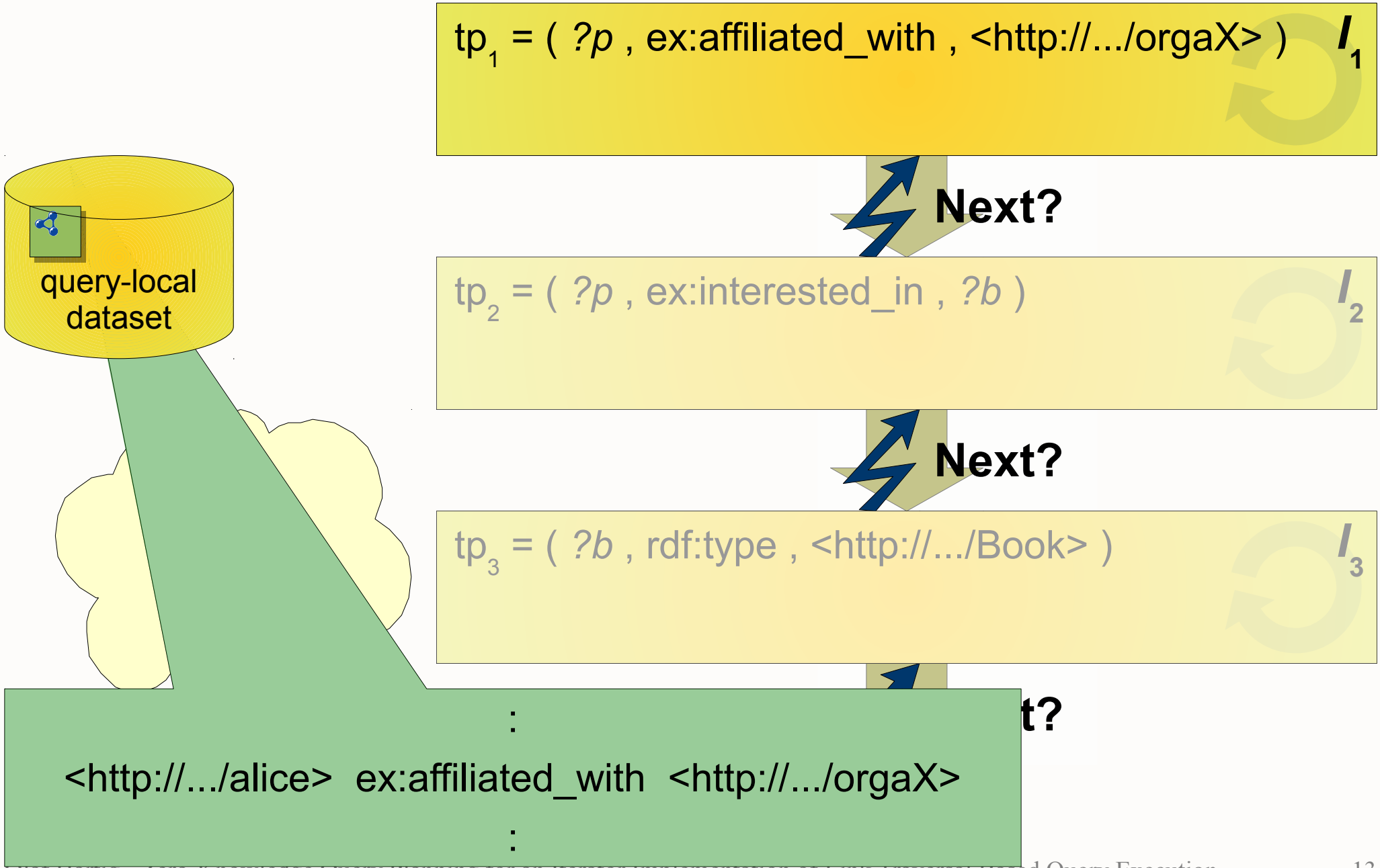
Iterator Based Execution Plan



Iterator Based Execution Plan

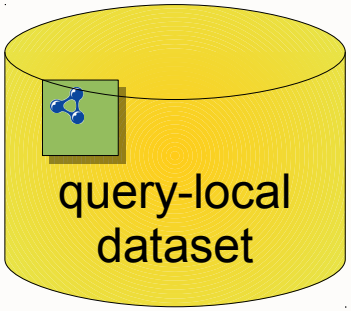


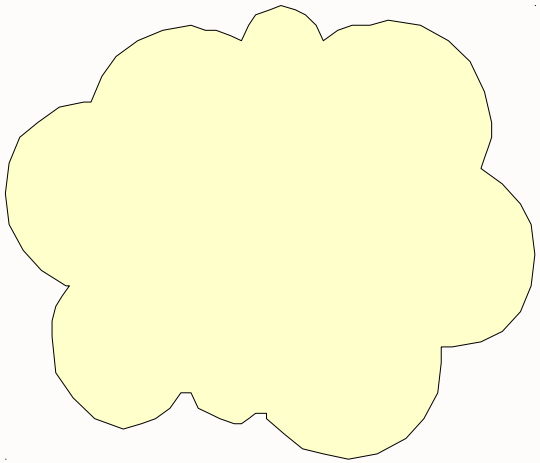
Iterator Based Execution Plan



Iterator Based Execution Plan

Iterator Based Execution Plan





tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

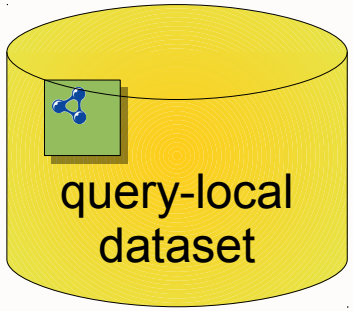
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

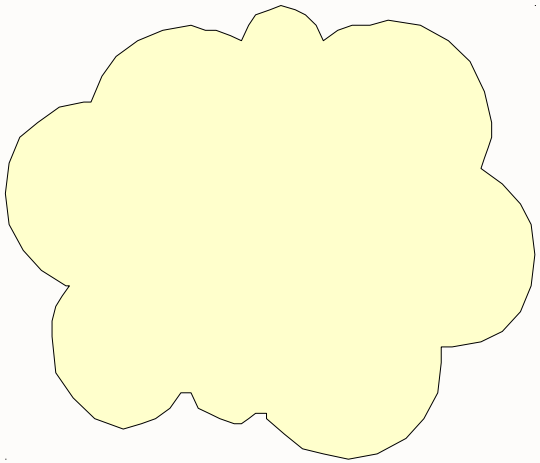
Next?

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

Iterator Based Execution Plan





tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

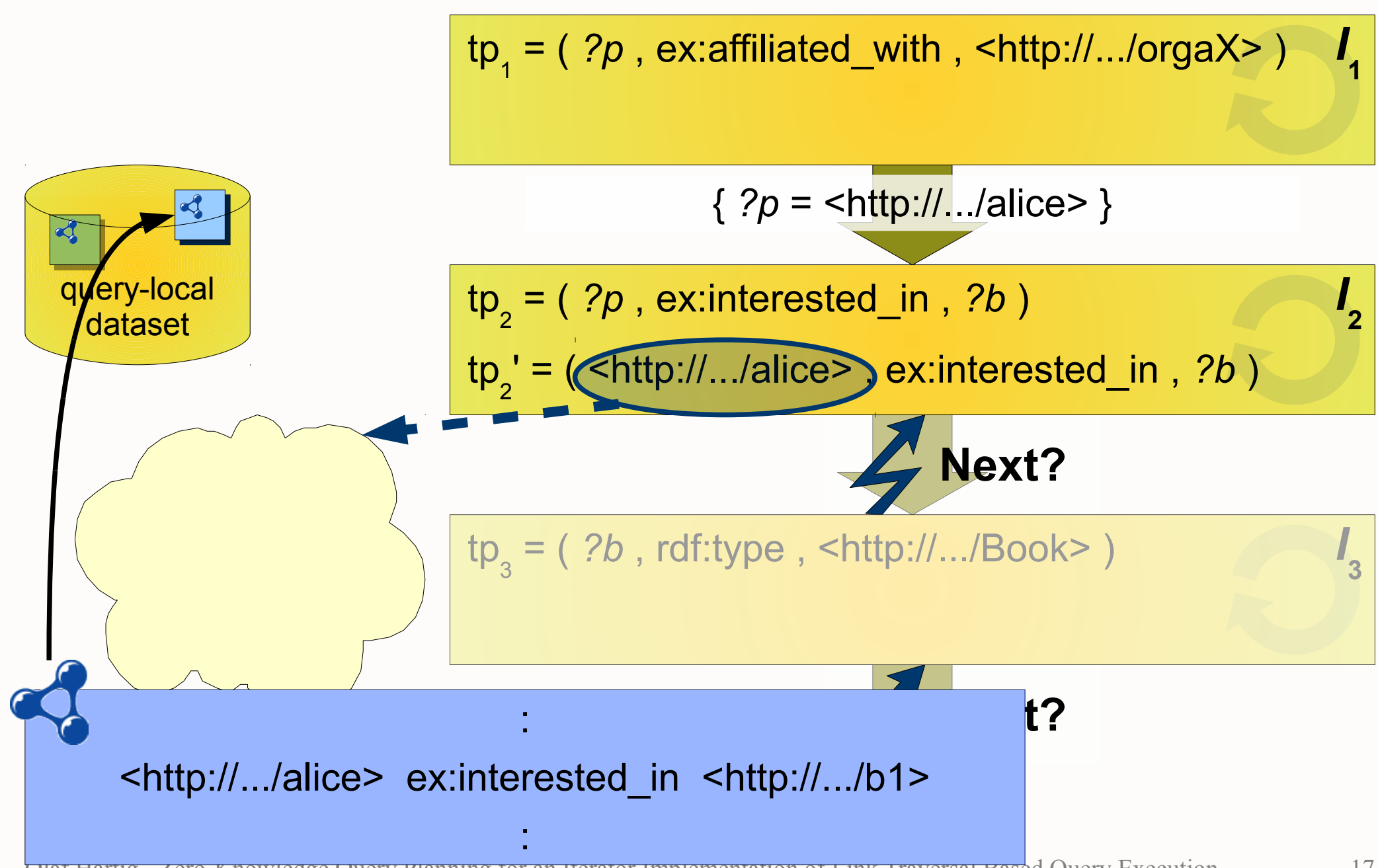
tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

Next?

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

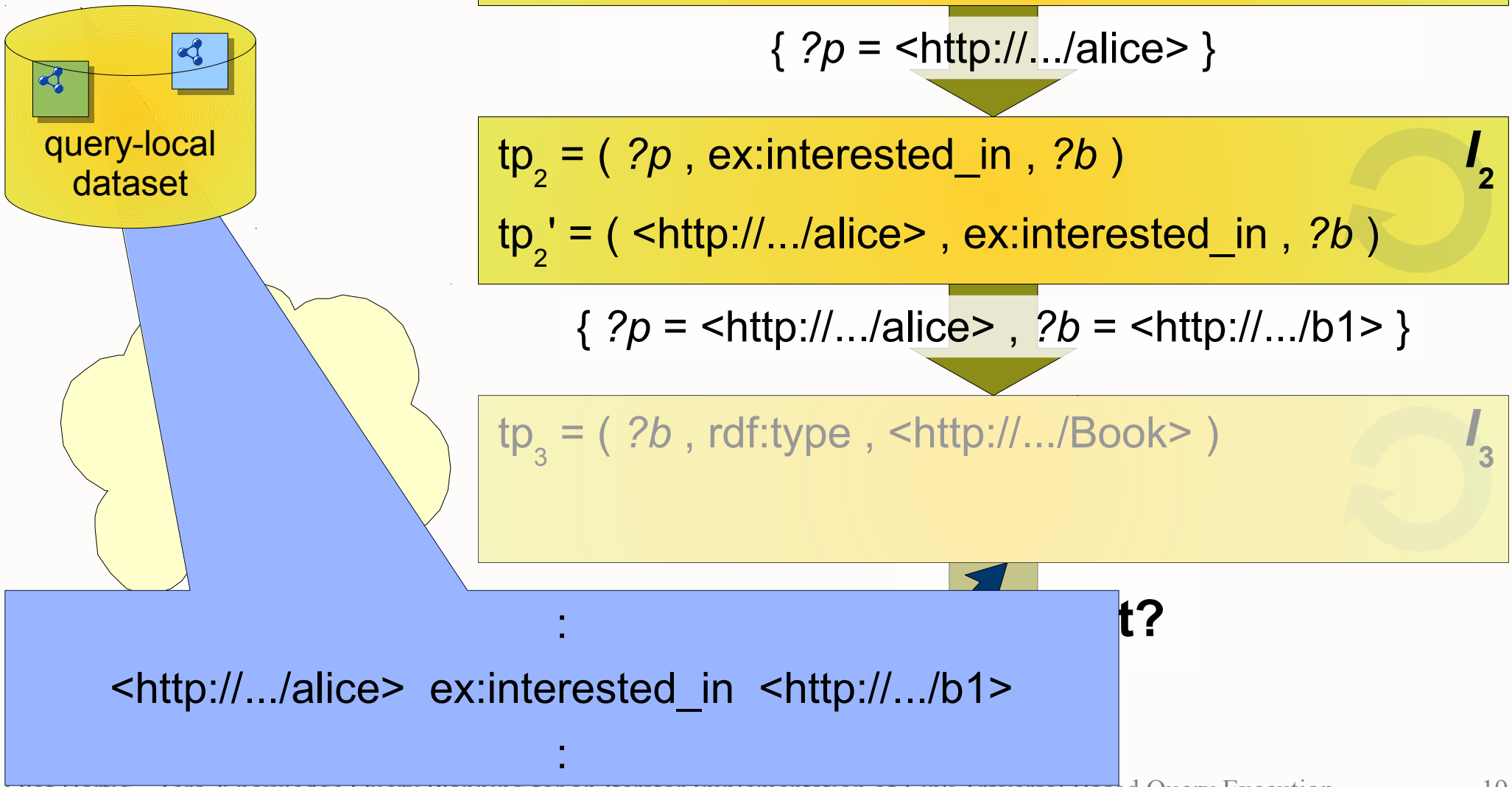
Next?

Iterator Based Execution Plan

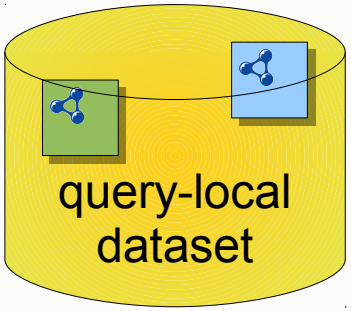


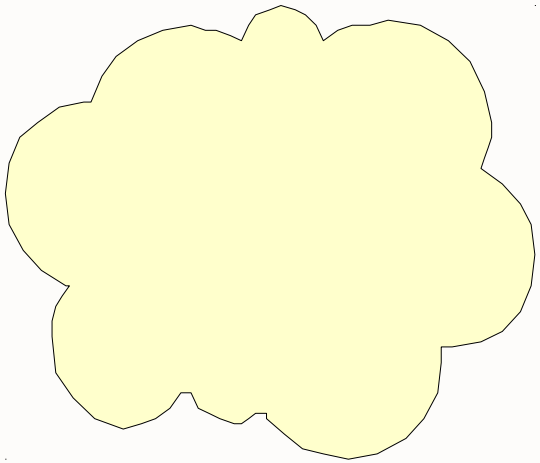
Iterator Based Execution Plan

Iterator Based Execution Plan



Iterator Based Execution Plan





tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

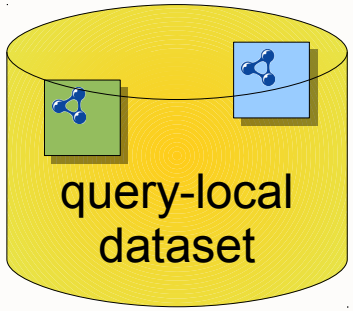
tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

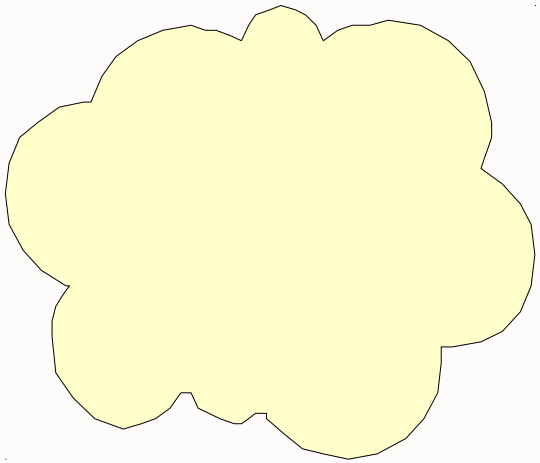
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

Iterator Based Execution Plan





tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) **I_1**

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) **I_2**

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

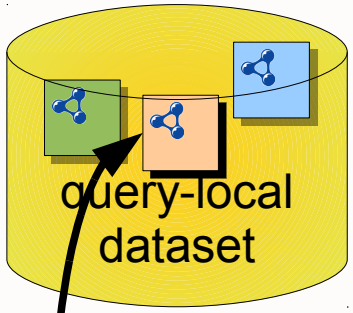
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) **I_3**

tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

Next?

Iterator Based Execution Plan



tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

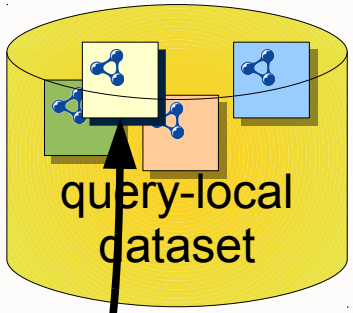
tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

:

<http://.../Book> rdfs:subClassOf <http://.../CreativeWork>

:

Iterator Based Execution Plan



tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

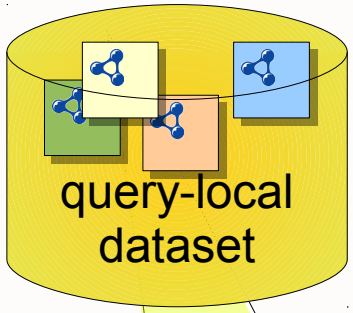
Next?

:

<http://.../b1> rdf:type <http://.../Book>

:

Iterator Based Execution Plan



tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

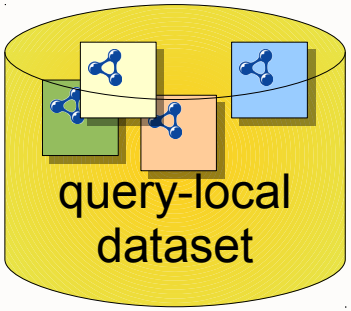
Next?

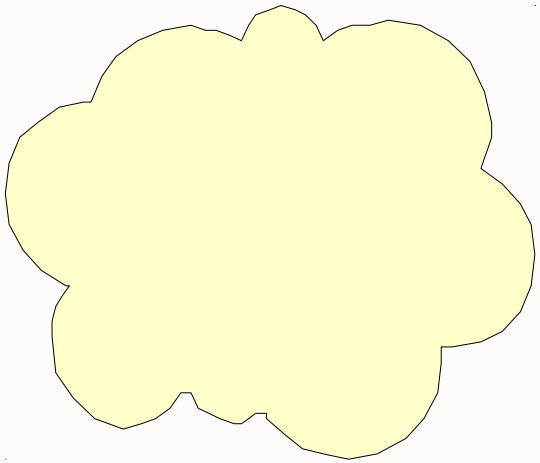
:

<http://.../b1> rdf:type <http://.../Book>

:

Iterator Based Execution Plan





tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) **I_1**

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) **I_2**

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

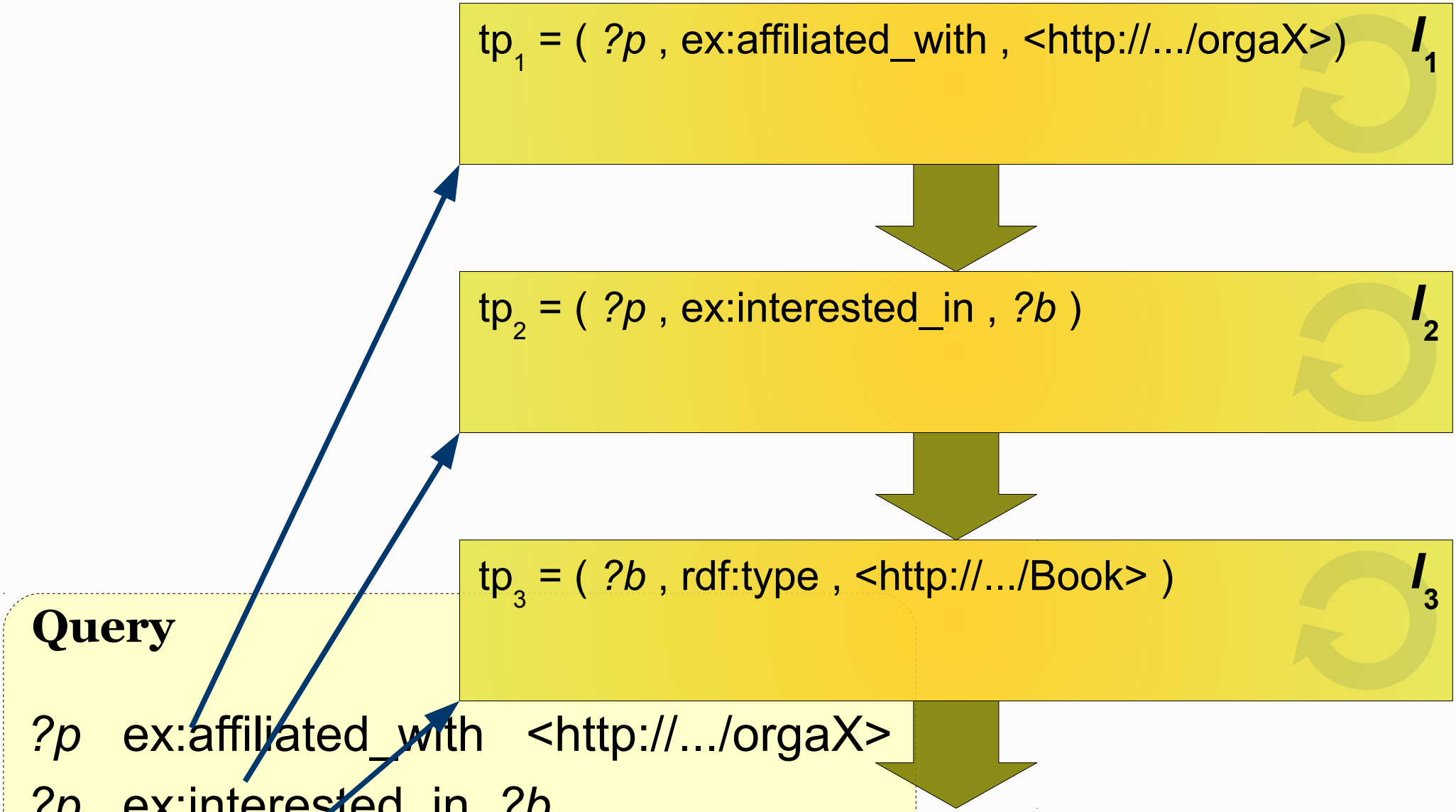
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

tp_3 = (*?b* , rdf:type , <http://.../Book>) **I_3**

tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

Example Query Execution Plan

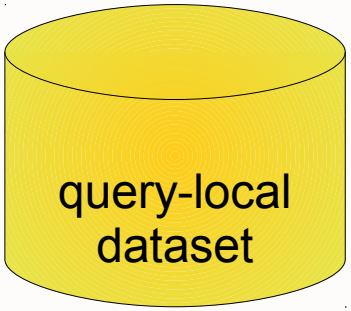


An Alternative Execution Plan

Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

An Alternative Execution Plan

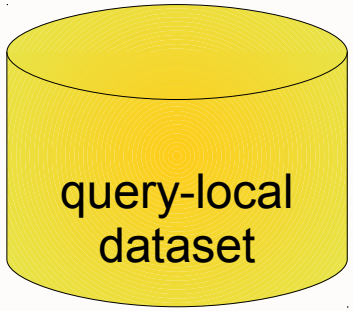


tp_1 = (*?b* , rdf:type , <http://.../Book>) I_1

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_3 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_3

An Alternative Execution Plan



tp_1 = (*?b* , rdf:type , <http://.../Book>) I_1

Next?

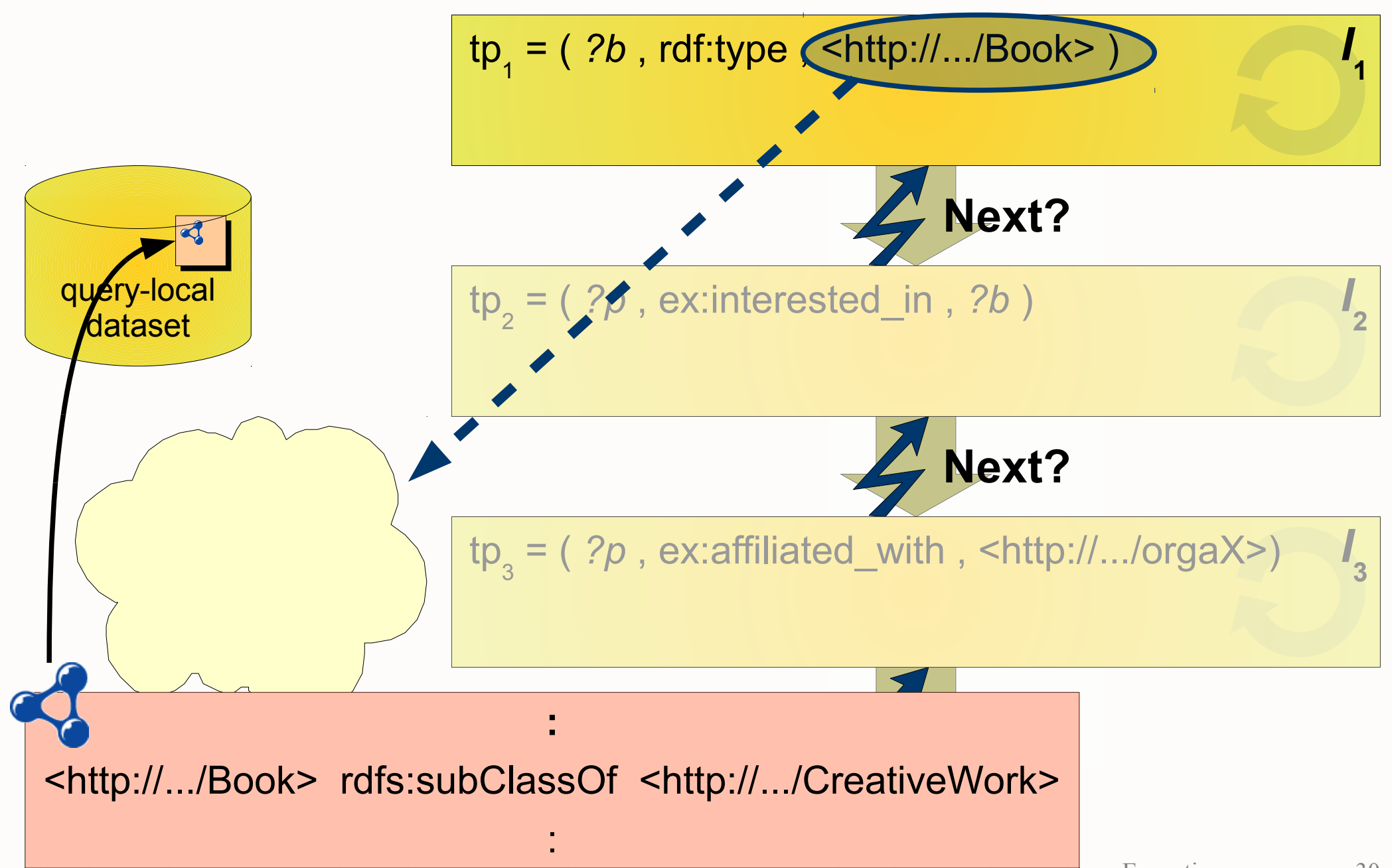
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

Next?

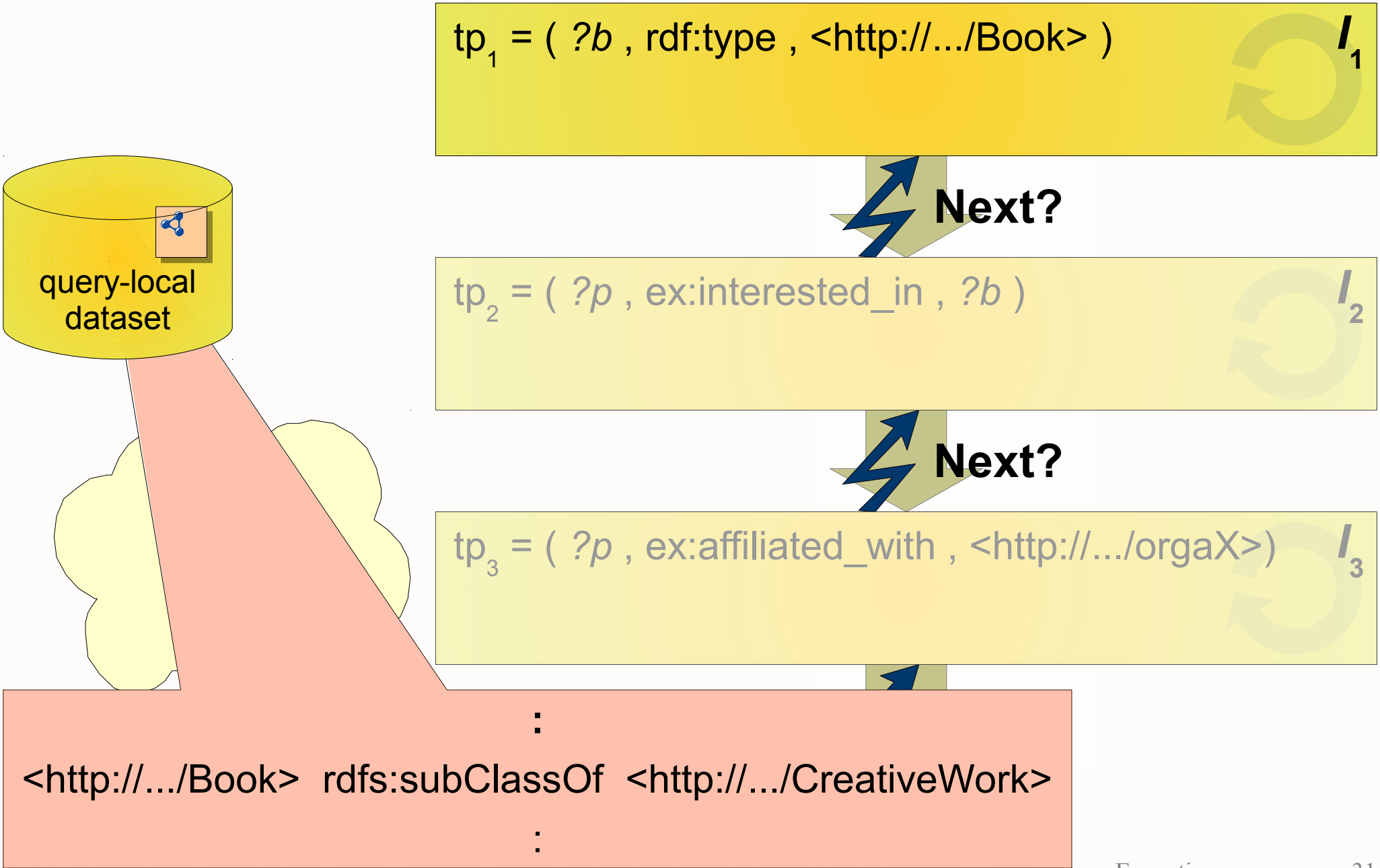
tp_3 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_3

Next?

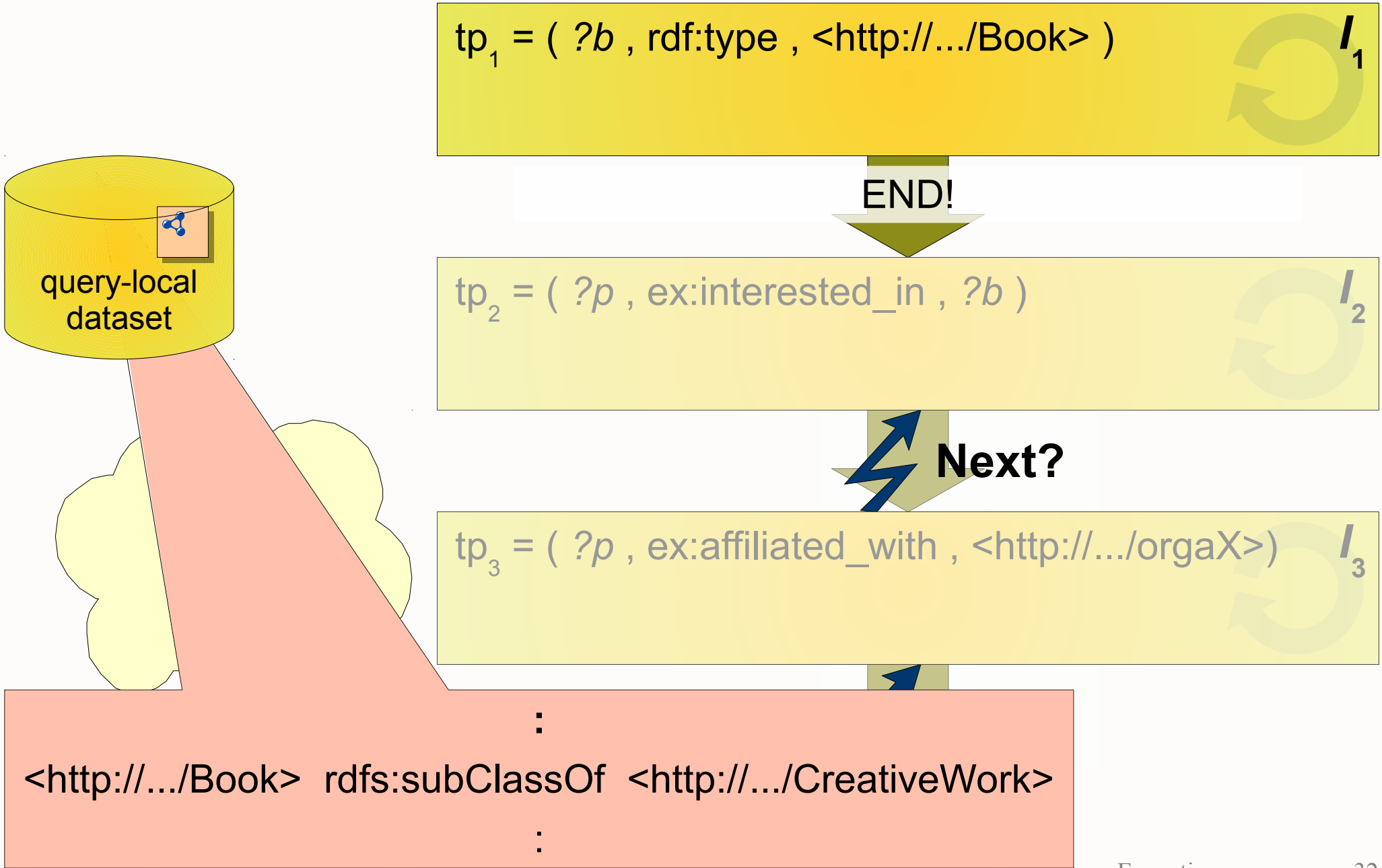
An Alternative Execution Plan



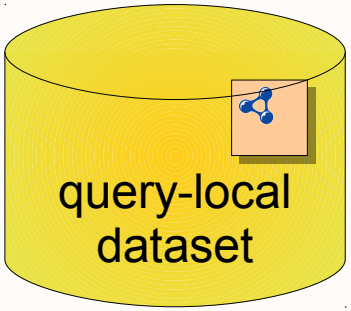
An Alternative Execution Plan

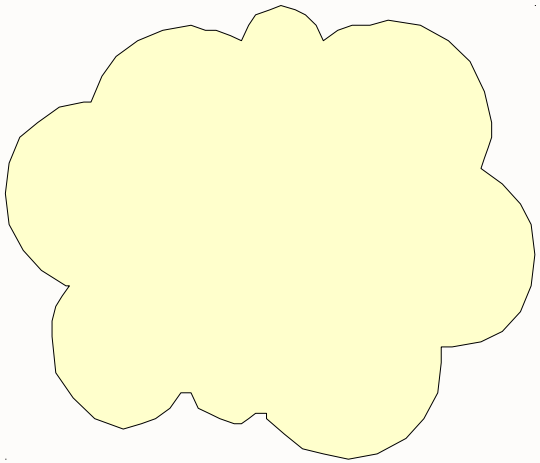


An Alternative Execution Plan



An Alternative Execution Plan





tp_1 = (*?b* , rdf:type , <http://.../Book>) I_1

END!

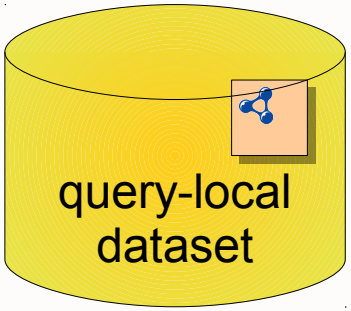
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

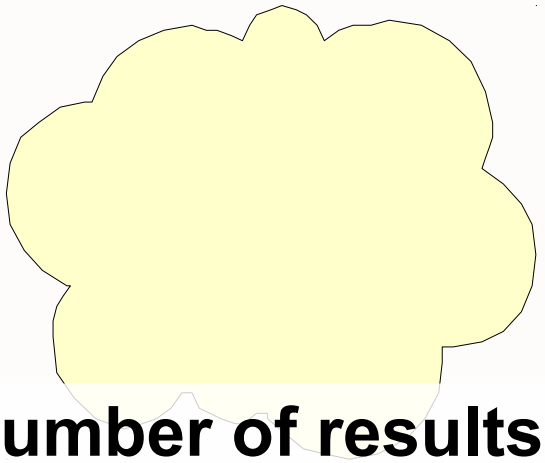
END!

tp_3 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_3

END!

An Alternative Execution Plan





tp_1 = (*?b* , rdf:type , <http://.../Book>) I_1

END!

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

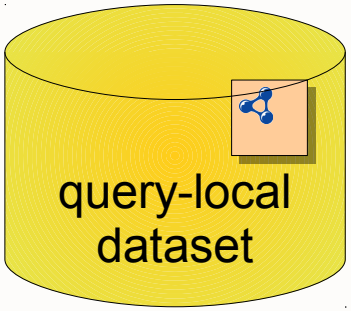
END!

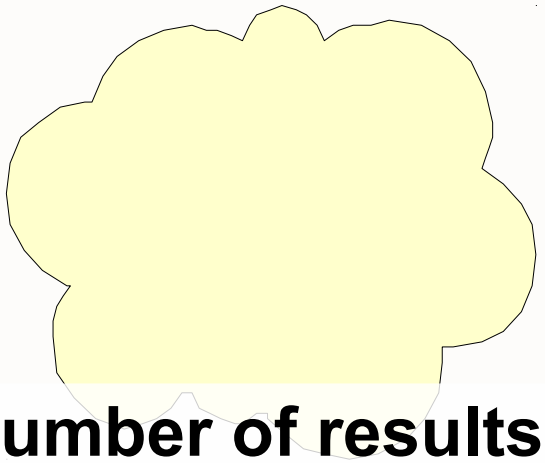
tp_3 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_3

END!

Number of results may depend on the order of triple patterns

An Alternative Execution Plan





tp_1 = (*?b* , rdf:type , <http://.../Book>) I_1

END!

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

END!

tp_3 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_3

END!

Number of results may depend on the order of triple patterns

= logical query execution plan

Query Plan Selection

- **Assessment criteria:**
  - Cost (query execution time)
  - Benefit (number of results)

Query Plan Selection

- **Assessment criteria:**
  - Cost (query execution time)
  - Benefit (number of results)
- **Cost and benefit must be estimated without plan execution**
- **Estimation impossible due to "*zero knowledge*"**

Query Plan Selection

- **Assessment criteria:**
  - Cost (query execution time)
  - Benefit (number of results)
- **Cost and benefit must be estimated without plan execution**
- **Estimation impossible due to "*zero knowledge*"**
- **Heuristic Based Plan Selection**
  - DEPENDENCY RESPECT RULE
  - SEED TP RULE
  - NO VOCAB SEED RULE
  - FILTERING TP RULE

SEED TP RULE

Use a plan with a *seed triple pattern*

- **Potential seed triple pattern**

  … is a triple pattern that contains at least one HTTP URI

Query

```
?p  ex:affiliated_with  <http://.../orgaX>
?p  ex:interested_in  ?b
?b  rdf:type  <http://.../Book>
```

SEED TP RULE

Use a plan with a *seed triple pattern*

- **Potential seed triple pattern**

  … is a triple pattern that contains at least one HTTP URI

Query

?p ex:affiliated_with <http://.../orgaX> √

?p ex:interested_in *?b* √

?b rdf:type <http://.../Book> √

SEED TP RULE

Use a plan with a *seed triple pattern*

- **Potential seed triple pattern**

  … is a triple pattern that contains at least one HTTP URI

- **Seed triple pattern of a plan**

  … is the first triple pattern in the plan and

  … is a potential seed triple pattern

Query

?p ex:affiliated_with <http://.../orgaX> √

?p ex:interested_in *?b* √

?b rdf:type <http://.../Book> √

SEED TP RULE

Use a plan with a *seed triple pattern*

- **Potential seed triple pattern**

  … is a triple pattern that contains at least one HTTP URI

- **Seed triple pattern of a plan**

  … is the first triple pattern in the plan and

  … is a potential seed triple pattern

Query

?p ex:affiliated_with <http://.../orgaX> √

?p ex:interested_in *?b* √

?b rdf:type <http://.../Book> √

- **Rationale: good starting point**

No Vocab Seed Rule

Avoid a seed triple pattern with vocabulary terms

- **Not only vocabulary term URIs in the seed triple pattern**
- **Patterns to avoid:** *?s* ex:any_property *?o*

  ?s rdf:type ex:any_class

Query

```
?p  ex:affiliated_with  <http://.../orgaX>
?p  ex:interested_in  ?b
?b  rdf:type  <http://.../Book>
```

No Vocab Seed Rule

Avoid a seed triple pattern with vocabulary terms

- **Not only vocabulary term URIs in the seed triple pattern**
- **Patterns to avoid:** *?s* ex:any_property *?o*

  ?s rdf:type ex:any_class

Query

?p ex:affiliated_with <http://.../orgaX> √

?p ex:interested_in *?b* ✗

?b rdf:type <http://.../Book> ✗

No Vocab Seed Rule

Avoid a seed triple pattern with vocabulary terms

- **Not only vocabulary term URIs in the seed triple pattern**
- **Patterns to avoid:** *?s* ex:any_property *?o*

  ?s rdf:type ex:any_class
- **Rationale: URIs for vocabulary term usually resolve to vocabulary definitions with little instance data**

Query

?p ex:affiliated_with <http://.../orgaX> √

?p ex:interested_in *?b* ✗

?b rdf:type <http://.../Book> ✗

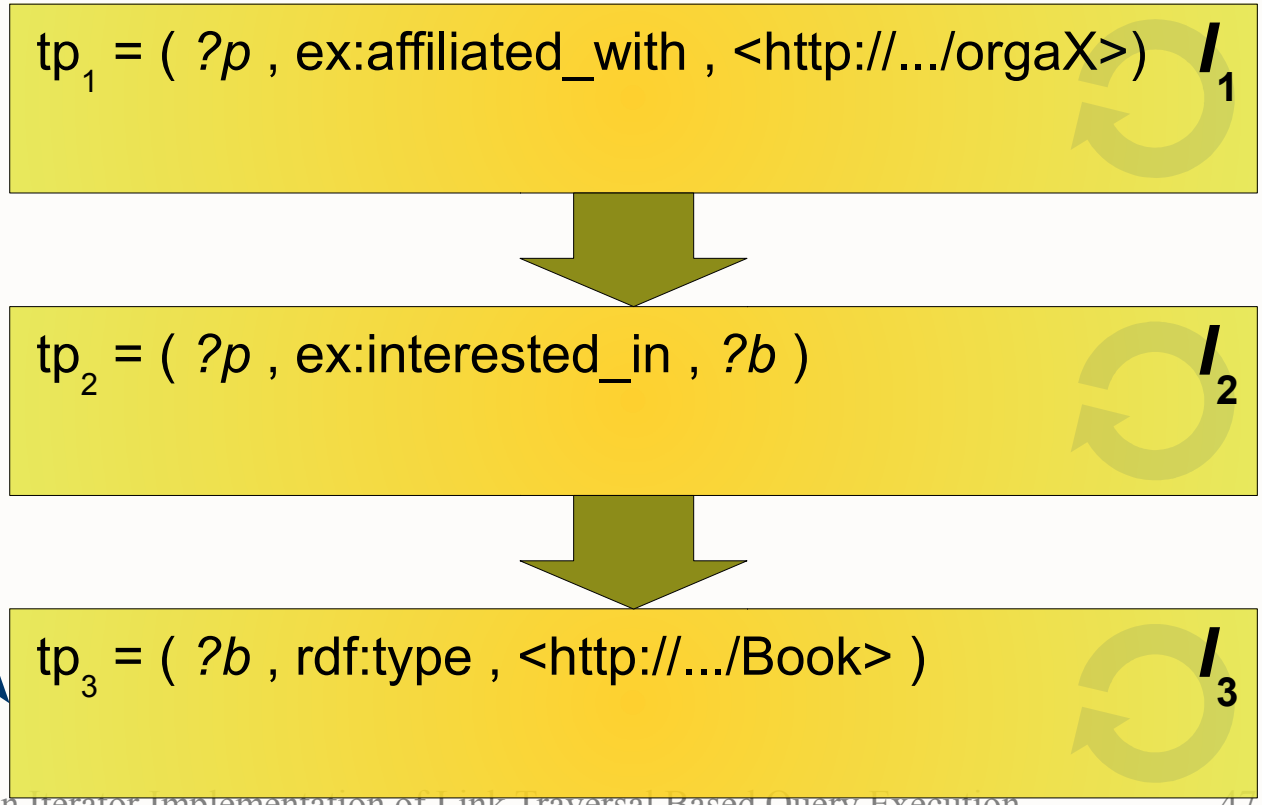
FILTERING TP RULE

Use a plan where all *filtering triple patterns* are as close to the seed triple pattern as possible

FILTERING TP RULE

Use a plan where all *filtering triple patterns* are as close to the seed triple pattern as possible

- **Filtering triple pattern**: ***each*** **variable already occurs in one of the preceding triple patterns**



FILTERING TP RULE

Use a plan where all *filtering triple patterns* are as close to the seed triple pattern as possible

- **Filtering triple pattern: *each* variable already occurs in one of the preceding triple patterns**
- **For each result consumed as input a filtering TP can only report 1 or 0 results as output**

tp_1 = (*?p* , ex:affiliated_with , <http://.../orgaX>) I_1

{ *?p* = <http://.../alice> }

tp_2 = (*?p* , ex:interested_in , *?b*) I_2

tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

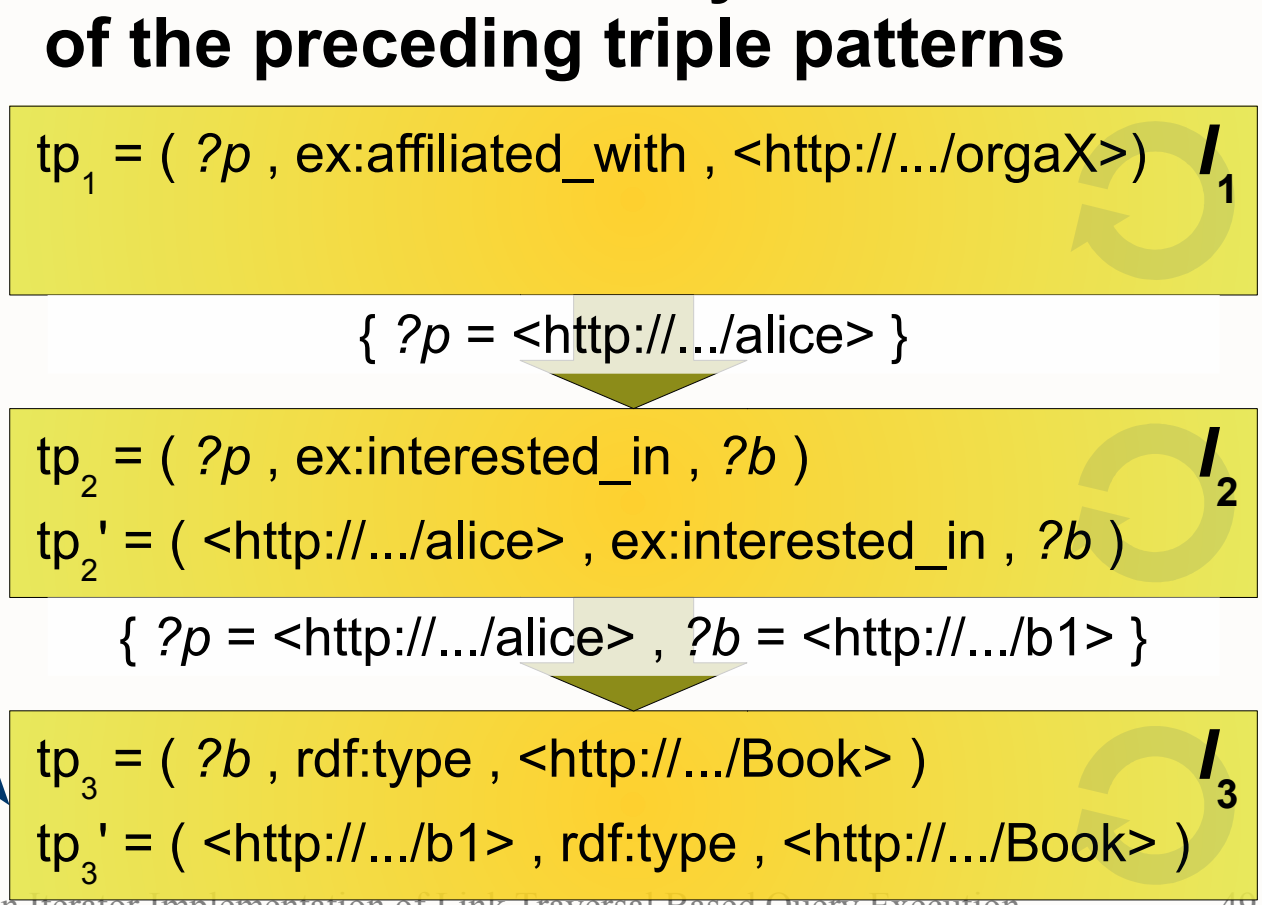
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

tp_3' = (<http://.../b1> , rdf:type , <http://.../Book>)

FILTERING TP RULE

Use a plan where all *filtering triple patterns* are as close to the seed triple pattern as possible

- **Filtering triple pattern**: ***each*** **variable already occurs in one of the preceding triple patterns**
- **For each result consumed as input a filtering TP can only report 1 or 0 results as output**
- **Rationale: Reduce cost**



Evaluation Procedure

- **Generate all possible plans**
- **Execute each plan:**
  - 5 runs (+ 1 initial warm-up run)
  - Use an initially empty query-local dataset for each run
- **Measure for each plan:**
  - Avg. execution time
  - Avg. number of descriptor objects retrieved during execution
  - Avg. number of query results

Evaluation Query (Example)

```
SELECT ?spec ?genus WHERE {
  geospecies:4qyn7 gs:inFamily ?fam .
  ?fam skos:narrowerTransitive ?spec .
  ?spec skos:closeMatch ?sp2 .
  ?sp2 rdfs:subClassOf ?genus .

  ?spec gs:isExpectedIn ?loc .
  geospecies:4qyn7 gs:isExpectedIn ?loc
  ?loc rdf:type gs:State . }
```

Of what genus are the species that are

- classified in the same family as the American Badger,
- and expected in the same states as the American Badger ?

Picture source: Wikipedia

Evaluation Query (Example)

```
SELECT ?spec ?genus WHERE {
  geospecies:4qyn7 gs:inFamily ?fam .
  ?fam skos:narrowerTransitive ?spec .
  ?spec skos:closeMatch ?sp2 .
  ?sp2 rdfs:subClassOf ?genus .

  ?spec gs:isExpectedIn ?loc .
  geospecies:4qyn7 gs:isExpectedIn ?loc
  ?loc rdf:type gs:State . }
```

- **2 potential seed triple patterns that satisfy our NO SEED VOCAB RULE**

Of what genus are the species that are

- classified in the same family as the American Badger,
- and expected in the same states as the American Badger ?

Picture source: Wikipedia

Evaluation Query (Example)

```
SELECT ?spec ?genus WHERE {
  geospecies:4qyn7 gs:inFamily ?fam .
  ?fam skos:narrowerTransitive ?spec .
  ?spec skos:closeMatch ?sp2 .
  ?sp2 rdfs:subClassOf ?genus .

  ?spec gs:isExpectedIn ?loc .
  geospecies:4qyn7 gs:isExpectedIn ?loc
  ?loc rdf:type gs:State . }
```

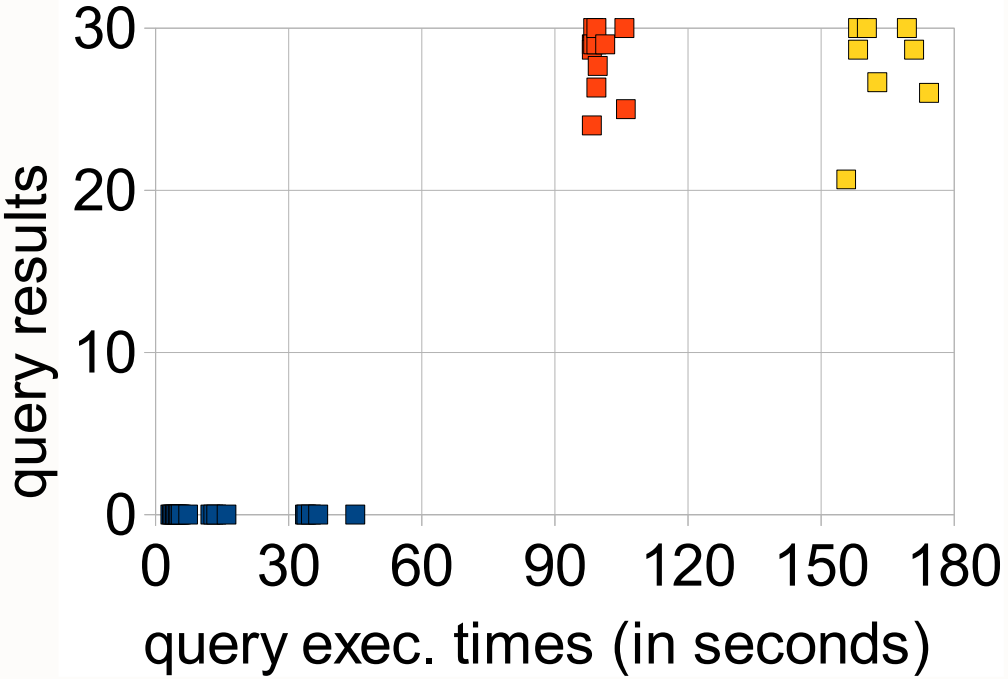
- **2 potential seed triple patterns that satisfy our NO SEED VOCAB RULE**
- **56 different plans, each contains 2 filtering triple patterns**

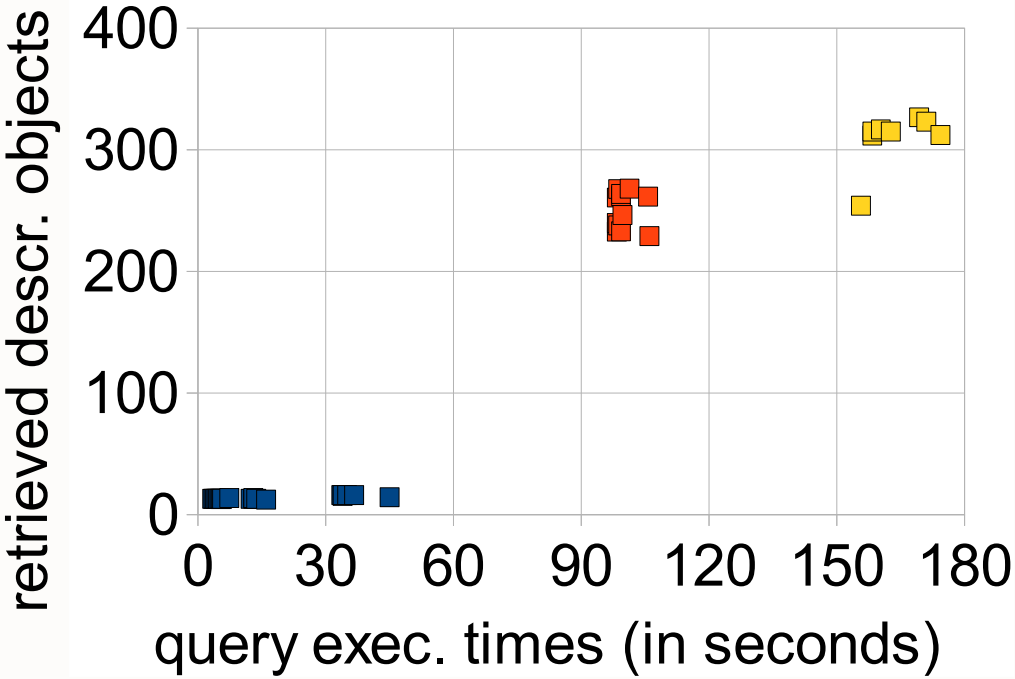
Of what genus are the species that are

- classified in the same family as the American Badger,
- and expected in the same states as the American Badger ?

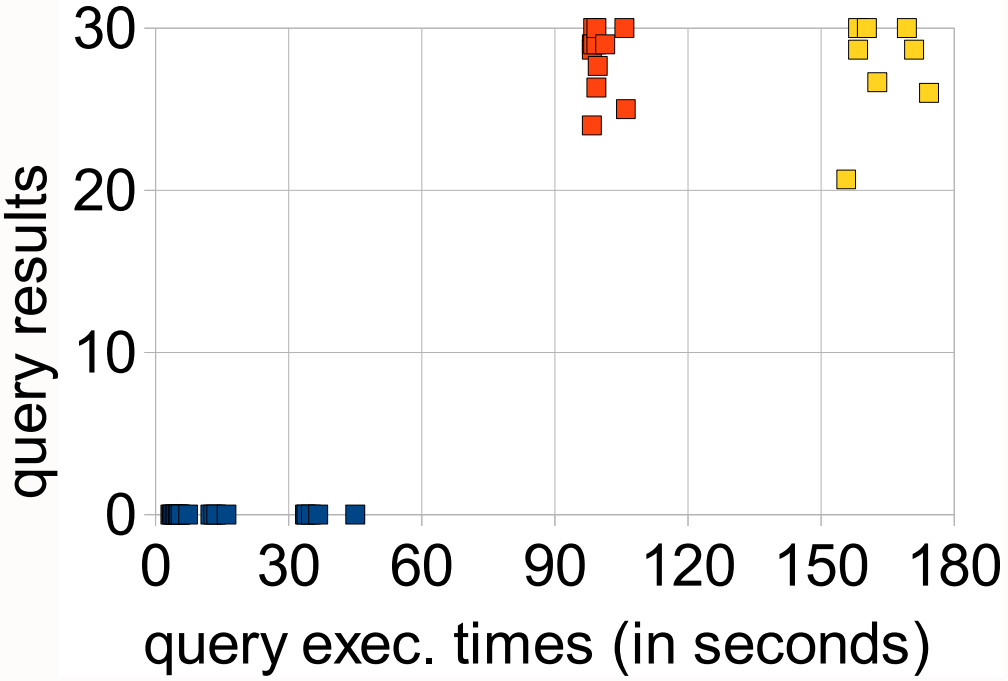
Picture source: Wikipedia

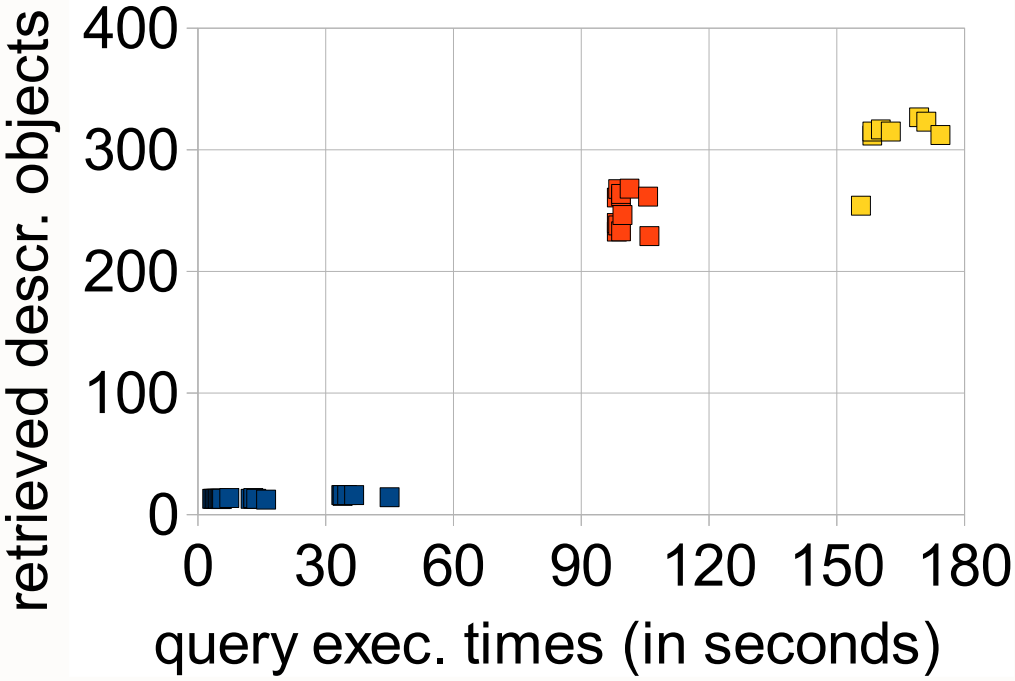
Measurements





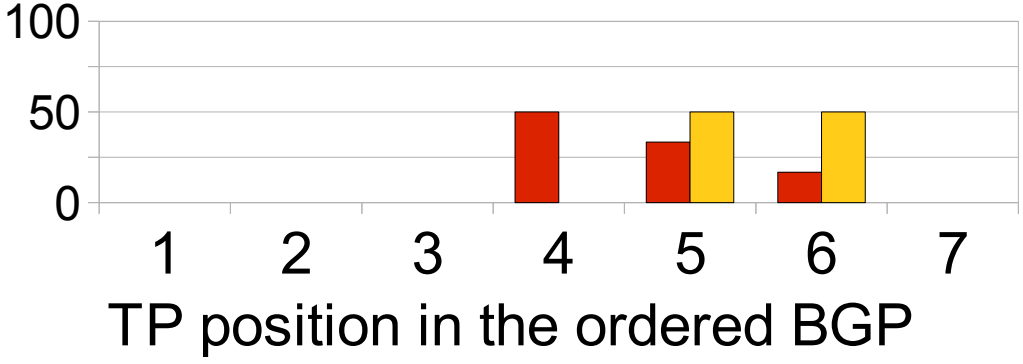
Measurements



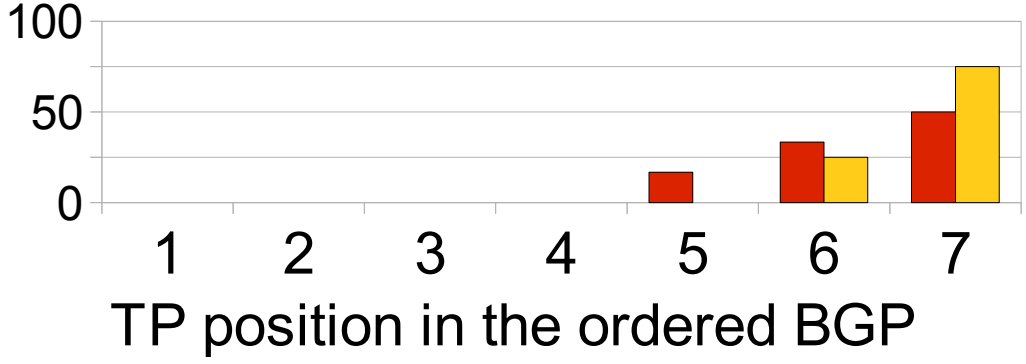


Percentage of plans in each group with a filtering TP in specific positions

1st Filtering TP



2nd Filtering TP



Conclusions

- **Approach that uses iterators to implement Link Traversal Based Query Execution**
  - … is sound
  - … guarantees termination
  - … cannot guarantee (reachability-) completeness
- **Degree of completeness depends on the query plans (i.e. orders of the BGP)**
- **Heuristic based plan selection**
- **Next steps:**
  - Algorithm to generate most suitable plans only
  - Integrate adaptive query processing techniques

Backup Slides

Outline

1. Link Traversal Based Query Execution
2. Characteristics of the Iterator Based Implementation Approach
3. Query Plan Selection
4. Evaluation

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

Link Traversal Based Querying

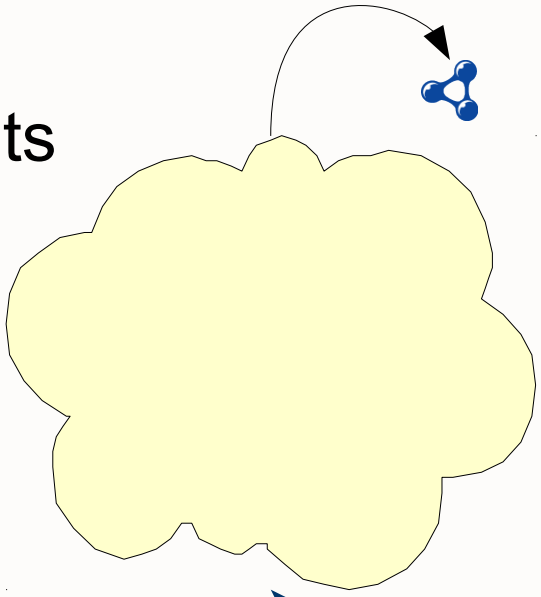
- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

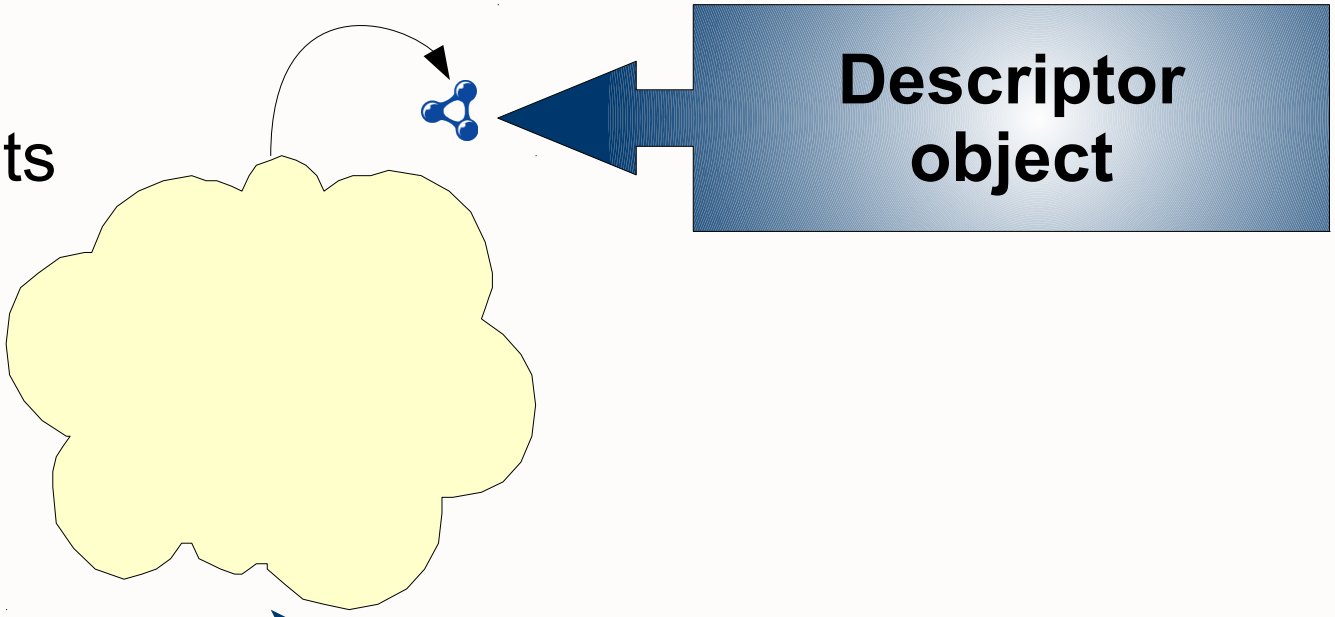


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

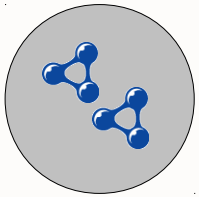


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

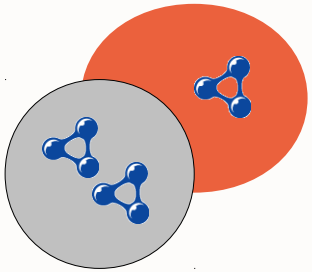


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

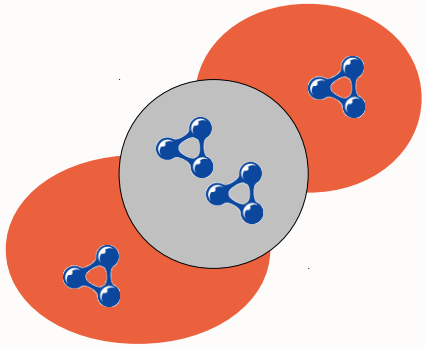


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

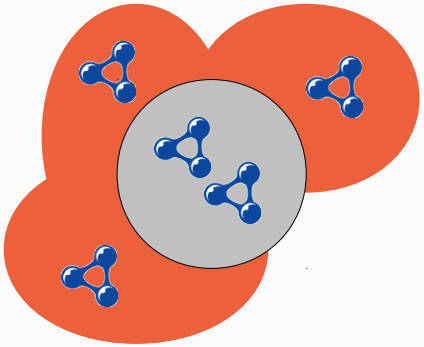


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

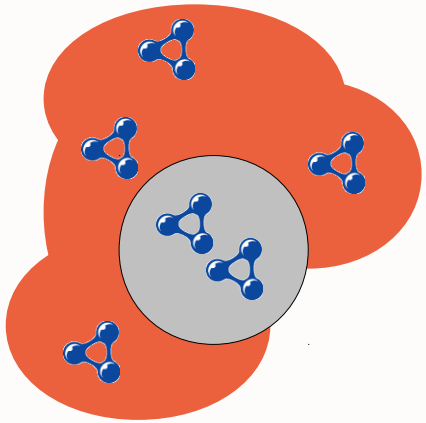


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

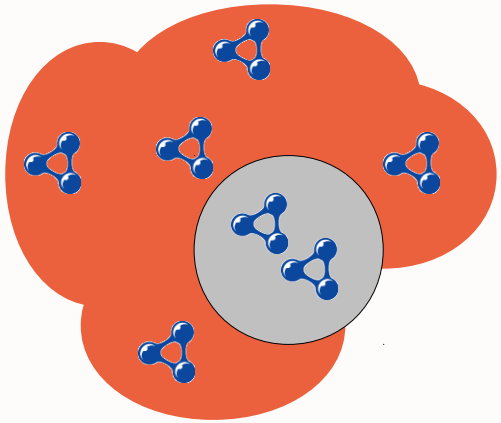


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

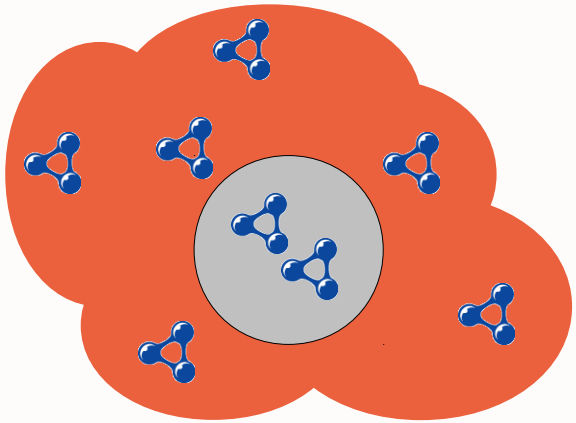


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

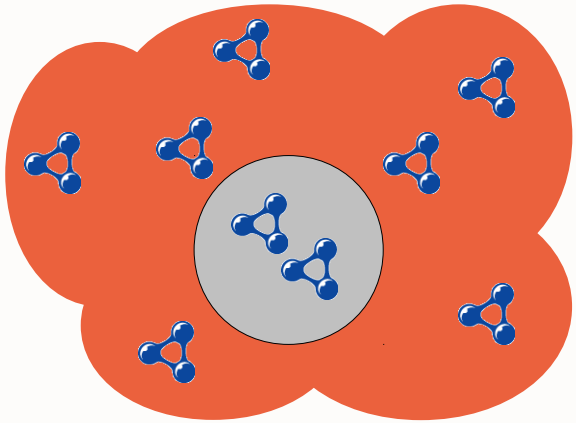


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

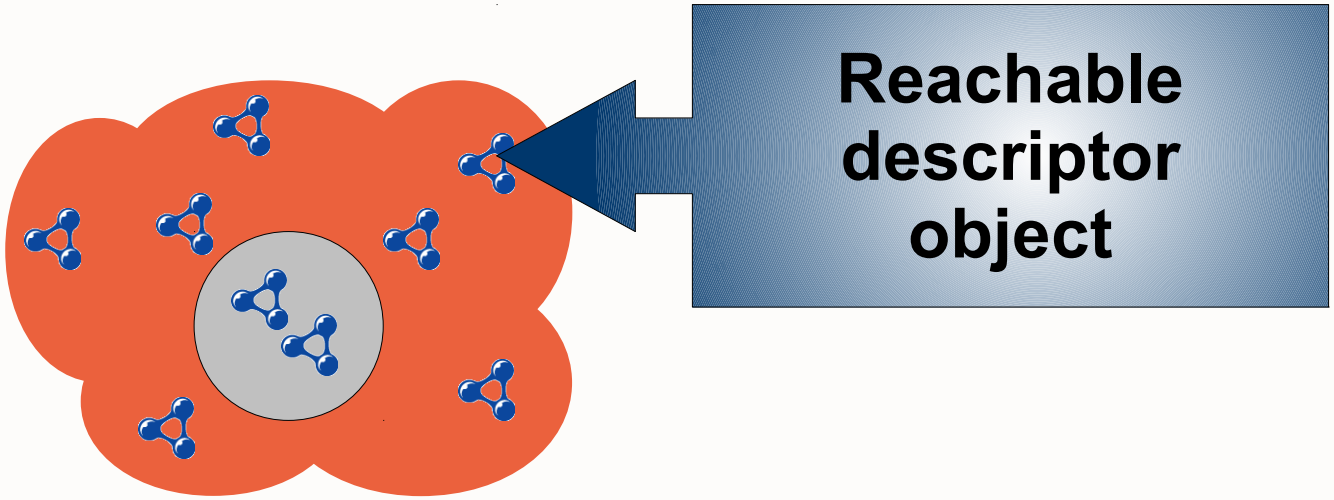


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results

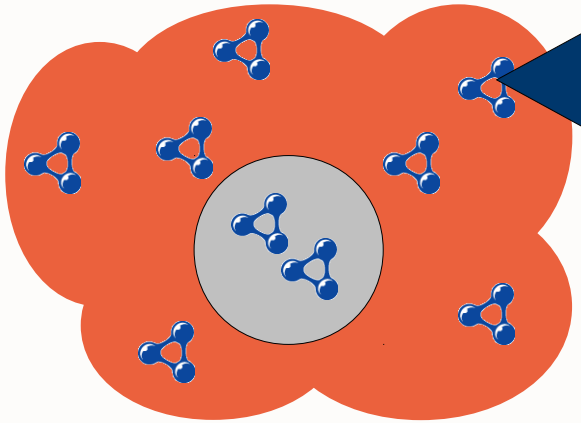


Query

?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results



Reachable descriptor object

The mapping $\mu : V \rightarrow U \cup B \cup L$ is a **result** to BGP *bgp* iff:

Condition 1: $dom(\mu) = vars(bgp)$

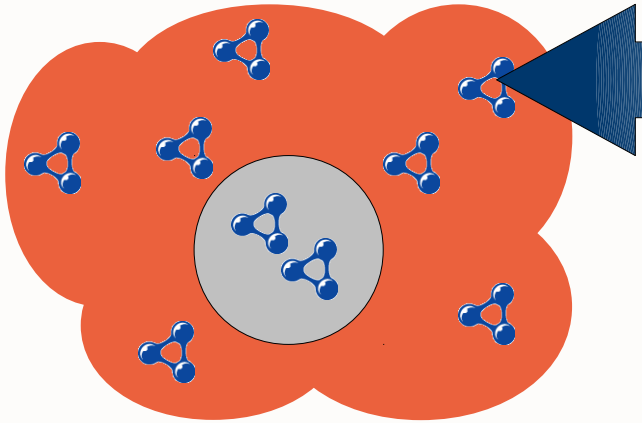
Condition 2: $\mu[bgp] \subseteq D_{reachable}$

Query

```
?p  ex:affiliated_with  <http://.../orgaX>
?p  ex:interested_in  ?b
?b  rdf:type  <http://.../Book>
```

Link Traversal Based Querying

- **Semantics defined in two phases:**
  1. Reachability
  2. Query Results



Reachable descriptor object

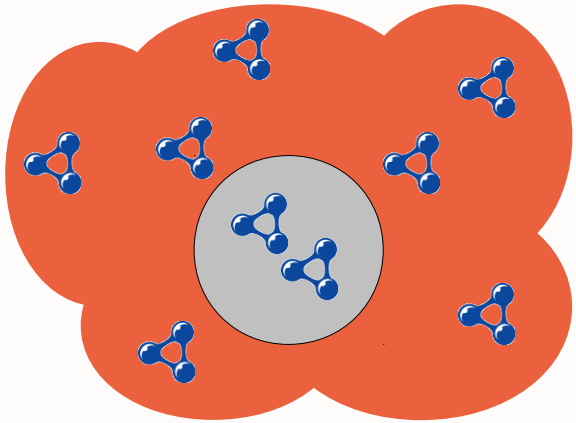
The mapping $\mu : V \rightarrow U \cup B \cup L$ is a **result** to BGP *bgp* iff:

Condition 1: $\text{dom}(\mu) = \text{vars}(bgp)$

Condition 2: $\mu[bgp] \subseteq D_{\text{reachable}}$

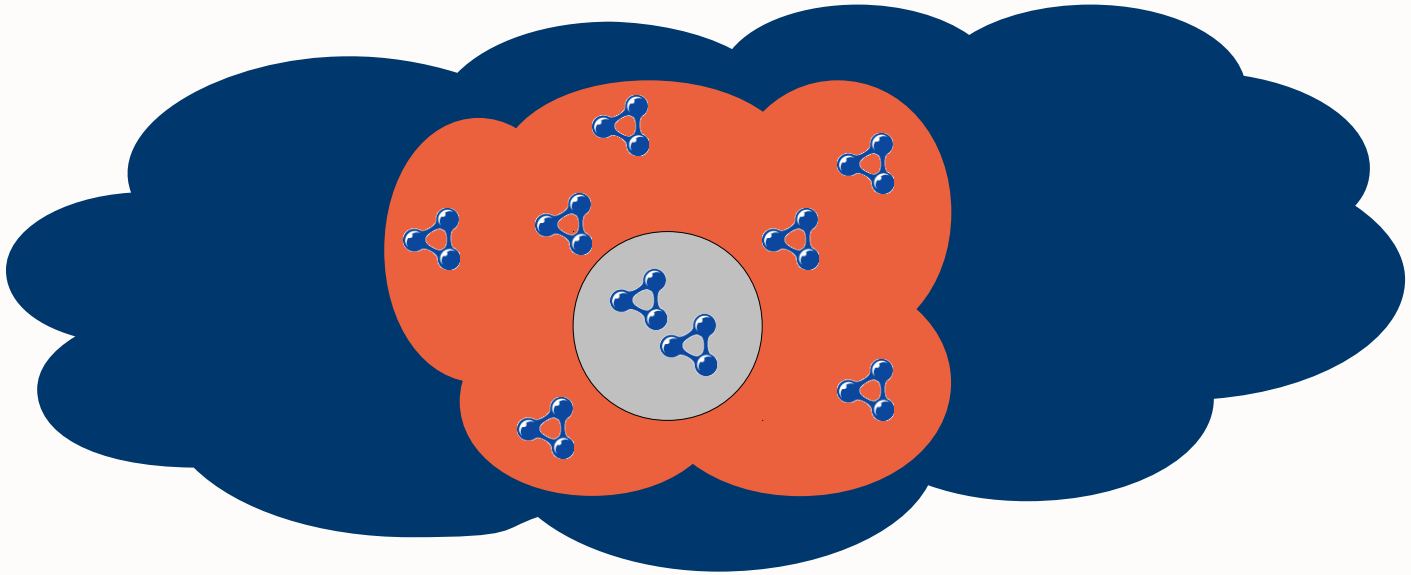
- **2 phases do not reflect idea of actual execution strategy**
  - Intertwine query evaluation with the traversal of data links

Characteristics



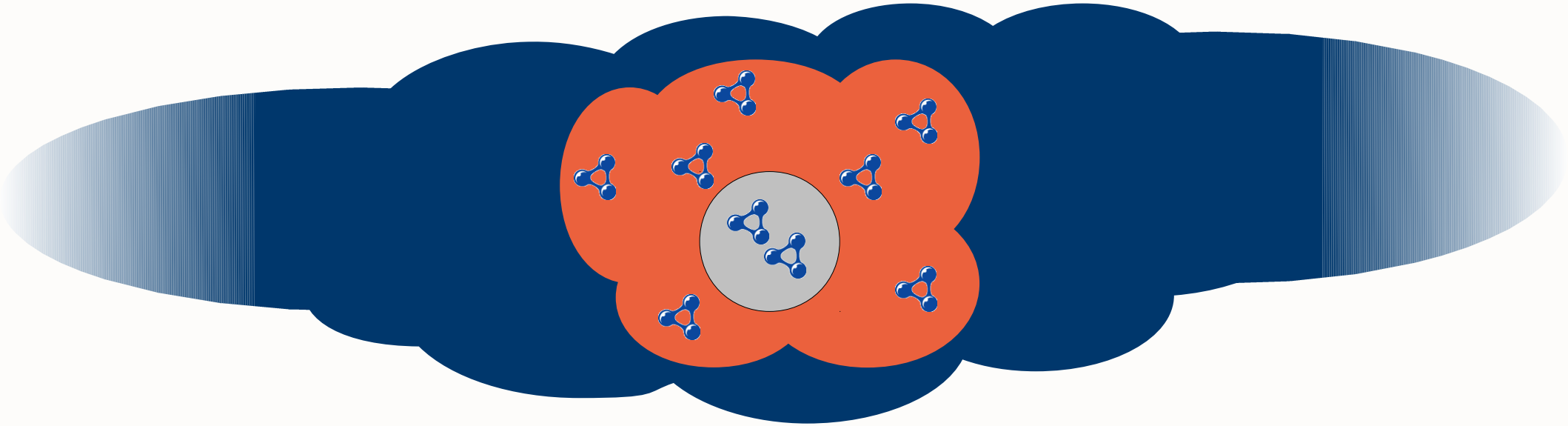
- **No need to know all data sources in advance**

Characteristics



- **No need to know all data sources in advance**
- **Never complete w.r.t. all data on the Web**
  - Reason: reachability based on links that match query patterns
  - New concept: reachability-completeness

Characteristics



- **No need to know all data sources in advance**
- **Never complete w.r.t. all data on the Web**
  - Reason: reachability based on links that match query patterns
  - New concept: reachability-completeness
- **No guarantee for termination**
  - Reason: Web of Data is infinite (at any point in time)

Outline

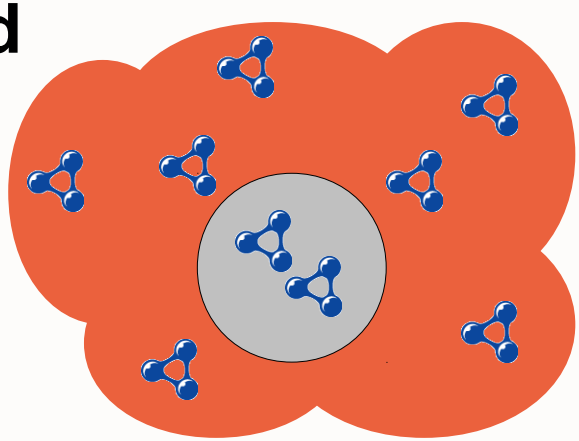
1. Characteristics of the Implementation Approach
2. Query Plan Selection
3. Evaluation

Characteristics

- **Sound**

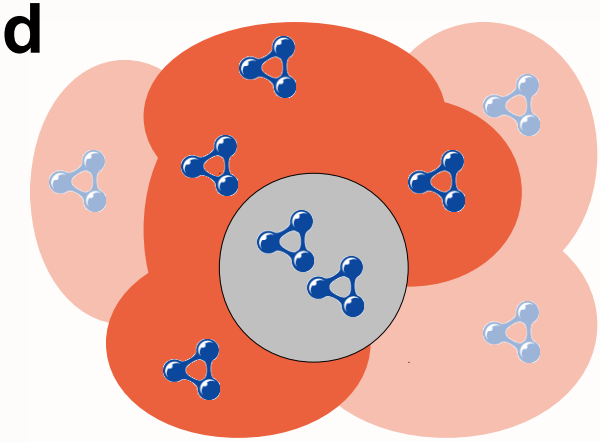
Characteristics

- **Sound**
- **Number of results may depend on order of triple patterns**
- **Reachability-completeness not guaranteed**



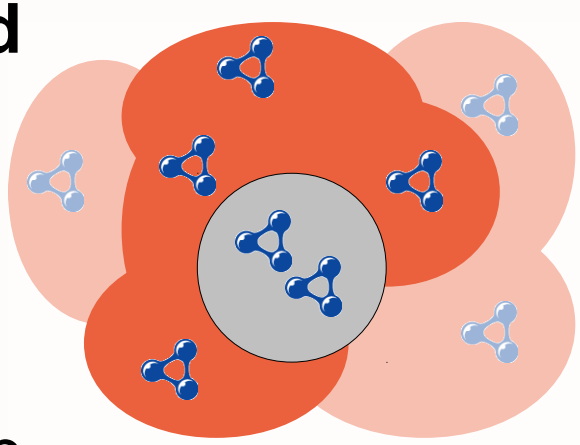
Characteristics

- **Sound**
- **Number of results may depend on order of triple patterns**
- **Reachability-completeness not guaranteed**
  - Main reason: inflexibility due to fixed order



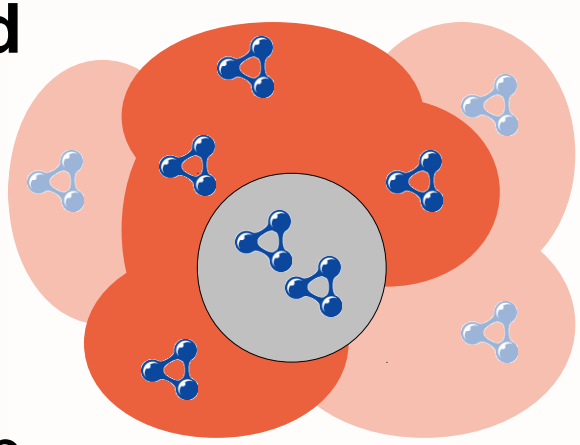
Characteristics

- **Sound**
- **Number of results may depend on order of triple patterns**
- **Reachability-completeness not guaranteed**
  - Main reason: inflexibility due to fixed order
- **Termination is guaranteed**
  - No such guarantee for link traversal based query execution in general because the Web of Data is infinite (at any point in time)



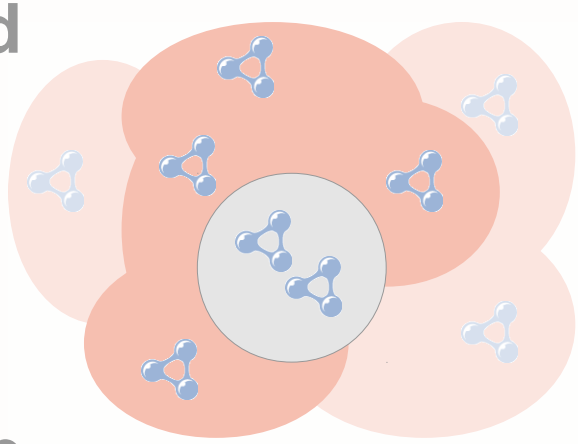
Characteristics

- **Sound**
- **Number of results may depend on order of triple patterns**
- **Reachability-completeness not guaranteed**
  - Main reason: inflexibility due to fixed order
- **Termination is guaranteed**
  - No such guarantee for link traversal based query execution in general because the Web of Data is infinite (at any point in time)
- **Efficient**
- **Easy to apply in existing query engines**



Characteristics

- Sound
- **Number of results may depend on order of triple patterns**
- Reachability-completeness not guaranteed
  - Main reason: inflexibility due to fixed order
- Termination is guaranteed
  - No such guarantee for link traversal based query execution in general because the Web of Data is infinite (at any point in time)
- Efficient
- Easy to apply in existing query engines



Characteristics

- Sound
- **Number of results may depend on order of triple patterns**
- Reachability-completeness
  - Main reason: inflexibility due to fixed order
- Termination is guaranteed
  - No such guarantee for link traversal based query execution in general because the Web of Data is infinite (at any point in time)
- Efficient
- Easy to apply in existing query engines

= logical query execution plan

Plan Selection Problem

- **Number of results may depend on order of triple patterns**

= logical query execution plan

→ **Problem: select a suitable plan**

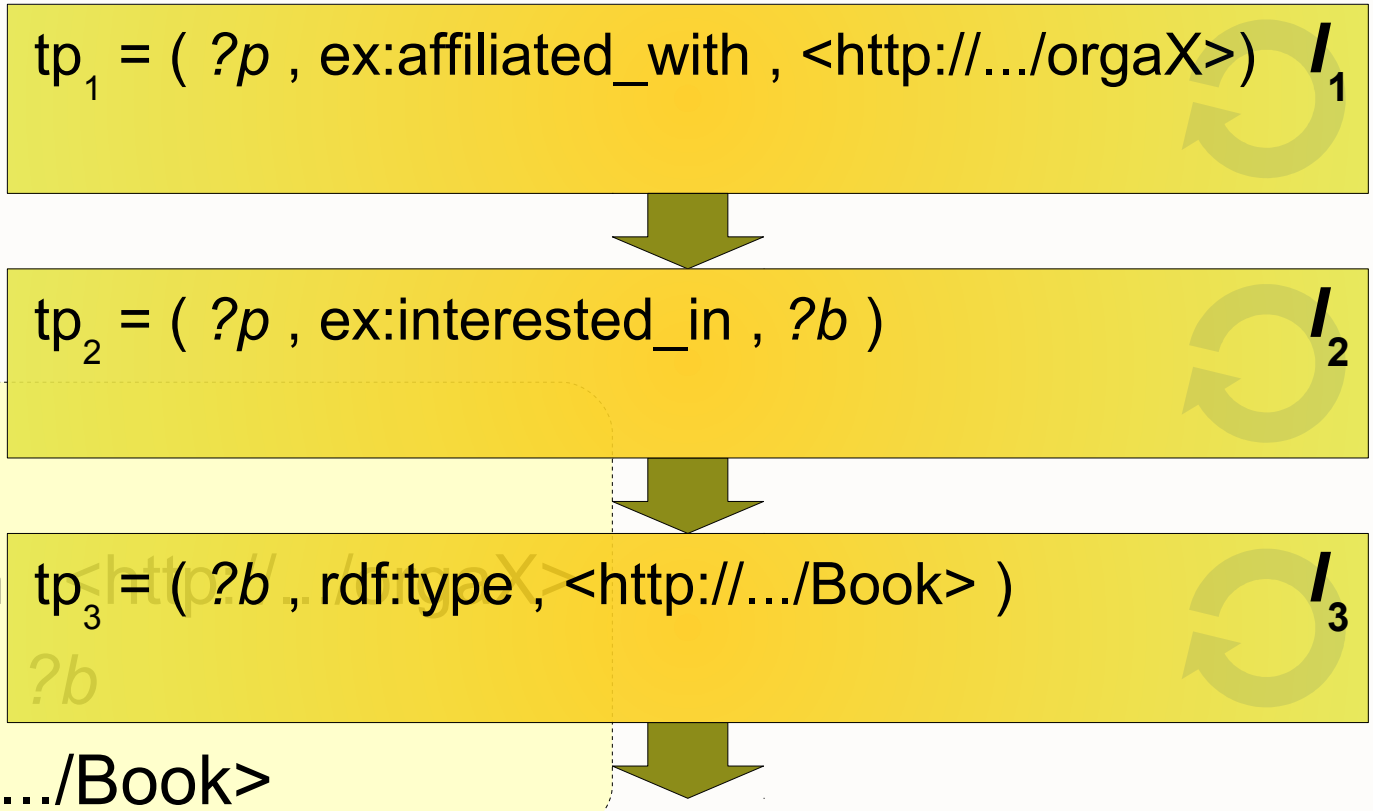
Outline

1. Link Traversal Based Query Execution
2. Characteristics of the Iterator Based Implementation Approach
3. **Query Plan Selection**
4. Evaluation

Dependency Respect Rule

Use a *dependency respecting* query plan

- **Dependency respect**: **a variable from each triple pattern already occurs in one of the preceding triple patterns**



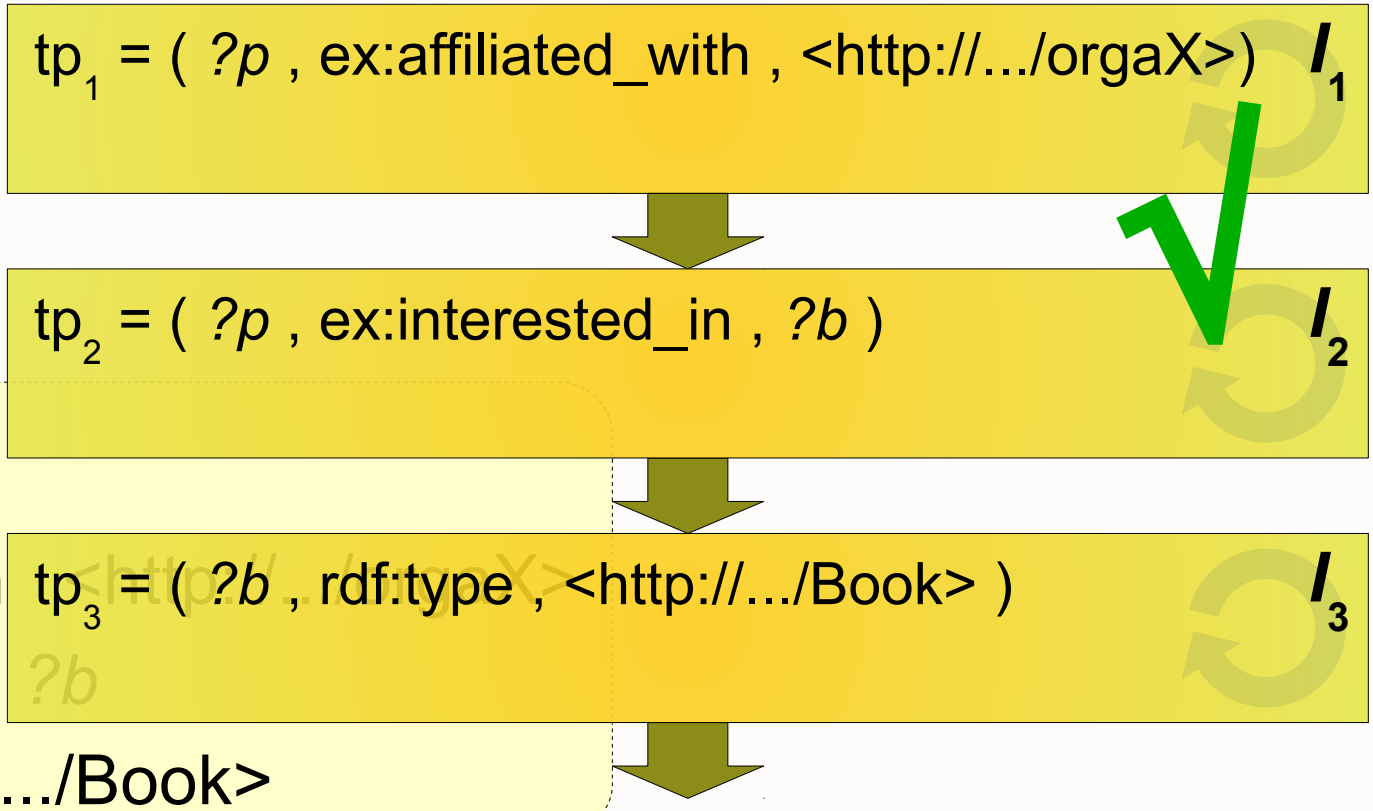
Query

```
?p  ex:affiliated_with <http://.../orgaX>
?p  ex:interested_in   ?b
?b  rdf:type   <http://.../Book>
```

Dependency Respect Rule

Use a *dependency respecting* query plan

- **Dependency respect**: **a variable from each triple pattern already occurs in one of the preceding triple patterns**



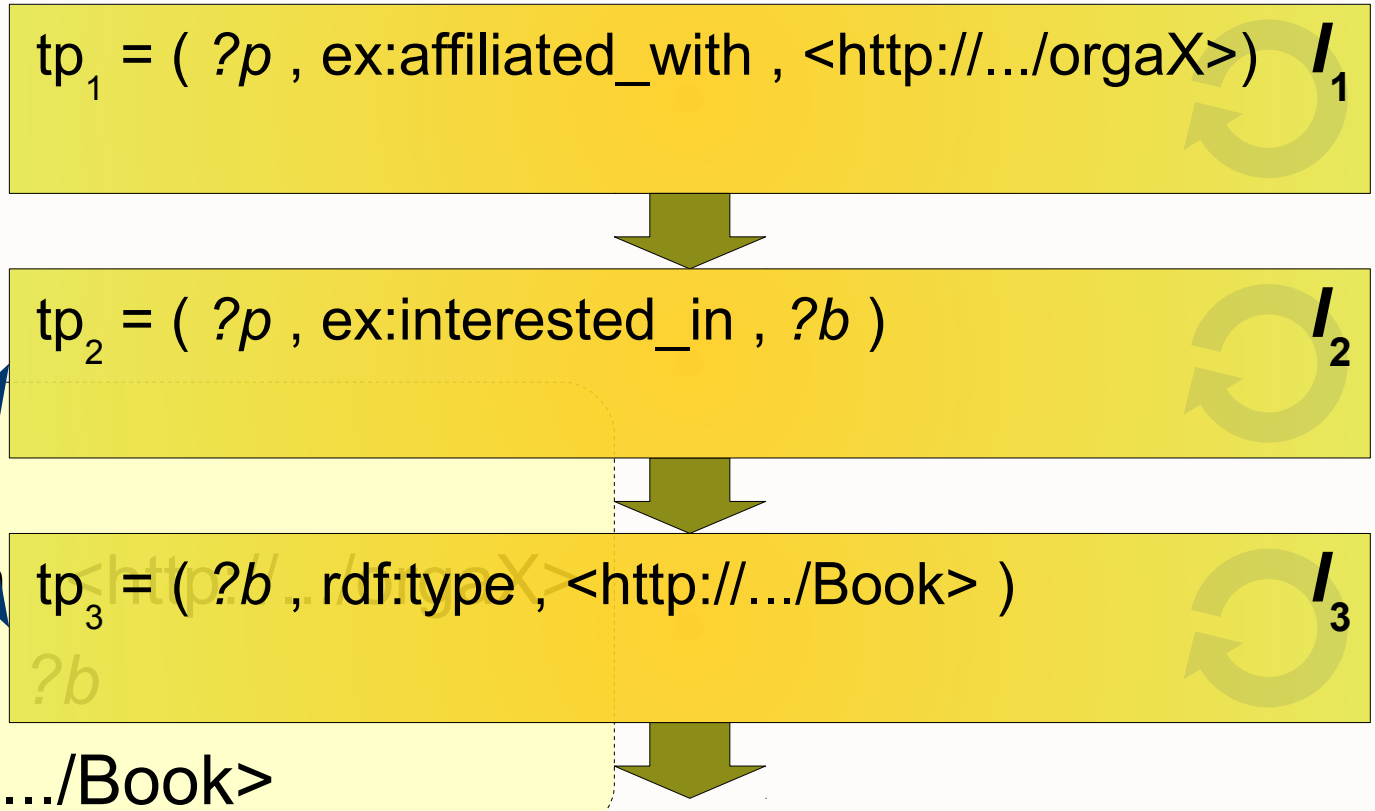
Query

```
?p  ex:affiliated_with <http://.../orgaX>
?p  ex:interested_in   ?b
?b  rdf:type   <http://.../Book>
```

Dependency Respect Rule

Use a *dependency respecting* query plan

- **Dependency respect**: **a variable from each triple pattern already occurs in one of the preceding triple patterns**



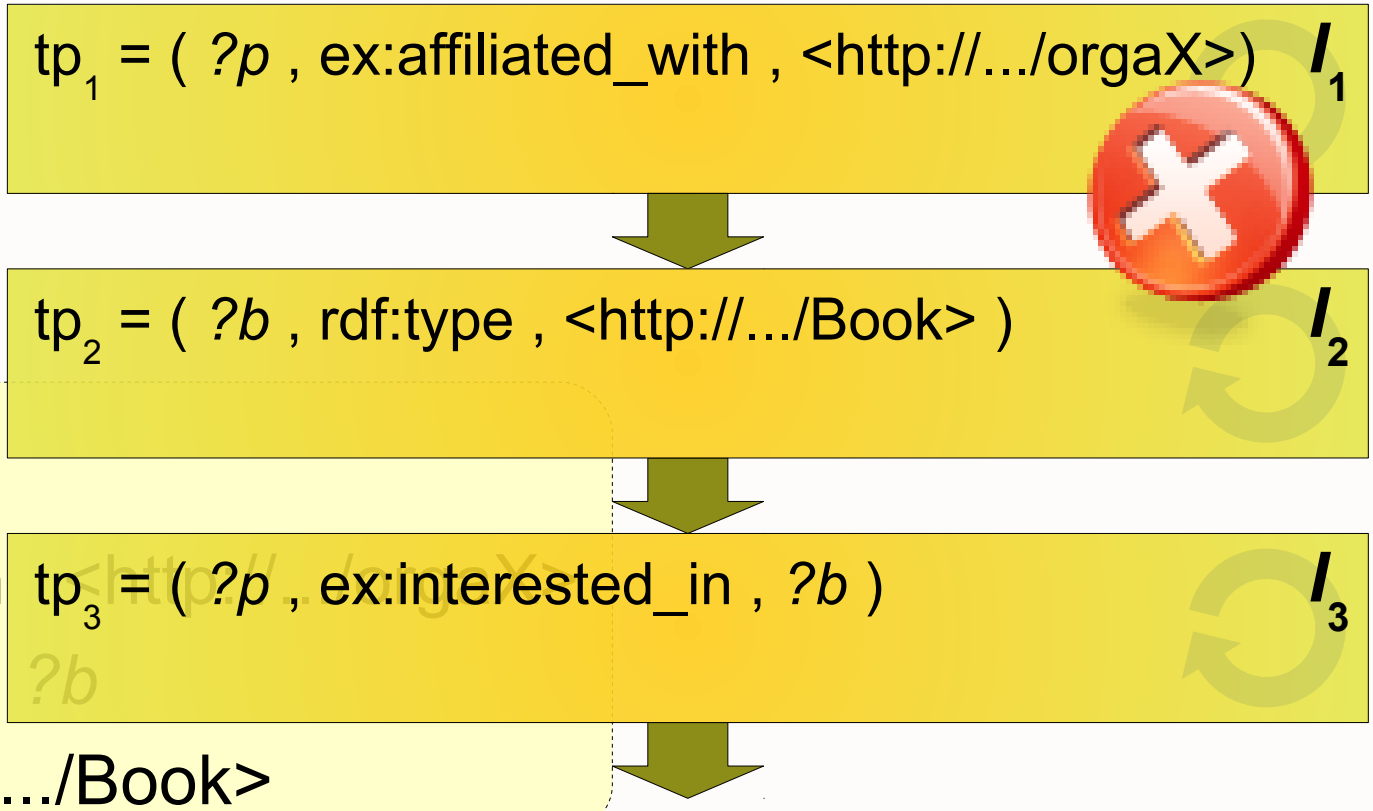
Query

```
?p  ex:affiliated_with  <http://.../orgaX> .
?p  ex:interested_in    ?b .
?b  rdf:type            <http://.../Book> .
```

Dependency Respect Rule

Use a *dependency respecting* query plan

- **Dependency respect**: **a variable from each triple pattern already occurs in one of the preceding triple patterns**



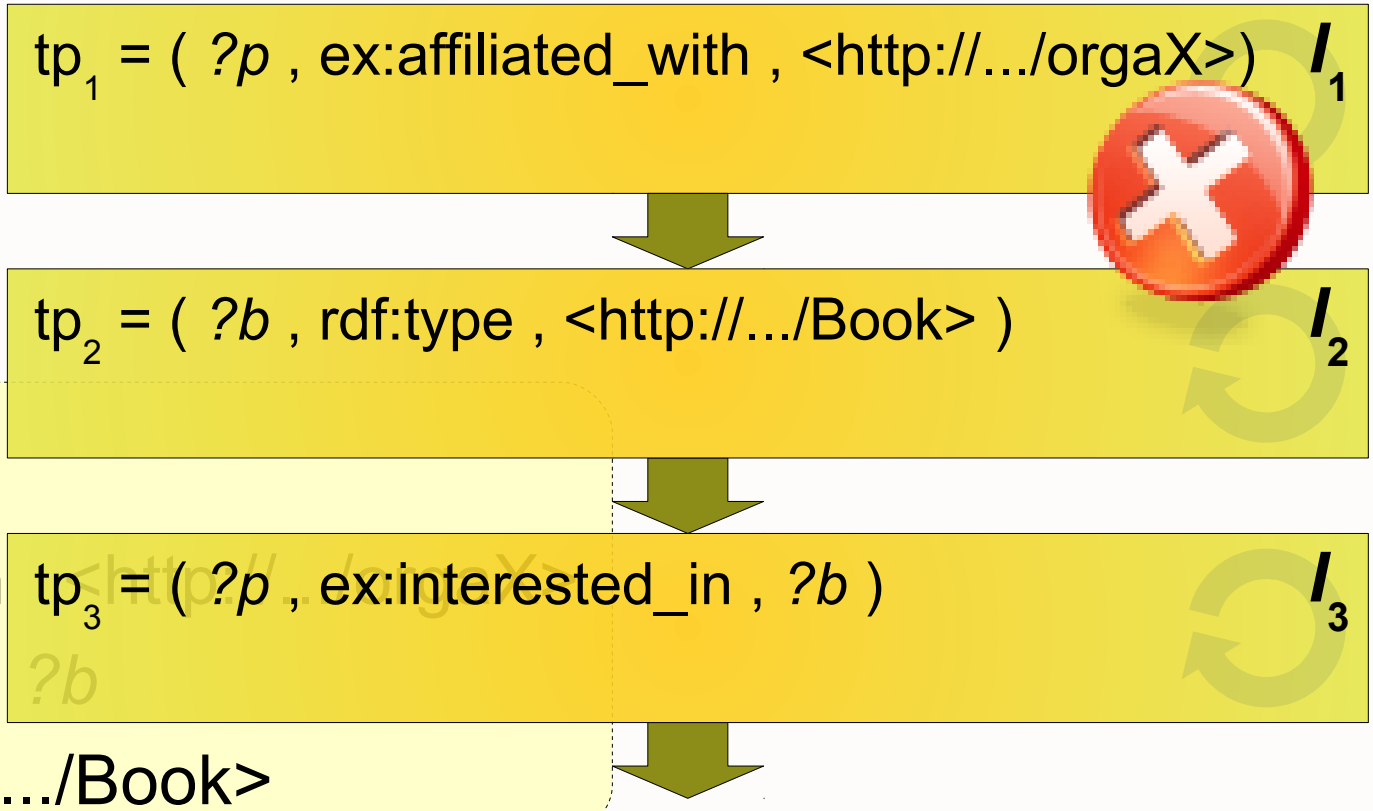
Query

```
?p  ex:affiliated_with <http://.../orgaX>
?p  ex:interested_in   ?b
?b  rdf:type   <http://.../Book>
```

Dependency Respect Rule

Use a *dependency respecting* query plan

- **Dependency respect: a variable from each triple pattern already occurs in one of the preceding triple patterns**
- **Rationale: Avoid cartesian products**



Query

```
?p  ex:affiliated_with <http://.../orgaX>
?p  ex:interested_in   ?b
?b  rdf:type   <http://.../Book>
```

These slides have been created by
Olaf Hartig

http://olafhartig.de