Linked Data Query Processing

Tutorial at the 22nd International World Wide Web Conference (WWW 2013)

May 14, 2013

http://db.uwaterloo.ca/LDQTut2013/

4. Execution Process

Olaf Hartig
University of Waterloo

Outline

- **Separated Execution**
- **Integrated Execution**
  - General Idea and Properties
  - Push-Based Implementation [LT10, LT11]
  - Link Traversal Based Query Execution

Separated Execution Approaches

… clearly separate

data retrieval

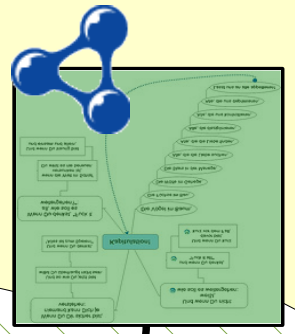
1

and

result construction

2

into two
consecutive phases



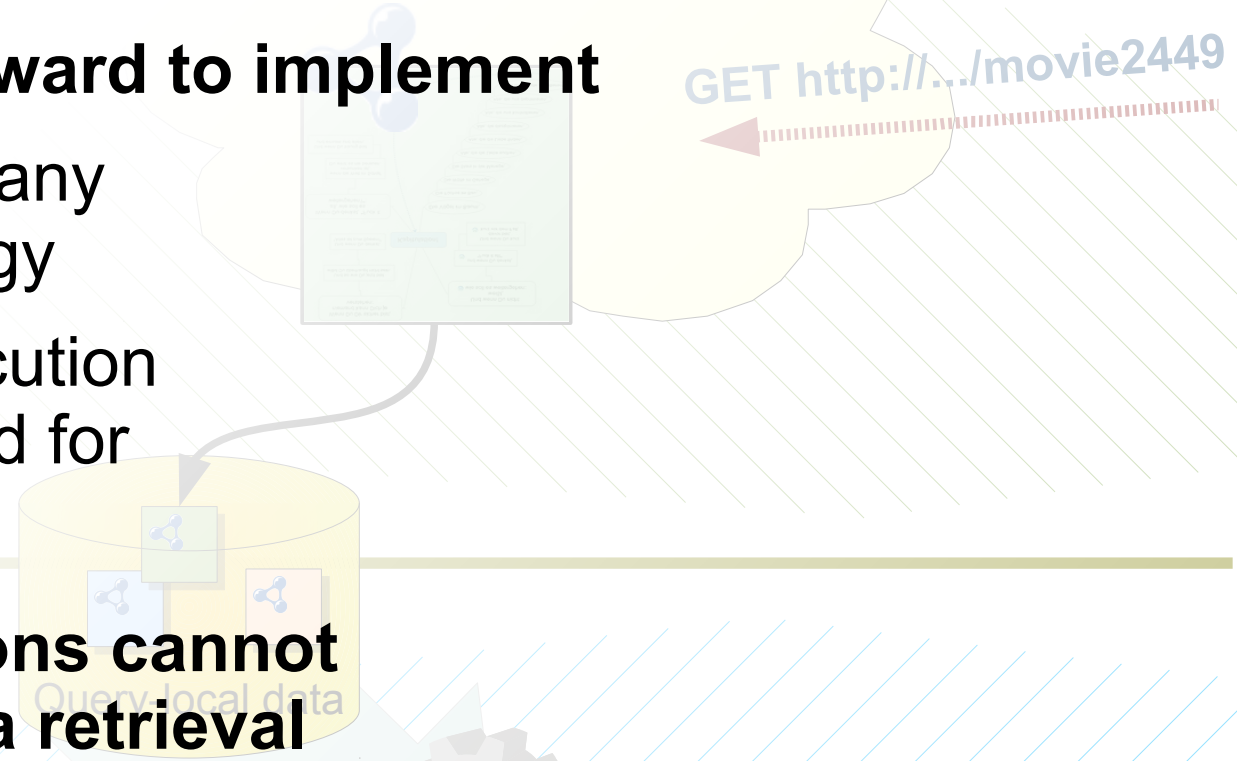
GET http://.../movie2449

Query-local data

?actor	?loc
http://mdb.../Paul http://mdb.../Ric	http://geo.../Berlin http://geo.../Rome

Properties of Separated Execution

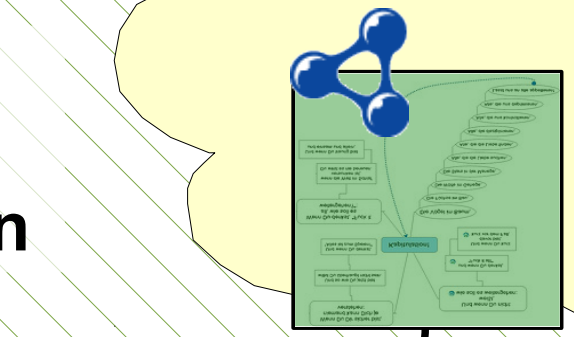
- **Advantage: straightforward to implement**
  - Can be combined with any source selection strategy
  - A traditional query execution plan might then be used for constructing the result
- **Downside: First solutions cannot be reported before data retrieval has been completed**



?actor	?loc
http://mdb.../Paul http://mdb.../Ric	http://geo.../Berlin http://geo.../Rome

Integrated Execution Approaches

… intertwine data retrieval and result construction



GET http://.../movie2449

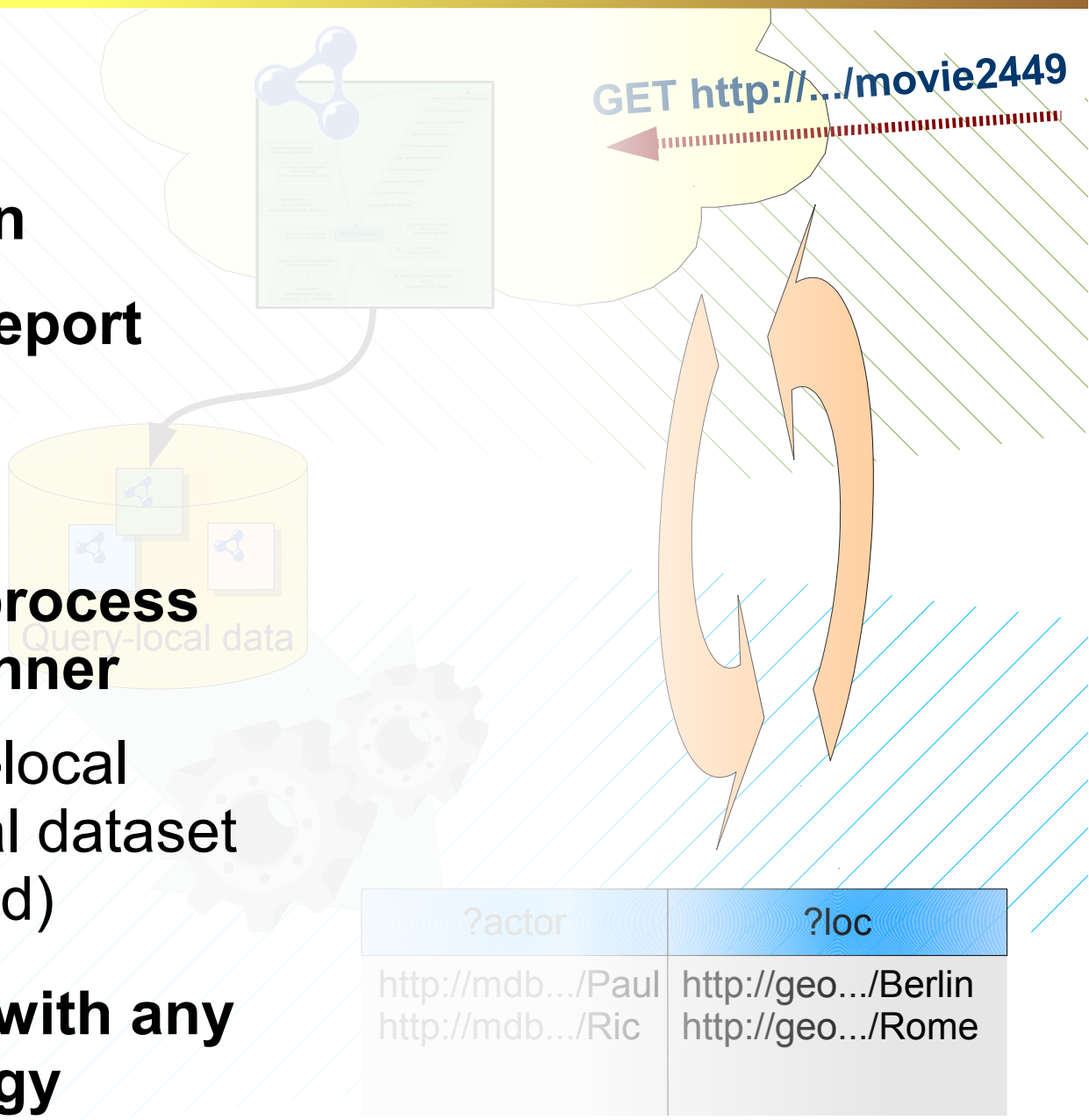
Query-local data

?actor	?loc
http://mdb.../Paul http://mdb.../Ric	http://geo.../Berlin http://geo.../Rome

Integrated Execution Approaches

… intertwine data retrieval and result construction

- **Implementations may report first solutions early**
  - For monotonic queries
- **Implementations may process data in a streaming manner**
  - May require less query-local memory (b/c query-local dataset need not be materialized)
- **Can also be combined with any source selection strategy**

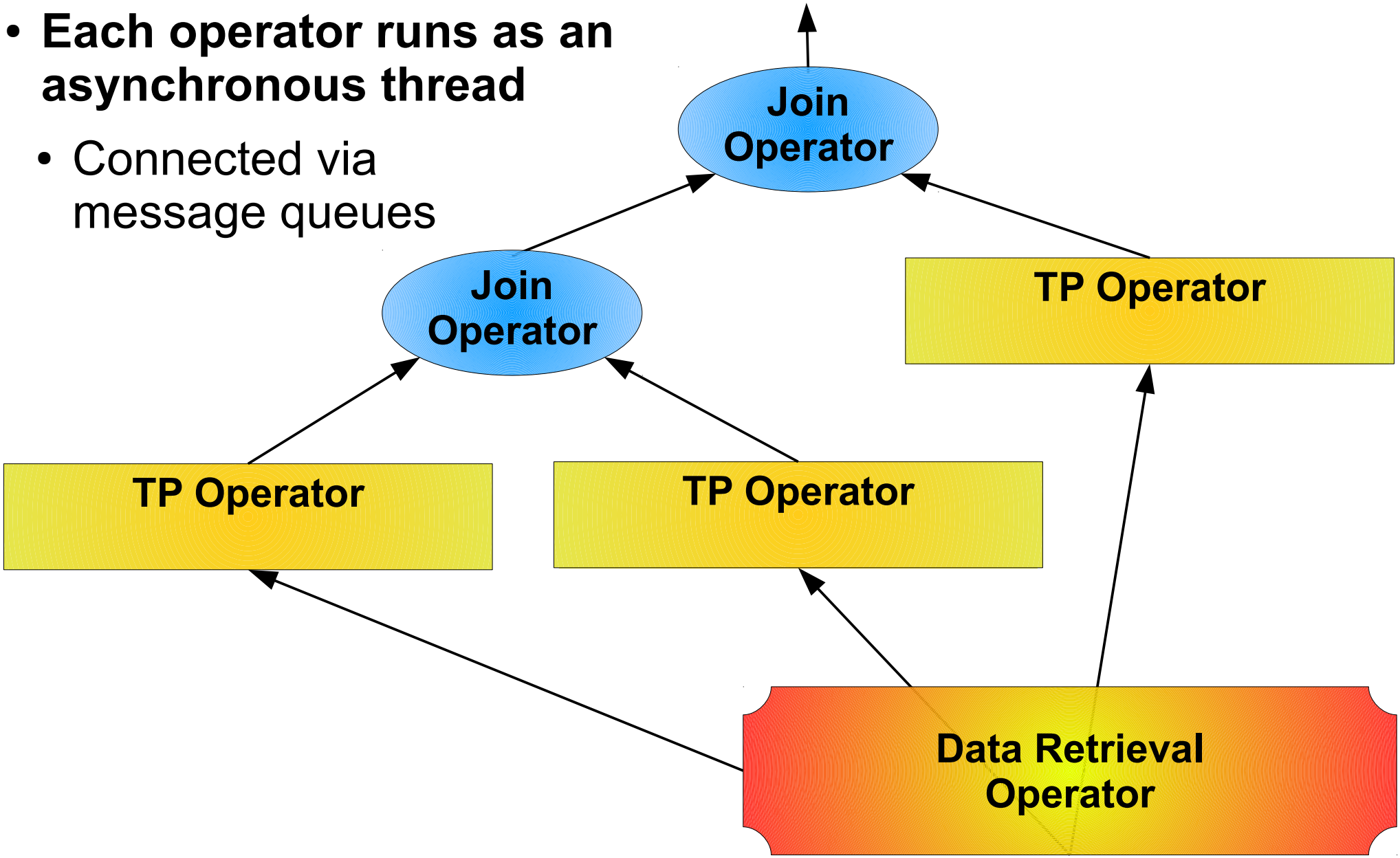


Outline

- **Separated Execution** √
- **Integrated Execution**
  - General Idea and Properties √
  - Push-Based Implementation [LT10, LT11]
  - Link Traversal Based Query Execution

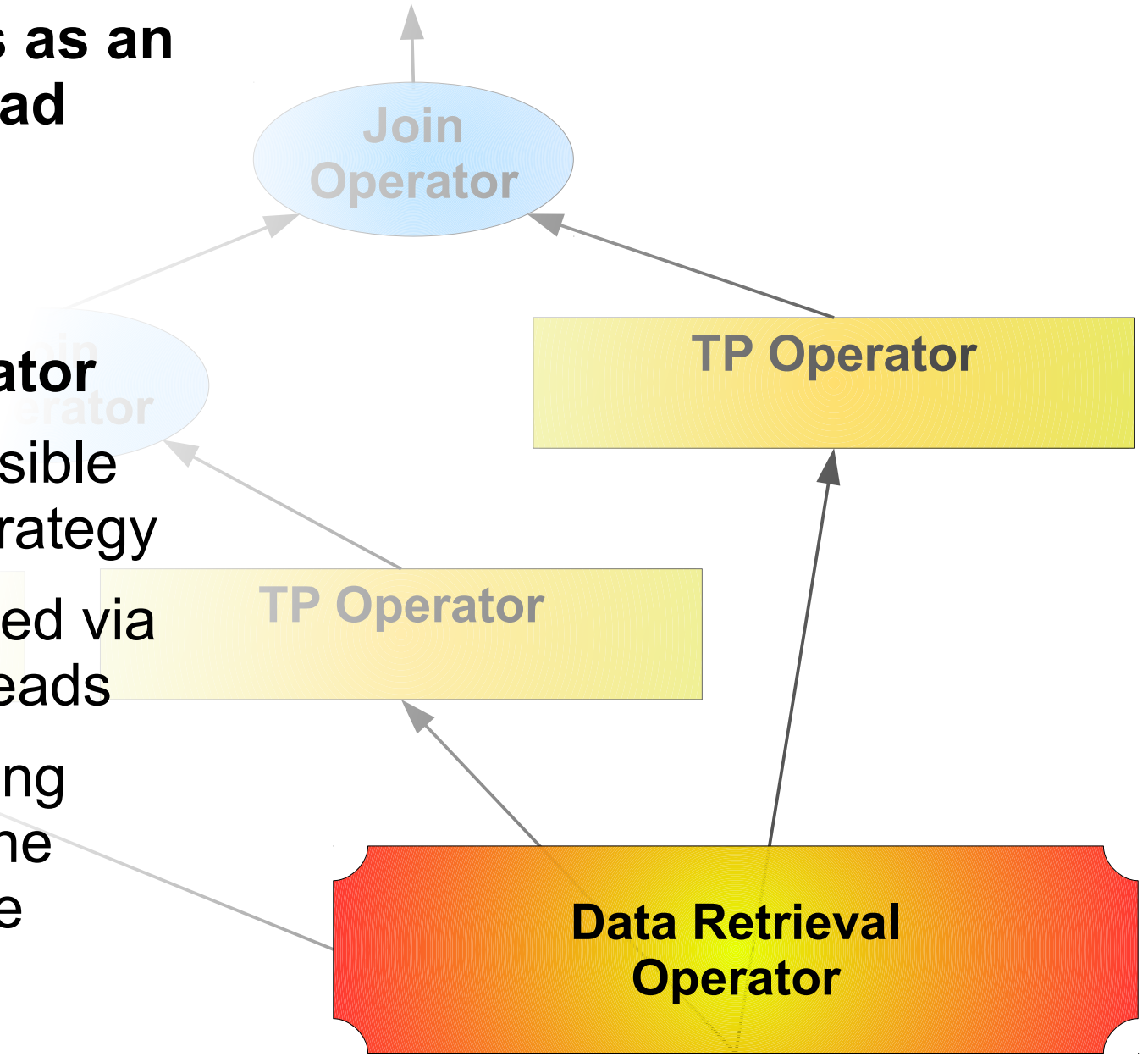
Push-Based Implementation

- **Each operator runs as an asynchronous thread**
  - Connected via message queues

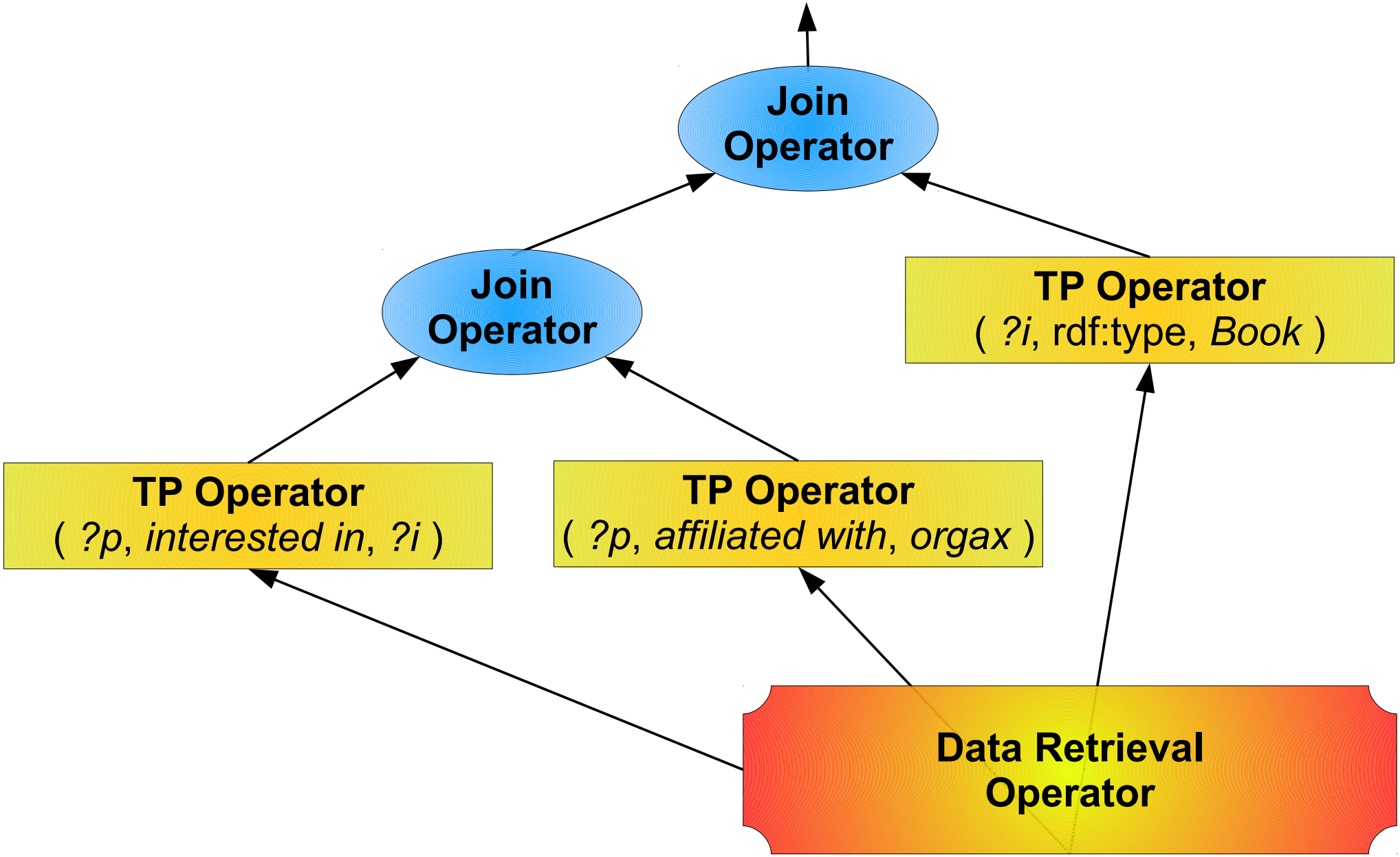


Push-Based Implementation

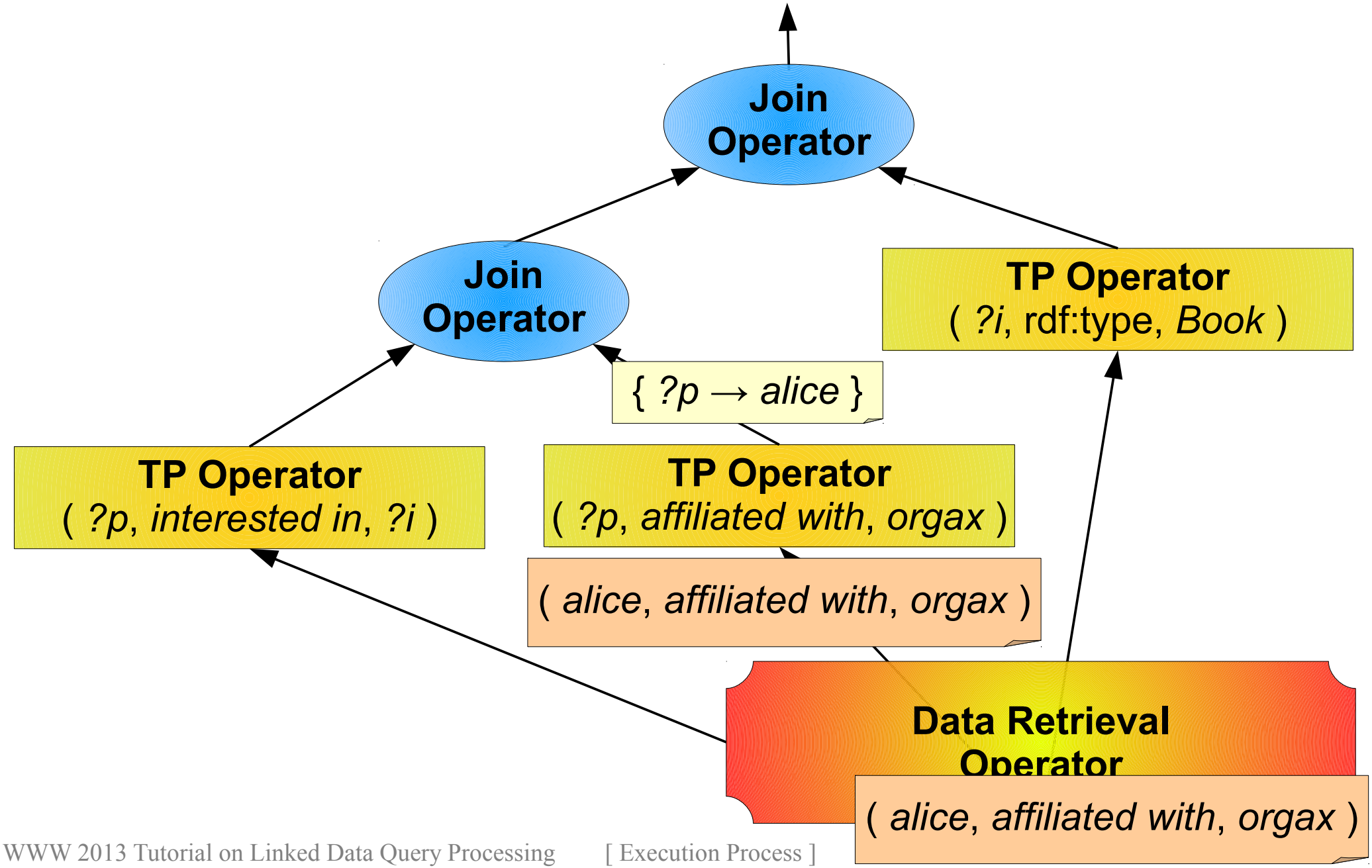
- **Each operator runs as an asynchronous thread**
  - Connected via message queues
- **Data retrieval operator**
  - May apply any possible source selection strategy
  - Usually, implemented via multiple lookup threads
  - Pushes any incoming matching triple to the corresponding triple pattern operator



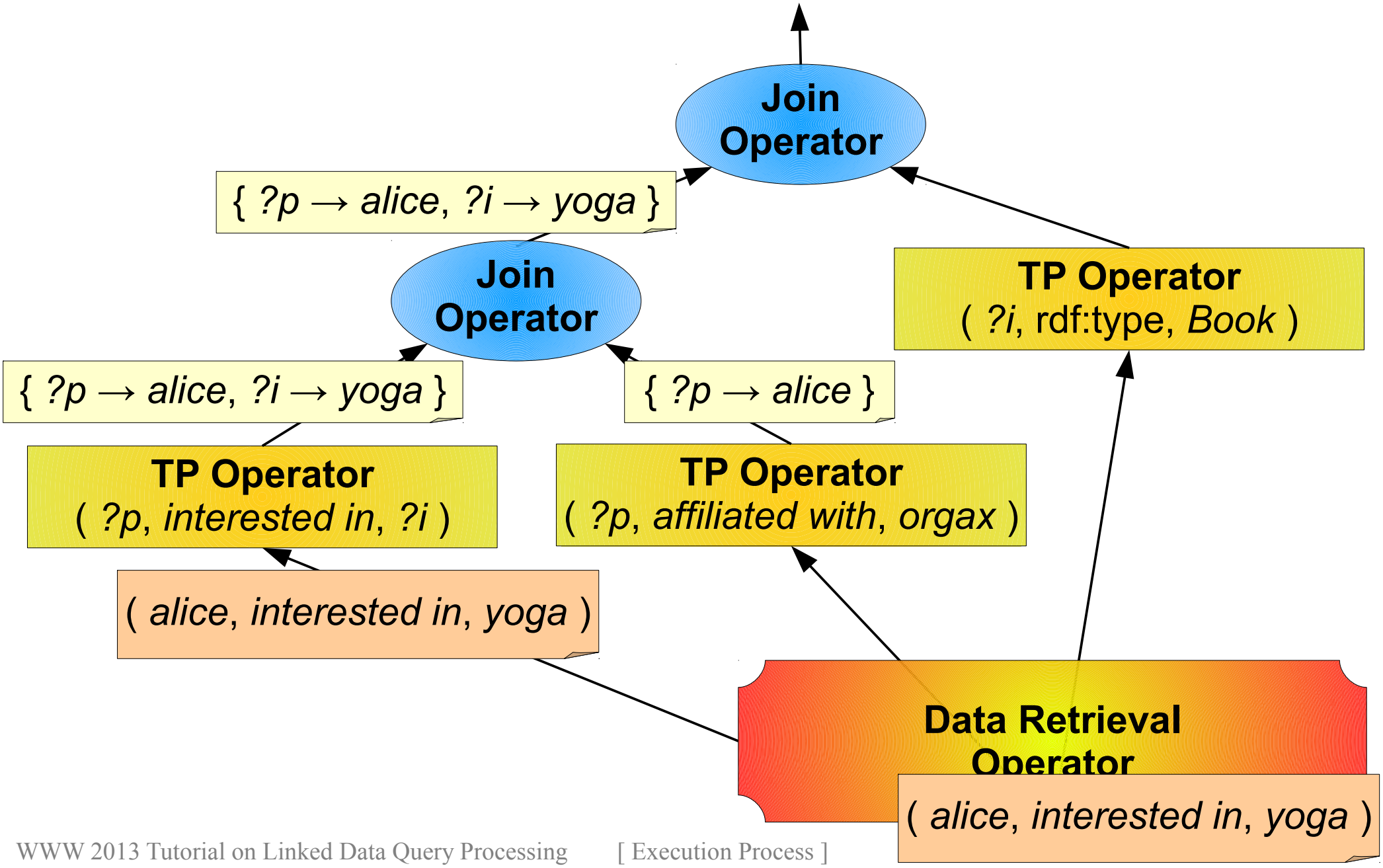
Push-Based Implementation (Example)



Push-Based Implementation (Example)

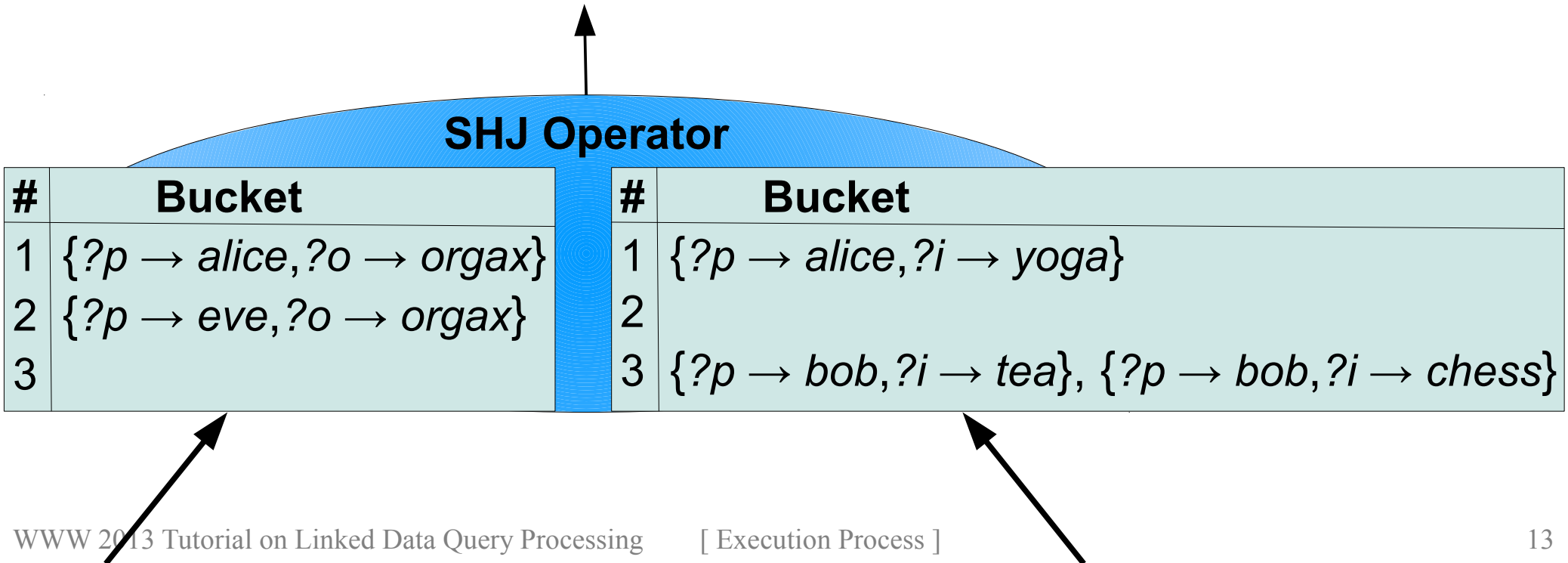


Push-Based Implementation (Example)



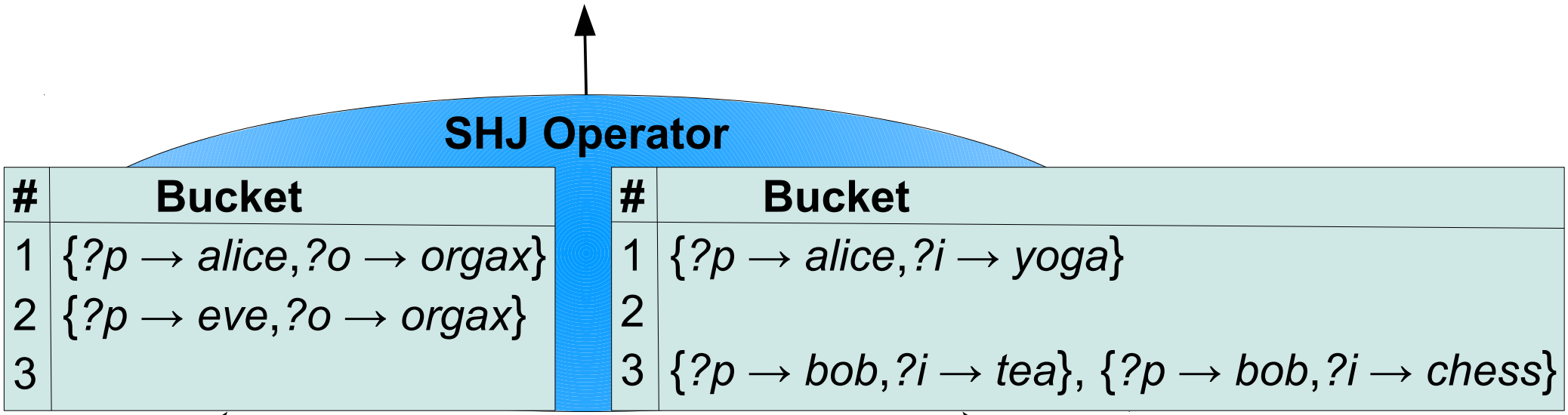
Symmetric Hash Join (SHJ)

- **Maintains a hash table for each input**
  - Same hash function for both tables
  - Each bucket contains a set of input solutions



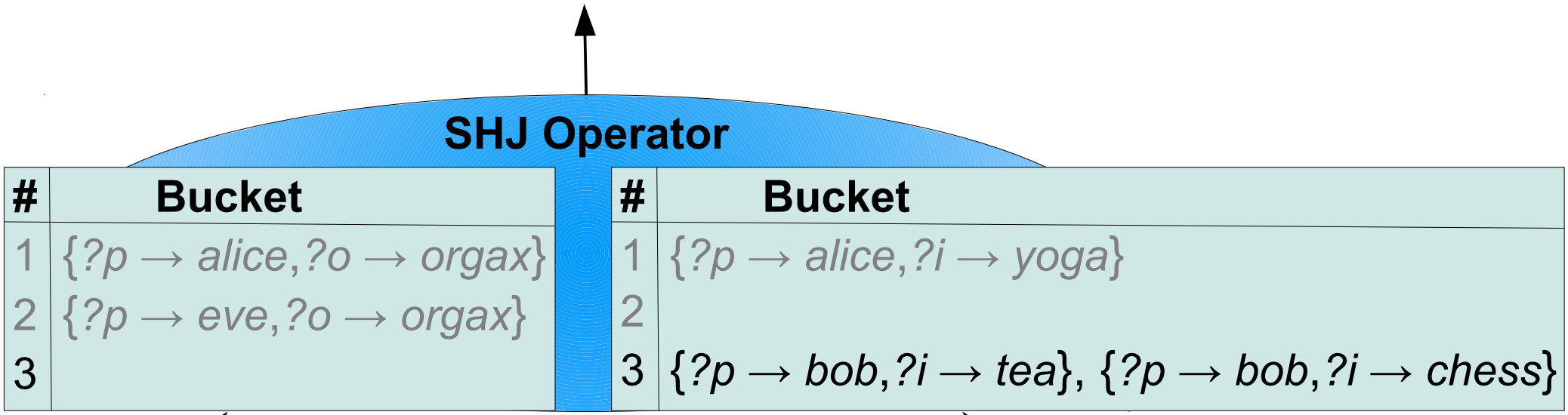
How SHJ Processes Incoming Solution

- **Hash the value that μ_{input} binds to the join variable**



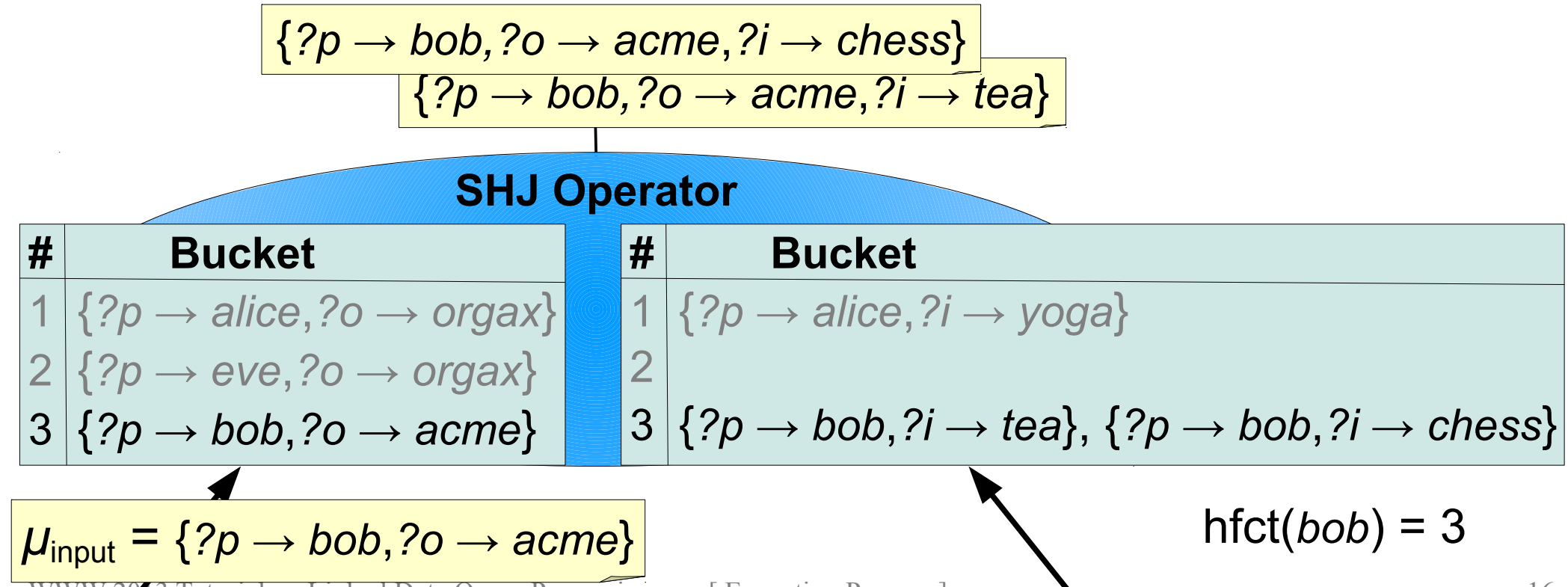
How SHJ Processes Incoming Solution

- **Hash the value that μ_{input} binds to the join variable**
- **Insert μ_{input} into the corresponding input hash table**



How SHJ Processes Incoming Solution

- **Hash the value that μ_{input} binds to the join variable**
- **Insert μ_{input} into the corresponding input hash table**
- **Probe into the other hash table to find join candidates**
- **Merge μ_{input} with each join candidate, push results to output**



Outline

- **Separated Execution** √
- **Integrated Execution**
  - General Idea and Properties √
  - Push-Based Implementation [LT10, LT11] √
  - Link Traversal Based Query Execution

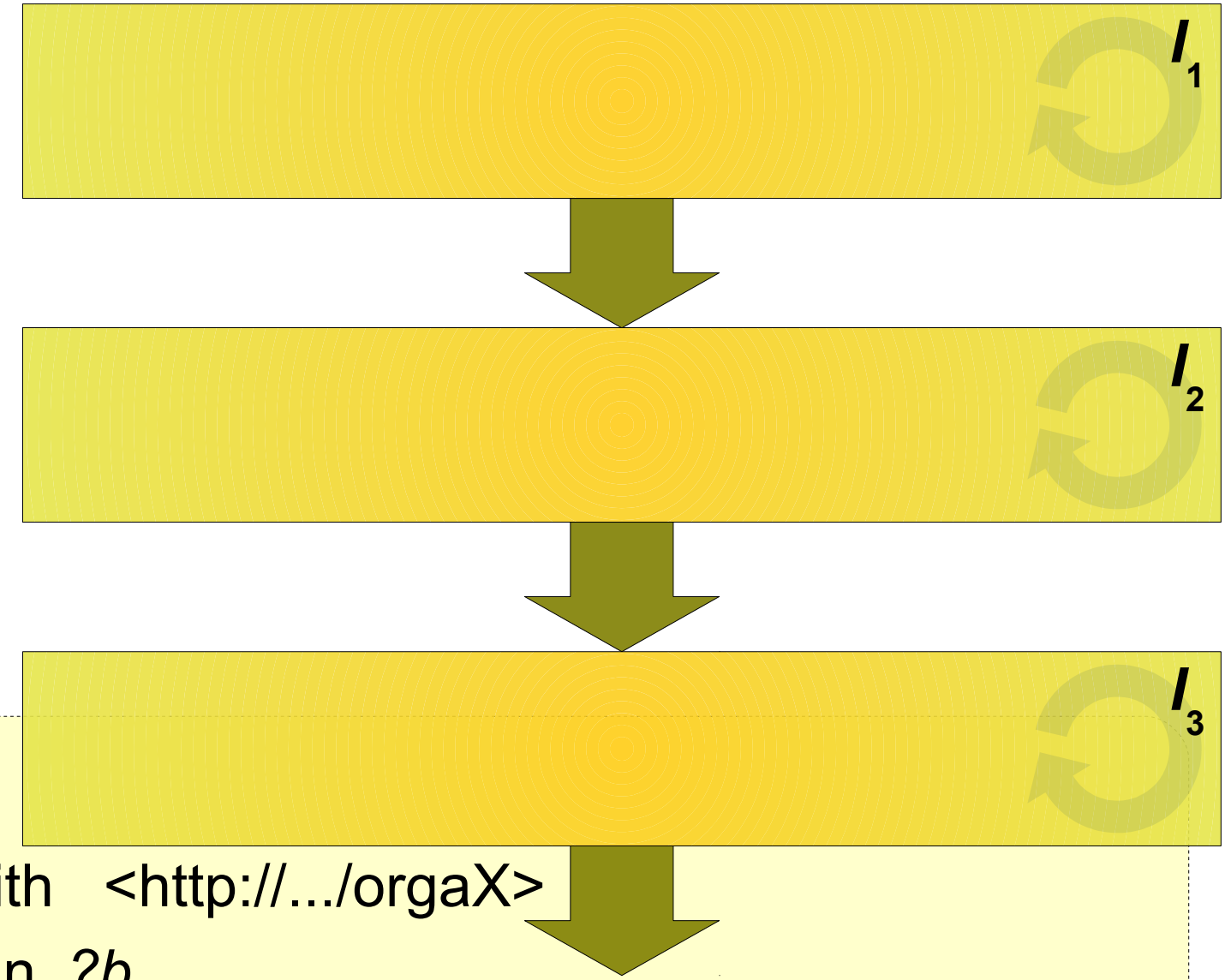
Link Traversal Based Query Execution

… is the combination of integrated execution and live exploration

(i.e., the push-based approach can be used as an implementation of LTBQE)

An even closer combination presents the following pull-based implementation approach ...

Pipeline of Link Traversing Iterators

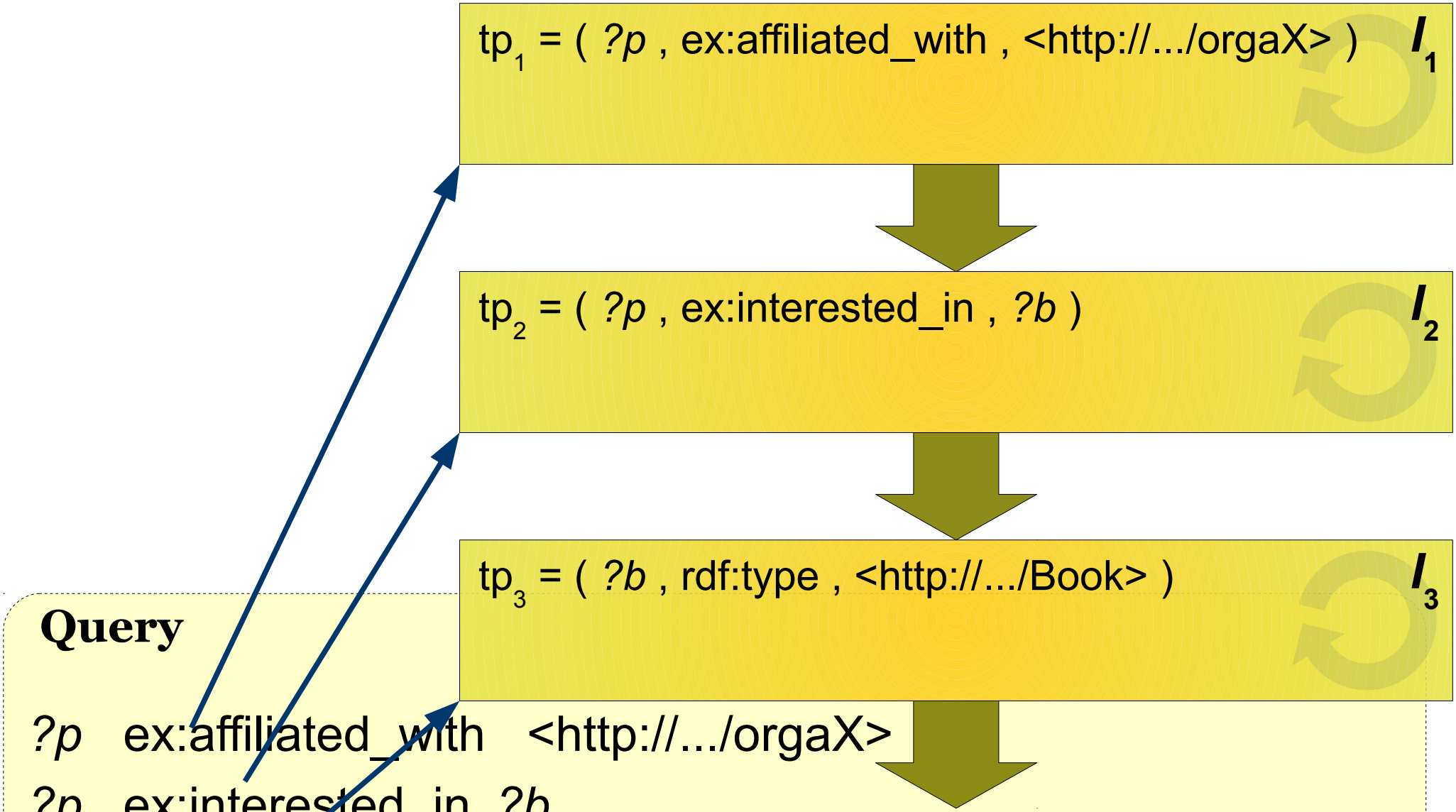


Query

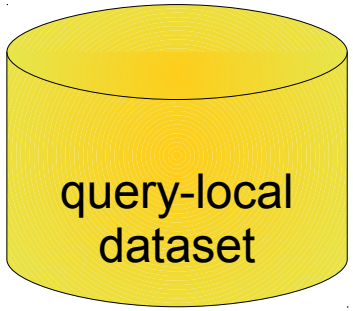
?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Seed: <http://.../orgaX>

Pipeline of Link Traversing Iterators



Pipeline of Link Traversing Iterators



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

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

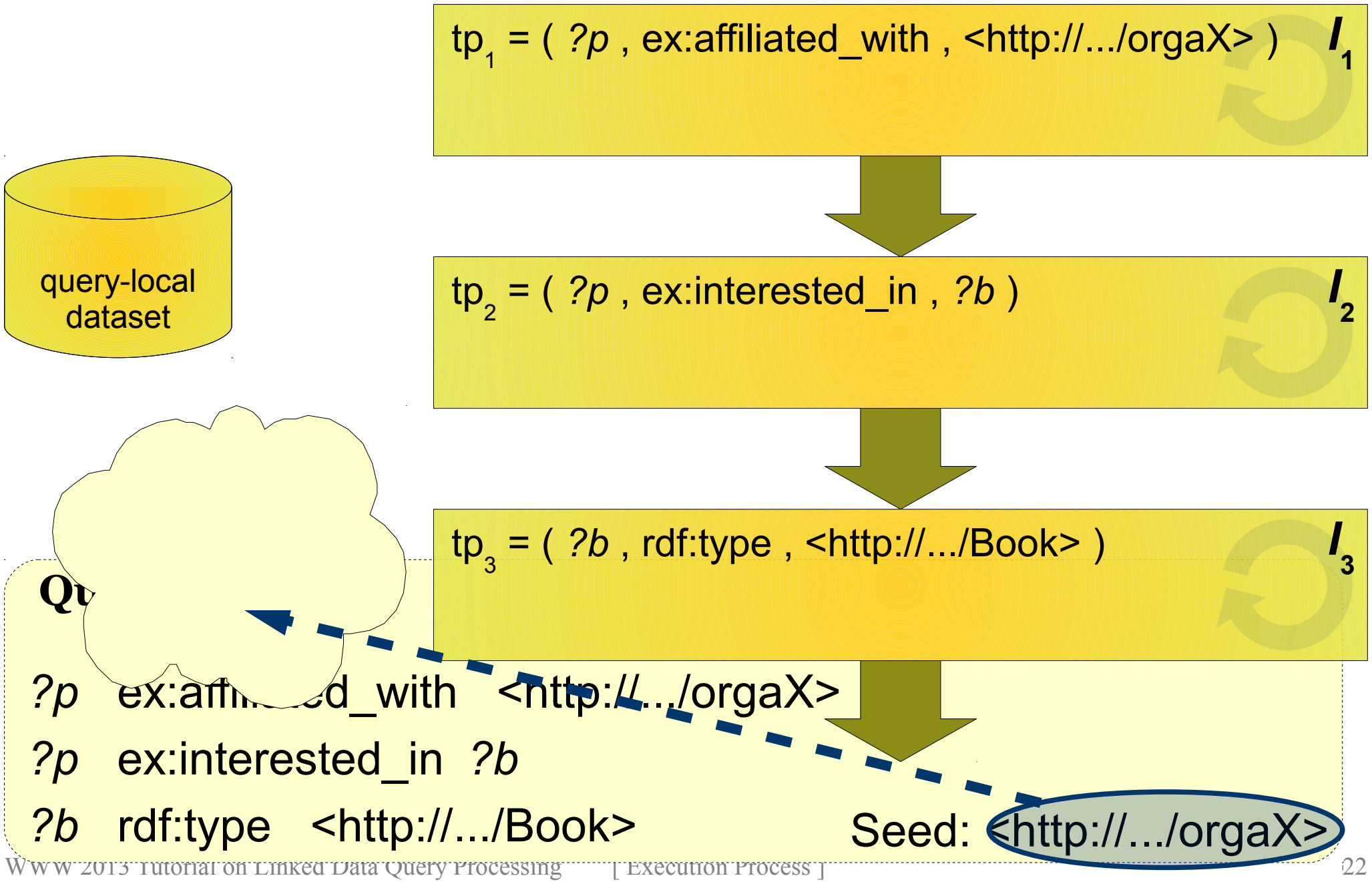
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Query

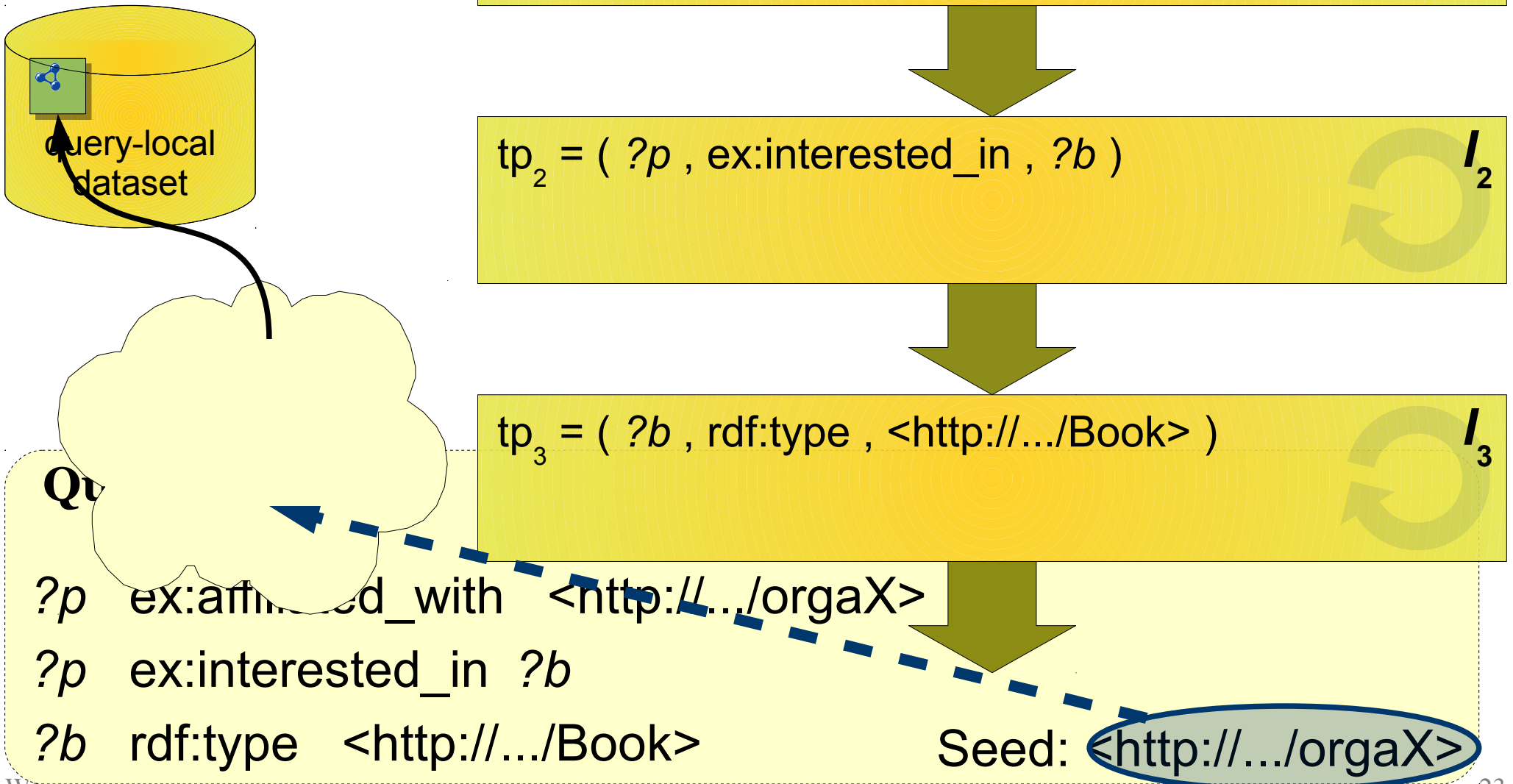
?p ex:affiliated_with <http://.../orgaX>
?p ex:interested_in *?b*
?b rdf:type <http://.../Book>

Seed: <http://.../orgaX>

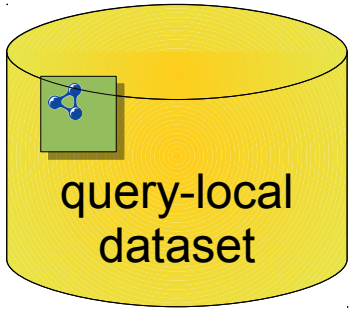
Pipeline of Link Traversing Iterators

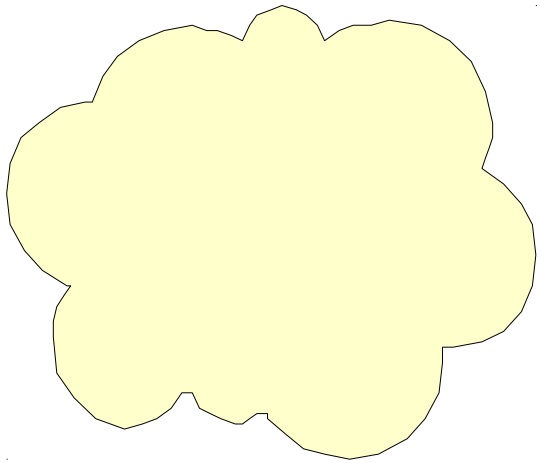


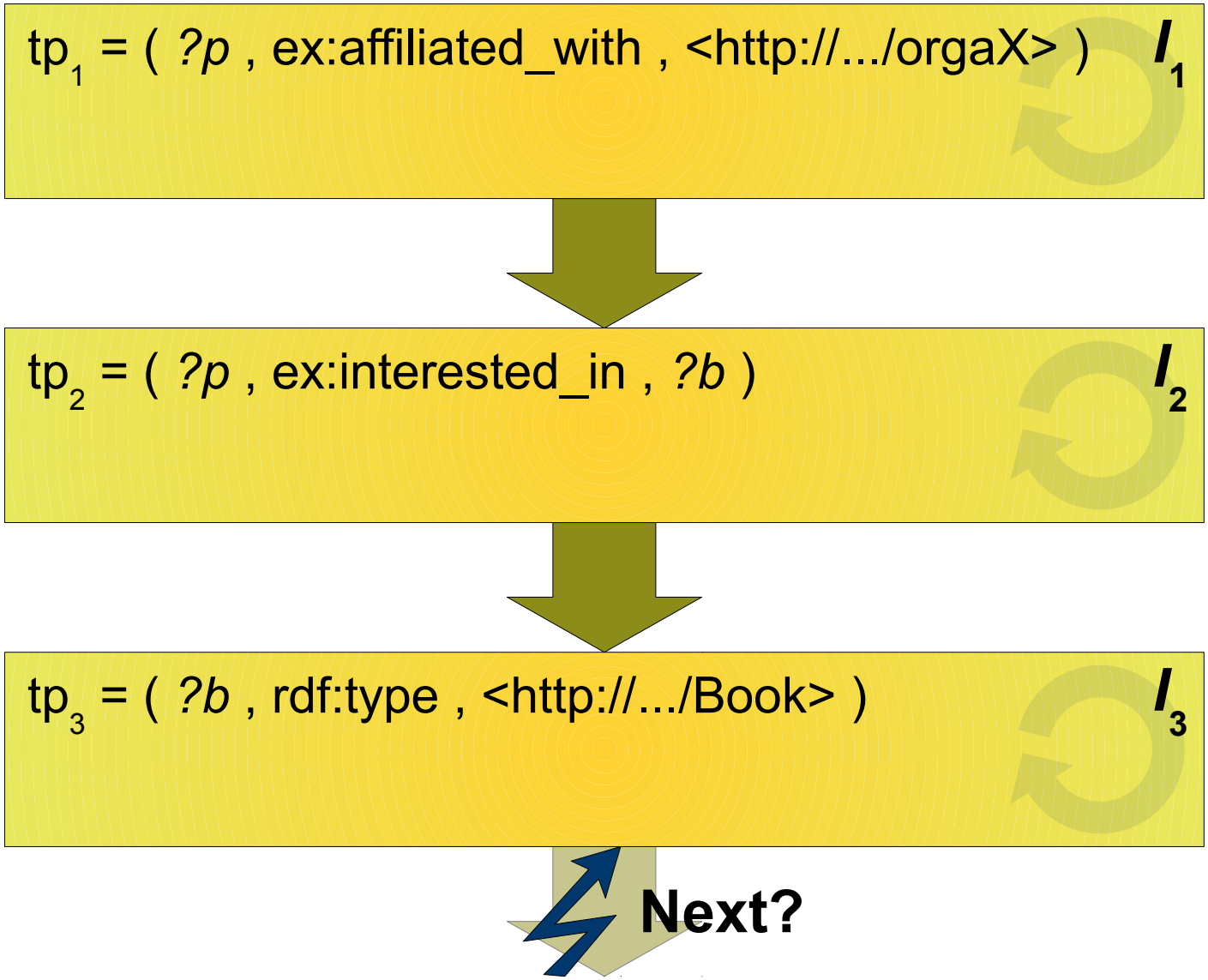
Pipeline of Link Traversing Iterators



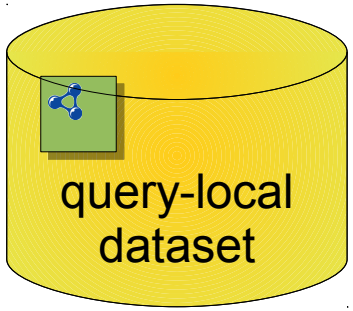
Pipeline of Link Traversing Iterators

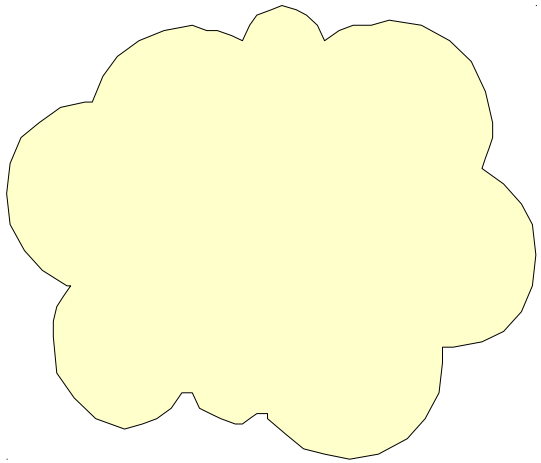






Pipeline of Link Traversing Iterators





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

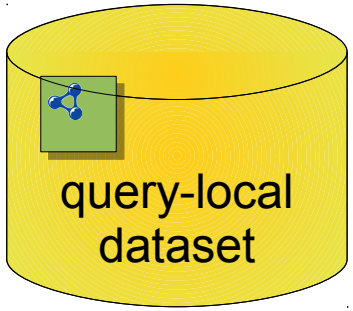
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

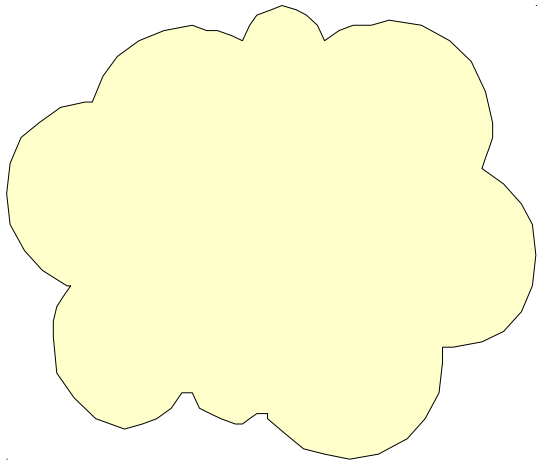
Next?

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

Next?

Pipeline of Link Traversing Iterators





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

Next?

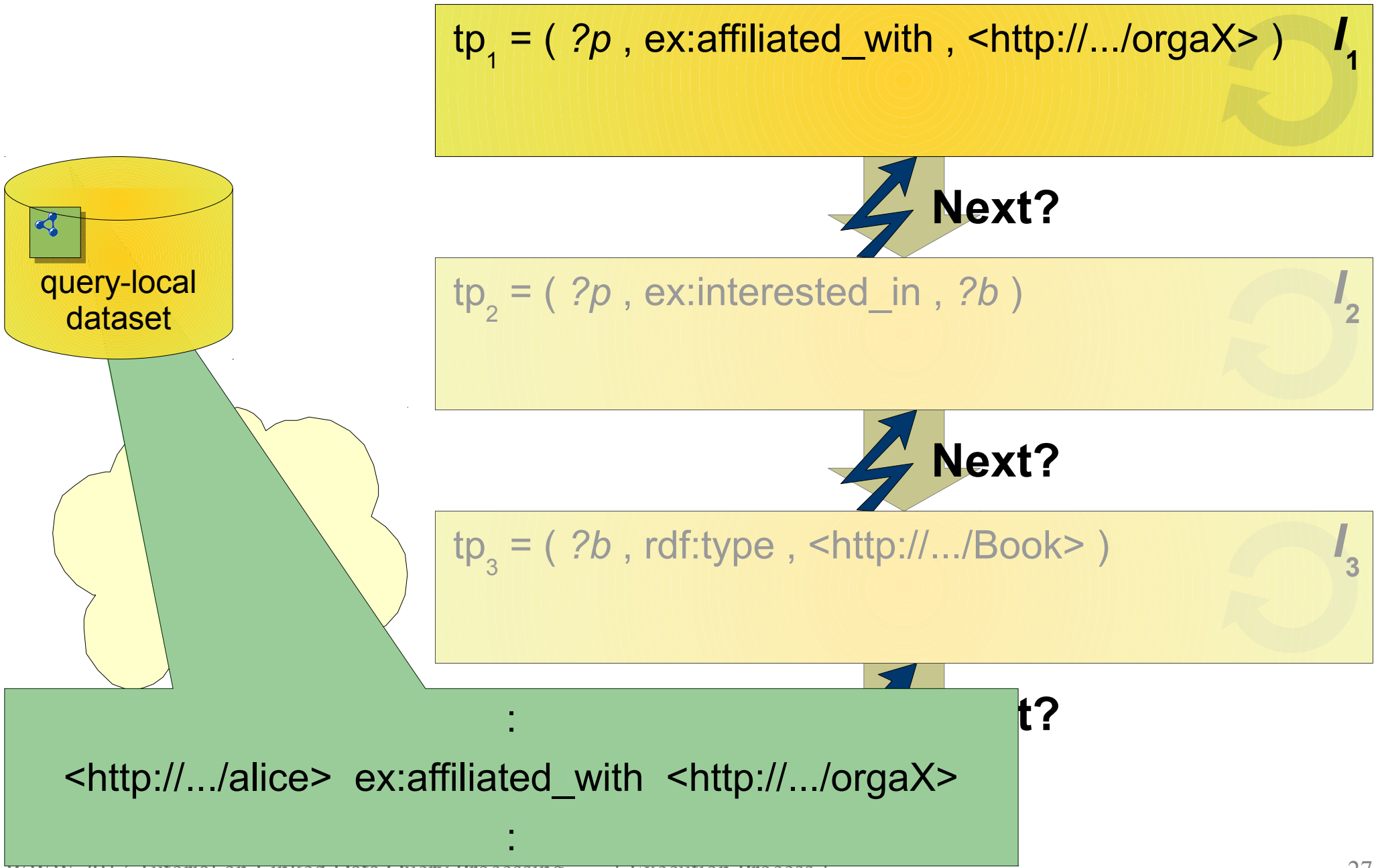
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

Next?

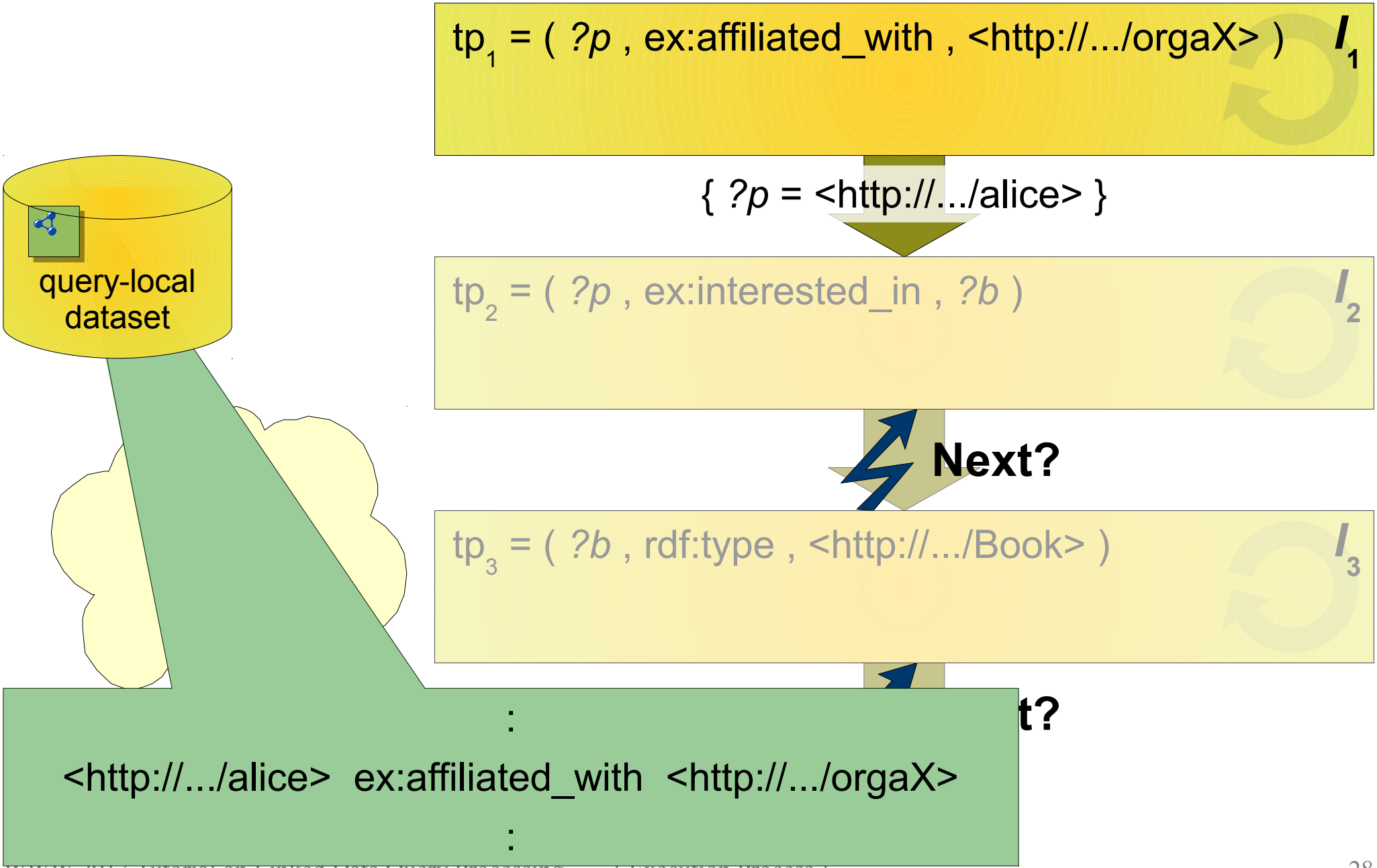
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

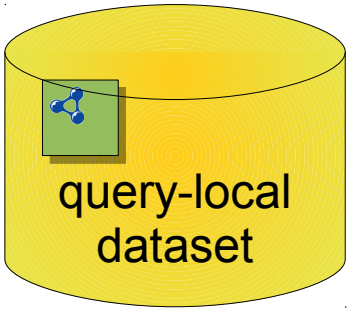
Pipeline of Link Traversing Iterators

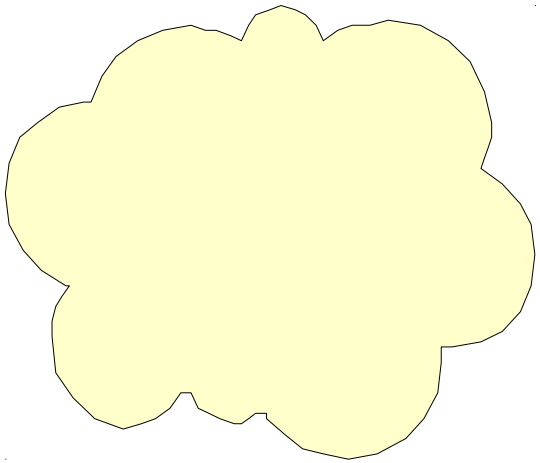


Pipeline of Link Traversing Iterators



Pipeline of Link Traversing Iterators





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

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

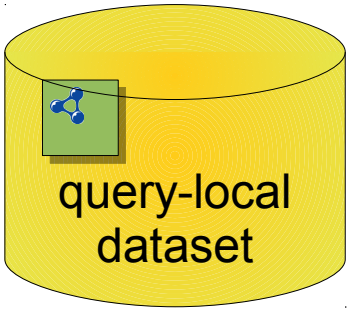
tp_2 = (*?p* , ex:interested_in , *?b*) I_2

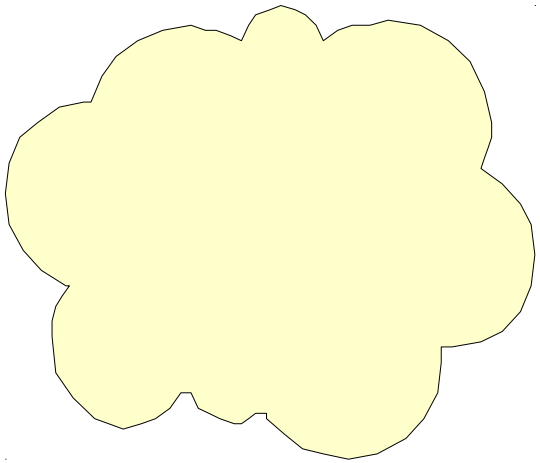
Next?

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

Next?

Pipeline of Link Traversing Iterators





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

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

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

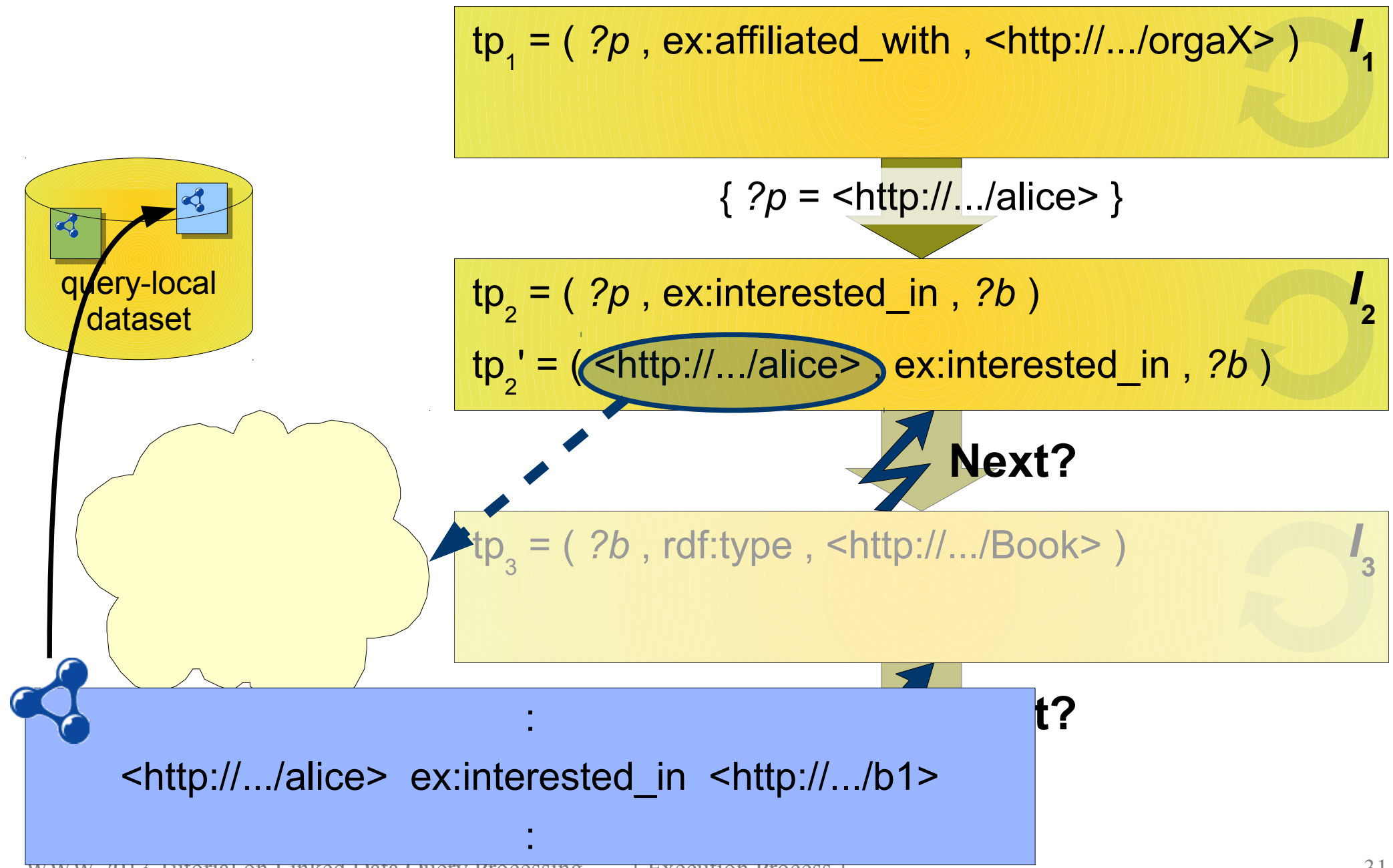
tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

Next?

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

Next?

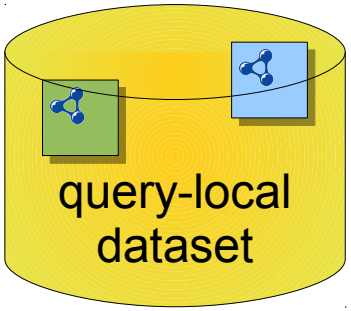
Pipeline of Link Traversing Iterators

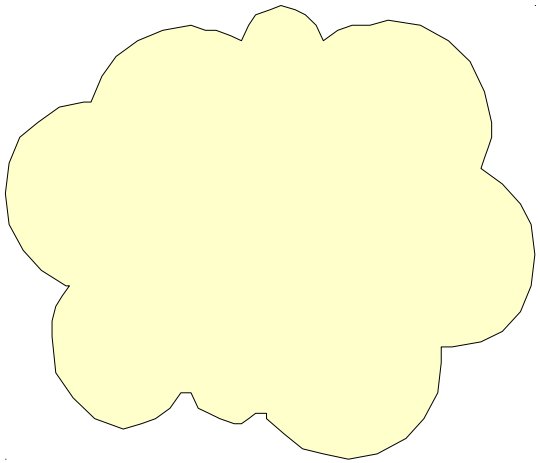


Pipeline of Link Traversing Iterators

Pipeline of Link Traversing Iterators

Pipeline of Link Traversing Iterators





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

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

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

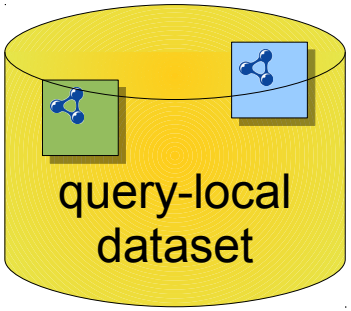
tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

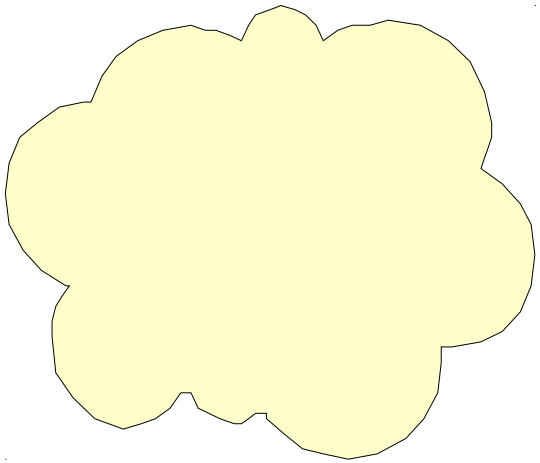
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

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

Next?

Pipeline of Link Traversing Iterators





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*)

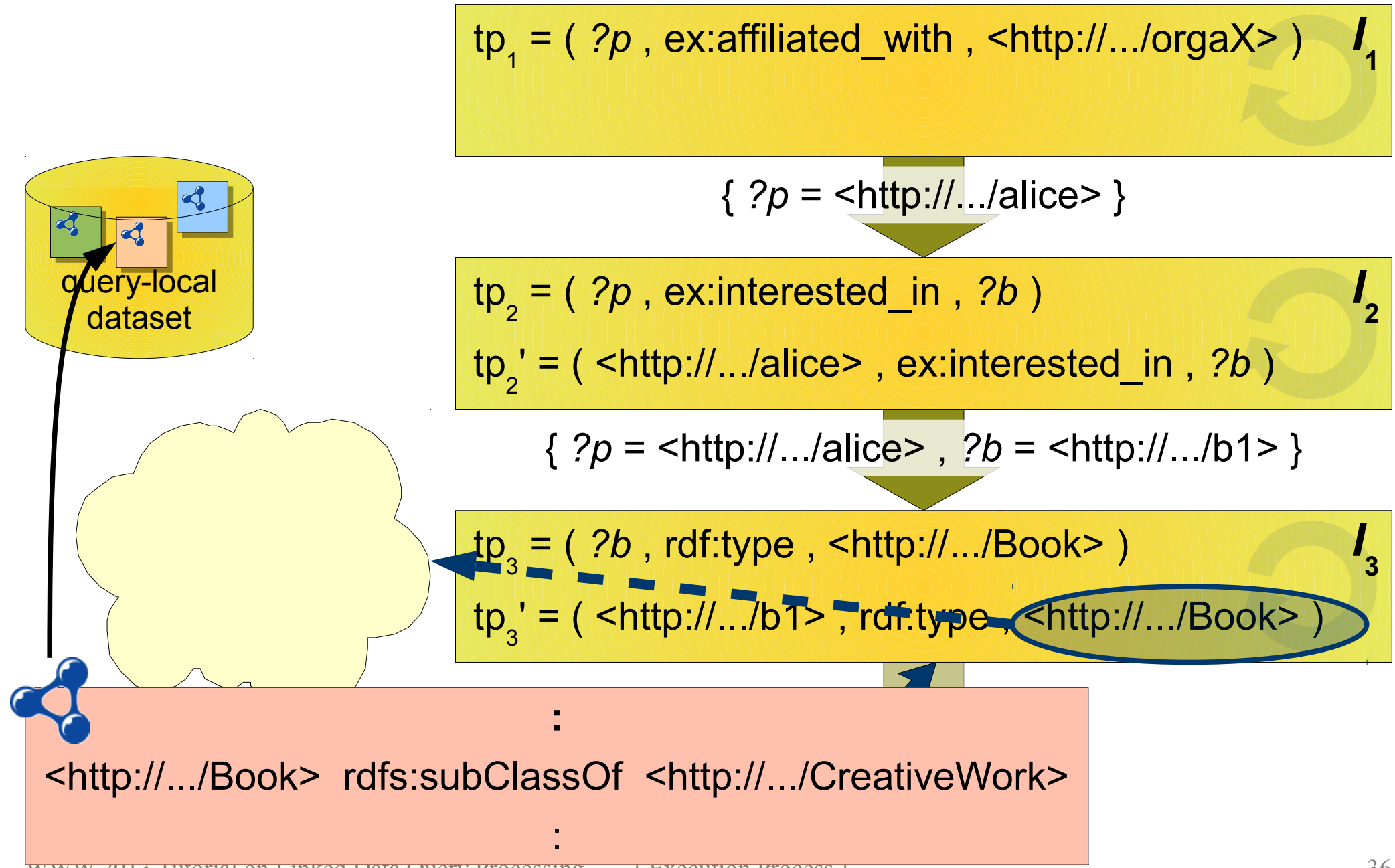
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

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

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

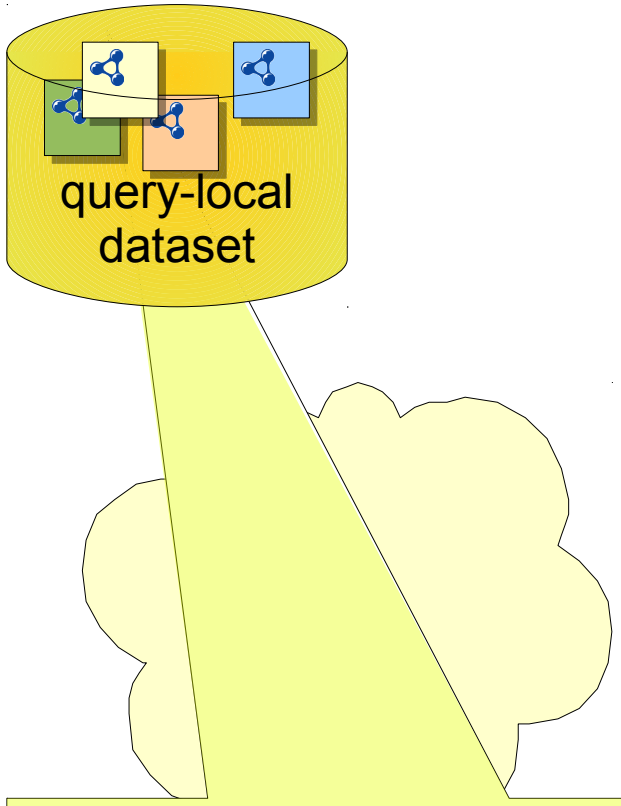
Next?

Pipeline of Link Traversing Iterators



Pipeline of Link Traversing Iterators

Pipeline of Link Traversing Iterators



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>)

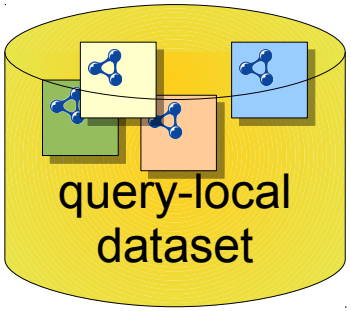
Next?

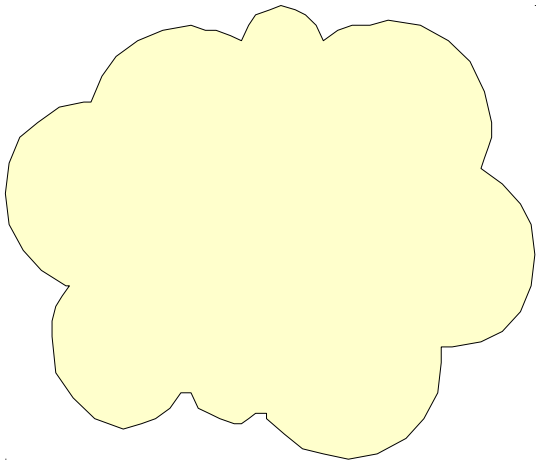
:

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

:

Pipeline of Link Traversing Iterators





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*)

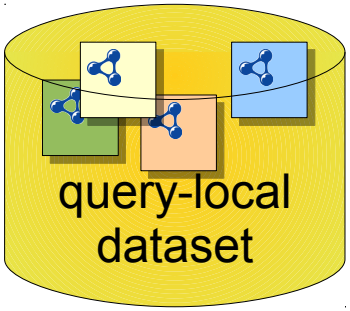
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

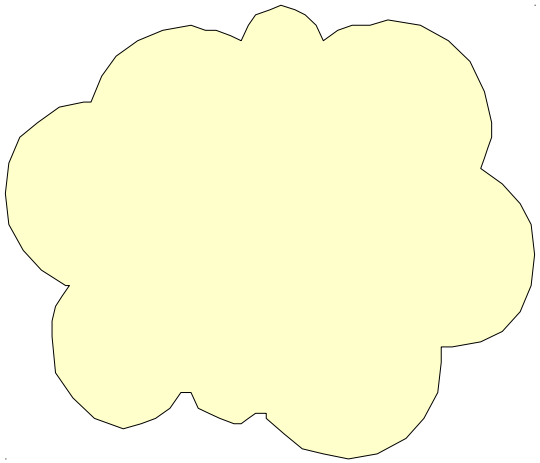
tp_3 = (*?b* , rdf:type , <http://.../Book>) **I_3**

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

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

Pipeline of Link Traversing Iterators





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*)

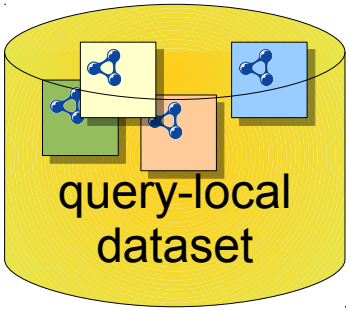
{ *?p* = <http://.../alice> , *?b* = <http://.../b1> }

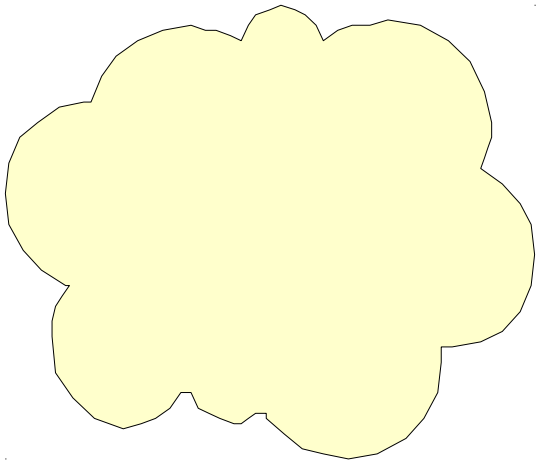
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

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

Next?

Pipeline of Link Traversing Iterators





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

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

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

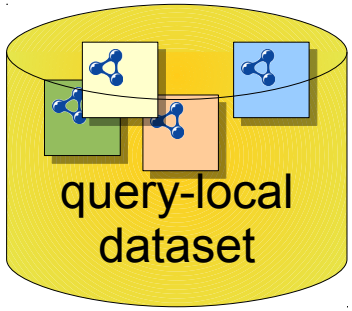
tp_2' = (<http://.../alice> , ex:interested_in , *?b*)

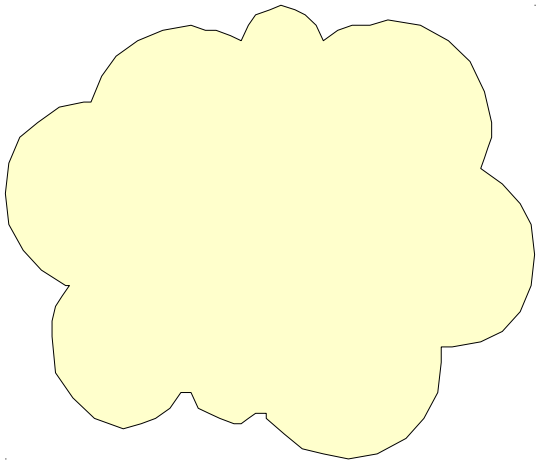
Next?

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

Next?

Pipeline of Link Traversing Iterators





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?

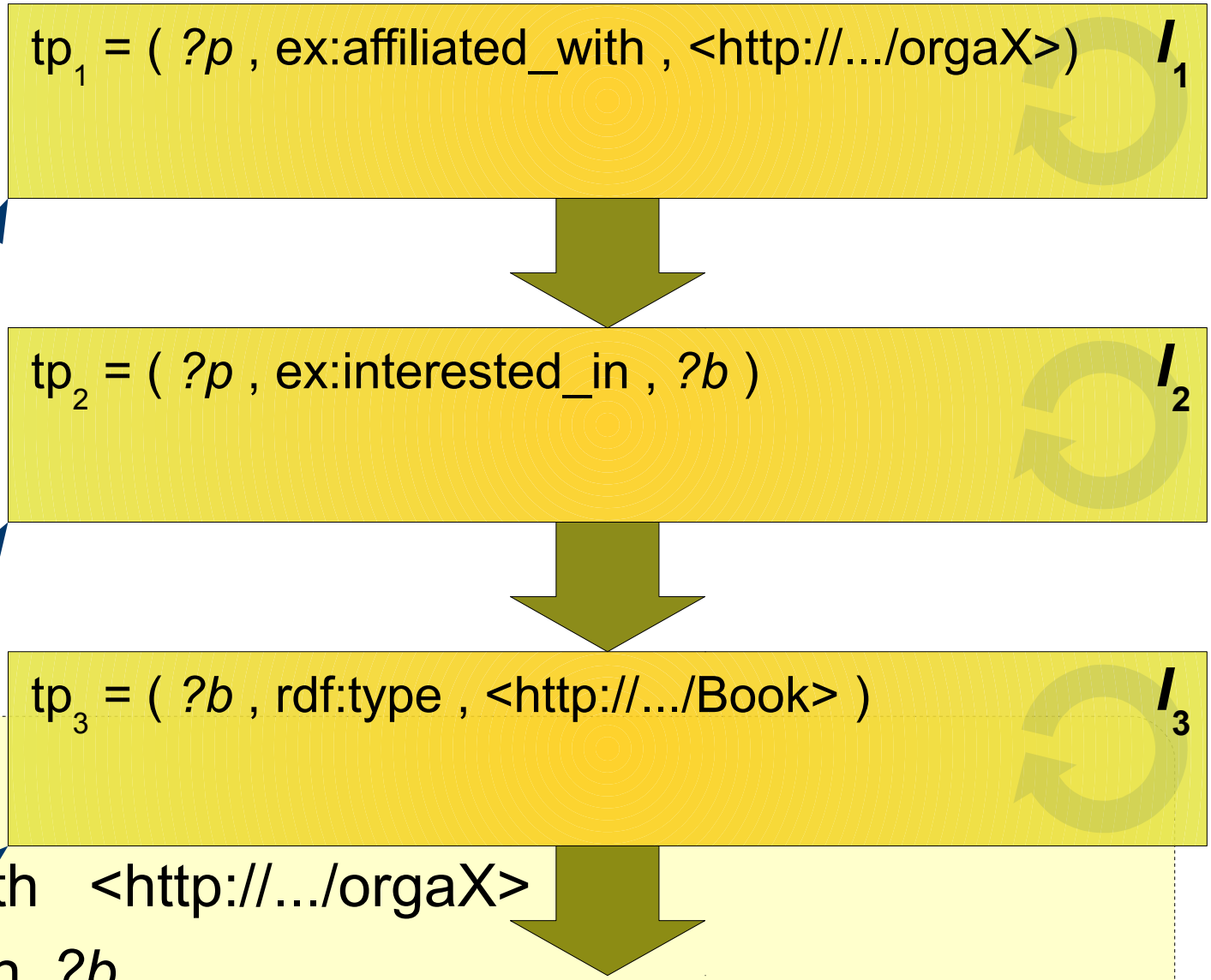
tp_3 = (*?b* , rdf:type , <http://.../Book>) I_3

Next?

Pipeline of Link Traversing Iterators

- **Deeply intertwines integrated execution + live exploration**
  - Data retrieval is a side-effect of calling link traversing iterators
  - No need for a separate data retrieval operator
- **Makes adding LTBQE to existing SPARQL engines easy**
  - Because those usually use iterators for execution
- **Caveat: does not achieve the full flexibility that is possible with LTBQE**
  - Iterators evaluate the triple patterns in a fixed order
  - Iterators discard each input solution after using it

Alternative Execution Order

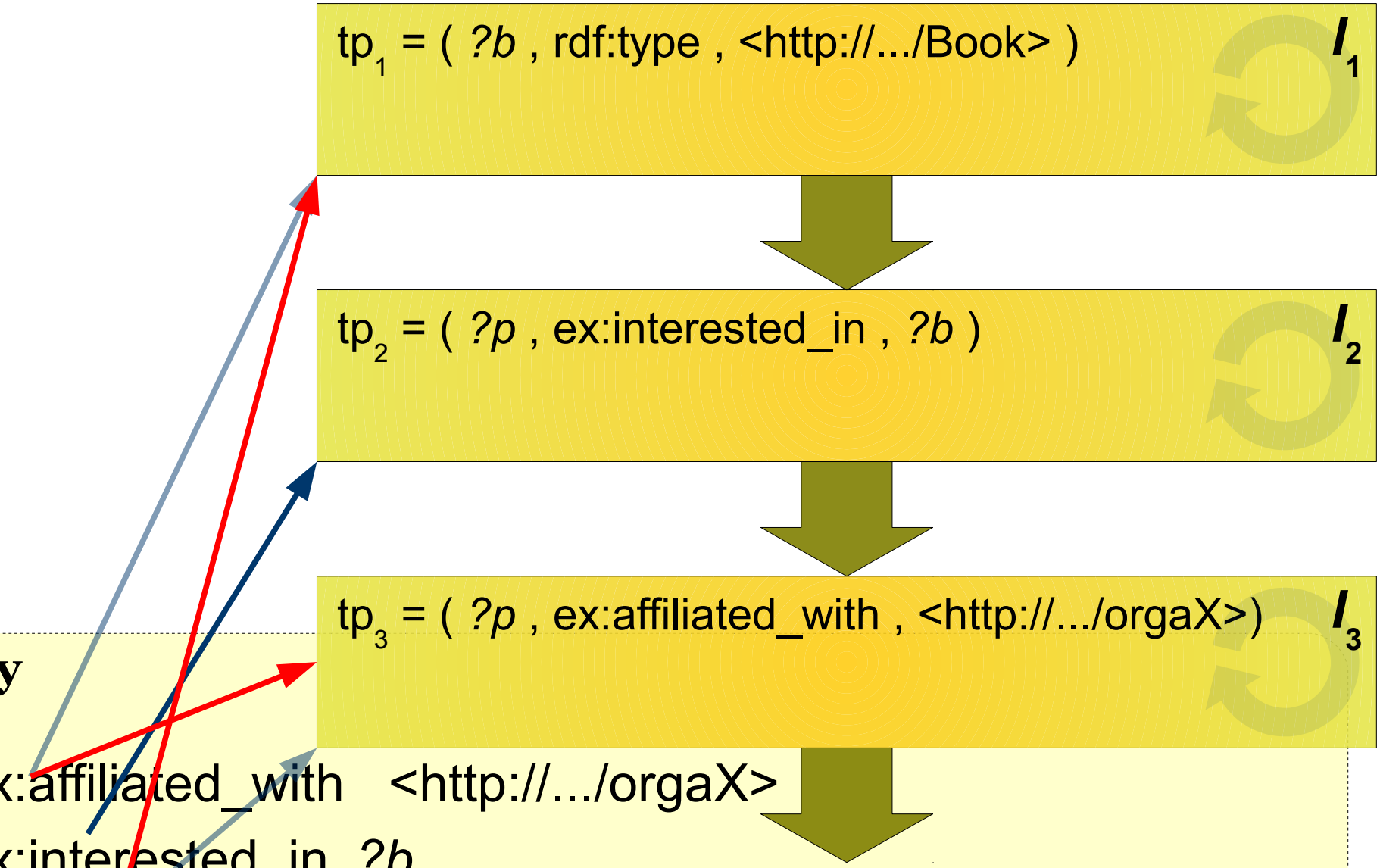


Query

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

Seed: <http://.../orgaX>

Alternative Execution Order

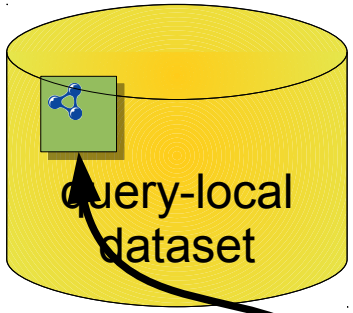


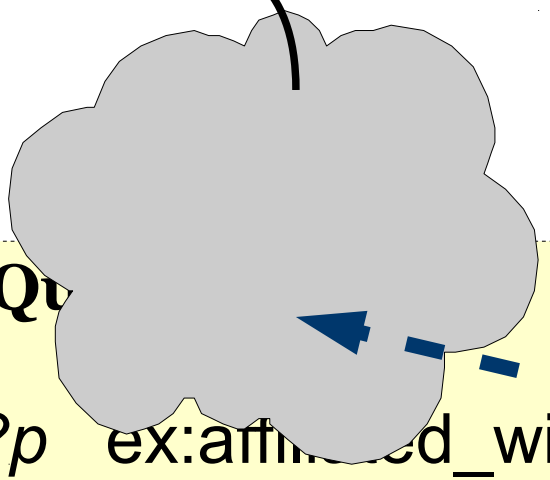
Query

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

Seed: <http://.../orgaX>

Iterator Based Execution





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

Qu

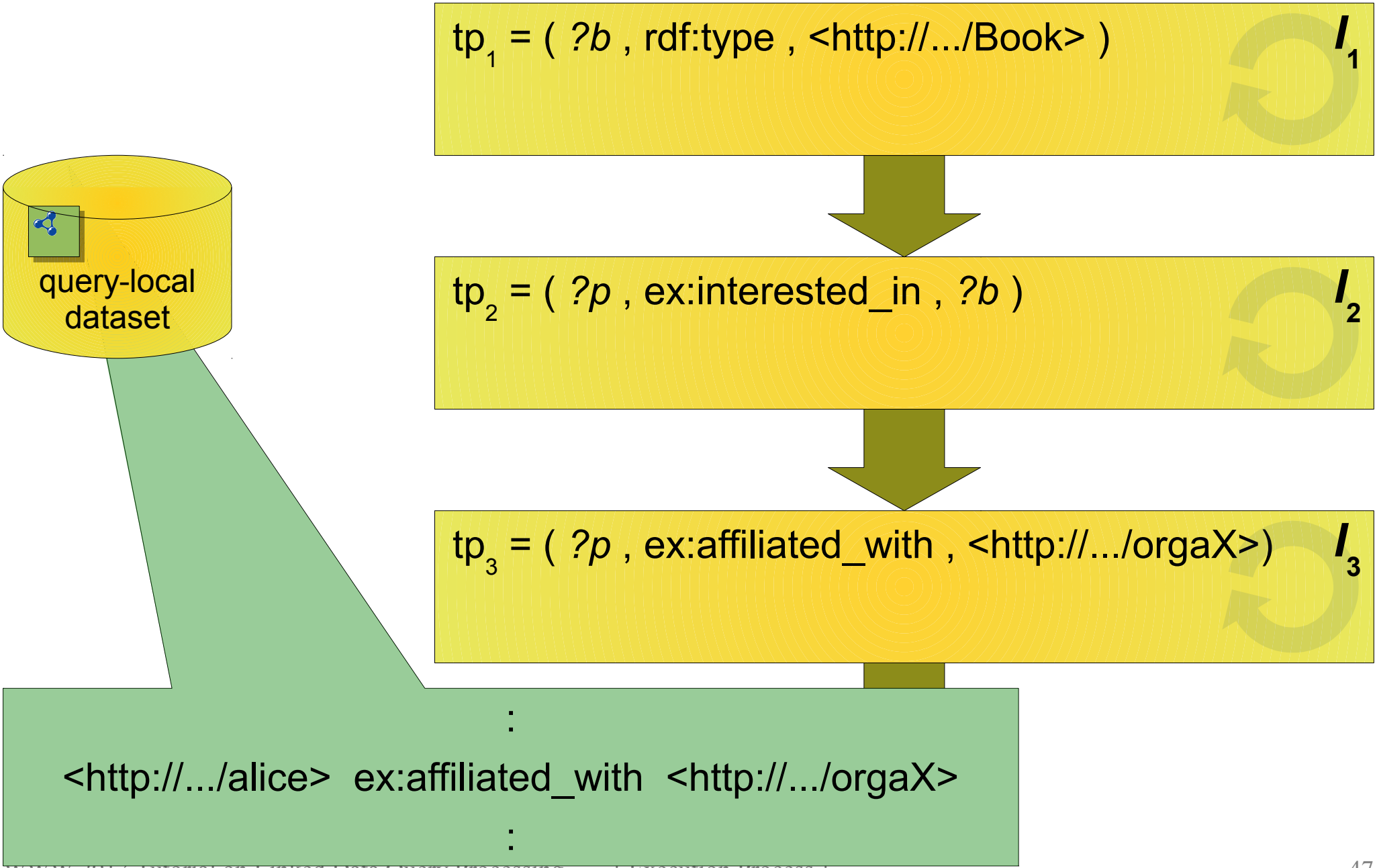
?p ex:affiliated_with <http://.../orgaX>

?p ex:interested_in *?b*

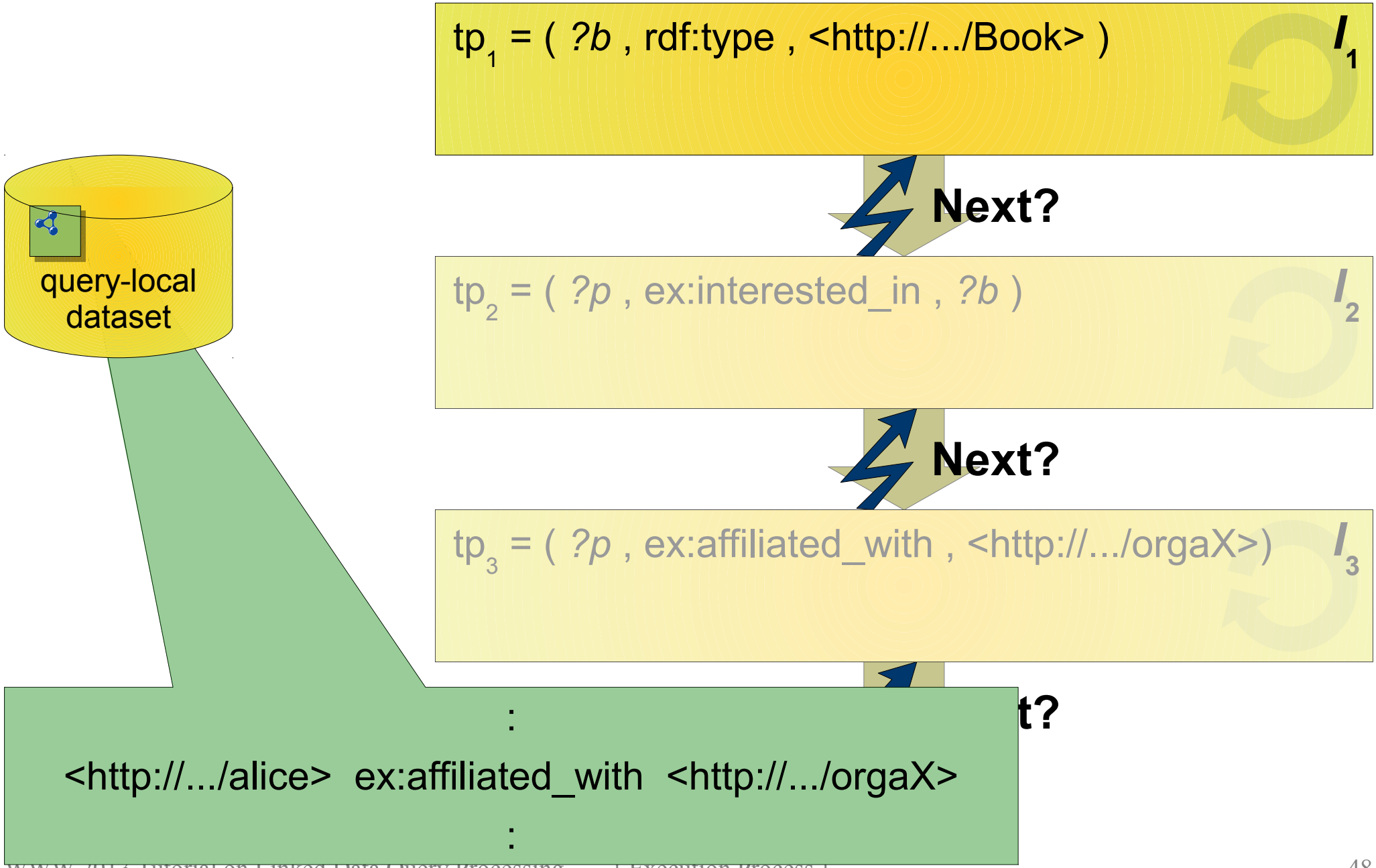
?b rdf:type <http://.../Book>

Seed: <http://.../orgaX>

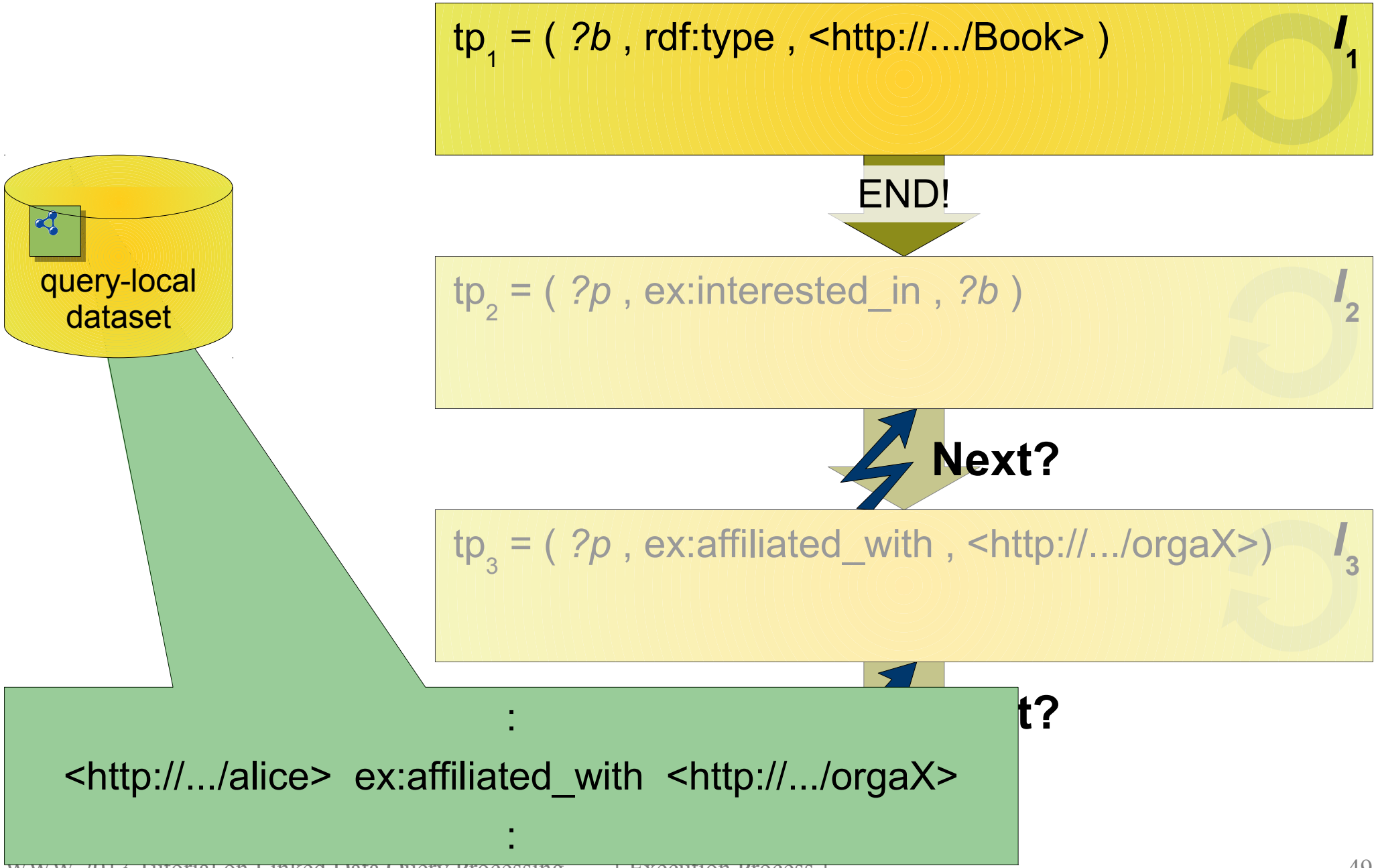
Alternative Execution Order



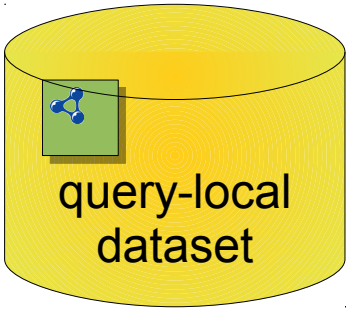
Alternative Execution Order



Alternative Execution Order



Alternative Execution Order



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!

Computed query result may depend on the order of triple patterns

= logical query execution plan

Outline

- **Separated Execution** √
- **Integrated Execution**
  - General Idea and Properties √
  - Push-Based Implementation [LT10, LT11] √
  - Link Traversal Based Query Execution √

Next part: 5. Query Planning and Optimization ...

These slides have been created by
Olaf Hartig
for the
WWW 2013 tutorial on
Link Data Query Processing

Tutorial Website: http://db.uwaterloo.ca/LDQTut2013/