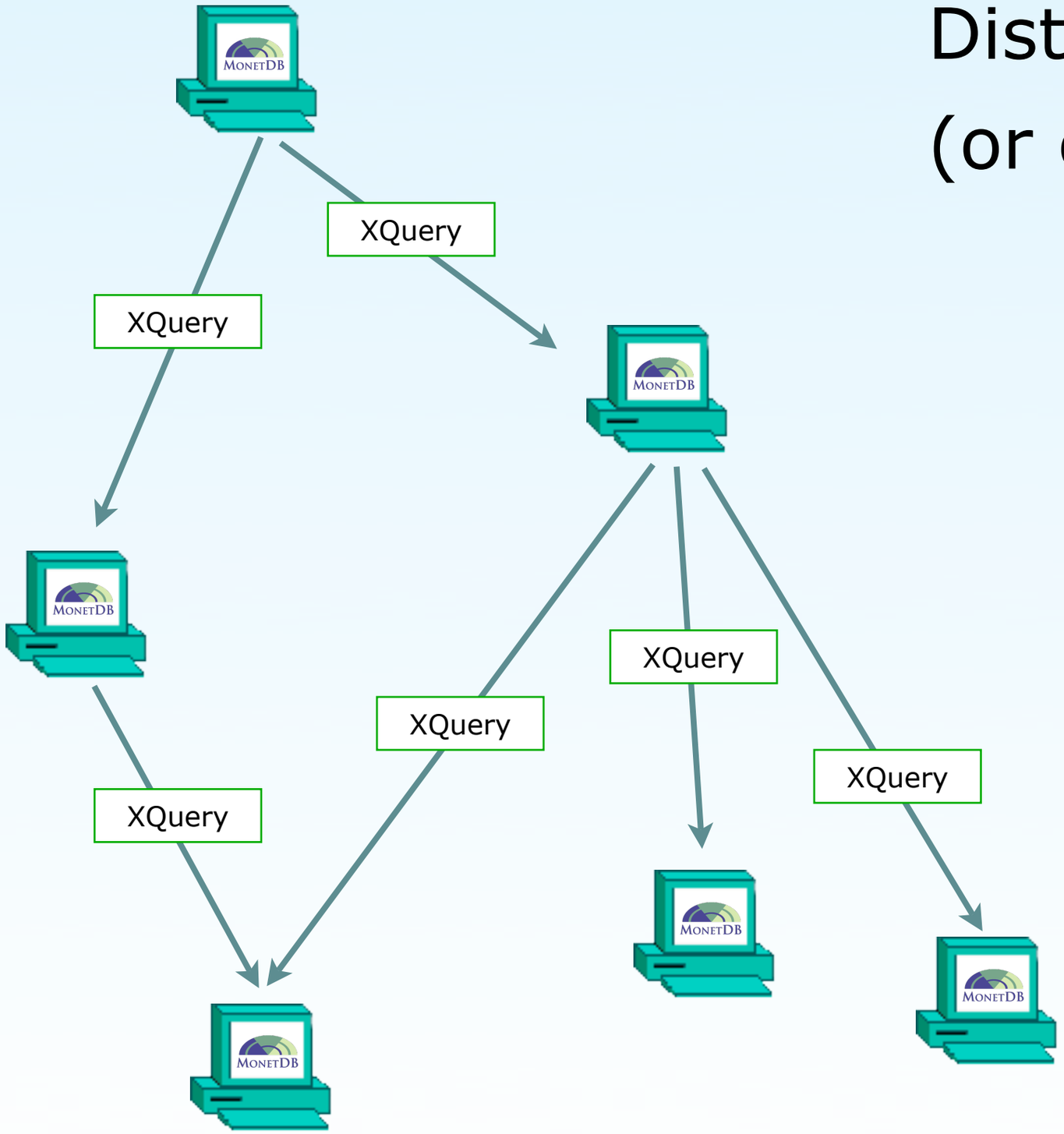
XRPC: Interoperable and Efficient Distributed XQuery

Ying Zhang

Peter Boncz

Centrum voor Wiskunde en Informatica

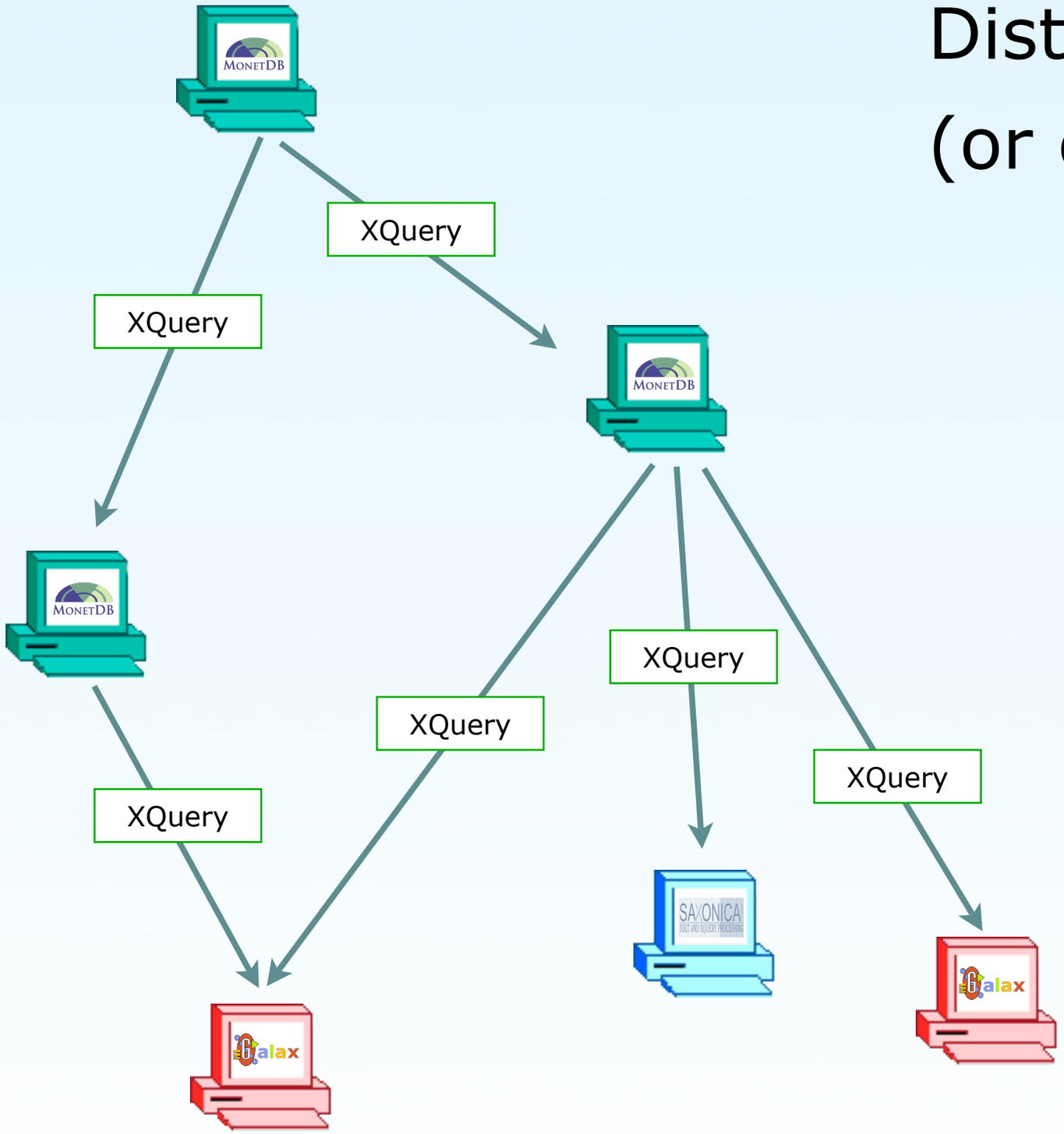
Goals of XRPC



Distributed XQuery

(or even P2P)

Goals of XRPC



Distributed XQuery

(or even P2P)

Design Goals

- Orthogonal
  - All XQuery semantics, typing and validation
  - also including XQuery Update Facility
- Interoperable
  - Network-layer specification!
    - SOAP-based, of course
- Potential for efficiency
  - Protocol exposes set-at-a-time opportunities
- Keep it simple
  - A minimal extension

XRPC Syntax Extension

`execute at` { *Expr* } { *FunApp(ParamList)* }

XRPC: XQuery + RPC

Distributed XQuery with XRPC: Possibilities

Peer3

for each `person` from `Vienna` in `doc1.xml`
who has bought something (`item`)
from `doc2.xml`
return ... something ...

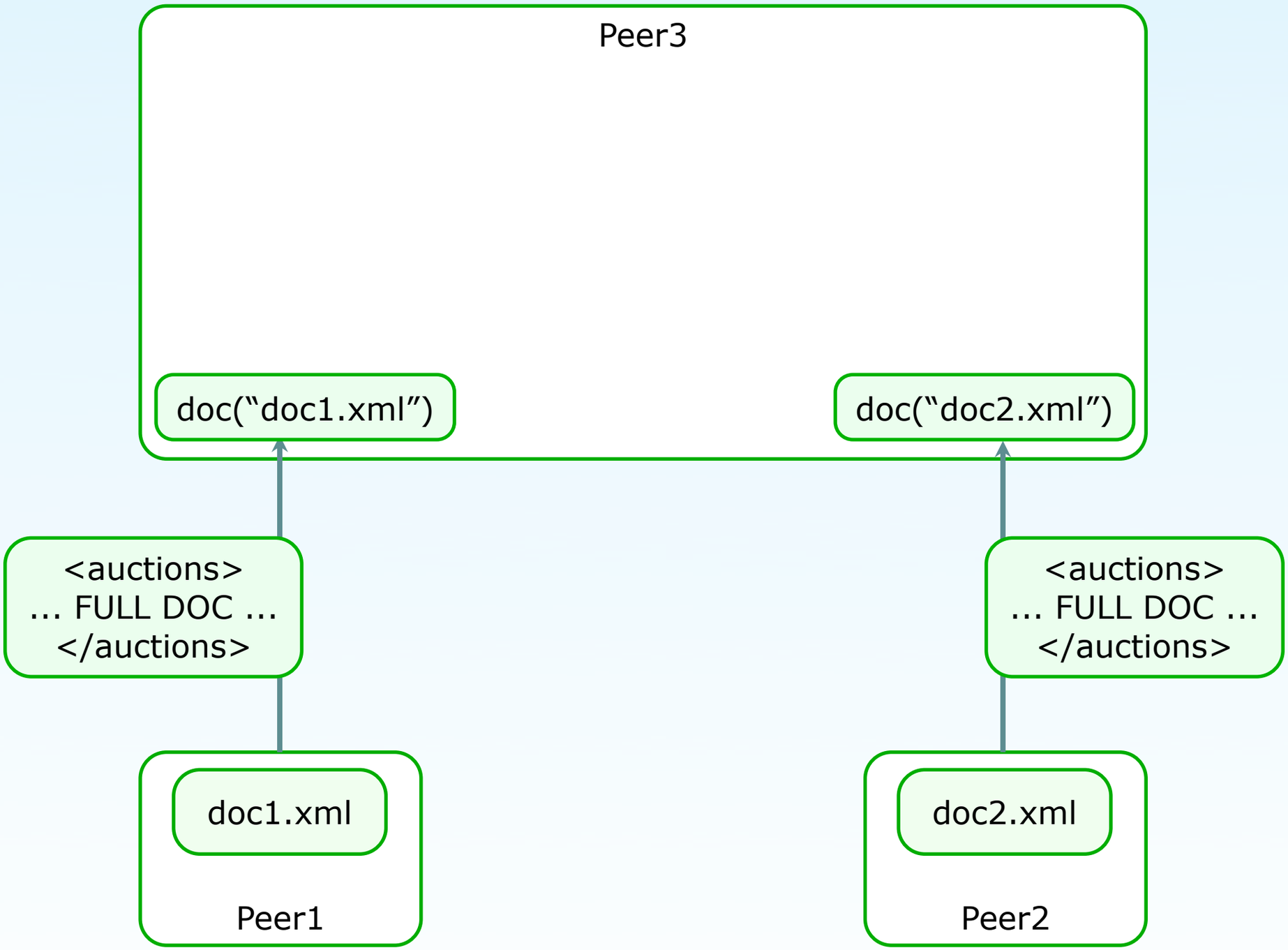
doc1.xml

Peer1

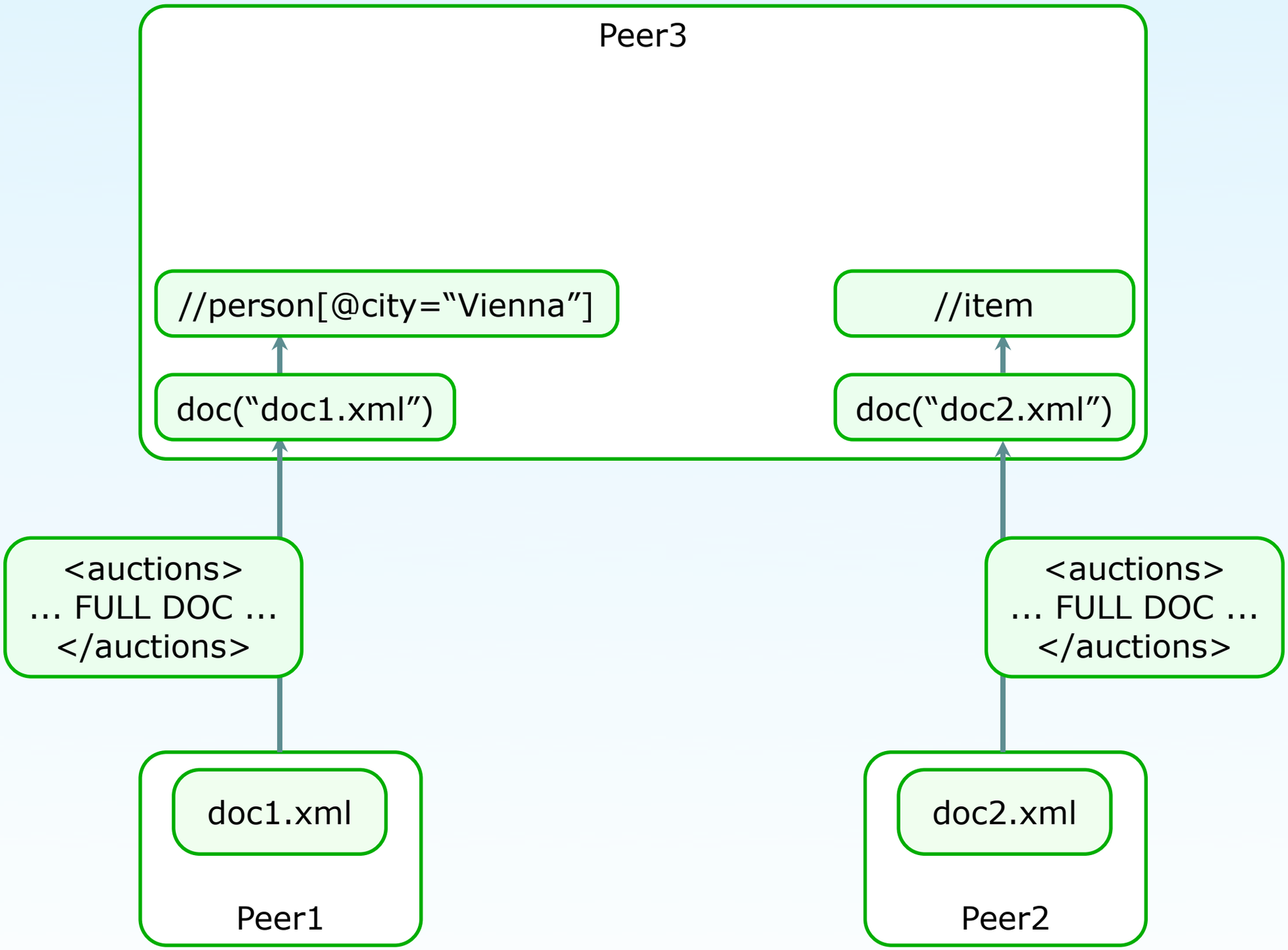
doc2.xml

Peer2

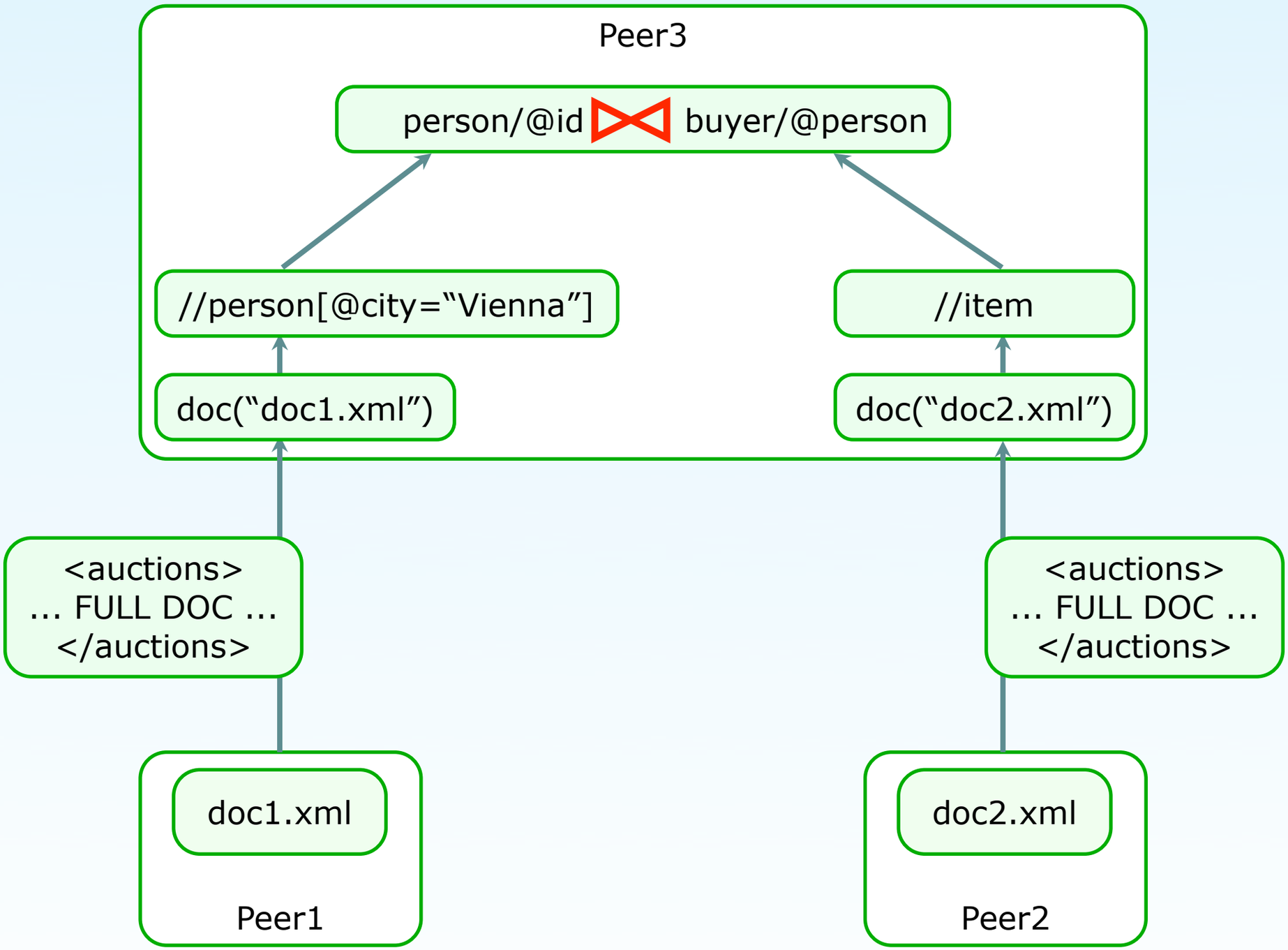
Distributed XQuery with XRPC: Possibilities



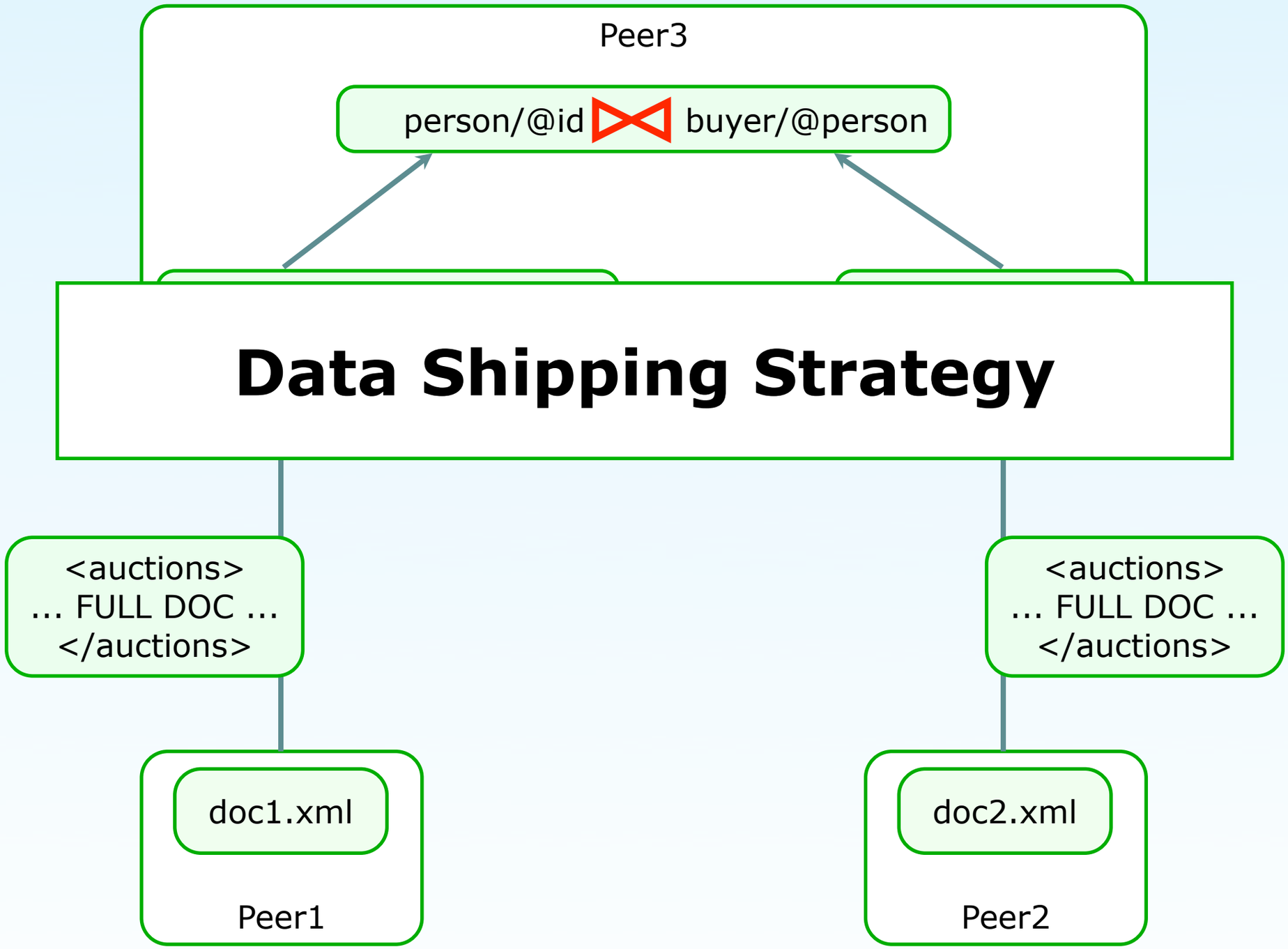
Distributed XQuery with XRPC: Possibilities



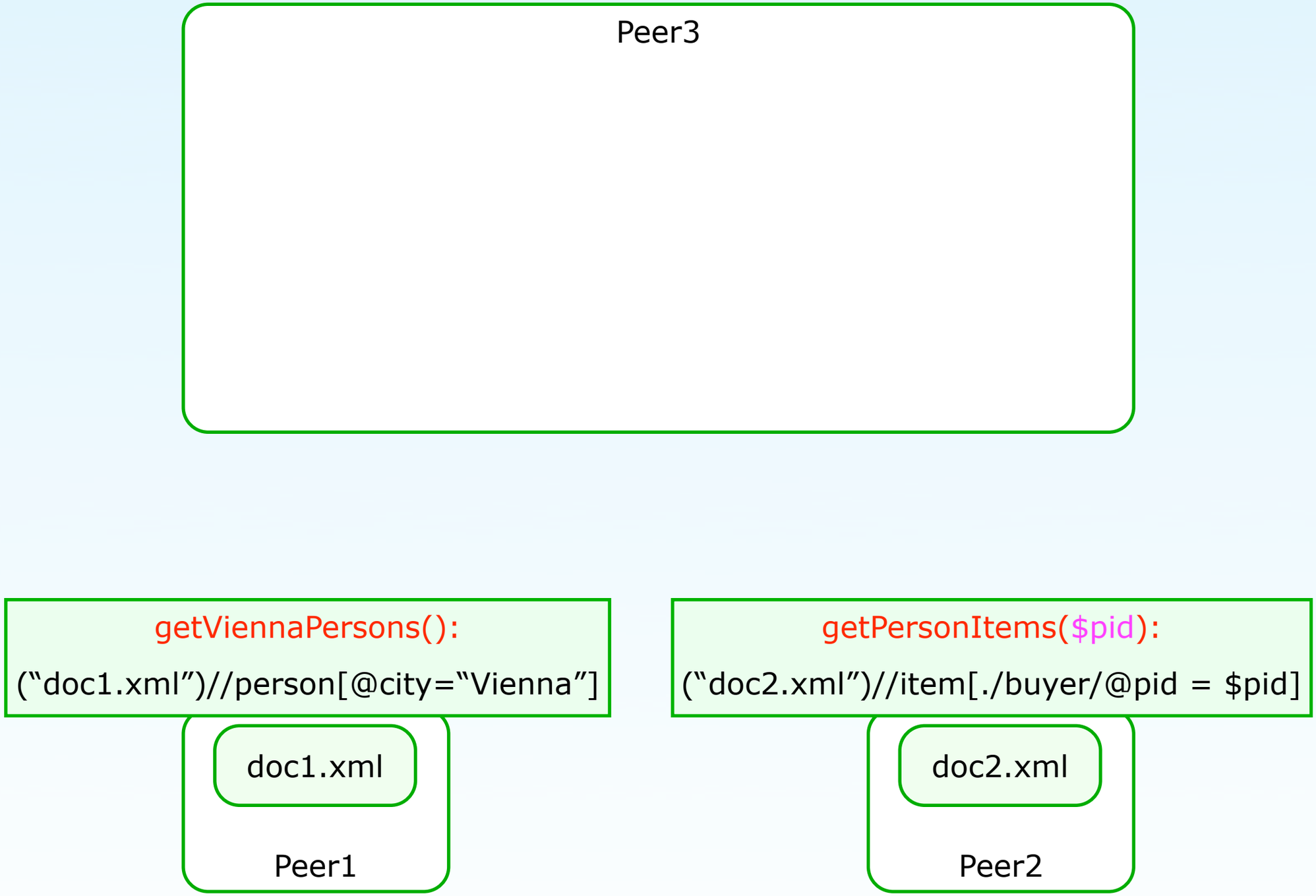
Distributed XQuery with XRPC: Possibilities



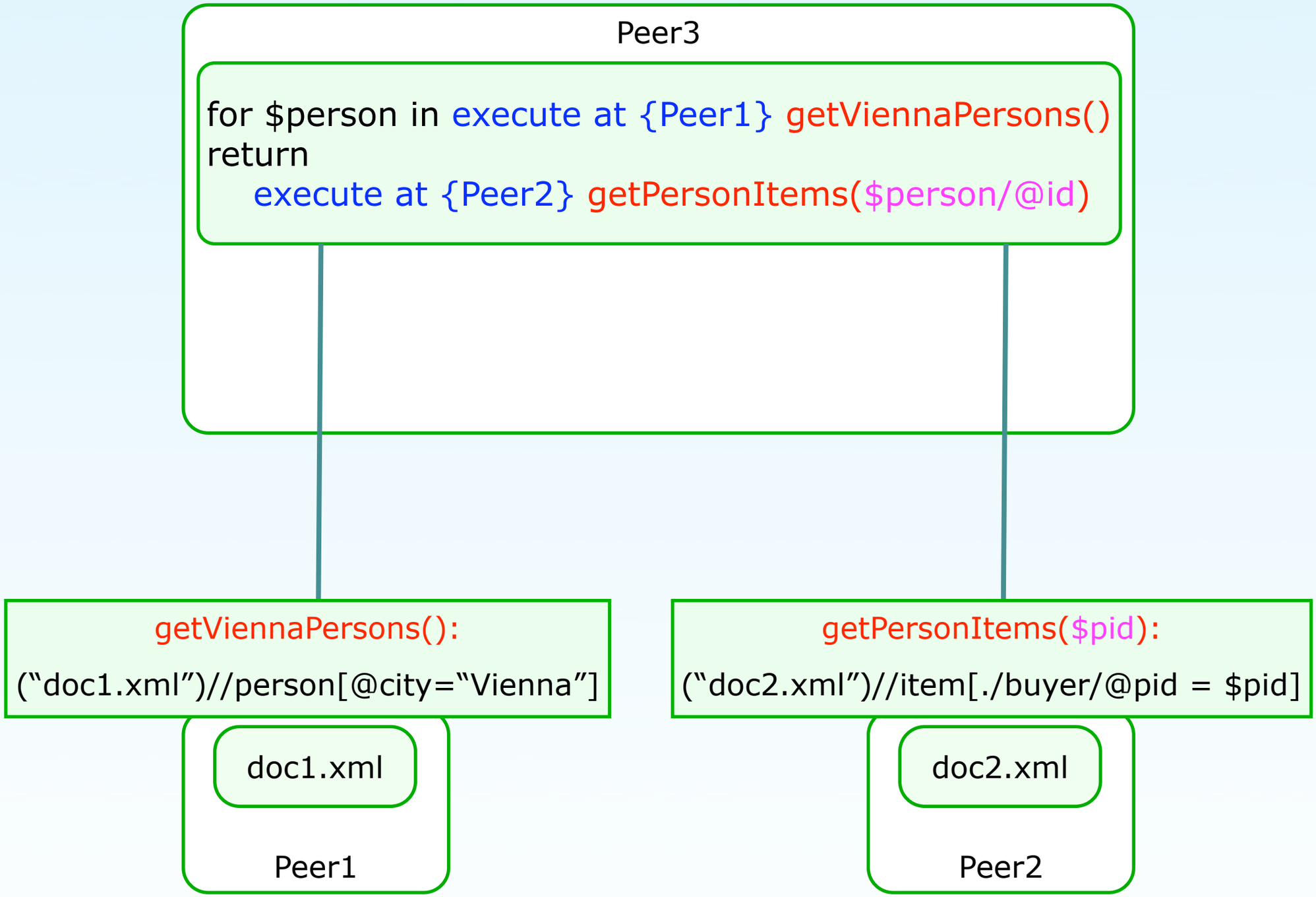
Distributed XQuery with XRPC: Possibilities



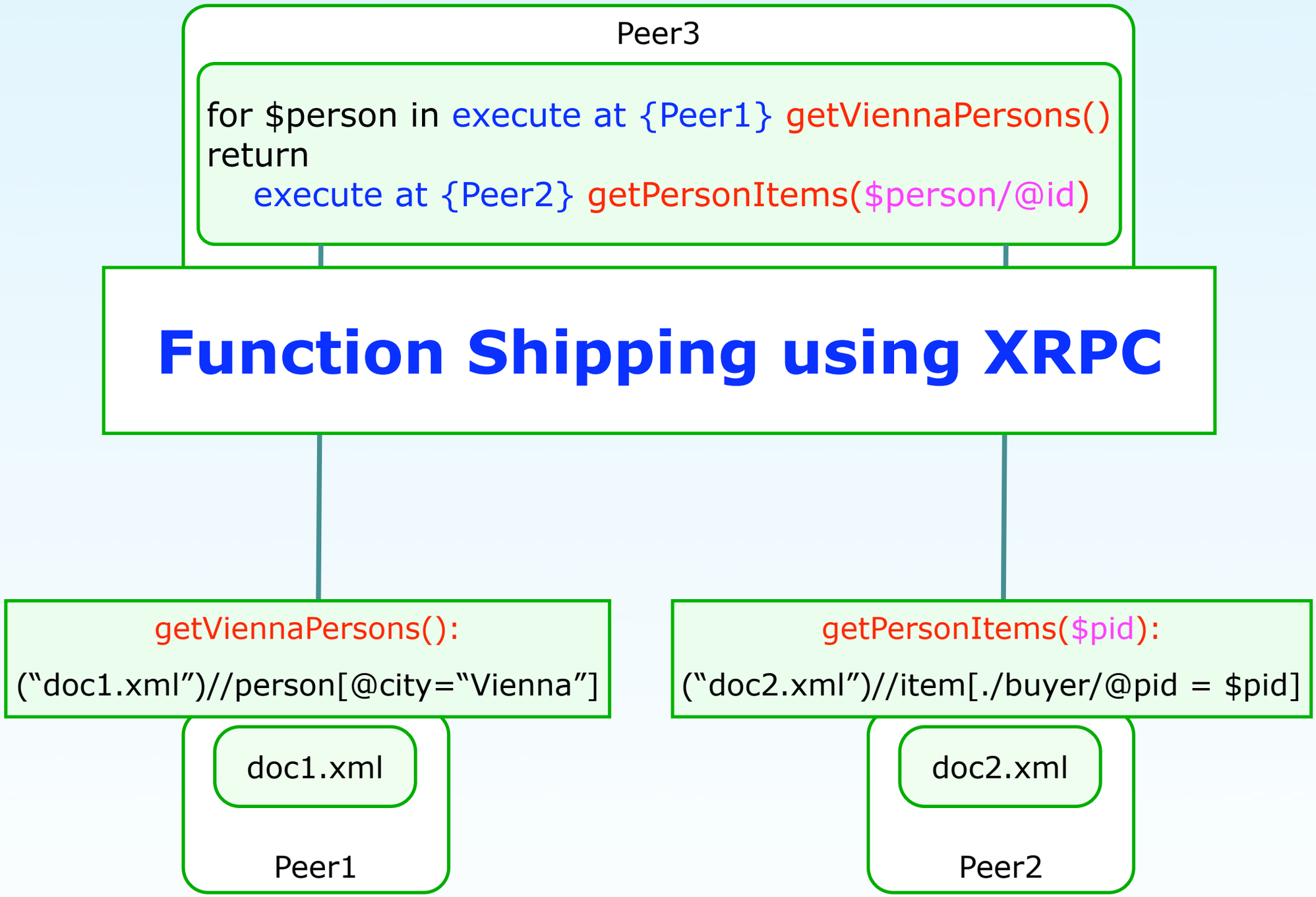
Distributed XQuery with XRPC: Possibilities



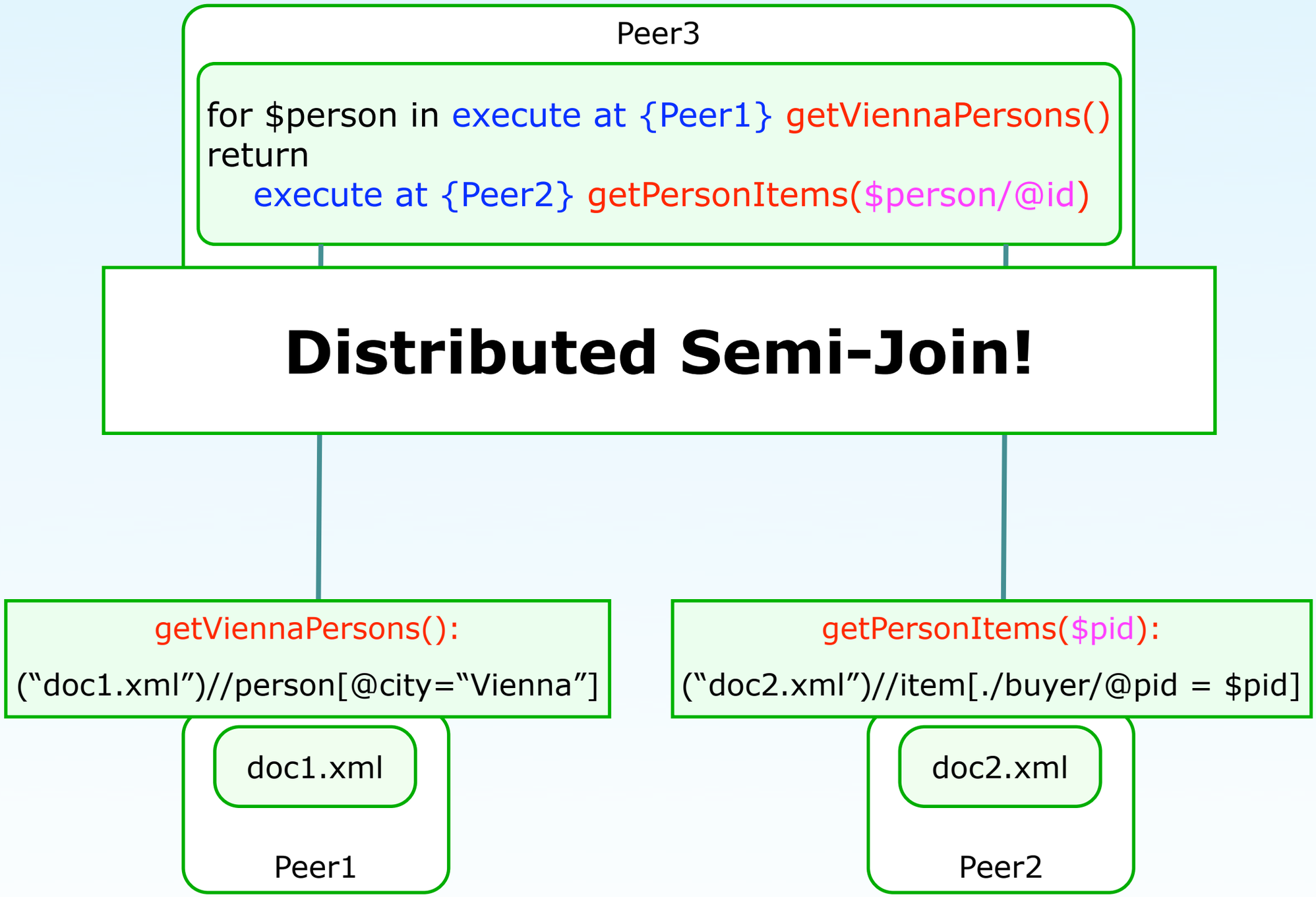
Distributed XQuery with XRPC: Possibilities



Distributed XQuery with XRPC: Possibilities



Distributed XQuery with XRPC: Possibilities



Distributed XQuery with XRPC: Possibilities

Peer3

```
for $person in execute at {Peer1} getViennaPersons()
return
    execute at {Peer2} getPersonItems($person/@id)
```

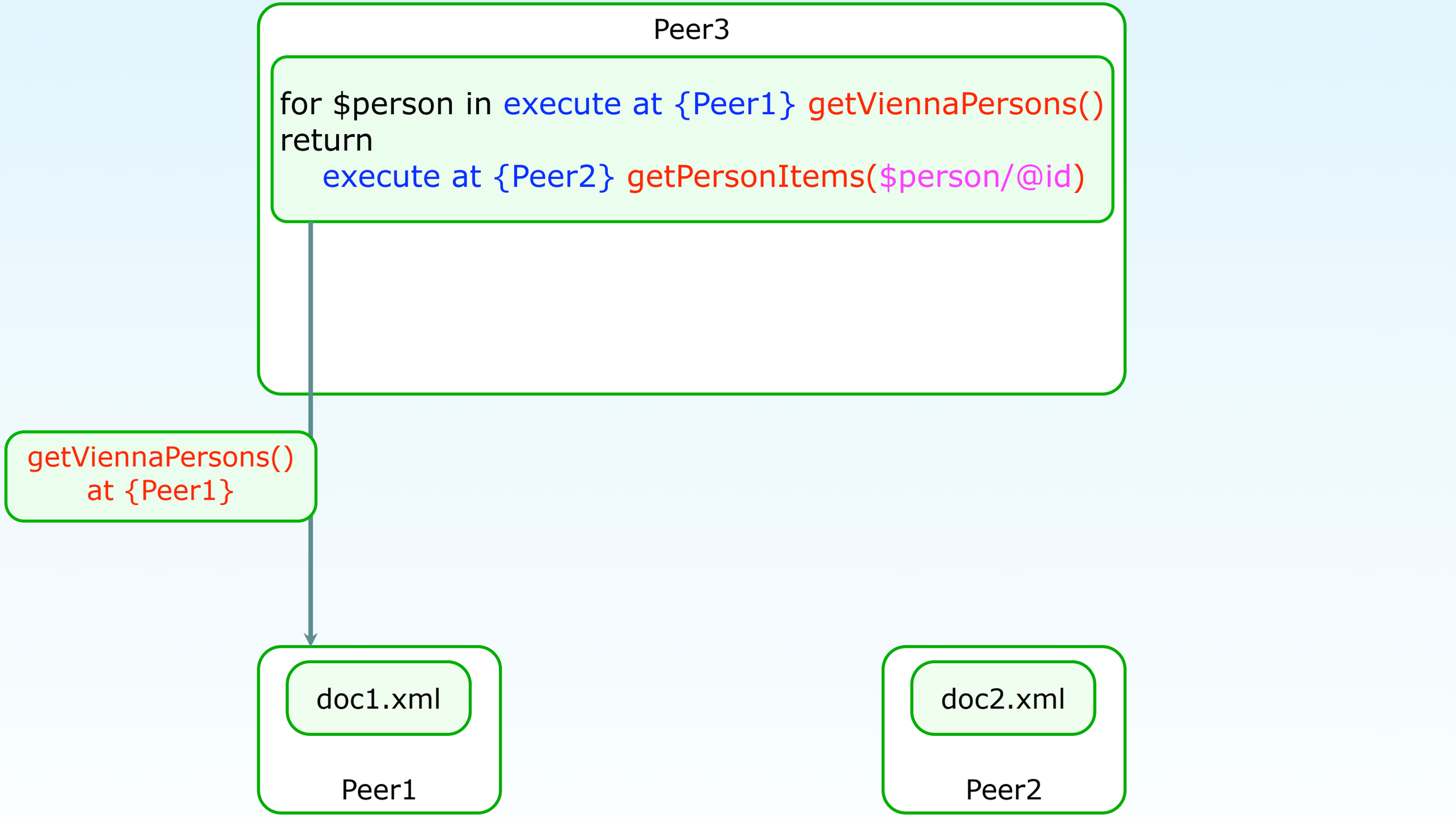
doc1.xml

Peer1

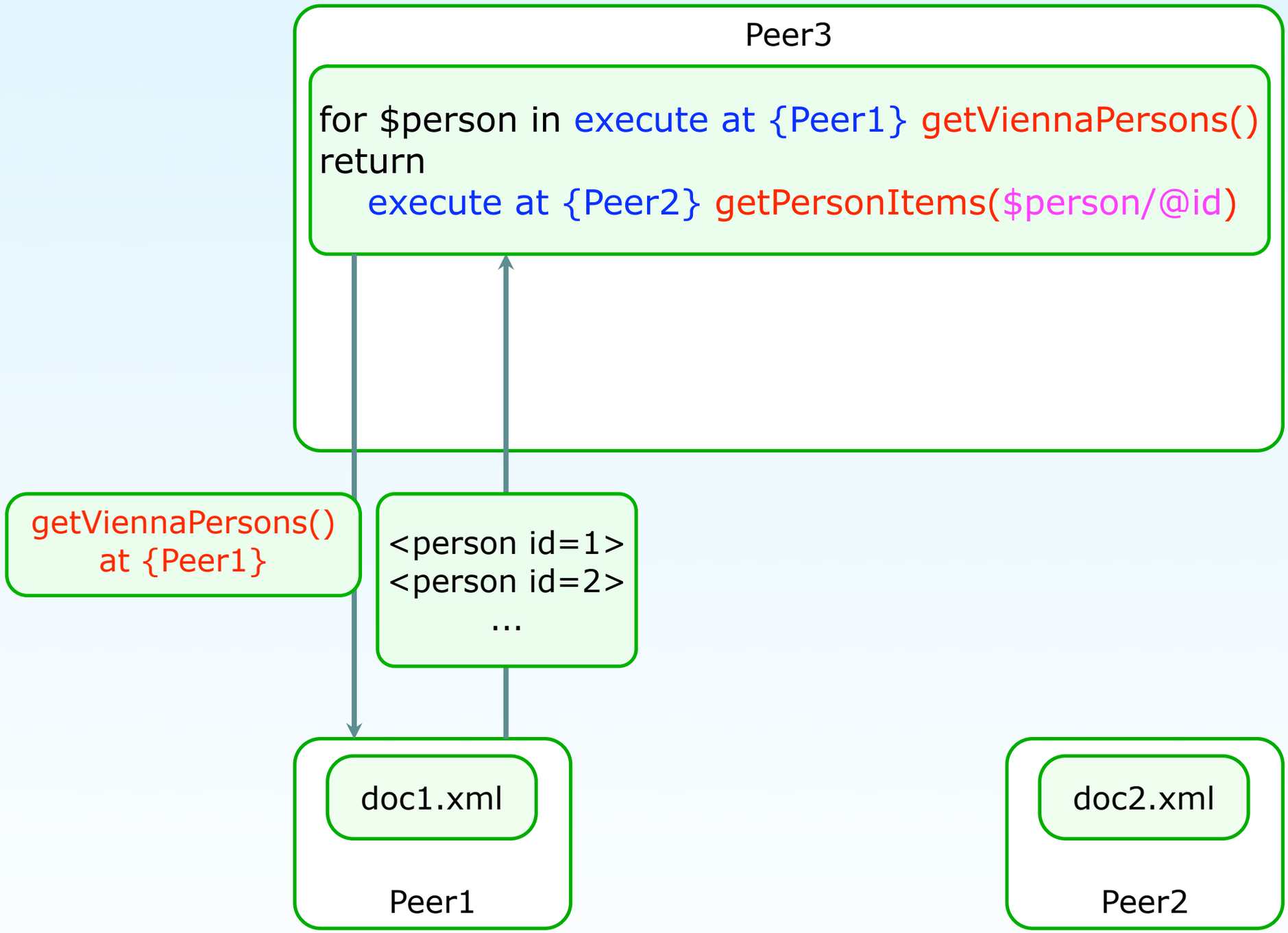
doc2.xml

Peer2

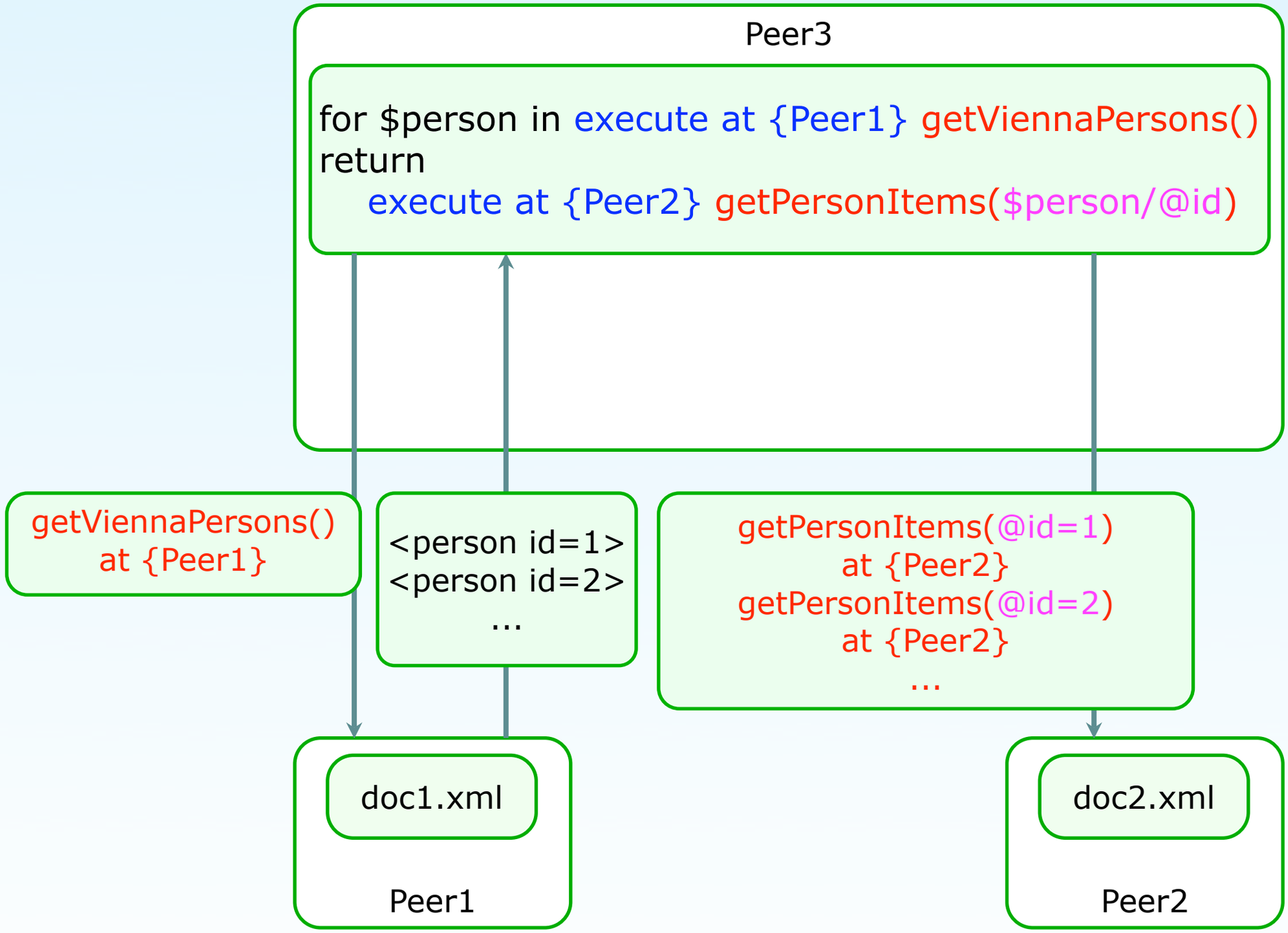
Distributed XQuery with XRPC: Possibilities



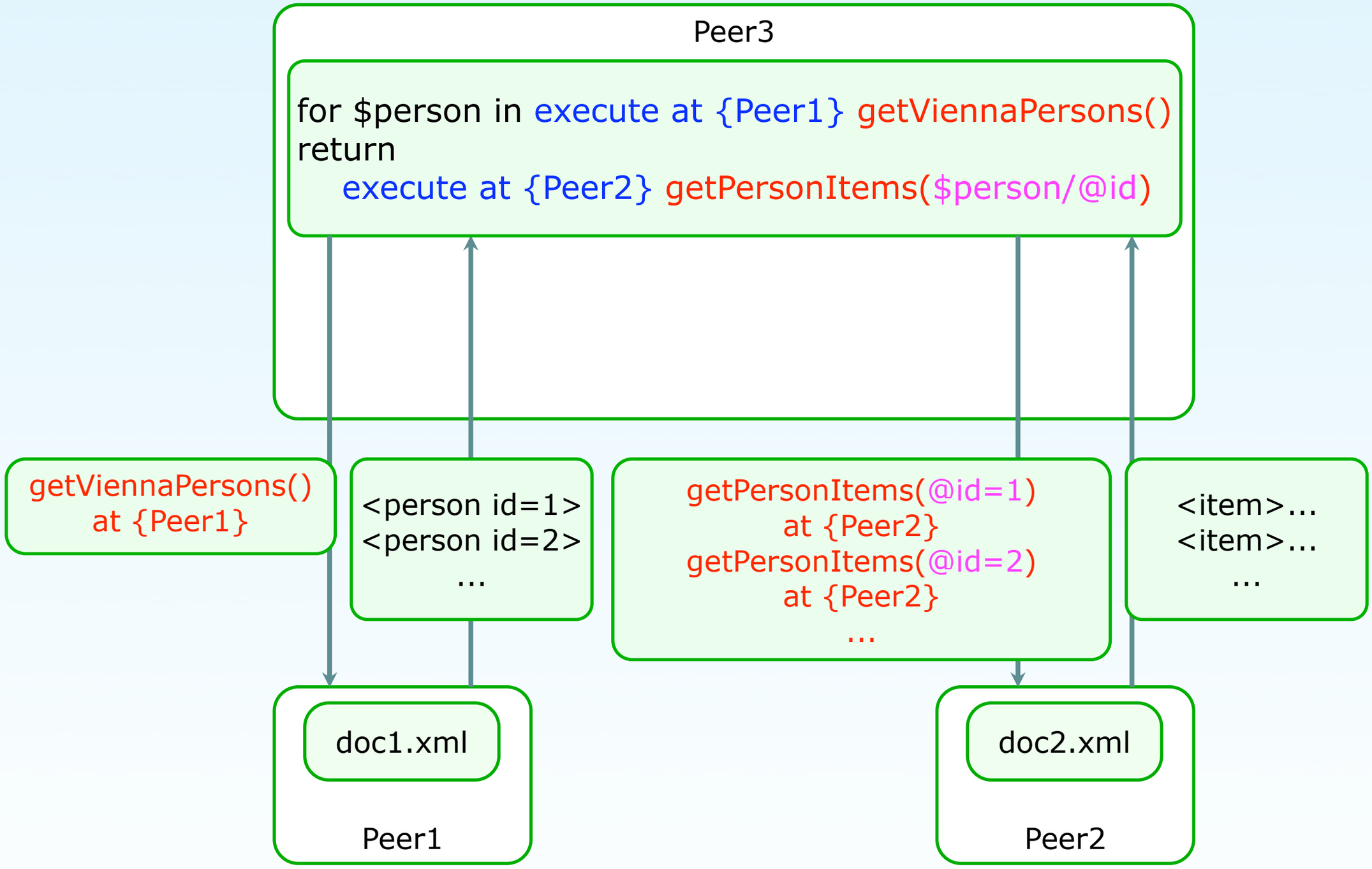
Distributed XQuery with XRPC: Possibilities



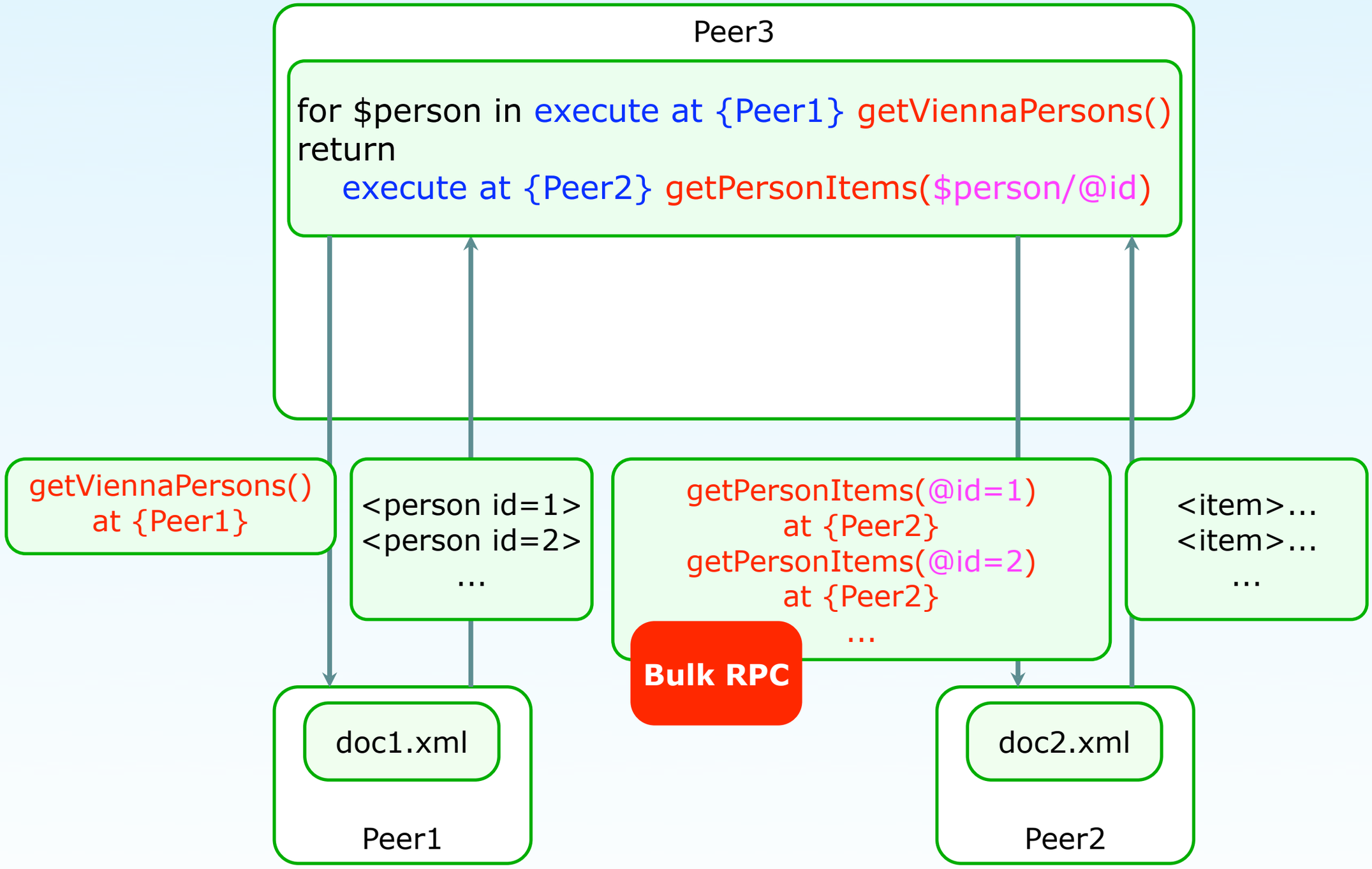
Distributed XQuery with XRPC: Possibilities



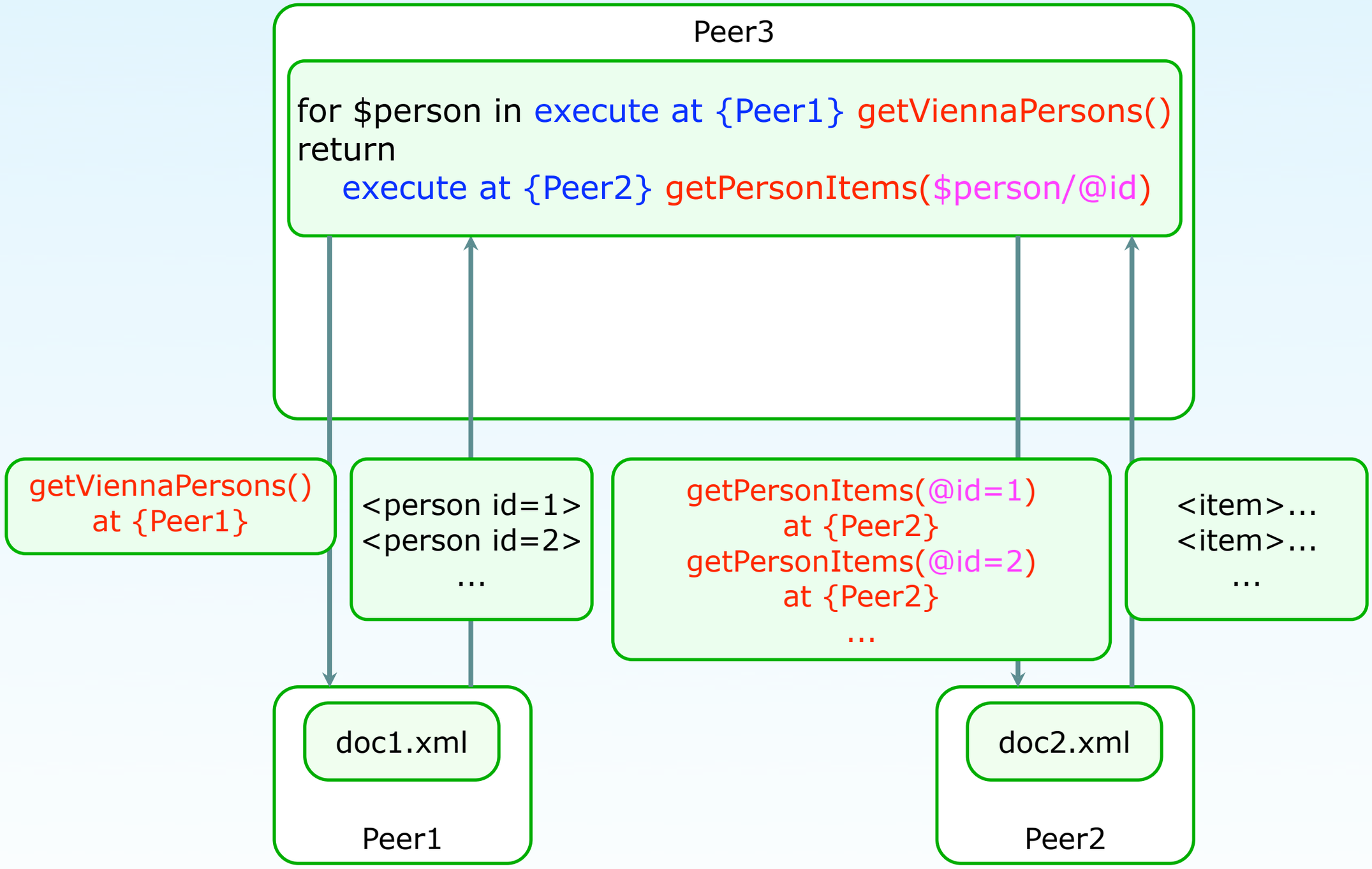
Distributed XQuery with XRPC: Possibilities



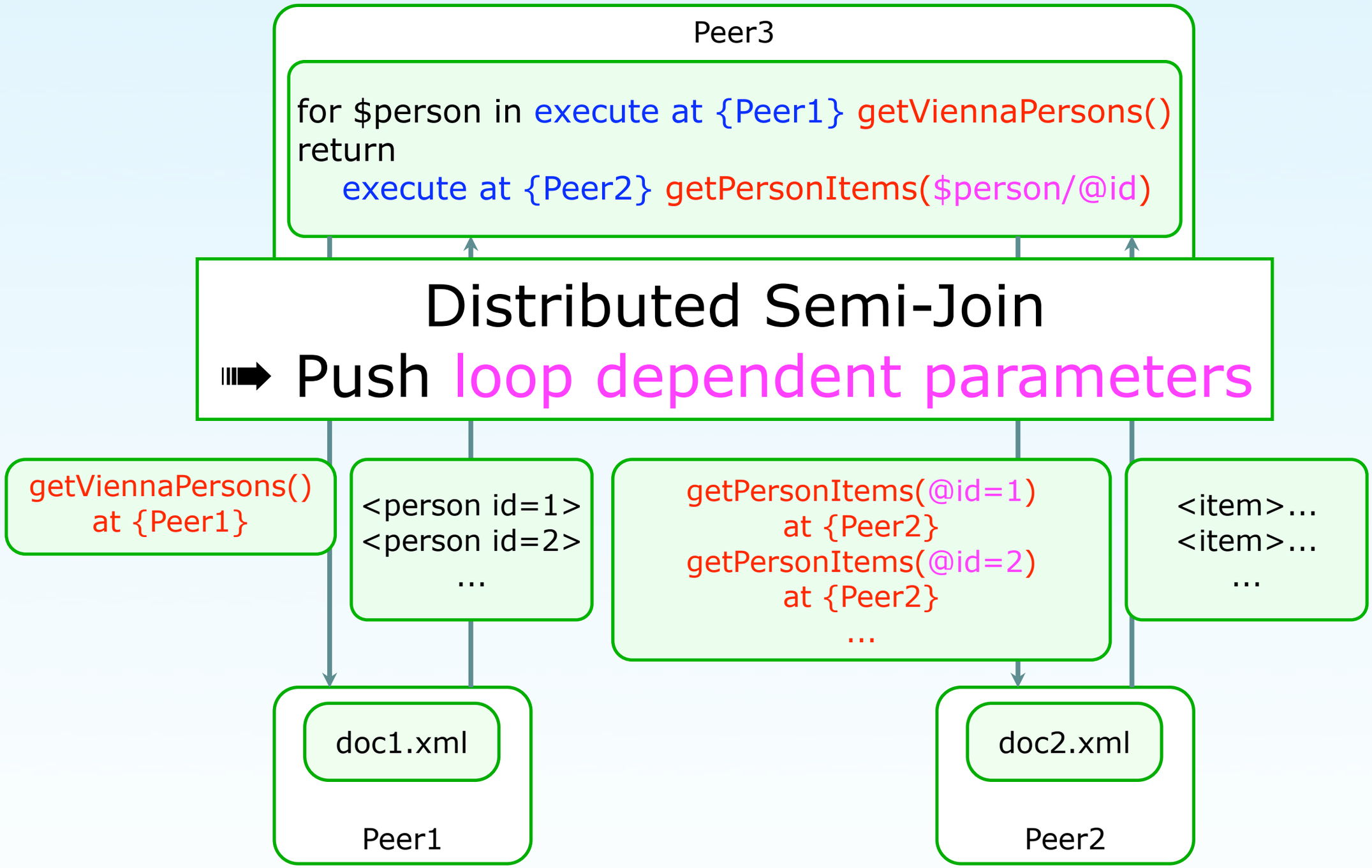
Distributed XQuery with XRPC: Possibilities



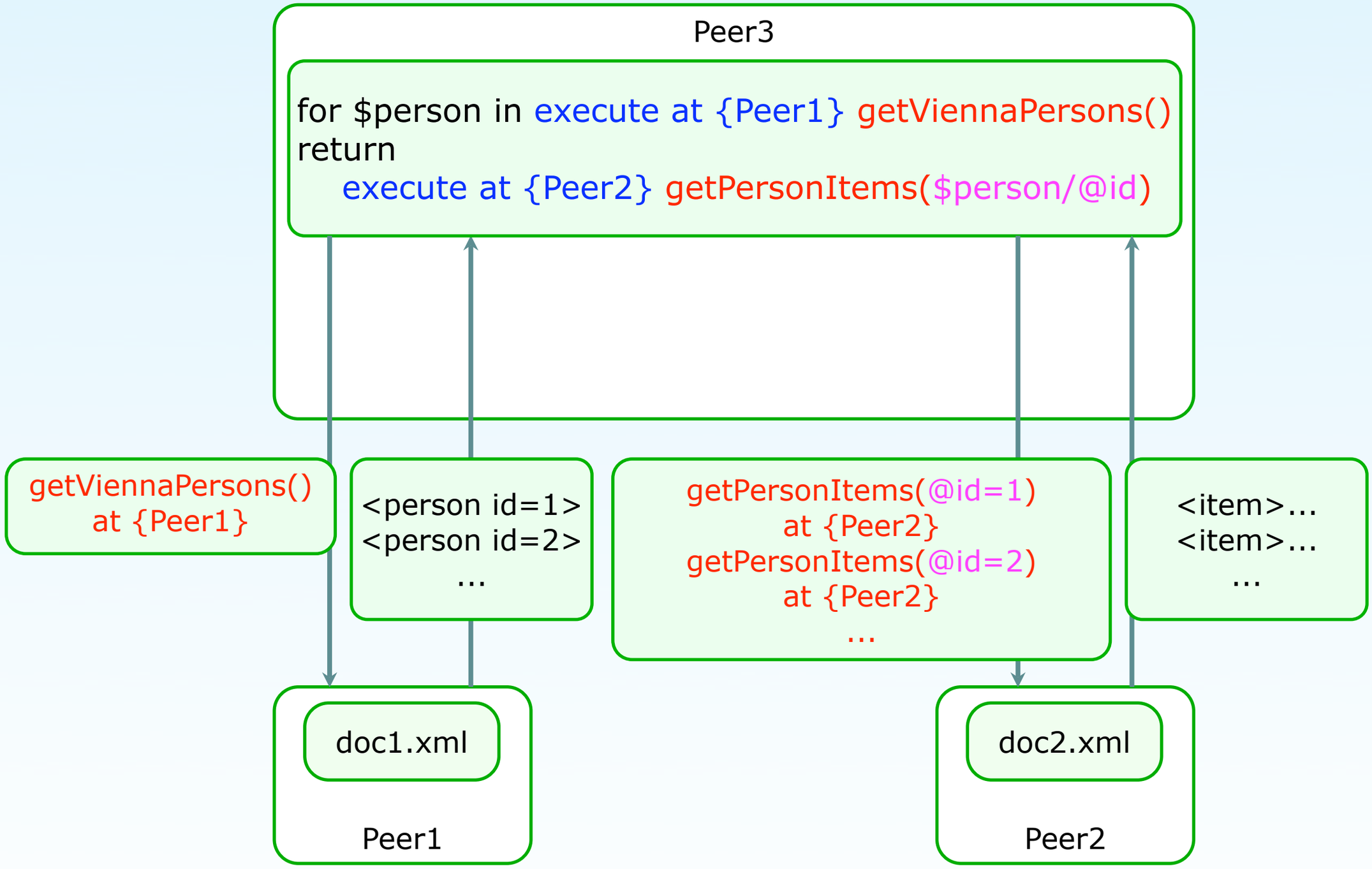
Distributed XQuery with XRPC: Possibilities



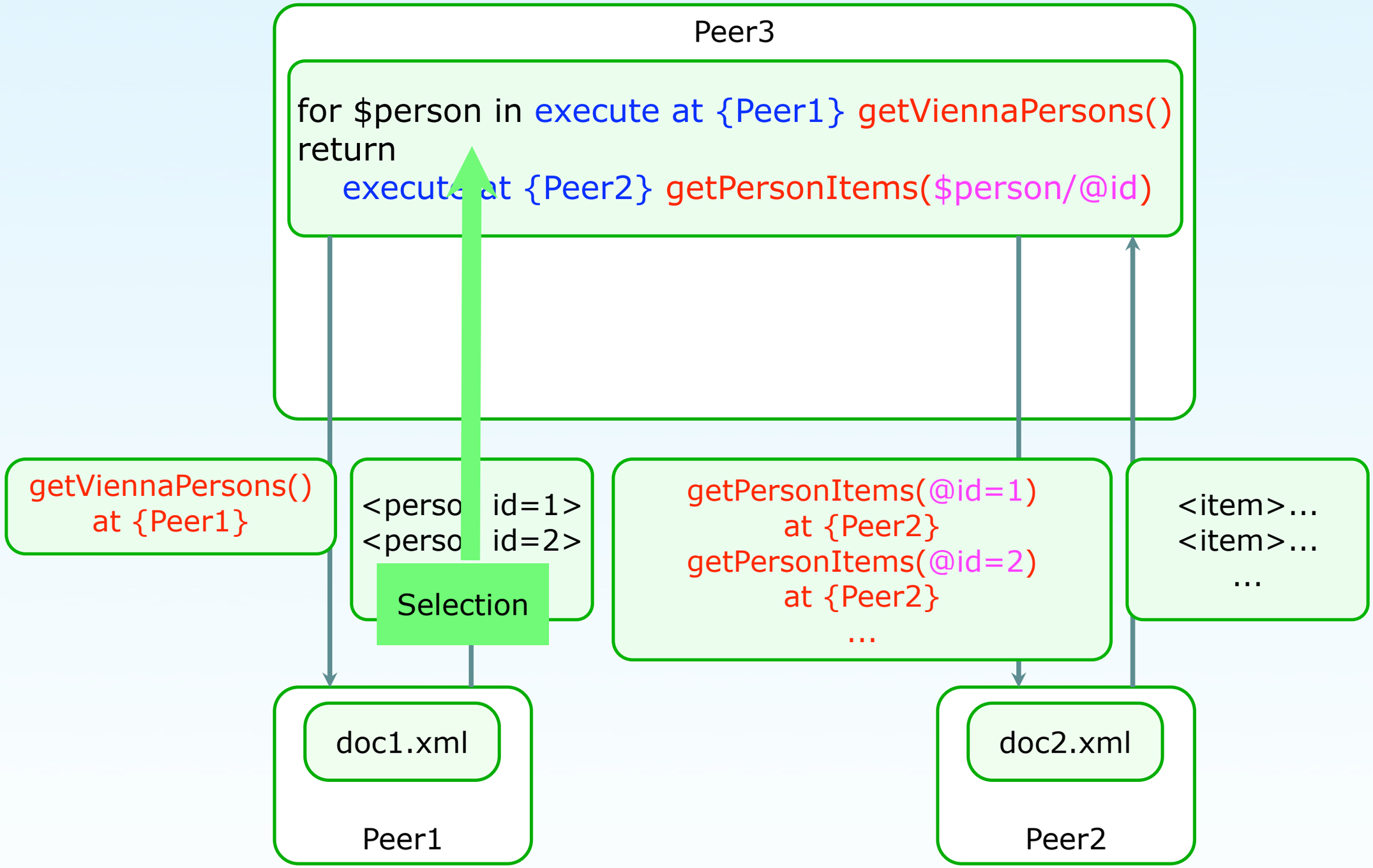
Distributed XQuery with XRPC: Possibilities



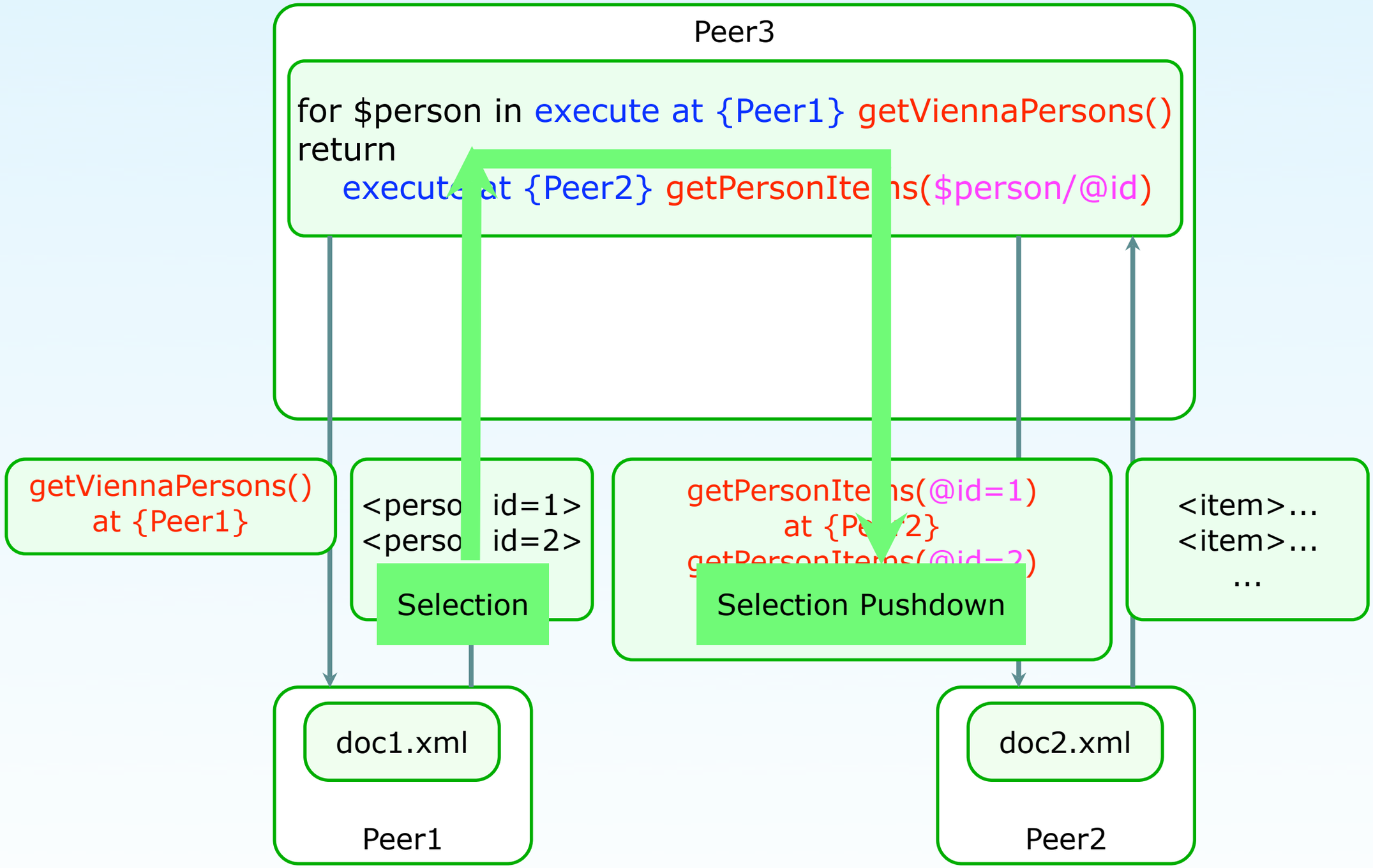
Distributed XQuery with XRPC: Possibilities



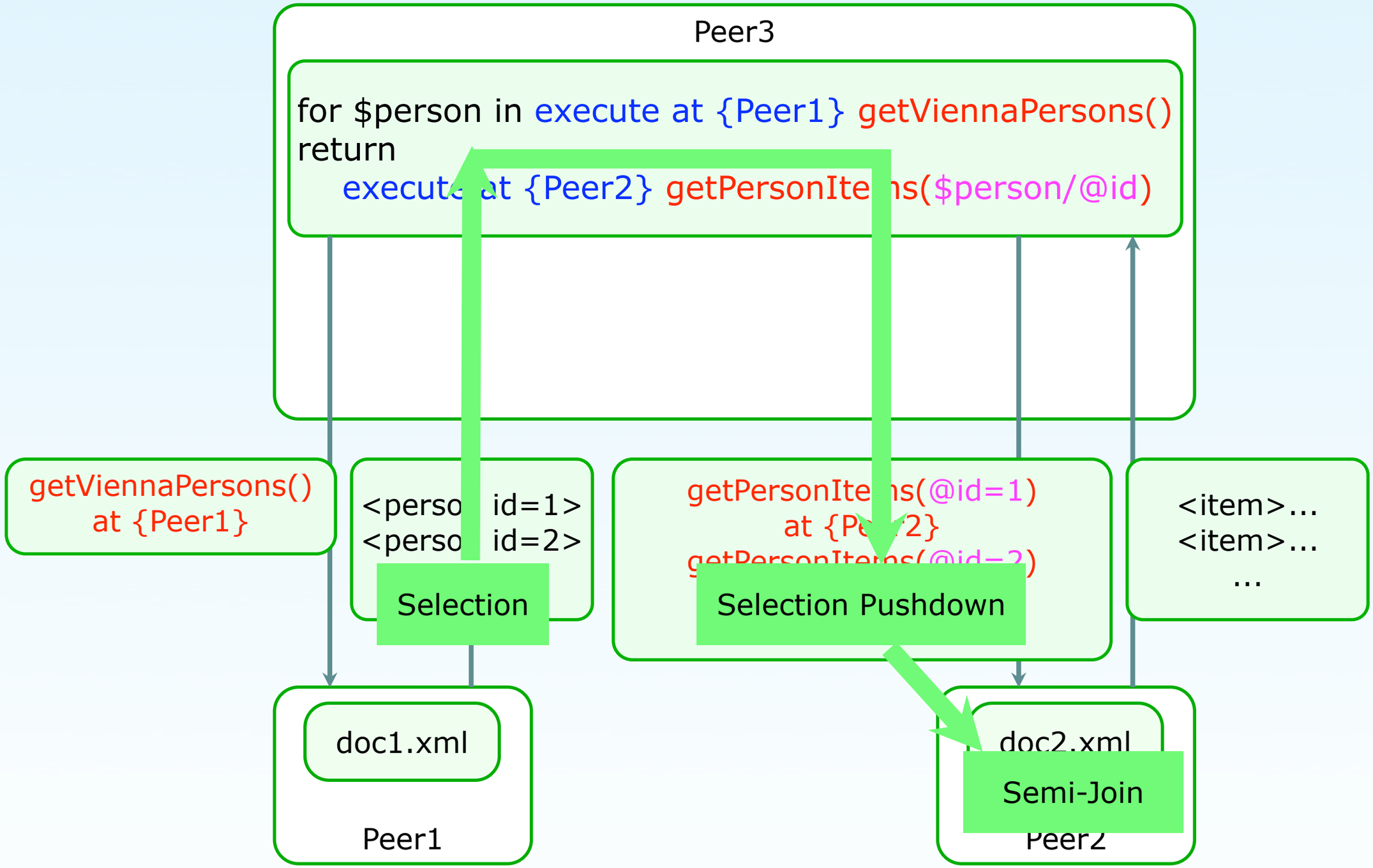
Distributed XQuery with XRPC: Possibilities



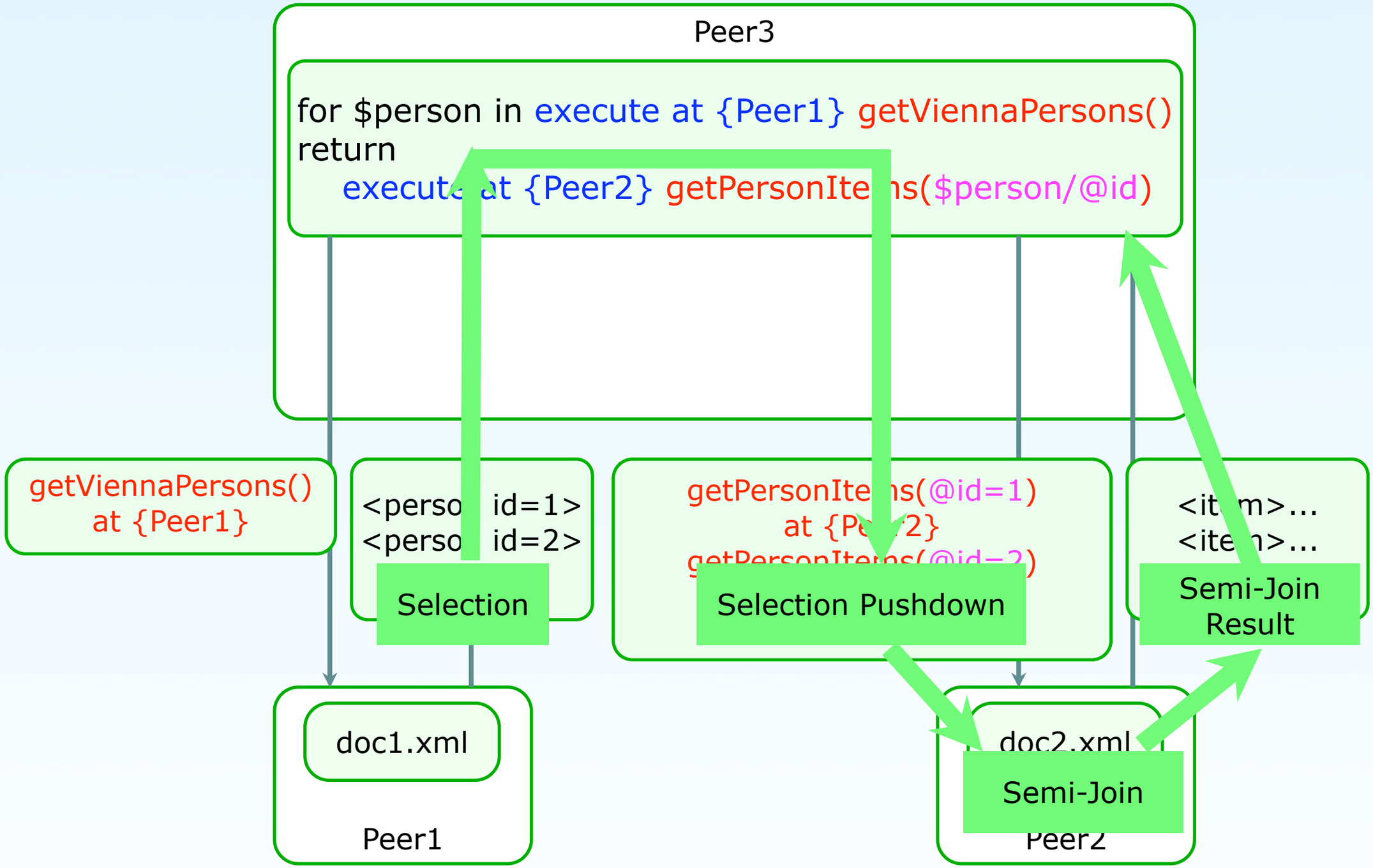
Distributed XQuery with XRPC: Possibilities



Distributed XQuery with XRPC: Possibilities



Distributed XQuery with XRPC: Possibilities

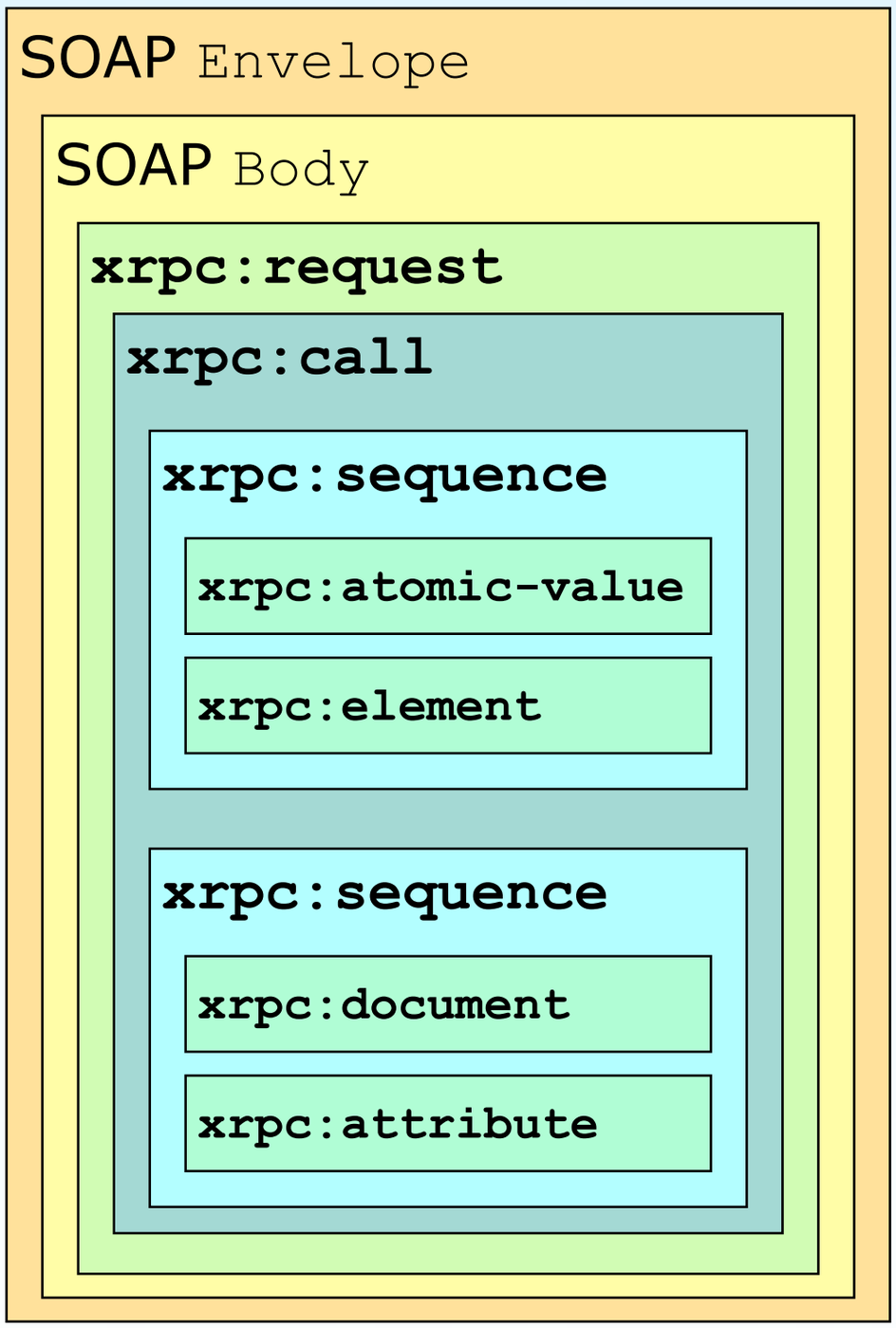


Remote Function Calls

- Functions: unit of distributed execution in XRPC
  - Defined in an XQuery `module`, identified by a URL
- XQuery: a compositional, functional language
  - each sub expression = function of its free variables
  - each function can be executed with XRPC on any peer
- Query decomposition & placement on peers
  - In this paper: by hand
  - Future work: automatic optimizer

The XRPC Network Protocol

The XRPC Network Protocol

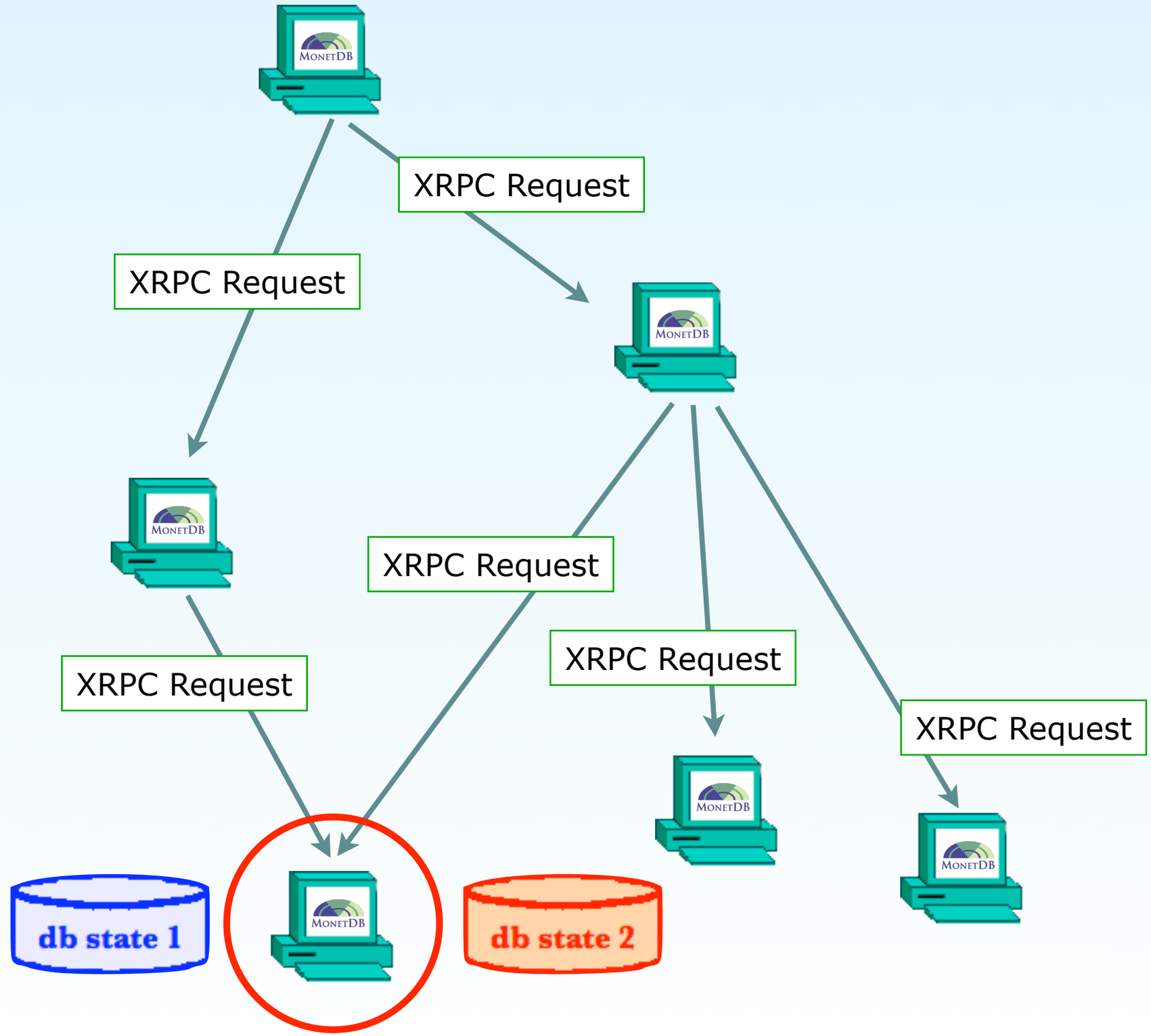


- SOAP XRPC
  - SOAP-based
  - XML over HTTP
- Full support of XDM types
  - All atomic types
  - All nodes types (pass-by-value)
  - User-defined schema types (namespace definition + `xsi:schemaLocation`)
  - Fully validate-able XRPC messages

Distributed Transactions with Isolation

- The basic XRPC protocol is stateless
- Higher level isolation
  - All calls: Repeatable-Reads
  - Updating calls: Distributed 2PC
  - Web Service Transactions Specifications
    - WS-Coordination
    - WS-Atomic Transactions: 2PC

Distributed Transactions with Isolation

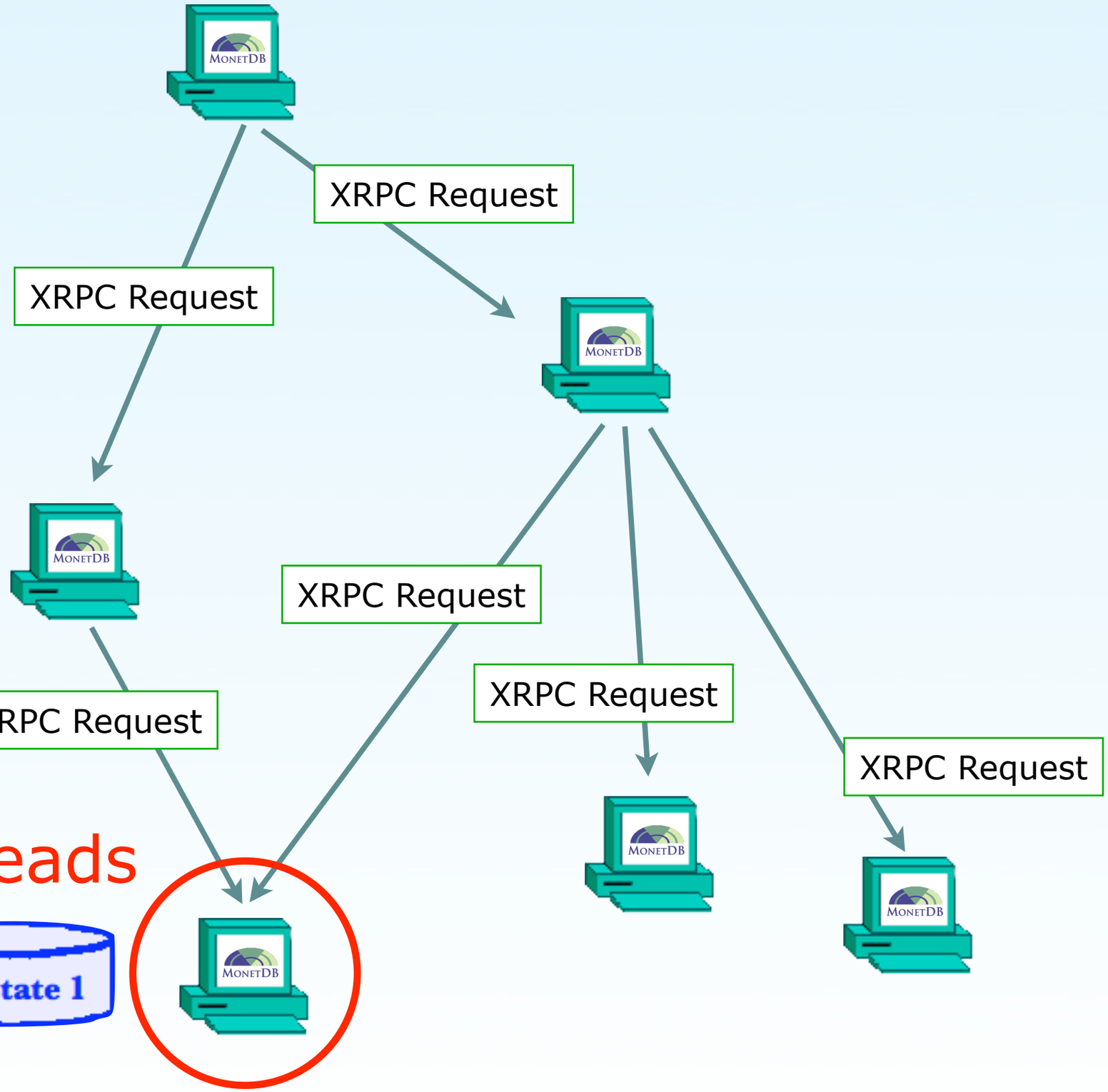


Distributed Transactions with Isolation

- The basic XRPC protocol is stateless

- Higher level isolation
  - **All calls: Repeatable-Reads**
  - Updating calls: Distributed 2PC
  - Web Service Transactions Specifications
    - WS-Coordination
    - WS-Atomic Transactions: 2PC

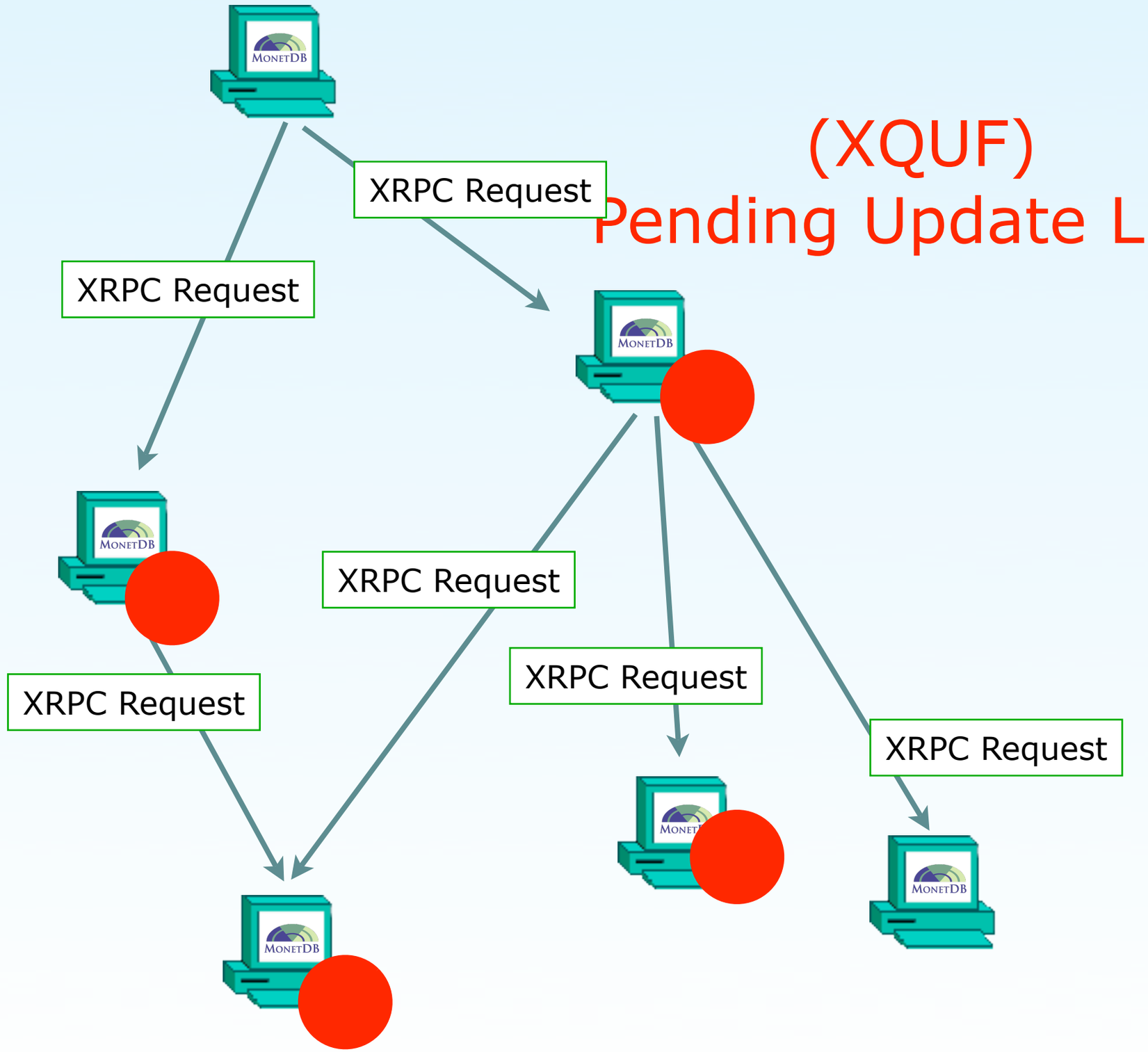
Distributed Transactions with Isolation



Distributed Transactions with Isolation

- The basic XRPC protocol is stateless
- Higher level isolation
  - All calls: Repeatable-Reads
  - **Updating calls: Distributed 2PC**
  - Web Service Transactions Specifications

Distributed Transactions with Isolation



Distributed Transactions with Isolation

- The basic XRPC protocol is stateless
- Higher level isolation
  - All calls: Repeatable-Reads
  - Updating calls: Distributed 2PC
  - **Web Service Transactions Specifications**
    - **SOAP based, of course**
    - **WS-Coordination**
    - **WS-Atomic Transactions: 2PC**

Bulk RPC

- What is it?
  - Inside a for-loop,
  - multiple requests ➠ single message
- Advantages:
  - Network latency + bandwidth
  - Exploit DBMS set-at-a-time infrastructure:
    bulk selections ➠ SOAP message ⋈ XML document

MonetDB/XQuery

- A pure relational XML DBMS
  - open-source, http://monetdb.cwi.nl
- The ***Pathfinder*** XQuery compiler
  - T. Grust, TUM
  - open-source, http://pathfinder-xquery.org
  - the ***Loop-Lifting*** technique

Performance

```
echoVoid() {() };
```

```
for $i in (1 to $x) return
execute at {"y.example.org" }{ t:echoVoid() }
```

	$x = 1	$x = 1000
one-at-a-time	2.6	2696
bulk	2.7	4

XRPC time of echoVoid() in msec

Distributed XQuery Standard?

Ideally:

Different XDBMS cooperatively answer queries

➠ standardize distributed XQuery

- If such a standard would be based on ideas like XRPC:
  - **Glue-less** integration
    - No middle-ware
    - No ODBC, JDBC

XRPC: Easy to Implement

- ☑ Serializer / Parser for SOAP messages
- ☐ Stub code (client) + request handler (server)

Meanwhile: XRPC Wrapper

- Plug-able component on top of the XQuery system
- No integration needed
  - Limitation: can not make XRPC calls

XRPC

MonetDB/XQuery

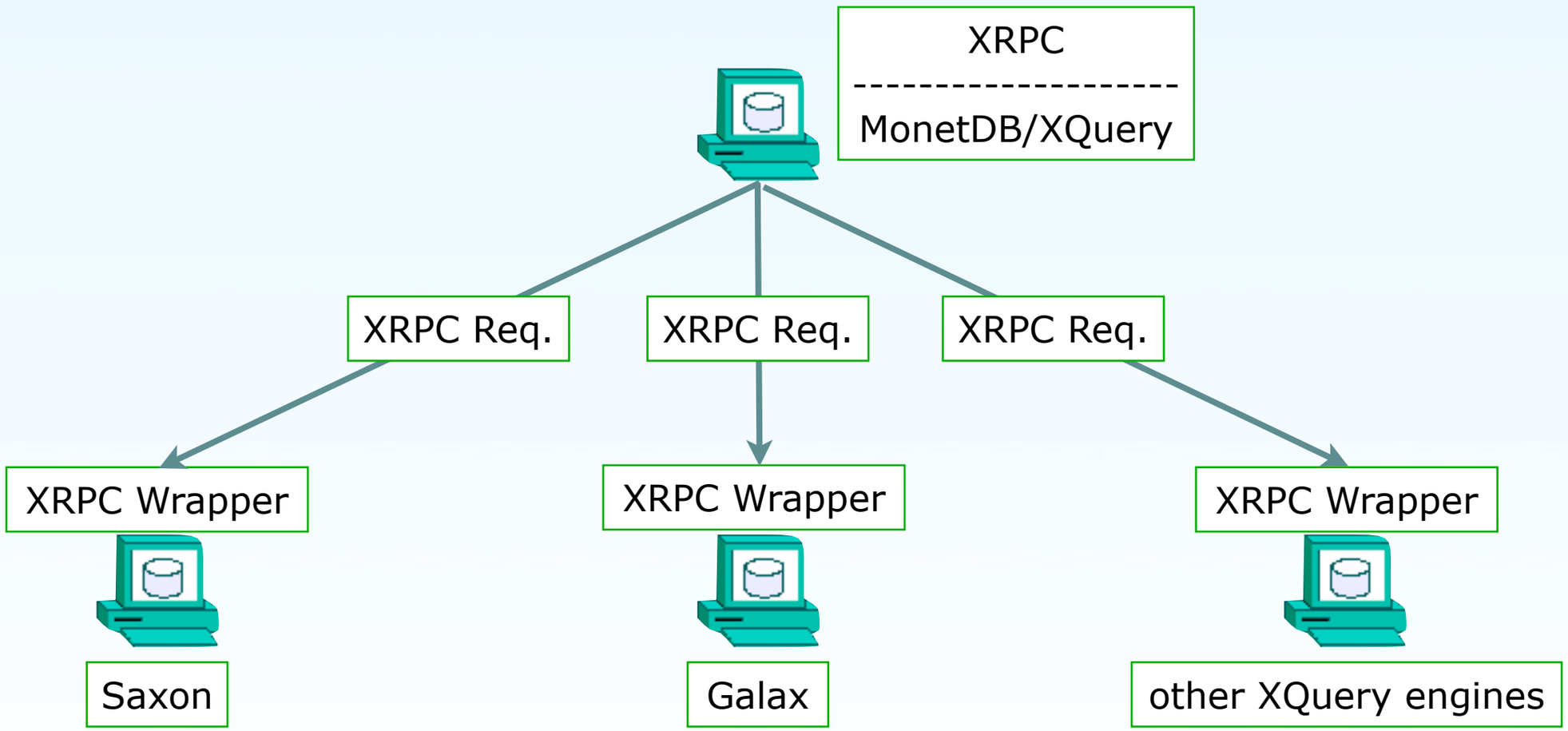
Saxon

Galax

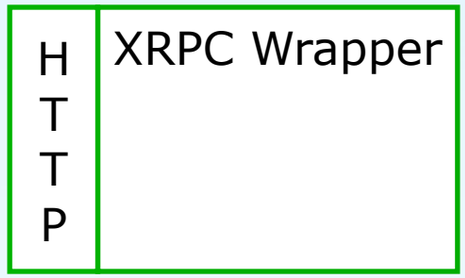
other XQuery engines

Meanwhile: XRPC Wrapper

- Plug-able component on top of the XQuery system
- No integration needed
  - Limitation: can not make XRPC calls

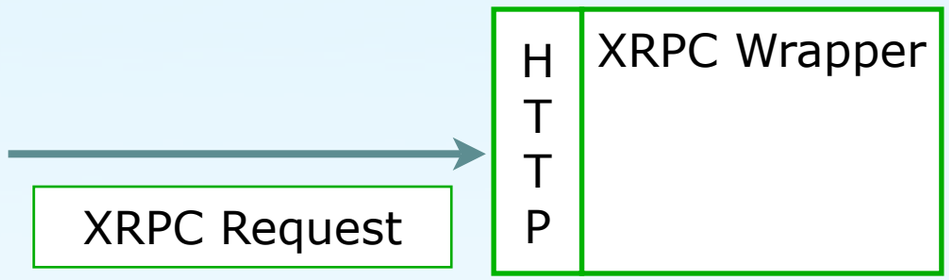


XRPC Wrapper Inside



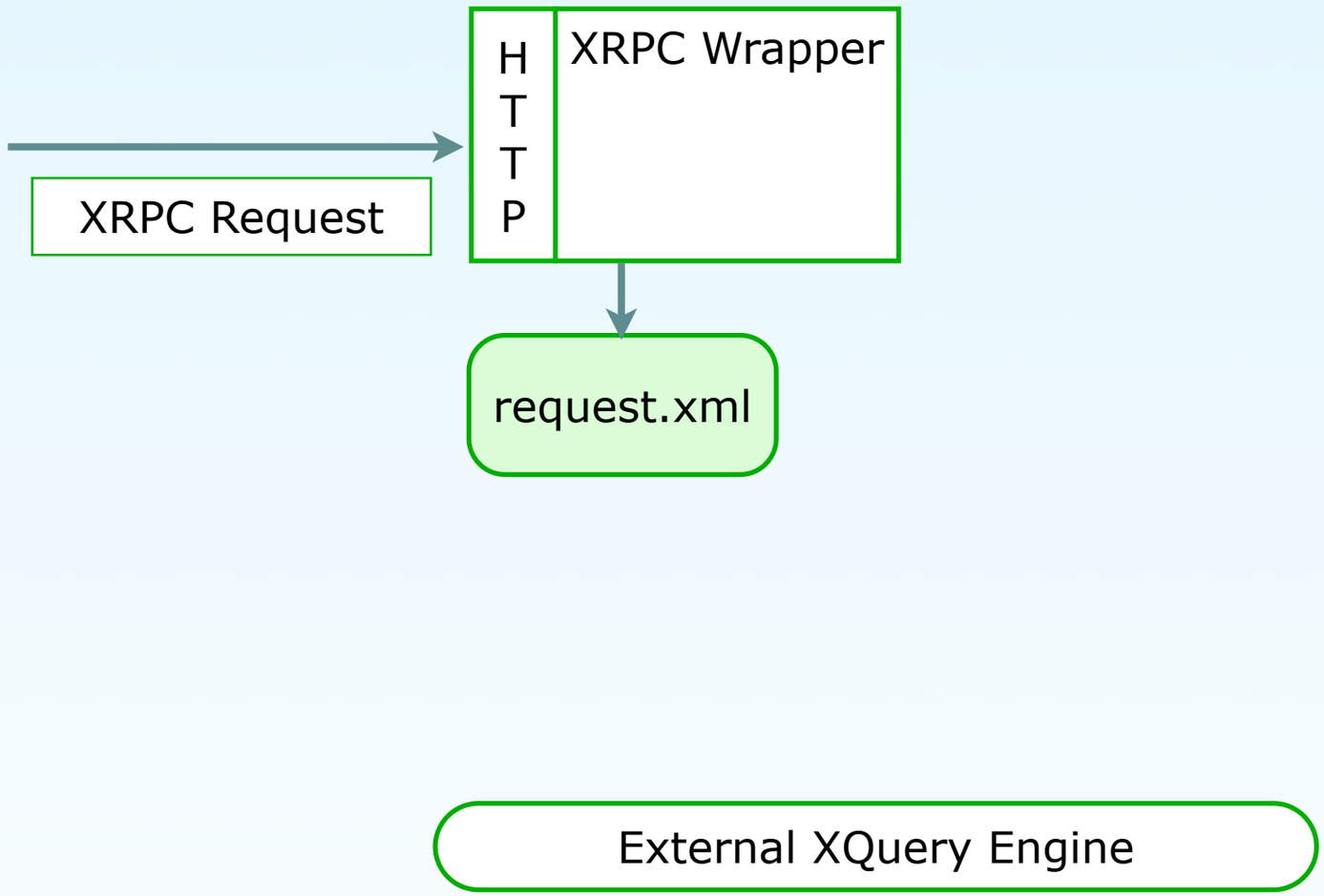
External XQuery Engine

XRPC Wrapper Inside

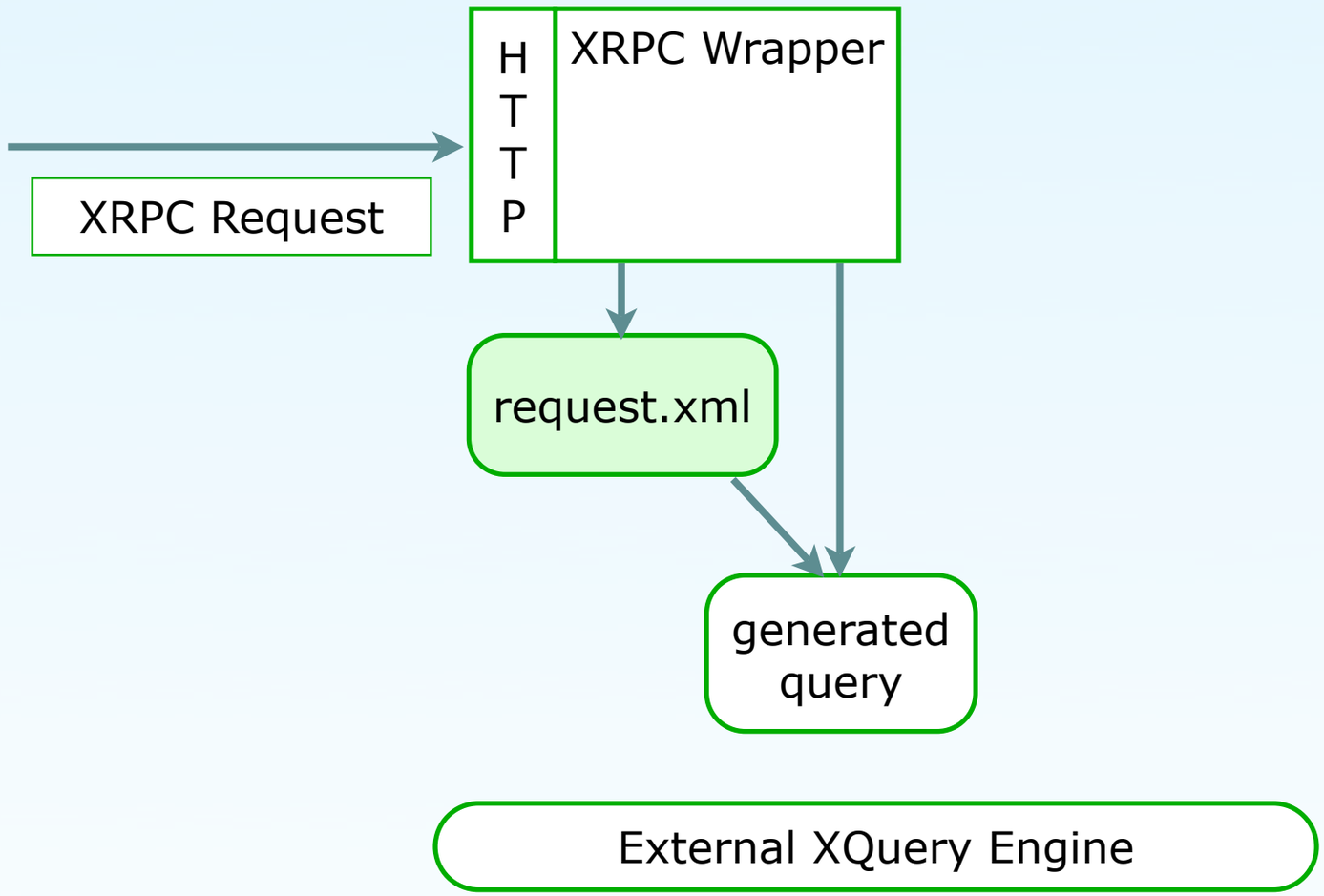


External XQuery Engine

XRPC Wrapper Inside



XRPC Wrapper Inside



XRPC Wrapper Inside

```
<xrpc:response>
for $call in doc("request.xml")//xrpc:call
  $param1:= deserialize($call/xrpc:sequence[1])
  return getAuctions($param1)
</xrpc:response>
```

Generated XQuery

request.xml

```
...
<xrpc:call>
  <xrpc:sequence>
    "person1"

<xrpc:call>
  <xrpc:sequence>
    "person3"
...
```

auctions.xml

```
...
<closed_auction>
  <price>...
  <buyer "person1">

<closed_auction>
  <price>...
  <buyer "person3">
...
```

XRPC Wrapper Inside

```
<xrpc:response>
for $call in doc("request.xml")//xrpc:call
  $param1:= deserialize($call/xrpc:sequence[1])
  return getAuctions($param1)
</xrpc:response>
```

Generated XQuery

request.xml

```
...
<xrpc:call>
  <xrpc:sequence>
    "person1"

<xrpc:call>
  <xrpc:sequence>
    "person3"
...
```

auctions.xml

```
...
<closed_auction>
  <price>...
  <buyer "person1">

<closed_auction>
  <price>...
  <buyer "person3">
...
```

XRPC Wrapper Inside

```
<xrpc:response>
for $call in doc("request.xml")//xrpc:call
  $param1:= deserialize($call/xrpc:sequence[1])
  return getAuctions($param1)
</xrpc:response>
```

Generated XQuery

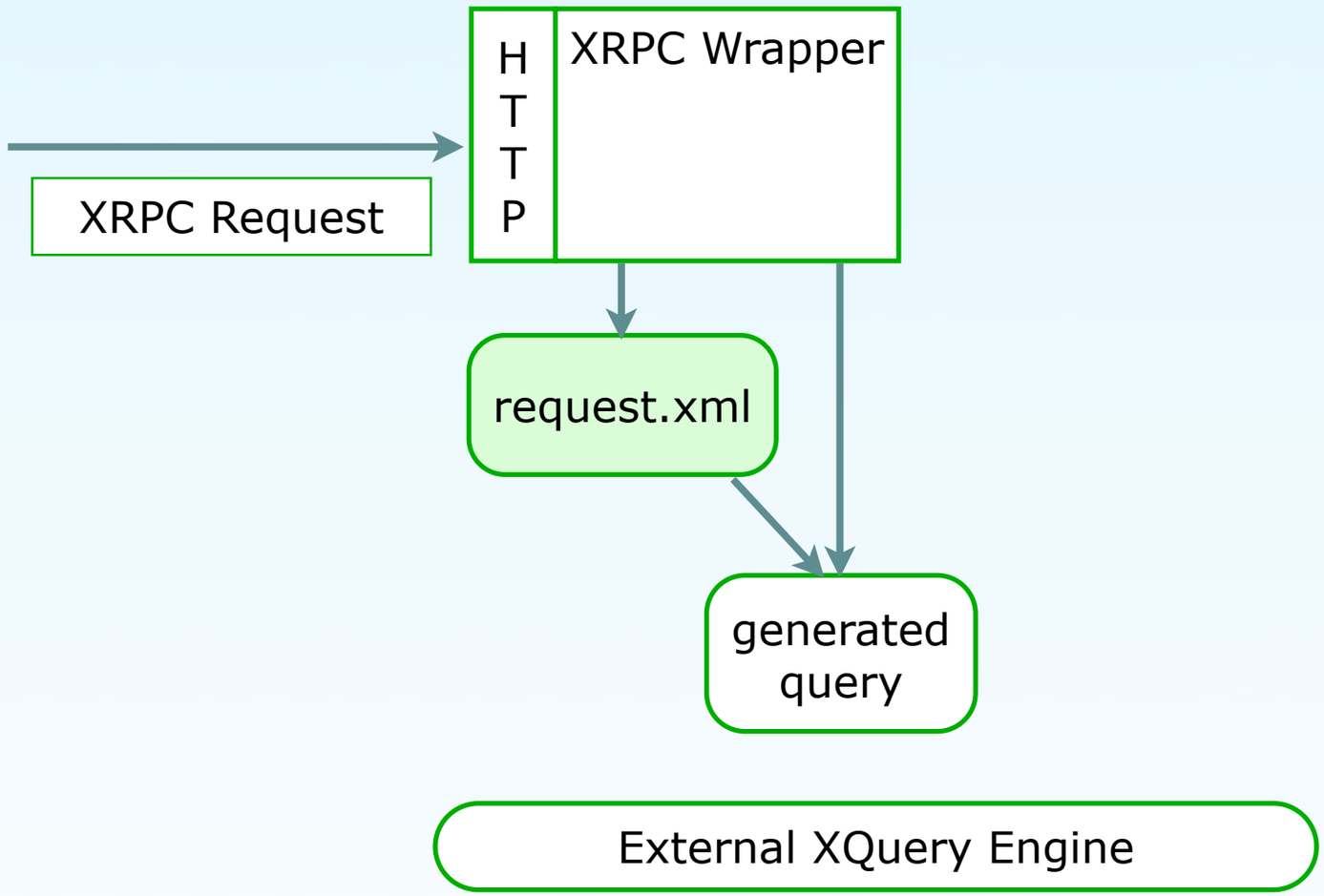
request.xml

```
...
<xrpc:call>
  <xrpc:sequence>
    "person1"

<xrpc:call>
  <xrpc:sequence>
    "person3"
...
```

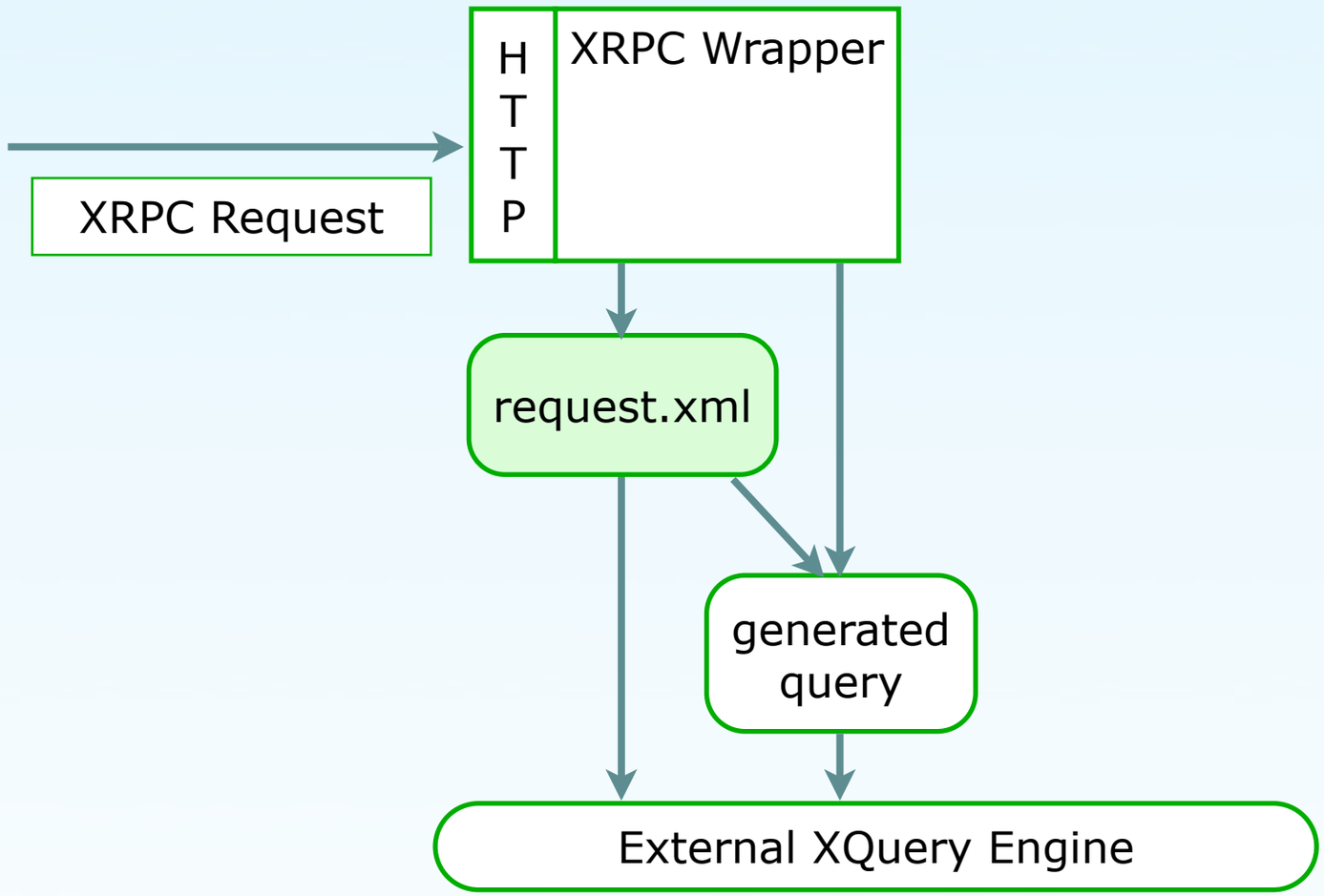
auctions.xml

```
...
<closed_auction>
  <price>...
  <buyer "person1">

<closed_auction>
  <price>...
  <buyer "person3">
...
```

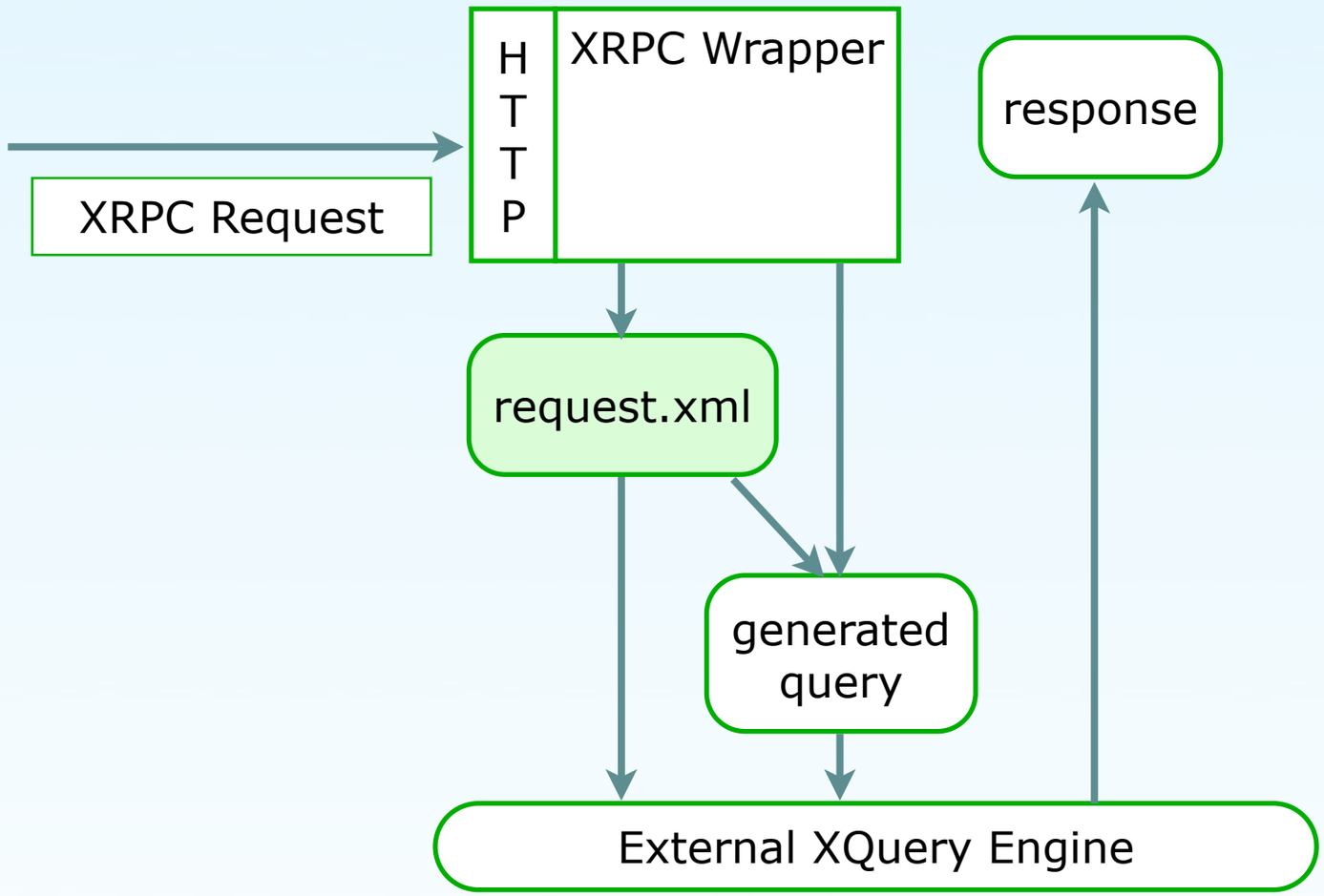
XRPC Wrapper Inside



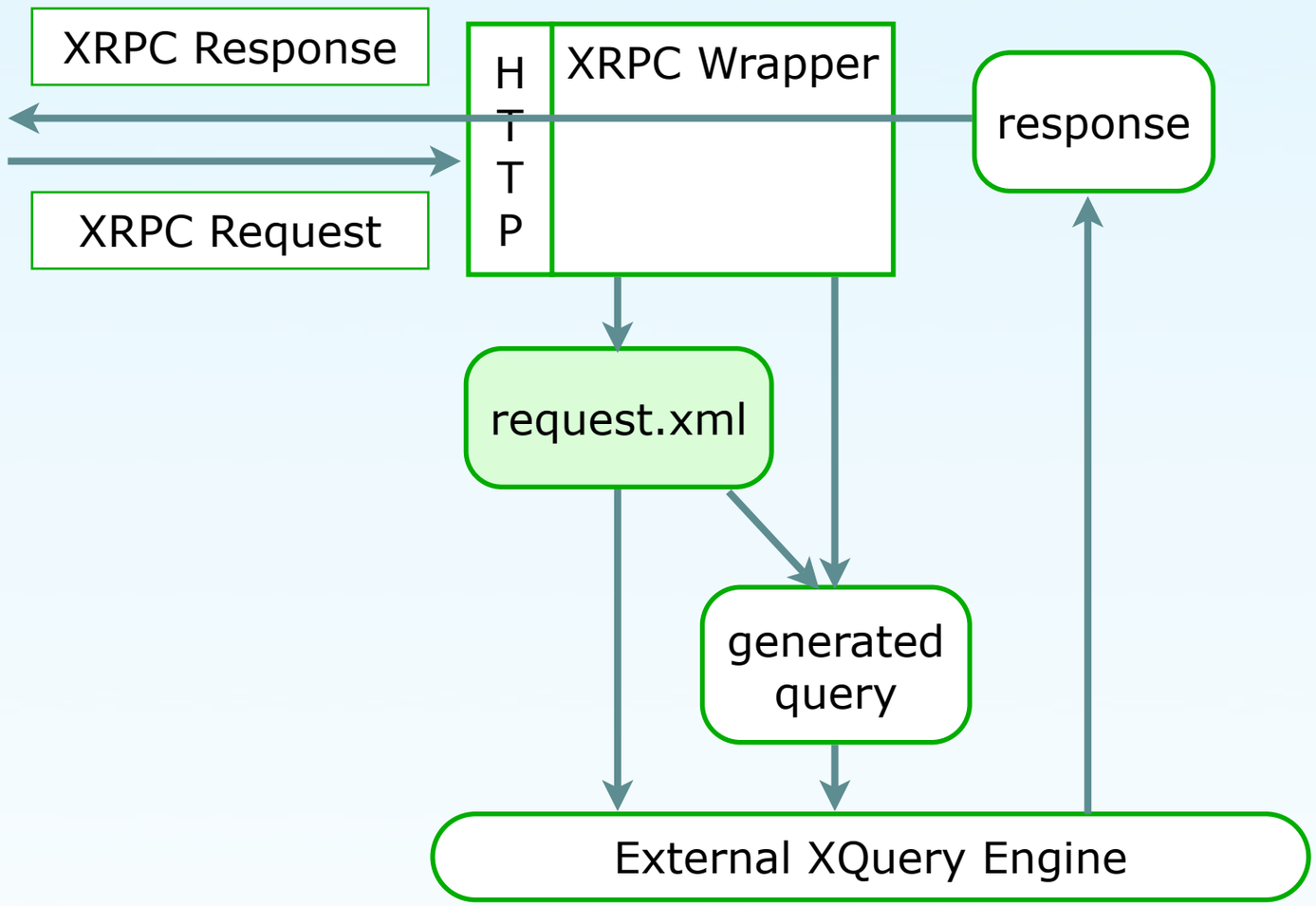
XRPC Wrapper Inside



XRPC Wrapper Inside



XRPC Wrapper Inside



Selection ➠ Join

	Total
getPerson $x = 1	**4276**
getPerson $x = 1000	**8167**

Query time XRPCWrapper/Saxon (msec)

```
getPerson($pid):

doc("xmark.xml")//person[@id = $pid]
```

Conclusion

- XRPC is a clean ***orthogonal*** extension of XQuery
  - Full XDM support and XQuery Update Facility
- It aims for ***interoperability*** in Distributed XQuery
  - Open SOAP based network protocol
  - Different XQuery engines can cooperate
- XRPC can be ***efficient*** thanks to Bulk RPC
  - Optimizes network & exposes bulk opportunities
- With XRPC, distributed XQuery can be ***simple***