SINN and XQuery:
Results and Implementation

Thomas Severiens
Thomas.Severiens@ISN-Oldenburg.de
Michael Schlenker
Michael.Schlenker@ISN-Oldenburg.de
Institute for Science Networking Oldenburg GmbH

Content

- Information Sources and Retrieval Mechanisms
- Query-Language
- Searching for Physics
- Distributed Network
- User Benefit
- Implementation DXQ – Structure
- Implementation DXQ – User-Interface
- Implementation DXQ – Examples

Information Sources and Retrieval Mechanisms

- Google: Fulltext Search on Distributed, Online, Free Information
- PhysNet: Fulltext Search on Distributed, Professional, Online, Free Information
- PhysDoc: Fulltext Search on Distributed, Professional, Online, Free Publications (Articles, PrePrints, ...)
- Inspec, Abstract-Services, Publishers, etc.: Metadata- & Abstract Search on (Distributed), Professional, (Online), Publications

Information Sources and Retrieval Mechanisms

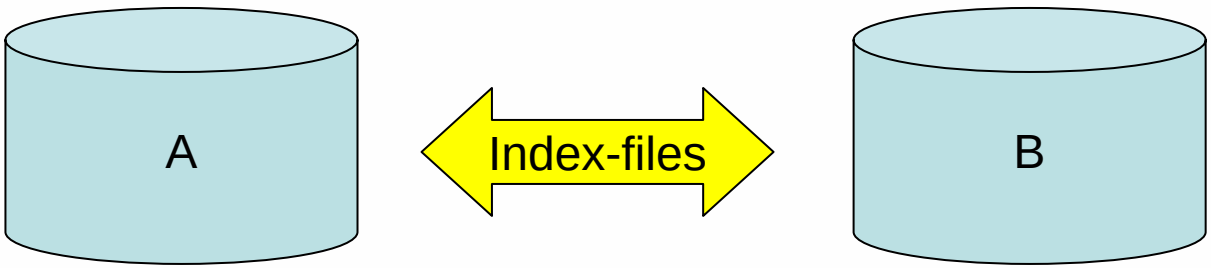
- Google: Simple Search, easy to use, not optimized for structured search
- PhysNet: Simple Search, easy to use, structured search not implemented
- PhysDoc: Structured Search, easy to use, metadata search implemented, booleans
- Inspec, Abstract-Services, Publishers, etc.: Query Language, for professional users, several easier to use web-interfaces
- PhysDoc-SINN: XML-Query, Professional Query Language, as web-service for other applications, e.g. user-interfaces

XML-Query

- Query-Language, optimized for highly structured search on highly structured data (XML).
- Query is XML, Data is XML, Results are XML
- Own datamodel and datatypes (closely leaned upon XML-Schema) (but Schema is buggy, so what to do?)
- Complete programming language
- Was optimized for database-world, could be adopted for necessities of internet-retrieval
- Problems: Namespace-Handling, Casting (solved on Sept. 3rd 2003)

Distributed Searchengines

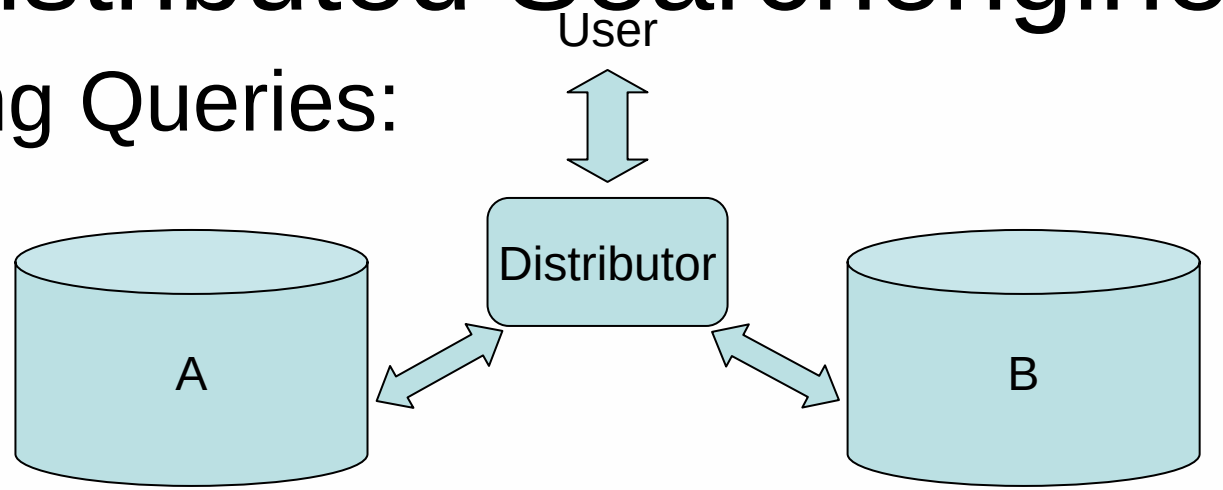
- Sharing Indexes:



A and B have similar content
User may ask A or B, getting similar results
+ For data, which is valid over long periods
- For dynamic data
- Broad bandwidth between A and B required
+ User needs connection to A or B only

Distributed Searchengines

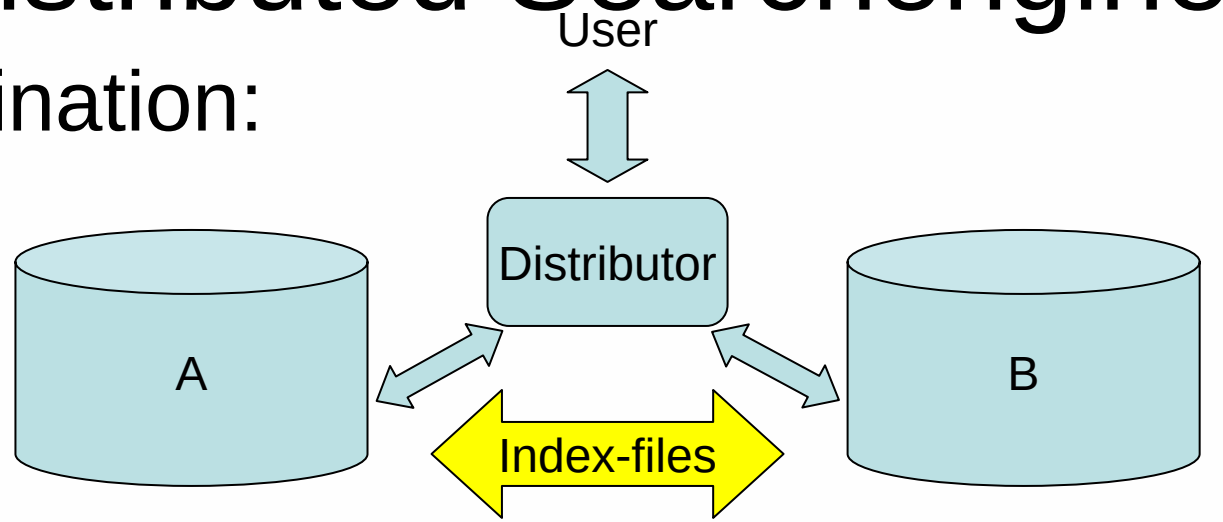
- Sharing Queries:



A and B may have different content
User asks Distributor to distribute queries (agents)
+ For dynamic data
- Results depend on connectivity
+ A and B share computing load
- Problem: ranking, merging algorithm, doublets

Distributed Searchengines

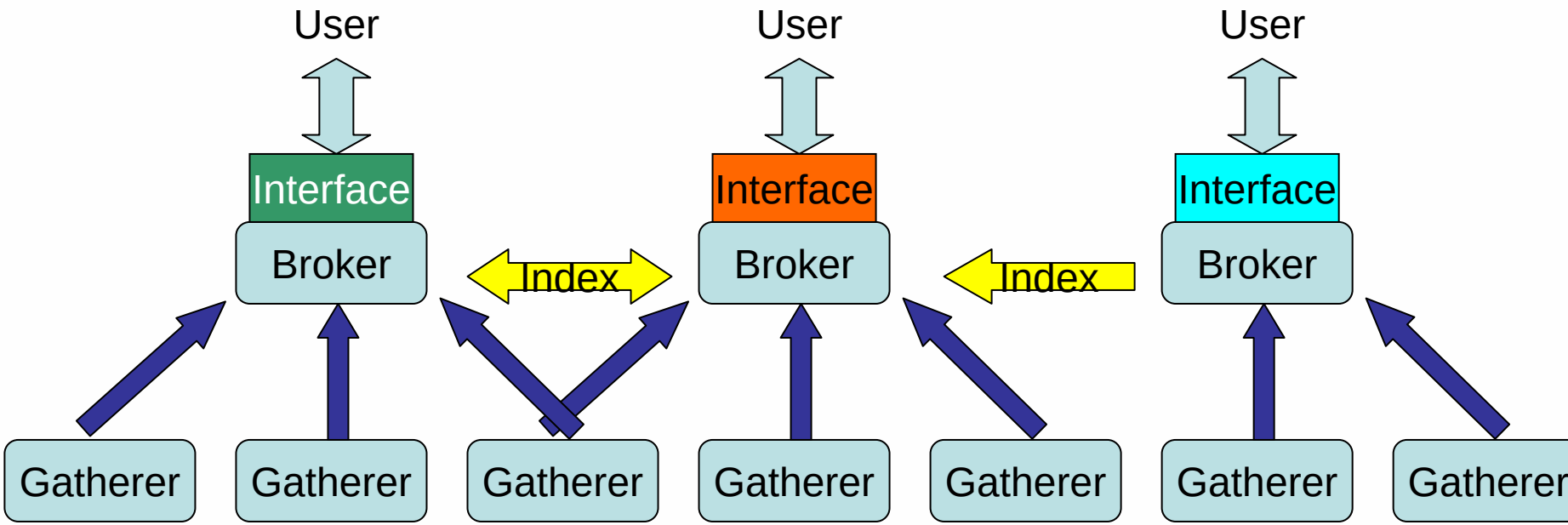
- Combination:



A and B share parts of their index-files, to optimize availability, redundancy of data, computing load of participating servers.
XML-Query allows the user to program merging algorithms, to be executed by the distributor.
XML-Query allows to send complex queries into the system.
Let's scale this model onto PhysDoc…

PhysDoc-Search – today

- Harvest-Software based network of search-engines (without DXQ-Software installed)

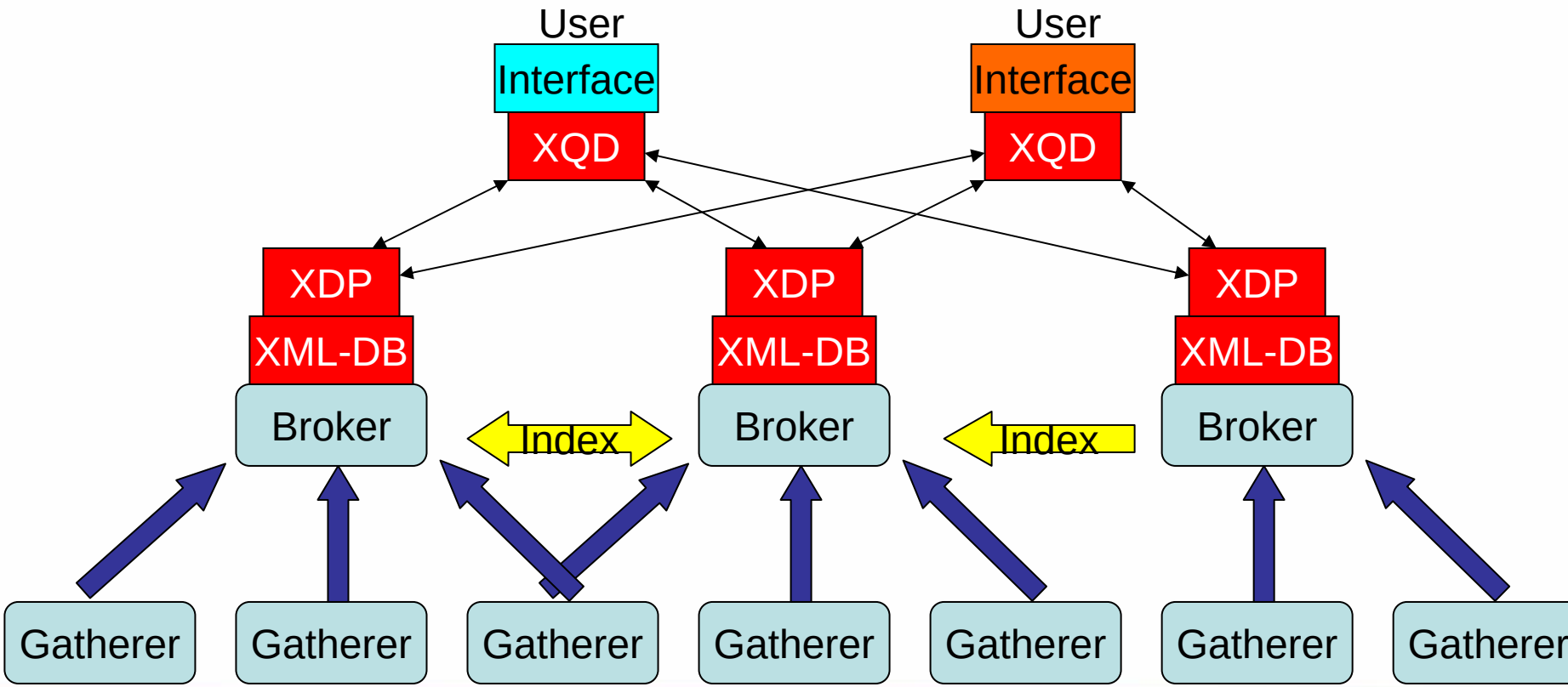


PhysDoc – the next step

- How to re-use the existing network
  - Network of software
  - Network of organizations
  - Network of people
    - Offering work power
    - Offering computer power
- SINN: Use the existing distributed workforce to implement a new, better, more intelligent search facility.

PhysDoc-Search – Step 1

- All software for step 1 is ready for implementation!

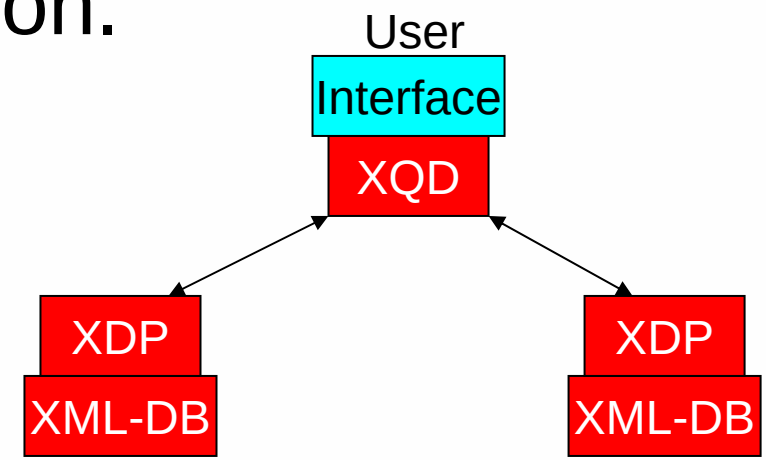


DXQ – Benefit for the User

- DXQ: „Distributed XML-Query“
- What are the benefits for the users?
  - Queries may be highly structured
  - XML-structured results
    - Better User-Interfaces possible
  - Same redundancy of data
  - Higher system-performance, due to load-information exchange
  - Reduced local computing load, due to sharing of workforce implemented

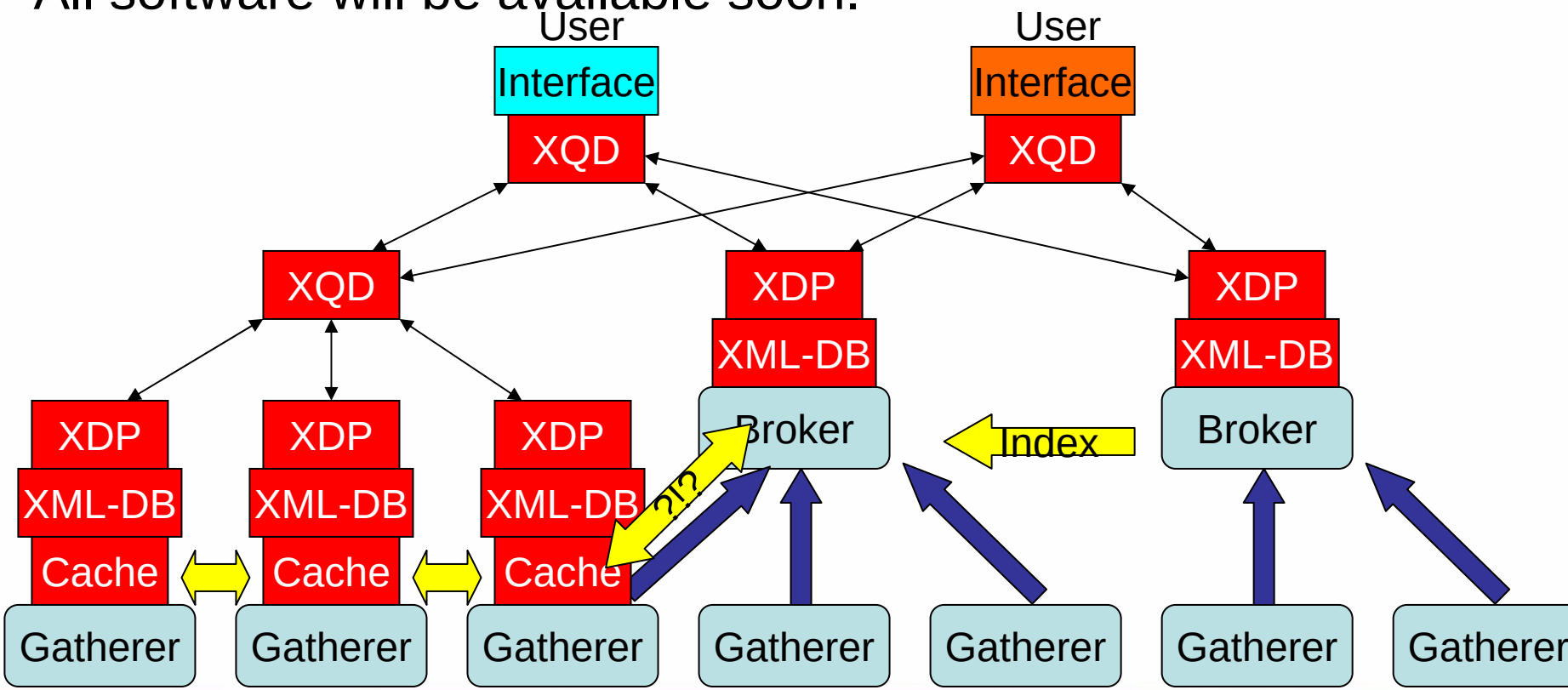
DXQ – A closer view

- For more information on the protocol: arXiv.org/abs/cs.DC/0309022
- XQD (Distributor) and XDP (Provider) exchange queries, results and status information.



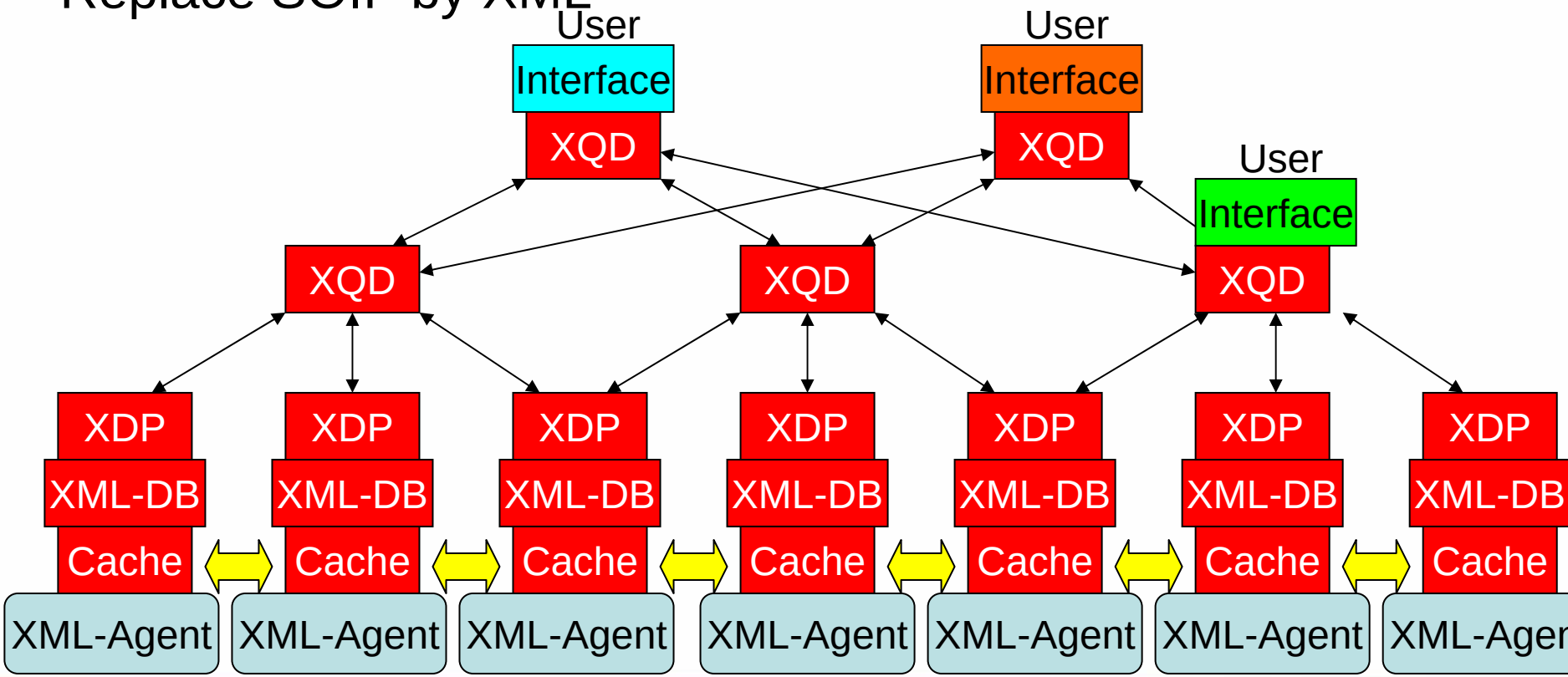
PhysDoc-Search – Step 2

- Most of the software is ready for implementation
- All software will be available soon.



PhysDoc-Search – Step 3

- Much work to do, post-SINN perspective
- Replace SOIF by XML

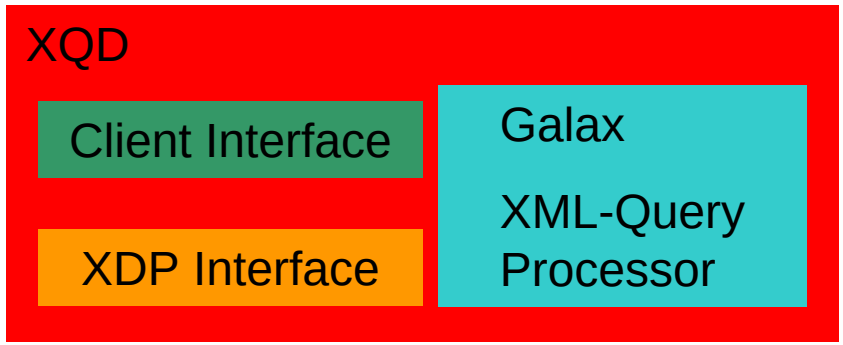


XDP: Problems to be solved

- XML-Database: Choose database, which supports native XML
- XML-Database: Choose database, which supports XML-Query
- XML-Processing results nearly always in very high computing load
- Find work-arounds...

XQD – Implementation

- Handles communication with User Clients
- Handles communication with Data Providers
- Aggregates results via predefined algorithms or user supplied XML-Query programs



Galax XML-Query Processor

- Open Source
- Provides various easy to use language bindings (C, Java, OCaml)
- XML-Projection feature to reduce memory consumption

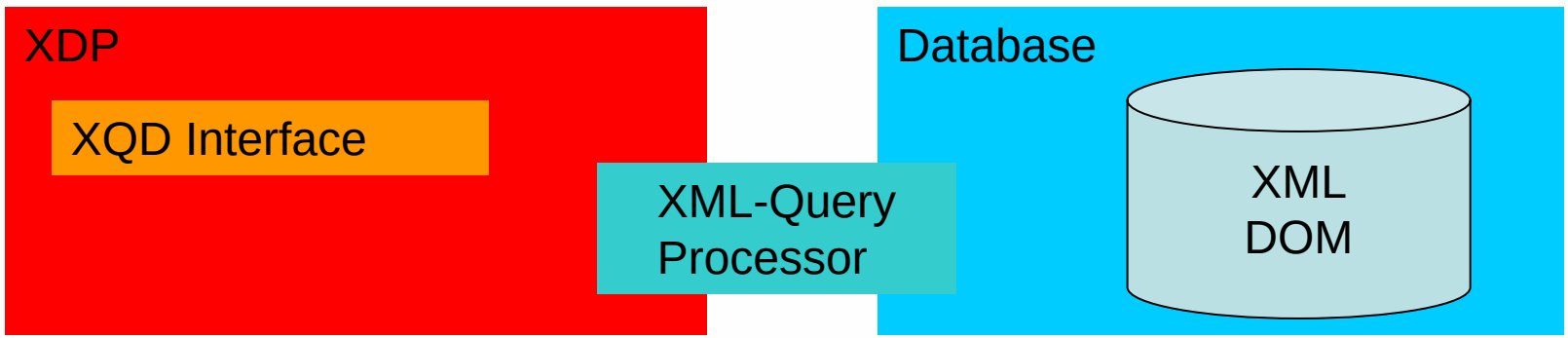
Galax

XML-Query Processor

http://db.bell-labs.com/galax

XDP- Implementation

- Communicates with XQD via DXQP
- Provides XML-Query interface to the database or uses an existing XML-Query interface



XML–DOM memory problems

- XML Document Object Model (DOM) uses large amounts of memory, especially most Java libraries
  - Jdom: 25x source xml document
  - Tdom: 3x source xml document
- XML-Query operates on the DOM
- Source xml documents for the search index are in the some hundred megabytes range

Solutions for the Memory Problem

SAX Stream Processing

- Low Complexity
- Document is reparsed for each XML-Query
- Very low memory consumption

Not useful for XML-Query on large documents.

Persistent DOM

- High Complexity
- Document is parsed once into a database
- Medium memory consumption

Usable for XML-Query on large documents.

XDP: Persistent DOM

- Use a database for persistence and efficient storage of the index
- Provide a virtual DOM style access to the database
- Plug the virtual DOM into the XML-Query processor
- Virtual DOM support for Galax is in current development

DXQ Client Implementation

- Provide functionality to send queries into the DXQ network
- Provide functionality to introspect XQDs
- Handle the DXQ protocol details for the user

DXQ Implementations

- C and Tcl based client implementations are available, with simple UI examples
- A C based XQD implementation is available using Galax as query processor
- A C based XDP implementation is available using Galax as query processor

DXQ Protocol

- DXQP is a message based protocol
- DXQP can be implemented via any message exchange mechanism (HTTP, Sockets, SMTP, ...)
- DXQ is Unicode based, so non-US character sets are supported

DXQP Message Example

```
DXQP-1.0 XML-QUERY
Msg-From: dxqp://metasearch.isn-oldenburg.de/dxq-xqd/
Msg-To: dxqp://physnet-mirror.isn-oldenburg.de:8750/
Transaction-ID: 1
Content-Length: 23

let $a := .//author return $a
```

DXQ Tcl Client Basic Example

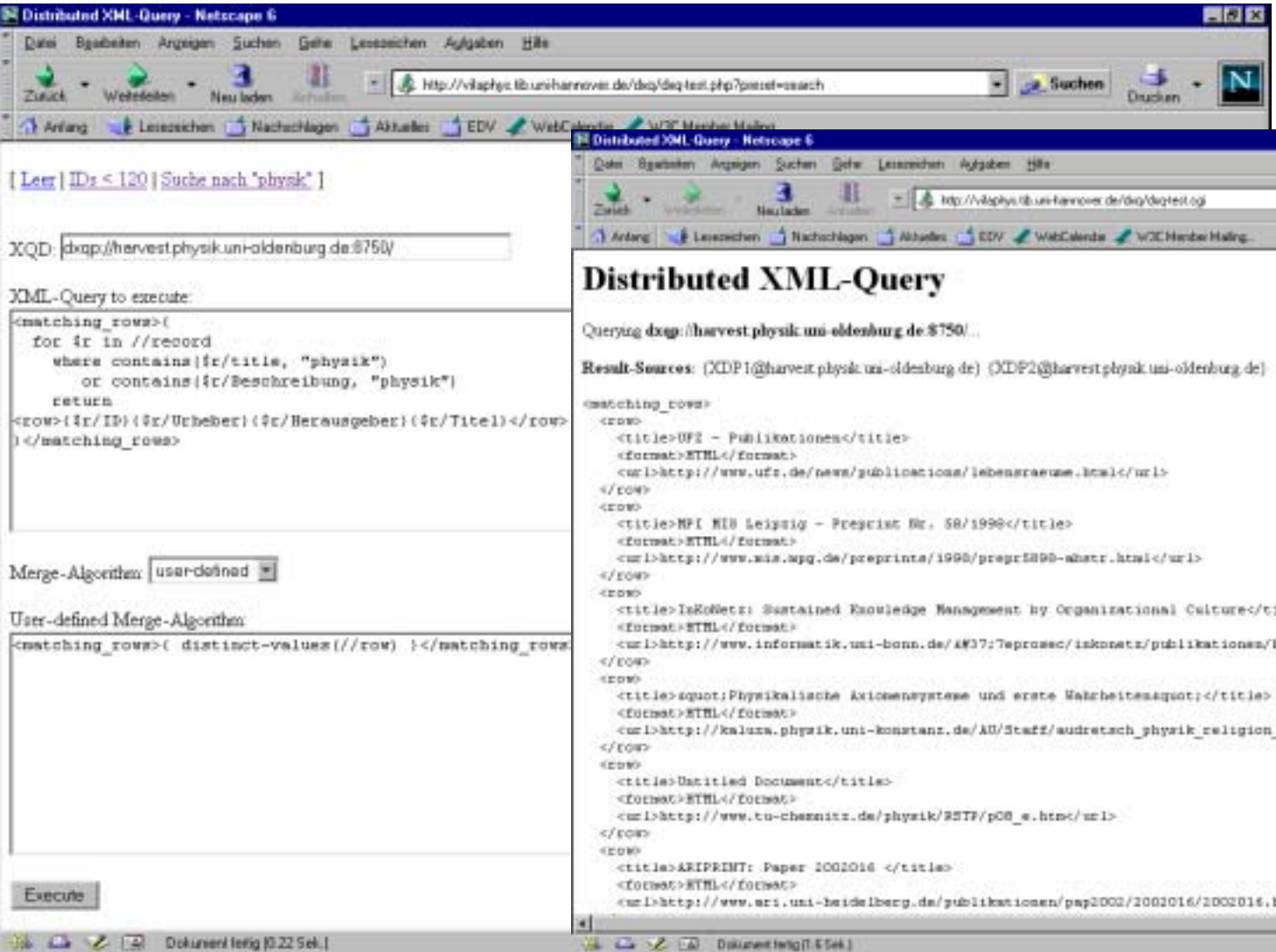
```
package require Tcl 8.4
package require ::dxqp::client
package require ::dxqp::tcp-transport

set c [::dxqp::client::DXQClient]
set t ::dxqp::tcp-transport::transport
set xqd dxqp://harvest.physik.uni-oldenburg.de:8750/

set query {<result>\{ for $r in //row where $r/ID < 2 return $ \}</result>}

puts [$c queryXQD $t $xqd $query concatenate]
```

DXQ C Client Web UI



Thank you for your Attention

Thomas Severiens
Thomas.Severiens@ISN-Oldenburg.de
Michael Schlenker
Michael.Schlenker@ISN-Oldenburg.de

For DXQ-Protocol: arXiv.org/abs/cs.DC/0309022
For the DXQ-Software: www.isn-oldenburg.de/projects/SINN/
For XML-Query: www.w3c.org/XML/Query