CLAROS

Some practical experiences with mid-scale RDF data stores

Graham Klyne
Image Bioinformatics Research Group
Zoology Department

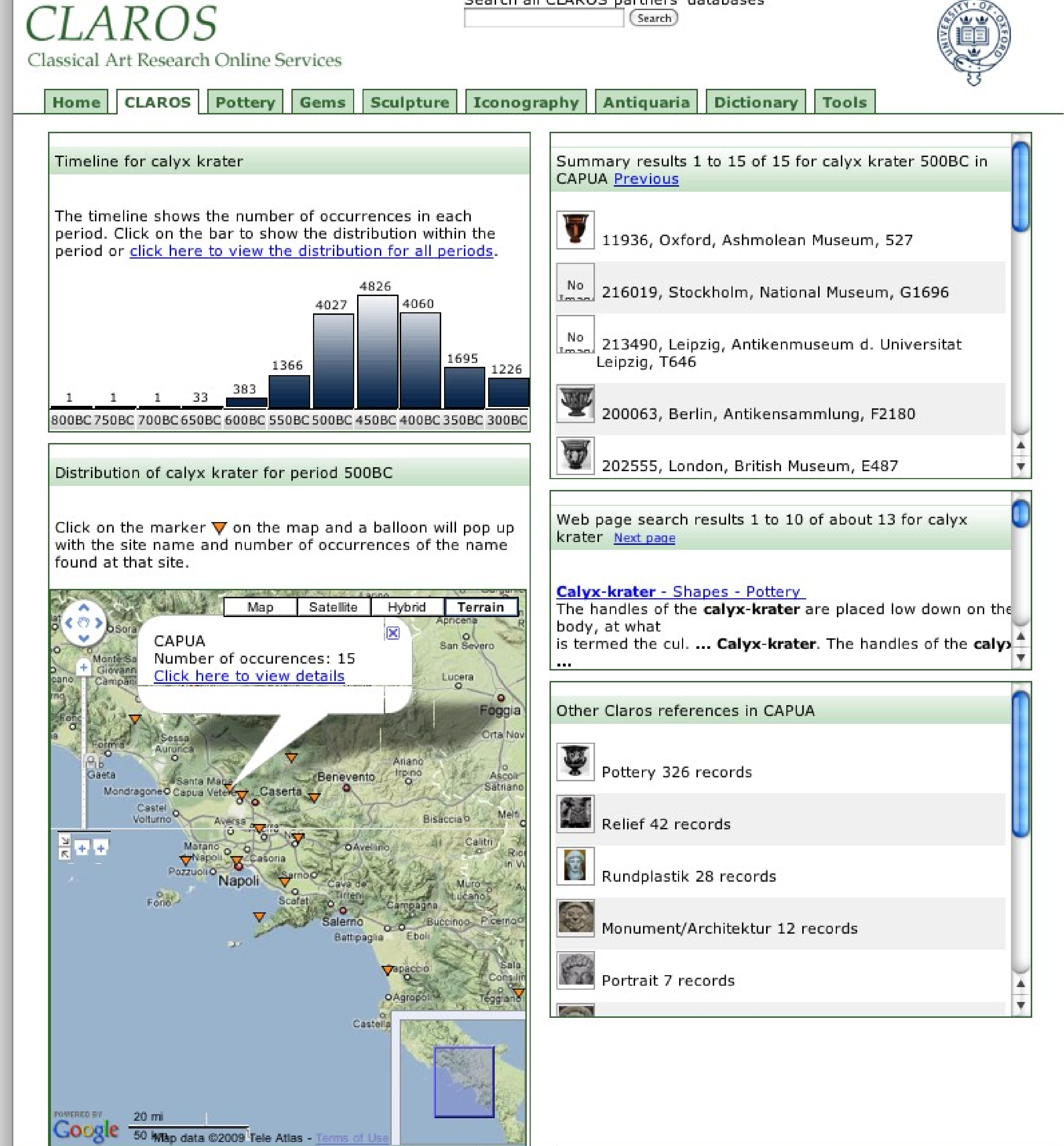
Outline

- Introducing CLAROS and its underlying construction
- Data volumes, complex queries and performance
- Query details
- Current performance-related work

The *CLAROS Explorer*

An example front-end

To the CLAROS data web



Timeline for calyx krater

The timeline shows the number of occurrences in each period. Click on the bar to show the distribution within the period or click here to view the distribution for all periods.

Distribution of calyx krater for period 500BC

Click on the marker on the map and a balloon will pop up with the site name and number of occurrences of the name found at that site.

Summary results 1 to 15 of 15 for calyx krater 500BC in CAPUA Previous

- 11936, Oxford, Ashmolean Museum, 527
- No Image 216019, Stockholm, National Museum, G1696
- No Image 213490, Leipzig, Antikenmuseum d. Universitat Leipzig, T646
- 200063, Berlin, Antikensammlung, F2180
- 202555, London, British Museum, E487

Web page search results 1 to 10 of about 13 for calyx krater Next page

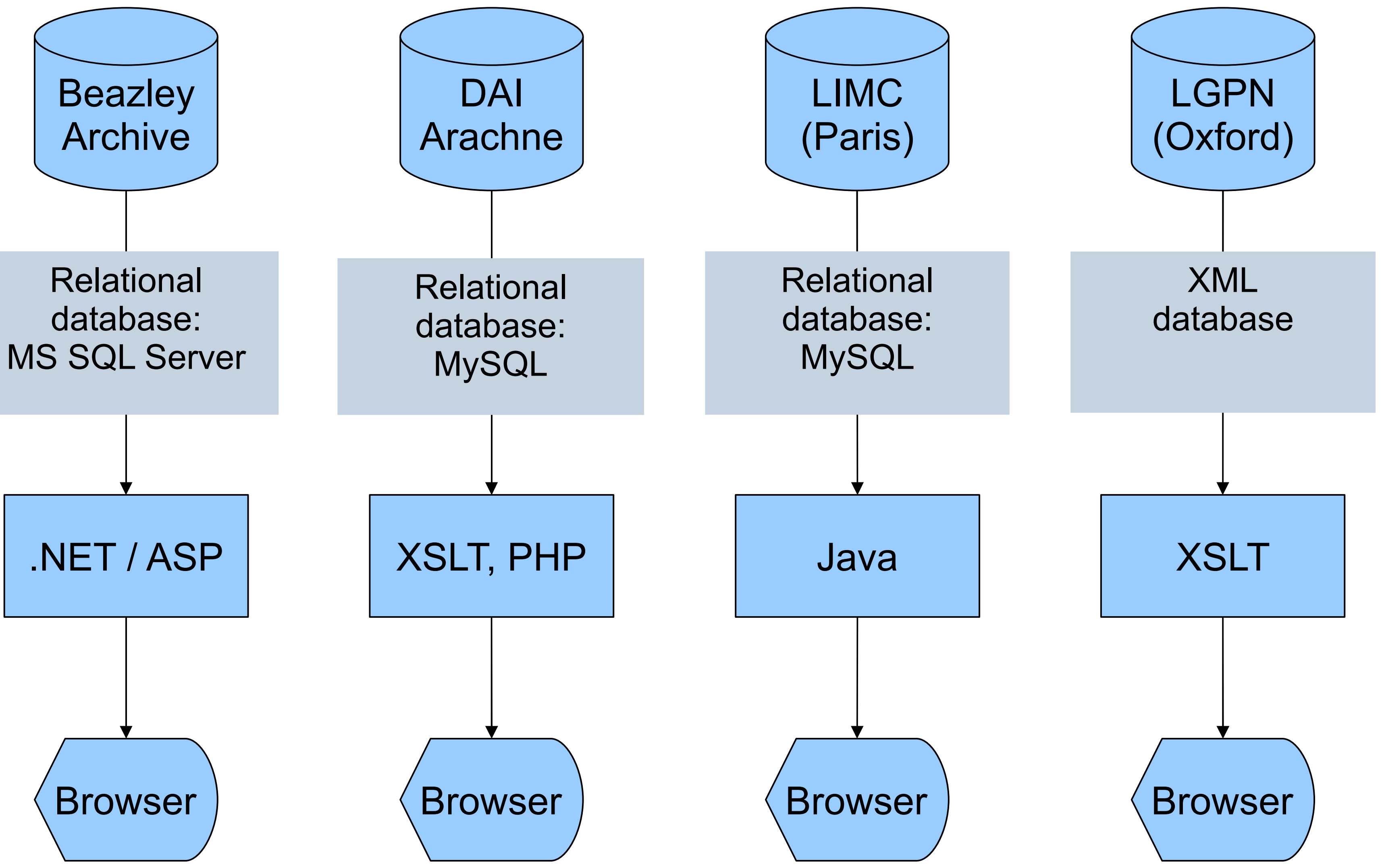
Calyx-krater - Shapes - Pottery

The handles of the **calyx-krater** are placed low down on the body, at what is termed the cul. ... **Calyx-krater**. The handles of the **caly** ...

Other Claros references in CAPUA

- Pottery 326 records
- Relief 42 records
- Rundplastik 28 records
- Monument/Architektur 12 records
- Portrait 7 records

Contributing Partner Systems



Beazley Archive

Relational database: MS SQL Server

.NET / ASP

Browser

DAI Arachne

Relational database: MySQL

XSLT, PHP

Browser

LIMC (Paris)

Relational database: MySQL

Java

Browser

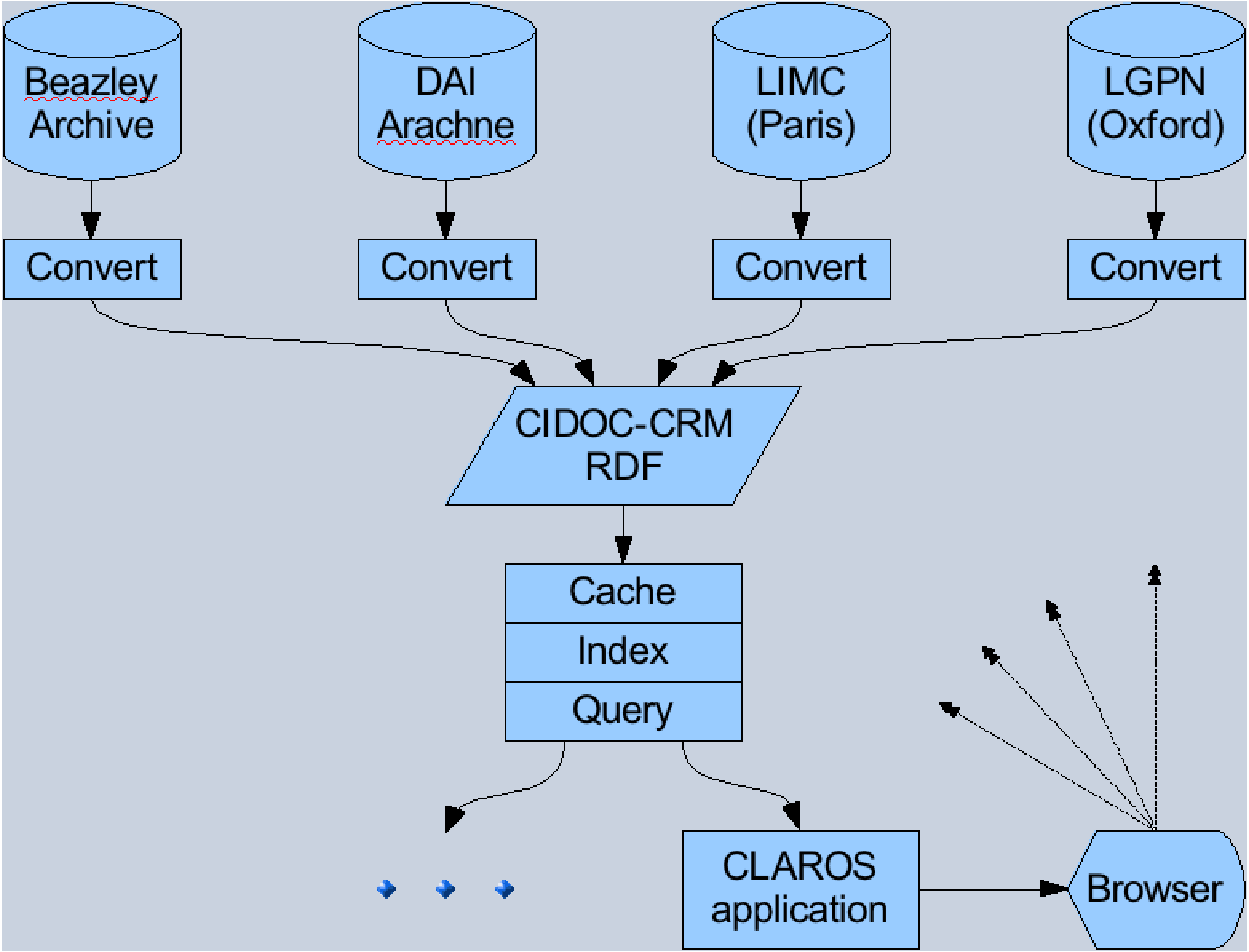
LGPN (Oxford)

XML database

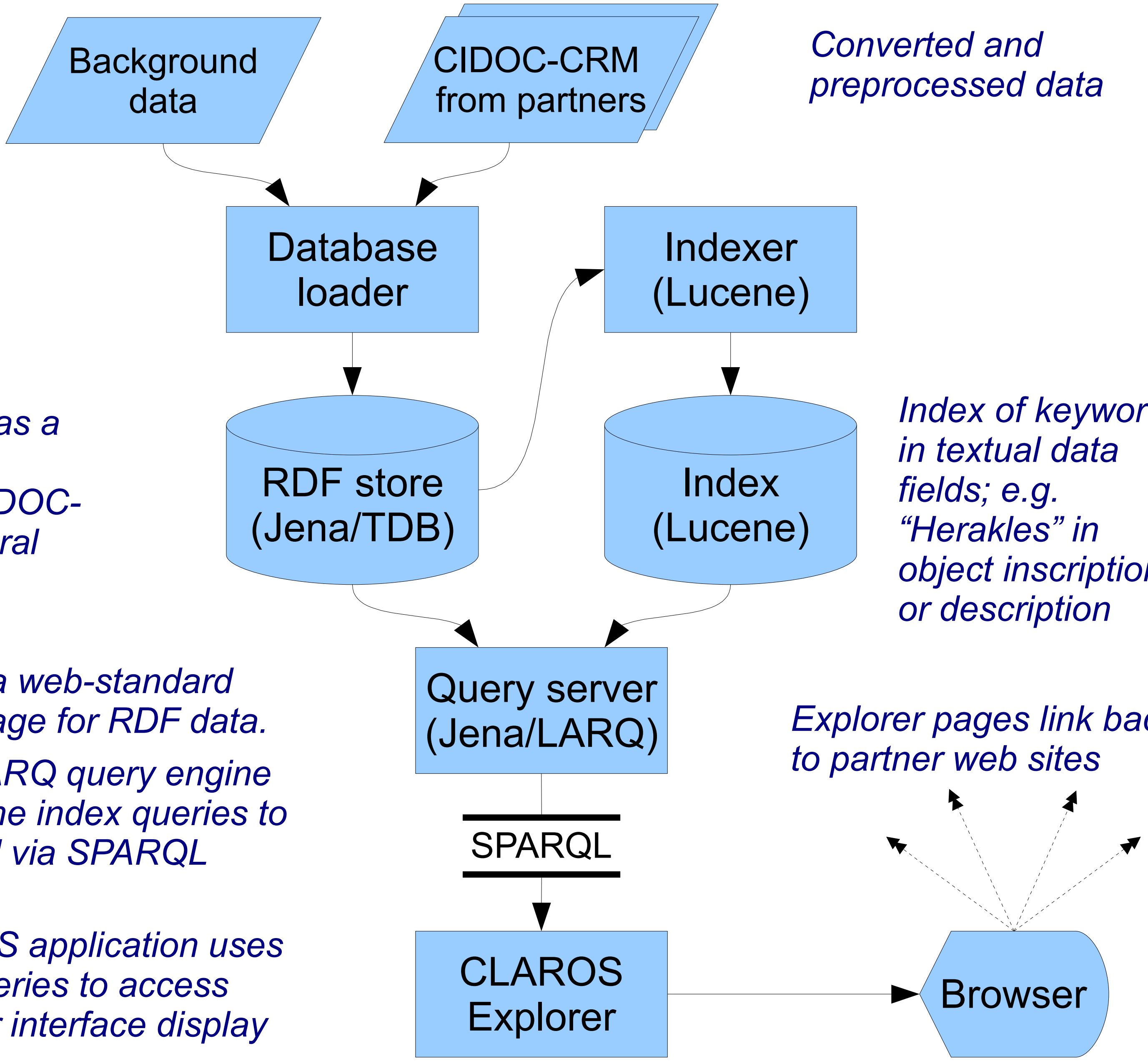
XSLT

Browser

CLAROS data web Components



Claros Explorer VRE back-end elements



Background data

CIDOC-CRM from partners

Database loader

Indexer (Lucene)

RDF store (Jena/TDB)

Index (Lucene)

Query server (Jena/LARQ)

SPARQL

CLAROS Explorer

Browser

E.g., dates for named eras used in Arachne

Converted and preprocessed data

Also known as a “triple store”. Captures CIDOC-CRM structural information

Index of keywords in textual data fields; e.g. “Herakles” in object inscription or description

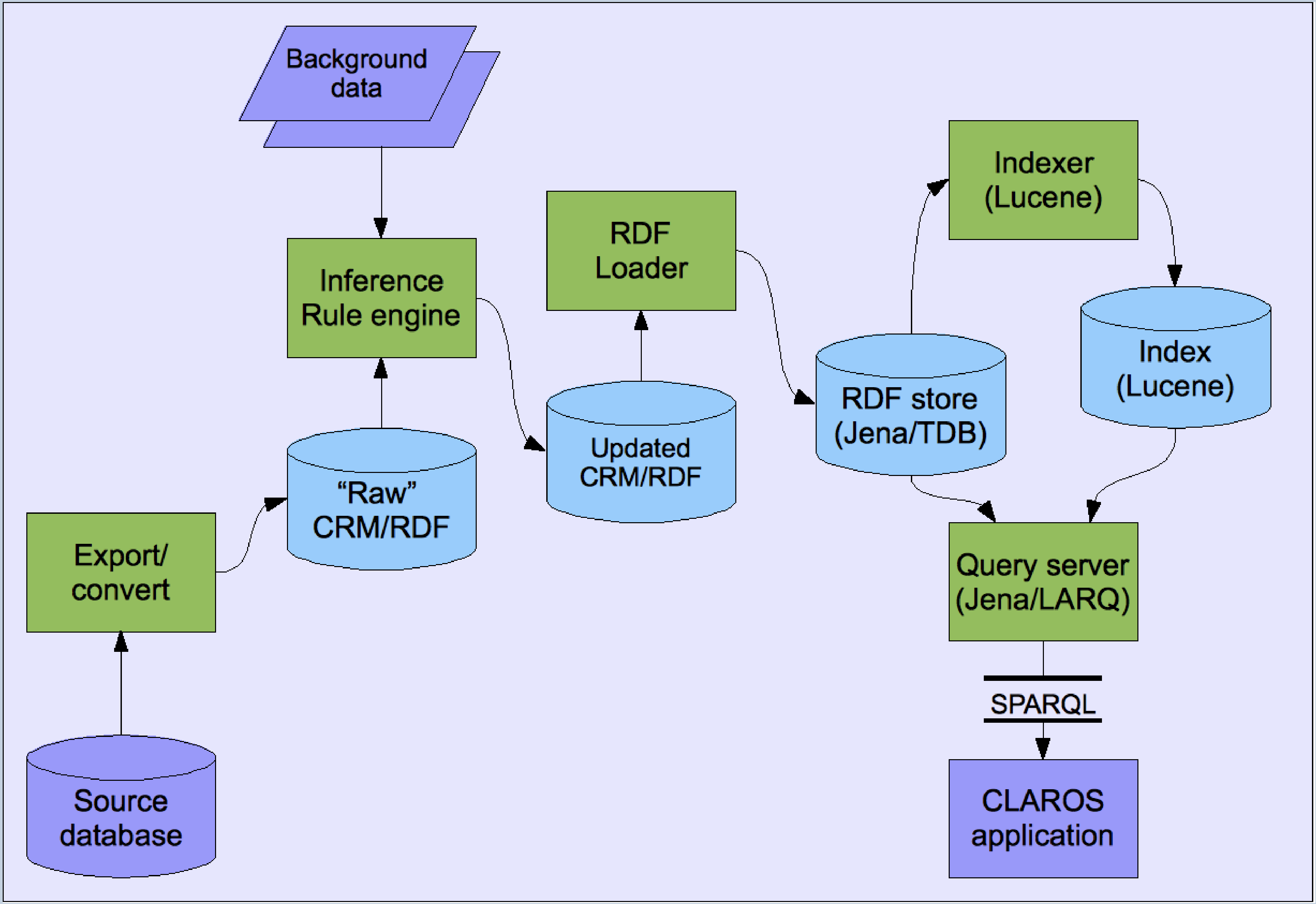
SPARQL is a web-standard query language for RDF data.

The Jena/LARQ query engine allows Lucene index queries to be accessed via SPARQL queries.

The CLAROS application uses SPARQL queries to access data for user interface display

Explorer pages link back to partner web sites

The CLAROS Processing Tool Chain



Making CLAROS

- The main project was conducted over 8 months with two part-time developers
- Almost no new code in run-time system
- Main work areas (comparable effort for each):
    - Analyzing data sources and building consensus for CRM vocabulary use
    - Select, configure, deploy and test software platform
    - Convert source data, load, test, tune
    - Design user presentation
    - Design and test SPARQL queries

Free text keyword queries

- Our development style is incremental, starting initially with a simple Google-style search over the data, based on LARQ integration of Lucene in SPARQL queries
- Getting the CRM vocabulary usage exactly right in the early stages was not critical
  - discussion of CRM use patterns was guillotined when we needed to move the project forward
- I believe this has been a key factor in allowing us to achieve what we did in the time available

But is it Linked Data?

- Not really, but...
- CLAROS is an example of an application that might be built using linked data, i.e. a “curated triple store”:

“I think the answer (for the moment at least) is to forget about querying the entire web of linked data and focus on supporting the easy creation of targeted, curated, triple stores that each incorporate a useful subset of the linked data that’s out there.”

Jeni Tennison: http://www.jenitennison.com/blog/node/143

- This description pretty well captures what we have been calling a “data web”

Data volumes

- Information on 100-200K entities
  - objects, people, places
- About 10 million RDF triples
  - Modest compared with large scale RDF stores
  - But still larger than “toy-scale”
- Currently:
  - 5.6Gb triple store data and indexes
  - 100Mb Lucene free-text index
  - Created from about 1.7Gb RDF source data

Hardware

- Virtual machine (not ideal, but workable)
  - NAS storage better than local virtual disks
- 3Gb RAM (2Gb was not quite enough)
  - Guide: 30% database size, including all indexes
- Performance for simple queries is quite reasonable, once the index cache has been primed - sub-second
  - Initial queries can be slow: e.g. 10's of seconds
  - Have not yet performed intensive load tests, or with multiple concurrent users

Complex queries

While data volumes may be modest,
many queries are quite complex

E.g. This query for detailed results for object type, provenance and date,

with range selection to allow paging through results

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  :
SELECT * WHERE
{ GRAPH ?g
  { { ?s rdf:type crm:E22.Man-Made_Object ;
        crm:P2.has_type
          [ rdf:type crm:E55.Type ;
            crm:P127.has_broader_term claros:ObjectType ;
            rdf:value  "Relief"  ;
          ] .
    }
    OPTIONAL {
      {
        ?s crm:P108I.was_produced_by
          [ rdf:type crm:E12.Production ;
            crm:P4.has_time-span
              [ rdf:type crm:E52.Time-Span ;
                crm:P82.at_some_time_within
                  [ rdf:type crm:E61.Time_Primitive ;
                    claros:not_before ?early ;
                    claros:not_after  ?late ;
          ]   ]   ]
    } }
    FILTER ( ( xsd:dateTime(arqfn:YearToDateString(?early)) <=
               xsd:dateTime("-0275-01-01T00:00:00") ) )
    FILTER ( ( xsd:dateTime(arqfn:YearToDateString(?late)) >=
               xsd:dateTime("-0625-01-01T00:00:00") ) )
    ?s crm:P16I.was_used_for
      [ rdf:type crm:E7.Activity ;
        crm:P2.has_type claros:Event_FindObject ;
        crm:P7.took_place_at ?loc ; ]
    ?loc rdf:type crm:E53.Place ; crm:P87.is_identified_by
      [ rdf:type crm:E48.Place_Name ;
        rdf:value ?nam ] .
      FILTER regex( ?nam,  "CAPUA" , 'i')
    { ?s rdf:type ?t . }
    { ?s crm:P102.has_title [ rdf:type crm:E35.Title ; rdf:value ?desc ] }
    { ?s crm:P70I.is_documented_in ?link }
  }
} ORDER BY ?g ?s ?lit OFFSET 0 LIMIT 5
```

Query patterns

- Preamble
- Generate
- Filter
- Extract
- Postamble

(the main body is similar to list comprehensions in Python or functional languages)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  :
SELECT * WHERE
{ GRAPH ?g

  { { ?s rdf:type crm:E22.Man-Made_Object ;
        crm:P2.has_type
          [ rdf:type crm:E55.Type ;
            crm:P127.has_broader_term claros:ObjectType ;
            rdf:value  "Relief"   ;
          ] .}

    ?s crm:P16I.was_used_for
      [ rdf:type crm:E7.Activity ;
        crm:P2.has_type claros:Event_FindObject ;
        crm:P7.took_place_at ?loc ; ]
    ?loc rdf:type crm:E53.Place ; crm:P87.is_identified_by
      [ rdf:type crm:E48.Place_Name ;
        rdf:value ?nam ] .
      FILTER regex( ?nam,  "CAPUA" , 'i')

    { ?s rdf:type ?t . }
    { ?s crm:P102.has_title
      [ rdf:type crm:E35.Title ; rdf:value ?desc ] }
    { ?s crm:P70I.is_documented_in ?link }
  }

} ORDER BY ?g ?s ?lit OFFSET 0 LIMIT 5
```

Poorly performing queries >>

- Queries with large intermediate results
- Large UNION queries
- Queries that depend on sorting on secondary keys
    - (a particular case of large intermediate results)

Crude keyword search (230s)

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pf:   <http://jena.hpl.hp.com/ARQ/property#>
PREFIX crm:  <http://purl.org/NET/crm-owl#>
SELECT DISTINCT ?s ?t ?lit ?lnk where
{ ?lit pf:textMatch 'naxos' .
  { ?s ?p1 ?lit . FILTER ( ! isBlank(?s) ) . ?s rdf:type ?t . }
  UNION
  { ?b1 ?p1 ?lit . FILTER ( isBlank(?b1) ) .
    { ?s ?p2 ?b1 . FILTER ( ! isBlank(?s) ) .?s rdf:type ?t . }
    UNION
    { ?b2 ?p2 ?b1 . FILTER ( isBlank(?b2) ) .
      { ?s ?p3 ?b2 . FILTER ( ! isBlank(?s) ) . ?s rdf:type ?t . }
      UNION
      { ?b3 ?p3 ?b2 . FILTER ( isBlank(?b3) ) .
        { ?s ?p4 ?b3 . FILTER ( ! isBlank(?s) ) . ?s rdf:type ?t . }
  } } }
  FILTER( (?t = crm:E22.Man-Made_Object) ||
          (?t = crm:E53.Place) ||
          (?t = crm:E21.Person) )
  OPTIONAL { ?s crm:P70I.is_documented_in ?lnk . }
} LIMIT 250
```

Reducing large intermediate results

- Query clause (sub-patterns) ordering
- Minimize reliance on FILTER expressions
- Apply filters early

UNION keyword search query (4.5s)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX crm: <http://purl.org/NET/crm-owl#>
PREFIX claros: <http://purl.org/NET/Claros/vocab#>
PREFIX pf: <http://jena.hpl.hp.com/ARQ/property#>
SELECT DISTINCT ?g ?s ?t ?lit ?lnk ?desc WHERE
{ GRAPH ?g
  { ?lit pf:textMatch 'naxos' .
    { { { ?title rdf:value ?lit ; rdf:type crm:E35.Title .
          ?s crm:P102.has_title ?title . }
        UNION
        { ?type rdf:value ?lit ; rdf:type crm:E55.Type .
          ?s crm:P2.has_type ?type . }
        UNION
        { ?ident rdf:value ?lit ; rdf:type crm:E42.Identifier .
          ?s crm:P48.has_preferred_identifier ?ident . }
        UNION
        { { { ?pr rdfs:label ?lit . }
            UNION
            { ?employed rdfs:label ?lit ; rdf:type crm:E57.Material .
              ?pr crm:P126.employed ?employed . }
            UNION
            { ?timeprimitive rdf:value ?lit ; rdf:type crm:E61.Time_Primitive .
              ?time crm:P82.at_some_time_within ?timeprimitive ; rdf:type crm:E52.Time-Span .
              ?pr crm:P4.has_time-span ?time . } }
          ?pr rdf:type crm:E12.Production .
          ?s crm:P108I.was_produced_by ?pr . }
        UNION
        { ?note crm:has_PrimitiveString ?lit ; rdf:type crm:E62.String .
          ?feature crm:P3.has_note ?note ; rdf:type crm:E25.Man-Made_Feature .
          ?addition crm:P111.added ?feature ; rdf:type crm:E79.Part_Addition .
          ?s crm:P11  1I.was_added_by ?addition . }
        UNION
        { ?type rdfs:label ?lit ; a crm:E55.Type .
          ?assignment crm:P42.assigned ?type ; a crm:E17.Type_Assignment .
          ?s crm:P14I.was_classified_by ?assignment . }
        UNION
        { ?location rdfs:label ?lit ; a crm:E53.Place .
          ?s crm:P53.has_former_or_current_location ?location . }
        UNION
        { ?image rdfs:label ?lit ; a crm:E38.Image .
          ?s crm:P138I.has_representation ?image . }
        UNION
        { ?document rdfs:label ?lit ; a crm:E31.Document .
          ?s crm:P67I.is_referred_to_by ?document . }
        UNION
        { { { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
              ?pl crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place . }
            UNION
            { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
              ?place crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place .
              ?pl crm:P89I.contains ?place . } }
          ?s crm:P53.has_former_or_current_location ?pl . }
        UNION
        { ?materialname rdf:value ?lit ; a crm:E41.Appellation .
          ?material crm:P1.is_identified_by ?materialname ; rdf:type crm:E57.Material .
          ?s crm:P45.consists_of ?material . } }
      UNION
      { ?personname rdf:value ?lit ; a crm:E82.Actor_Appellation .
        ?s crm:P131.is_identified_by ?personname ; rdf:type crm:E21.Person . }
      UNION
      { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
        ?s crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place . } }
  ?s rdf:type ?t .
  OPTIONAL { ?s crm:P102.has_title [ rdf:type crm:E35.Title ; rdf:value ?desc ] }
  OPTIONAL { ?s crm:P70I.is_documented_in ?lnk }
} LIMIT 250
```

UNION keyword search query (4.5s)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX crm: <http://purl.org/NET/crm-owl#>
PREFIX claros: <http://purl.org/NET/Claros/vocab#>
PREFIX pf:  <http://jena.hpl.hp.com/ARQ/property#>
SELECT DISTINCT ?g ?s ?t ?lit ?lnk ?desc WHERE
{ GRAPH ?g
  { ?lit pf:textMatch 'naxos' .
    { { { ?title rdf:value ?lit ; rdf:type crm:E35.Title .
          ?s crm:P102.has_title ?title . }
        UNION
        { ?type rdf:value ?lit ; rdf:type crm:E55.Type .
          ?s crm:P2.has_type ?type . }
        UNION

         :

      UNION
      { ?personname rdf:value ?lit ; a crm:E82.Actor_Appellation .
        ?s crm:P131.is_identified_by ?personname ; rdf:type crm:E21.Person .
      UNION
      { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
        ?s crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place . } }
  ?s rdf:type ?t .
  OPTIONAL { ?s crm:P102.has_title [ rdf:type crm:E35.Title ; rdf:value ?desc
  OPTIONAL { ?s crm:P70I.is_documented_in ?lnk }
} LIMIT 250
```

Avoiding large UNION queries

- Precalculate values – use a simple inference engine to “materialize” additional RDF properties and query on these

Query with precomputed results (700ms, 95 results)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX crm: <http://purl.org/NET/crm-owl#>
PREFIX claros: <http://purl.org/NET/Claros/vocab#>
PREFIX pf:  <http://jena.hpl.hp.com/ARQ/property#>
SELECT ?g ?t (count(distinct ?s) as ?c) WHERE
{
  GRAPH ?g
  {
    {
      ?lit pf:textMatch ('chios' 50000) .
      ?s claros:hasLiteral ?lit .
      ?s rdf:type ?t .
    }
  }
} GROUP BY ?g ?t
```

```
SELECT DISTINCT ?g ?s ?t ?lit ?lnk ?desc WHERE
{ GRAPH ?g
  { ?lit pf:textMatch 'chios' .
    { { { ?title rdf:value ?lit ; rdf:type crm:E35.Title .
          ?s crm:P102.has_title ?title . }
        UNION
        { ?type rdf:value ?lit ; rdf:type crm:E55.Type .
          ?s crm:P2.has_type ?type . }
        UNION
        { ?ident rdf:value ?lit ; rdf:type crm:E42.Identifier .
          ?s crm:P48.has_preferred_identifier ?ident . }
        UNION
        { { { ?pr rdfs:label ?lit . }
            UNION
            { ?employed rdfs:label ?lit ; rdf:type crm:E57.Material .
              ?pr crm:P126.employed ?employed . }
            UNION
            { ?timeprimitive rdf:value ?lit ; rdf:type crm:E61.Time_Primitive .
              ?time crm:P82.at_some_time_within ?timeprimitive ; rdf:type crm:E52.Time-
Span .
              ?pr crm:P4.has_time-span ?time . } }
          ?pr rdf:type crm:E12.Production .
          ?s crm:P108I.was_produced_by ?pr . }
        UNION
        { ?note crm:has_PrimitiveString ?lit ; rdf:type crm:E62.String .
          ?feature crm:P3.has_note ?note ; rdf:type crm:E25.Man-Made_Feature .
          ?addition crm:P111.added ?feature ; rdf:type crm:E79.Part_Addition .
          ?s crm:P11  1I.was_added_by ?addition . }
        UNION
        { ?type rdfs:label ?lit ; a crm:E55.Type .
          ?assignment crm:P42.assigned ?type ; a crm:E17.Type_Assignment .
          ?s crm:P14I.was_classified_by ?assignment . }
        UNION
        { ?location rdfs:label ?lit ; a crm:E53.Place .
          ?s crm:P53.has_former_or_current_location ?location . }
        UNION
        { ?image rdfs:label ?lit ; a crm:E38.Image .
          ?s crm:P138I.has_representation ?image . }
        UNION
        { ?document rdfs:label ?lit ; a crm:E31.Document .
          ?s crm:P67I.is_referred_to_by ?document . }
        UNION
        { { { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
              ?pl crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place . }
            UNION
            { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
              ?place crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place .
              ?pl crm:P89I.contains ?place . } }
          ?s crm:P53.has_former_or_current_location ?pl . }
        UNION
        { ?materialname rdf:value ?lit ; a crm:E41.Appellation .
          ?material crm:P1.is_identified_by ?materialname ; rdf:type crm:E57.Material .
          ?s crm:P45.consists_of ?material . } }
      UNION
      { ?personname rdf:value ?lit ; a crm:E82.Actor_Appellation .
        ?s crm:P131.is_identified_by ?personname ; rdf:type crm:E21.Person . }
      UNION
      { ?placename rdf:value ?lit ; rdf:type crm:E48.Place_Name .
        ?s crm:P87.is_identified_by ?placename ; rdf:type crm:E53.Place . } }
    ?s rdf:type ?t .
    OPTIONAL { ?s crm:P102.has_title [ rdf:type crm:E35.Title ; rdf:value ?desc ] }
    OPTIONAL { ?s crm:P70I.is_documented_in ?lnk }
} LIMIT 250
```

Sorted secondary key

- Example:
  - Find the earliest dated occurrence of a pot described has having shape “Oinochoe”

Sorted secondary key (5s)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX crm: <http://purl.org/NET/crm-owl#>
PREFIX claros: <http://purl.org/NET/Claros/vocab#>
PREFIX pf:  <http://jena.hpl.hp.com/ARQ/property#>
PREFIX arqfn:  <java:uk.ac.ox.zoo.sparqlite.>
SELECT ?g ?s ?early WHERE
{ GRAPH ?g
  { { ?lit pf:textMatch  "Oinochoe"  . }
    { ?s claros:hasLiteral ?lit . }
    { ?s crm:P14I.was_classified_by
          [ rdf:type crm:E17.Type_Assignment ;
            crm:P42.assigned
              [ a crm:E55.Type ; crm:P127.has_broader_term claros:Shape ;
                rdfs:label ?lit ] ] }
    { ?s crm:P108I.was_produced_by
        [ rdf:type crm:E12.Production ;
          crm:P4.has_time-span
            [ rdf:type crm:E52.Time-Span ;
              crm:P82.at_some_time_within
                [ rdf:type crm:E61.Time_Primitive ;
                  claros:not_before ?early ;
                ] ] ] } }
} ORDER BY ASC (arqfn:YearSortingString(?early)) LIMIT 1
```

MILARQ: The Problem

- Some queries are very slow
  - So much so that we can't yet make CLAROS public
  - e.g.

    Find the earliest known occurrence of a greek vase with the shape/style known as “Oinochoe”
- With typical “naive” triple-store indexing the system has to find ***all*** occurrences of “Oinochoe”, and sort them by date in memory

The MILARQ project

- In contrast with a typical triple-store, which is essentially schema-free, a RDB solution would use multiple schema-defined indexes
- LARQ and SPARQLite is being extended to support multiple configurable Lucene indexes
- E.g. an index by (shape, date) could directly access the earliest “Oinochoe”
- This is a simple, pragmatic solution for some practical triple-store performance problems

MILARQ: Problem Response

- Augment the triple store with an index on the composite key (shape, date)
- Associate the new index with a new RDF property with tuple-values domain
  - (This is essentially what LARQ already does for free-text searching using Lucene indexes)
- Results for a given shape are returned from the index in date order
- Enhance the ARQ query processor to use index ordering to handle ORDER BY queries

Beyond MILARQ

- Specialized indexes (e.g., geospatial, shape-similarity)

Other approaches

- Automatic query analysis and supplementary index creation
- Reasoner-supported query planning and optimization

Conclusions – our experience

- Combining RDF queries with free-text searches has been invaluable (probably essential) for CLAROS
    - Especially in allowing incremental refinement of the vocabulary used, rather than full up-front definition
- Schema-free RDF store is great for flexibility of data and queries, but can present performance problems for some queries
- Complex queries can present a different problem to very large data sets
    - Some research groups are working on general query performance improvement
    - Meanwhile, ad hoc approaches can be used to deal with performance of specific queries

Acknowledgements

- Andy Seaborne (HP Labs and Talis), for the RDF store (TDB) and query engine (ARQ, LARQ), and also for much helpful advice along the way
- Robert Kummer (Köln University)
- Sebastian Rahtz (OUCS)
- Donna Kurtz, Greg Parker (Beazley Archive)