EXRT:
Towards a Simple Benchmark for XML Readiness Testing

Michael Carey, Ling Ling,
Matthias Nicola*, and Lin Shao

UC Irvine
**IBM Corporation*

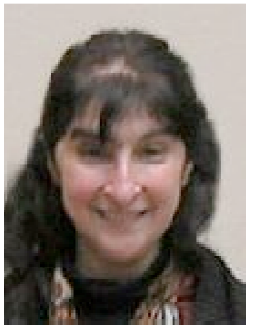
http://isg.ics.uci.edu

TPCTC 2010
Singapore

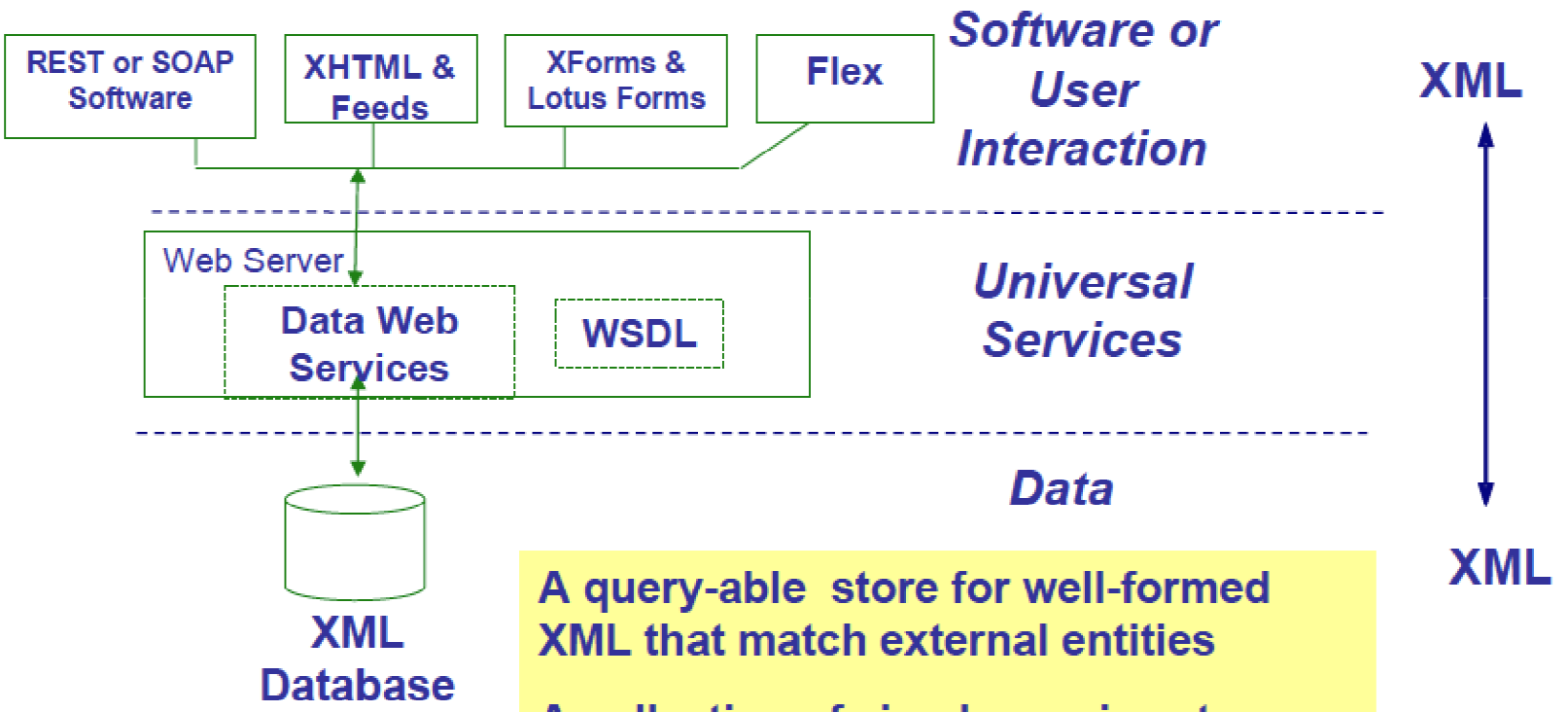
XML (in the Enterprise)

- Early roots in document markup
  - SGML → XML
- Now widely used in enterprise data scenarios
  - Internet-based information exchange
  - Web services and SOA infrastructure
- Result is a strong (emerging) need to store and query large collections of XML documents
  - Data retention and sharing
  - Auditing and compliance requirements
  - Simpler, more flexible database design

“It’s Time for XML End-to-End!”



– Susan Malaika, IBM



A query-able store for well-formed XML that match external entities

A collection of simple services to manipulate the stored XML

“XQuery is the Answer!”

– Dana Florescu, Oracle

- Claim: RDBMS support for XML and XQuery has matured to where it can/should be relied on
  - Suggested that XML data integration middleware should move to pushing XQuery (or SQL/XML) to RDBMSs, instead of SQL, for data retrieval
  - Suggested that XML-typed columns ought to be used for XML storage and caching
- Related beliefs
  - Time to seriously look at binary XML transport
  - Time to center a “no more tiers” approach to Web applications around XML in the back and front ends

UCIrvine
http://isg.ics.uci.edu

So … Are They Right?

- XML/XQuery support seems to be getting there
  - XML support provided by most major RDBMSs
    - SQL Server, DB2, Oracle, Sybase, …
  - Native XML DBMSs are available as well
    - Tamino, MarkLogic, EMC xDB (X-Hive), …
- In particular, native XML support is becoming common in commercial RDBMS systems
  - From tables + shredding to XML columns + XQuery
  - Native storage formats now co-exist with tabular storage in several XML-enabled RDBMSs

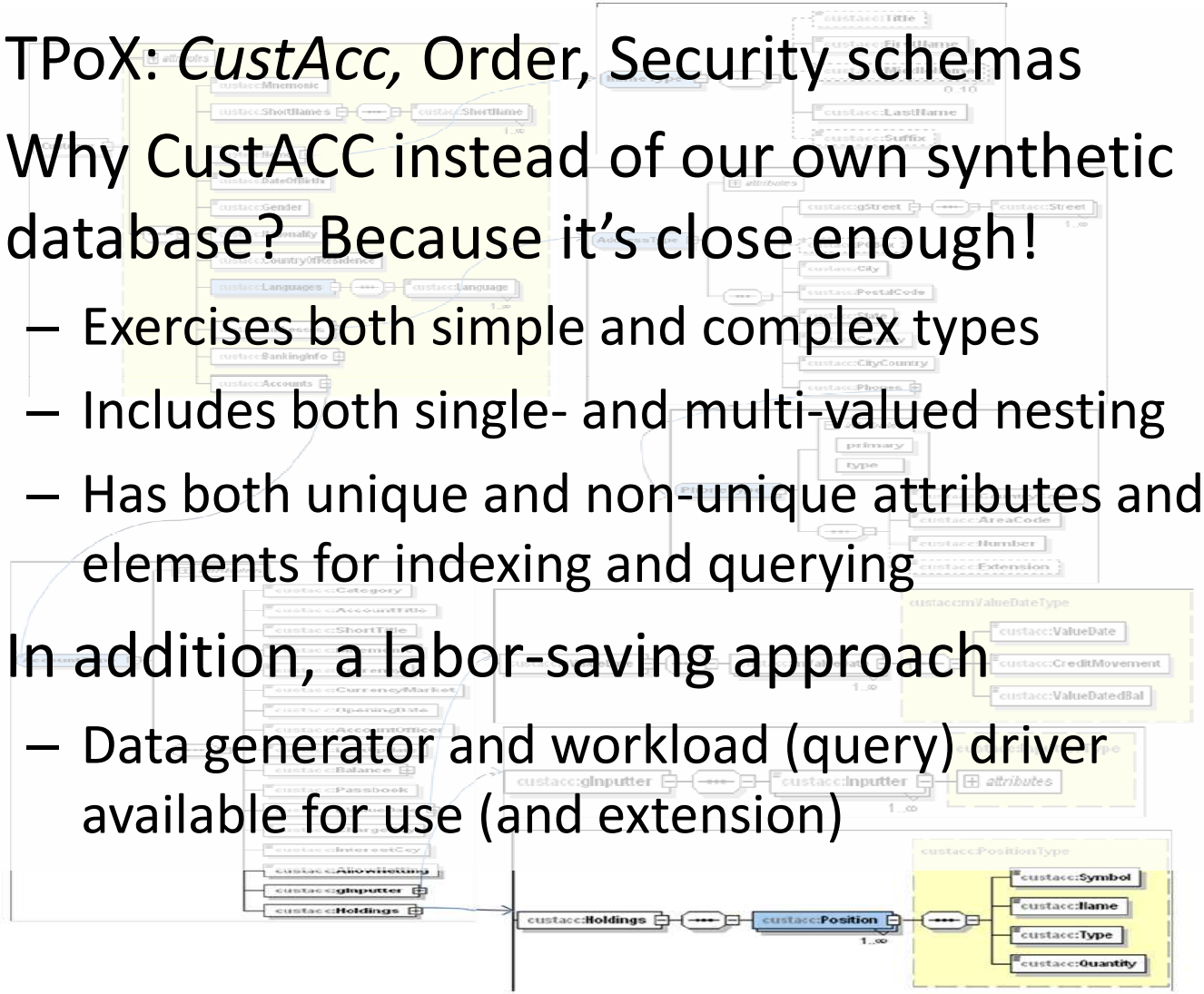
➢ Time to investigate these capabilities “for real”…!

Experimental XML Readiness Test

- Existing XML benchmarks (also see paper)
  - *TPoX, X-Mach1:* multi-user, application-oriented, scales by number of documents (think TPC-C ☺)
  - *XMark, XPathMark, X007:* single-document, single-user, exercise all aspects of XPath/XQuery using an artificial application (i.e., micro-benchmarks)
  - *MBench, MemBeR:* abstract single-document micro-benchmarks
- Enter the ***EXRT*** benchmark
  - Focus on core "data XML" operations (think Wisconsin ☺)
  - Single-user micro-benchmark, scales by number of documents
  - Target audience: XML application developers (as opposed to XQuery engine builders, as in other micro-benchmarks)
  - Cover XQuery and SQL/XML, shredded and native XML

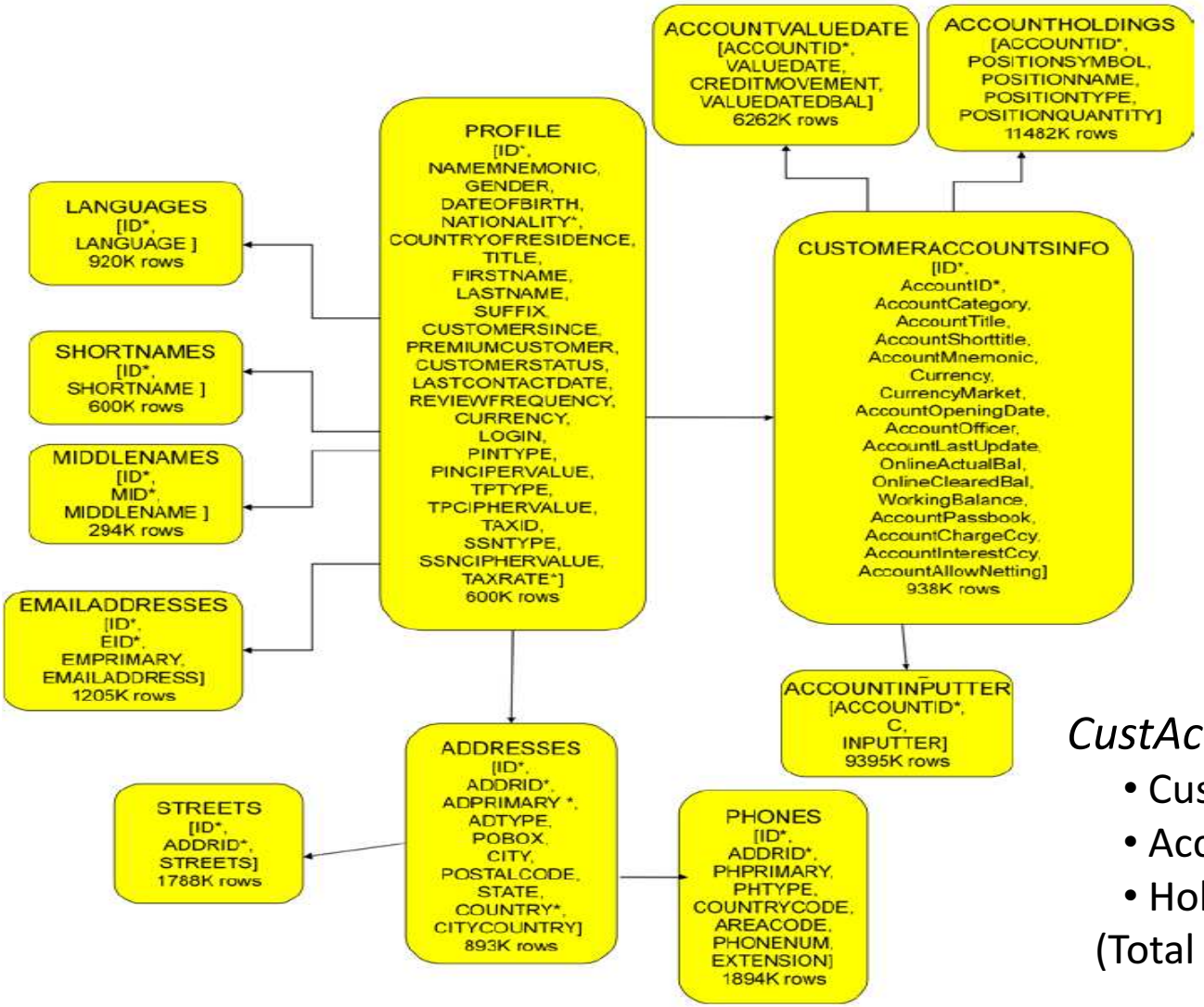
EXRT Schema (TPoX-based XML)

- TPoX: *CustAcc,* Order, Security schemas
- Why CustACC instead of our own synthetic database? Because it's close enough!
  - Exercises both simple and complex types
  - Includes both single- and multi-valued nesting
  - Has both unique and non-unique attributes and elements for indexing and querying
- In addition, a labor-saving approach
  - Data generator and workload (query) driver available for use (and extension)



UCIrvine
http://isg.ics.uci.edu

EXRT Schema (Shredded)



Note: * = indexed column

CustAcc documents:

- Customer data
- Account details
- Holdings information

(Total of 4-20 KB apiece)

EXRT Data Storage and Queries

- Three storage options considered:
  - *Shredded tables:* Normalize *CustAcc* XML schema into the 12 tables depicted on the previous slide
  - *XML column:* Store *CustAcc* XML instances, one per row, in an RDB table with one XML-typed column
  - *XML database:* Store *CustAcc* XML instances, one per document, in a collection in a native XML DB
- Two XML query language options considered:
  - *SQL/XML:* Can apply to the first two options above
  - *XQuery:* Can apply to the latter two options above

    (*Note:* Examples coming in a few slides…)

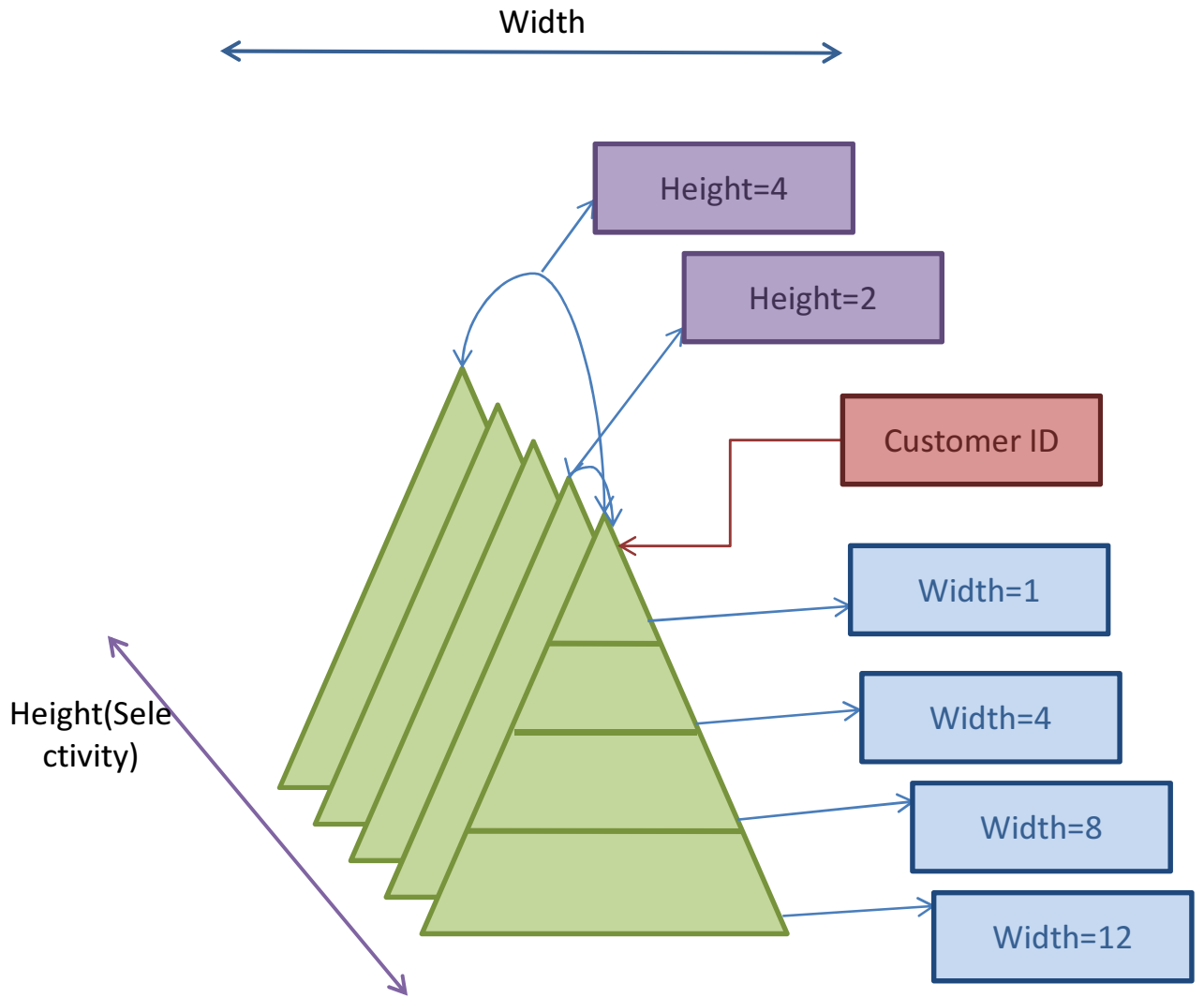
EXRT Benchmark Operations (Queries I)

Op	Description	Width
Q1	Given customer IDs, fetch the customers' *minimal profile* – consisting of their customer ID, title, first and last names, and suffix	1
Q2	Given customer IDs, fetch the customers' *basic profile* – adding middle and short names and languages to the minimal profile	2
Q3	Given customer IDs, fetch the customers' *complete profile* – adding e-mail info, addresses, streets, and phones to the basic profile	4
Q4	Given customer IDs, fetch *all of the customers' information* – including all of the info about their accounts and holdings	8

Note: Two variants of Q4 tested:

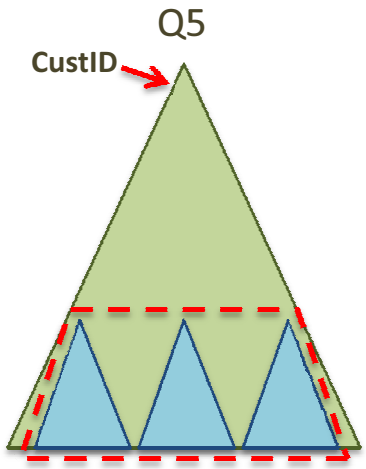
- Just fetch XML – Q4
- Reconstruct XML – Q4(re)

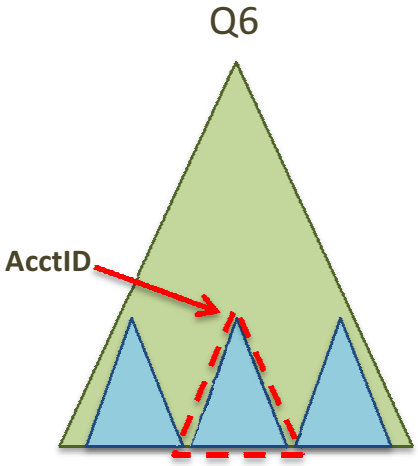
EXRT Query Result Shape

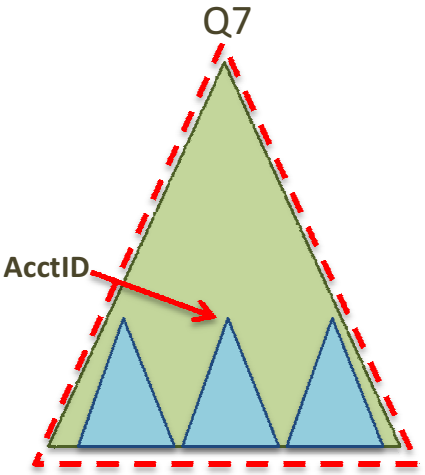


EXRT Benchmark Operations (Queries II)

Op	Description	Width
Q5	Given *customer IDs,* get the complete info for all of their *accounts*	5
Q6	Given *account IDs,* get the complete info for the *accounts*	4
Q7	Given *account IDs,* get all of the info for the *account-owning customers*	12







UCIrvine
http://isg.ics.uci.edu

EXRT Benchmark Operations (Queries III)

Op	Description	Width
Q7	Get the average number of accounts for customers of a given nationality	1
Q8	Given a country name and a tax rate, return the average account balance for customers in the specified country who have a tax rate greater than the specified tax rate	3

EXRT Benchmark Operations (Basic Updates)

Op	Description	Width
I	Given an XML string containing all of the info for a new customer, insert the new customer into the database	12
D	Given a customer ID, delete all info about this customer and his or her accounts	12

EXRT Benchmark Operations (Node Updates I)

Op	Description	Width
NI1	Given a customer ID and an XML string with a new Address element, add the new address to the specified customer	3
NI2	Given a customer ID and XML strings with a new Address element and a new e-mail address, add both to the specified customer	4
NI3	Given a customer ID and XML strings with a new Address element, a new e-mail address, and a new account, add them to the customer	8
ND1	Given a customer ID plus an integer positional indication (1, 2, or 3), delete the indicated Address node from the customer's list of addresses	3
ND2	Given a customer ID plus positional indicators for their address and e-mail lists, delete the indicated Address and Email nodes	4
ND3	Given a customer ID, an account ID, and positional indicators for their address and e-mail lists, delete the indicated Account, Address, and Email nodes	8

EXRT Benchmark Operations (Node Updates II)

Op	Description	Width
NU1	For a customer ID, update the customer's last contact date	1
NU2	Given a customer ID, a contact date, and the name of a new account officer, update the last contact date, upgrade the customer to premium	2
NU3	Given a customer ID, a contact date, the name of a new account officer, and an XML string with a list of addresses, update the customer's last contact date, upgrade the customer to premium status, update the assigned account officer's, and replace the customer's current list of addresses with the new list	5

Q1: XQuery on XML

XQuery query

```
XQUERY
declare default element namespace "http://tpox-benchmark.com/custacc";
for $cust in fn:xmlcolumn('CUSTACC.CADOC')/Customer[@id >= |1 and @id < |2    + |3 ]
return element Profile {
    attribute CustomerId { $cust/@id },
    element Name {
        $cust/Name/Title,
        $cust/Name/FirstName,
        $cust/Name/LastName,
        $cust/Name/Suffix } }
```

Note: similar as TPoX, |1, |2 and |3 are the numbered parameter markers which will be replaced by actual literal values supplied from different input sets.

UCIrvine
http://isg.ics.uci.edu

Q1: SQL/XML on XML

SQL/XML query

```
SELECT XMLQUERY('declare default element namespace "http://tpox-benchmark.com/custacc";
for $cust in $cadoc/Customer
    return element Profile {
        attribute CustomerId { $cust/@id },
        element Name {
            $cust/Name/Title,
            $cust/Name/FirstName,
            $cust/Name/LastName,
            $cust/Name/Suffix } }' PASSING CUSTACC.cadoc AS "cadoc")
    FROM CUSTACC
    WHERE XMLEXISTS
    ('declare default element namespace "http://tpox-benchmark.com/custacc";
    $cadoc/Customer[@id>=$id1 and @id<$id2 + $tallness]'
    PASSING cadoc AS "cadoc", cast (? as int) as "id1",
                              cast(? as int) as "id2",
                              cast(? as int) as "tallness")
```

UCIrvine
http://isg.ics.uci.edu

Q1: SQL/XML on Relations

Relational query:

```
select
xmldocument(
xmlelement(name "Customer",
   xmlnamespaces( DEFAULT 'http://our-benchmark.com/custacc'),
   xmlattributes(ID as "id"),
   xmlelement(name "Name",
      xmlelement(name "Title", title),
      xmlelement(name "FirstName", firstname),
      xmlelement(name "LastName", lastname),
      xmlelement(name "Suffix", suffix)
   ))) from profile
where profile.id>= cast(? as int) and profile.id < cast(? as int) + cast(? as int)
```

UCIrvine
http://isg.ics.uci.edu

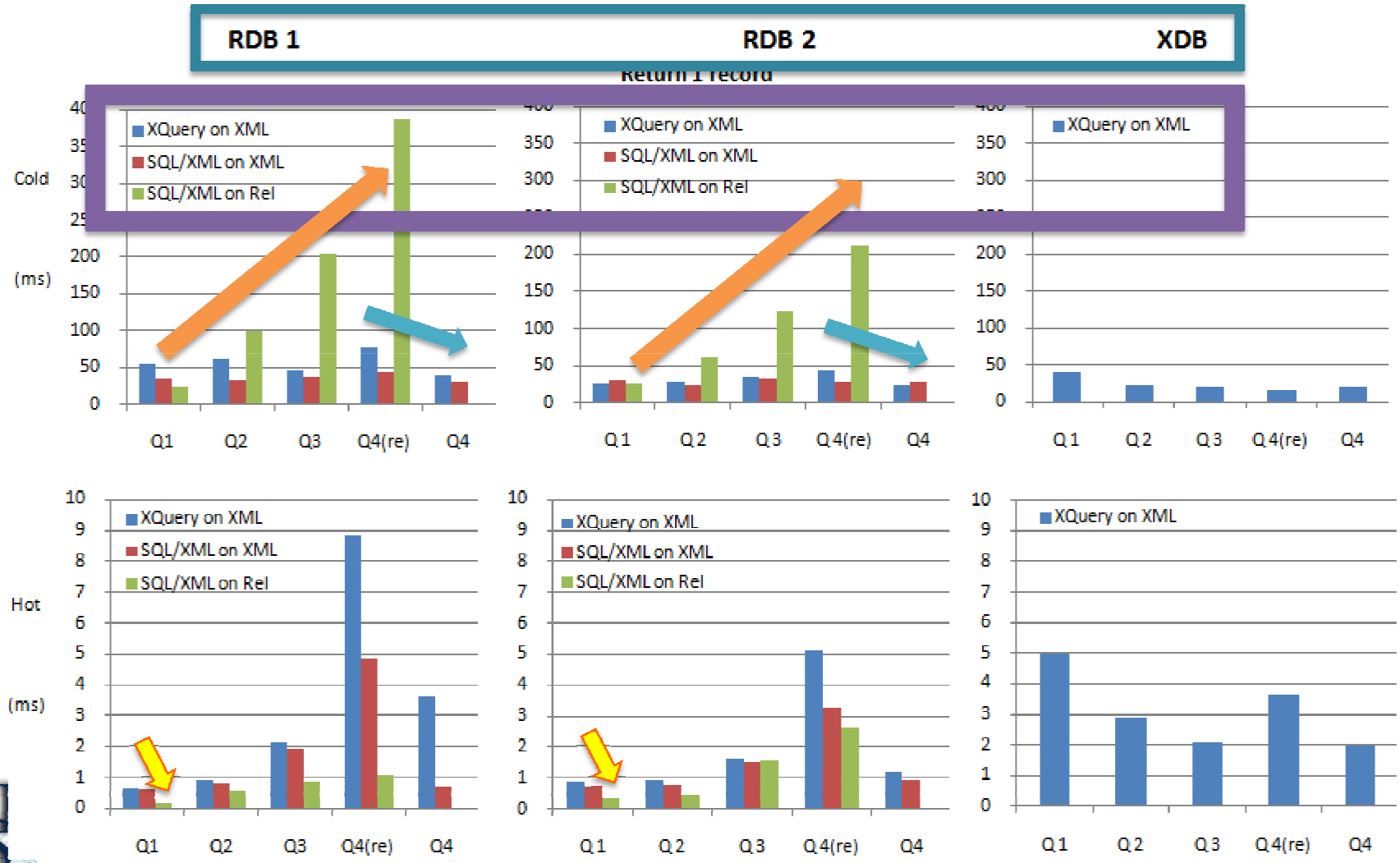
Benchmarking Procedure

- Data generated using the TPoX data generator
  - 600K *CustAcc* document instances
- EXRT brackets the performance of each query
  - *Cold runs:* clear buffer pool, use new parameters
  - *Hot runs:* run same query with same parameters
- A few procedural details
  - Vendor-specific buffer clearing procedure
  - JDBC (or equivalent) API with parameterized queries
    (Note: see paper for a bit of fine print in this area)
  - Parameterized query times exclude compilation time
  - Query parameter values pre-computed (offline) and then used at runtime

Benchmarking Procedure *(cont.)*

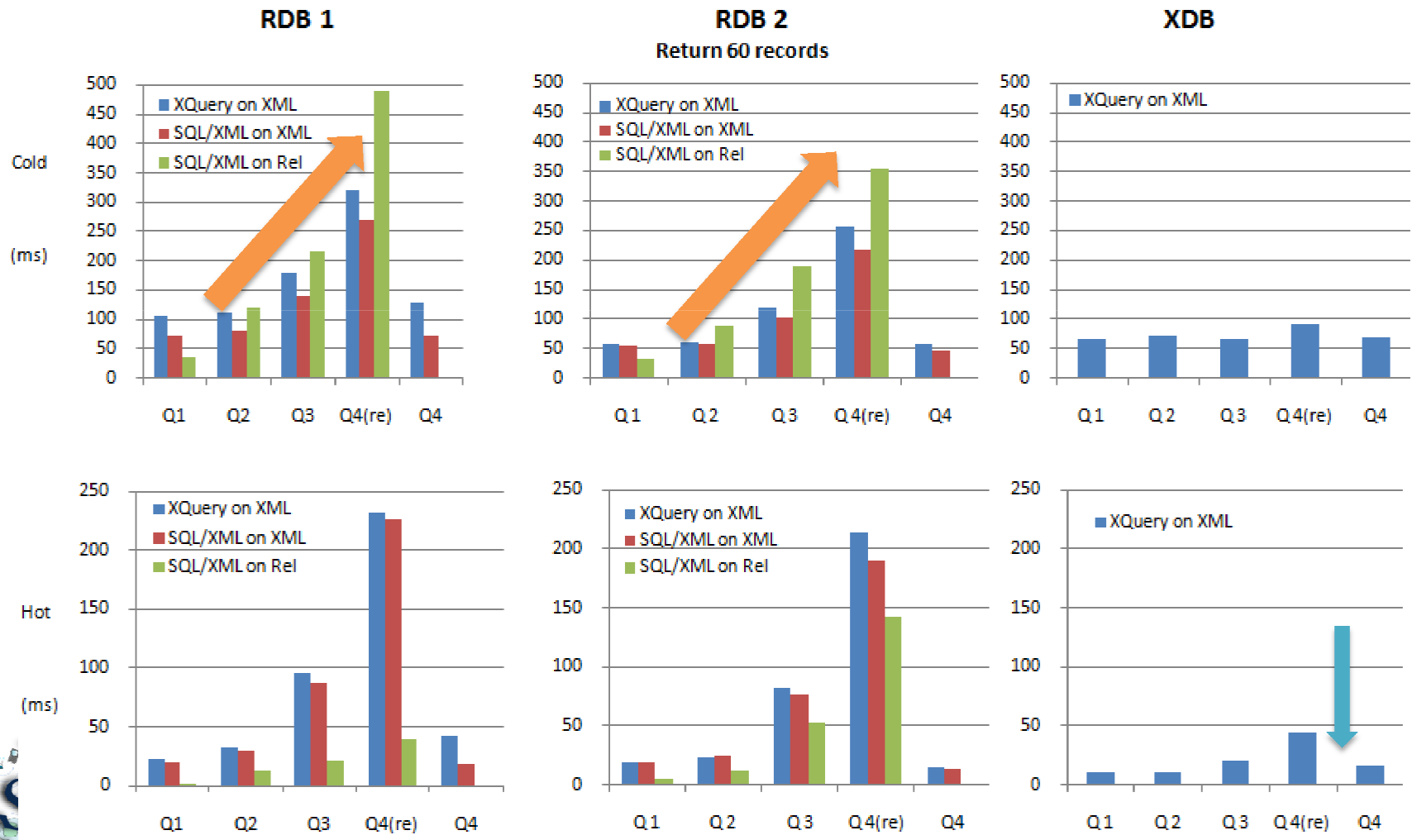
- Tested 3 systems – RDB1, RDB2, and XDB – all configured "identically"
  - Dell 3.16 GHz Intel Core 2 Duo D8500 CPU
  - 4GB of main memory, two 320 GB 7200 RPM disks
  - Striped Linux file system spanning both disks
  - Each DB system gets to use all of memory
- In terms of the tested software:
  - RDB2 was used out of the box
  - XDB was used out of the box (except for cache clearing)
  - RDB1 needed one patch applied to address a JDBC driver issue with XML result sets, required a bit of trickery to work around an index statistics bug for binary double indexes, and needed small changes in a few query predicates to assist its query optimizer

Q1-Q4: Customer Queries (1)

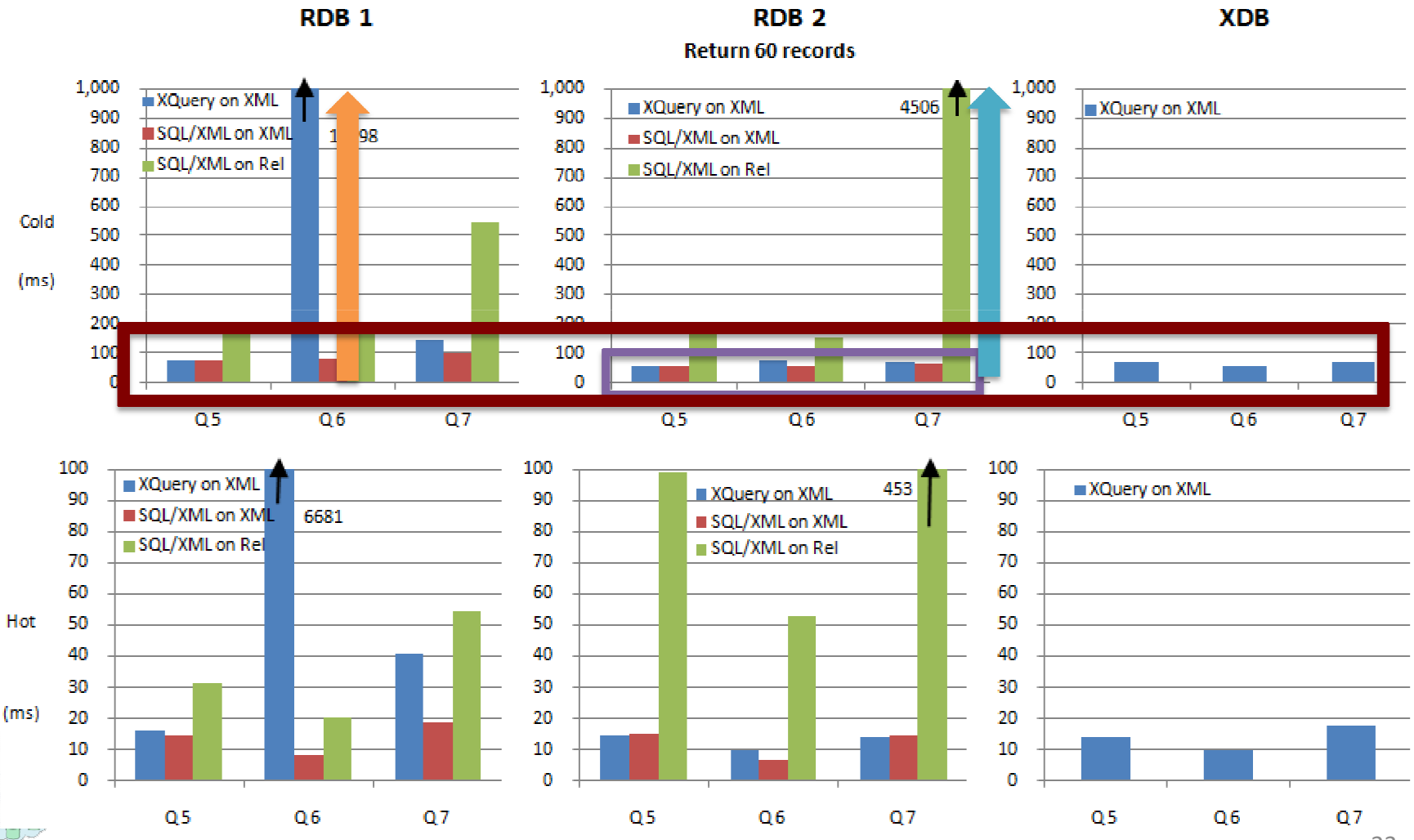


UCIrvine
http://isg.ics.uci.edu

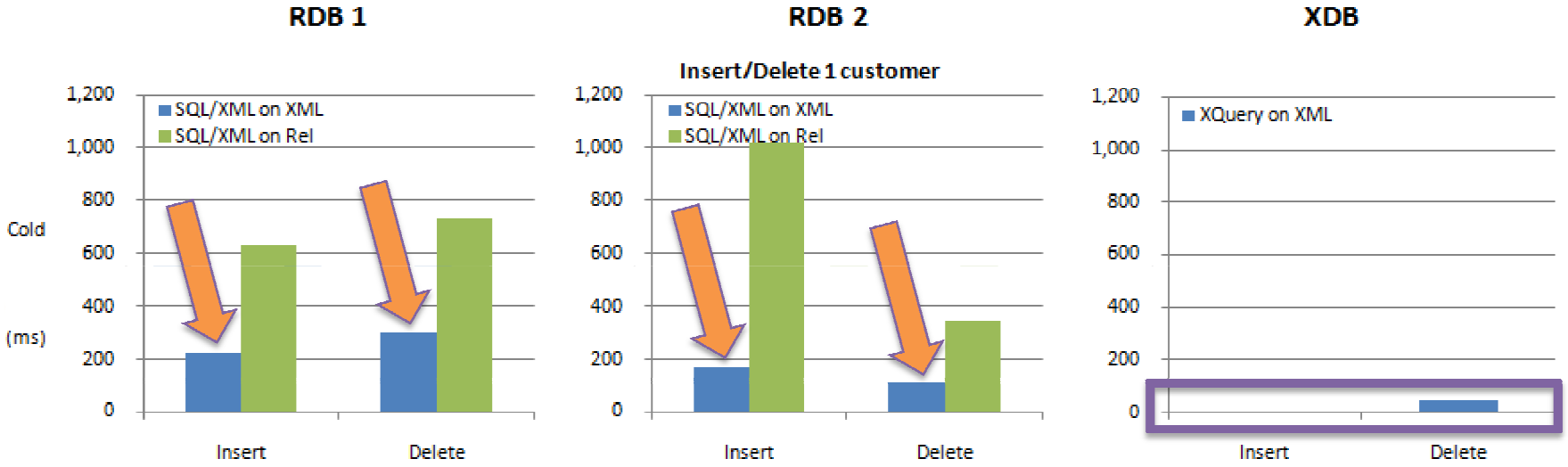
Q1-Q4: Customer Queries (60)



Q5-Q7: Path Queries (60)



Updates: Customer Insert/Delete



Note:

•Cold, single-document insert/delete tests difficult to conduct, especially comparably across systems.

•Possible alternative would be to measure steady state performance for a large stream of inserts (e.g., thousands of documents).

UCIrvine
http://isg.ics.uci.edu

Summary

- EXRT is a micro-benchmark for testing a DB system's readiness for enterprise data XML usage
  - Shredded versus native XML storage
  - Performance of various commercial systems
  - Saw similarities, differences, and a few bugs
- Initial performance lessons from applying EXRT
  - Shredded versus native is width-dependent
  - Native storage faster for inserts, for non-constructing retrievals, and for wide query results
  - Need better steady-state update performance methodology
- Possible futures (us or you ☺)
  - Share an implementation of EXRT
  - Try EXRT on more commercial systems
  - Do something EXRT-like for content-oriented use cases

Acknowledgements

- We would like to sincerely thank certain engineers from each of the vendors whose systems we tested; their help and feedback were both invaluable…!
  - *Person 1*
  - *Person 2*
  - *Person 3*

  (☺)

UCIRVINE
http://isg.ics.uci.edu