IBM Chat with Lab for Greater China Group

- **Host:** Frank Ning, Manager, DB2 LUW Install and Up/Running Development
- **Executive introduction** (audio)

  Sal Vella, Vice President, Development, Distributed Data Servers and Data Warehousing
- **Presentation: DB2 pureXML Technical Overview**

  Shumin Wu, Software Engineer at the Silicon Valley Lab

Jan, 2011

Executive Introduction

Sal Vella

Vice President, Development, Distributed Data Servers and Data Warehousing

IBM Software Group

DB2 pureXML Technical Overview

- Chat with the Silicon Valley Lab for the Greater China Group

Shumin Wu, Software Engineer
and the DB2 Silicon Valley Lab Team

Jan, 2011

Agenda

- Introduction to pureXML
  - XPath Data Model
  - Location Path with Examples
  - Qualified Names and Namespaces
  - APIs Supported
  - Document Encoding
  - When to use XQuery and when to use SQL/XML
  - Summary
- Compression
  - Why Compression? Why DB2 LUW?
  - Static Row Compression (Table)
  - XML Compression
  - Index Compression

XML Data versus Relational Data

```
<customerinfo>
        <name> John Smith  </name>
        <addr country="Canada">
                <street>Fourth</street>
                <city>Calgary</city>
                <prov-state>Alberta</prov-state>
                <pcode-zip>M1T 2A9</pcode-zip>
        </addr>
        <phone type="Work">
            963-289-4136 </phone>
</customerinfo>
```

Name	Street	City	State	Zip	Phone
John Smith	Fourth	Calgary	Alberta	M1T 2A9	963-289-4136

- **XML data is hierarchical and nested, relational data is flat**
  - "Find the name of all employees (at any level)"
- **XML is self-describing; data is mixed together with meta-data (tags)**
  - Can query both the data and the meta-data at the same time:
- **XML content has an intrinsic order, in contrast with tables :**
  - The order of chapters within a book is important
- **XML data is extensible, is not regular and homogeneous like tables**
  - Every document can have a different structure, and different data-types
- **XML data must be well-formed**
  - No concept of well-formedness in relational data
- **A single XML document often represents an external business record**
  - Typically multiple relational rows represent a business record

Why New Query Languages for XML?

- XML data is sufficiently different than relational data
- Hierarchical nature of XML data requires a navigation language
- SQL can not handle the heterogeneous nature of XML data
- Use XQuery for XML data and SQL for relational data
  - We discuss when to use which later
- SQL/XML, an extension to SQL, is suitable for hybrid data (mixed relational and XML data)

Processing XML with SQL

ô **Problem:**

- SQL lacks the expressive power to look inside XML data and query the structure and content.
- Using only SQL, can only select entire XML document. For example,

Select **Info** from Customer where Cid = 1000

```
Info
<customer>
    <name>Kathy Smith</name>
    <addr country="Canada">
       <street>5 Rosewood</street>
       <city>Toronto</city>
       <prov-state>Ontario</prov-state>
       <pcode-zip>M6W-1E6</pcode-zip>
    </addr>
    <phone type="work">416-555-1358</phone>
 </customer>
```

Treated like a black box

□ **Solution:**

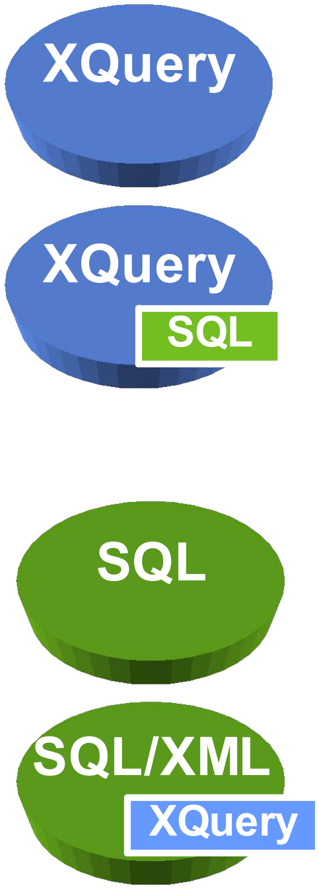
- SQL/XML adds new SQL functions for querying, searching and transforming XML data, e.g.:
  - XMLQUERY – Query and retrieve in SELECT list
  - XMLEXISTS – new predicate in WHERE clause to search values inside XML documents to restrict the set of rows the query operates on.
  - XMLTABLE – Transform XML data to tabular format in FROM clause
- The new functions rely on XPath and XQuery from W3C

Two Ways to Query XML in DB2

ô **XQuery**

- – XQuery as the primary language
- – Optional: SQL embedded in XQuery

- **SQL/XML**
  - – SQL as the primary language
  - – Optional: XQuery embedded in SQL

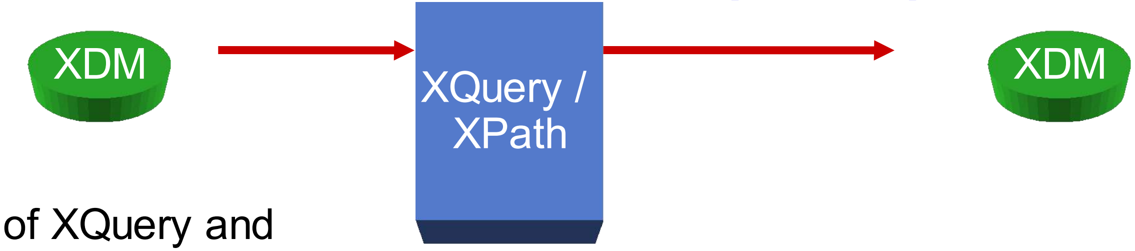


http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606nicola

Agenda – Introduction to XQuery

- XPath Data Model
- Location Path with Examples
- Qualified Names and Namespaces
- APIs Supported
- Document Encoding
- When to use XQuery and when to use SQL/XML
- Summary

XQuery and XPath Data Model (XDM)

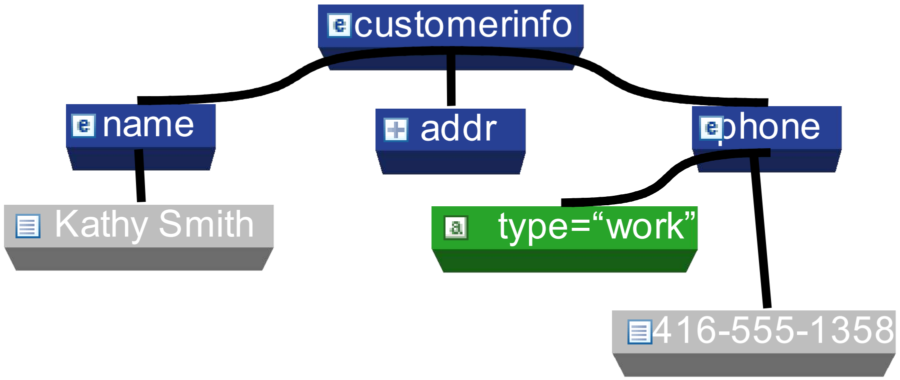


- The input and output of XQuery and XPath expressions are XDM instances
- XDM is produced through the parsing process
- XDM is the hierarchical representation of XML data

Textual representation

```
<?xml version="1.0" encoding="UTF-8"?>
<customerinfo Cid="1000">
  <name> Kathy Smith </name>
  <addr country="Canada">
    <street> 5 Rosewood </street>
    <city> Toronto </city>
    <prov-state> Ontario </prov-state>
    <pcode-zip> M6W 1E6 </pcode-zip>
  </addr>
  <phone type="work">
    416-555-1358
  </phone>
</customerinfo>
```

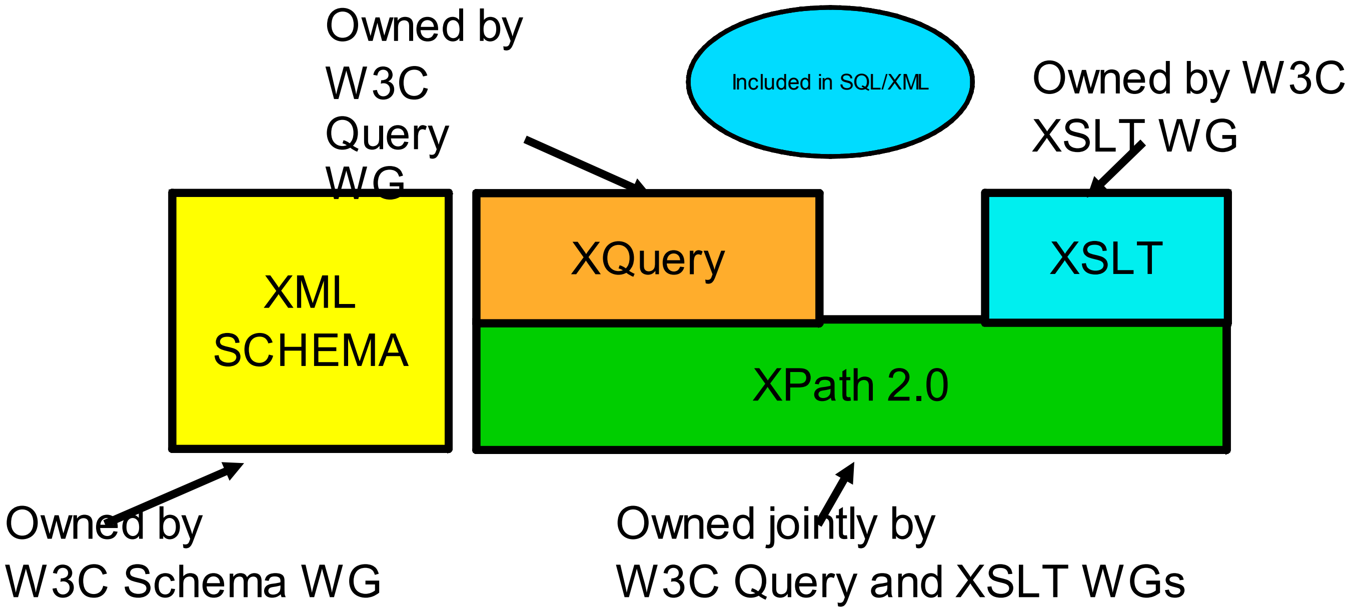
Node-tree representation



http://w3.org/TR/XPath-datamodel

What is XQuery?

- A query language designed for XML data sources. XQuery = XML Query
- Developed by a W3C Working Group over five years
- In “Recommendation” status
- The industry standard query language for XML data
- Provides declarative access to XML data, much as SQL does for relational data
- Included (by reference) in the ISO SQL Standard, Part 14 (SQL/XML)
- Supported interface in various IBM products including DB2



“The W3C XML Query Working Group worked with the W3C XML Schema Working Group and the W3C XSL Working Group to make a set of specifications that all work together.”

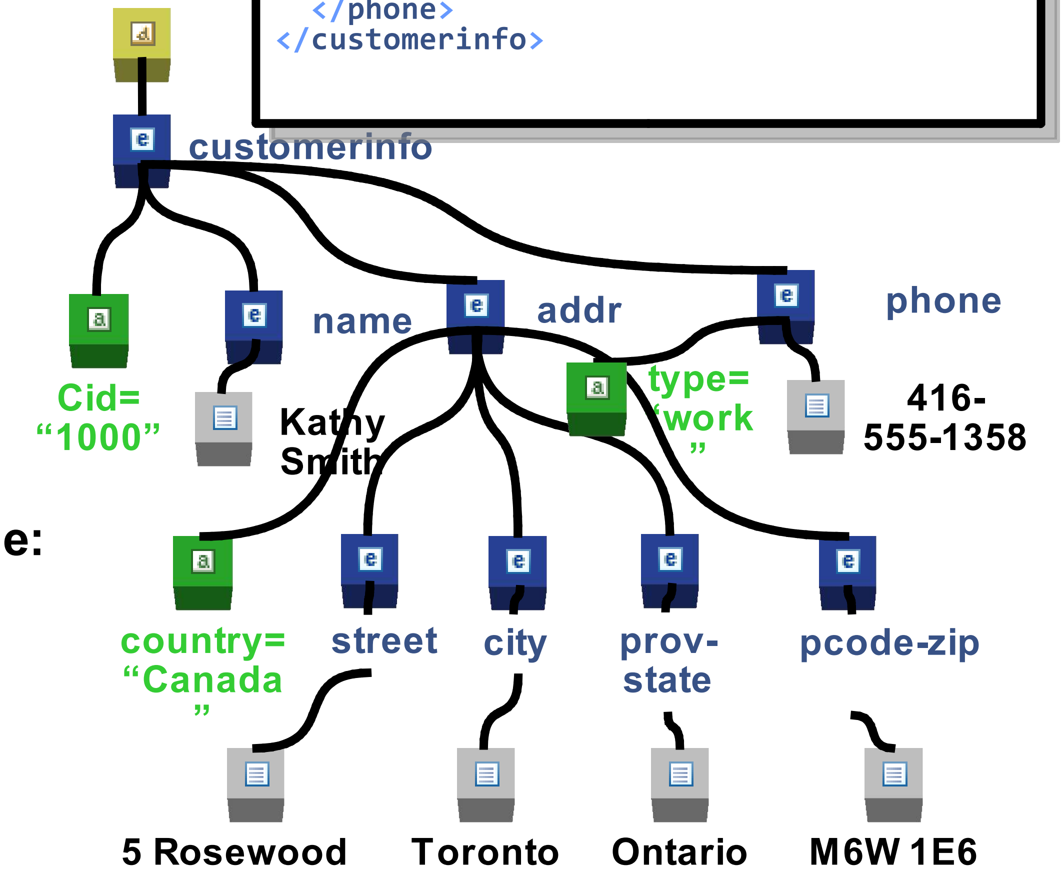
“Use XQuery to take data from multiple databases, from XML files, from remote Web documents, even from CGI scripts, and to produce XML results that you can process with XSLT.”

http://www.w3.org/XML/Query/

XDM Node Types

- **6 kinds of nodes in an XML document**
  - Document Node
  - Element Node
  - Attribute Node
  - Text Node
  - Comment Node
  - Processing Instruction Node
- **Each node has a set of properties which may include:**
  - A name
  - A type *(e.g. "xs:decimal")*
  - A string value *(e.g. "47")*
  - A typed value *(e.g. 47)*

```
<?xml version="1.0" encoding="UTF-8"?>
<customerinfo Cid="1000">
  <name> Kathy Smith </name>
  <addr country="Canada">
    <street> 5 Rosewood </street>
    <city> Toronto </city>
    <prov-state> Ontario </prov-state>
    <pcode-zip> M6W 1E6 </pcode-zip>
  </addr>
  <phone type="work">
    416-555-1358
  </phone>
</customerinfo>
```



XDM Sequences

- **An instance of XDM is an ordered sequence of 0 or more items (<name>John Doe </name>, <name> Mary Ray </name>)**
- **An item is either an XML node or atomic value**
  - Node: **<tag>value</tag>**
  - Atomic value: an instance of one of the built-in atomic data types
    - **“Mary”**
- **XML Schema Types: 19 built-in primitive, 25 built-in derived**
- **Sequence**
  - Never nested
  - Can be heterogeneous: (<name> Mary Ray </name> , “John”, 45)

XPath: XML Path Language

- **XPath 2.0 is a syntactic subset of XQuery. SQL/XML statements can include XPath or XQuery expressions**
- **XPath is used to refer to particular parts of XML documents that conform to the XQuery/XPath Data Model (XDM).**
  - XDM provides a tree structured representation of XML documents
- **Location paths are the most common expressions in XPath**
  - Are used to navigate nodes in the document
  - Consist of one or more steps separated by “/” or “//”
  - Each step can be another axis step (/node) or a filter expression
  - The value of the **location path** is the sequence of the items produced by the final step in the path
    - This sequence can contain either nodes or atomic values, not both

    /root/node1/node2/node3
  - Input Sequence: Input is sequence of nodes
  - Result Sequence: Output is sequence of distinct nodes in document order, or a sequence of atomic values

XPath: Using Location Paths to Retrieve Nodes

```
<customerinfo>
        <name>John Smith</name>
        <addr country=“Canada">
                <street>Fourth</street>
                <city>Calgary</city>
                <prov-state>Alberta</prov-state>
                <pcode-zip>M1T 2A9</pcode-zip>
        </addr>
        <phone type="work">
       963-289-4136
        </phone>
</customerinfo>
```

/

/customerinfo

/customerinfo/name

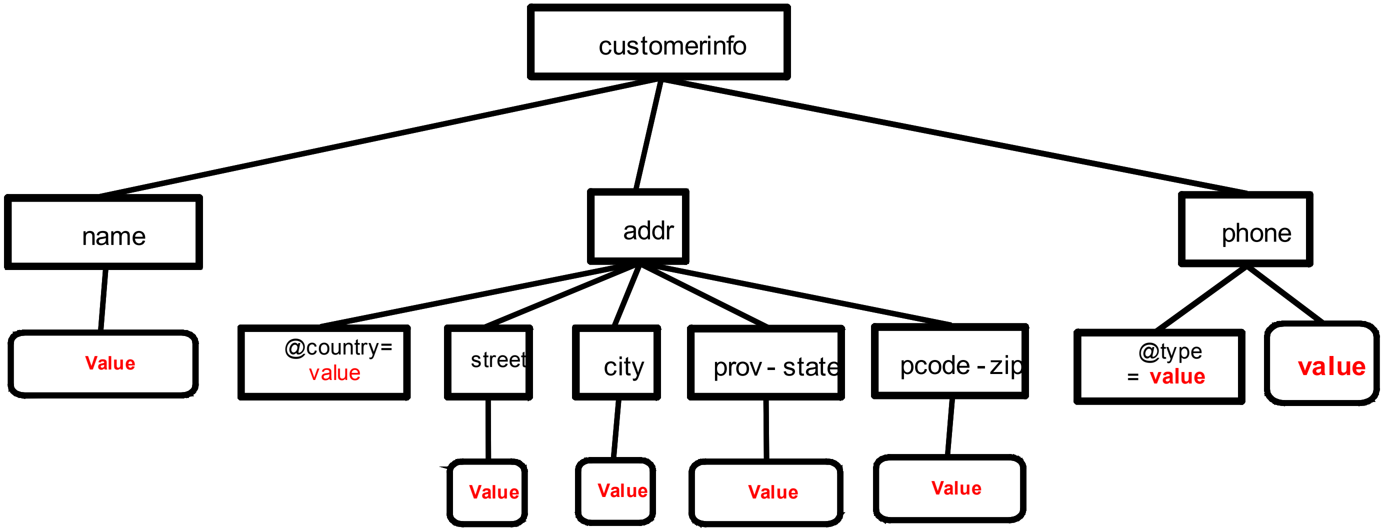
/customerinfo/addr/@country

/customerinfo/addr/street

/customerinfo/addr/city

/customerinfo/addr/prov-state

(...)



15

XPath: Abbreviated Syntax

	Sample Paths:
@ - attribute	/node/@attribute
// - decendent or self node	/node//
. - self node	/node/.
.. - parent node	/node/..
text() - text node	/node/text()
* - any node	/node/*

XPath: Built-in functions

- **Numeric**
  - `min`, `max`, `ceiling`, `floor` …
- **String**
  - `contains`, `starts-with`, `replace`, `substring` …
- **Sequence**
  - `distinct-values`, `data`, `last`, `position`, `count` …
- **Boolean**
  - `not`, `true` …

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.xml.doc/doc/xqrfncategory.htmll

XPath 1

/customerinfo

XPath

```
<customerinfo>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                  Fourth
                </street>
                <city>
                  Calgary
                </city>
                <prov-state>
                  Alberta
                </prov-state>
                <pcode-zip>
                   M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                  963-289-4136
        </phone>
</customerinfo>
<customerinfo>
        <name>
                Fred Doe
        </name>
        <addr country=“USA">
                <prov-state>
                  Iowa
                </prov-state>
                <pcode-zip>
                  987653
                </pcode-zip>
        </addr>
        <phone type=“Mobile">
                  244-775-3151
        </phone>
</customerinfo>
```

Collection of 2 Documents: Input Sequence

```
<customerinfo>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                        Fourth
                </street>
                 <city>
                        Calgary
                </city>
                <prov-state>
                        Alberta
                </prov-state>
                <pcode-zip>
                        M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                963-289-4136
        </phone>
</customerinfo>
<customerinfo>
        <name>
                Fred Doe
        </name>
        <addr country=“USA">
                <prov-state>
                        Iowa
                </prov-state>
                <pcode-zip>
                        987653
                </pcode-zip>
        </addr>
        <phone type=“Mobile">
                244-775-3151
        </phone>
</customerinfo>
```

Result Sequence

XPath 2

/customerinfo/addr/prov-state

```
<customerinfo>
      <name>
            John Smith
      </name>
      <addr country=“Canada">
            <street>
                  Fourth
            </street>
            <city>
                  Calgary
            </city>
            <prov-state>
               Alberta
            </prov-state>
            <pcode-zip>
                  M1T 2A9
            </pcode-zip>
      </addr>
      <phone type=“work">
                  963-289-4136
      </phone>
</customerinfo>

<customerinfo>
      <name>
            Fred Doe
      </name>
      <addr country=“USA">
            <prov-state>
                  Iowa
            </prov-state>
            <pcode-zip>
                  987653
            </pcode-zip>
      </addr>
      <phone type=“Mobile">
                  244-775-3151
      </phone>
</customerinfo>
```

```
<customerinfo>
            <name>
                  John Smith
            </name>
            <addr country=“Canada">
                  <street>
                        Fourth
                  </street>
                  <city>
                        Calgary
                  </city>
                  <prov-state>
                        Alberta
                  </prov-state>
                  <pcode-zip>
                        M1T 2A9
                  </pcode-zip>
            </addr>
            <phone type=“work">
                  963-289-4136
            </phone>
      </customerinfo>

      <customerinfo>
            <name>
                  Fred Doe
            </name>
            <addr country=“USA">
                  <prov-state>
                        Iowa
                  </prov-state>
                  <pcode-zip>
                        987653
                  </pcode-zip>
            </addr>
            <phone type=“Mobile">
                  244-775-3151
            </phone>
      </customerinfo>
```

Result sequence

```
<prov-state>
      Alberta
</prov-state>

<prov-state>
      Iowa
</prov-state>
```

XPath 3

/customerinfo/addr[prov-state=‘Iowa’]/@country

```
<customerinfo>
    <name>
        John Smith
    </name>
    <addr country=“Canada">
        <street>
            Fourth
        </street>
        <city>
            Calgary
        </city>
        <prov-state>
            Alberta
        </prov-state>
        <pcode-zip>
            M1T 2A9
        </pcode-zip>
    </addr>
    <phone type=“work">
        963-289-4136
    </phone>
</customerinfo>

<customerinfo>
    <name>
        Fred Doe
    </name>
    <addr country=“USA">
        <prov-state>
            Iowa
        </prov-state>
        <pcode-zip>
            987653
        </pcode-zip>
    </addr>
    <phone type=“Mobile">
        244-775-3151
    </phone>
</customerinfo>
```

```
<customerinfo>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                        Fourth
                </street>
                 <city>
                        Calgary
                </city>
                <prov-state>
                        Alberta
                </prov-state>
                <pcode-zip>
                        M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                963-289-4136
        </phone>
</customerinfo>

<customerinfo>

        <name>
                Fred Doe
        </name>
        <addr country=“USA">
                <prov-state>
                        Iowa
                </prov-state>
                <pcode-zip>
                        987653
                </pcode-zip>
        </addr>
        <phone type=“Mobile">
                244-775-3151
        </phone>
</customerinfo>
```

Result sequence

USA

XPath 4 /customerinfo /addr/city[.=‘Calgary’]/../@country

```
<customerinfo>
     <name>
          John Smith
     </name>
     <addr country=“Canada">
          <street>
               Fourth
          </street>
          <city>
               Calgary
          </city>
          <prov-state>
            Alberta
          </prov-state>
          <pcode-zip>
                M1T 2A9
          </pcode-zip>
     </addr>
     <phone type=“work">
               963-289-4136
     </phone>
</customerinfo>

<customerinfo>
     <name>
          Fred Doe
     </name>
     <addr country=“USA">
          <prov-state>
                Iowa
          </prov-state>
          <pcode-zip>
               987653
          </pcode-zip>
     </addr>
     <phone type=“Mobile">
               244-775-3151
     </phone>
</customerinfo>
```

```
<customerinfo>
          <name>
                    John Smith
          </name>
          <addr country=“Canada">
                    <street>
                              Fourth
                    </street>
                     <city>
                              Calgary
                    </city>
                    <prov-state>
                              Alberta
                    </prov-state>
                    <pcode-zip>
                              M1T 2A9
                    </pcode-zip>
          </addr>
          <phone type=“work">
                    963-289-4136
          </phone>

</customerinfo>


<customerinfo>
          <name>
                    Fred Doe
          </name>
          <addr country=“USA">
                    <prov-state>
                              Iowa
                    </prov-state>
                    <pcode-zip>
                              987653
                    </pcode-zip>
          </addr>
          <phone type=“Mobile">
                    244-775-3151
          </phone>
</customerinfo>
```

Result sequence

Canada

21

XPath 5 fn:substring(/customerinfo/addr/city[.=‘Calgary’]/../@country, 1,1)

```
<customerinfo>
     <name>
          John Smith
     </name>
     <addr country=“Canada">
          <street>
               Fourth
          </street>
          <city>
               Calgary
          </city>
          <prov-state>
             Alberta
          </prov-state>
          <pcode-zip>
                M1T 2A9
          </pcode-zip>
     </addr>
     <phone type=“work">
               963-289-4136
     </phone>
</customerinfo>

<customerinfo>
     <name>
          Fred Doe
     </name>
     <addr country=“USA">
          <prov-state>
                Iowa
          </prov-state>
          <pcode-zip>
               987653
          </pcode-zip>
     </addr>
     <phone type=“Mobile">
               244-775-3151
     </phone>
</customerinfo>
```

```
<customerinfo>
          <name>
                    John Smith
          </name>
          <addr country=“Canada">
                    <street>
                              Fourth
                    </street>
                     <city>
                              Calgary
                    </city>
                    <prov-state>
                              Alberta
                    </prov-state>
                    <pcode-zip>
                              M1T 2A9
                    </pcode-zip>
          </addr>
          <phone type=“work">
                    963-289-4136
          </phone>

</customerinfo>


<customerinfo>
          <name>
                    Fred Doe
          </name>
          <addr country=“USA">
                    <prov-state>
                              Iowa
                    </prov-state>
                    <pcode-zip>
                              987653
                    </pcode-zip>
          </addr>
          <phone type=“Mobile">
                    244-775-3151
          </phone>
</customerinfo>
```

Result sequence

C

XPath 6

/customerinfo//text()

//text()

```
<customerinfo>
       <name>
              John Smith
       </name>
       <addr country="“Canada">
              <street>
                     Fourth
              </street>
              <city>
                     Calgary
              </city>
              <prov-state>
                  Alberta
              </prov-state>
              <pcode-zip>
                      M1T 2A9
              </pcode-zip>
       </addr>
       <phone type="“work">
                     963-289-4136
       </phone>
</customerinfo>

<customerinfo>
       <name>
              Fred Doe
       </name>
       <addr country="“USA">
              <prov-state>
                     Iowa
              </prov-state>
              <pcode-zip>
                     987653
              </pcode-zip>
       </addr>
       <phone type="“Mobile">
                     244-775-3151
       </phone>
</customerinfo>
```

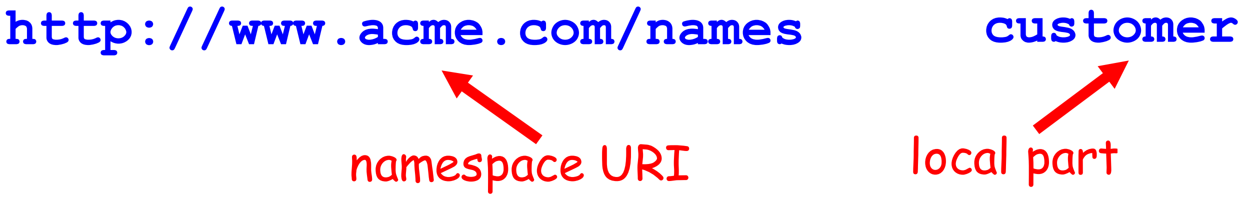
```
<customerinfo>
           <name>
                  John Smith
           </name>
           <addr country=“Canada">
                  <street>
                           Fourth
                  </street>
                   <city>
                           Calgary
                  </city>
                  <prov-state>
                           Alberta
                  </prov-state>
                  <pcode-zip>
                           M1T 2A9
                  </pcode-zip>
           </addr>
           <phone type=“work">
                  963-289-4136
           </phone>
</customerinfo>


<customerinfo>

           <name>
                  Fred Doe
           </name>
           <addr country=“USA">
                  <prov-state>
                           Iowa
                  </prov-state>
                  <pcode-zip>
                           987653
                  </pcode-zip>
           </addr>
           <phone type=“Mobile">
                  244-775-3151
           </phone>
</customerinfo>
```

Result Sequence

John Smith
Fourth
Calgary
Alberta
M1T 2A9
963-289-4136
Fred Doe
Iowa
987653
244-775-3151

Qualified Names

- **In XML, a name can be qualified by a namespace.**
- **A Qualified Name (QName) has two parts:**



- In an XML document or XPath, the URI part is represented by a short prefix: acme:customer
- Each prefix is bound to a namespace URI
- Element, attribute and function names are all QNames
- Variables in XQuery are written :**Variables:** $x $myVariable

Defining Default Namespaces

- **A default element namespace can be defined**
  - In the query prolog
    - declare default element namespace ‘http://www.acme.org/names’
  - In element constructors
    - Overrides the default defined in the prolog
    - Local; valid only within the element constructor defining it
    - <book xmlns = “http://www.acme.org/names”> ….. </book>
- **Default element namespace applies to element names with no prefix**
  - **<title> is equivalent to <acme:title>**
  - $book/title is equivalent to $book/acme:title
    - Where acme is bound to the default namespace
- **Default namespace does not apply to attribute names**
  - Attribute names with no prefix is in no namespace
  - $book/@year is equivalent to $book/@year
- **A number of namespaces are predefined in DB2 following namespace prefixes are predefined e.g., fn**

XPath with Namespace 1

//text()

declare default element namespace ‘http://acme.org/emp’ /customerinfo//text()

```
<customerinfo xmlns = “http://acme.org/emp”>
      <name>
            John Smith
      </name>
      <addr country=“Canada">
            <street>
                  Fourth
            </street>
            <city>
                  Calgary
            </city>
            <prov-state>
               Alberta
            </prov-state>
            <pcode-zip>
                   M1T 2A9
            </pcode-zip>
      </addr>
      <phone type=“work">
                  963-289-4136
      </phone>
</customerinfo>
```

```
<customerinfo>
      <name>
            Fred Doe
      </name>
      <addr country=“USA">
            <prov-state>
                  Iowa
            </prov-state>
            <pcode-zip>
                  987653
            </pcode-zip>
      </addr>
      <phone type=“Mobile">
                  244-775-3151
      </phone>
</customerinfo>
```

```
<customerinfo xmlns = “http://acme.org/emp”>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                        Fourth
                </street>
                 <city>
                        Calgary
                </city>
                <prov-state>
                        Alberta
                </prov-state>
                <pcode-zip>
                        M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                963-289-4136
        </phone>
</customerinfo>
```

```
<customerinfo>

        <name>
                Fred Doe
        </name>
        <addr country=“USA">
                <prov-state>
                        Iowa
                </prov-state>
                <pcode-zip>
                        987653
                </pcode-zip>
        </addr>
        <phone type=“Mobile">
                244-775-3151
        </phone>
</customerinfo>
```

Result sequence

John Smith
Fourth
Calgary
Alberta
M1T 2A9
963-289-4136

XPath with Namespace 2

/*:customerinfo//text()

```
<customerinfo xmlns = “http://acme.org/emp”>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                        Fourth
                </street>
                <city>
                        Calgary
                </city>
                <prov-state>
                    Alberta
                </prov-state>
                <pcode-zip>
                         M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                        963-289-4136
        </phone>
</customerinfo>


<customerinfo>
        <name>
                Fred Doe
        </name>
        <addr country=“USA">
                <prov-state>
                        Iowa
                </prov-state>
                <pcode-zip>
                        987653
                </pcode-zip>
        </addr>
        <phone type=“Mobile">
                        244-775-3151
        </phone>
</customerinfo>
```

```
<customerinfo xmlns = “http://acme.org/emp”>
        <name>
                John Smith
        </name>
        <addr country=“Canada">
                <street>
                        Fourth
                </street>
                 <city>
                        Calgary
                </city>
                <prov-state>
                        Alberta
                </prov-state>
                <pcode-zip>
                        M1T 2A9
                </pcode-zip>
        </addr>
        <phone type=“work">
                963-289-4136
        </phone>
</customerinfo>


    <customerinfo>

            <name>
                    Fred Doe
            </name>
            <addr country=“USA">
                    <prov-state>
                            Iowa
                    </prov-state>
                    <pcode-zip>

987653
                    </pcode-zip>
            </addr>
            <phone type=“Mobile">
                    244-775-3151
            </phone>
</customerinfo>
```

Result sequence

John Smith
Fourth
Calgary
Alberta
M1T 2A9
963-289-4136
Fred Doe
Iowa
987653
244-775-3151

XML Access through all DB2 APIs

JDBC
.NET
ODBC
CLI
pureQuery
Embedded SQL
- in C++
- in COBOL
- in PLI
- in Basic Assembler

Session on JDBC and pureXML
https://www.ibm.com/developerworks/wikis/display/db2xml/devotee#devotee-jdbc

Article on using pureQuery with pureXML
http://www.ibm.com/developerworks/data/library/techarticle/dm-0901rodrigues/

Article on using Cobol with pureXML
http://www.ibm.com/developerworks/data/library/techarticle/dm-0905purexmlapp1/

XML Encoding Declaration

```
<?xml version="1.0" encoding="UTF-8" ?>
<customer>
      <name>John Smith</name>
      <addr country=“Canada">
            <street>Fourth</street>
            <city>Calgary</city>
            <state>Alberta</state>
            <zip>M1T 2A9</zip>
      </addr>
      <phone type="work">963-289-4136</phone>
</customer>
```

XML Encoding Declaration

XML Declaration

- XML declaration and encoding declaration are optional.
- XML declaration is not stored with a document in DB2
- XML declaration can be generated by DB2 upon retrieval (based on API or query options)
- XML is stored in DB2 in UTF-8

Approaches for Querying XML

- Plain SQL without any XQuery or XPath
  - only useful for full-document retrieval and operations such as insert, delete, and update of whole documents.
- SQL/XML with XQuery or XPath embedded in SQL statements provides the broadest functionality and the least restrictions.
- XQuery
  - powerful query language, specifically designed for querying XML data.
  - good option if your applications require querying and manipulating XML data only, and do not involve any relational data.
- XQuery with embedded SQL
  - good choice if you want to leverage relational predicates and indexes as well as full-text search to pre-filter the documents from an XML column which are then input to an XQuery.
  - SQL embedded in XQuery also allows you to run external functions on the XML columns.

Summary

- **DB2 9 supports two query languages: XQuery and SQL/XML**
- **XPath**
  - Cornerstone for both XQuery and SQL/XML standard.
  - Used in other XML technologies and interfaces, e.g., XSLT
  - Provides the ability to navigate within XML documents
  - Supports namespaces, comparison operators, and built-in functions
- **XPath is used in other parts of pureXML support, eg., defining XML indexes**
- **DB2 query support is namespace aware**
- **XML is stored in DB2 in UTF-8**
- **SQL/XML with embedded XPath provides broadest functionality**

Resources

1. DB2 pureXML cookbook: http://tinyurl.com/pureXML
   Comprehensive coverage of pureXML in DB2 for Linux, UNIX, Windows andDB2 for z/OS
2. DB2 pureXML enablement wiki:
   http://www.ibm.com/developerworks/wikis/display/db2xml

- DB2 pureXML for zOS Wiki: http://www.ibm.com/developerworks/wikis/display/db2xml/DB2+for+zOS+pureXML
- DB2 pureXML alphaWorks: http://www.alphaworks.ibm.com/tech/purexml (demo downloads)
- DB2 pureXML devotee site: http://www.ibm.com/developerworks/wikis/display/db2xml/devotee
- DB2 pureXML bootcamps: http://www.ibm.com/developerworks/wikis/display/db2xml/BootCamps

Agenda - Compression

- Why Compression? Why DB2 LUW?
- Static Row Compression (Table)
- XML Compression
- Index Compression

From the existing DB2 9 Static Compression …

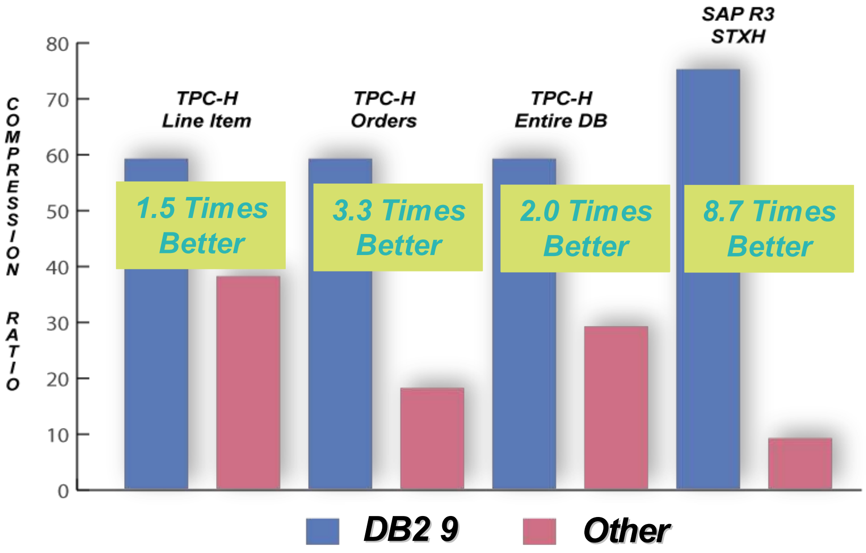
*"With DB2 9, we're seeing **compression rates up to 83%** on the Data Warehouse. The projected **cost savings are more than $2 million initially with ongoing savings of $500,000 a year.**"* - Michael Henson

SUNTRUST

*"We achieved a 43 per cent saving in total storage requirements when using DB2 with Deep Compression for its SAP NetWeaver BI application, when compared with the former Oracle database, **The total size of the database shrank from 8TB to 4.5TB, and response times were improved by 15 per cent**. Some batch applications and change runs were reduced by a factor of ten when using IBM DB2."* - Markus Dellermann

SCHAEFFLER GROUP

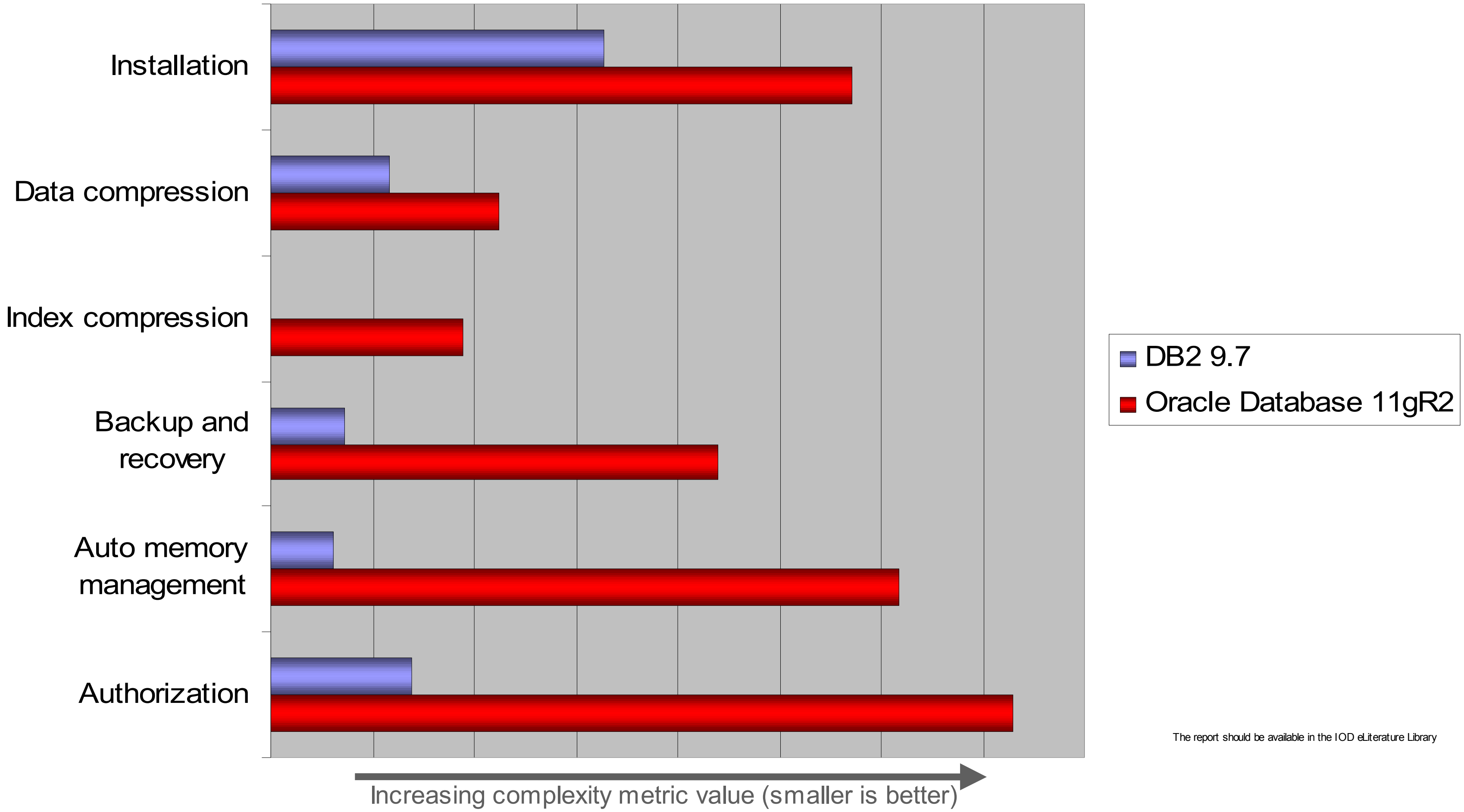
- **Reduce storage costs**
- **Improve application performance**
- **Easy to implement**



Why Compression? Why with DB2 LUW?

- **Compression**
  - Reduces storage and administration cost.
- **Why with DB2 LUW?**
  - On disk data (data, logs) is compressed
    - Reduces IO bandwidth requirement
    - Improves application response time (IUD, queries)
  - Data in bufferpool is compressed
    - Increased bufferpool hit rate
    - Reduces page latch and increases concurrency
  - Automatic
  - Reduces bigger storage cost using idle CPU cycles
  - Complementary technologies (adaptive compression complements static compression)
  - IBM is investing BIG in compression. Your savings will go up!!

DB2 makes administration easier, less complex



The report should be available in the IOD eLiterature Library

Source: “*Comparing DBA Productivity: An Oracle/DB2 Task Complexity Analysis”, Triton Consulting, October 2010*.

Static Compression (Table)

Unique in the industry

- **V9.1**
  - INSPECT ROWCOMPESTIMATE to estimate storage saving
  - CREATE TABLE …. COMPRESS YES
  - Classic REORG TABLE RESETDICTIONARY/ KEEPDICTIONARY creates dictionary and compresses the data.
  - Online INSPECT creates the dictionary. New data will be compressed.
  - Leave the rest upon DB2!!
  - Monitor compression effectiveness
    - New columns in catalog views SYSCAT.TABLES and SYSSTAT.TABLES

Static Compression (Table)

Unique in the industry

- **V9.5**
  - ADC – automatic dictionary creation
  - Instead of whole table scan, stop when it has found minimum data required to create very good dictionary
  - Manual control:
    - LOAD REPLACE … RESETDICTIONARY/KEEPDICTIONARY
  - SQL administration function to estimate storage savings:
    - ADMIN_GET_TAB_COMPRESS_INFO
- **V9.7**
  - Static compression is compatible with replication (DATA CAPTURE CHANGES)

Best Practices (Table Compression)

- **Choose the right table**
  - Biggest ones first
    - SQL administration function: SYSPROC.ADMIN_GET_TAB_INFO
    - Column: DATA_OBJECT_P_SIZE
  - Tables with higher read/write ratio (70/30 and higher) benefit the most.
  - Compression ratio is typically higher for range-partition and DPF tables
- **DSS queries benefit the most**
- **If possible, go with LARGE tablespaces**
  - Can insert more records to a page
  - Regular tablespaces: ~250 records per page maximum.
- **Classic REORG and INSPECT produces best dictionary**
  - If possible, perform a classic REORG once, to compress all existing data
  - INDEXSCAN keyword uses index-scan (two passes over table)
  - Default is table-scan + sort (recommended if we have enough memory)
- **After compression, reduce HWM to reclaim free space, still allocated to tablespace**
  - db2dart /dhwm

More Compression (XML)

- **V9.5**
  - Compress inline XML documents
    - CREATE (or ALTER) TABLE T1(…C1 XML inline length 4000) COMPRESS YES
- **V9.7**
  - ADMIN_EST_INLINE_LENGTH function - Estimate length required to inline data
  - ADMIN_IS_INLINED function – Tells if XML document is inlined
  - Compress all XML documents (including large ones)
    - CREATE TABLE T1 (c1 XML) COMPRESS YES

Example

Query

```
db2 => SELECT PK, ADMIN_IS_INLINED(xml_doc1) as IS_INLINED,
  ADMIN_EST_INLINE_LENGTH(xml_doc1) as EST_INLINE_LENGTH
  from TAB1
```

Result

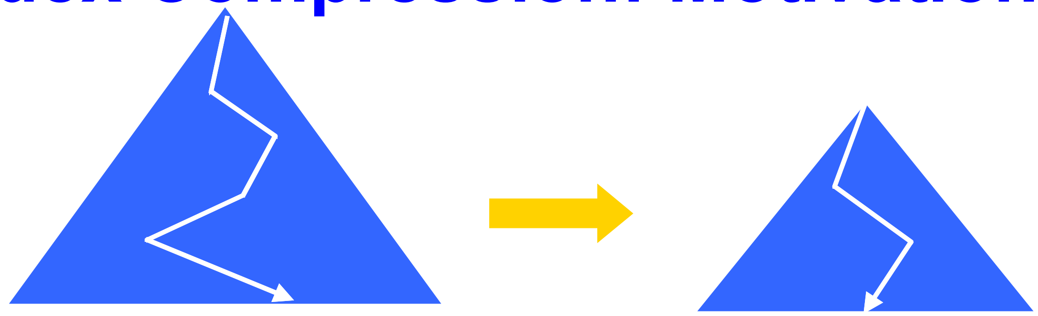
```
PK   IS_INLINED   EST_INLINE_LENGTH
---  ----------   -----------------
1       1           1028
2       0           2092
3       0           -1               /* too big */
```

More Compression (Index)

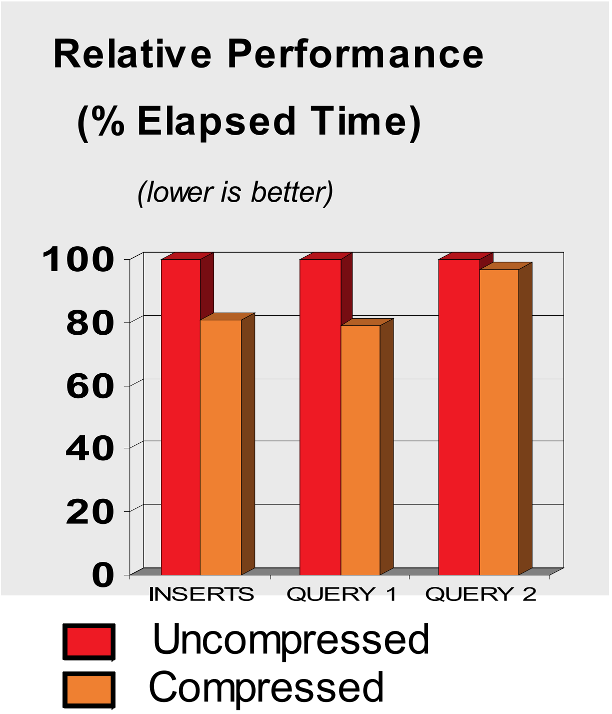
Unique in the industry

- **V9.7**
  - CREATE TABLE T1 …COMPRESS YES
    - Will compress all indexes by default
    - Can override with “CREATE/ALTER INDEX …COMPRESS YES/NO”
    - Perform INDEX reorg to rebuild indexes
    - REORGCHK command to estimate compression savings.
    - ADMIN_GET_INDEX_COMPRESS_INFO table function
      - PCT_PAGES_SAVED column
      - Reports compression saving

Index Compression: Motivation and Benefits



- Fewer index leaf pages
  - Fewer logical and physical I/Os for index scans
  - Fewer splits
  - Better bufferpool hit ratio
- Fewer index levels
  - Fewer logical and physical I/Os for key search (insert, delete, select)
  - Better bufferpool hit ratio
- Performance
  - Some additional CPU cycles needed for compress / decompress
    - 0-10% in various measurements
  - Typically outweighed by reduction in I/O resulting in higher overall throughput



Index Compression : Overview

- New INDEX compression attribute
  - CREATE INDEX … **COMPRESS YES**
  - ALTER INDEX … **COMPRESS YES**
    - Following ALTER INDEX, index rebuild/reorg required to compress

- An INDEX will be defined as compressed when created if:
  - The index COMPRESS attribute is specified as YES, or,
  - The index COMPRESS attribute is not specified and the table COMPRESS attribute is YES
    - Catalog, MDC block, and XML meta/path indexes are not eligible for compression

- New columns added to SYSCAT.INDEXES and SYSCAT.INDEXPARTITIONS
  - COMPRESS
  - PCTPAGESSAVED

Index Compression *(continued)*

- Two new table functions for indexes and index compression
  - ADMIN_GET_INDEX_INFO
  - ADMIN_GET_INDEX_COMPRESS_INFO

- Index compression is dynamic and on-the-fly
  - No need to reorg indexes to retain high compression rates
    - Table row compression is based on a static data dictionary and can benefit from reorg

- Index pages stored compressed on disk and in bufferpool
  - As with table row compression

- Three different compression techniques applied to index leaf pages
  - Meta-data compression

Index Compression Estimation

- ADMIN_GET_INDEX_COMPRESS_INFO()
  - Table function *similar* to ADMIN_GET_TABLE_COMPRESS_INFO
  - Compression attribute/status (also in ADMIN_GET_INDEX_INFO)
    - COMPRESS_ATTR, as defined by DDL
    - INDEX_COMPRESSED, actual physical status
  - Compression statistics
    - COMPRESS_ATTR = ‘N’, will estimate percent of pages saved if index were to be compressed
    - COMPRESS_ATTR = ‘Y’, will report actual statistics from SYSCAT views

Index Compression: Estimation and Statistics

```
SELECT index_name, pages_saved_percent, compress_attr, index_compressed
FROM TABLE (SYSPROC.ADMIN_GET_INDEX_COMPRESS_INFO
                              (‘T’, ‘myschema’, ‘T1’, ‘’, ‘’)) AS T

INDEX_NAME  PERCENT_PAGES_SAVED   COMPRESS_ATTR  INDEX_COMPRESSED
----------  -------------------   -------------  ----------------
INDEX1      57                    N              N
```

Estimated savings

```
ALTER INDEX index1 COMPRESS YES
REORG INDEXES ON TABLE t1
RUNSTATS ON TABLE t1
SELECT index_name, pages_saved_percent, compress_attr FROM SELECT index_name,
pages_saved_percent, compress_attr, index_compressed     FROM TABLE
(SYSPROC.ADMIN_GET_INDEX_COMPRESS_INFO
              (‘T’, ‘myschema’, ‘T1’, ‘’, ‘’)) AS T

INDEX_NAME  PERCENT_PAGES_SAVED   COMPRESS_ATTR  INDEX_COMPRESSED
----------  -------------------   -------------  ----------------
INDEX1      58                    Y              Y
```

Actual savings

Index Compression Comparison 101

DB2 9.7 Index Compression:

- Combines three compression algorithms – variable slot directory, RID-list compression and prefix compression
  - Higher compression ratios
- DB2 automatically selects all beneficial compressions and optimal column prefixes
  - Easier setup, no manual maintenance
- DB2 dynamically adjusts index prefixes during DML to optimize compression ratios
  - Compression ratios are always optimal
  - No manual maintenance is required to optimize the columns to prefix as table data changes
  - DB2 dynamically identifies beneficial prefixes for repeating column values
    - Index size will never increase

Oracle Index Compression:

- Contains only one compression algorithm – prefix compression
  - Oracle has no RID-lists
  - Lower compression ratios
- DBA must manually identify and define the optimal number of leading columns to prefix
  - Requires more effort to use and maintain
- The number of columns to prefix is statically defined in the index DDL
  - As data changes, the optimal number of columns to prefix can change
  - DBA must manually ALTER and rebuild the index to optimize the compression ratio
  - Defined prefixes are always stored in block header even if they don't repeat
    - Index size could actually increase!

Additional Information

- **References:**
  - **DB2 Best Practices** http://www.ibm.com/developerworks/data/bestpractices/
  - **Static Compression – Best Practises** http://www.ibm.com/developerworks/data/bestpractices/deepcompression/#bestpractices
  - **Index Compression** http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.dbobj.doc/doc/c0054539.html
  - **Value Compression** http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/c0007306.html

- **Feedback**
  - Presentation format and contents
  - Additional DB2 topics you are interested
  - Follow on questions for the presentation

- **Contact: fning@ca.ibm.com**