Common pureXML Use Cases
With Practical Examples

Philip Nelson
Lloyds Banking Group / ScotDB Limited
teamdba@scotdb.com
Agenda

- When To Use XML (And When Not To)
- Use Cases Covered In This Presentation
- Show Us The Code …
- Other pureXML Tips and Tricks
When To Use XML

- From the pureXML Cookbook (Chapter 1)
  - When the schema is volatile
  - When data is inherently hierarchical in nature
  - When data represents business objects
  - When objects have sparse attributes
  - When data needs to be exchanged
Volatile Schemas

- When the data model frequently changes
- Reasons why this may happen –
  - External change (e.g. HMRC schemas)
  - Evolving use (e.g. document store)
- Not to be confused with bad design
  - e.g. using data values as elements
Hierarchical Data

- Many “real world” situations
- Examples –
  - Bill of Materials
  - Departmental hierarchies
- Issues with relational representation
  - Complex coding
  - Potential for poor performance
Data As Business Objects

- Complex data structures
- Normally read / written as a whole
- Issues with relational representation
  - Cost of composition / decomposition greater than the cost of all other processing
  - Many tables blurs understanding
    - Views can help a bit but are still “flat” in structure
Sparse Attributes

- Data objects with many attributes
- Only a small subset of attributes known
- Relational representation: null columns
- Issues with relational representation
  - Wasted storage (compression helps though)
  - Unable to see the wood for the trees …
Data Exchange

- XML’s original use
  - Self-describing data
  - Many industry standards
- Not only useful for external interfaces
  - Exchange between internal divisions
  - Exchange between different technologies
- Special case
  - Basis of web service interfaces
Use Cases To Be Covered

- Audit trails
- XML construction for data interchange
- XML data as a business object
- Simplifying web service construction
Audit Trail : External Messages

- Need to store all messages coming into or going out from our “domain”
- Facility EMA = External Message Audit
- Interfaces to –
  - Customers (e.g.
  - Suppliers
  - Regulatory and government bodies
Audit Trail Table

CREATE TABLE DBEMA001.BOUNDARY_EVENT
(EVENT_TIMESTAMP TIMESTAMP
    NOT NULL WITH DEFAULT CURRENT TIMESTAMP,
    EVENT_XML XML,
    DIRECTION_FLAG CHAR(1),
    VALIDATION_FAILURE_REASON VARCHAR(300)
) IN TS_EMA
;

- Timestamp (auto-generated) for sequencing
- XML column for message
- Direction is (I)nbound or (O)utbound
- If validation failure still store message but record reason for failure
Register XSDs

REGISTER XMLSCHEMA "http://www.govtalk.gov.uk/CM/envelope"
   FROM xsd/envelope-v2-0-HMRC.xsd AS govtalk_envelope;
ADD XMLSCHEMA DOCUMENT TO govtalk_envelope
   ADD "mov-2010-v1-0.xsd" FROM xsd/mov-2010-v1-0.xsd;
ADD XMLSCHEMA DOCUMENT TO govtalk_envelope ADD
   "http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/xmldsig-core-schema.xsd"
   FROM xsd/xmldsig-core-schema.xsd;
ADD XMLSCHEMA DOCUMENT TO govtalk_envelope
   ADD "EOY-2010-v1-0.xsd" FROM "xsd/EOY-2010-v1-0.xsd";
ADD XMLSCHEMA DOCUMENT TO govtalk_envelope
   ADD "http://www.hmrc.gov.uk/schemas/core-v2-0.xsd"
   FROM "xsd/core-v2-0.xsd";
COMPLETE XMLSCHEMA govtalk_envelope;
CREATE PROCEDURE DBEMA001.SP001SAVE_INBOUND_EVENT ( IN pBoundaryEventXML XML, OUT pValid CHAR(1))
LANGUAGE SQL SPECIFIC sp001save_inbound_event
BEGIN
  -- variable declarations removed for space reasons
  --
  -- Define name for XML validation SQLSTATE
  --
  DECLARE VALIDATION_FAILURE CONDITION FOR '2200M';
  --
  -- Exception handler to store inbound XML which fails validation
  --
  DECLARE EXIT HANDLER FOR VALIDATION_FAILURE
  BEGIN
    GET DIAGNOSTICS exception 1 vTokenString = DB2_TOKEN_STRING, vErrorMessage = MESSAGE_TEXT;
    SELECT SQLSTATE, SQLCODE INTO vSQLSTATE, vSQLCODE FROM SYSIBM.SYSDUMMY1;
    SET pValid = 'N';
    INSERT INTO DBEMA001.T0010BOUNDARY_EVENT
    (event_xml,direction_flag,validation_failure_reason)
    VALUES
    (pBoundaryEventXML,'I',vErrorMessage);
  END;
END;
Message Storage SP (2)

--
-- Main procedure body
--
BEGIN
  SET pValid = 'Y';
  --
  -- Save inbound XML with validation
  --
  INSERT INTO DBEMA001.T0010BOUNDARY_EVENT
    (BUSINESS_EVENT_XML, DIRECTION_FLAG)
  VALUES
    (XMLVALIDATE(pBoundaryEventXML),'I')
;
END;
END#
Validation : Schema Identification

XMLVALIDATE() uses SchemaLocation to locate XSD in XML Schema Repository

- Fine if we have complete control over XML
- Allows generic stored procedure for all XSDs
- XMLs from external sources
  - So far 100% need specific storage SPs

XMLVALIDATE(<xml> ACCORDING TO XMLSCHEMA ID <xsd-ref>)

15
XML Construction Example

- Build XML containing a day’s transactions
- Add summary (counts and money totals)
SELECT XMLDOCUMENT(
    XMLELEMENT(NAME "six:Transactions",
        XMLNAMESPACES(
            'http://v1.six.im.scottishwidows.co.uk' AS "six",
            'http://www.w3.org/2001/XMLSchema-instance' AS "xsi"),
        XMLATTRIBUTES(
            'http://v1.six.im.scottishwidows.co.uk SIX.xsd'
            AS "xsi:schemaLocation"),
        XMLELEMENT(NAME "six:ExtractNumber",0),
        XMLELEMENT(NAME "six:ExtractDate", CURRENT TIMESTAMP),
        XMLAGG(XMLELEMENT(NAME "six:Transaction",
            XMLFOREST(A.TX_ID     AS "six:TranID",
                A.TX_AMOUNT AS "six:TranAmount",
                …
            ))),
        XMLELEMENT(NAME "six:TranCount",0),
        XMLELEMENT(NAME "six:TranTotal",0)
    ) as xmldoc
from
    DBCDI001.TRAN_TABLE A;
XQuery Updates

XMLQUERY('copy $new := $XMLDOC modify (  
    do replace value of $new/*:Transactions/*:ExtractNumber with $ex,
    for $j in $new/*:Transactions/*:Transaction/*:TranID
        return
    do replace value of $j with $j + $pi,
    do replace value of $new/*:Transactions/*:TranCount
        with $new/*:Transactions/count(*:Transaction),
    do replace value of $new/*:Transactions/*:TranTotal
        with
        $new/*:Transactions/sum(*:Transaction/*:TranAmount)
    )
    return $new'
    PASSING EXTRACT_NUMBER AS "ex",
    PREV_XACT_ID AS "pi"
)
from
( .. the previous SQL/XML construction .. and two other columns .. )
Business Object as XML
(Florist Shop Directory)

- Directory of shops provided as XML
- Size of directory is approximately 10 meg
- Recommended weekly refresh except ...
  - Busy times (Valentines Day, Mothers Day)
  - Recommended 15 minute refresh
- Not using latest directory affects business
  - Florists signal availability via updated directory entry
  - May send to a florist who has already signalled no longer able to deliver (e.g. out of red roses)
  - By time rejection message sent, all others may also be out of stock ...
<?xml version="1.0" encoding="UTF-8" ?>
<memberDirectoryInterface>
    <searchShopResponse>
        <errors/>
        <shops>
            <shop>
                <shopCode>B4110000</shopCode>
                <name>HACKENSACK FLORIST</name>
                <address>
                    <attention>Peter B. Tournas</attention>
                    <addressLine1>357 Essex Street</addressLine1>
                    <city>HACKENSACK</city>
                    <state>NJ</state>
                    <zip>07601</zip>
                    <countryCode>USA</countryCode>
                    <timeZone>EAST</timeZone>
                </address>
            </shop>
        </shops>
    </searchShopResponse>
</memberDirectoryInterface>
<sundayIndicator>N</sundayIndicator>
<phoneNumber>2013436400</phoneNumber>
<faxNumber>2013876214</faxNumber>
</email/>
<servicedZips>
  <zip>07071</zip>
  <zip>07407</zip>
</servicedZips>
<nonDeliveryDates>
  <date>01/09/2008</date>
  <date>25/12/2008</date>
</nonDeliveryDates>
<shopStatus>A</shopStatus>
<communicationCode>1</communicationCode>
</shop>
</shops>
</searchShopResponse>
Florist Directory Handling

- Specification
  - Store retrieved directory intact (timestamped)
  - Always used the latest directory for queries
  - Retrieve all florists who serve a given zipcode

- Some design decisions
  - All “wire service” messages in one table
  - Messages stored in an XML column
SELECT MESSAGE_XML FROM DBFOX001.WIRE_SERVICE_MESSAGES a WHERE XML EXISTS('{$MESSAGE_XML/memberDirectoryInterface/searchShopResponse/shops/shop/servicedZips[zip > ""]}'
AND WIRE_SERVICE_ID =
(SELECT ID FROM DBFOX001.WIRE_SERVICES WHERE WS_NAME = 'BloomNet')
AND INBOUND_MESSAGE = 1
AND MESSAGE_TIMESTAMP =
(SELECT MAX(MESSAGE_TIMESTAMP) FROM DBFOX001.WIRE_SERVICE_MESSAGES A
-- Uses zip simply because XML index defined already
WHERE XML EXISTS('{$MESSAGE_XML/memberDirectoryInterface/searchShopResponse/shops/shop/servicedZips[zip > ""]}'
AND WIRE_SERVICE_ID =
(SELECT ID FROM DBFOX001.WIRE_SERVICES WHERE WS_NAME = 'BloomNet')
AND INBOUND_MESSAGE = 1
);
Finding Florist By Zip Code

```
select
t.ShopCode, t.shopname, t.zipcode,
case when a.shopcode is not null and t.zipcode = pZipCode then 1
    when a.shopcode is not null then 2
    when a.shopcode is null and t.zipcode = pZipCode then 3
else 4
end as display_priority
from ( ... inner select ...) as i,
xmltable('{$c/memberDirectoryInterface/searchShopResponse/shops/
    shop[servicedZips/zip = $z]' as "c",
    passing i.message_xml as "c", pZipCode as "z"
    columns shopname varchar(100) path 'name',
           zipcode  varchar(100) path 'address/zip',
           ShopCode varchar(100) path 'shopCode') as t
left outer join dbfox001.bloomnet_toptiers a
on t.ShopCode = a.shopcode
order by display_priority, t.ShopCode, t.ZipCode;
```
Web Service Construction

- Web service to handle orders (for florists)
- Extension to existing web application
  - Written using Ruby on Rails
  - Order details stored in relational tables
Example Order XML (Part 1)

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<FoxOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="FoxOrder.xsd">
  <Security>
    <Username>my-userid</Username>
    <Password>my-password</Password>
  </Security>
  <Purchaser>
    <FirstName>Angus</FirstName>
    <Surname>McLeod</Surname>
  </Purchaser>
  <Recipient>
    <FirstName>Fiona</FirstName>
    <Surname>Macgregor</Surname>
  </Recipient>
  <DeliveryDetails>
    <DeliveryDate>2008-02-14</DeliveryDate>
  </DeliveryDetails>
</FoxOrder>
```
Example Order XML (Part 2)

```xml
<Products>
  <Product>
    <ProductCode>TF31-1</ProductCode>
    <ProductName>Big Red Roses</ProductName>
    <DisplayPriceDesc>Deluxe</DisplayPriceDesc>
    <Quantity>1</Quantity>
    <ProductPrice>59.95</ProductPrice>
    <Discount>5.00</Discount>
  </Product>
</Products>

<ConfirmationNumber>123456</ConfirmationNumber>

<PaymentDetails>
  <CCNumber>encryptednumber</CCNumber>
  <CCTypeID>1</CCTypeID>
  <CCAuthCode>12ABCD</CCAuthCode>
  <CCSecurityCodeResponse>1</CCSecurityCodeResponse>
</PaymentDetails>

</FoxOrder>
```
Rails Code To Handle the XML

- **Controller** *(app/controllers/orders_controller.rb)*
  ```ruby
def public_order
  render :layout=>false
  porder = params[:data]
  preply = Order.find_by_sql(
    "call dbfox001.sp004public_order(?), porder")
  return @reply = preply[0][1]
end
```

- **Routing** *(config/routes.rb)*
  ```ruby
  map.purchases '/purchases',
    :controller => 'orders',
    :action => 'public_order'
  ```

- **View** *(app/views/orders/public_order.rhtml)*
  ```ruby
  <%= @preply %>
  ```
Order Processing Stored Proc

- Rails code is very simple
- “Heavy lifting” is in stored procedure
- Following extracts show –
  - Converting character input to XML datatype
  - Extraction of data using SQL/XML
  - Use of XQuery to check document contents
- Outline specification
  - Store inbound document intact (audit trail)
  - Check security credentials against database
  - Extract some data to relational format (for Rails use)
  - Validate all data has been processed successfully
CREATE PROCEDURE DBFOX001.SP004PUBLIC_ORDER
(IN pOrder1 CLOB(32k))
LANGUAGE SQL
SPECIFIC SP004PUBLIC_ORDER
DYNAMIC RESULT SETS 1
...
DECLARE pOrder XML;
DECLARE pReply XML;
BEGIN
  -- Store order (turn into XML data type in process)
  SELECT ID, MESSAGE_TIMESTAMP, MESSAGE_XML
  INTO vInboundOrderID, vInboundOrderTimestamp, pOrder
  FROM FINAL TABLE
  (INSERT INTO DBFOX001.INBOUND_ORDERS
    (MESSAGE_XML) VALUES (pOrder1))
  );
...

BEGIN
    -- Validate username and password are valid
    DECLARE EXIT HANDLER FOR NOT FOUND
        SIGNAL cSecurityError
            SET MESSAGE_TEXT = 'Security Credentials Invalid';
    SELECT 1 INTO vCheck FROM DBFOX001.SOURCE_WEBSITES
        WHERE (USERNAME, PASSWORD, ID) IN
            (SELECT T.Username, T.Password, T.URLUsed FROM
              XMLTABLE('$o/FoxOrder' PASSING pOrder AS "o"
                COLUMNS
                    Username  VARCHAR(50)   PATH './Security/Username',
                    Password  VARCHAR(50)   PATH './Security/Password',
                    URLUsed   INTEGER     PATH './Marketing/URLUsed'
                     ) AS T);
END;
Validate Number of Orders: XQuery

-- ... lots of other XML extractions precede this ... 
-- Compare line items added against the number on XML 
--

SET vInProductCount = XMLCAST( XMLQUERY('count($o/FoxOrder/Products/Product)'
PASSING pOrder AS "o") as INTEGER);

IF vInProductCount <> vStoredProductCount THEN
    SIGNAL cProductCountError
    SET MESSAGE_TEXT = 'Problem storing products';
END IF;
CREATE PROCEDURE DBFOX001.SP003BLOOMNET_GET () DYNAMIC RESULT SETS 1
P1: BEGIN
  DECLARE cursor1 CURSOR WITH RETURN FOR
  SELECT
    XMLSERIALIZE (XMLDOCUMENT (  
      XMLELEMENT NAME "ForeignSystemInterfaceOutboundRequest",  
      XMLCONCAT (XMLELEMENT (NAME "security",  
        XMLCONCAT (XMLELEMENT (NAME "username", ws_user),  
        XMLELEMENT (NAME "password", ws_pw),  
        XMLELEMENT (NAME "shopCode", ws_shopcode)),  
        XMLELEMENT (NAME "fulfillerShopeode", ws_shopcode),  
        XMLELEMENT (NAME "systemType", 'GENERAL'))) AS CLOB  
    FROM (SELECT WS.TRANSMISSION_USERID AS ws_user,  
        WS.TRANSMISSION_PASSWORD AS ws_pw,  
        WS.CUSTOMER_REFERENCE AS ws_shopcode  
        FROM DBFOX001.WIRE_SERVICES AS WS  
        WHERE WIRE_SERVICE_NAME = 'BloomNet') AS doc_list;
  OPEN cursor1;
END P1#