Querying XML data with Oracle through the semantic layer

Marc Daniau – Product group

marc.daniau@sap.com

Business Objects™
an SAP company

Introduction

This presentation describes how to query XML data

- Inside the database from a hybrid table, an XML table or a function returning an XML output
- Outside the database from a file system or web data.

It shows an example of heterogeneous query joining a relational table with RSS feed data.

It includes also a WebI report viewing XML data transformed into HTML using XSL style sheets.

Topics

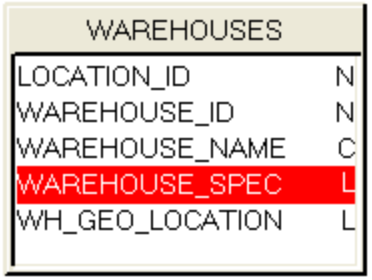
Business Objects
an SAP company

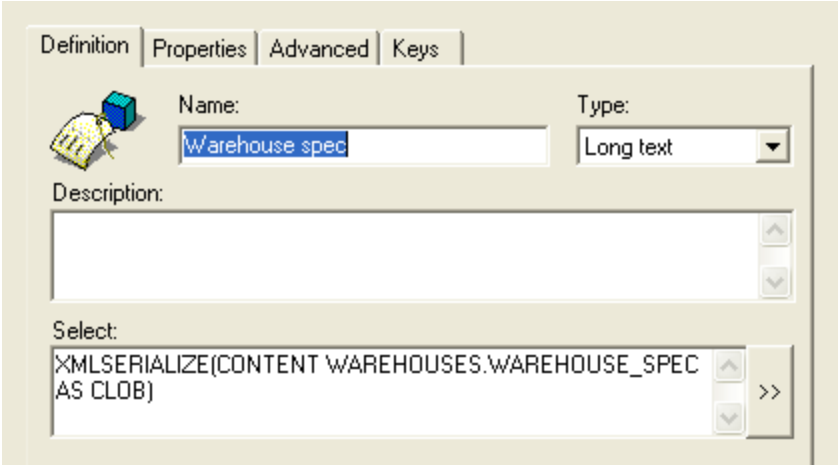
- **Querying XML data inside the database**
- Querying XML data outside the database
- Transforming XML into HTML
- Transforming relational into HTML

XMLSERIALIZE() function
Converting XML into long text

The warehouses demo table is a hybrid table that contains an XML column among relational columns. We built an object against the warehouse_spec XML column.

```
SQL> desc warehouses
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 WAREHOUSE_ID                  NOT NULL NUMBER(3)
 WAREHOUSE_SPEC                         SYS.XMLTYPE
 WAREHOUSE_NAME                         VARCHAR2(35)
 LOCATION_ID                            NUMBER(4)
 WH_GEO_LOCATION                        MDSYS.SDO_GEOMETRY
```





XMLSERIALIZE() function
Converting XML into long text

We restitute the XML content as one block.

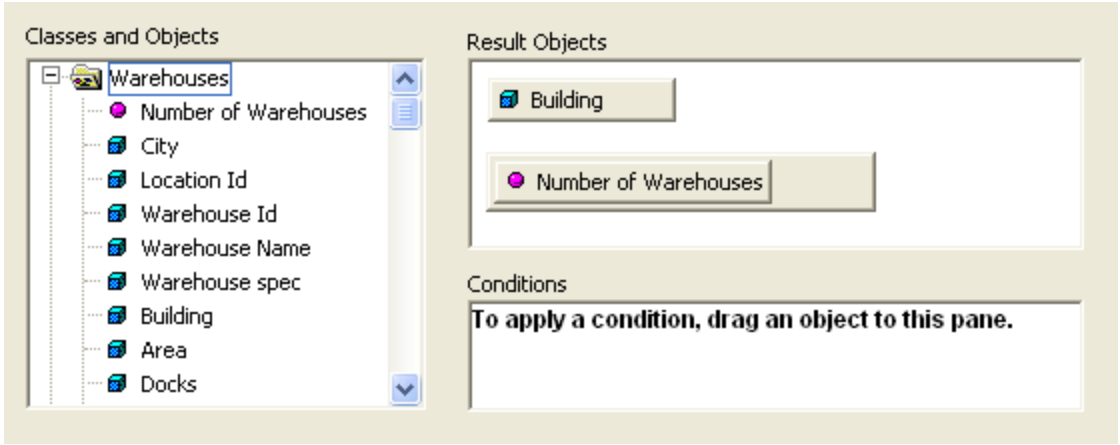
The following report has one row per warehouse.

Warehouse Id	Warehouse Name	Warehouse spec
1	Southlake, Texas	<?xml version="1.0"?> <Warehouse> <Building>Owned</Building> <Area>25000</Area> <Docks>2</Docks> <DockType>Rear load</DockType> <WaterAccess>Y</WaterAccess> <RailAccess>N</RailAccess> <Parking>Street</Parking> <VClearance>10 ft</VClearance> </Warehouse>
2	San Francisco	<?xml version="1.0"?> <Warehouse> <Building>Rented</Building> <Area>50000</Area> <Docks>1</Docks> <DockType>Side load</DockType> <WaterAccess>Y</WaterAccess> <RailAccess>N</RailAccess> <Parking>Lot</Parking> <VClearance>12 ft</VClearance> </Warehouse>
3	New Jersey	<?xml version="1.0"?> <Warehouse> <Building>Rented</Building> <Area>85700</Area> <DockType></DockType> <WaterAccess>N</WaterAccess> <RailAccess>N</RailAccess> <Parking>Street</Parking> <VClearance>11.5 ft</VClearance> </Warehouse>

EXTRACTVALUE() function
Shredding an XML node into columns

We define one object for each XML tag in warehouse_spec.

Here is a sample query using the « Building » object.



Building	Number of Warehouses
<none>	5
Owned	2
Rented	2
Total	9

EXTRACTVALUE() function
Shredding an XML node into columns

The warehouses XML information broken down.

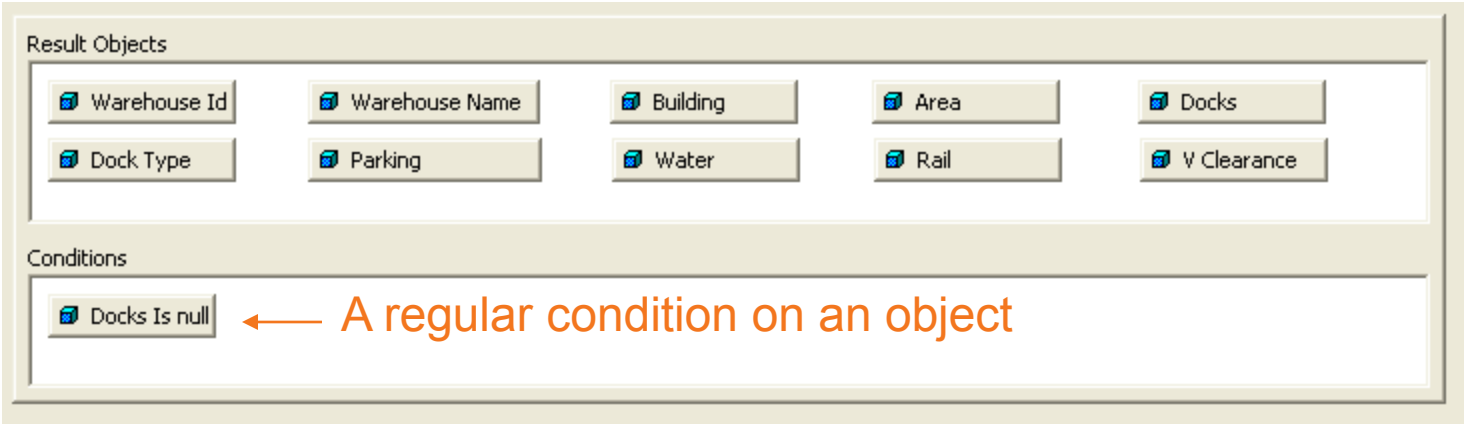
Note that only US warehouses have spec data.

Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	V Clearance
1	Southlake, Texas	Owned	25000	2	Rear load	Street	Y	N	10 ft
2	San Francisco	Rented	50000	1	Side load	Lot	Y	N	12 ft
3	New Jersey	Rented	85700			Street	N	N	11.5 ft
4	Seattle, Washington	Owned	103000	3	Side load	Lot	N	Y	15 ft
5	Toronto								
6	Sydney								
7	Mexico City								
8	Beijing								
9	Bombay								

EXTRACTVALUE() function
Filtering the extracted value

The condition can be defined in the query panel.

```
WHERE EXTRACTVALUE(WAREHOUSES.WAREHOUSE_SPEC, '/Warehouse/Docks') IS NULL
```



Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	V Clearance
3	New Jersey	Rented	85700			Street	N	N	11.5 ft
5	Toronto								
6	Sydney								
7	Mexico City								
8	Beijing								
9	Bombay								

EXISTSNODE() function
Binary condition on a XML node

We have defined the filter « Docks is empty » in the universe. It returns “New Jersey” only. Warehouses where warehouse_spec is null are discarded.

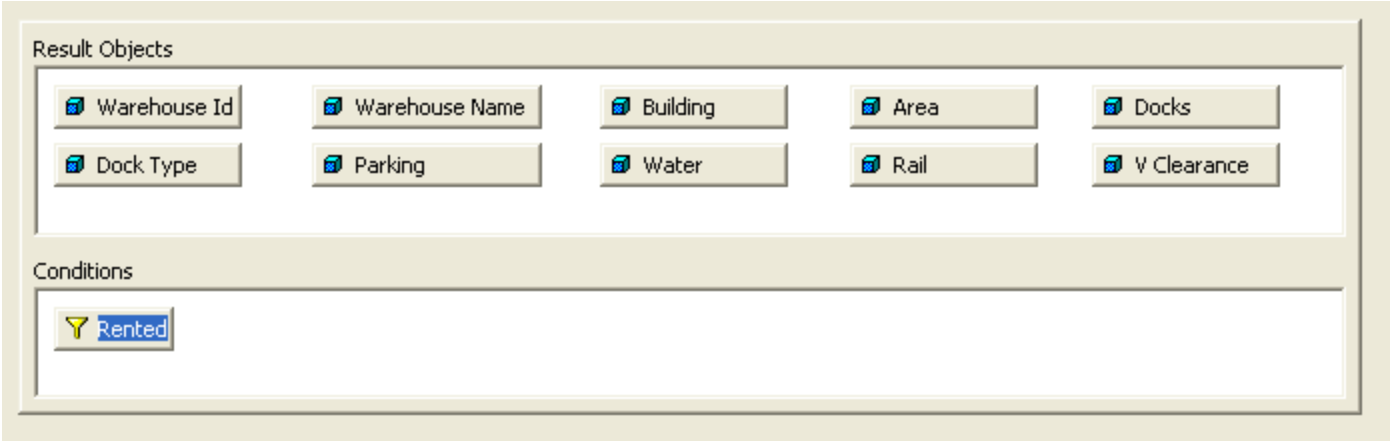
```
WHERE EXISTSNODE(WAREHOUSES.WAREHOUSE_SPEC, '/Warehouse/Docks') = 0
```

Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	V Clearance
3	New Jersey	Rented	85700			Street	N	N	11.5 ft

EXISTSNODE() function
Binary condition on a XML node

A filter using an XPath predicate.

```
WHERE EXISTSNODE(WAREHOUSES.WAREHOUSE_SPEC,
'/Warehouse[Building="Rented"]') = 1
```

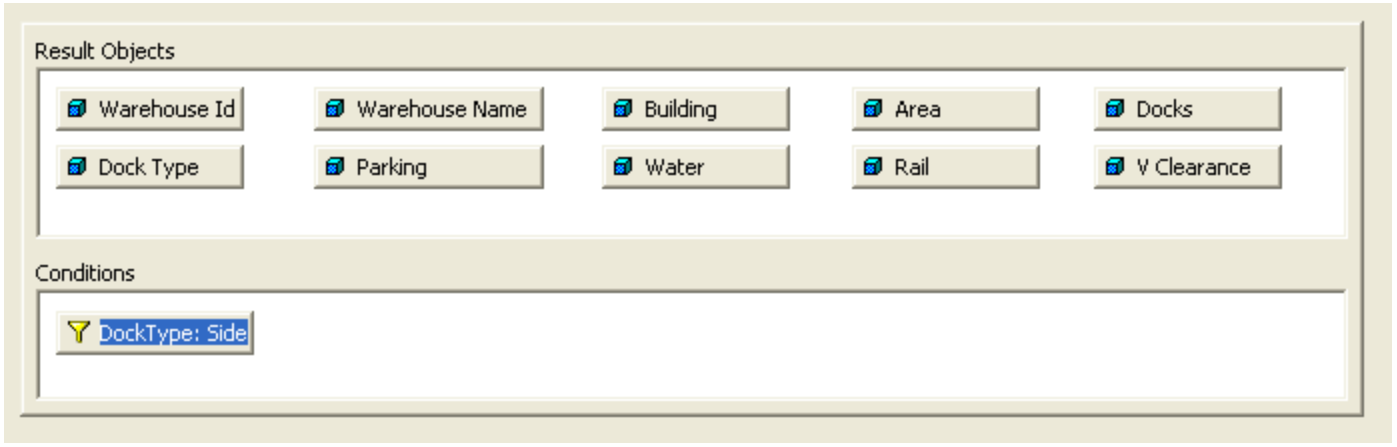


Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	V Clearance
2	San Francisco	Rented	50000	1	Side load	Lot	Y	N	12 ft
3	New Jersey	Rented	85700			Street	N	N	11.5 ft

EXISTSNODE() function
Searching text

A filter using the `ora:contains` XPath function.

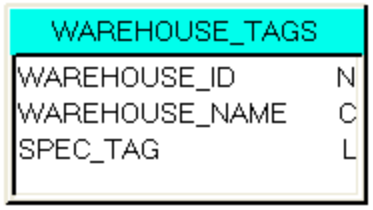
```
WHERE EXISTSNODE(WAREHOUSES.WAREHOUSE_SPEC,
'/Warehouse[ora:contains(DockType/text(), "Side")>0]',
'xmlns:ora="http://xmlns.oracle.com/xdb') = 1
```



Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	V Clearance
2	San Francisco	Rented	50000	1	Side load	Lot	Y	N	12 ft
4	Seattle, Washington	Owned	103000	3	Side load	Lot	N	Y	15 ft

XMLSEQUENCE() function
Shredding an XML node into rows

Unlike the extractvalue() objects presented earlier, with the derived table below, we don't need to know in advance the tags that constitute the XML node. The solution is more dynamic: a new tag is automatically taken into account in the report without having to change the Universe.




```
SELECT w.WAREHOUSE_ID, w.WAREHOUSE_NAME,
       XMLSERIALIZE(CONTENT VALUE(p) as CLOB) as SPEC_TAG
FROM WAREHOUSES w,
TABLE(XMLSEQUENCE(EXTRACT(warehouse_spec, '/Warehouse/*'))) p
```

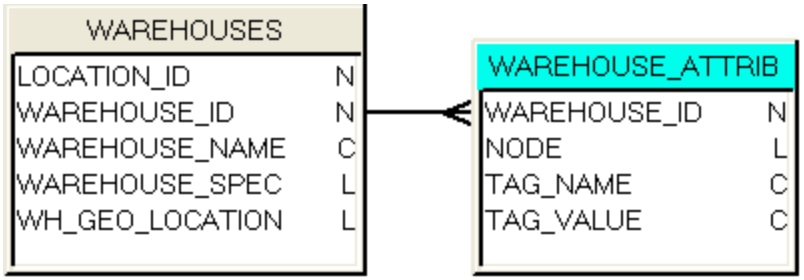
XMLSEQUENCE() function
Shredding an XML node into rows

The warehouses specification sub-nodes presented in rows.

Warehouse Id	Warehouse Name	Warehouse Spec
1	Southlake, Texas	<Area>25000</Area>
		<Building>Owned</Building>
		<Docks>2</Docks>
		<DockType>Rear load</DockType>
		<Parking>Street</Parking>
		<RailAccess>N</RailAccess>
		<VClearance>10 ft</VClearance>
		<WaterAccess>Y</WaterAccess>
Warehouse Id	**Warehouse Name**	**Warehouse Spec**
2	San Francisco	<Area>50000</Area>
		<Building>Rented</Building>
		<Docks>1</Docks>
		<DockType>Side load</DockType>
		<Parking>Lot</Parking>
		<RailAccess>N</RailAccess>
		<VClearance>12 ft</VClearance>
		<WaterAccess>Y</WaterAccess>

XMLSEQUENCE() function
Shredding an XML node into rows

We separate the metadata (tag name) from the data (tag value) in order to make the reporting more flexible.




```
SELECT w.WAREHOUSE_ID,
       XMLSERIALIZE(CONTENT VALUE(p) as CLOB) as NODE,
       value(p).getrootelement()              as TAG_NAME,
       extractvalue(VALUE(p), '/*')           as TAG_VALUE
  FROM WAREHOUSES w,
       TABLE(XMLSEQUENCE(EXTRACT(warehouse_spec, '/Warehouse/*'))) p
```

XMLSEQUENCE() function
Shredding an XML node into rows

The “Tag name” is now an object of its own.

Tag name: Area

Warehouse Id	Warehouse Name	Tag Value	Node
1	Southlake, Texas	25000	<Area>25000</Area>
2	San Francisco	50000	<Area>50000</Area>
3	New Jersey	85700	<Area>85700</Area>
4	Seattle, Washington	103000	<Area>103000</Area>

Tag name: Building

Warehouse Id	Warehouse Name	Tag Value	Node
1	Southlake, Texas	Owned	<Building>Owned</Building>
2	San Francisco	Rented	<Building>Rented</Building>
3	New Jersey	Rented	<Building>Rented</Building>
4	Seattle, Washington	Owned	<Building>Owned</Building>

Tag name: Docks

Warehouse Id	Warehouse Name	Tag Value	Node
1	Southlake, Texas	2	<Docks>2</Docks>
2	San Francisco	1	<Docks>1</Docks>
4	Seattle, Washington	3	<Docks>3</Docks>

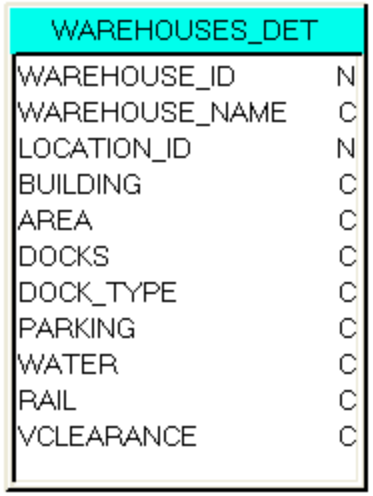
Tag Name	Number of Warehouses
Area	4
Building	4
Docks	3
DockType	4
Parking	4
RailAccess	4
VClearance	4
WaterAccess	4

XMLTABLE() function
Turning an XML node into a table

We build a table out of the warehouse_spec xml column.

The derived table is the following.

```
SELECT WAREHOUSE_ID, WAREHOUSE_NAME, LOCATION_ID, W2.BUILDING, W2.AREA,
       W2.DOCKS, W2.DOCK_TYPE, W2.PARKING, W2.WATER, W2.RAIL, W2.VCLEARANCE
   FROM WAREHOUSES,
   XMLTABLE('/Warehouse'  PASSING warehouses.warehouse_spec
      COLUMNS
         BUILDING   varchar2(35) PATH 'Building',
         AREA       varchar2(6)  PATH 'Area',
         DOCKS      varchar2(6)  PATH 'Docks',
         DOCK_TYPE  varchar2(35) PATH 'DockType',
         WATER      varchar2(1)  PATH 'WaterAccess',
         PARKING    varchar2(35) PATH 'Parking',
         VCLEARANCE varchar2(10) PATH 'VClearance',
         RAIL       varchar2(1)  PATH 'RailAccess'
     ) w2
```



XMLTABLE() function
Turning an XML node into a table

Notice that the warehouses with empty warehouse_spec are not returned.

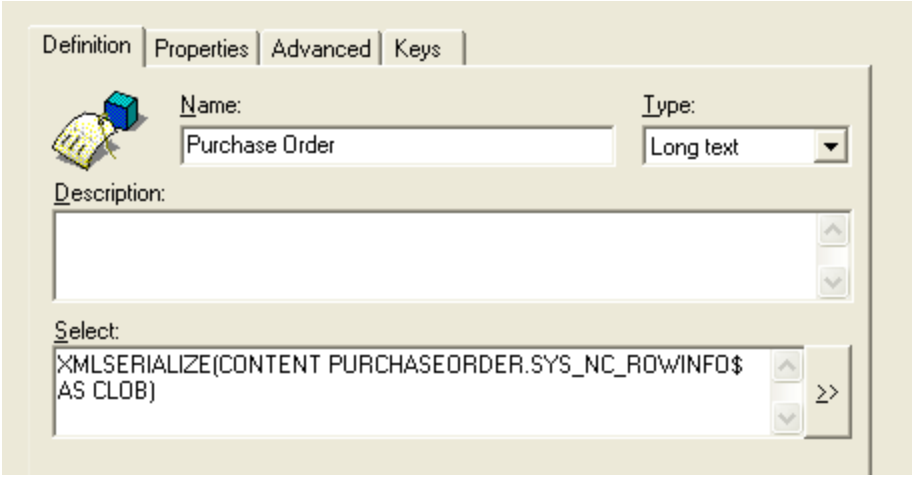
Warehouse Id	Warehouse Name	Building	Area	Docks	Dock Type	Parking	Water	Rail	Vclearance
1	Southlake, Texas	Owned	25000	2	Rear load	Street	Y	N	10 ft
2	San Francisco	Rented	50000	1	Side load	Lot	Y	N	12 ft
3	New Jersey	Rented	85700			Street	N	N	11.5 ft
4	Seattle, Washington	Owned	103000	3	Side load	Lot	N	Y	15 ft

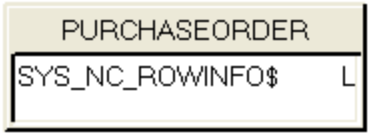
```
SQL> SELECT WAREHOUSE_ID, WAREHOUSE_NAME FROM WAREHOUSES WHERE WAREHOUSE_SPEC is null;

WAREHOUSE_ID WAREHOUSE_NAME
------------ ----------------------------------
           5 Toronto
           6 Sydney
           7 Mexico City
           8 Beijing
           9 Bombay
```

Querying an XML table

The purchaseorder demo table is an XML-Schema based table. We convert the XML system column into long text.

```
SQL> desc purchaseorder
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
TABLE of SYS.XMLTYPE(XMLSchema "http://localhost:8080/source/schemas/poSource/xsd/purchaseOrder.xsd"
```





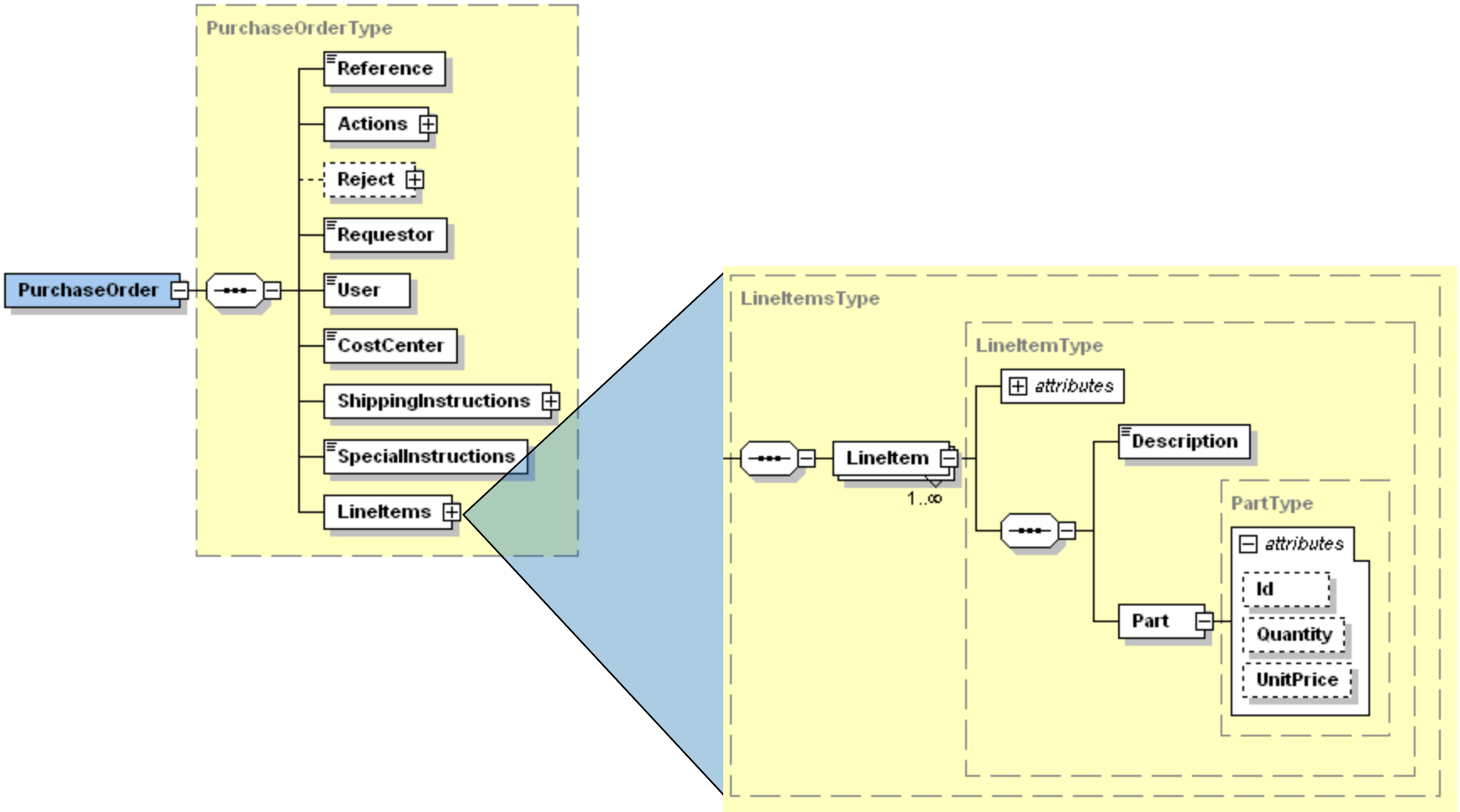
Querying an XML table

The report returns one “big” row for all orders and line items.

Purchase Orders

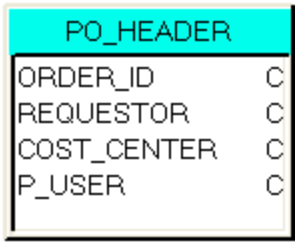
```
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocati ...
  <Reference>AMCEWEN-20021009123335370PDT</Reference>
 <Actions>
   <Action>
     <User>KPARTNER</User>
   </Action>
 </Actions>
 <Reject/>
 <Requestor>Allan D. McEwen</Requestor>
  <User>AMCEWEN</User>
 <CostCenter>S30</CostCenter>
 <ShippingInstructions>
   <name>Allan D. McEwen</name>
   <address>Oracle Plaza
Twin Dolphin Drive
Redwood Shores
CA
94065
USA</address>
    <telephone>650 506 7700</telephone>
 </ShippingInstructions>
 <SpecialInstructions>Ground</SpecialInstructions>
 <LineItems>
   <LineItem ItemNumber="1">
    <Description>Juliet of the Spirits</Description>
    <Part Id="37429165829" UnitPrice="29.95" Quantity="1"/>
   </LineItem>
   <LineItem ItemNumber="2">
    <Description>I Know Where I&amp;apos;m Going</Description>
    <Part Id="37429154427" UnitPrice="39.95" Quantity="3"/>
```


XML Schema Definition: purchaseorder.xsd



Remodeling the purchaseorder XML table

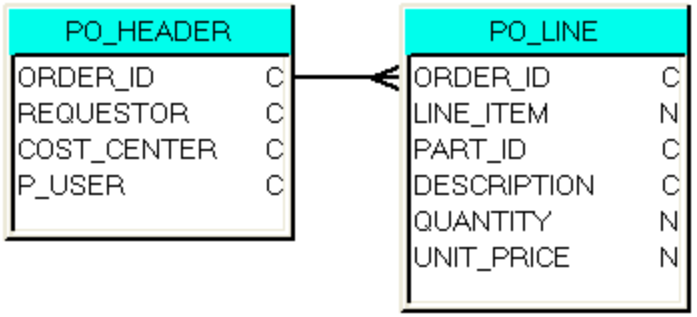
We extract data to build the po_header derived table.




```
SELECT
extractValue(p2.OBJECT_VALUE, '/PurchaseOrder/Reference')  as ORDER_ID,
extractValue(p2.OBJECT_VALUE, '/PurchaseOrder/Requestor')  as REQUESTOR,
extractValue(p2.OBJECT_VALUE, '/PurchaseOrder/CostCenter') as COST_CENTER,
extractValue(p2.OBJECT_VALUE, '/PurchaseOrder/User')       as P_USER
FROM purchaseorder p2
```

Remodeling the purchaseorder XML table

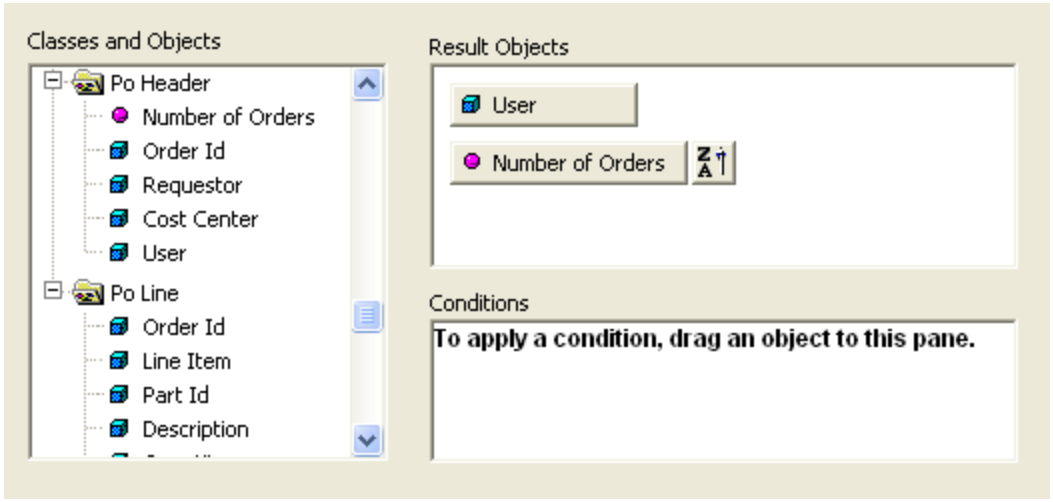
We extract line items elements to build the po_line table.




```
SELECT
extractValue(p.OBJECT_VALUE, '/PurchaseOrder/Reference')  as ORDER_ID,
extractValue(value(li),      '/LineItem/@ItemNumber')     as LINE_ITEM,
extractValue(value(li),      '/LineItem/Part/@Id')        as PART_ID,
extractValue(value(li),      '/LineItem/Description')     as DESCRIPTION,
extractValue(value(li),      '/LineItem/Part/@Quantity')  as QUANTITY,
extractValue(value(li),      '/LineItem/Part/@UnitPrice') as UNIT_PRICE
FROM purchaseorder p,
     table(XMLSequence(extract(p.OBJECT_VALUE,
                       '/PurchaseOrder/LineItems/LineItem'))) li
```

Remodeling the purchaseorder XML table

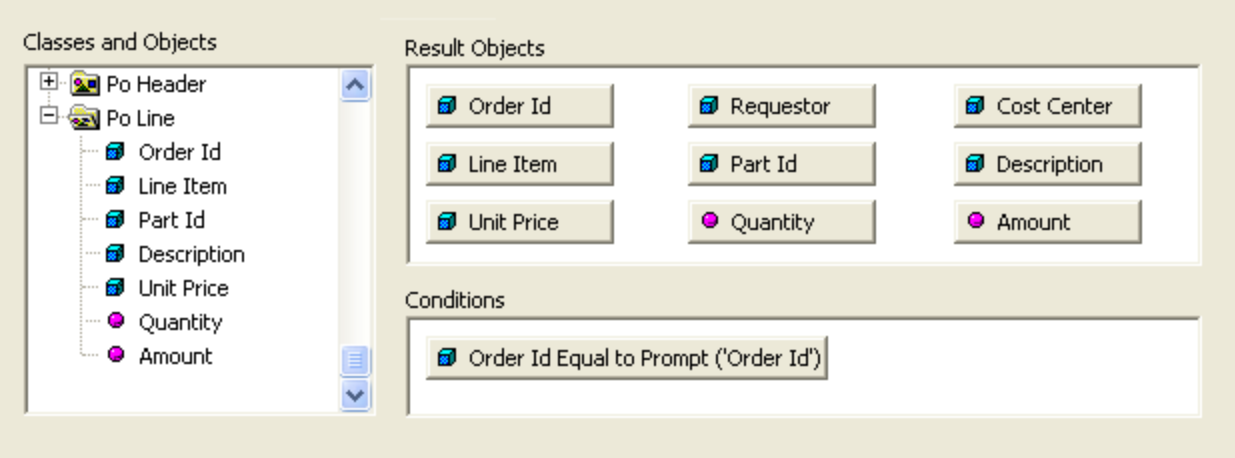
A query against the header data.



User	Number of Orders
SBELL	13
SKING	13
TFOX	13
VJONES	11
AWALSH	10
SMCCAIN	10
AMCEWEN	9
CJOHNSON	9
DAUSTIN	9
JCHEN	9
LSMITH	8
PTUCKER	7
WSMITH	7
EABEL	4
Total	132

Remodeling the purchaseorder XML table

A query against the header and line items data.



Order Id: SBELL-20021009123336O1PDT **Requestor:** Sarah J. Bell **Cost Center:** S30

Line Item	Part Id	Description	Quantity	Unit Price	Amount
1	715515009058	A Night to Remember	2	$39.95	$79.90
2	37429140222	The Unbearable Lightness Of Being	2	$29.95	$59.90
3	715515011020	Sisters	4	$29.95	$119.80
				Total:	**$259.60**

An Oracle PMML data mining model

Output from dbms_data_mining.get_model_details_xml()

```
- <PMML version="2.1">
    <Header copyright="Copyright (c) 2004, Oracle Corporation. All rights reserved." />
  - <DataDictionary numberOfFields="9">
      <DataField name="AFFINITY_CARD" optype="categorical" />
      <DataField name="AGE" optype="continuous" />
      <DataField name="BOOKKEEPING_APPLICATION" optype="continuous" />
      <DataField name="CUST_MARITAL_STATUS" optype="categorical" />
      <DataField name="EDUCATION" optype="categorical" />
      <DataField name="HOUSEHOLD_SIZE" optype="categorical" />
      <DataField name="OCCUPATION" optype="categorical" />
      <DataField name="YRS_RESIDENCE" optype="continuous" />
      <DataField name="Y_BOX_GAMES" optype="continuous" />
    </DataDictionary>
  - <TreeModel modelName="DT_SH_CLAS_SAMPLE" functionName="classification"
    + <Extension name="buildSettings">
    + <MiningSchema>
    - <Node id="0" score="0" recordCount="1500">
        <True />
        <ScoreDistribution value="0" recordCount="1120" />
        <ScoreDistribution value="1" recordCount="380" />
      + <Node id="1" score="0" recordCount="712">
      + <Node id="2" score="0" recordCount="788">
      </Node>
    </TreeModel>
  </PMML>
```

The attributes of the decision tree model

The hierarchy of nodes

Turning the PMML tree into relational

The definition of the derived table « DT_DYNL ».

```
SELECT
Z.NODE, Z.PARENT, Z.LEVELN, Z.NSCORE, Z.NPREDICATE, Z.NSURROGATE, Z.RECORDS,
SYS_CONNECT_BY_PATH(NODE, '/') as PATH
FROM
(SELECT
   lpad(to_char(X.node_id),2,0) as NODE, lpad(to_char(X.parent_node_id),2,0) as PARENT,
   X.score as NSCORE, X.record_count as RECORDS, X.node_level as LEVELN,
   to_char(dbms_xmlgen.convert(X.predicate.getclobval(), 1)) as NPREDICATE,
   to_char(dbms_xmlgen.convert(X.surrogate.getclobval(), 1)) AS NSURROGATE
 FROM
 XMLTABLE('
 for $n in /PMML/TreeModel//Node
 let $pn := /PMML/TreeModel//Node[Node/@id=$n/@id]
    return
   <Tag id="{$n/@id}" parent_id="{$pn/@id}" score="{$n/@score}"
record_count="{$n/@recordCount}" depth="{count($n/ancestor::*)-2}">
     <Predicate>
       {
         if ($n/SimplePredicate instance of element(SimplePredicate)) then
           concat(
             string($n/SimplePredicate/@field),string(' '),
             if ($n/SimplePredicate/@operator = "greaterThan") then

...  removed part  ...

   </Tag>
   ' PASSING dbms_data_mining.get_model_details_xml(@Prompt('Mining
Model','C','Mining Models\Model Name',mono,free))
   COLUMNS
     "NODE_ID"         NUMBER   PATH '/Tag/@id',
     "PARENT_NODE_ID"  NUMBER   PATH '/Tag/@parent_id',
     "SCORE"           NUMBER   PATH '/Tag/@score',
     "RECORD_COUNT"    NUMBER   PATH '/Tag/@record_count',
     "NODE_LEVEL"      NUMBER   PATH '/Tag/@depth',
     "PREDICATE"       XMLType PATH '/Tag/Predicate/text()',
     "SURROGATE"       XMLType PATH '/Tag/Surrogate/text()'
  ) X
) Z
WHERE LEVEL > 1 AND Z.LEVELN = LEVEL-1
CONNECT BY PRIOR NODE = PARENT
```

XQuery flwor

The XMLTable() function

Calling the function get_model_details_xml()

The Oracle hierarchical operator « connect by »

DT_DYNL	
NODE	C
PARENT	C
LEVELN	N
NSCORE	N
NPREDICATE	C
NSURROGATE	C
RECORDS	N
PATH	C

The PMML model rendered in DeskI

The decision tree visualized in BOBJ documents.

Decision Tree Model: DT_SH_CLAS_SAMPLE

Hierarchical Structure	Score	Negative answers	Positive answers	Percent Negative	Percent Positive	Predicate
Node00	No	1120	380	74.7 %	25.3 %	
L.. Node01	No	382	330	53.7 %	46.3 %	CUST_MARITAL_STATUS isIn "Married"
L.. Node09	Yes	67	179	27.2 %	72.8 %	EDUCATION isIn "Assoc-A" "Bach." "Masters" "PhD" "Prof
L.. Node10	No	315	151	67.6 %	32.4 %	EDUCATION isIn "10th" "11th" "12th" "1st-4th" "5th-6th" "7th
L.. Node02	No	738	50	93.7 %	6.3 %	CUST_MARITAL_STATUS isIn "Divorc." "Mabsent" "Mar
L.. Node03	No	143	31	82.2 %	17.8 %	EDUCATION isIn "Assoc-A" "Bach." "Masters" "PhD" "Prof
L.. Node04	No	66	4	94.3 %	5.7 %	YRS_RESIDENCE <= 3.5
L.. Node05	No	41	4	91.1 %	8.9 %	AGE > 25.5
L.. Node12	No	22	4	84.6 %	15.4 %	AGE <= 28.5
L.. Node13	No	19	0	100.0 %	0.0 %	AGE > 28.5
L.. Node11	No	25	0	100.0 %	0.0 %	AGE <= 25.5
L.. Node14	No	77	27	74.0 %	26.0 %	YRS_RESIDENCE > 3.5
L.. Node06	No	595	19	96.9 %	3.1 %	EDUCATION isIn "10th" "11th" "12th" "1st-4th" "5th-6th" "7th
L.. Node07	No	475	7	98.5 %	1.5 %	YRS_RESIDENCE <= 4.5
L.. Node08	No	244	7	97.2 %	2.8 %	AGE > 26.5
L.. Node16	No	44	0	100.0 %	0.0 %	BOOKKEEPING_APPLICATION <= .5
L.. Node17	No	200	7	96.6 %	3.4 %	BOOKKEEPING_APPLICATION > .5
L.. Node15	No	231	0	100.0 %	0.0 %	AGE <= 26.5
L.. Node18	No	120	12	90.9 %	9.1 %	YRS_RESIDENCE > 4.5

Decision Tree Model: DT_SH_CLAS_SAMPLE

Hierarchical Structure	Score	Records	% Negative versus % Positive
Node00	No	1500	
L.. Node01	No	712	
L.. Node09	Yes	246	
L.. Node10	No	466	
L.. Node02	No	788	
L.. Node03	No	174	
L.. Node04	No	70	
L.. Node05	No	45	
L.. Node12	No	26	
L.. Node13	No	19	
L.. Node11	No	25	
L.. Node14	No	104	
L.. Node06	No	614	
L.. Node07	No	482	
L.. Node08	No	251	
L.. Node16	No	44	
L.. Node17	No	207	
L.. Node15	No	231	
L.. Node18	No	132	

% Positive % Negative

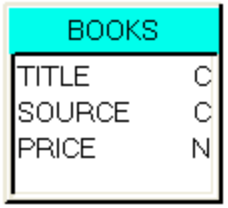
Topics

Business Objects
an SAP company

- Querying XML data inside the database
- **Querying XML data outside the database**
- Transforming XML into HTML
- Transforming relational into HTML

Query against XML file

A derived table on top of a simple xml file.

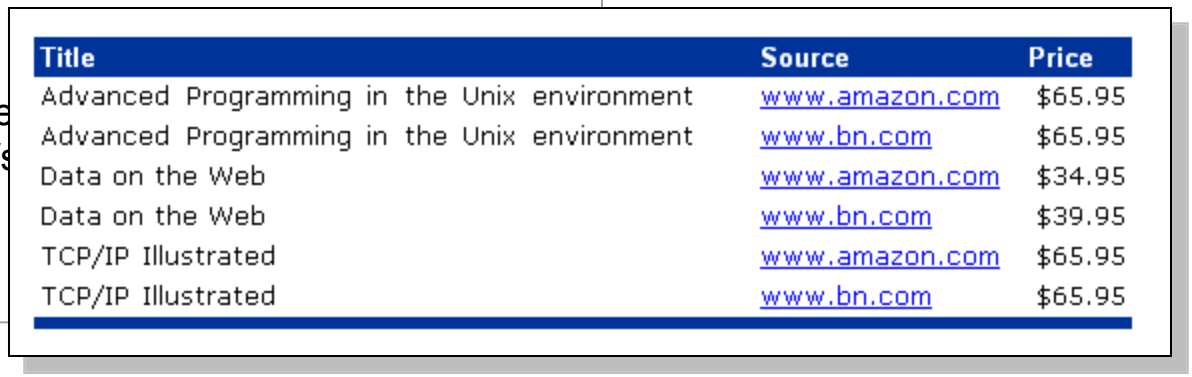



```
select TITLE, SOURCE, to_number(PRICE) as PRICE
from XMLTABLE('/prices/book' passing
HTTPURITYPE('http://localhost:1158/public/books.xml').getxml()
columns
  TITLE    varchar2(100) path '/book/title',
  SOURCE   varchar2(100) path '/book/source',
  PRICE    varchar2(40)  path '/book/price'
)
```


Query against XML file

The xml data source and the BOBJ document.

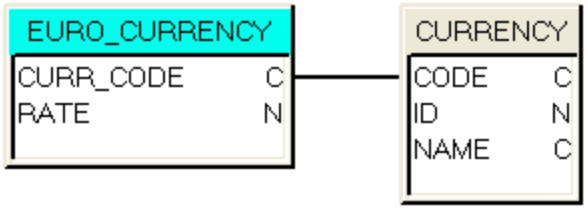
```
<?xml version="1.0"?>
<prices>
  <book>
    <title>Advanced Programming in the Unix environment</title>
    <source>www.amazon.com</source>
    <price>65.95</price>
  </book>
  <book>
    <title>Advanced Programming in the Unix environment</title>
    <source>www.bn.com</source>
    <price>65.95</price>
  </book>
 <book>
    <title> TCP/IP Illustrated </title
    <source>www.amazon.com</s
    <price>65.95</price>
  </book>
...
```



Title	Source	Price
Advanced Programming in the Unix environment	www.amazon.com	$65.95
Advanced Programming in the Unix environment	www.bn.com	$65.95
Data on the Web	www.amazon.com	$34.95
Data on the Web	www.bn.com	$39.95
TCP/IP Illustrated	www.amazon.com	$65.95
TCP/IP Illustrated	www.bn.com	$65.95

Query against RSS feed

We have defined a derived table on top of a RSS feed. We then joined it to a relational table that contains the labels.




```
select substr(TXT,9,3) as CURR_CODE,
to_number(replace(substr(TXT,14,8),',',')) as RATE
from XMLTABLE('/rss/channel/item' passing
HTTPURITYPE('http://currencysource.com/RSS/EUR.xml').getxml()
columns
  TXT  varchar2(100)  path '/item/title'
)
```

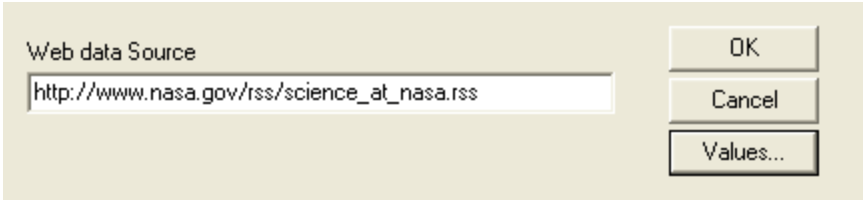
Query against RSS feed

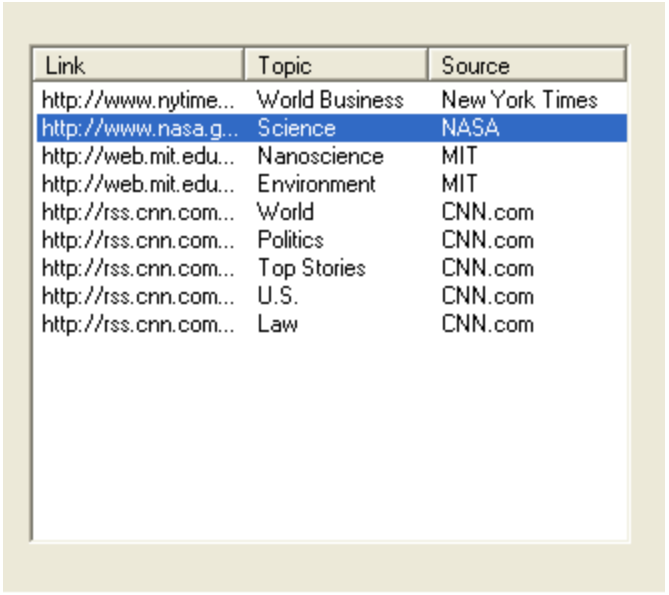
A heterogeneous query joining together a relational table and RSS feed data.

Name	Currency Code	Exchange Rate
Argentine Peso	ARS	4.055063
Australian Dollar	AUD	1.586423
Bahrain Dinar	BHD	0.505268
Botswana Pula	BWP	8.386356
Brazilian Real	BRL	2.604100
Brunei Dollar	BND	2.067317
Canadian Dollar	CAD	1.437496
Chilean Peso	CLP	708.973400
Chinese Yuan	CNY	10.241670
Colombian Peso	COP	2,635.310000
Cyprus Pound	CYP	0.583304
Czech Koruna	CZK	28.661120
Danish Krone	DKK	7.432293
Euro	EUR	1.000000
Hungarian Forint	HUF	247.545100
Icelandic Krona	ISK	84.679480
Indian Rupee	INR	55.050430

Query against RSS feed

A second example: we prompt RSS feeds to the end-user.





Link	Topic	Source
http://www.nytime...	World Business	New York Times
http://www.nasa.g...	Science	NASA
http://web.mit.edu...	Nanoscience	MIT
http://web.mit.edu...	Environment	MIT
http://rss.cnn.com...	World	CNN.com
http://rss.cnn.com...	Politics	CNN.com
http://rss.cnn.com...	Top Stories	CNN.com
http://rss.cnn.com...	U.S.	CNN.com
http://rss.cnn.com...	Law	CNN.com

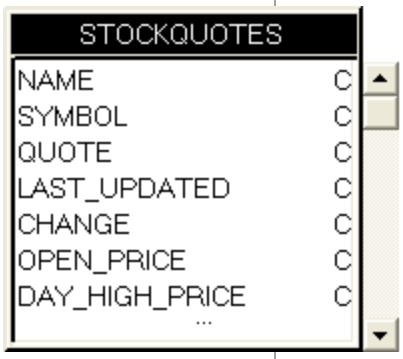
NASA: Science

Title	Full story
A Meteoroid Hits the Moon	http://www.nasa.gov ...
Electric Ice	http://www.nasa.gov ...
Huge Storms Converge	http://www.nasa.gov ...
Lunar X Games	http://www.nasa.gov ...
Mercury Transits Sun	http://www.nasa.gov ...
Moonquakes	http://www.nasa.gov ...
SMART-1 to Crash the Moon	http://www.nasa.gov ...
Strange Moonlight	http://www.nasa.gov ...
The Sky is Falling	http://www.nasa.gov ...
Wide Awake on the Sea of Tranquillity	http://www.nasa.gov ...

Query against Web data

Getting Stock Quotes from the Web.

```
select NAME, SYMBOL, QUOTE, LAST_UPDATED, CHANGE,
       OPEN_PRICE, DAY_HIGH_PRICE, DAY_LOW_PRICE,
       VOLUME, MARKET_CAP, YEAR_RANGE, EX_DIVIDEND_DATE,
       DIVIDEND_YIELD, DIVIDEND_PER_SHARE, PERCENT_CHANGE
from XMLTABLE(XMLNAMESPACES(default 'http://swanandmokashi.com'),
'/ArrayOfQuote/Quote' passing HTTPURITYPE(
'http://www.swanandmokashi.com/HomePage/WebServices/StockQuotes.asmx/
GetQuotes?QuoteTicker=AAPL,GOOG,MSFT,NOK').getxml()
columns
  NAME           varchar2(100) path '/Quote/CompanyName',
  SYMBOL         varchar2(40)  path '/Quote/StockTicker',
  QUOTE          varchar2(40)  path '/Quote/StockQuote',
  LAST_UPDATED   varchar2(80)  path '/Quote/LastUpdated',
  CHANGE         varchar2(40)  path '/Quote/Change',
  OPEN_PRICE     varchar2(40)  path '/Quote/OpenPrice',
  DAY_HIGH_PRICE varchar2(40)  path '/Quote/DayHighPrice',
  …
```



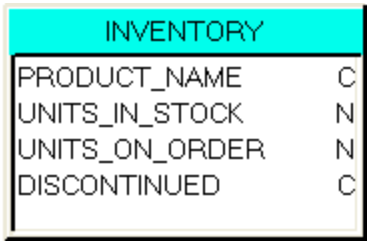
Query against Web data

The BusinessObjects document.

	APPLE INC	GOOGLE	MICROSOFT CP	NOKIA CP ADS
Symbol	AAPL	GOOG	MSFT	NOK
Quote	121.14	529.17	29.61	28.47
Last Updated	7/2/2007 11:48am	7/2/2007 11:48am	7/2/2007 11:48am	7/2/2007 11:43am
Change	-0.90	+6.4699	+0.14	+0.36
Percent Change	-0.74%	+1.24%	+0.48%	+1.28%
Open Price	121.10	525.49	29.71	28.20
Day Range	119.30 - 122.09	524.20 - 531.85	29.56 - 29.80	28.18 - 28.60
Volume	21079562	1941773	19277260	3051900
Market Cap	104.8B	164.9B	283.3B	111.6B
Year Range	50.16 - 127.61	363.36 - 534.99	22.23 - 31.48	18.23 - 29.19
Ex Dividend Date	21-Nov-95	N/A	May 15	May 4
Dividend Yield	N/A	N/A	1.32	2.01
Dividend Per Share	0.00	0.00	0.39	0.565

Query against Web data

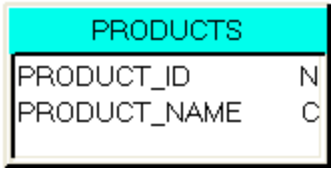
A second example: Checking Inventory.




```
select
extractvalue(TXT,'/ns1:DataSet/ns2:diffgram/NewDataSet/ProductQuantity/ProductName',
'xmlns:ns1="http://www.ACMEDistribution.com/Products" xmlns:ns2="urn:schemas-microsoft-com:xml-
diffgram-v1"') as PRODUCT_NAME,
to_number(extractvalue(TXT,'/ns1:DataSet/ns2:diffgram/NewDataSet/ProductQuantity/UnitsInStock',
'xmlns:ns1="http://www.ACMEDistribution.com/Products" xmlns:ns2="urn:schemas-microsoft-com:xml-
diffgram-v1"')) as UNITS_IN_STOCK,
to_number(extractvalue(TXT,'/ns1:DataSet/ns2:diffgram/NewDataSet/ProductQuantity/UnitsOnOrder',
'xmlns:ns1="http://www.ACMEDistribution.com/Products" xmlns:ns2="urn:schemas-microsoft-com:xml-
diffgram-v1"')) as UNITS_ON_ORDER,
extractvalue(TXT,'/ns1:DataSet/ns2:diffgram/NewDataSet/ProductQuantity/Discontinued',
'xmlns:ns1="http://www.ACMEDistribution.com/Products" xmlns:ns2="urn:schemas-microsoft-com:xml-
diffgram-v1"') as DISCONTINUED
from XMLTABLE('/' passing HTTPURITYPE(
'http://www.xmlforasp.net/CodeBank/webservices/ACMECorp/ACMECorp.asmx/CheckStock?productID=@Pro
mpt('Product ID','N','Products\Product Id',mono,free)').getxml() columns
  TXT    XMLType path '/' )
```

Query against Web data

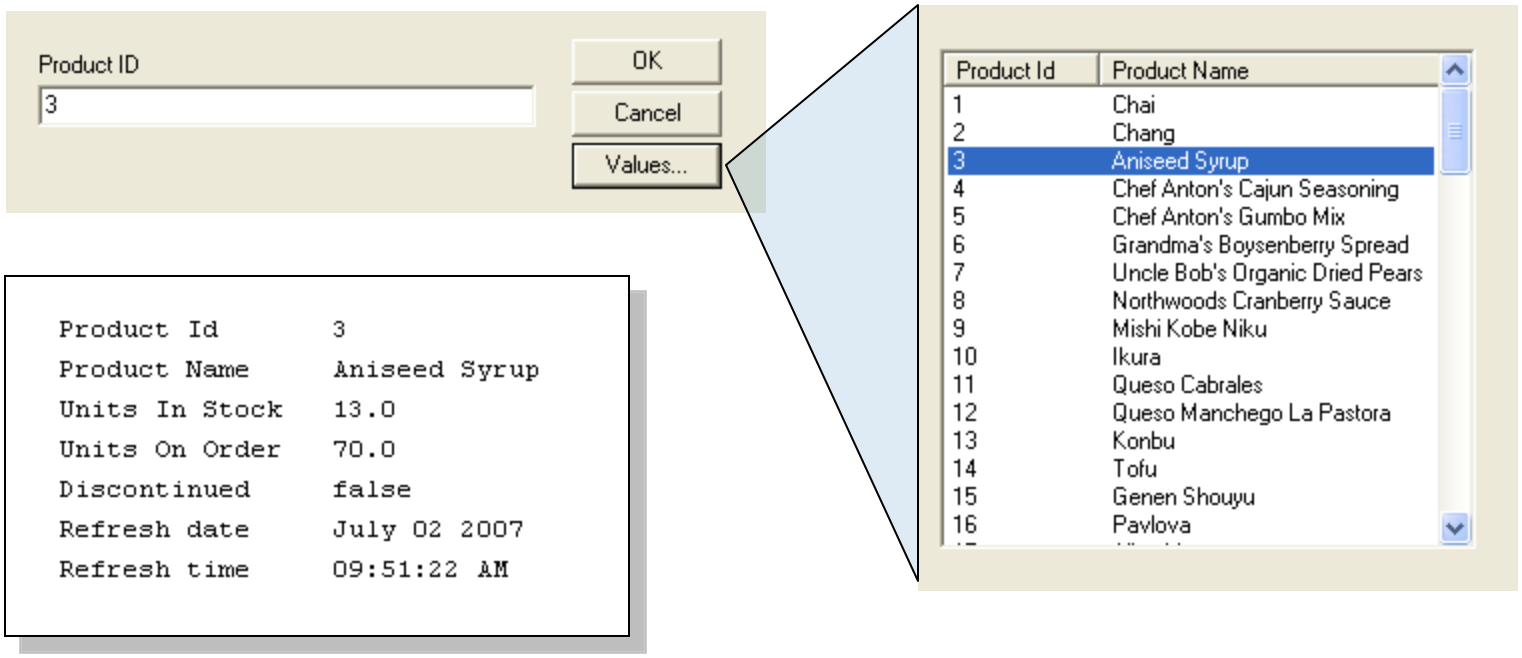
The products table enables us to build a list of values used in the report illustrated next.




```
select
to_number(extractvalue(value(al2),'/Products/ProductID')) as PRODUCT_ID,
extractvalue(value(al2),'/Products/ProductName') as PRODUCT_NAME
from XMLTABLE('/' passing
HTTPURITYPE('http://www.xmlforasp.net/CodeBank/webservices/ACMECorp/ACMECorp
.asmx/GetProducts').getxml() columns
  TXT    XMLType path '/'
) al1, table(xmlsequence(extract(al1.txt,
'/ns1:DataSet/ns2:diffgram/NewDataSet/Products',
'xmlns:ns1="http://www.ACMEDistribution.com/Products"
xmlns:ns2="urn:schemas-microsoft-com:xml-diffgram-v1"'))) al2
```

Query against Web data

The BusinessObjects document.



Topics

Business Objects
an SAP company

- Querying XML data inside the database
- Querying XML data outside the database
- **Transforming XML into HTML**
- Transforming relational into HTML

The Oracle OE demo database

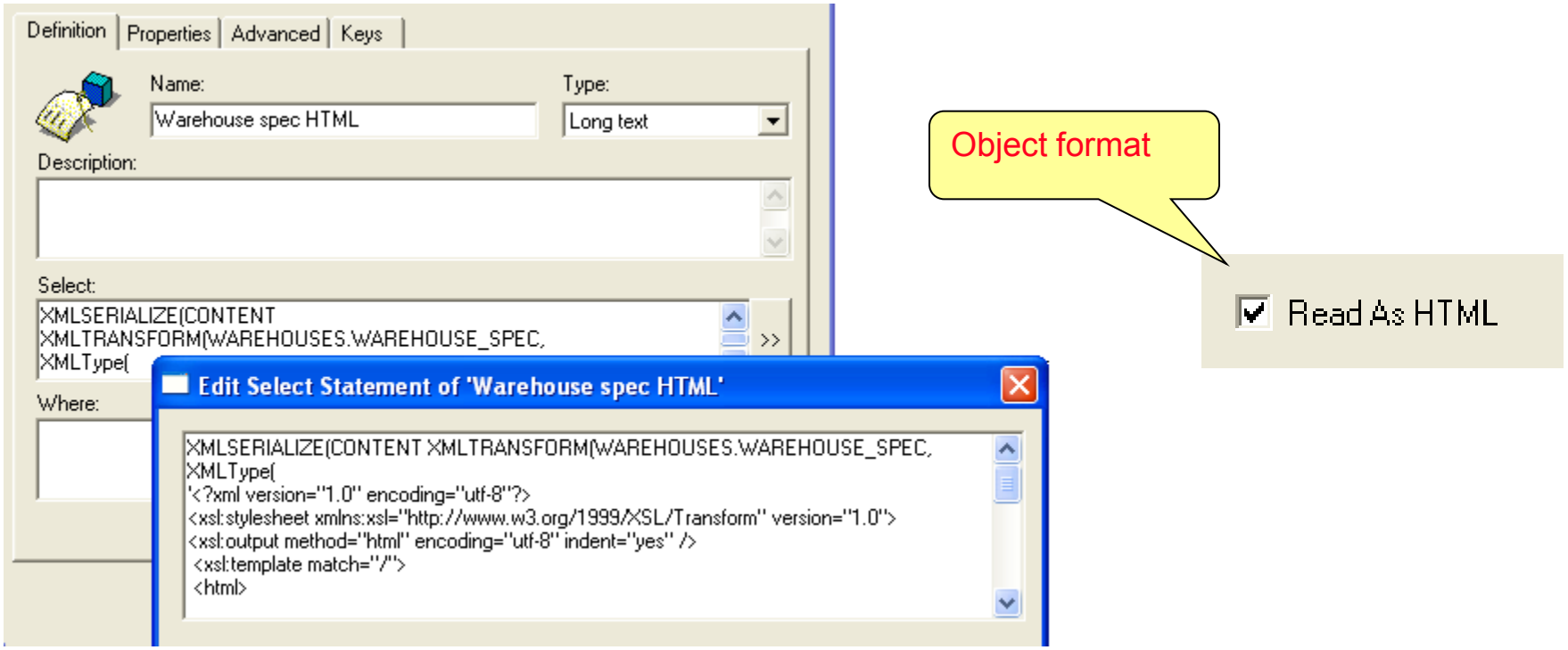
The warehouses.warehouse_spec XML column

```
SQL> select warehouse_spec from warehouses where warehouse_id=1;

WAREHOUSE_SPEC
--------------------------------------------------------------------
<?xml version="1.0"?>
<Warehouse>
<Building>Owned</Building>
<Area>25000</Area>
<Docks>2</Docks>
<DockType>Rear load</DockType>
<WaterAccess>Y</WaterAccess>
<RailAccess>N</RailAccess>
<Parking>Street</Parking>
<VClearance>10 ft</VClearance>
</Warehouse>
```


Transforming XML into HTML against the Oracle demo database

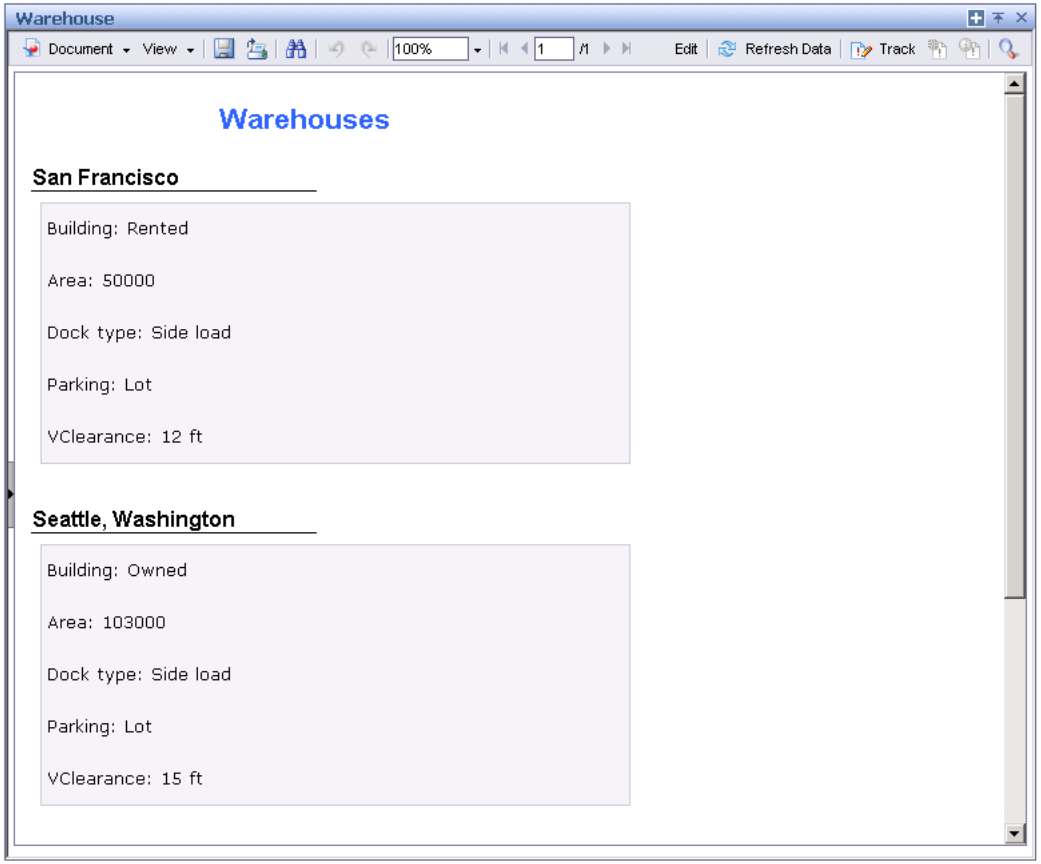
We define an object on the warehouse_spec XML column

We include the xsl stylesheet as part of the object definition



Transforming XML into HTML against the Oracle demo database

The warehouse spec rendered as HTML in WebI



Topics

Business Objects
an SAP company

- Querying XML data inside the database
- Querying XML data outside the database
- Transforming XML into HTML
- **Transforming relational into HTML**

Generating an XML document from the Oracle demo sh user account

```
create view xprod as
with
CATEGORY as (
SELECT distinct PROD_CATEGORY_ID as CAT_ID, PROD_CATEGORY as CAT_NAME, PROD_CATEGORY_DESC as CAT_DESC
from PRODUCTS ),
LTSALES as (
SELECT p.PROD_CATEGORY_ID as CAT_ID, p.PROD_ID, p.PROD_NAME,
       to_char(round(p.PROD_LIST_PRICE), 'FML9,999,999,999') as PRICE,
       to_char(round(sum(s.AMOUNT_SOLD)),'FML9,999,999,999') as LIFETIME_REVENUE,
       RANK() OVER (ORDER BY sum(s.AMOUNT_SOLD) DESC) as PROD_RANK
FROM
  PRODUCTS p, SALES s
WHERE
  p.PROD_ID=s.PROD_ID
GROUP BY
  p.PROD_CATEGORY_ID, p.PROD_ID, p.PROD_NAME, p.PROD_LIST_PRICE
)
SELECT 1 as ID, XMLRoot(  XMLElement( "sales",
XMLAgg(
  XMLElement( "category",
    XMLForest(c.CAT_ID as "id", c.CAT_NAME as "name", c.CAT_DESC as "description"),
    XMLElement( "product_list",
      (SELECT XMLAgg(
                XMLElement( "product",
                  XMLForest(l.PROD_ID as "id", l.PROD_NAME as "name", l.PRICE as "price",
                  l.LIFETIME_REVENUE as "lifetime_revenue", l.PROD_RANK as "prod_rank")))
        FROM
          LTSALES l
        WHERE l.CAT_ID = c.CAT_ID )))) )  , VERSION '1.0', STANDALONE YES) as xdata
FROM
  CATEGORY c;
```

Building a view

Aggregating and ranking sales data by product

Generating XML

Generating an XML document from the Oracle demo sh user account

The „xprod" view returns an XML document as illustrated here in SQL Plus

```
SQL> select xdata from xprod;

XDATA
--------------------------------------------------------------------------------
<?xml version="1.0" standalone="yes"?>
<sales>
  <category>
    <id>202</id>
    <name>Hardware</name>
    <description>Hardware</description>
    <product_list>
      <product>
        <id>18</id>
        <name>Envoy Ambassador</name>
        <price>$1,300</price>
        <lifetime_revenue>$15,011,643</lifetime_revenue>
        <prod_rank>1</prod_rank>
      </product>
      <product>
        <id>15</id>
        <name>Envoy 256MB - 40GB</name>
        <price>$1,000</price>
        <lifetime_revenue>$5,635,963</lifetime_revenue>
        <prod_rank>6</prod_rank>
      </product>
    </product_list>
  </category>
  <category>
    <id>201</id>
    <name>Electronics</name>
    <description>Electronics</description>
                                        …
```

XSL templates

We create a table for storing XSL templates

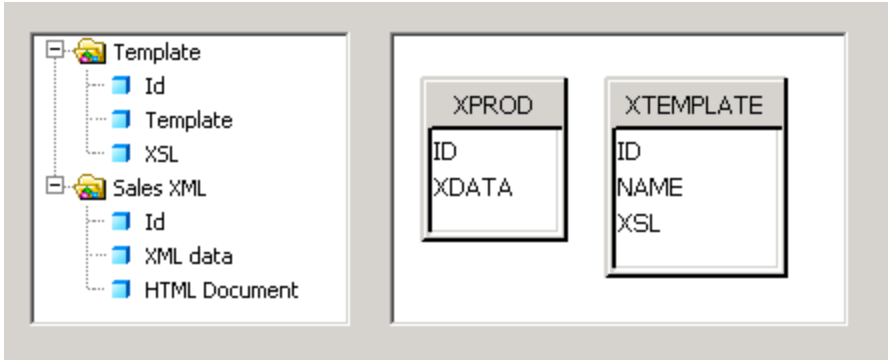
```
create table XTEMPLATE
(
 ID     number,
 NAME   varchar2(80),
 XSL    XMLTYPE
)
;
```

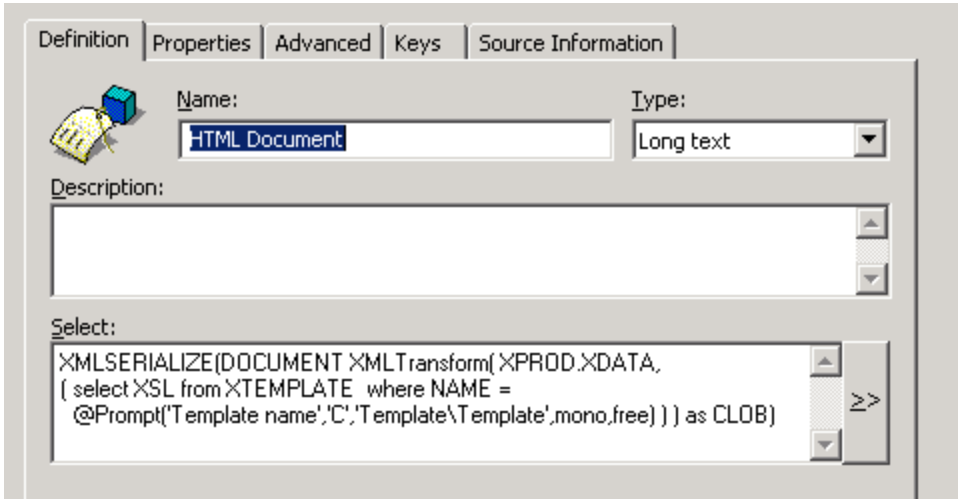
XSL templates
Filling the ‘xtemplate’ table

```
INSERT INTO XTEMPLATE VALUES ( 1, 'Alphabetical List', XMLTYPE.createxml(
'<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="utf-8" indent="yes" />
 <xsl:template match="/">
 <html>
  <style>
    .txt { font-family: Verdana, sans-serif; font-size: xx-small; vertical-align: center; text-align:left;
padding: .5em; background-color: beige;}
    .num { font-family: Verdana, sans-serif; font-size: xx-small; vertical-align: center; text-align:right;
padding: .5em; background-color: beige;}
    .hdr { font-family: Verdana, sans-serif; font-size: smaller;vertical-align: center; text-align:left;
padding: .5em; background-color: #9bd49d;}
    .ctr { font-family: Verdana, sans-serif; font-size: xx-small; vertical-align: center; text-align:center;
padding: .5em; background-color: beige;}
  </style>
   <body>
    <table>
       <tr>
         <th class="hdr">Product name</th>
         <th class="hdr">Product Id</th>
         <th class="hdr">Unit Price</th>
         <th class="hdr">Life time revenue</th>
       </tr>
         <xsl:for-each select="sales/category/product_list/product">
         <xsl:sort select="name" data-type="text" order="ascending"/>
       <tr>
       <td Class="txt"><em><xsl:value-of select="name"/></em></td>
       <td Class="ctr"><xsl:value-of select="id"/></td>
       …
```

XMLTRANSFORM()
Turning XML into HTML

A Universe against the „xprod" view and the „xtemplate" table



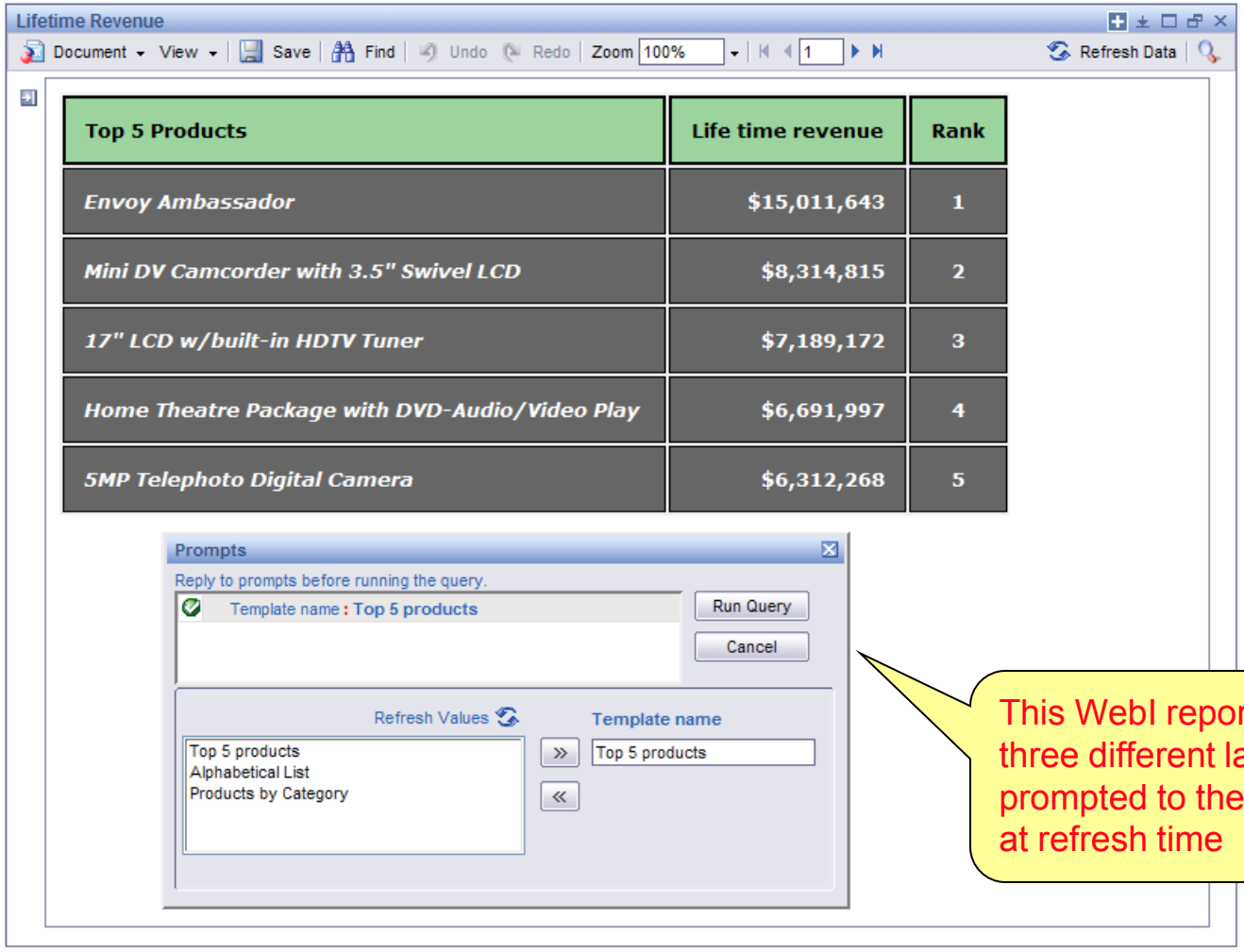


XMLTRANSFORM()
Turning XML into HTML

Viewing the HTML document in WebI

Product name	Product Id	Unit Price	Life time revenue
1.44MB External 3.5" Diskette	31	$9	$219,618
128MB Memory Card	137	$53	$571,334
17" LCD w/built-in HDTV Tuner	14	$1,000	$7,189,172
18" Flat Panel Graphics Monitor	21	$900	$5,498,728
256MB Memory Card	138	$70	$591,705
3 1/2" Bulk diskettes, Box of 100	126	$29	$370,205
3 1/2" Bulk diskettes, Box of 50	125	$16	$254,129
5MP Telephoto Digital Camera	13	$900	$6,312,268
64MB Memory Card	136	$33	$27,933
8.3 Minitower Speaker	29	$500	$3,845,387
Adventures with Numbers	146	$12	$175,564
Bounce	139	$20	$244,596
CD-R Mini Discs	113	$23	$384,554
CD-R with Jewel Cases, pACK OF 12	119	$7	$170,406
CD-R, Professional Grade, Pack of 10	117	$9	$170,270
CD-RW, High Speed Pack of 5	116	$12	$230,233
CD-RW, High Speed, Pack of 10	115	$9	$106,468
Comic Book Heroes	143	$20	$101,215
DVD-R Disc with Jewel Case, 4.7 GB	120	$7	$577,421

XMLTRANSFORM()
Turning XML into HTML



This WebI report offers three different layouts prompted to the end-user at refresh time

References

- W3C XQuery: http://www.w3.org/XML/Query/
- Oracle 10g Release 2: SQL reference
- Oracle 10g Release 2: XML DB Developer guide

© 2008 SAP AG. All rights reserved.