Information Integration Systems
Chapter 2. Federated Databases
(3: XML and JSON format)

SIA & SDBIS

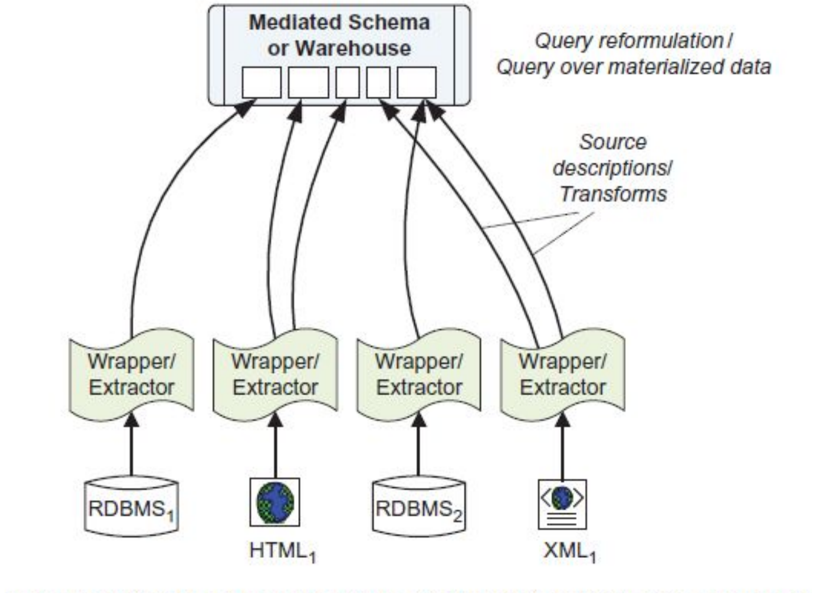
XML and JSON Document Data Source Access

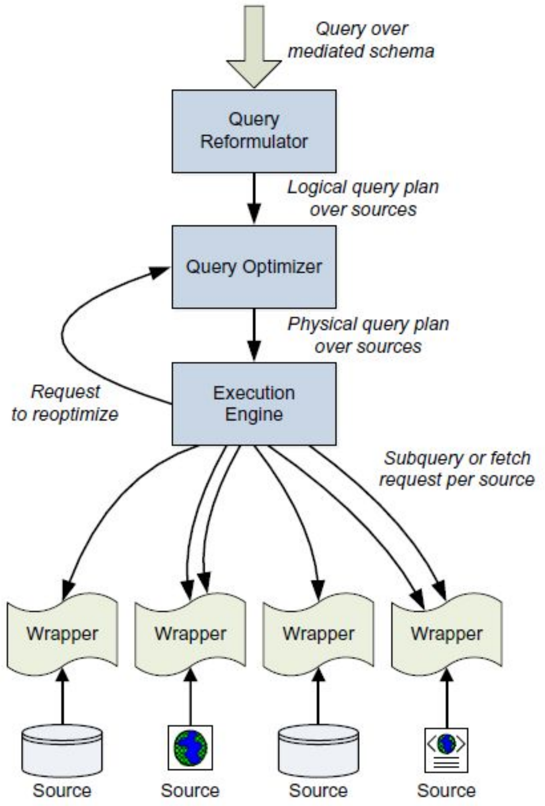
Web External Data Sources Integration

Federated Database Systems. Concepts (Review)

- Federation: “disparate data(bases) integration into a unified logical structure”.
- Federated MDBS
  - MDBS: Multidatabase System: “A distributed DBMS in which each site maintains complete autonomy”
  - F-MDBS: “... is a cross between distributed DBMS and centralized DBMS; it is a distributed system for global users and a centralized system for local users”.
- Federated Database: “The sources are independent, but one source can call on others to supply information.”

Data-based *perspective* [1, 10..14]: Virtual Database (Review)

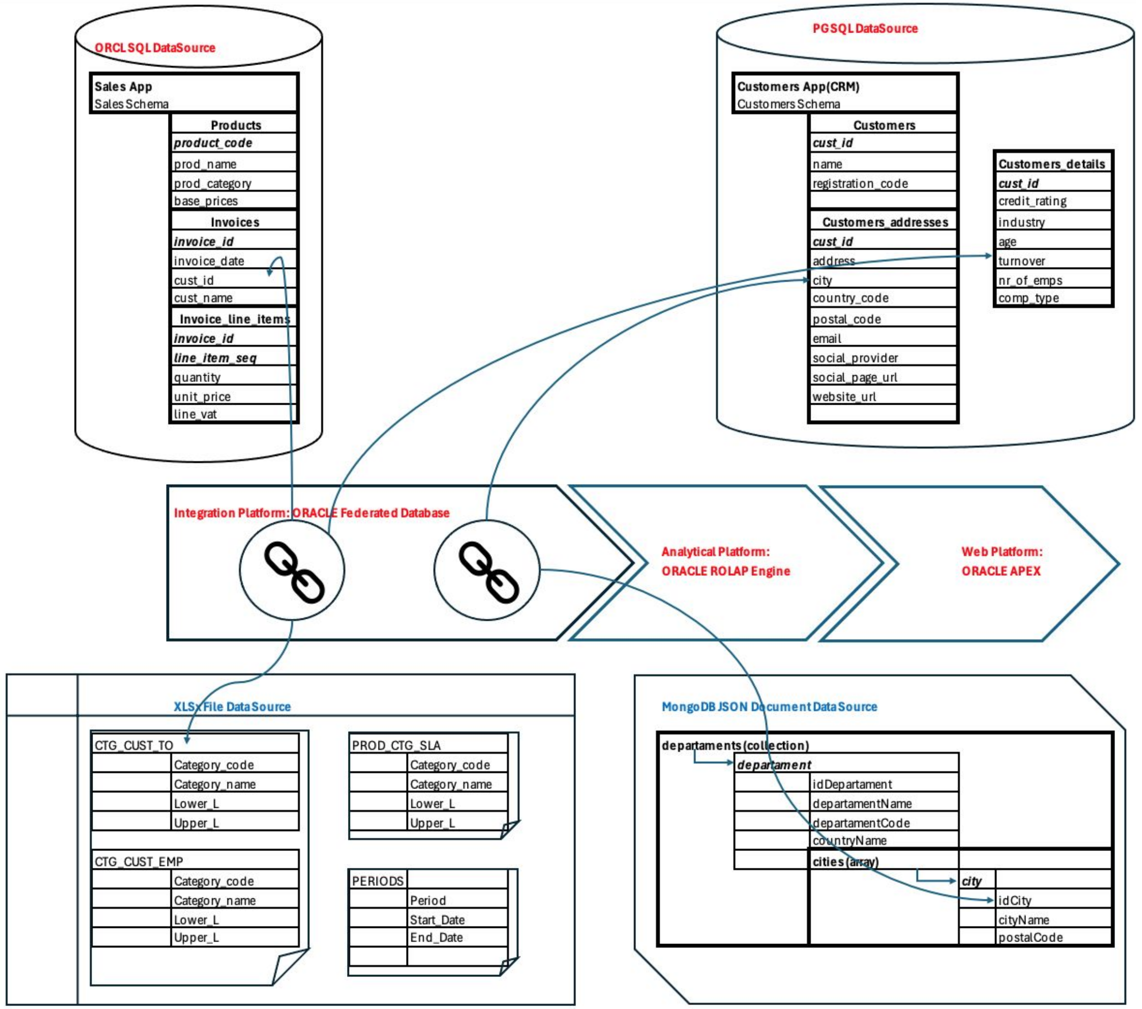

Mediated Schema or Warehouse
Query reformulation / Query over materialized data
Source descriptions/ Transforms
Wrapper/ Extractor
Wrapper/ Extractor
Wrapper/ Extractor
Wrapper/ Extractor
RDBMS1
HTML1
RDBMS2
XML1


Query over mediated schema
Query Reformulator
Logical query plan over sources
Query Optimizer
Physical query plan over sources
Request to reoptimize
Execution Engine
Subquery or fetch request per source
Wrapper
Wrapper
Wrapper
Wrapper
Source
Source
Source
Source

Case Study (Review)

- Data Sources
  - SQL: Oracle DB Database 12c/18c/19c/21c,
  - SQL: PostgreSQL 9/10/12,
  - CSV/XLSx: Local FileSystem, Virtual File System (FTP)
  - XML: Local FileSystem or (Web)REST Data Services
  - JSON: Local FileSystem or (Web)REST Data Services

- Data source Access Model
  - External Tables
  - Remote Views
    - SQL Remote Views
    - XML Remote Views
    - JSON Remote Views
    - REST Remote Views
  - Local Tables (ETL)
- Integration Model
  - Consolidation Views
  - Analytical Views
- Web Model:
  - ORDS REST Views,
  - APEX Reports and Charts


ORCL SQL DataSource
Sales App
Sales Schema
Products
product_code
prod_name
prod_category
base_prices
Invoices
invoice_id
invoice_date
cust_id
cust_name
Invoice_line_items
invoice_id
line_item_seq
quantity
unit_price
line_vat
PGSQL DataSource
Customers App(CRM)
Customers Schema
Customers
cust_id
name
registration_code
Customers_addresses
cust_id
address
city
country_code
postal_code
email
social_provider
social_page_url
website_url
Customers_details
cust_id
credit_rating
industry
age
turnover
nr_of_emps
comp_type
Integration Platform: ORACLE Federated Database
Analytical Platform:
ORACLE ROLAP Engine
Web Platform:
ORACLE APEX
XLSx File DataSource
CTG_CUST_TO
Category_code
Category_name
Lower_L
Upper_L
CTG_CUST_EMP
Category_code
Category_name
Lower_L
Upper_L
PROD_CTG_SLA
Category_code
Category_name
Lower_L
Upper_L
PERIODS
Period
Start_Date
End_Date
MongoDB JSON Document DataSource
departaments (collection)
departament
idDepartament
departamentName
departamentCode
countryName
cities (array)
city
idCity
cityName
postalCode

2.2 Architecture and components FDB

- 2.2.1 Federated Database System Concept
- 2.2.2 Data Source Model and Access Components
- 2.2.3 Integration and Analytical Model
- 2.2.4 Integration Web Model

2.2.2 Data Source Model and Access Components

- SQL Data Source Access Components
- CSV, XLS Data Source Access Components
- **XML, JSON Data Source Access Components**

Data Source Model & Format

- Integration Strategy: access to hierarchical XML and JSON local files.
    - Model|Schema Matching:
        - Hierarchical Documents (collections of documents) -> Oracle.SQL FDB SQL Table Schema
- Data Source Format:
    - XML tag-based format;
        - hierarchical-connected tags: tags with sub-tags;
        - key-values
            - as named-tag with tag-content;
                - <tag-name>textual-tag-value</tag-name>
            - as attribute:value within tag declaration
                - <tag-name attribute=value />
    - JSON key-value (attribute-name: value) documents
        - collection of document as JSON-arrays;
        - document (JSON-object) as collection of attribute:value pairs;
        - attribute of document as sub-document or JSON-array.

(1) Data Source Access Components for XML Data Source (external file access channel)

- DIRECTORY object to access external files from disk.
    - Grant necessary privileges.
- Strategy 1: REMOTE VIEW from external file
    - Using PL/SQL function BFILENAME() to point to external files.
    - Using XMLTABLE() function to load and parse XMLType document (XML hierarchical - to - SQL relational model matching).
- Strategy 2: LOCAL TABLE to load data from external file as CLOB or as XMLTYPE format
    - To load data from external files:
        - use PL/SQL function BFILENAME(),
        - (or) use EXTERNAL TABLE by ORACLE_LOADER.
    - Local views on external table could transform/consolidate/(re)map/match data with XMLTABLE function.

Oracle XML Processing Types and Functions

- XMLTYPE object type_
  - manages XML data from CLOB documents:
    - Constructor: *XMLTYPE(clob_value|varchar_value);*
    - Member functions: *getClobVal(), getStringVal().*
- **XMLTABLE()** SQL operator for model matching: from XML to SQL-Table:
  - parses XML data into rows and projects columns from rows with following parameters:
    - *path* argument: search-parsing path using XPath specific expressions;
    - *passing xmltype(blob) value: XML data* in XMLTYPE format;
    - *columns: definitions* to project parsed XML rows;
  - parses nested-XML tags also.

Oracle XML Processing Types and Functions

```
select x.idDepartament, x.departamentName, x.departamentCode, x.countryName
    from XMLTABLE(
        '/departaments/departament[departamentName="Iasi"]'
        passing xmltype(
            bfilename('EXT_FILE_DS', '14_DS_XML_Locations.xml')
            , nls_charset_id('AL32UTF8')
        )
        columns
              idDepartament     integer      path 'idDepartament'
            , departamentName   varchar2(20) path 'departamentName'
            , departamentCode   varchar2(20) path 'departamentCode'
            , countryName       varchar2(20) path 'countryName'
        ) x;
```

XQuery (XPath Queries)

XPath	Result
`/departament/cities/city/postalCode`	the postalCodes of all cities in the departament
`//postalCode`	all postalCodes (from all nested levels)
`/departament/*`	all things in the departament(s) (which are cities).
`/departament//cityName`	the cityName of every city in the departament(s).
`//cities/city[1]`	the first element of cities (first city)
`//cities/city[last()]`	the last element of cities (last city) in order.
`//cities/city[position()<3]`	the first two cities
`//cities/city[cityName="Iasi"]`	filter all cities with cityName based predicate
`//cities/city[postalCode<2000]`	filter all cities with postal code less than 2000
`//*`	All Elements in XML document.

CASE STUDY: ORCL FDB XMLTABLE

Case Study: FDB access XML Views

- SQL Script Example:
  - 24_AM_XML_ExtTbl_View.sql
- Links:
  - Oracle-Base: XMLTable_1, XML_Table_2,
  - Loading XML File: XML_File
  - Oracle-Base: external table enhancement
  - Oracle XML-based external file solution stackoverflow
  - BFilename function: https://docs.oracle.com/database/121/SQLRF/functions020.htm#SQLRF00610

(2) Data Source Access Components for JSON Data Source (external file)

- Defining DIRECTORY to access external files from disk.
  - Grant necessary privileges to federated schema owner.
- Strategy 1: REMOTE VIEW from external file
  - Using User-defined PL/SQL function to call BFILENAME() to point to external files and DBMS_LOB.LOADFROMFILE PL/SQL procedure to load external data from local filesystem.
  - Create views parsing JSON content with **JSON_TABLE()** function (JSON hierarchical - to - SQL relational model matching) as REMOTE Views on top of UDF-loading function.
- Strategy 2: LOCAL TABLE to load data from external file to store JSON content as CLOB, loading data:
  - From external data file to local SQL table:
    - using EXTERNAL TABLE with ORACLE_LOADER.
  - Create views parsing JSON content with **JSON_TABLE()** function as **local** views from **local** table.

Oracle JSON Processing Support

- JSON_QUERY function: parses JSON document by *json-path parameter* and return one or more JSON values.
- JSON_VALUE function: parses JSON document by *json-path parameter* and return one single scalar value.
- JSON_TABLE function:
    - **parses** and **maps** JSON documents into rows and projects columns from rows with parameters:
        - *json-value* in CLOB or VARCHAR2(4000) format;
        - *json-path* argument: search-parsing path (using JSONPath query expressions);
        - *columns path-definitions* to project parsed JSON parsed rows.
    - combines JSON_QUERY and JSON_VALUE.

JSONPath queries

JSONPath	Result
`$.departament.cities.city[*].postalCode`	The postalCodes of all cities in the departament
`$.departament.cities.city[*]?(@.cityName == "Iasi")`	Filter all cities/city with setted cityName
`$[*]`	All members of JSON array structure.
`$`	All Elements in JSON document.

PL/SQL Custom function to load external data

From JSON or XML data files

```
create or replace FUNCTION get_external_data(
    default_directory VARCHAR2,
    file_path VARCHAR2)
RETURN CLOB IS
    json_file bfile :=
bfilename(UPPER(default_directory),file_path);
    json_clob clob;
    l_dest_offset   integer := 1;
    l_src_offset    integer := 1;
    l_bfile_csid    number  := 0;
    l_lang_context  integer := 0;
    l_warning       integer := 0;
```

```
begin
    dbms_lob.createtemporary(json_clob,true);
    dbms_lob.fileopen(json_file,
                      dbms_lob.file_readonly);
    dbms_lob.loadclobfromfile (
    dest_lob      => json_clob,
    src_bfile     => json_file,
    amount        => dbms_lob.lobmaxsize,
    dest_offset   => l_dest_offset,
    src_offset    => l_src_offset,
    bfile_csid    => l_bfile_csid ,
    lang_context  => l_lang_context,
    warning       => l_warning);
    dbms_lob.fileclose(json_file);
    return json_clob;
End;
/
```

JSON View External Data Source

```
with
json as
    (select get_external_data('EXT_FILE_DS','15_DS_JSON_Locations.json') doc from dual)
SELECT idDepartament , departamentName,  departamentCode, countryName
FROM  JSON_TABLE(                                      json-value
             (select doc from json),
             '$.departaments.departament[*]'                  json-path
             COLUMNS ( idDepartament   PATH '$.idDepartament'
                     , departamentName PATH '$.departamentName'
                     , departamentCode PATH '$.departamentCode'
                     , countryName     PATH '$.countryName'
                     )
);
```

External TABLE File Data Source

```
create table departaments_raw (DOC_JSON CLOB
    -- CONSTRAINT ck_JSON CHECK(DOC_JSON IS JSON)
) organization external
(
  type oracle_loader
  default directory EXT_FILE_DS
  access parameters
  (
    records delimited by newline
    nobadfile nodiscardfile nologfile
    fields missing field values are null
    reject rows with all null fields
    ( json_filename char(80) )
    column transforms (DOC_JSON from lobfile (json_filename) from (dir) CLOB)
  )
  location ('jsonfiles.txt')
) reject limit unlimited;
```

jsonfiles.txt
15_DS_JSON_Locations.json

JSON View on local TABLE from External Data Source

```
with
json as ( select j.doc_json.departaments.departament doc from departaments_raw j )
SELECT idDepartament , departamentName,  departamentCode, countryName
FROM  JSON_TABLE( (select doc from json) , '$[*]'
            COLUMNS ( idDepartament   PATH '$.idDepartament'
                    , departamentName PATH '$.departamentName'
                    , departamentCode PATH '$.departamentCode'
                    , countryName     PATH '$.countryName'
                    )
);
```

XPath vs JSONPath Queries

XPath	JSONPath	Description
/	$	the root object/element
.		the current object/element
/	. or []	child operator
..		parent operator
//	..	recursive descent
*	*	wildcard. All objects/elements regardless their names.
@		attribute access. JSON structures have fields (by default).
[]	[]	subscript operator. XPath uses it to iterate over element collections and for predicates. In Javascript and JSON it is the native array operator.
[]	?()	applies a filter (script) expression.
()		grouping in Xpath

XPath vs JSONPath Queries

XPath/XQuery	JSONPath	Result
`/departament/cities/city/postalCode`	`$.departament.cities.city[*].postalCode`	the postalCodes of all cities in the departament
`//postalCode`	`$..postalCode`	all postalCodes
`/departament/*`	`$.departament.*`	all things in departament, which are cities.
`/departament//cityName`	`$.departament..cityName`	the cityName of every city in the departament.
`//cities/city[1]`	`$..cities.city[0]`	the first cities/city

`//cities/city[last()]`	`$..cities.city[(@.length-1)]`
	`$..cities.city[-1:]`
		the last cities/city in order.

`//cities/city[position()<3]`	`$..cities.city[0,1]`
	`$..cities.city[:2]`
		the first two cities/city elements

`//cities/city[cityName]`	`$..cities.city[?(@.cityName)]`	filter all cities/city elements with cityName
`//cities/city[postalCode<2000]`	`$..cities.city[?(@.postalCode<2000)]`	filter all cities/city elements with postal code less than 2000
`//*`	`$[*]`	All Elements in XML document. All members of JSON array structure.

CASE STUDY: ORCL FDB JSON_TABLE

Case Study: JSON Views

- SQL Script Example:
    - 24_AM_JSON_ExtTbl_View.sql
- Links:
    - Oracle-Base: Oracle_JSON_Type
    - Oracle-Base: JSON in 12c and JSON Functions in 12c
    - Oracle-Doc: JSON in OracleDB
    - AskTom: JSON_File
    - LiveOracle: JSON array parsing
        - AskTom: JSON NestedArray

(3) Data Source Access Components for XLSx and XML docs from Web Data Sources

- PL/SQL Utility functions
  - **UTL_HTTP** to open and read web data streams.
    - There is a UTL_TCP utility that could be used to manage FTP connections.
    - Also, there is a UTL_FTP (open source) library.
  - **DBMS_LOB** to manage data streams as CLOB or BLOB data types.
  - **HTTPURITYPE** to sum-up web stream processing flow.
- Data Integration Process:
  - Set-up SSL Certification Wallet (for HTTPS-based access URLs)
  - Set-up ACL policy to allow access to external data sources.
  - Get web data from URL using PL/SQL tools.
  - Process web data (in XML, XLS, JSON format) with SQL tools: XMLTABLE function, ExcelTable package or JSON_TABLE function.
  - Store SQL Processing Result as a Remote View or as a Local Table.

Set-up SSL Certification Wallet (for HTTPS-based access URLs)

```
-- (1) Access https://data.gov.ro and download cert files ---------------------
-- Chrome URL: https://data.gov.ro, Info-button -> Connection is secure ->
--             Cerificate is Valid -> Details: Certification Path ->
--             Select Root Certificate in Certificate Hierarchy -> Export
--                  DigiCert Global Root G2.pem
-- -> View Certificate -> Details -> Copy to File
-- Copy to File: "D:\fdbo_study_case\1_DataSources\cert_data.gov.ro\DigiCert Global Root G2.pem"
--
-- Docker Copy to File: "/opt/oracle/oradata/fdbo_study_case/1_DataSources/cert_data.gov.ro/DigiCert Global Root G2.pem"
--
-- (2) Create local wallet -----------------------------------------------------
-- SET JAVA_HOME="C:\Program Files\Java\jdk1.8.0_73"
-- CD D:\fdbo_study_case\cert_data.gov.ro\
-- Docker: cd /opt/oracle/oradata/fdbo_study_case/1_DataSources/cert_data.gov.ro/
--
-- orapki wallet create -wallet wallet -pwd trust.01 -auto_login
-- orapki wallet add -wallet wallet -trusted_cert -cert "DigiCert Global Root G2.pem" -pwd trust.01
```

Set-up ACL policy to allow access to external data sources
Use SYS/SYSTEM administrative accounts

```
begin
  DBMS_NETWORK_ACL_ADMIN.append_host_ace (
      host       => '*',
      lower_port => NULL,
      upper_port => NULL,
      ace        => xs$ace_type(privilege_list => xs$name_list('http'),
                                principal_name => 'fdbo',
                                principal_type => xs_acl.ptype_db));
  end;
/
```

Get web data from URL using PL/SQL tools
Use integration user/schema account

```
-- Set Wallet to UTL_HTTP PL/SQL package ----------------------------------
begin
    UTL_HTTP.set_wallet('file:' ||
      'D:\fdbo_study_case\cert_data.gov.ro\wallet', 'trust.01');
end;
/

-- Execute HTTPURITYPE-based Query using ExcelTable.getRows() function
with web_data as (SELECT HTTPURITYPE.
                  createuri('http://data.gov.ro/storage/f/2013-11-01T14%3A30%3A22.936Z/infocod-oct-2013.xls)
                        .getblob() as doc from dual)
select t.* from web_data r, TABLE(
    ExcelTable.getRows( r.doc , 'Localitati > 50.000 loc' ,   '"Judet" VARCHAR2(100),
            "Localitate" VARCHAR2(200), "Tip artera" VARCHAR2(200), "Denumire artera" VARCHAR2(200),
            "Numar/Bloc" VARCHAR2(200), "Codpostal"   VARCHAR2(200)'
       , 'A2')
) t;
```

Set-up SSL Certification Wallet (for HTTPS-based access XML URLs)

```
-- (1) Access https://www.bnr.ro and download cert files ----------------------
-- Chrome URL: https://www.bnr.ro, Info-button -> Connection is secure ->
--             Cerificate is Valid -> Details -> Certification Path ->
--             Root Certificate in Certificate Hierarchy -> Export
--                  "OU=certSIGN ROOT CA,O=certSIGN,C=RO.pem"
-- -> View Certificate -> Details -> Copy to File
-- Copy to File: "D:\fdbo_study_case\1_DataSources\cert_bnr.ro\OU=certSIGN ROOT CA,O=certSIGN,C=RO.pem"
--
-- Docker Copy to File: "/opt/oracle/oradata/fdbo_study_case/1_DataSources/cert_bnr.ro/OU=certSIGN ROOT CA,O=certSIGN,C=RO.pem"

-- (2) Create local wallet ----------------------------------------------------
-- SET JAVA_HOME="C:\Program Files\Java\jdk1.8.0_73" -- or JDK11+
-- CD D:\fdbo_study_case\cert_bnr.ro\
--
-- Docker: cd /opt/oracle/oradata/fdbo_study_case/1_DataSources/cert_bnr.ro/
--
-- orapki wallet create -wallet wallet -pwd trust.01 -auto_login
-- orapki wallet add -wallet wallet -trusted_cert -cert "OU=certSIGN ROOT CA,O=certSIGN,C=RO.pem" -pwd trust.01
```

Get web data from XML URL using PL/SQL tools
Use integration user/schema account

```
-- Set Wallet to UTL_HTTP PL/SQL package ----------------------------------
begin
    UTL_HTTP.set_wallet('file:' ||
      'D:\fdbo_study_case\cert_bnr.ro\wallet', 'trust.01');
end;
/
------ Cursul BNR la zi [http://www.bnr.ro/nbrfxrates.xml]: XML Raw Data
CREATE OR REPLACE VIEW xmldata_bnr_rest_view AS
with
    xmldata_raw as
        (SELECT HTTPURITYPE.createuri('https://www.bnr.ro/nbrfxrates.xml').getclob() as doc from dual)
SELECT SUBSTR(x.doc,
            INSTR(x.doc, '<Body>'),
            INSTR(x.doc, '</Body>') - INSTR(x.doc, '<Body>') + LENGTH('</Body>')) as doc
            FROM xmldata_raw x;
```

Get web data from XML URL using PL/SQL tools
Use integration user/schema account

```
------ Cursul BNR la zi [http://www.bnr.ro/nbrfxrates.xml]  --------------------
select x.currency, x.multiplier, x.curs, x.multiplier * x.curs as multiplier_curs
    from xmldata_bnr_rest_view r,
        XMLTABLE(
            '/Body/Cube/Rate'
            passing XMLTYPE(r.doc)
            columns
                     currency   VARCHAR2(100) path '@currency'
                    ,multiplier  VARCHAR2(100) path '@multiplier' DEFAULT 1
                    ,curs NUMERIC(8,4) path '/'
        ) x;
```

CASE STUDY: ORCL FDB Web access with HTTPURITYPE

Case Study: XML and XLSx Data Source Web HTTP Views

- SQL Script Example:
  - 25_AM_Web_XTbl_View_datagov.sql
  - 25_AM_Web_XTbl_View_BNR.sql
- Links:
  - https://oracle-base.com/articles/misc/retrieving-html-and-binaries-into-tables-over-http
  - https://docs.oracle.com/database/121/ARPLS/u_http.htm
  - https://docs.oracle.com/database/121/ARPLS/t_dburi.htm#ARPLS71705
  - https://oracle-base.com/articles/misc/utl_http-and-ssl
  - APEX.19.x XML/JSON Processing
    - https://blogs.oracle.com/apex/super-easy-csv-xlsx-json-or-xml-parsing-about-the-apex_data_parser-package

Misc.Links

- https://oracle-base.com/articles/misc/retrieving-html-and-binaries-into-tables-over-http
- https://oracle-base.com/articles/12c/fine-grained-access-to-network-services-enhancements-12cr1
- http://blog.whitehorses.nl/2010/05/27/access-to-https-via-utl_http-using-the-orapki-wallet-command/
- https://davidkyanek.blogspot.com/2017/04/using-oracle-wallets-to-connect-ssl.html