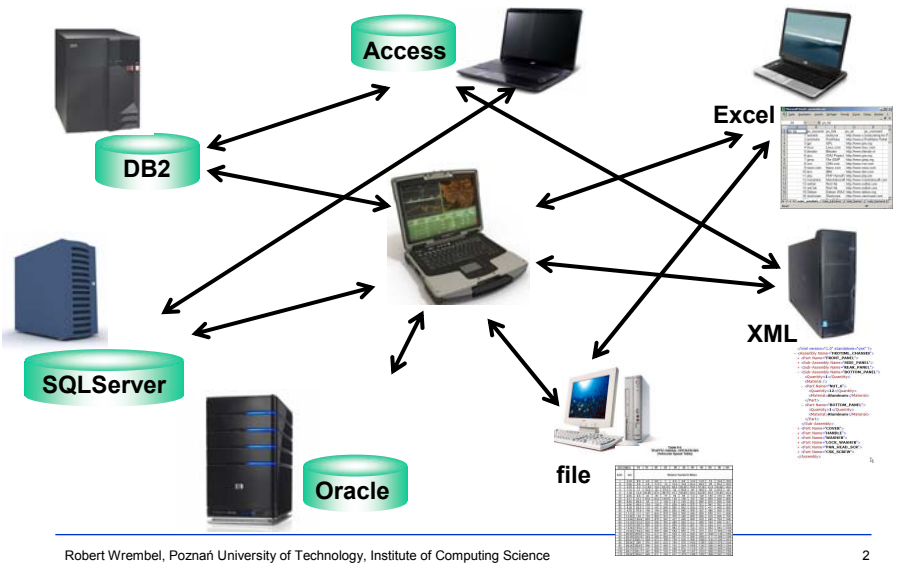
On Building Integrated and Distributed Database Systems

Data Integration Problems and Architectures

Robert Wrembel
Poznań University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Need for data integration



Need for data integration

- **Accessing all important data gathered the lifecycle of a company**
  - **data analysis**
  - **decision support**

Data integration problems

- **Source systems features**
  - **geographically distributed**
  - **autonomous**
    - **managed independently**
      - separate users
      - turned off/of at any time
    - **may evolve independently**
      - structure (schema)
      - new software
  - **heterogeneous**

Heterogeneity of source systems

- Different software producers
  - eg., IBM, Oracle, Microsoft, Sybase, NCR, ...
- Different implementation technologies
  - .Net, C++, C#, Java, PHP, ...
- Different functionality
  - databases / pseudo-databases / none-databases
  - SQL dialects
  - data access drivers
  - data access and processing techniques

Heterogeneity of source systems

- Different data models
  - hierarchical, network
  - relational
  - object
  - object-relational
  - multidimensional
  - semistructured (XML)
- Different data types
  - smallint, int, biging, decimal (SQLServer)
  - smallint, int, biging, float, real, double (DB2)
  - number, binary_integer (Oracle)
  - constant and variable length string data types

Heterogeneity of source systems

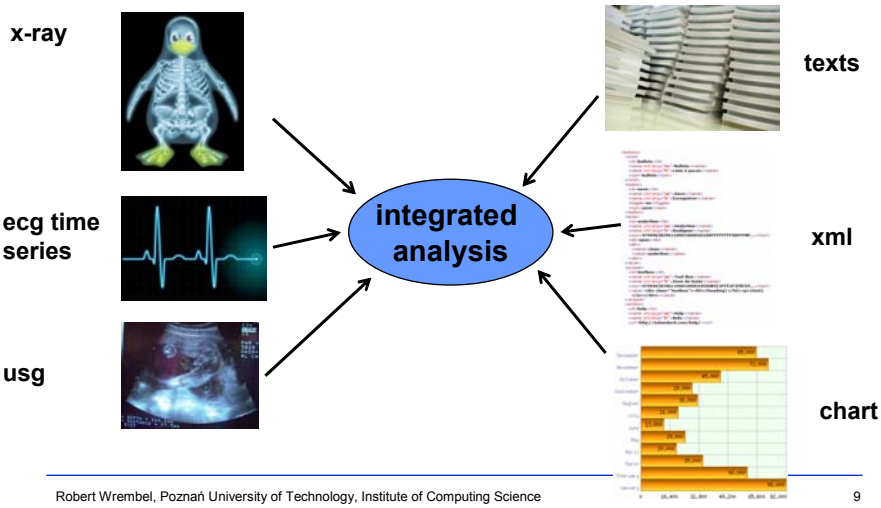
- Different data structures representing the same information
  - Dealer A
    - table Vehicles {VId, make, model, engineType, engineCap, horsepower, vType, maxPersons, maxLoad
    - stores trucks, SUVs, ...
  - Dealer B
    - table Trucks {engineNo, make, model, engineType, maxLoad}
    - table SUVs {engineNo, make, model, engineType, engineCap, horsepower}

Heterogeneity of source systems

- Duplicated data
- Missing and wrong data
- Different measurement units
  - price (EUR, GBP, USD, ...}
  - weight {pounds, kilograms, ...}
- Different abbreviations and symbolic values
  - sex: {F, M}, {1, 0}, {female, male}
  - GB, Great Britain, G.Britain, ...
  - prof. ⇨ professor, prof. ⇨ professional
- Homonymes
  - Supplier.code ⇨ postal code
  - Product.code ⇨ bar code
- Synonymes
  - Patient.SSN
  - Patient.Id ⇨ storing the value of SSN

Heterogeneity of source systems

- Complex data ⇨ typical problem in IS for health services



Heterogeneity of source systems

- Complex data ⇨ typical problem in
  - GISs
  - weather forecasting and monitoring systems

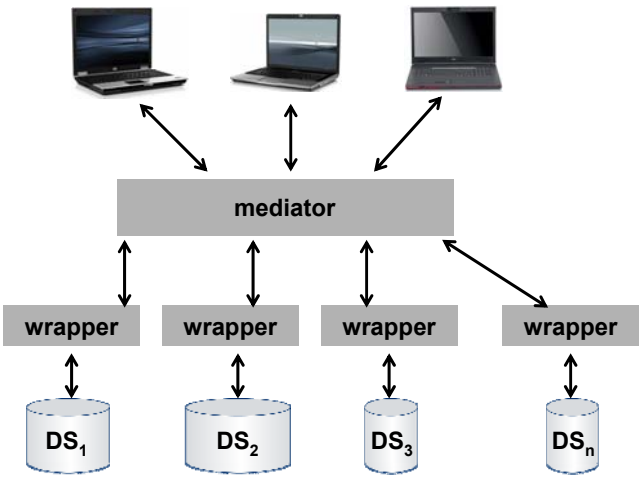
Data integration architectures

- Mediated
- Federated
- Peer to peer (P2P)
- Data warehouse

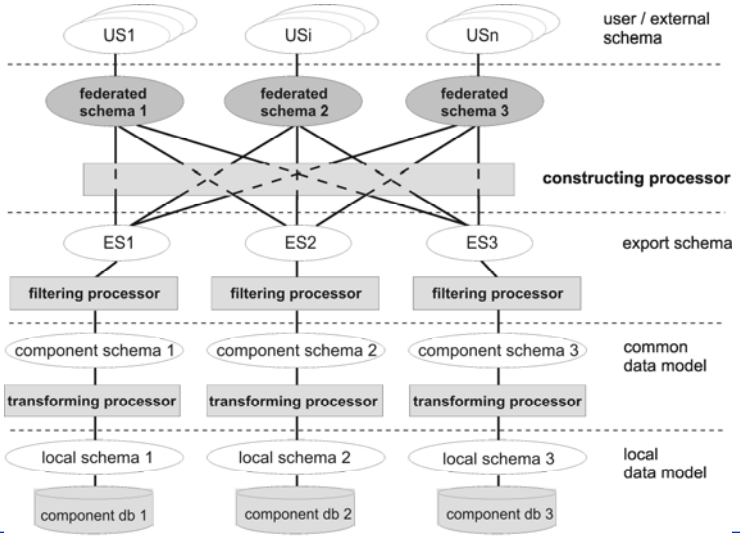
virtual integration

physical integration

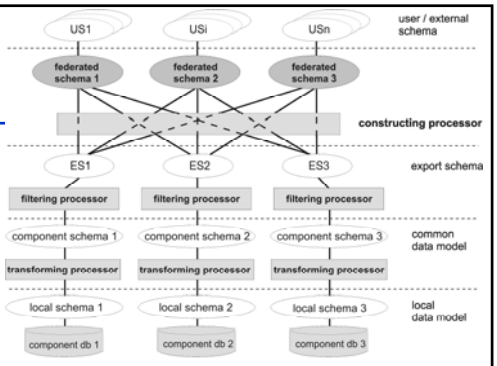
Mediated



Federated

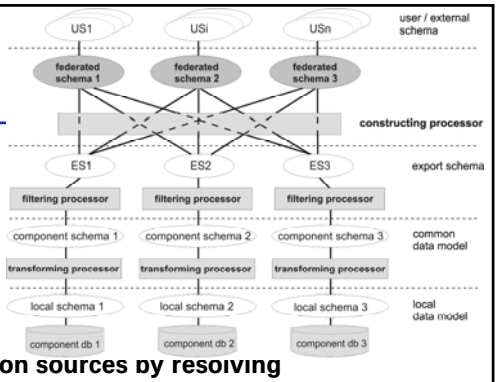


Federated



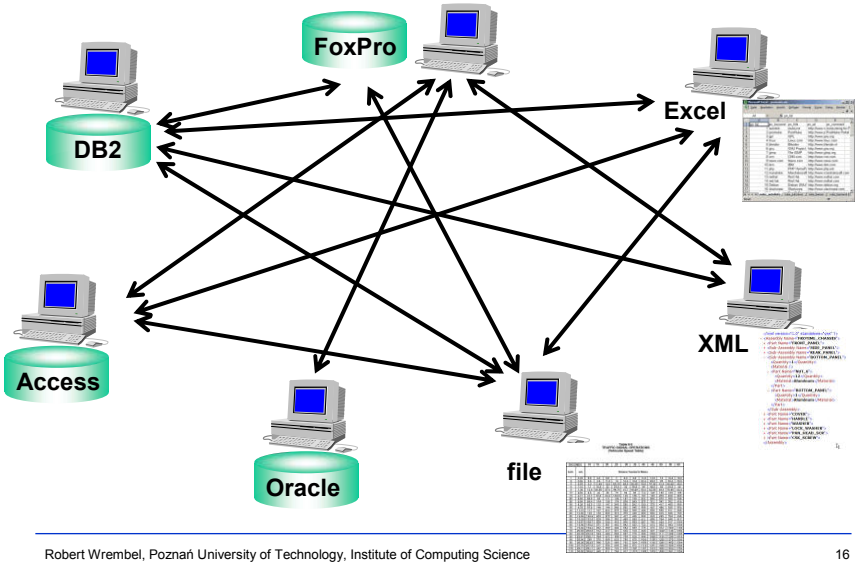
- **Transforming processor**
  - **maintaining mappings between local and component schema elements**
  - **translation of commands from a federated query language to a query language of a component database**
  - **translation of data from a local to common data format**
- **Filtering processor**
  - **controls the set of operations that are issued for a component schema using the information about data visibility and access control specified in an export schema**

Federated



- **Constructing processor**
  - **integrating different information sources by resolving inconsistencies and conflicts between them**
  - **determining the set of data sources capable to answer a given query that was issued and formulated in terms of a federated schema**
  - **decomposing, optimising, and transforming the query into local queries, that is queries for each of the data sources**
  - **sending each local query to appropriate data source**
  - **receiving query results from data sources, translating, filtering, and merging these results to form a global result**

P2P



WEB Services

- Applications' integration technology
  - allows programs written in different languages on different platforms to communicate with each other in a standards-based way
  - software service
- Exchange data between different applications and different platforms
- Program-to-program communications model, built on existing and emerging standards such as HTTP, XML, SOAP, WSDL, and UDDI
- Exchanged data encoded as XML

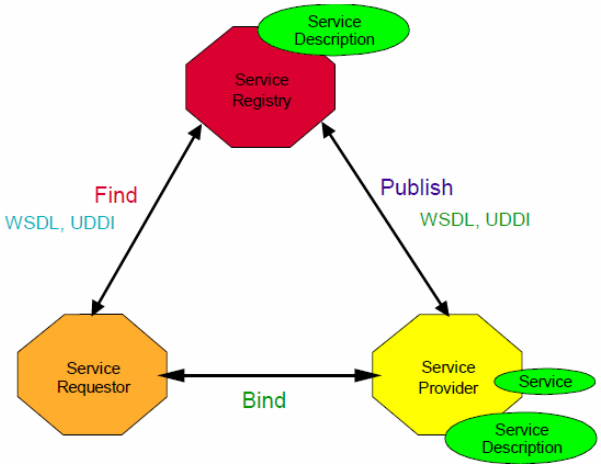
WEB Services

- Transport protocol
  - XML-based Simple Object Access Protocol (SOAP) to let applications exchange information over HTTP
- Web services provide a way to describe their interfaces so that other applications can communicate with Web services
  - this description is usually provided in an XML document called a Web Services Description Language (WSDL) document
- Web services are registered in a directory for the purpose of finding them
  - registration is done by means of Universal Discovery Description and Integration (UDDI)

WEB Services

- Example
- A purchasing application
  - automatically obtain price information from multiple vendors
  - select a vendor
  - submit the order
  - track the shipment until it is received
- A vendor application
  - exposing its services on the Web
  - check the customer's credit
  - charge the customer's account
  - set up the shipment with a shipping company

WEB Services



Web Services Conceptual Architecture (IBM)

WEB Services

- **Service provider**
  - **makes available a software (service)**
  - **provides the service description** (data types, operations, binding information and network location)
  - **publishes the description to a service registry or requestor**
- **Service requestor**
  - **finds and retrieves the service description**
  - **uses the service description to bind with the service provider and invoke or interact with the Web service implementation**
- **Service registry**
  - **searchable registry of service descriptions where service providers publish their service descriptions**
  - **service requestors find services and obtain binding information for services during development for static binding or during execution for dynamic binding**

SOAP

- **Simple Object Access Protocol → Communication protocol for Web services**
- **SOAP is a specification that defines**
  - **the XML format for messages, i.e., how to represent data**
  - **the format of an HTTP message that contains a SOAP message**
- **Independent of application programming languages**
- **SOAP has been implemented on many different hardware and software platforms → applications' (systems') integration technology**

SOAP

- In practice, SOAP messages are created and parsed by various toolkits that translate function calls from some kind of language to a SOAP message.
  - Microsoft SOAP Toolkit translates COM function calls to SOAP
  - Apache Toolkit translates JAVA function calls to SOAP
  - types of function calls and the data types of the parameters depend on a SOAP implementation

WSDL

- Web Services Description Language
- WSDL is an XML document that describes a set of SOAP messages and how the messages are exchanged
  - in practice generated by toolkits based on existing program interfaces
  - parsed by software
- WSDL describes Web services and is used to locate them

WSDL

- The service interface includes
  - WSDL:binding - describes among others a protocol, data format, security for a particular service interface (WSDL:portType)
  - WSDL:portType - defines operations (conterpart of a method signature in a programming language) of a Web service, i.e., what XML messages can appear in the input and output data flows
  - WSDL:message - specifies which XML data types constitute various parts of a message (e.g., input and output parameters of an operation)
  - WSDL:type - describes complex data types

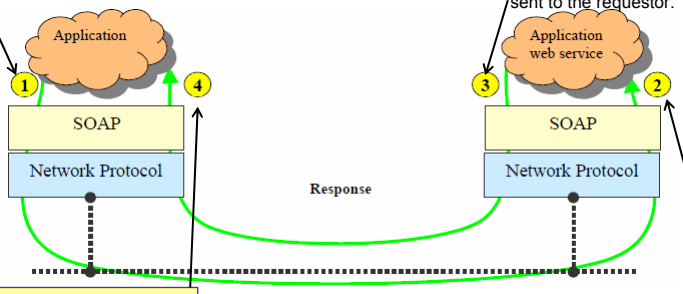
UDDI

- Universal Discovery Description and Integration
- A UDDI directory entry is an XML file that describes a business and the services it offers
- Three parts to the entry in the UDDI directory
  - info about a company offering the service: name, address, contacts, ...
  - industrial categories based on standard taxonomies such as the North American Industry Classification System and the Standard Industrial Classification
  - description of the interface to the service

WEB Services' interaction

Application creates a SOAP message (XML) - a request that invokes the Web service operation at the service provider. SOAP client runtime sends the message to the provided network address using a network protocol (e.g., HTTP).

The service processes the message and produces response that is encoded into SOAP message (XML). The message is sent to the requestor.

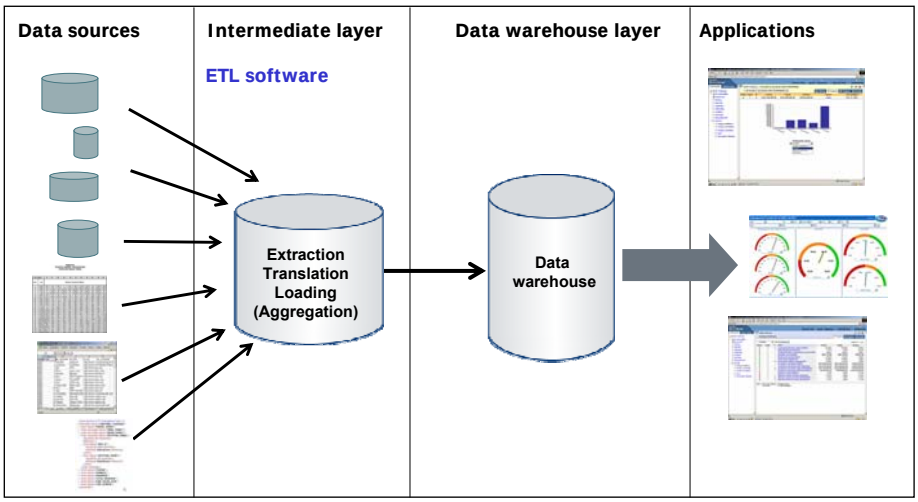


The message is received, converted into native application objects (in a target progr. language), and presented to the application.

The message is delivered the service provider's SOAP runtime (server). The runtime converts the message into programming language-specific objects and routes the request message to the service provider's Web service.

Web Services Conceptual Architecture (IBM)

Data warehouse



Virtual vs. physical integration

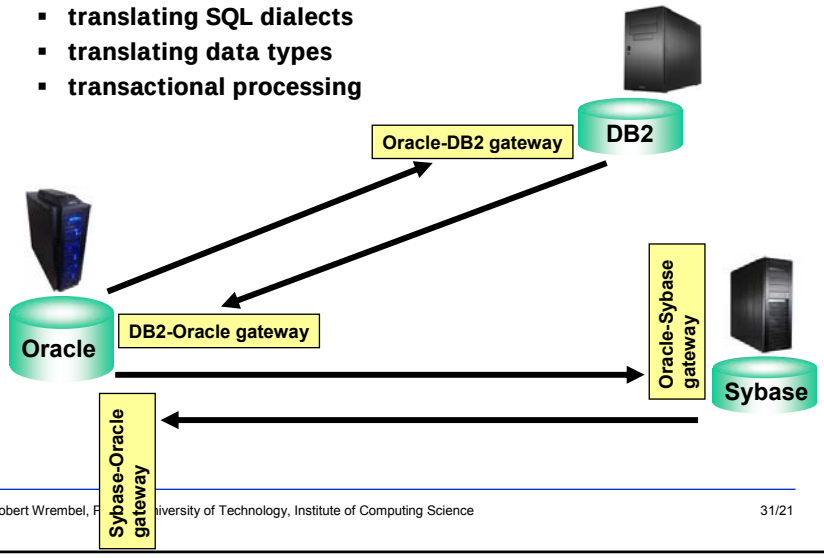
- Virtual → disadvantages
  - results of a query may arrive with a long delay caused by a slow network, or a low response time of data sources
  - decomposition and translation of a query as well as merging the results of a query incur additional time overhead
  - queries coming from a federated system may interfere with queries executed locally in component databases, as a consequence, federated queries may slow down the execution of the local queries
  - some of the component data sources may be temporarily unavailable, thus making the query results incomplete or unavailable
- Virtual → advantages
  - no data redundancy
  - access to up to date data
  - user can query any data that is available

Virtual vs. physical integration

- Physical → disadvantages
  - data redundancy
  - need of data refreshing
- Physical → advantages
  - quicker access to data (data stored locally)
  - no need of query and data transformations
  - independence on unavailability of data sources

Accessing heterogeneous data sources

- Wrappers/gateways
  - translating SQL dialects
  - translating data types
  - transactional processing



Accessing heterogeneous data sources

- ODBC
  - standardized access methods (API) for multiple data sources (databases, text files, dbf files, ...)
  - ODBC/JDBC driver ➔ API
  - OS: MS Windows, Unix, Linux, OS/2, OS/400, IBM i5/OS, Mac OS X
- OLE DB (Object Linking and Embedding DataBase)
  - API for accessing multiple data sources under Windows from COM-based programs (e.g., VB)
- Dedicated drivers for flat and i XML files
- JDBC
  - counterpart of ODBC for Java applications
- unixODBC
  - Linux Red Hat, Mandriva, and Gentoo

Software

- SAP
  - BusinessObjects Data Integrator - ETL
  - BusinessObjects Data Federator - mediated
    - acces to relational and non-relational sources (Oracle, IBM, Microsoft, the SAP NetWeaver Business Warehouse component, SAS, Teradata, Web service, and XML)
- IBM
  - InfoSphere Information Server (from Ascential)
  - InfoSphere DataStage
  - InfoSphere Change Data Capture
  - InfoSphere Quality Stage
  - InfoSphere Federation Server - mediated/federated

Software

- Oracle
  - Transparent Gateways
    - access to IBM DB2 and Informix, Sybase Adaptive Server Enterprise, MS SQL Server, Teradata
  - Warehouse Builder
  - Data Integrator
- Microsoft
  - SQL Server Integration Services
    - access to Oracle, XML, ODBC and OLE DB data sources

XML → database

➲ Loading XML document into a DB (Oracle)

- **XML file → CLOB attribute**

```
DBMS_LOB.LoadFromFile(dest_lob=>clob_ptr, src_bfile=>bfile_ptr,
                      amount=>DBMS_LOB.GetLength(bfile_ptr))
```

or

```
CREATE TABLE myxml (doc_id NUMBER(6), doc XMLType);
INSERT INTO myxml VALUES (1, XMLType.CreateXML(xmldoc));
```

- **CLOB → table**

```
rows_processed:=DBMS_XMLSAVE.InsertXML(insCtx, xmlDoc);
```


Database → XML

➲ Getting XML data out of database (Oracle)

- **Query result → CLOB attribute**

```
result CLOB;
result:=DBMS_XMLQUERY.GetXML(query);
insert into export_xml values(rec_id, result);
```

or

```
xml CLOB;
xml:=DBMS_XMLGEN.GetXML(ctx_query);
```

- **CLOB → file**

```
process in loop the whole CLOB content
UTL_FILE.Put(FileHandle,
             DBMS_LOB.Substr(myXML, Remainder, myCounter));
```

or

```
DBMS_LOB.Read(xmlString,amount,position,charString);
DBMS_OUTPUT.Put_line(charString);
```

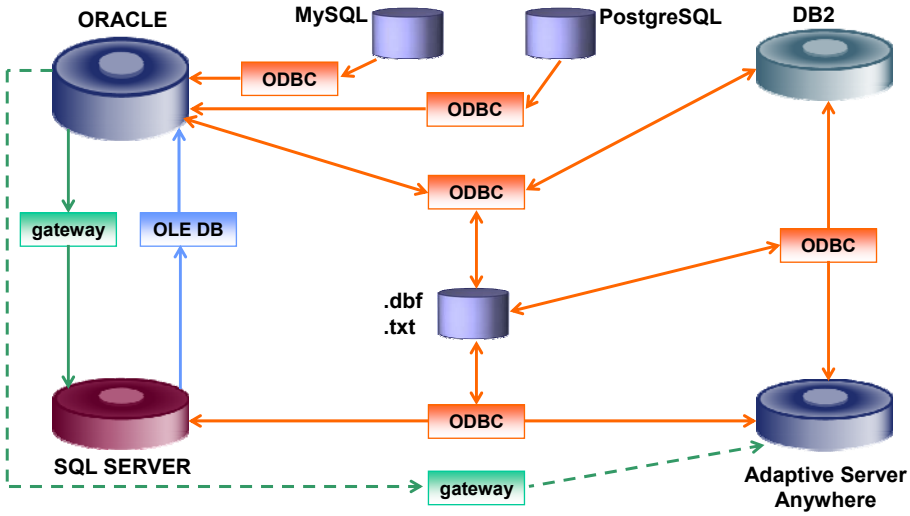
Database → XML

- Relational data into XML document
  - use object types and views for creating nested rows
  - use cast (multiset (select ...) ...)

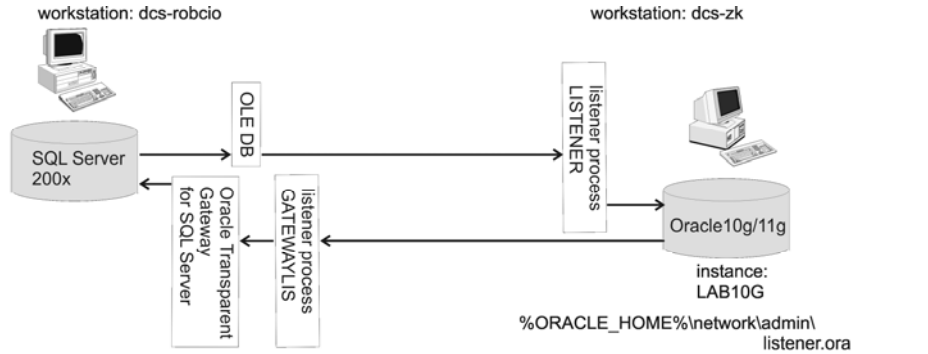
```
create or replace view vo_przedsiebiorstwo as
select prz.nazwa_przeds nazwa_przedsiebiorstwa, ...,
       woj.nazwa wojewodztwo, ..., ttr.nazwa typ_transportu, ...
       cast (multiset (select tdz.nazwa
                       from   dzialalnosc_przeds dzp, typ_dzialalnosci tdz
                       where  dzp.kod_przedsieb=prz.kod
                       and    tdz.obj_id=dzp.oid_typ_dzialaln
                      ) as TAB_TYP_DZIAL
            ) typ_dzialalnosci,
       cast (multiset (select nar.nazwa, ...,
                              reg.nazwa regulacja, sna.kwota, ...
                       from   zaplanowana_kp zkp, ...
                       where  zkp.kod_przedsieb=prz.kod
                       and    zkp.kod=kno.kod_kprz
                       ...
                      ) as TAB_TYP_NARUSZENIE
            ) naruszenie
from   przedsiebiorstwo prz, ...
where  ...
```

collection object types

Case study 1



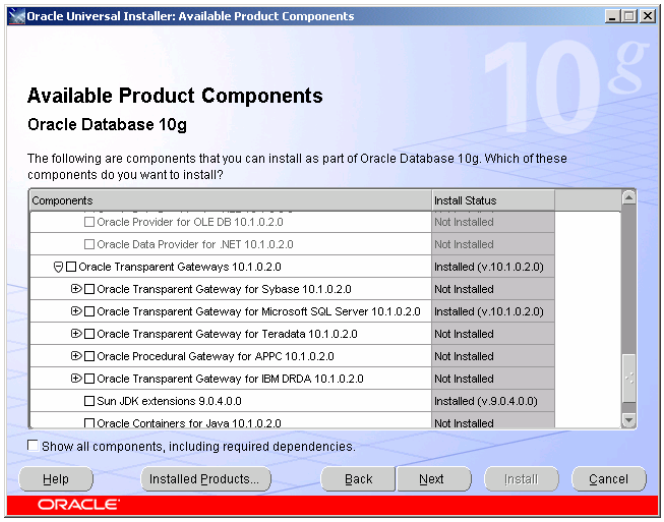
Oracle ←→ SQL Server



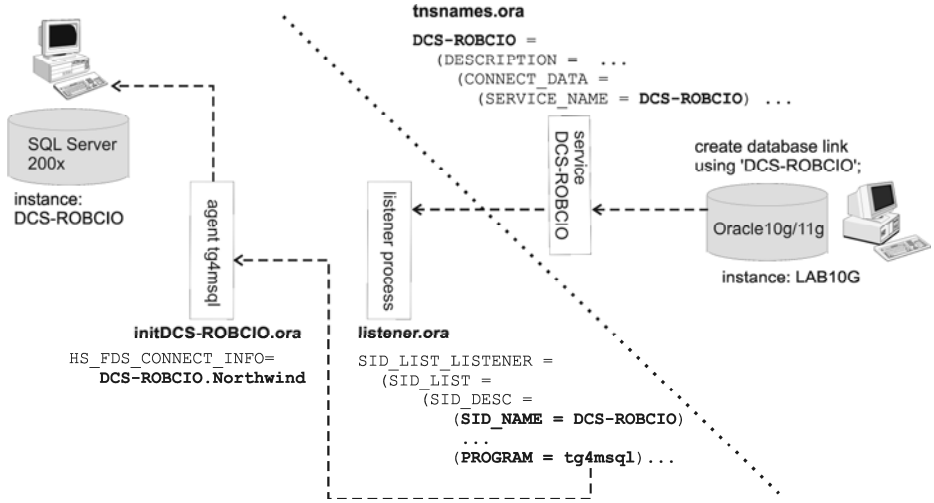
%ORACLE_HOME%\network\admin\
listener.ora
tnsnames.ora
sqlnet.ora
%ORACLE_HOME%\tg4msql\admin\initg4msql.ora

%ORACLE_HOME%\network\admin\
listener.ora
tnsnames.ora
sqlnet.ora
%ORACLE_HOME%\tg4msql\admin\initg4msql.ora

Installing gateways

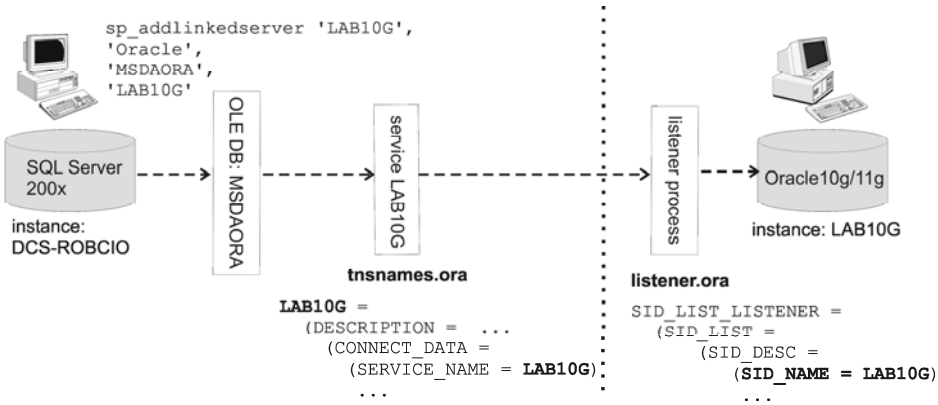


Oracle ➔ SQL Server



SQL Server ➔ Oracle

ODBC data source named LAB10G was defined

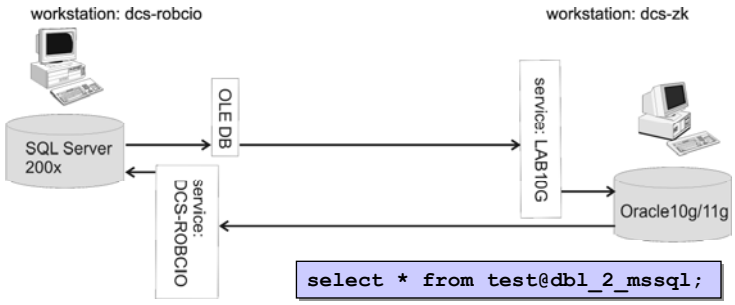


Accessing SQL Server from Oracle

- Database links

```
create database link dbl_1_mssql using 'DCS-ROBCIO';
```

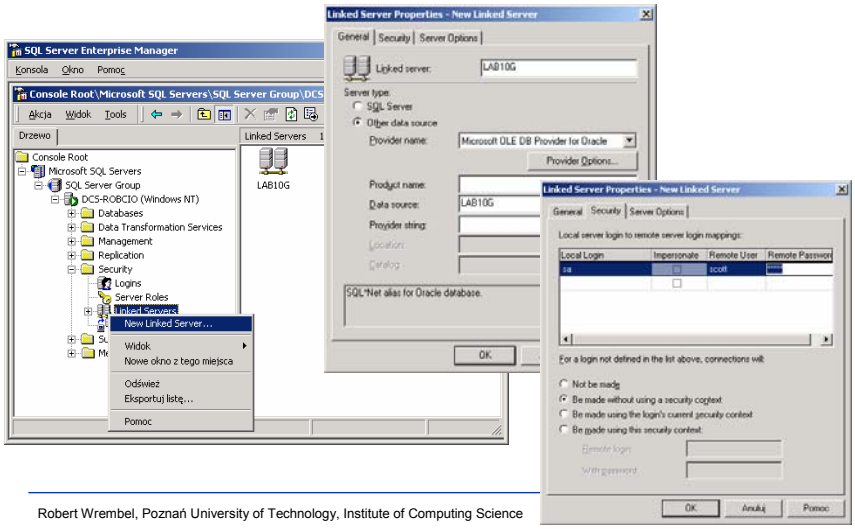
```
create database link dbl_2_mssql
   connect to scott identified by tiger
   using 'DCS-ROBCIO';
```




```
select * from test@dbl_2_mssql;
```


Accessing Oracle from SQL Server (1)

- Using linked server



Accessing Oracle from SQL Server (2)

```
sp_addlinkedserver 'LAB10G', 'Oracle', 'MSDAORA', 'LAB10G'
```

linked server name

product type

OLE DB driver type

ODBC data source

```
sp_addlinkedsrvlogin 'LAB10G', false, 'sa', 'scott', 'tiger'
```

linked server name

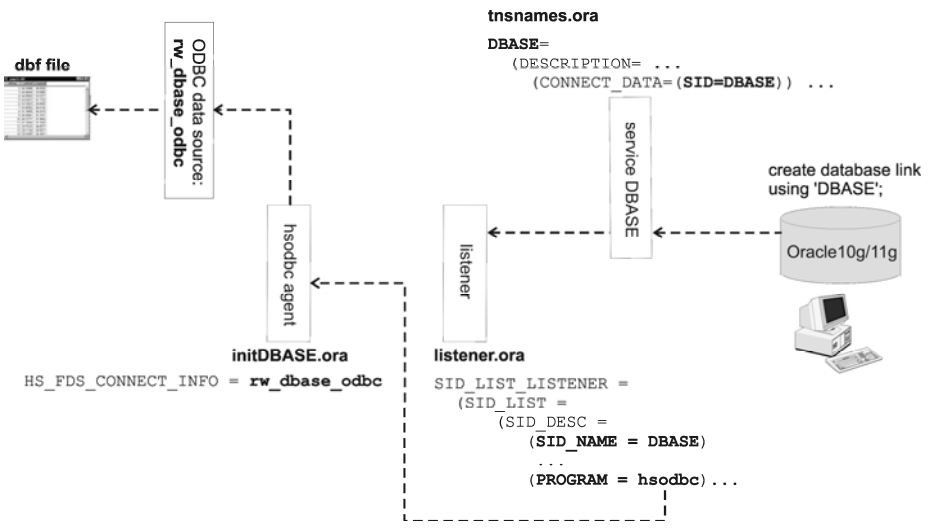
subsequent arguments represent users mapping

SQLServer user

Oracle user and his password

```
select * from LAB10G..SCOTT.EMP
```


Oracle ➔ Access, dbf



Oracle ➔ IBM DB2

%ORACLE_HOME%\network\admin\tnsnames.ora

```
DB2=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
                 (HOST = lab234-d)
                 (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = DB2)
    )
    (HS = OK)
  )
```

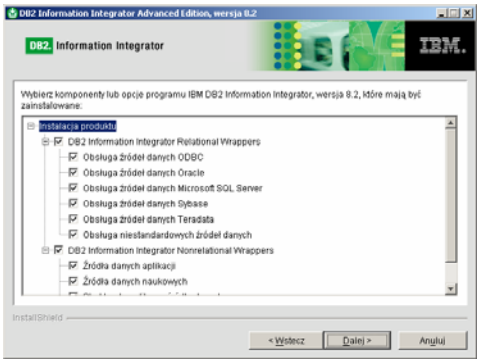
%ORACLE_HOME%\network\admin\listener.ora

```
(SID_DESC =
      (SID_NAME = DB2)
      (ORACLE_HOME = d:\oracle\ora92)
      (PROGRAM = hsodbc)
```

%ORACLE_HOME%\hs\admin\initDB2.ora

```
HS_FDS_CONNECT_INFO = ZRÓDŁO_ODBC_DB2
```


IBM DB2 ➔ Oracle (1)



1. Install ODBC driver for Oracle (component of DB2 Information Integrator)
2. Define ODBC Data Source Name connecting to Oracle

IBM DB2 ➔ Oracle (2)

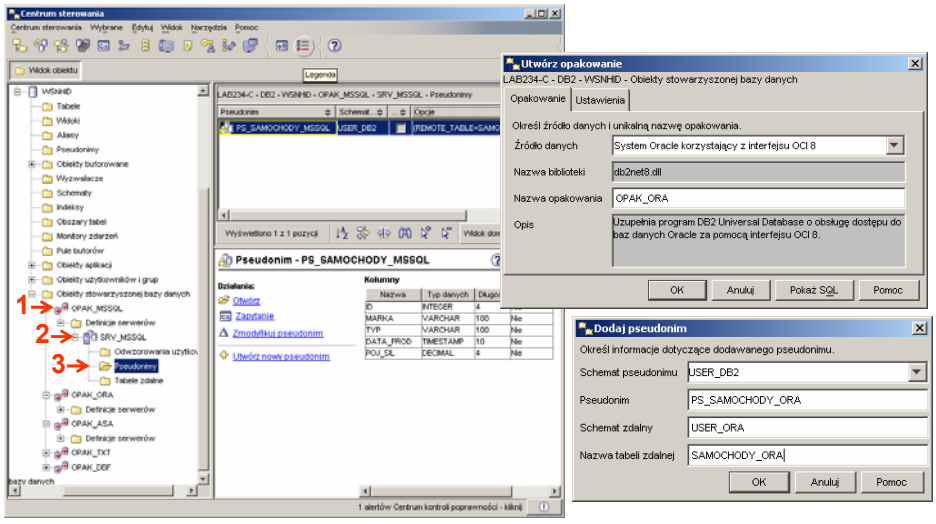
- Accessing external database from DB2 – database objects used
  - wrapper
    - defines data source type and its name
    - includes a server definition → represents in DB2 external data source
  - mapping a DB2 user into an external user

```
CREATE USER MAPPING FOR "USER" SERVER "DB2"
OPTIONS ( ADD REMOTE_AUTHID 'USER_ORA',
ADD  REMOTE_PASSWORD 'TEST') ;
```

  - pseudonim
    - represents in DB2 external object (e.g., table)

```
select * from pseudonim;
```


IBM DB2 ➔ Oracle (3)



Oracle ➔ ASA

%ORACLE_HOME%\network\admin\tnsnames.ora

```
ASA=
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
                 (HOST = lab234-d)
                 (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ASA)
    )
    (HS = OK)
  )
```

%ORACLE_HOME%\network\admin\listener.ora

```
(SID_DESC =
      (SID_NAME = ASA)
      (ORACLE_HOME = d:\oracle\ora92)
      (PROGRAM = hsodbc)
```

%ORACLE_HOME%\hs\admin\initASA.ora

```
HS_FDS_CONNECT_INFO = ZRÓDŁO_ODBC_ASA
```


ASA ➔ Oracle (1)

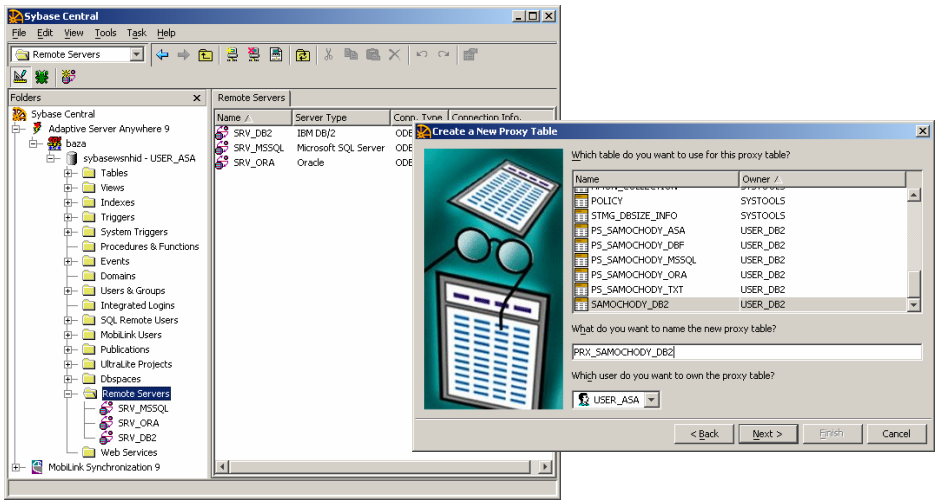
1. Install ODBC driver for Oracle
2. Define ODBC Data Source Name connecting to Oracle

ASA ➔ Oracle (2)

- Accessing external database from ASA → db objects
  - remote server
    - defines remote server type (Oracle, DB2, MS SQL), driver type (ODBC, JDBC), ODBC data source name, and users' mapping
  - proxy table
    - represent in ASA a remote table

```
select * from proxy;
```


ASA ➔ Oracle (3)



Further configurations

- ASA ➔ SQL Server
- ASA ➔ DB2
- DB2 ➔ ASA
- DB2 ➔ SQL Server
- SQL Server ➔ ASA
- SQL Server ➔ DB2
- Oracle ➔ MySQL
- Oracle ➔ PostgreSQL

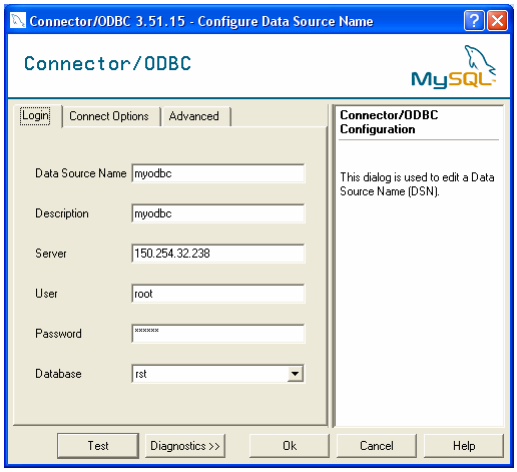
1. Install dedicated ODBC driver for databases being integrated
2. Define Data Source Name for every external database

Further configurations

- Create data source representing MySQL

```
MYSQL =
 (DESCRIPTION =
  (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)
                       (HOST = 150.254.32.226)
                       (PORT = 1521))
  )
  (CONNECT_DATA =
            (SERVICE_NAME = myodbc)
  )
  (HS = OK)
 )

initmyodbc.ora:
HS_FDS_CONNECT_INFO = myodbc
```



Observations (1)

- SQL SERVER instance name has to be single character string
  - wrongly named instance: DCS-RW\SQLSERV
    - problem in configuring the ageng of heterogeneous services (config file wold be named as `initDCS-RW\SQLSERVER.ora`)
  - correctly named instance: SQLSERV
- SQL SERVER → ORACLE
  - insert into a table in Oracle must contain values for all the attributes in the table, even if some attributes may have NULL values
- ORACLE → SQL SERVER
  - inserts from Oracle into an SQL Server table lock the whole table → in SQL SERVER even data reads are impossible

Observations (2)

- SQL SERVER → DB2
  - insert and update from DB2 result in error:

```
Server: Msg 7399, Level 16, State 1, Line 1
OLE DB provider 'MSDASQL' reported an error.
[OLE/DB provider returned message: [IBM][CLI Driver] CLI0150E
  Sterownik nie może wykonać operacji. SQLSTATE=S1C00]
```

- ASA → SQL SERVER
  - translation error into type *decimal* in ASA
  - solution: replace *decimal* with *real*
- Oracle → MySQL
  - ODBC driver for MySQL should have appropriate version (3.51 in our environment was correct, version 5 caused error)
- Oracle → PostgreSQL
  - failure

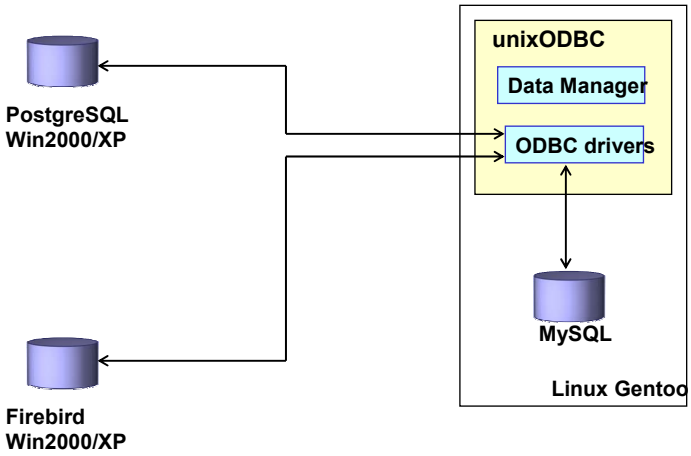
Observations (3)

- Oracle ➔ ASA/ASE (via gateway)

```
ERROR at line 1:ORA-01017:
invalid username/password;
logon denied[Transparent gateway for SYBASE][A07B]
Illegal username and/or password were supplied for datasource
'tg4sybs'ORA-02063: preceding 2 lines from LINK_SYBASE
```

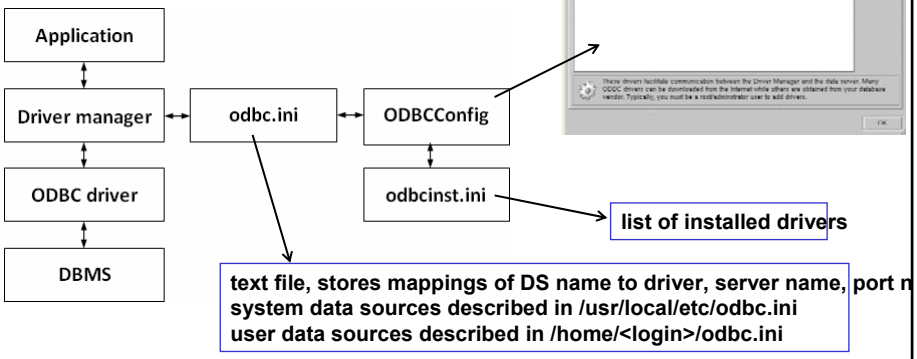
- Accessing dbf files
  - file names may have maximum 8 haracters
  - no transactions
- Accessing MSAccess
  - transactional processing
- General remark: names of db objects and attributes are case sensitive

Integration under Linux



unixODBC

➲ **Software components**



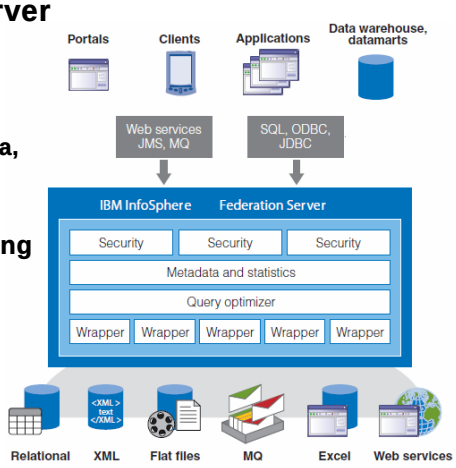
unixODBC

➲ **Problem**

- **driver for Firebird had to be added manually**
- **driver for MySQL did not compile**
- **connection to Firebird ended with error (unsolved)**

IBM approach

- **InfoSphere Federation Server**
  - virtual integration
  - multiple DBMSs
    - Sybase, Informix, MS SQLServer,Oracle, Teradata, Adabas, ...
  - ODBC/JDBC data sources
  - software installed on existing DB2



http://public.dhe.ibm.com/common/ssi/ecm/en/imd11790usen/IMD11790USEN.P

InfoSphere Federation Server

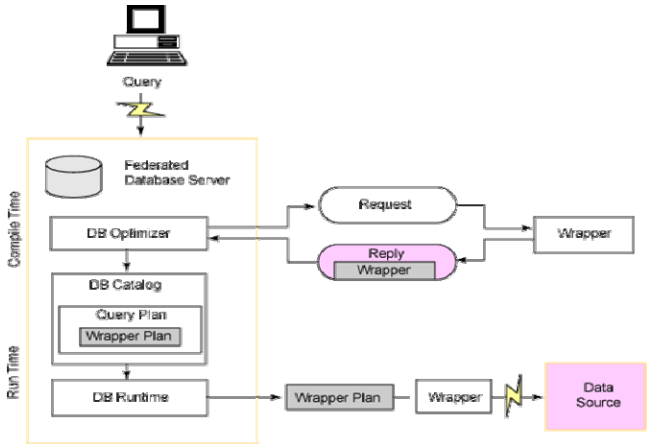
- **Wrapper**
  - set of predefined wrappers (implemented as libraries - their names differ for different OSs)
  - user-implemented wrappers (C++, Java)
- **Server**
  - defines external data source (connection parameters, query execution plan, managing sessions)
- **User mapping**
  - required for external DBMSs
  - maps a local to an external user
- **Nickname**
  - local name for a remote object

Defining wrappers

- db2dj.ini defines OS variables that allow to localize external system
- UNIX: DB2instance_home/sqllib/cfg/db2dj.ini
- Windows: %DB2PATH%\cfg\db2dj.ini
- Required variables in db2dj.ini
  - for Oracle: ORACLE_HOME (localize tnsnames.ora, if not default localization then define TNS_ADMIN)
  - for SQLServer:
    - path do ODBC driver: DJX_ODBC_LIBRARY_PATH, e.g., DJX_ODBC_LIBRARY_PATH=c:\windows\system32
    - ini file for ODBC ODBCINI, e.g., ODBCINI=c:\windows\odbc.ini

Wrapper

- Managing user requests via wrapper



Wrapper for Oracle

➲ Create entry to tnsnames.ora

```
CREATE WRAPPER NET82 LIBRARY 'db2net8.dll'; -- predefined driver

CREATE SERVER ora_serv_name TYPE oracle VERSION 10g
WRAPPER net8
OPTIONS (NODE 'tnsnames_service_name');

CREATE USER MAPPING FOR local_user
SERVER ora_serv_name
OPTIONS (REMOTE_AUTHID 'remote_user', REMOTE_PASSWORD 'remote_passwd');

--testing connection
SET PASSTHRU ora_serv_name
SELECT count(1) FROM schema.table
SET PASSTHRU RESET

CREATE NICKNAME nick FOR ora_serv_name.remote_schema.remote_table;
```


Wrapper for SQLServer

➲ Define ODBC data source → System DSN

```
-- predefined driver
CREATE WRAPPER sqlserver_wrapper LIBRARY 'db2mssql3.dll';

CREATE SERVER sql_serv_name TYPE MSSQLSERVER VERSION 2005
WRAPPER sqlserver_wrapper
OPTIONS (NODE 'DNS_name', DBNAME 'database_name');

CREATE USER MAPPING FOR local_user
SERVER sql_serv_name
OPTIONS (REMOTE_AUTHID 'remote_user', REMOTE_PASSWORD 'remote_passwd');

CREATE NICKNAME nick FOR sql_serv_name.remote_schema.remote_table;
```

Wrapper for Excell

- Excell wrapper
  - access to the firs sheet only
  - Excell has to be installed on the same host as federated server
- ODBC wrapper
  - access to any sheets
  - no need to install Excell

```
-- predefined driver
CREATE WRAPPER excel_wrapper LIBRARY 'db2lsxls.dll';
CREATE SERVER excel_server WRAPPER excel_wrapper;

CREATE NICKNAME nick
(col1 {DATE|DOUBLE|FLOAT|INTEGER|VARCHAR(n)},
 col2 {DATE|DOUBLE|FLOAT|INTEGER|VARCHAR(n)},
       ...)
FOR SERVER excel_server
OPTIONS (FILE_PATH 'c:\...\...\.xls', RANGE 'A1:Zn');
```


Creating wrapper

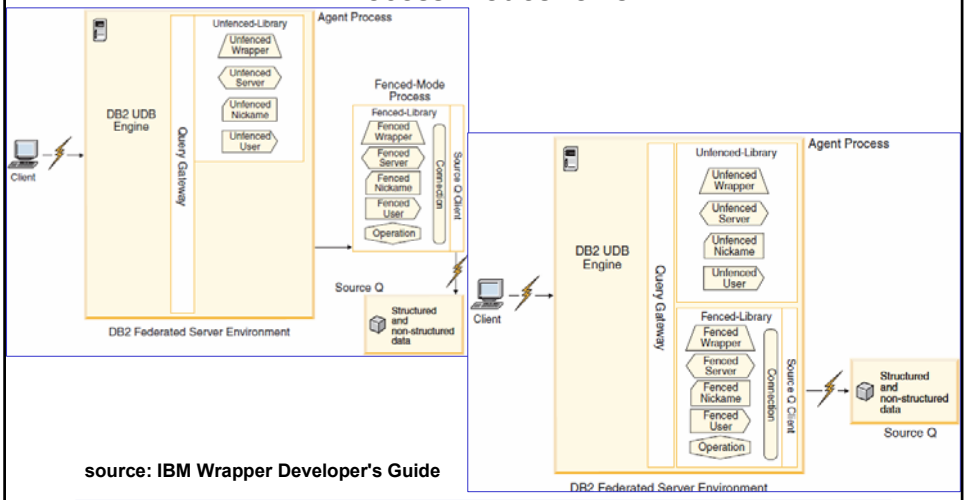
- Wrapper components
  - Wrapper - initializes a wrapper and makes it available in a federation system, defines to which servers the wrapper allows to connect
  - Server - represents data sources the wrapper is designed for, by means of nicknames
  - Nickname - represents the set of tables accessible by wrapper
  - User - user authorization information
  - Connection - maintains connections to an external system, manages transactions
  - Operation - represents commands (DQL, DML), query mode, passthru mode

Creating wrapper

- **Fenced mode - separate the command execution process from other processes - errors do not cause crash of the federation server**
- **Unfenced mode - all processes are executed within the FS**

Creating wrapper

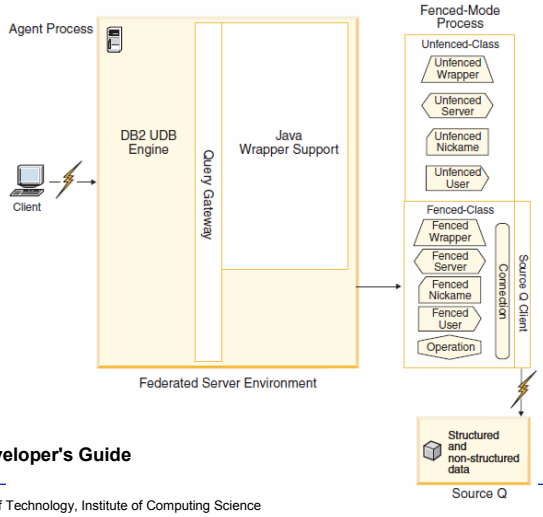
- **Process modes for C++**



source: IBM Wrapper Developer's Guide

Creating wrapper

➲ **Process modes for Java**

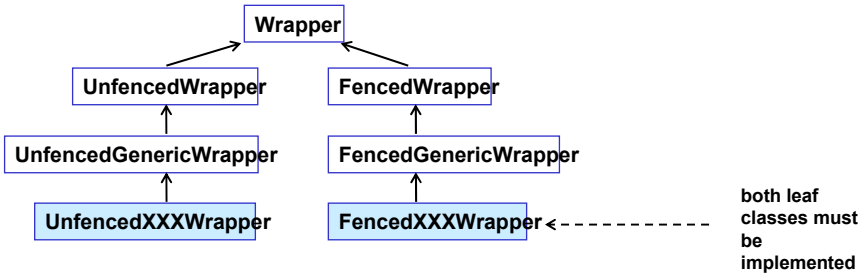


source: IBM Wrapper Developer's Guide

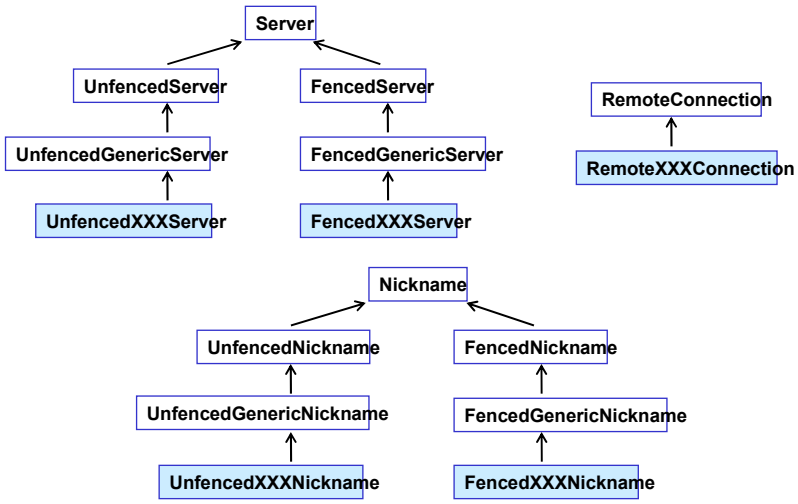
Creating wrapper

➲ **Class hierarchies for implementing wrappers**

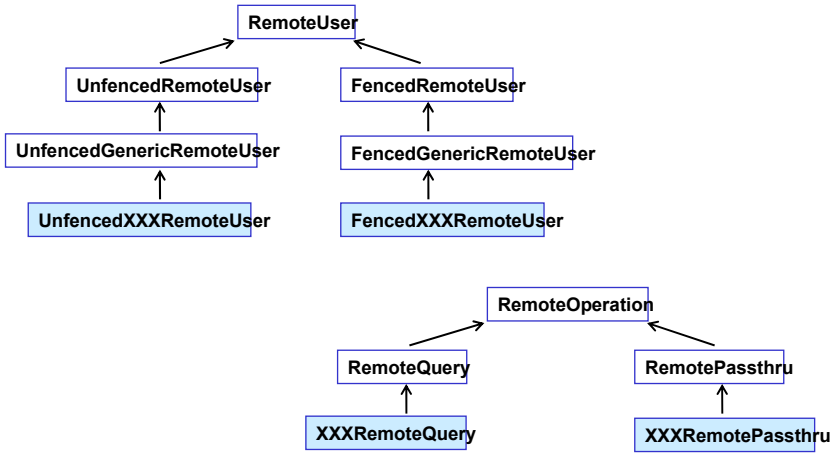
- **library db2qgjava.jar**



Creating wrapper



Creating wrapper



Creating wrapper

```
CREATE WRAPPER RW_wrapper OPTIONS
(UNFENCED WRAPPER CLASS 'UnfencedXXXWrapper')
```

- **While creating a wrapper the system executes the following methods**
  - **constructor of UnfencedXXXWrapper.class**
  - **for object UnfencedXXXWrapper.** verifyMyRegisterWrapperInfo()
  - **UnfencedXXXWrapper.** initializeMyWrapper()

Creating wrapper

```
CREATE SERVER RW_server WRAPPER RW_wrapper;
```

- **While creating a server the system executes the following methods**
  - **create a server object by means of UnfencedXXXWrapper.** createServer()
  - **UnfencedXXXServer.** verifyMyRegisterServerInfo()
  - **UnfencedXXXServer.** initializeMyServer()

Creating wrapper

```
CREATE USER MAPPING FOR user SERVER RW_server;
```

- **While creating a user mapping the system executes the following methods**
  - **create object** UnfencedXXXUser by means of its constructor
  - **UnfencedXXXUser.**verifyMyRegisterUserInfo()
  - **UnfencedXXXServer.**initializeMyUser()

Creating wrapper

```
CREATE NICKNAME nick FOR SERVER RW_server.remote_schema.remote_table
```

- **While creating a nickname the system executes the following methods**
  - **UnfencedXXXServer.createNickname() creates object UnfencedXXXNickname using a constructor of its class**
  - **UnfencedXXXNickname.verifyMyRegisterNicknameInfo()**
  - **UnfencedXXXNickname.initializeMyNickname()**
  - **constructor of class FencedXXXWrapper.class**
  - **FencedXXXWrapper.initializeMyWrapper()**
  - **FencedXXXWrapper.createServer() creates object FencedXXXServer using a constructor of its class**
  - **FencedXXXServer.initializeMyServer()**
  - **FencedXXXServer.createRemoteUser() creates object FencedXXXUser using a constructor of its class**
  - **FencedXXXUser.initializeMyUser()**
  - **FencedXXXServer.createRemoteConnection() creates object XXXConnection using a constructor of its class**
  - **XXXConnection.connect()**
  - **FencedXXXServer.createNickname() creates object FencedXXXNickname using a constructor of its class**
  - **FencedXXXNickname.verifyMyRegisterNicknameInfo()**
  - **FencedXXXNickname initializeMyNickname()**

Creating wrapper

- Internal processing of a query (select * from nick)
  - UnfencedXXXWrapper.createServer() creates object UnfencedXXXServer using a constructor
  - UnfencedXXXServer.createNickname() creates object UnfencedXXXNickname using a constructor
  - UnfencedXXXServer.planRequest()
  - UnfencedXXXServer.createRemoteUser() creates object UnfencedXXXUser
  - FencedXXXWrapper.createServer() creates object FencedXXXServer
  - FencedXXXServer.createRemoteUser() creates object FencedXXXUser
  - FencedXXXServer.createRemoteConnection() creates object XXXConnection
  - XXXConnection.connect()
  - XXXConnection.createRemoteQuery() creates object XXXQuery
  - XXXQuery.open()
  - XXXQuery.fetch()
  - XXXQuery.close()

Example implementation

- **Class** UnfencedFileWrapper
  - need to implement methods
    - UnfencedFileWrapper() and createServer()

```
import com.ibm.db2.wrapper.*;

public class UnfencedFileWrapper extends UnfencedGenericWrapper
{

  public UnfencedFileWrapper()
  {
    super();  <-- calls constructor of UnfencedGenericWrapper
  }

  protected Server createServer(String serverName)
  {
    Server s = new UnfencedFileServer(serverName, this);
    return s;
  }

}
```

Example implementation

- **Class** UnfencedFileServer
  - UnfencedFileServer() - constructor
  - createNickname() - creates nickname object
  - planRequest() - create execution plan

```
1  import java.io.IOException;
2  import com.ibm.db2.wrapper.*;
3
4  public class UnfencedFileServer extends UnfencedGenericServer
5  {
6
7    public UnfencedFileServer(String serverName,
8  UnfencedFileWrapper wrapper)
9    {
10     super(serverName, wrapper);
11   }
12
13   protected Nickname createNickname(String schemaName, String
14 nickname)
15   {
16     return new UnfencedFileNickname(schemaName, nickname, this);
17   }
18
```


Method planRequest()

- **Creates execution plan**
  - **Class Request - its instance represents fragments of a command (columns, clauses, predicates) send to wrapper for execution**
  - **Class Reply - its instances represent fragments of the command that will be executed by the wrapper on remote source**

1. **Create reply object → use createReply() of class UnfencedGenericServer**
2. **Create execution descriptor**
3. **Add quantifier to the reply object**

nb of nicknames in the FROM clau

```
1  Reply reply = createReply(request);
2  FileExecDesc execDesc = new FileExecDesc();
3  reply.addQuantifier(request.getQuantifier(1));
```

Execution descriptor

- Pass to the execution descriptor object a path to a data source file

```
4  UnfencedFileNickname nick = null;
5  nick = (UnfencedFileNickname) request.getNickname(1);
6  CatalogOption path = nick.getInfo().getOption("FILE_PATH");
7  execDesc.setFilePath(path.getValue());
```

- Pass to the execution descriptor the number of columns in the select clause

```
8  int n = request.getNumberOfHeadExp();
9  execDesc.setOutputColumns(n);
```

creates empty array of size n

Object reply

- Add to reply columns from the SELECT clause
- Map columns from SELECT to nickname columns

```
10 for(int i = 1; i <= n; i++)
11 {
12   RequestExp column = request.getHeadExp(i);
13   reply.addHeadExp(column);
14
15   execDesc.setOutputColumn(i-1, nick.getInfo().getColumn
16 (column.getColumnName()).getColumnID());
17 }
```

get i-th column from SELECT

add columns to reply

map columns

array index of columns from SELECT

index of column from nickname

```
18 reply.setExecDesc(execDesc);
19 return reply;
```

add execution descriptor to repl

Example implementation

- **Class** UnfencedFileNickname
  - UnfencedFileNickname() - constructor
  - verifyMyRegisterNicknameInfo() - verifying data about nickname, only when nickname uses OPTIONS (FILE_PAHT in our example)

```
1  public UnfencedFileNickname(String schema, String name,
2   UnfencedGenericServer server)
3  {
4    super(schema, name, server);
5  }

6  protected NicknameInfo verifyMyRegisterNicknameInfo(NicknameInfo
7   nicknameInfo)
8  {
9    return null;
10 }
```

the body could implement algorithm for verifying OPTIONS, this implementa does not verify it, but the method must implemented

Example implementation

- **Class** FencedFileWrapper
  - identical as UnfencedFileWrapper → change names of classes, variables, and methods
- **Class** FencedFileServer
  - FencedFileServer() and createNickname() are identical as in class UnfencedFileServer → change names of classes, variables, and methods
  - createRemoteConnection() - creates object of remote connection → for accessing a file this method is not used

Example implementation

```
import com.ibm.db2.wrapper.*;

public class FencedFileServer extends FencedGenericServer
{

  public FencedFileServer(String serverName, FencedFileWrapper
wrapper)
  {
    super(serverName, wrapper);
  }

  public Nickname createNickname(String schemaName, String
nickname)
  {
    return new FencedFileNickname(schemaName, nickname, this);
  }

  public RemoteConnection createRemoteConnection(
FencedRemoteUser user, int kind, long id) throws Exception
  {
    return new FileConnection(this, user, kind, id);
  }
}
```

user mapping object

ID of RemoteConnection objec

transaction support/no support

Example implementation

- **Class** FencedFileNickname
  - identical as UnfencedFileNickname → change names of classes, variables, and methods

```
import com.ibm.db2.wrapper.*;

public class FencedFileNickname extends FencedGenericNickname
{

  public FencedFileNickname(String schema, String name,
FencedFileServer server)
  {
    super(schema, name, server);
  }

  protected NicknameInfo verifyMyRegisterNicknameInfo(
NicknameInfo nicknameInfo) throws WrapperException
  {
    return null;
  }

}
```

Example implementation

- **Class** FileQuery
  - responsible for executing a query on a remote data source
  - FileQuery() - constructor
  - additional properties
    - execDesc - execution descriptor
    - fileReader - data buffer with query results

```
1  public FileQuery(RemoteConnection activeConnection, long id)
2  {
3    super(activeConnection, id);
4  }
5  private FileExecDesc execDesc = null;
6  private BufferedReader fileReader = null;
```


Class FileQuery

- **Methods open() and close()**

```
7  public void open() throws Exception
8  {
9    execDesc = (FileExecDesc) getExecDesc();
10   fileReader = new BufferedReader(new FileReader(
11 execDesc.getFilePath()));
12
13   setStatus(OPEN);
14 }

15 public void close(short status) throws Exception
16 {
17   if(fileReader != null)
18   {
19     fileReader.close();
20     fileReader = null;
21   }
22 }
```

get execution descriptor objec

create file reader objec

get path to a file

object representing a query is open

Class FileQuery

➲ **Method fetch() - reads data**

```
23  String line = null;
24  line = fileReader.readLine();
25
26  if(line == null)
27  {
28    reportEof();
29    break;
30  }
31  else
32  {
33
34    ...
35
36  }
```

line processed in a file

Class FileQuery

➲ **Method fetch() - the else { } code**

table of strings, value separator

stores output data from wrapper

get the number of columns

```
37  String[] tokens = line.split(";");
38  RuntimeDataList outputDataList = getOutputData();
39  int valuesCount = outputDataList.getNumberOfValues();
40
41  for(int i = 0; i < valuesCount; i++)
42  {
43
44    ...
45
46  }
```

get column data

```
47  RuntimeData data = outputDataList.getValue(i);
48  int columnIndex = execDesc.getOutputColumn(i);
49  if(data.getDataType() == RuntimeData.CHAR)
50    data.setString(tokens[columnIndex]);
51  if(data.getDataType() == RuntimeData.VARCHAR)
52    data.setString(tokens[columnIndex]);
53  if(data.getDataType() == RuntimeData.INT)
54    data.setInt(Integer.parseInt(tokens[columnIndex]));
```

match columns from SELECT and pseudonym

Class FileExecDesc

- Execution descriptor - back box used during query execution
  - created by wrapper during query plan execution
  - available in object reply
  - returned to the federated server
  - implementation
    - part 1: constants and variables - store various data used by wrapper
    - part 2: set() methods - set values of variables, used during query plan generation
    - part 3: get() methods - read values of variables

Class FileExecDesc

```
public class FileExecDesc implements java.io.Serializable
{
  public void setOutputColumns(int number)
  {
    _outputColumns = new int[number];
  }
  public void setOutputColumn(int outputIndex, int columnIndex)
  {
    _outputColumns[outputIndex] = columnIndex;
  }
  public void setFilePath(String filePath)
  {
    _filePath = filePath;
  }
  public int getOutputColumn(int index)
  {
    return _outputColumns[index];
  }
  public String getFilePath()
  {
    return _filePath;
  }
  private String _filePath = null;
  private int[] _outputColumns = null;
}
```

object of this class will be seriali
→ transforming the object into th
stream of bytes

mapping columns in SELECT
in nickname

store the path to the file

Class FileConnection

- Represents connection between a wrapper and an external data source (file)
  - must return object of class RemoteQuery

```
import com.ibm.db2.wrapper.*;

public class FileConnection extends RemoteConnection
{

  public FileConnection(FencedServer remoteServer,
FencedRemoteUser remoteUser, int connectionKind, long id)
  {
    super(remoteServer, remoteUser, connectionKind, id);
  }

  public RemoteQuery createRemoteQuery(long id) throws Exception
  {
    return new FileQuery(this, id);
  }

}
```


Adding data sources

- New data sources (wrappers) must be added into the federated server
  - to add a custom wrapper one must create XML configuration files for the wrapper → use a graphical application (wizard)
    - **AIX**: `/usr/opt/db2_08_01/lib/db2wrapperconfig`
    - **HP/Sun/Linux**: `/opt/IBM/db2/V8.1/lib/db2wrapperconfig`
    - **Windows**: `%DB2PATH%\bin\db2wrapperconfig.bat`
  - wrapper data that must be provided include among others
    - wrapper name, wrapper's data source, supported operating systems
    - the wrapper library or class name to use for each operating system
    - wrapper options that users must provide and which of them are optional
    - whether the CREATE SERVER statement requires a user ID
    - server options, user mapping options, nickname options, column options
    - DB2 data types that the wrapper supports
    - environment variables that users must provide and the location where they will be set

Adding data sources

- The wizard creates two files:
  - XML configuration file with the properties specified (cf. previous slide); the file name is the wrapper name (extension .xml)
  - properties file, which contains the literal text strings that will be displayed in the DB2 Control Center for the wrapper and its options (extension .properties)

Compiling wrapper

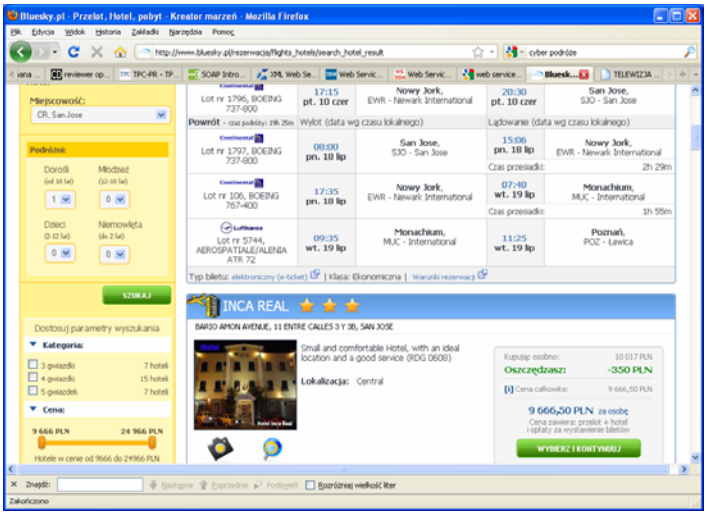
```
1  db2 drop wrapper file_wrapper
2  db2 terminate
3  db2stop
4  cd "%DB2PATH%"\java\jdk\bin
5  del *.java
6  del *.class
7  cd "%NetBeansProjects_PATH%"\FileWrapperProject\src
8
9  copy *.java "%DB2PATH%"\java\jdk\bin
10 cd "%DB2PATH%"\java\jdk\bin
11
12 javac -cp db2qgjava.jar UnfencedFileWrapper.java
13 UnfencedFileServer.java UnfencedFileNickname.java
14 FencedFileWrapper.java FencedFileServer.java
15 FencedFileNickname.java FileConnection.java FileQuery.java
16 FileExecDesc.java
17
18 copy *.class "%DB2PATH%"\FUNCTION
19 cd "%DB2PATH%"\BIN
20
21 db2start
22 db2 connect to sample
23 db2 create wrapper file_wrapper library 'filelib\db2qgjava.dll'
24 options(UNFENCED_WRAPPER_CLASS 'UnfencedFileWrapper',
25 FENCED_WRAPPER_CLASS 'FencedFileWrapper')
26
27 db2 create server file_server wrapper file_wrapper
28
29 db2 create nickname file_a(number integer, text char(20))
30 for server file_server options(FILE_PATH 'C:\sample.txt')
```

- Check if Java API library (db2qgjava.jar) for wrappers exists in "%DB2PATH%"\java\jdk\bin

db2qgjava.dll for Windows
libdb2qgjava.a for AIX
libdb2qgjava.so for other UNIX operating syste

copy the compiled class files

Real system



Real system

- Trip, itinerary, holiday planner
- On-line integration of thousands of data sources
  - airlines
  - hotels
  - car rental agencies
  - travel agencies
  - Google maps

Problems

- Time zones (local time at source != local time at destination), working hours of services
- Geo object localization
  - different names for the same object → use geo coordinates
- Data visualization
  - important data vs. what user is interested in seeing
- Managing payments
- Efficiency → response waiting time < 60 sec
- Client is usually interested in the lowest prices
  - cost of a trip: waiting time, the number of hops, price, airline reputation, ...
- Client profiles
  - recommender system
  - business intelligence - decision support

Problems

- Binding cities with airports
  - the closest distance? → islands?
- Daily hundreds of new hotels (destinations) may appear
- The same destinations, flights, cities, ..., may be fetched from multiple sites (different names) → duplicate elimination
- Different cities with the same names
  - San Jose in Costa Rica, USA, Mexico, Bolivia, Philippines

Technology

- WEB Services
- XML data, databases
- AJAX + JavaScript
- Data dictionaries
- Google Maps API
- Manual data integration