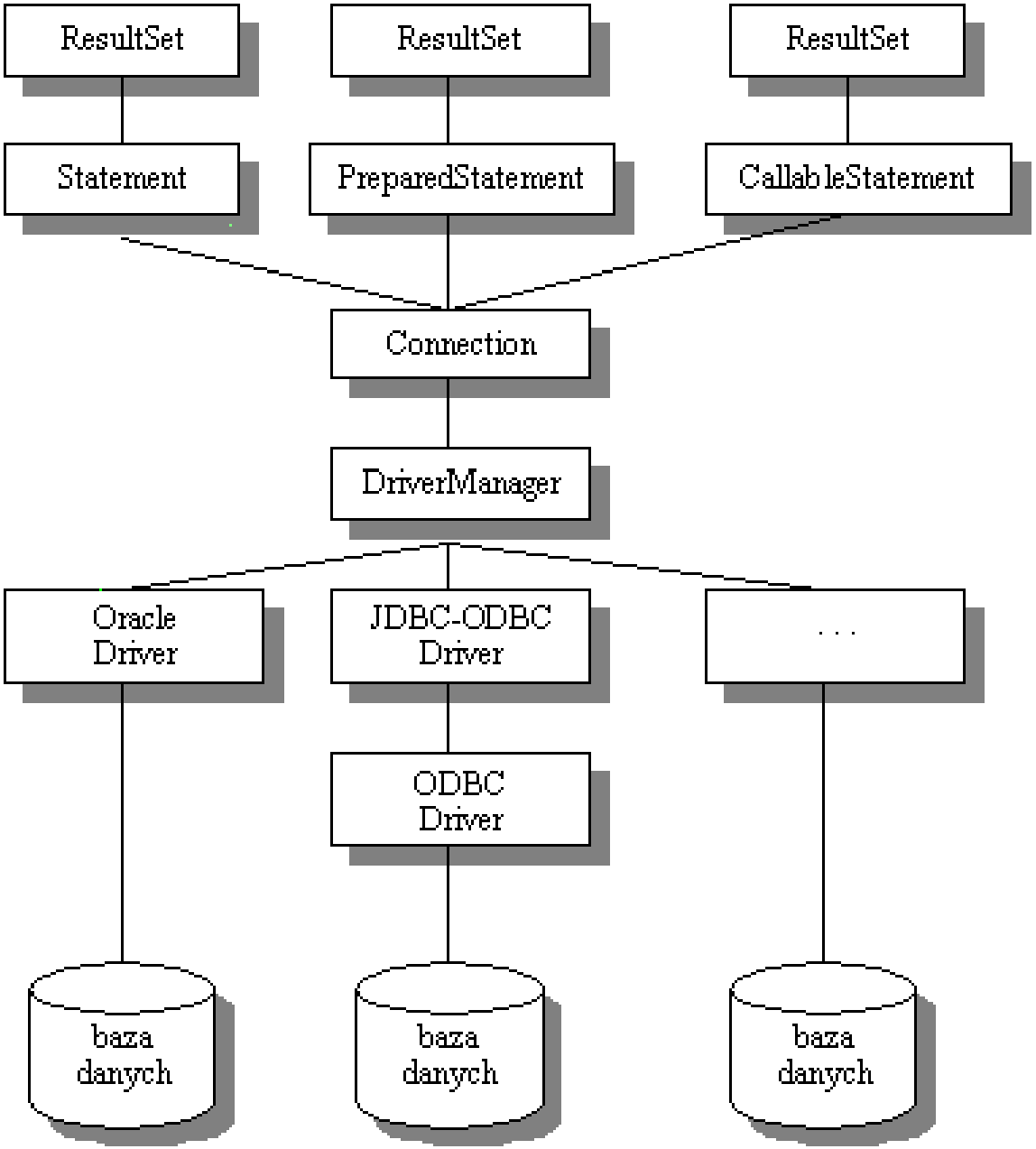
Wprowadzenie do JDBC z wykorzystaniem bazy H2

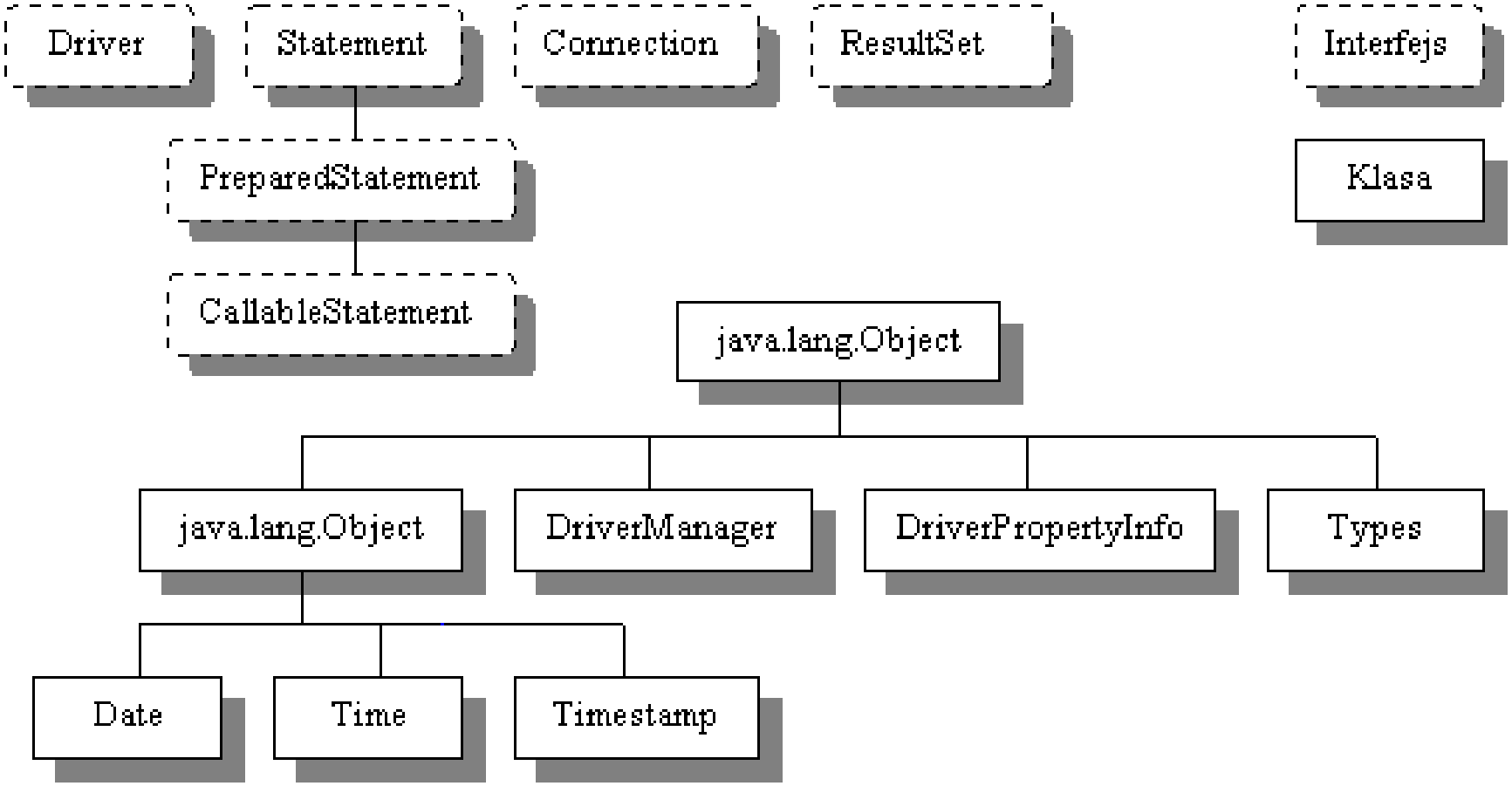
JDBC (ang. Java DataBase Connectivity).

- Biblioteka stanowiąca interfejs umożliwiający aplikacjom napisanym w języku Java porozumiewać się z bazami danych za pomocą języka SQL.
- Interfejs ten operuje na poziomie typowego dla baz danych języka SQL (ang. Structured Query Language) i pozwala w prosty sposób konstruować zapytania do bazy danych oraz wykorzystywać ich wyniki w kodzie Javy.
- Środowisko Java zawiera API JDBC, natomiast użytkownik musi dodatkowo uzyskać sterownik JDBC dedykowany do swojej bazy danych.

Struktura JDBC



Klasy i interfejsy JDBC



JDBC – ogólna zasada wykorzystania

1. Uzyskanie połączenia ze źródłem danych
2. Wykonywanie operacji na bazie danych (przekazywanie poleceń w postaci kwerend SQL)
3. Przetwarzanie danych pobranych z bazy w efekcie wykonania kwerend

JDBC – ogólna zasada wykorzystania

```
// Łączenie z bazą:
    Connection con = DriverManager.getConnection(
                            "jdbc:myDriver:myDatabase",
                            username,
                            password);
// Wykonywanie kwerendy:
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(
                        "SELECT a, b, c FROM Table1");
// Przetwarzanie pobranych danych
    while (rs.next()) {
        int x = rs.getInt("a");
        String s = rs.getString("b");
        float f = rs.getFloat("c");
    }
```

Język SQL

Język SQL pozwala na zarządzanie bazą danych, między innymi poprzez:

- tworzenie nowych tabel
- dodawanie danych do tabel
- modyfikację danych zawartych w tabelach
- modyfikację struktury tabel
- usuwanie danych z tabel oraz usuwanie całych tabel
- pobieranie danych z tabel

Język SQL – tworzenie przykładowej tabeli

```
DROP TABLE IF EXISTS `waluty`;

CREATE TABLE `waluty` (
  `Id` int(6) unsigned NOT NULL auto_increment,
  `data` date default NULL,
  `USD` float default NULL,
  `EUR` float default NULL,
  `GBP` float default NULL,
  PRIMARY KEY  (`Id`)
) ;
```

Język SQL – tworzenie przykładowej tabeli

```
DROP TABLE IF EXISTS `Osoby`;

CREATE TABLE `Osoby` (
  `Id` int(6) unsigned NOT NULL auto_increment,
  `Imie` varchar(20) NOT NULL,
  `Nazwisko` varchar(40)NOT NULL,
  `data_urodzenia` date default NULL,
   PRIMARY KEY  (`Id`)
) ;
```

Język SQL – dodawanie danych

```
INSERT INTO `waluty` VALUES (1,'2000-01-03‘
,4.1171,4.165,6.6576);

INSERT INTO `waluty` (`Id`,`data`,`USD`,`EUR`,`GBP`)
VALUES (2,'2000-01-04',4.1267,4.2232,6.7571);

INSERT INTO `waluty` (`data`,`USD`,`EUR`,`GBP`) VALUES
('2000-01-05',4.1299,4.2797,6.776);

INSERT INTO waluty (data,USD) VALUES ('2000-01-06‘
,4.1129);
```

Język SQL – komenda SELECT

- **SELECT column1, column2, ... FROM table_name [WHERE condition] [ORDER BY o_column]**

  gdzie:
- **column1, column2,...** to nazwy kolumn, których zawartość ma być wyświetlona jako wynik zapytania.
- **table_name** to nazwa tabeli, z której pobierane są dane
- **condition** to warunek jaki muszą spełniać wiersze tabeli, aby zostać wyświetlone
- **o_column** to kolumna, wg. której mają być posortowane wyniki zapytania

Język SQL – komenda SELECT

Chcąc pobrać wszystkie kolumny z tableli zamiast wypisywać ich nazwy można użyć ***** (gwiazdka), np komenda:

SELECT * FROM osoby

zwróci wszystkie kolumny z tabeli osoby.

Parametry **[WHERE condition]** oraz **[ORDER BY o_column]** są opcjonalne.

Język SQL – komenda SELECT

Załóżmy, że tabela osoby zawiera trzy kolumny: *imie*, *nazwisko* oraz *wiek.* W takim przypadku komenda:

SELECT imie FROM osoby WHERE wiek = 18
zwróci imiona osób, które mają 18 lat.

SELECT nazwisko, imie FROM osoby ORDER BY wiek
zwróci nazwiska i imiona wszystkich osób z tabeli, posortowane według wieku.

SELECT nazwisko, imie FROM osoby ORDER BY nazwisko DESC
zwróci nazwiska i imiona wszystkich osób z tabeli, posortowane alfabetycznie według nazwisk, w odwróconej kolejności od Z do A (modyfikator **DESC** sprawia, że kolejność sortowania jest odwócona)

Język SQL – komenda SELECT

W warunkach występujących po **WHERE** można stosować:

- operatory porównania **=**, **>=**, **>** itp.
- operatory logiczne **AND** i **OR**
- do porównywania łańcuchów można stosować polecenie **LIKE** akceptujące wzorce napisów, w których znak ’_’ zastępuje dowolną literę, zaś znak ’%’ zastępuje dowolny ciąg znaków

  SELECT * FROM osoby WHERE nazwisko LIKE 'A%'
  zwróci wszystkie dane osób, których nazwisko rozpoczyna się na literę A

  W "warunkach" można używać również operatora **BETWEEN**, który pozwala wybierać dane z określonego zakresu, np.:
  SELECT * FROM osoby WHERE wiek BETWEEN 18 AND 30
  zwróci rekordy z osobami o wieku od 18 do 30 lat. Jest to równoważne z zapisem:
  SELECT * FROM osoby WHERE wiek > 18 AND wiek < 30

Operacje na bazie z poziomu JAVY:

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Tab1");
```

Trzy podstawowe metody wywołania kwerend:

```
stmt.executeQuery(...),
stmt.executeUpdate(...),
stmt.execute(...)
```

Różnią się to sposobem przekazywania wyniku. W pierwszym przypadku otrzymamy wynik w postaci tabeli (wynik zapytania). W drugim przypadku otrzymamy ilość pomyślnie przeprowadzonych zmian w bazie.
Trzecia opcja jest uogólnieniem dwóch poprzednich. Zwraca wynik, z tym, że to użytkownik musi sprawdzić jaki wynik otrzymał. executeQuery() stosuje się zazwyczaj do zapytań typu SELECT...., a executeUpdate() do zapytań typu CREATE TABLE... lub INSERT...

ResultSet

- Wyniki wykonanej kwerendy dostępne są w obiekcie klasy **ResultSet**.

- ResultSet zawiera linie (rekordy) z wyniku wykonanej operacji. Początkowo jest ustawiony przed pierwszą linią. Aby przesunąć go na następna linijkę i przy okazji sprawdzić, czy jest więcej wyników, wykorzystuje się metodę next().

ResultSet, ResultSetMetaData

```
ResultSet rs = statement.getResultSet();

ResultSetMetaData md  = rs.getMetaData();

//wyswietlanie nazw kolumn:
for (int ii = 1; ii <= md.getColumnCount(); ii++){
        System.out.print(md.getColumnName(ii)+ " | ");
}
System.out.println();
// wyswietlanie poszczegolnych wierszy
while (rs.next()) {
        for (int ii = 1; ii <= md.getColumnCount(); ii++){
          System.out.print( rs.getObject(ii) + " | ");
        }
System.out.println();
}
```

Prepared Statement

Zamiast interfejsu Statement, w którym wykorzystujemy dokładną (statyczną) treść kodu SQL, można wykorzystać typ PreparedStatement, który pozwala na parametryzację.

Użycie prekompilowanych komend jest wygodne, jeśli wywołujemy wiele podobnych komend na różnych argumentach.

```
PreparedStatement prep = conn.prepareStatement
   ("INSERT into waluty(data, usd, eur, gbp) values (?, ?, ?, ?)");
prep.setString(1, "2000-01-03");
prep.setString(2, "4.1171");
prep.setString(3, "4.165");
prep.setString(4, "6.6576");
prep.executeUpdate();
```

Co jest równoważne:

```
statement.executeUpdate("INSERT INTO waluty (Id,data,USD,EUR,GBP)
                VALUES (1,'2000-01-03',4.1171,4.165,6.6576);");
```

Dodawanie bibliotek do projektu Eclipse - uzupełnienie

Dodawanie bibliotek poprzez „Add External JAR…” (jak pokazano na poprzednim wykładzie) może być niewygodne przy przenoszeniu eksportowaniu/importowaniu projektów, gdyż każdorazowo trzeba kopiować/ściągać potrzebne piłki JAR oraz konfigurować „Build Path”.

To ograniczenie łatwo ominąć poprzez umieszczanie bibliotek bezpośrednio w projekcie, wówczas przy eksporcie będą one dodawane do tworzonego archiwum zaś „Build Path” będzie aktualny po imporcie na inny komputer.

Dodawanie bibliotek do projektu Eclipse

Jak to zrobić?

1. Utworzyć w katalogu z projektem podkatalog, w którym będą umieszczone pliki JAR z bibliotekami (np. o nazwie „lib”).
2. Skopiować wymagane pliki JAR do katalogu

Tworzenie katalogu i kopiowanie można wykonać z poziomu Eclipse (File->New->Folder, Copy->Paste), lub operując bezpośrednio na katalogu workspace na dysku, ale wówczas po skopiowaniu plików projekt należy odświeżyć (Refresh), żeby pliki były widzialne w Eclipse

Dodawanie bibliotek do projektu Eclipse

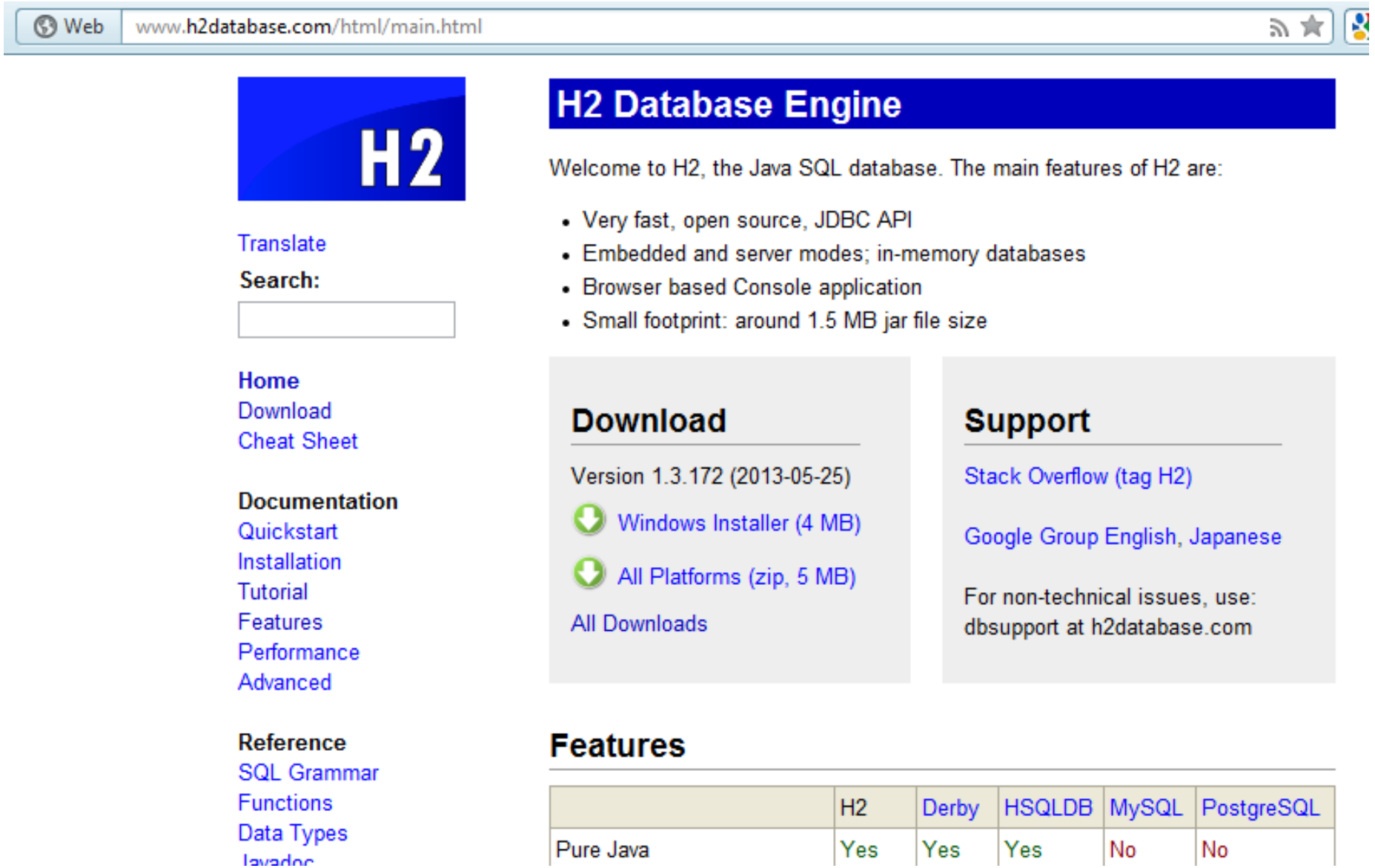
3. Następnie należy dodać biblioteki do „Build Path” projektu:

- 1 metoda: rozwinięcie katalogu lib i kliknięcie prawym klawiszem na wybranym pliku JAR, a potem wybranie opcji „Build Path-> Add to Build Path”

- 2 metoda: konfiguracja „Build Path” projektu i dodanie biblioteki poprzez „Add JAR” (nie jak poprzednio Add External JAR) i wskazanie lokalizacji w podkatalogu projektu.

Przykład – utworzenie nowego projektu zawierającego bibliotekę bazy H2.

Baza danych H2 http://www.h2database.com



Baza danych H2

Features

	H2	Derby	HSQLDB	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (jar/dll size)	~1 MB	~2 MB	~1 MB	~4 MB	~6 MB

See also the detailed comparison.

Tworzenie bazy z poziomu JAVY

TworzenieBazy.java

(tworzenie nowego projektu z osadzoną biblioteką)

```
conn = DriverManager.getConnection("jdbc:h2:nazwabazy", "sa", "");
// domyslnie nazwa uzytkownika to "sa" a dostep jest bez hasla - ""

// Proba podlaczenia do bazy H2, ktora nie istnieje
// domyslnie powoduje utworzenie nowej instancji pustej bazy
// (w postaci pliku z rozszerzeniem *.db, np. nazwabazy.h2.db)

// Dymyslne tworzenie pustej bazy danych czasem moze generowac bledy,
// dlatego mozliwe jest wylaczenie domyslnego tworzenia pustej bazy
// conn =
DriverManager.getConnection("jdbc:h2:nazwabazy5;IFEXISTS=TRUE", "sa",
"");
```

TworzenieTabeli.java

```
Statement statement = conn.createStatement();
// Usuwanie tabeli jeśli już istnieje - kolejne uruchomienie
// przykladu nie wygeneruje bledu:
statement.executeUpdate("DROP TABLE IF EXISTS `waluty`;");

// Tworzenie tabeli o okreslonej strukturze danych
statement.executeUpdate("CREATE TABLE `waluty` ("+
  "`Id` int(6) unsigned NOT NULL auto_increment,"+
  "`data` date default NULL,"+
  "`USD` float default NULL,"+
  "`EUR` float default NULL,"+
  "`GBP` float default NULL,"+
  "PRIMARY KEY  (`Id`)"+
") ;");
```

TworzenieTabeli.java

Dodawanie danych do tabeli:

```
 statement.executeUpdate("INSERT INTO `waluty`
(`Id`,`data`,`USD`,`EUR`,`GBP`) VALUES (1,'2000-01-
03',4.1171,4.165,6.6576);");

// Przykladowe rownowazne polecenia SQL:
statement.executeUpdateUpdate("INSERT INTO `waluty` VALUES (1,'2000-
01-03',4.1171,4.165,6.6576);");

statement.executeUpdate("INSERT INTO `waluty`
(`data`,`USD`,`EUR`,`GBP`) VALUES ('2000-01-03‘ , 4.1171 , 4.165 ,
6.6576 );");

statement.executeUpdate("INSERT INTO waluty VALUES (1,'2000-01-
03',4.1171,4.165,6.6576);");

statement.executeUpdate("INSERT INTO waluty (Id,data,USD,EUR,GBP)
VALUES (1,'2000-01-03',4.1171,4.165,6.6576);");
```

TworzenieTabeli.java

Dodawanie danych do tabeli:

```
// Rownowazne wywolanie z wykorzystaniem PreparedStatement

   /*  PreparedStatement prep = conn.prepareStatement("INSERT
into waluty(data, usd, eur, gbp) values (?, ?, ?, ?)");
 prep.setString(1, "2000-01-03");
 prep.setString(2, "4.1171");
 prep.setString(3, "4.165");
 prep.setString(4, "6.6576");
 prep.executeUpdate();*/
```

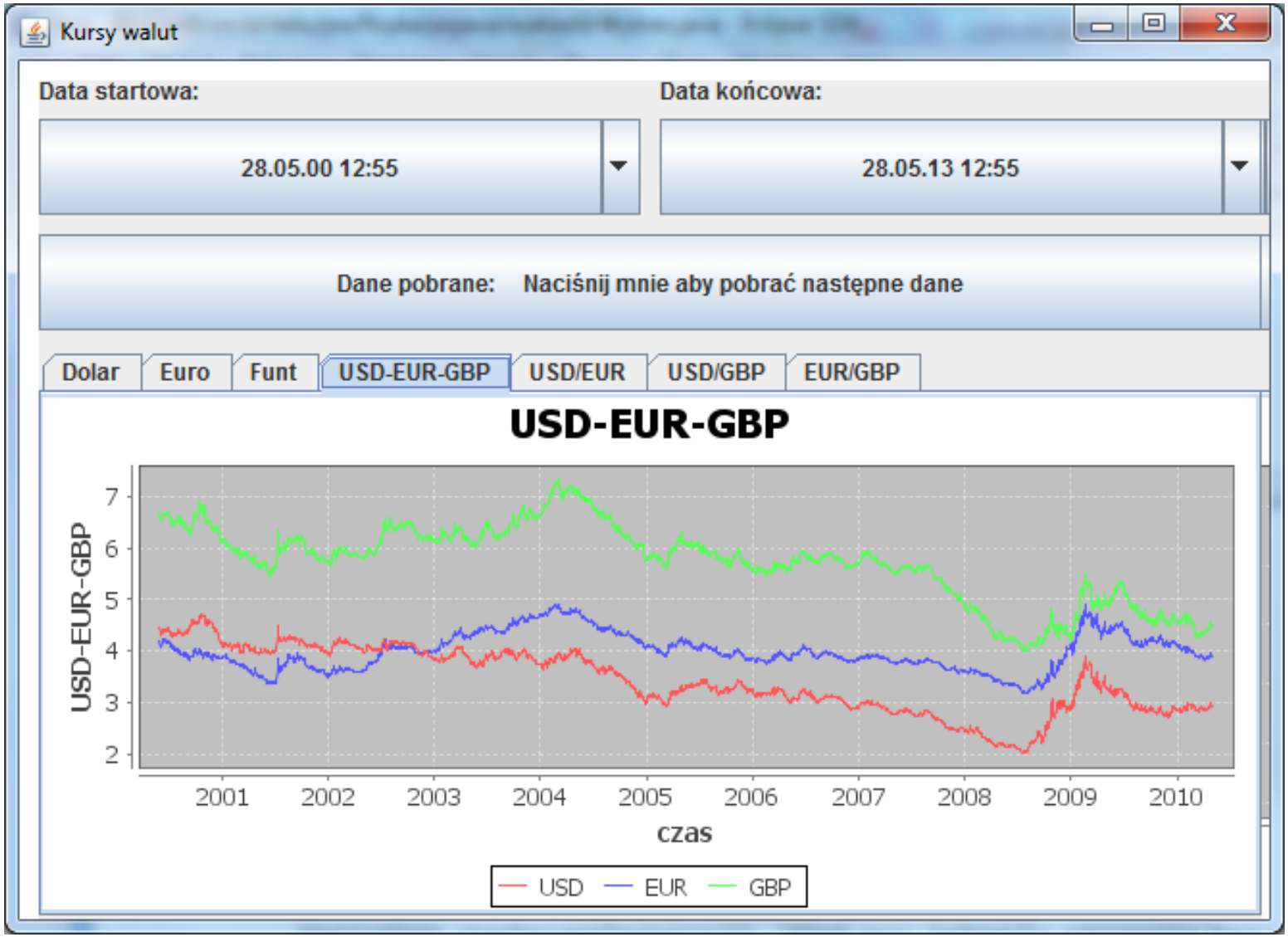
WypisywanieTabeli.java

```
//Wyswietlanie calej tabeli:
statement.execute("SELECT * FROM waluty");

//Ograniczenie do 10 pierwszych rekordow
statement.execute("SELECT * FROM waluty limit 10" );

// Przykladowe kwerendy z dodatkwoym warunkiem:
statement.execute("SELECT * FROM waluty where data <
                    '2001-03-27'");
statement.execute("SELECT * FROM waluty where usd > 4.50");
statement.execute("SELECT * FROM waluty where usd > 4.50 and
                    eur < 3.83");
```

WypisywanieTabeli.java

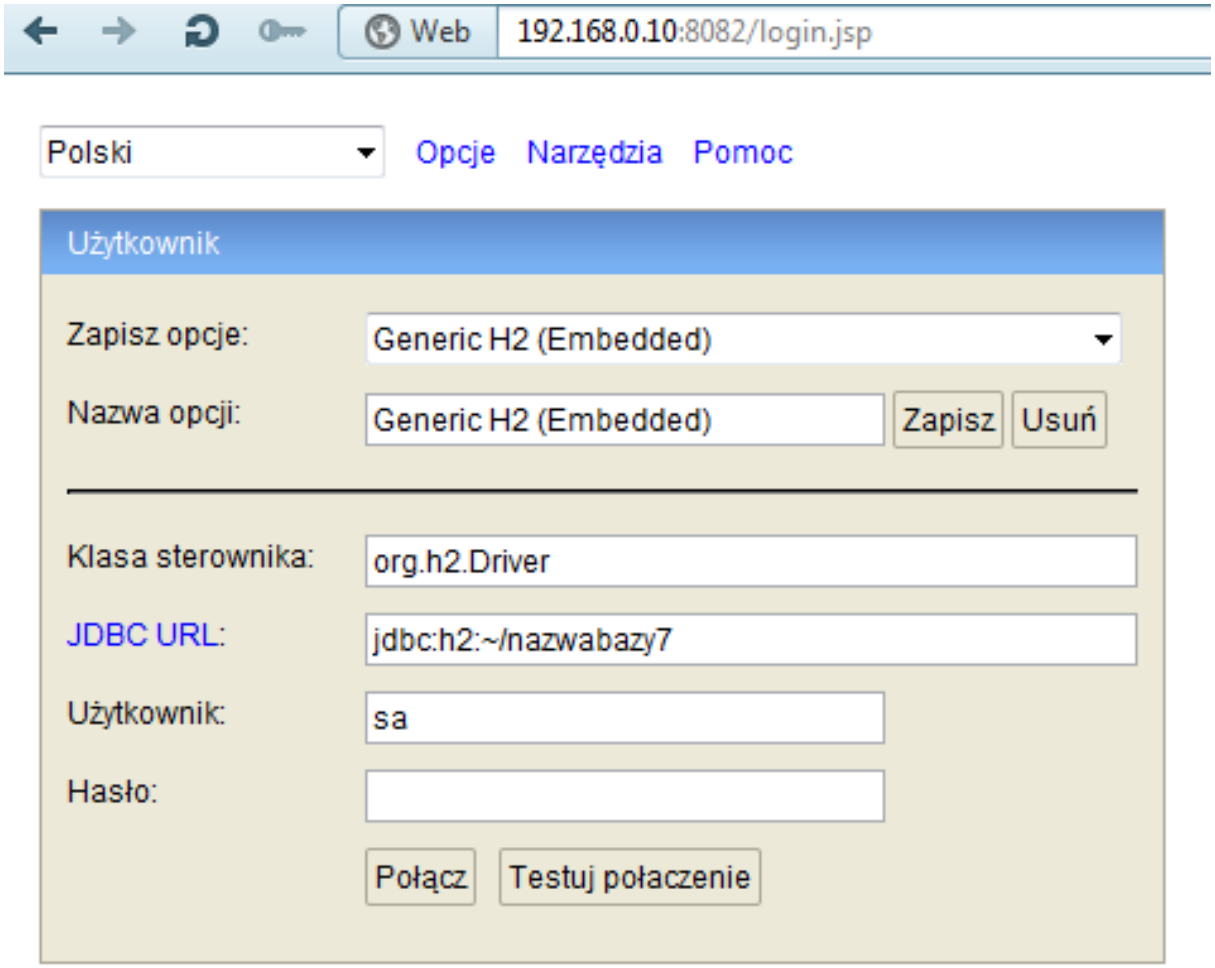
```
ResultSet rs = statement.getResultSet();

ResultSetMetaData md  = rs.getMetaData();


for (int ii = 1; ii <= md.getColumnCount(); ii++){
        System.out.print(md.getColumnName(ii)+ " | ");

}
System.out.println();

while (rs.next()) {
for (int ii = 1; ii <= md.getColumnCount(); ii++){
        System.out.print( rs.getObject(ii) + " | ");
}
System.out.println();
```

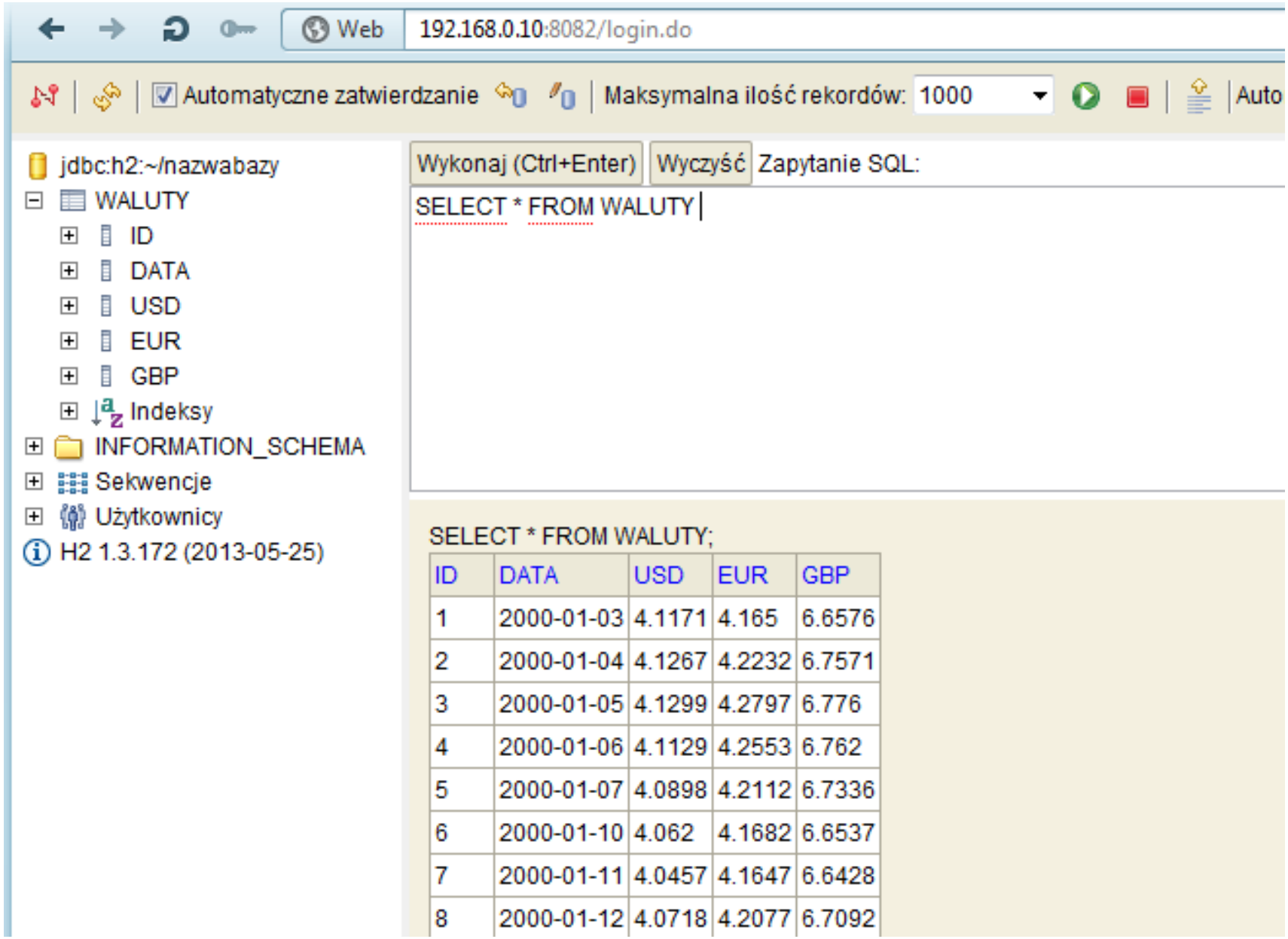
Wykres.java



Tworzenie pliku JAR – konieczny dostęp do pliku z bazą danych

- H2 umożliwia łączenie się z bazą z panelu kontrolnego z poziomu przeglądarki (plik z bazą musi być w katalogu domowym użytkownika)





demonstracja działania na przykładzie wygenerowanej bazy