Programowanie komputerów

Wykład 10: „Dostęp do baz danych z poziomu aplikacji Java”

(JDBC Database Access)

Relacyjne bazy danych

- Baza danych to zbiór danych powiązanych ze sobą pewnymi relacjami.
- System bazodanowy (ang. Database Management System) zarządza przechowywanymi danymi i użytkownikami tych danych.
- System zarządzania relacyjnymi bazami danych (ang. Relational Database Management System) to system bazodanowy, w którym organizacja danych bazuje na pojęciu relacji z matematycznej teorii mnogości.

Relacyjne bazy danych

Dane w bazie są zorganizowane w postaci tabel, widoków i schematów.

- Tabela (ang. table) składa się z wierszy (rekordów) i kolumn (pól).
- Widok (ang. view) to zakres widocznych wierszy i kolumn z jednej lub kilku tabel.
  - Tabela zajmuje fizyczne miejsce w pamięci, a widok to jakby tabela wirtualna.
- Schemat (ang. scheme) to logiczne uporządkowanie tabel i widoków.
  - Baza danych jest więc zestawem schematów, tabel i widoków, pomiędzy którymi zachodzą pewne relacje.

Relacyjne bazy danych

System zarządzania relacyjną bazą danych zapewnia spójność danych poprzez:

- Więzy jednoznaczności (ang. unique constraint) – klucz pierwotny (ang. primary key);
- Spójność referencyjną (ang. referential constraint) – klucze obce (ang. foreign keys);
- Warunki na wartości kolumn (ang. check constraint) – zawężenia wartości pól;
- Wyzwalacze (ang. triggers) – procedury automatycznie uruchamiane w procesie modyfikacji danych w bazie.

Relacyjne bazy danych

- Indeksy to dodatkowa struktura związana z tabelą, która przyspiesza dostęp do rekordów.
  - Indeksy, podobnie jak klucze, mogą odnosić się do jednej lub kilku kolumn w tabeli. Indeksy powodują nieznaczny spadek wydajności maszyny bazodanowej.
- Klastry, podobnie jak indeksy, przyspieszają dostęp do danych.
  - Klastry to sposób przechowywania powiązanych ze sobą danych w tym samym miejscu na dysku. Klastry powodują znaczny spadek wydajności maszyny bazodanowej w przypadku modyfikacji danych.

Relacyjne bazy danych

- Ochrona danych (ang. security) to aspekty związane z dostępem do danych przez poszczególnych użytkowników. Jest ona realizowana poprzez:
  - Przywileje (ang. privileges) można użytkownikowi nadawać albo odbierać; przywileje dotyczą określonych operacji na rzecz określonych danych;
  - Role (ang. roles) to zbiory przywilejów; rolę można przypisać do konkretnego użytkownika lub do innej roli.
- Transakcje (ang. transactions) gwarantują integralność danych w bazie. Transakcja to grupa operacji wykonywanych przez maszynę bazodanową, która tworzy logiczną całość (albo wszystkie operacje wykonają się poprawnie albo żadna nie dojdzie do skutku).

Relacyjne bazy danych

- SQL (ang. Structured Query Language) to język, w którym można rozmawiać z systemem bazodanowym.
- SQL nie jest językiem proceduralnym.
- Za pomocą SQLa można:
  - utworzyć bazę danych;
  - zarządzać danymi w bazie (CRUD);
  - pobierać dane z bazy;
  - zarządzać systemem bazodanowym.

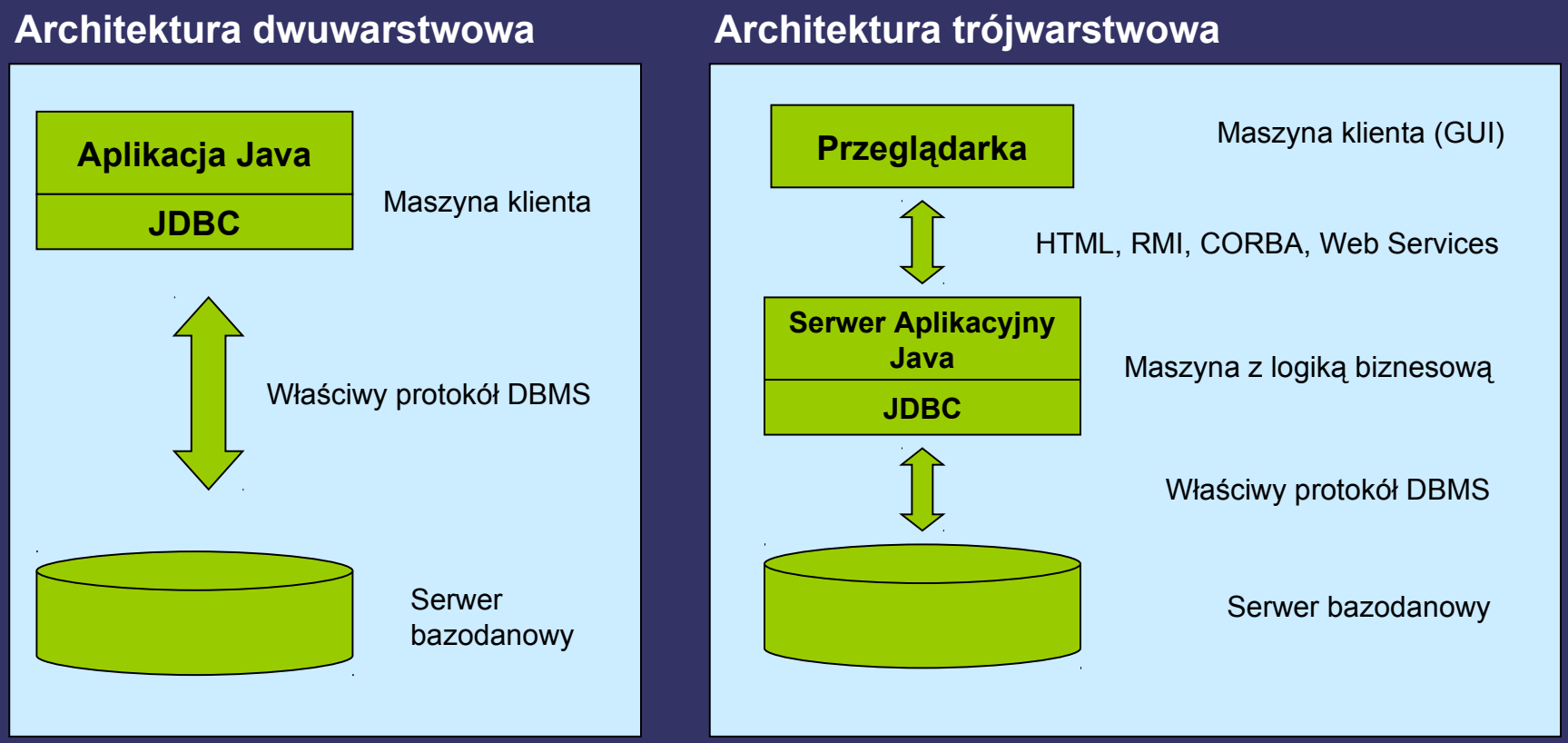
JDBC

- Profesjonalne systemy bazodanowe udostępniają interfejsy programistyczne (API), dzięki którym można uzyskiwać dostęp do baz danych.
  - Interfejsy takie są zdefiniowane dla popularnych języków programowania na różne platformy systemowe.
- Programistyczny interfejs dostępu do baz danych z poziomu Javy (ang. Java Database Connectivity API) nie zależny od:
  - maszyny bazodanowej,
  - platformy sprzętowej,
  - systemu operacyjnego.

JDBC

- JDBC to zestaw klas i interfejsów umieszczonych w pakiecie java.sql, który umożliwia:
  - połączenie z relacyjną bazą danych,
  - wykonywanie instrukcji SQL na bazie,
  - przetwarzanie wyników instrukcji SQL (w tym tabel wynikowych).
- Obecna wersja JDBC to 4.0 API.

Architektura aplikacji bazodanowej



Zmiany w strukturze pozyskania danych realizowane są na poziomie serwera aplikacyjnego bez ingerencji w aplikacje u klienta.

Struktura JDBC

- Sterownik JDBC do określonej bazy danych to zestaw klas, które implementują interfejsy w pakiecie `java.sql`.
- Niezależność bazodanowa osiągnięta dzięki zestawowi interfejsów w pakiecie `java.sql`, implementowanych przez różnych producentów.
- Zadaniem JDBC jest ukrycie przed programistą wszelkich specyficznych właściwości określonej bazy danych.

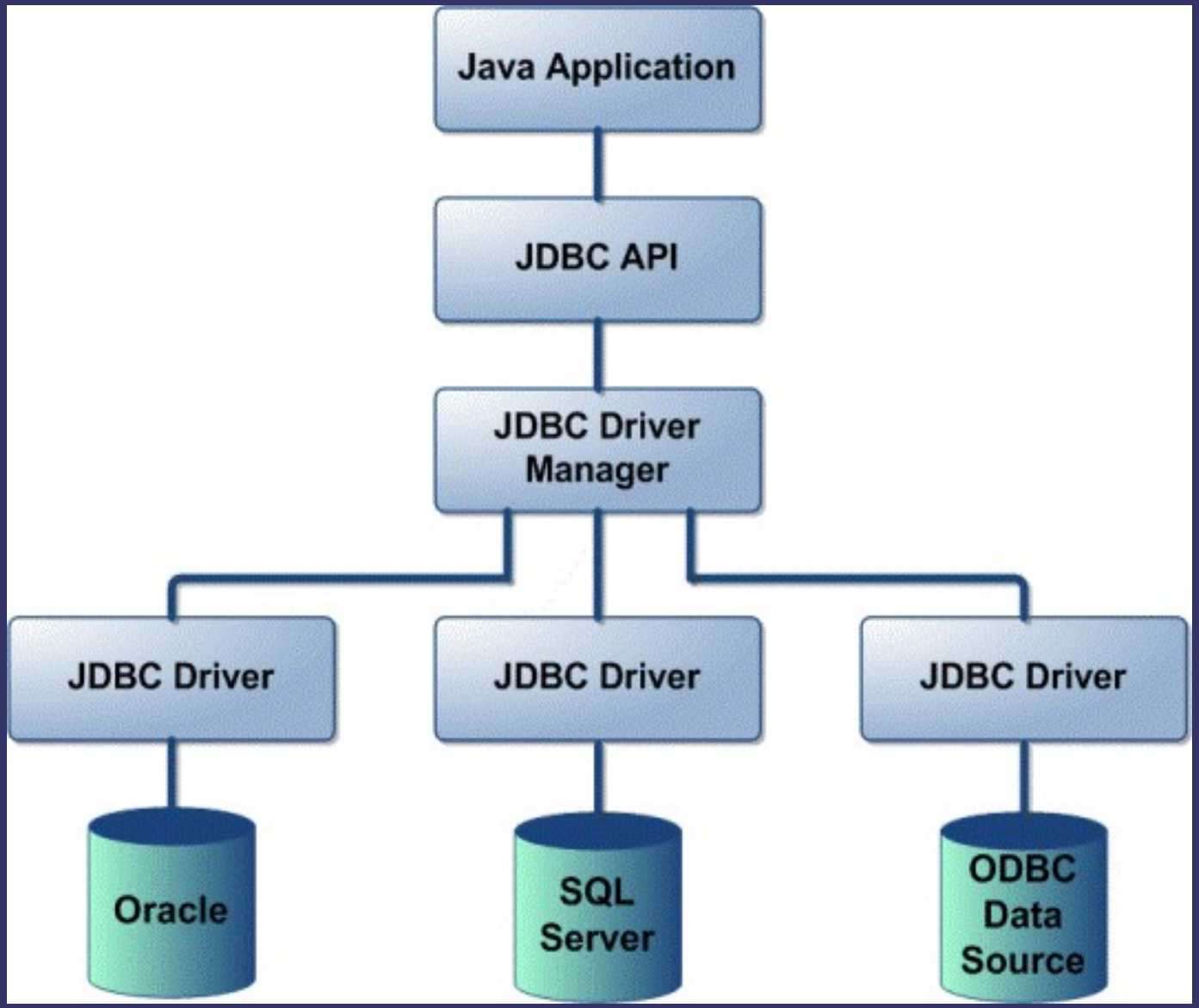
Struktura JDBC

- **`java.sql.Driver`** to interfejs odpowiedzialny za nawiązanie połączenia z bazą danych oraz za tłumaczenie zapytań SQLowych na język określonej bazy.
- **`java.sql.DriverManager`** to klasa, która odpowiada za zarządzanie listą dostępnych sterowników.
- **`java.sql.Connection`** to interfejs, który reprezentuje pojedynczą transakcję bazodanową.
- **`java.sql.Statement`** jest to interfejs reprezentujący zapytanie SQLowe.
- **`java.sql.ResultSet`** to interfejs reprezentujący zbiór rekordów, będących wynikiem zapytania SQLowego; wynik zapytania znajduje się po stronie bazy danych.

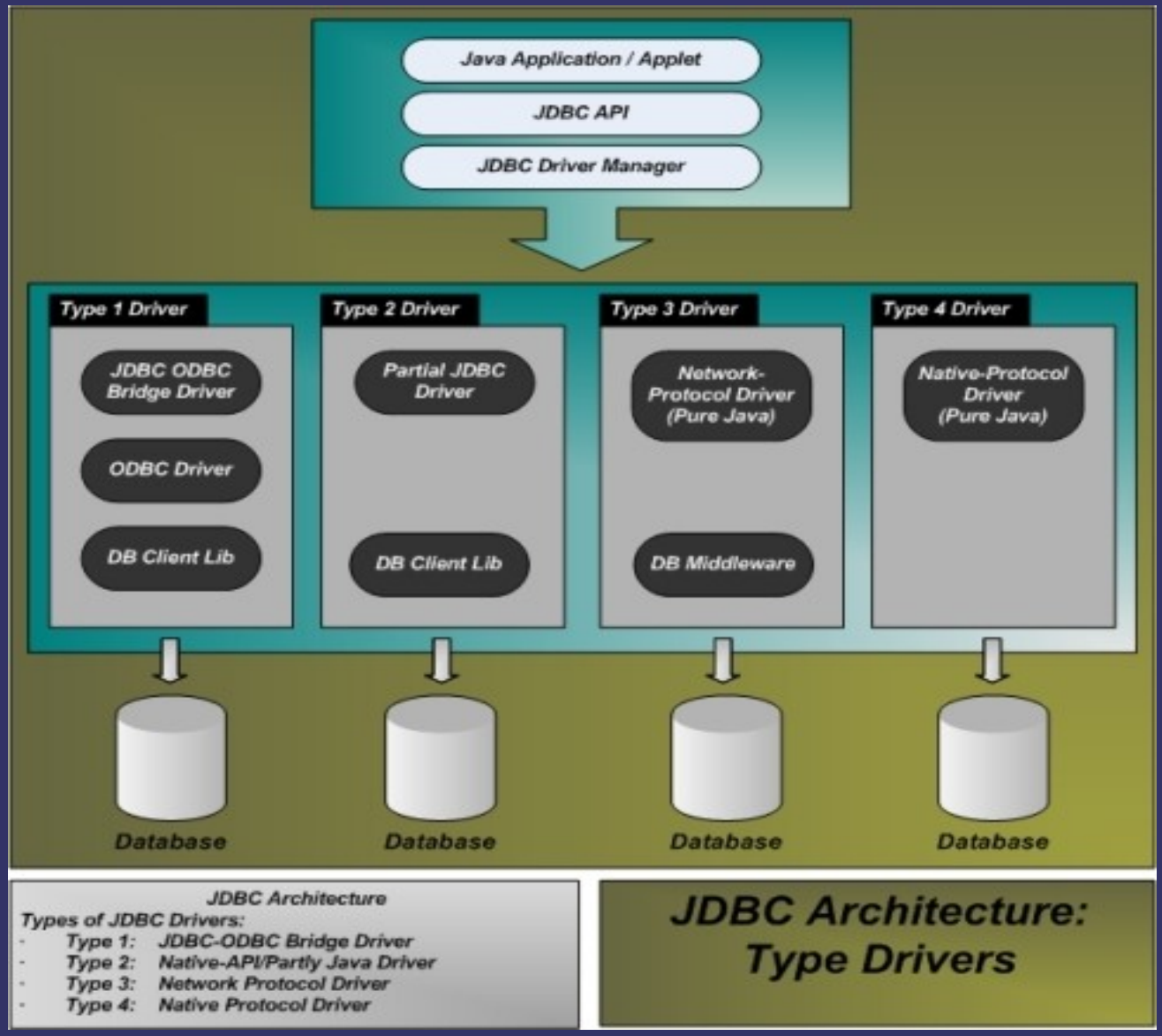
Sterowniki JDBC (1)

- JDBC Bridge Driver – to sterownik pomostowy, czyli wykorzystujący inny ogólny mechanizm dostępu do baz danych (na przykład ODBC).
- Native API Partly Java Driver – to sterowniki napisane w Javie, które wykorzystują do uzyskania dostępu do bazy danych biblioteki napisane w innych językach (na przykład w C++).
- Net-Protocol Pure Java Driver – to sterownik, który uzyskuje dostęp do bazy danych na poziomie serwera, używając do tego celu protokołu sieciowego (dostęp pośredni przez serwer).
- Native-Protocol Pure Java Driver – to sterowniki, które wykorzystują protokoły sieciowe wbudowane w maszyny bazodanowe (dostęp bezpośredni do systemu bazodanowego).

Sterowniki JDBC (2)



Sterowniki JDBC (3)



Łączenie się z bazą danych

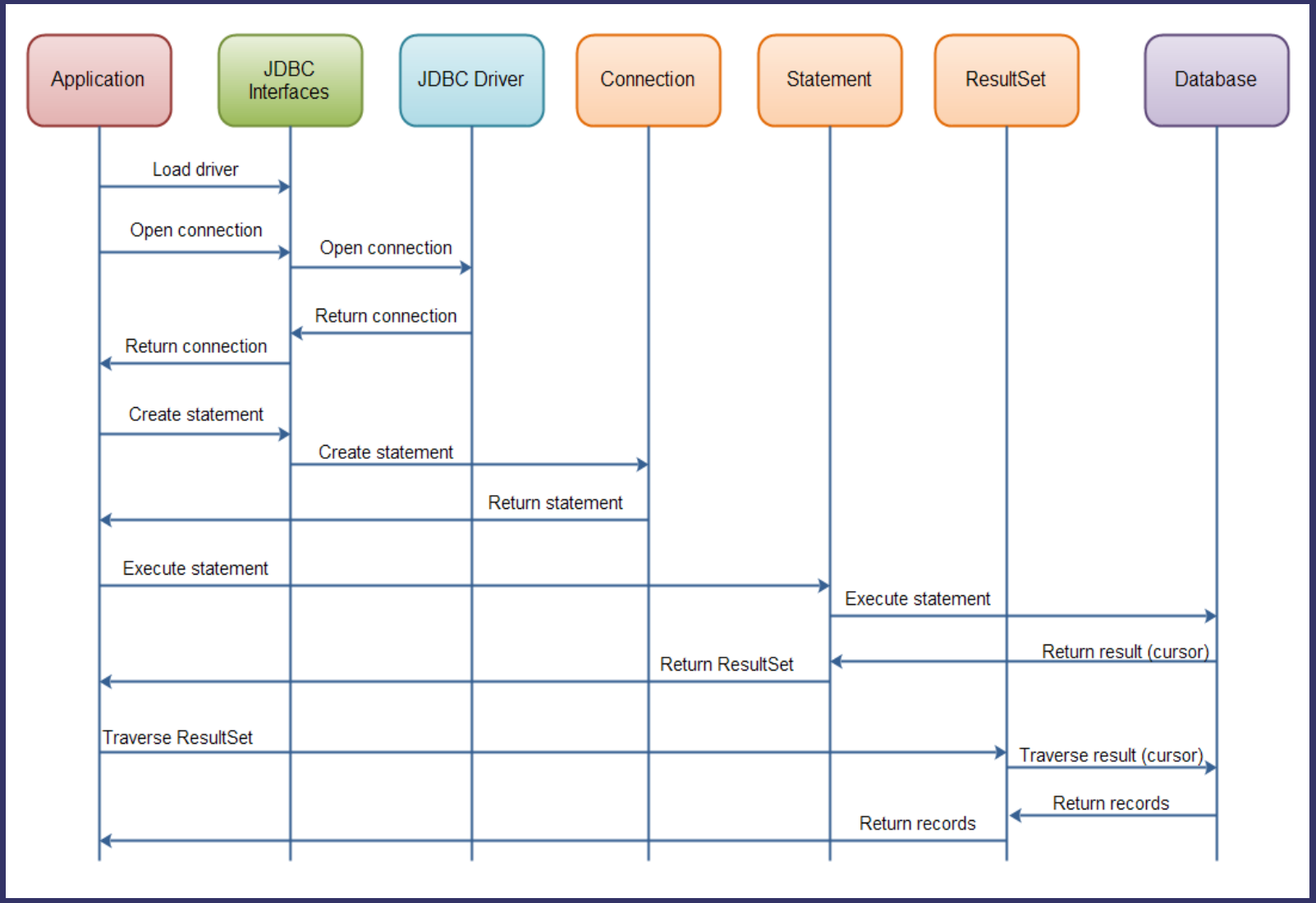
- Połączenie z bazą uzyskujemy za pomocą klasy `DriverManager`:

```
public Connection getConnection (String urlDB)
throws SQLException
{
    Connection conn = null;
    Properties connectionProps = new Properties();
    connectionProps.put("user", this.userName);
    connectionProps.put("password", this.password);
    conn = DriverManager.getConnection(
        urlDB,  connectionProps);
    System.out.println("Connected to database");
    return conn;
}
```

- Połączenie z bazą należy na końcu zamknąć:

```
try {
    if (conn!=null) conn.close();
} catch (SQLException ex) {/*…*/}
```

Interakcje podstawowych części JDBC podczas wykonywania zapytania



URL dla bazy danych

- URL dla bazy danych umożliwia identyfikację bazy, załadowanie odpowiedniego sterownika i uzyskanie połączenia z bazą.
- Składnia URLa używanego w JDBC:
  `jdbc:<subprtokół>:<subnazwa>`
  gdzie `<subprtokół>` to nazwa sterownika lub mechanizmu obsługi połączenia z bazą danych, a `<subnazwa>` to identyfikator bazy danych.

URL dla bazy danych

- Przykład URLa dla połączenia z bazą zarejestrowaną w ODBC:
  `jdbc:odbc:moja_baza`
- Przykłady URL dla różnych systemów bazodanowych:
  - MySQL: `jdbc:mysql://localhost:3306/mydb`
  - MS-SQL:
    `jdbc:sqlserver://127.0.0.1:1433/mydb`
  - PostgreSQL: `jdbc:postgresql://serv:5432/mydb`
  - Mini-SQL: `jdbc:msql://dbserver:1234/mydb`
  - Oracle: `jdbc:oracle:thin:@localhost:1521:orcl`
  - Java-DB: `jdbc:derby:newdb;create=true`

Metainformacje o bazie danych

```
Connection con; // połączenie z bazą
DatabaseMetaData md; // metadane
// należy połączyć się z bazą danych…
md = con.getMetaData();
// …
  md.getDatabaseProductName();
  md.getDatabaseProductVersion();
   md.getDriverName();
   md.getURL();
   md.getUserName();

  md.supportsAlterTableWithAddColumn();
  md.supportsAlterTableWithDropColumn();
  md.supportsBatchUpdates();
  md.supportsPositionedDelete();
  md.supportsPositionedUpdate();
  md.supportsTransactions();

md.supportsResultSetType(ResultSet.TYPE_SCROLL_INSENSITIVE);
md.supportsResultSetType(ResultSet.TYPE_SCROLL_SENSITIVE);
```

Wyjątki SQL

- Jeśli operacja na bazie danych kończy się niepowodzeniem, zgłaszany jest wyjątek SQLException.
- W czasie pojedynczej operacji na bazie może zostać zgłoszonych kilka wyjątków SQLException, będą one się kolejkować.
- Łapanie wyjątku SQLException na przykładzie:

```
try {
    // operacje na bazie danych
     }
catch (SQLException ex) {
    do {
        System.err.println(ex.getMessage());
         } while ((ex=ex.getNextException())!=null);
     }
```

- Czasami pojawiają się tylko ostrzeżenia w postaci wyjątków SQLWarning dziedziczących po SQLException.

Wykonywanie zapytań do bazy (1)

Najprostszym sposobem wykonania zapytania do bazy jest użycie obiektu Statement.

- Obiekt Statement nie tworzymy bezpośrednio, tylko używamy metody createStatement obiektu Connection:

```
Statement stm = conn.createStatement();
```

- Zapytanie SQLowe SELECT, które da nam w wyniku tabelę z rekordami możemy wykonać metodą executeQuery:

```
String query = "SELECT * FROM tabela";
ResultSet rs = stm.executeQuery(query);
```

Wykonywanie zapytań do bazy (2)

- Obiekt ResultSet reprezentuje tabelę rekordów z danymi;
  - poruszanie się po tych rekordach jest realizowane za pomocą metod next i previous.
- Obiekt ResultSet jest powiązany ze swym macierzystym obiektem Statement;
  - jeśli macierzysty obiektem Statement użyjemy do wykonania kolejnego zapytania, to obiekt ResultSet zostanie automatycznie zamknięty.

Przykład prostego użycia JDBC

```
String db = "jdbc:default:connection";
String login = "itsme";
String pass = "*****";
Connection conn = null;
Statement stm = null;
String query = "SELECT a, b, c FROM table";
try {
  conn = DriverManager.getConnection(db,login,pass);
  stm = conn.createStatement();
  ResultSet rs = stm.executeQuery(query);
  while (rs.next()) {
    int x = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
    // do something with x, s, f
  }
}
finally {
  try { if (stm != null) stm.close();
  } catch (Exception ex) { }
  try { if (conn != null) conn.close();
  } catch (Exception ex) { }
}
```

Modyfikowanie danych w bazie

- Do modyfikowania danych w tabelach korzystamy z poleceń SQLowych UPDATE, INSERT i DELETE; polecenia te nie zwracają wyniku w postaci tabeli rekordów, tylko liczbę określającą ilość dokonanych modyfikacji.
- Do modyfikowania danych w tabelach używamy metody executeUpdate:

```
Statement stm = con.createStatement();
String query = "DELETE FROM os WHERE
wiek<18";
int cnt = stm.executeUpdate(query);
con.close();
//…
```

Polecenia SQLowe do bazy

- W sytuacji, gdy nie wiemy czy polecenie SQLowe będzie wyciągać rekordy z danymi z bazy, czy będzie modyfikować dane, używamy polecenia *execute.*
- Polecenie execute zwraca wartość typu boolean, która jest true, gdy baza odpowiada tabelą rekordów (obiekt ResultSet) albo false, gdy otrzymujemy liczbę zmodyfikowanych rekordów.
- Po wykonaniu polecenia metodą execute można otrzymać referencję do obiektu ResultSet:

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

  albo liczbę zmian dokonanych w tabeli:

  ```
  int cnt = stm.getUpdateCount();
  ```

Obsługa pól z wartością NULL

- Odczytując wartość pola numerycznego wskazywanego przez ResultSet metodą *getInt()* możemy otrzymać wartość 0 lub -1 – wtedy nie wiemy, czy jest to wartość zapisana w tym polu czy może efekt przekształcenia null do typu int.
- Rozwiązaniem tego problemu jest metoda *wasNull()*, która zwraca boolean i mówi nam czy ostatnio przeczytana wartość była *null*:

```
int wiek = rs.getInt("wiek");
if (rs.wasNull()) {/*…*/}
```

- Innym rozwiązaniem jest użycie metody *getObject()*:

```
Integer wiek = (Integer)rs.getObject("wiek");
if (wiek==null) {/*…*/}
```

Obsługa transakcji

- Obsługa transakcji jest w JDBC sterowana obiektem Connection.
- Domyślnie nowe połączenie z bazą jest typu auto-commit – każde polecenie do bazy jest pojedynczą transakcją.
- Aby samodzielnie sterować transakcjami trzeba właściwość auto-commit ustawić na false:
  conn.setAutoCommit(false);
- Po wykonaniu kilku poleceń na bazie możemy je zatwierdzić poleceniem commit albo anulować poleceniem rollback.
- Istnieje też metoda getAutoCommit(), która informuje o trybie auto-commit obiektu Connection.

Literatura

- M.Grochala: *Java – aplikacje bazodanowe Wydanie 2. Rozdział 5: JDBC, rozdział 6: URL w aplikacjach bazodanowych, rozdział 7: Aplikacje bazodanowe.* Wydawnictwo HELION, Gliwice 2001.
- K.Barteczko: *Java – od podstaw do technologii. Tom 2, część C, rozdział 1: Java i bazy danych (JDBC).* Wydawnictwo MIKOM, Warszawa 2004.
- C.S.Horstmann, G.Cornell: *Core Java – techniki zaawansowane. Wydanie 8. Rozdział 4: Połączenia do baz danych (JDBC).* Wydawnictwo HELION, Gliwice 2009.