JDBC

Veerle Ongenae

JDBC

- Wat is JDBC?
- JDBC-drivers
- Gebruik

Java Database Connectivity (JDBC) API

- Toegang tot relationele data vanuit Java
- SQL-compatibele relationele gegevensbanken
- Een API om SQL-statements uit te voeren en de verkregen resultaten te verwerken
- Abstractie van productafhankelijke details
  - Veralgemening meest voorkomende gegevensbankfuncties
  - Applicatie bruikbaar met verschillende gegevensbanken

Verschillende versies

- Elke versie omvat de vorige
- JDBC 1.0
  - Basisfunctionaliteit: sinds JDK 1.1.x
- JDBC 2.0
  - Basisfunctionaliteit: sinds JDK 1.2
  - Optioneel pakket: sinds JDK 1.4
- JDBC 3.0
  - Basisfunctionaliteit en optioneel pakket
    - sinds JDK 1.4

JDBC

- JDBC Core API: **`java.sql`**
  - Gegevensbankverbinding via JDBC-drivers
  - Basis SQL-operaties uitvoeren
  - Bij alle versies van Java
- JDBC Optional Package API: **`javax.sql`**
  - Voor application servers
  - Beheer en pooling van JDBC-connecties, gedistribueerde transacties, rowsets, ...
  - Standaard bij JSDK vanaf versie 1.4

JDBC

- Wat is JDBC?
- JDBC-drivers
- Gebruik

JDBC Driver

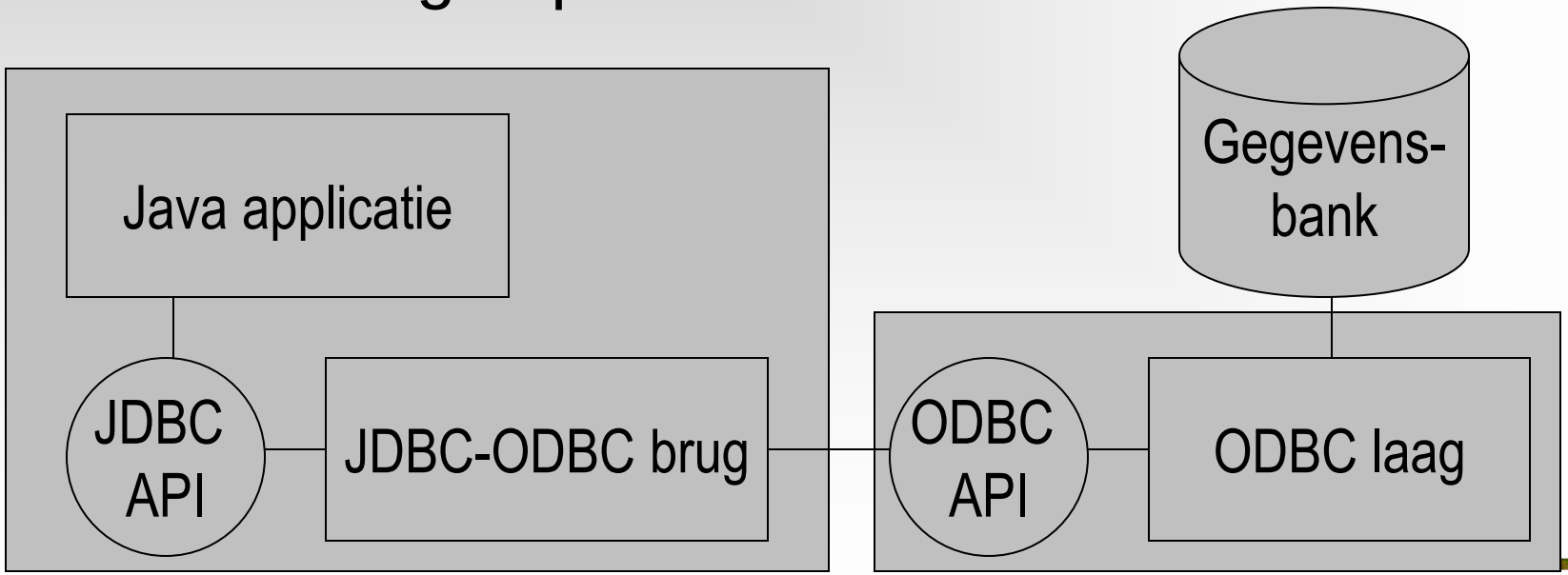
- Clientprogramma
  - Productafhankelijke API
    - Specifiek per gegevensbank
  - JDBC API
    - Algemeen
    - Onafhankelijk van een gegevensbank
    - Enkel interfaces en een paar basisklassen
- JDBC-driver
  - Implementatie van de JDBC-API
  - Vertaalt JDBC-oproepen naar productafhankelijke API

Types JDBC Drivers

- JDBC-ODBC brug
- Deels Java, deels productafhankelijke driver
- Intermediate database access server (JDBC-Net)
- Zuivere java drivers

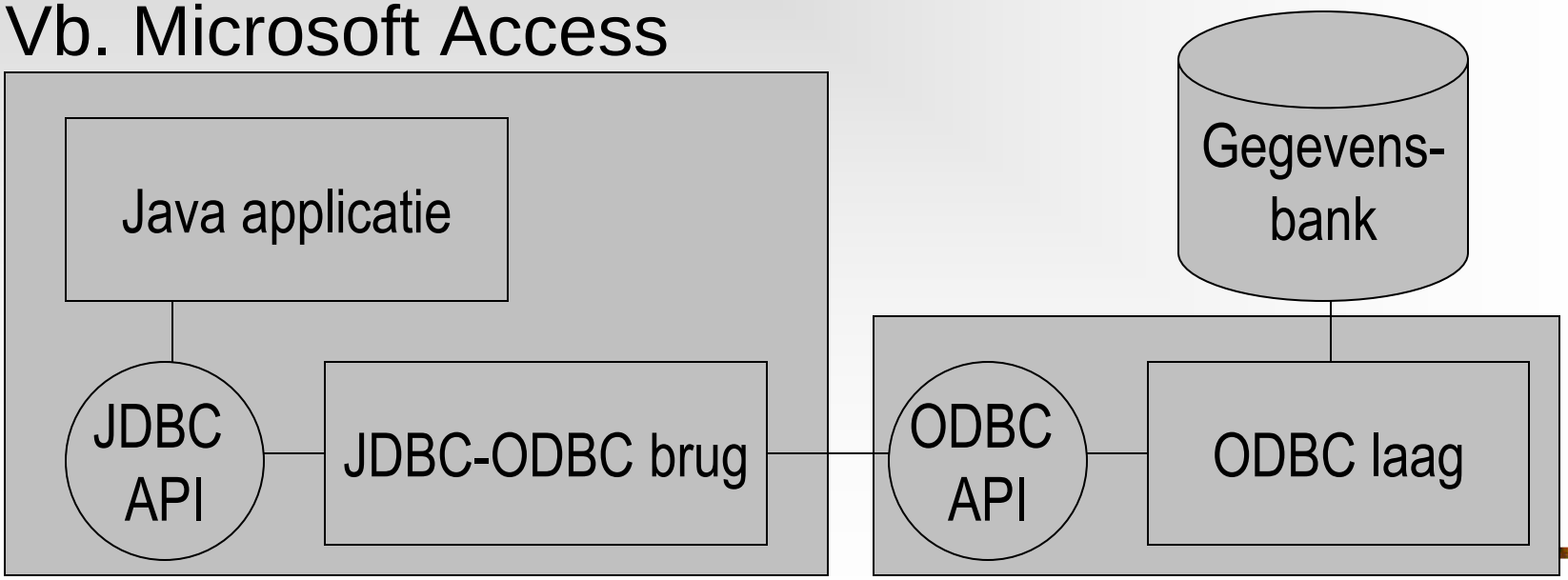
JDBC-ODBC brug

- Vertaalt JDBC API naar ODBC API
- Open Database Connectivity (ODBC):
  - Microsoft API voor gegevensbankdrivers
- ODBC-laag implementeert ODBC API



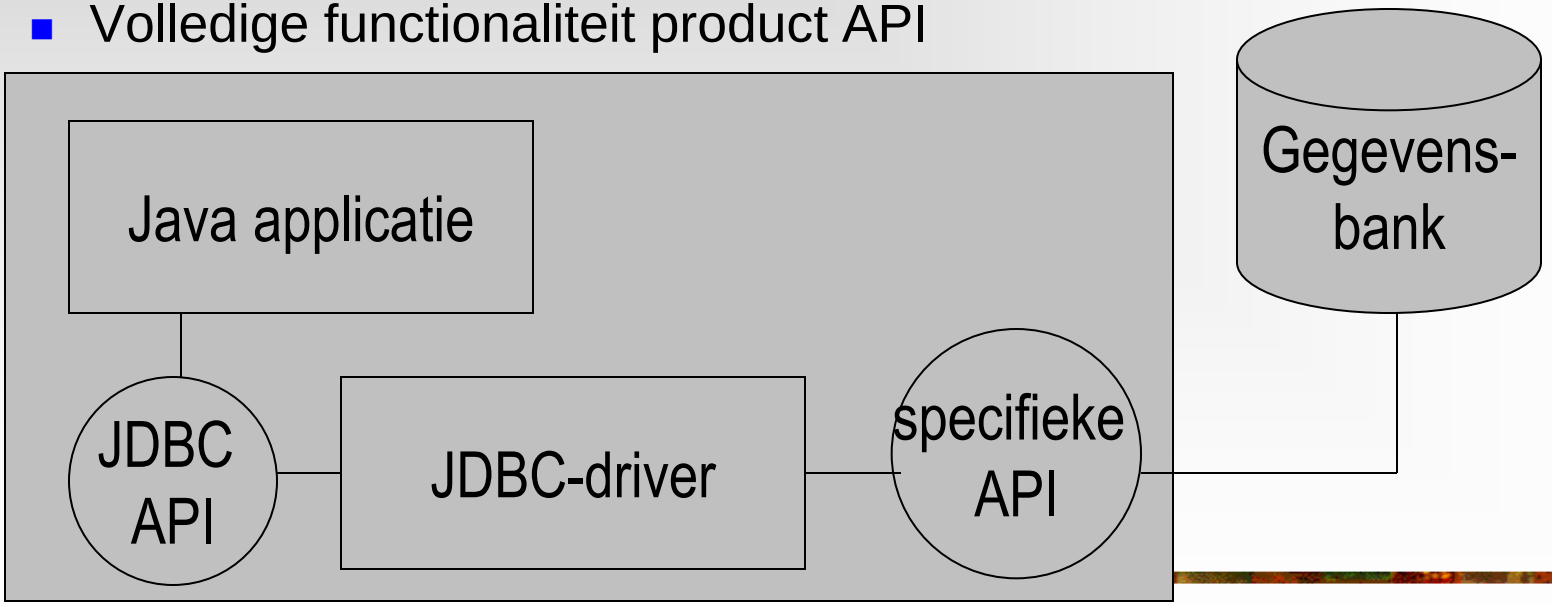
JDBC-ODBC brug

- Nadelen
  - Functionaliteit beperkt tot functionaliteit ODBC-driver
  - Lage performantie
- Vb. Microsoft Access



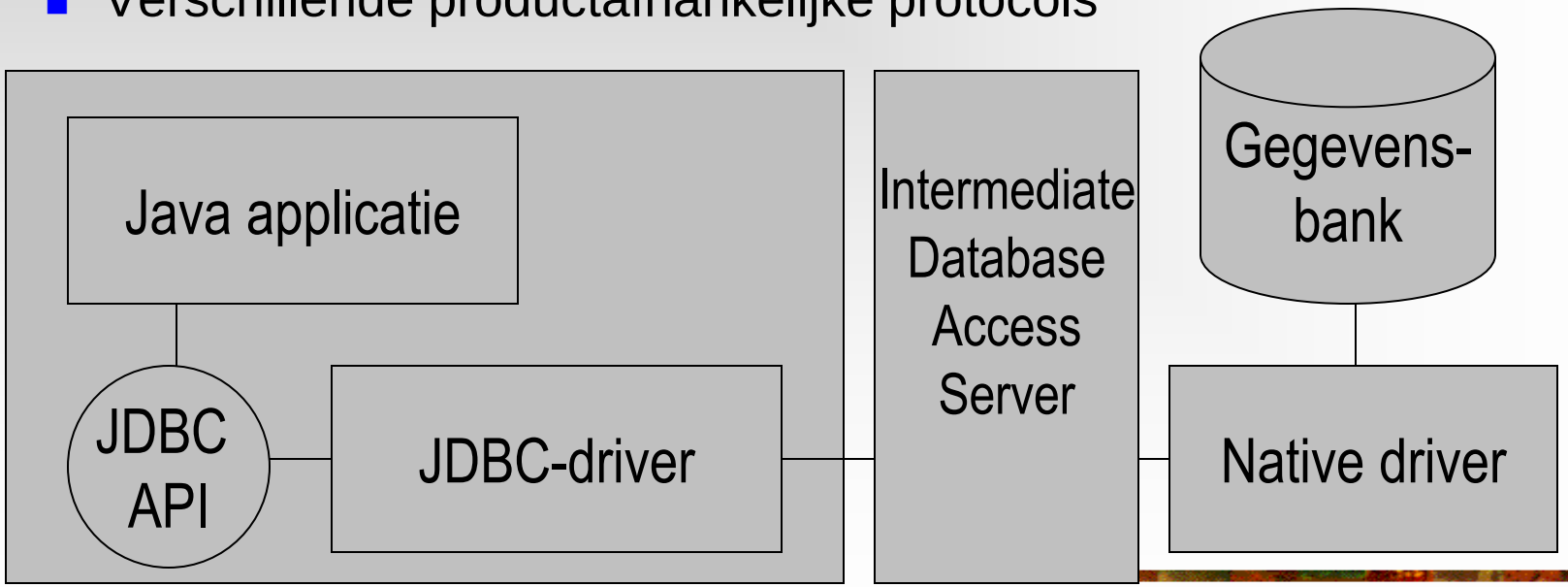
Deels Java, deels productafhankelijke driver

- Vertaalt de JDBC API naar de API voorzien door de gegevensbankproducent
- Voordelen
  - Efficiënter dan vorig type
  - Volledige functionaliteit product API



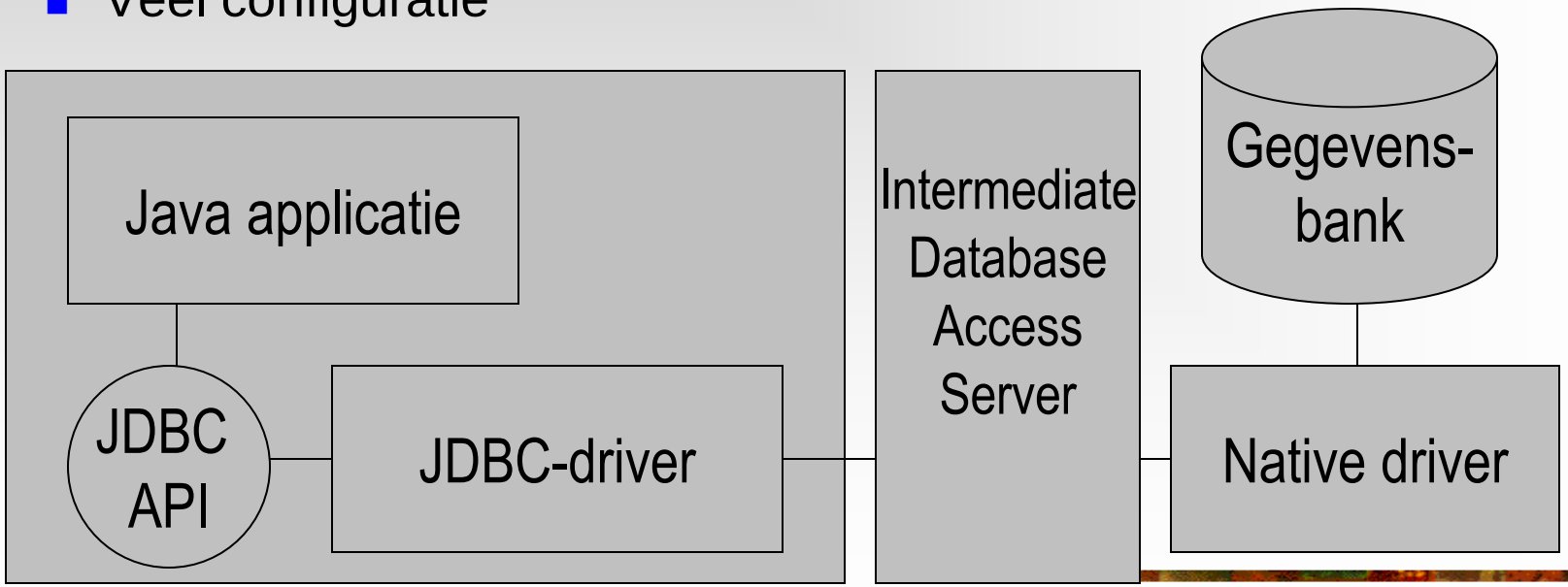
Intermediate Database Access Server

- Vertaalt JDBC-opdrachten naar opdrachten van de tussenliggende gegevensbankserver
- De tussenliggende server
  - Verschillende productafhankelijke protocols



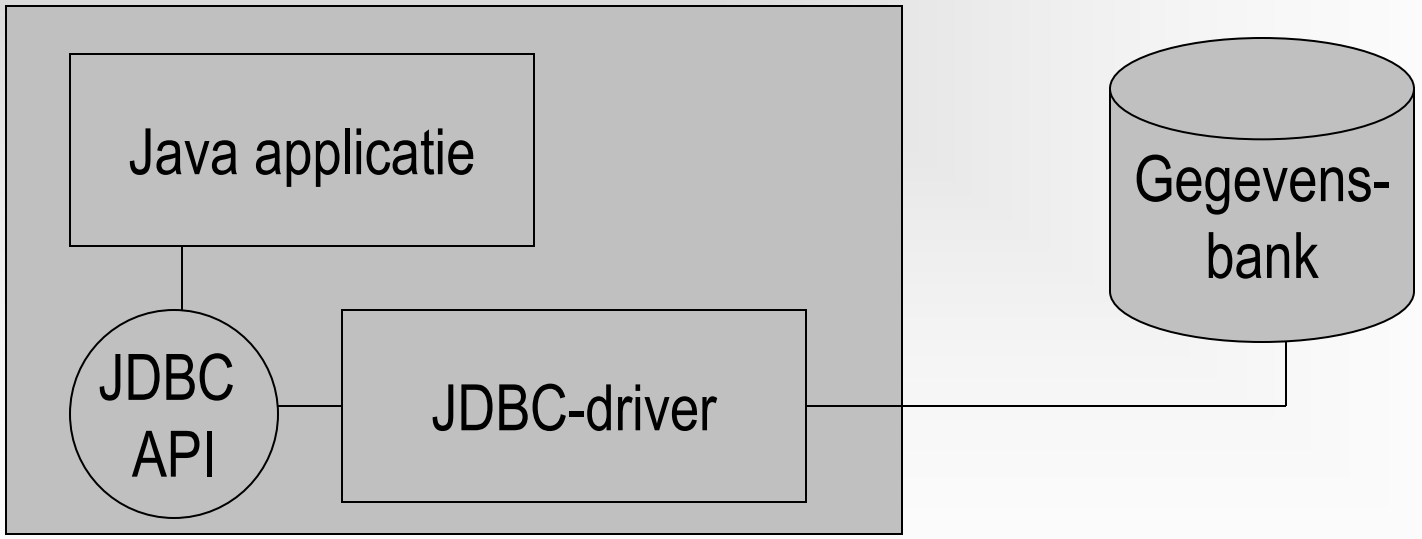
Intermediate Database Access Server

- Voordelen
  - Gelijktijdig gebruik van verschillende gegevensbanken
- Nadeel
  - Veel configuratie



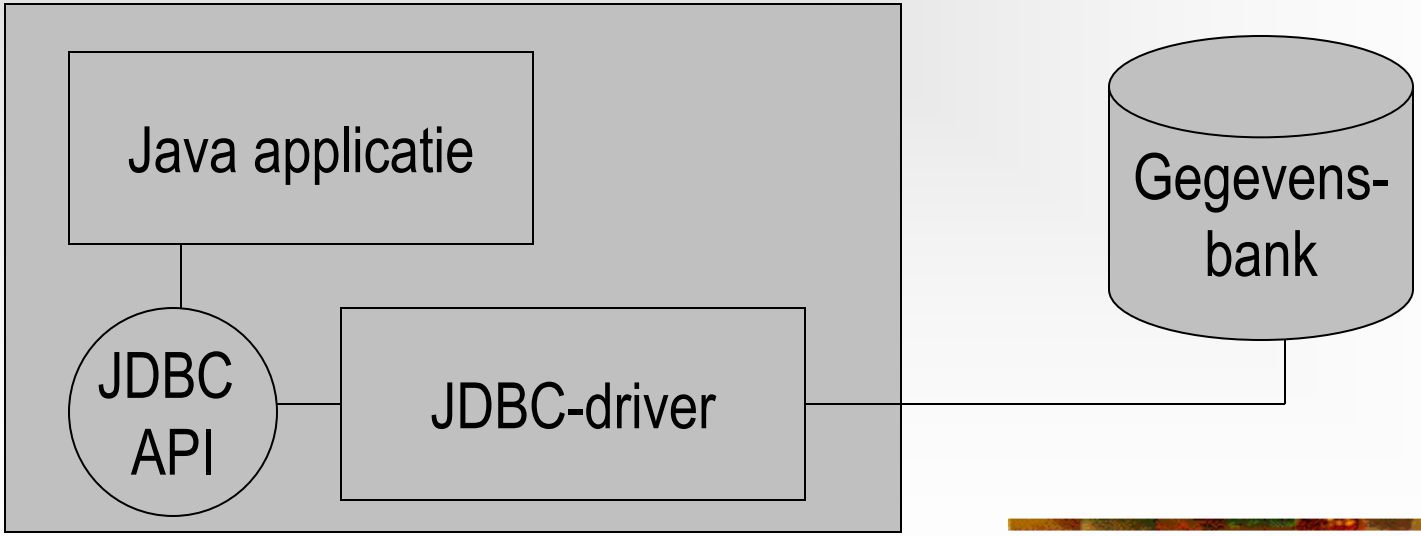
Zuivere Java drivers

- Directe netwerkverbindingen
- JDBC-opdrachten worden rechtstreeks vertaald



Zuivere Java drivers

- Voordelen
  - Performant
  - Minder configuratie
- Alle grote gegevensbanken: Oracle, Sybase, Microsoft SQL Server, IBM DB2, Informix, …



JDBC

- Wat is JDBC?
- JDBC-drivers
- Gebruik

Gebruik

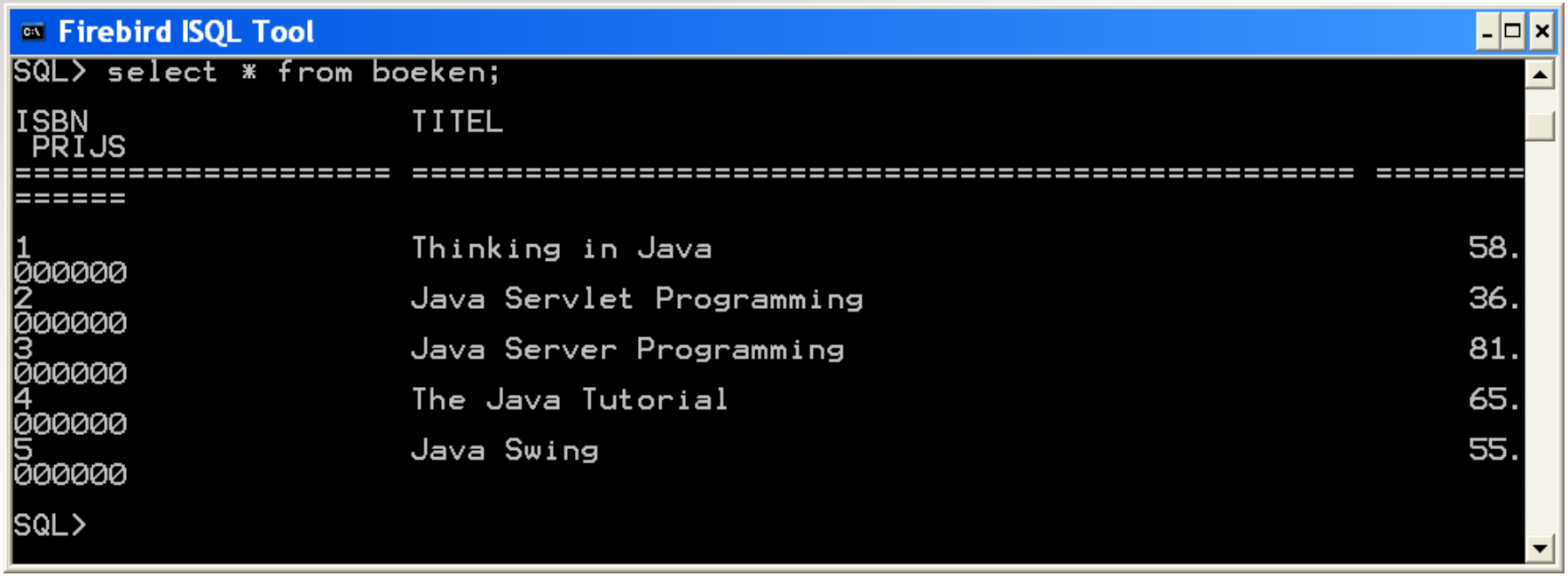
- Voorbeeld
  - Boekapplicatie
    - Overzicht boeken
    - Boeken bestellen
    - Eenvoudig
  - Demo
- JDBC-opdrachten

Voorbeeld

- Structuur gegevensbank
- Java pakketten
  - Business objects en data-interface
  - Implementatie data-interface
    - Communicatie met de gegevensbank
  - Eigenlijke applicatie

Structuur gegevensbank

- Tabel boeken
  - Isbn, Titel, Prijs



Structuur gegevensbank

- Tabel bestellingen
  - Email klant, Isbn van het boek, Aantal

Firebird ISQL Tool

```
ISBNBOEK              KLANT
                                                        AANTAL
===================== =============================================================
===================== ============

2                     vo@hogent.be
                                                             2
2                     vo@test.be
                                                             3
1                     vo@hogent.be
                                                             1
1                     vo@test.be
                                                             1
2                     vo@geenidee.be
                                                             2
1                     vo@geenidee.be
                                                             1
2                     help@hogent.be
                                                             2
1                     help@hogent.be
                                                             1

SQL> _
```

Structuur gegevensbank

- Gebruiker iii
  - Paswoord iiipwd
- Type gegevensbank
  - Firebird
  - Zuiver javadriver: jaybird
- Naam gegevensbank
  - boeken

Voorbeeld

- Structuur gegevensbank
- Java pakketten
  - Business objects en data-interface
  - Implementatie data-interface
    - Communicatie met de gegevensbank
  - Eigenlijke applicatie

Business objecten en data-interface

- Pakket be.hogent.iii.boekenwinkel.bo
  - Boek
  - Bestelling
  - Bestellijn
  - DataFabriek
  - BoekenNietBeschikbaarExceptie
  - BestellingMisluktExceptie

Boek

Boek
+ Boek(String isbn, String titel, double prijs) + String getIsbn() + String getTitle() + double getPrijs()

Bestellijn

- Stelt één lijn in een factuur toe
  - Boek
  - Aantal Exemplaren
  - Prijs per stuk
  - Totale prijs

Bestellijn
+ Bestellijn(Boek boek) + Boek getBoek() + int getHoeveelheid() + void voegToe() + void verwijder() + double totalePrijs()

Bestelling

- Volledige bestelling
- Bestaat uit verschillende “Bestellijnen”

Bestelling
+ Bestelling() + void voegToe(Boek boek) + void verwijder(Boek boek) + void verwijder(String isbn) + void verwijderAlle() + Bestellijn[] getBestellijnen()

DataFabriek

- Communicatie met gegevensopslag (verschillende mogelijkheden)
- Te implementeren interface

<<interface>> DataFabriek
+ Boek geefBoek(String isbn) throws BoekenNietBeschikbaarExceptie + Boek[] geefBoeken() throws BoekenNietBeschikbaarExceptie + void bewaarBestelling(Bestelling bestelling, String email) throws BestellingMisluktExceptie

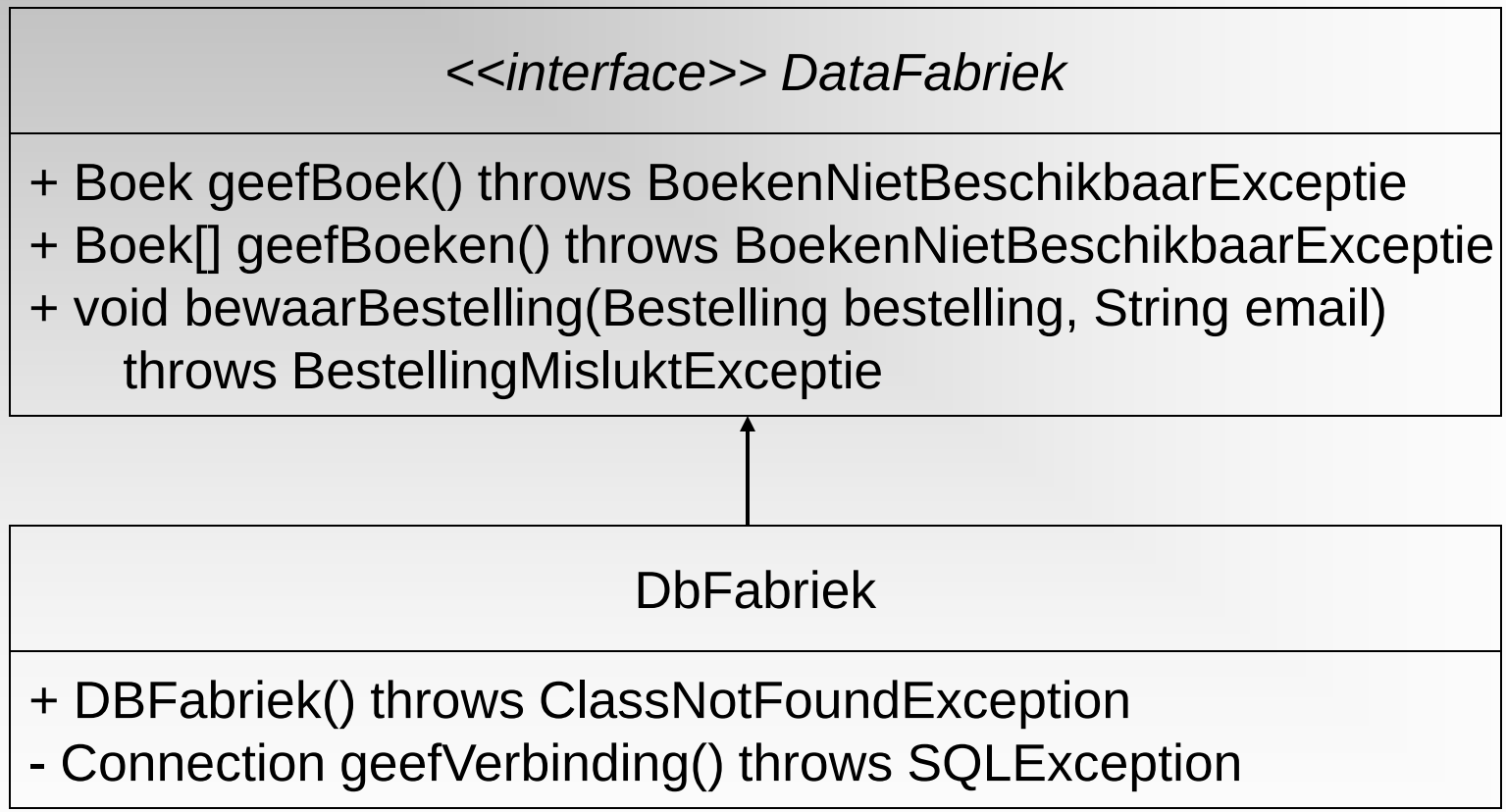
Voorbeeld

- Structuur gegevensbank
- Java pakketten
  - Business objects en data-interface
  - Implementatie data-interface
    - Communicatie met de gegevensbank
  - Eigenlijke applicatie

Implementatie data-interface

- Implementatie DataFabriek
  - DBFabriek
- Pakket: be.hogent.iii.boekenwinkel.implDb
  - DBFabriek
  - SQLConstanten
    - Bevat SQL-opdrachten, namen van kolommen, …
    - Kan ook in configuratie
  - DBConstanten
    - Bevat loginnaam, wachtwoord, driver, locatie databank
    - Kan ook in configuratie

DbFabriek


<<interface>> DataFabriek
+ Boek geefBoek() throws BoekenNietBeschikbaarExceptie
+ Boek[] geefBoeken() throws BoekenNietBeschikbaarExceptie
+ void bewaarBestelling(Bestelling bestelling, String email)
throws BestellingMisluktExceptie
DbFabriek
+ DBFabriek() throws ClassNotFoundException
- Connection geefVerbinding() throws SQLException

Voorbeeld

- Structuur gegevensbank
- Java klassen
  - Business objects
  - Communicatie met de gegevensbank
  - Eigenlijke applicatie

Eigenlijke applicatie

- Klasse
  - be.hogent.iii.boekenwinkel.BoekApplicatie
- Bestand
  - BoekApplicatie.java
- Functionaliteit
  - Toont menu
  - Verwerkt keuze gebruiker
- Maakt enkel gebruik van DataFabriek
  - Andere implementaties mogelijk
    - Bv. in het geheugen (testfase) of in XML

Gebruik

- Voorbeeld
- JDBC-opdrachten

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

Ophalen driver

- Gebruik maken van driver
  - Driver inladen
- DBFabriek
  - Constructor
- Driverklasse
  - Opgegeven in DBConstanten
  - Alternatief: configuratie

Ophalen driver

```
import java.sql.*;
…
    try {
        Class.forName(“org.firebirdsql.jdbc.FBDriver");
    } catch (ClassNotFoundException e) {
        System.out.println("driver niet gevonden: "+
          e.getMessage());
        throw new RuntimeException();
    }
```

Ophalen driver

- Laadt de Driver klasse voor firebird in de JVM
- Automatisch wordt een instantie van deze klasse gemaakt en geregistreerd bij de DriverManager
- DriverManager
  - Gemeenschappelijke toegangslaag verschillende drivers van een applicatie

Driver-klasse

- Driver
  - Implementeert de interface java.sql.Driver
- Opmerking
  - Niet elke driver ondersteunt alle JDBC-versies

Configuratie driver

- De driverklasse
  - Geen deel uit van de standaard Java API
  - Meegeleverd met de gegevensbank
    - jar-bestand
    - Toevoegen aan CLASSPATH

Configuratie driver

- Praktisch (linux):
  - CLASSPATH=.:/…/jaybird-full-2.1.0.jar
  - export CLASSPATH
- Alternatieven
  - CLASSPATH opgeven als optie bij het java-commando
  - In netbeans: toevoegen aan map met bibliotheekbestanden

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Transacties

Verbinding openen en sluiten

- Verbinding vragen aan DriverManager
  - Parameters
    - JDBC-URL: kenmerkt gegevensbank
    - Login
    - Wachtwoord
- DBFabriek
  - Methode geefVerbinding
- DBConstanten
  - Parameters

Verbinding openen en sluiten

```
try {
    String url =
    "jdbc:firebirdsql:localhost:C:\\…\\gegevensbanken\\BO
    EKEN.FDB “;
    String login = "iii4";
    String paswoord = "iii4pwd";
    Connection conn =
    DriverManager.getConnection(url,login,paswoord);
    try {
        … // iets doen
    } finally {
        conn.close();
    }
} catch (SQLException e) {
    …
}
```

Openen en sluiten externe ‘resources’

```
try {
    openen resource
    try {
        iets doen
    } finally {
        sluiten resource
    }
} catch (Exception e) {
    fouten opvangen en verwerken
}
```

- externe resource: bestanden, statements, verbindingen met gegevensbank, mail servers, netwerkverbindingen, ...

JDBC URL

- Vorm: jdbc:<subprotocol>:<subname>
- Subprotocol: identificeert gegevensbankdriver
  - Firebird: firebirdsql
- Subname
  - Afhankelijk van driver
  - Firebird:
    - host:padNaarDatabase
    - host/port:padNaarDatabase

JDBC API

- Interface Connection
  - Methode

  public void **close**() throws SQLException
- Klasse DriverManager
  - Methode

  public static Connection **getConnection**(String url,
  String user, String password) throws SQLExcepion
- Klasse SQLException
  - Fouten

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

SQL-opdrachten doorsturen

```
try {
    Statement stmt = conn.createStatement();
    try {
        // iets doen met stmt
        // (kan SQLException werpen)
    } finally {
        stmt.close();
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

SQL-opdrachten uitvoeren

- Methode executeUpdate (van Statement)
  - API

```
public int executeUpdate(String sql)
        throws SQLException;
```

    - Argument: uit te voeren SQL-statement
    - Waarde: aantal aangepaste rijen

SQL-opdrachten uitvoeren

- Gebruik
  - Data definition language (DDL)

```
String opdr = “create table temp (nr numeric)”;
stmt.executeUpdate(opdr);
```

  - Waarde: 0

SQL-opdrachten uitvoeren

- Insert, delete en update

```
String insertSql = “insert into temp values (1)”;
String updateSql = “update temp set nr=2 where nr=1”;
stmt.executeUpdate(insertSql);
int updateCnt = stmt.executeUpdate(updateSql);
```

  - Waarde: aantal aangepaste rijen

JDBC API

- Interface Connection

  public Statement **createStatement**() throws SQLException

- Interface Statement

  public void **close**() throws SQLException

  public int **executeUpdate**(String sql) throws SQLException

- Klasse Statement- en Connection-object?
  - KlassenDB

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

Zoekopdrachten in voorbeeld

- DBFabriek
  - Methode geefBoeken
- SQLConstanten
  - Eigenlijke zoekopdracht

Zoekopdrachten uitvoeren

- Interface Statement

  public ResultSet **executeQuery**(String sql)
  throws SQLException;

- Interface ResultSet
  - Rijen
    - Kolommen
  - Cursor (wijzer)
    - Overlopen rijen
    - Start
      - Voor eerste rij

Resultaten ophalen

- **Resultset** interface

  public boolean **next**() throws SQLException
  - Ophalen van volgende rij

  public xxx **getXxx**(int columnIndex) throws SQLException
  - Waarde van kolom *columnIndex* uit huidige rij

  public xxx **getXxx**(String columnName) throws SQLException
  - Waarde van kolom met naam *columnName* uit huidige rij

Opmerkingen ResultSet

- Sluiten ResultSet-object
  - Bij sluiten Statement-object
  - Eventueel methode close
- Implementatie ResultSet
  - Verschillende van driver tot driver
  - Keuze
    - Inhoud onmiddellijk ophalen
    - Inhoud ophalen bij methode getXxx

Opmerkingen getXxx

- Kolommen worden genummerd vanaf 1
- Methode getXxx /setXxx
  - Converteert gegevens
    - Java-type <--> SQL-type

JDBC Types

- Java <--> SQL
  - String <--> CHAR, VARCHAR
  - double <--> DOUBLE
  - int <--> INTEGER
  - float <--> REAL
  - ...

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

Prepared Statements

- Voorgecompileerd in gegevensbank
- Opdrachten die verschillende keren uitgevoerd moeten worden
- Reductie uitvoertijd
- SQL-opdrachten met parameters

Prepared Statements

- Werkwijze
  - Aanmaken
    - SQL-opdracht
  - Parameters invullen
  - Uitvoeren

Prepared Statements

- In voorbeeld
  - DBFabriek
    - bewaarBestelling
    - geefBoek
  - SQLConstanten

JDBC API

- Interface Connection

  public PreparedStatement **prepareStatement**(String sql)
  throws SQLException

- Interface PreparedStatement
  - Afgeleid van Statement

  public void **close**() throws SQLException
  - Methodes

  public int **executeUpdate**() throws SQLException

  public void **setXxx**(int index, xxx value) throws
  SQLException

  public ResultSet **executeQuery**() throws SQLException

Toepassingen

- SQL-opdrachten met parameters
- Extra voordelen
  - Gegevensconversie
  - Voorkomen SQL-injectie

Gegevensconversie

- Aanhalingsteken in naam
- SQL-opdracht met knippen en plakken

```
Statement stmt = conn.createStatement();
String opdracht = “select * from studenten where naam = \”” +
    naam + “\””;
stmt.executeQuery(opdracht);
```

- PreparedStatement

```
String opdracht = “select * from studenten where naam = ? ”;
PreparedStatement pstmt = conn.prepareStatement(opdracht);
pstmt.setString(1,naam);
pstmt.executeQuery();
```

Gegevensconversie

- Aanhalingsteken in naam
  - Knippen en plakken geeft fout
- PreparedStatement
  - Bij aanmaak
    - Compileren in gegevensbank
  - Bij uitvoeren
    - Parameters doorsturen
    - Methode setXxx zorgt voor conversie

SQL-injectie

- Meestal in een webapplicatie
- Gebruiker
  - Parameter ingeven
  - Combineert parameter met SQL-opdracht
    - Injecteert SQL-opdrachten
      - Niet-toegankelijke gegevens bekijken
      - Gegevens verwijderen
      - …

SQL-injectie: voorbeeld

- Ingave gebruiker
  - Studentennummer
- SQL-opdracht

String opdracht = “select * from studenten where studnr = ” + nummer;

- Malafide ingave
  - 200200234 OR 1=1
  - 200200234; DROP TABLE studenten
  - 0 UNION SELECT username, password FROM users

SQL-injectie voorkomen door gebruik PreparedStatements

- Opdracht wordt gecompileerd

  select * from studenten where studnr = ?
- Methode setString converteert invoer naar een VARCHAR
- Deze VARCHAR is een parameter voor de voorgecompileerde SQL-opdracht
- Expliciet testen
  - Niet ondersteund door beperkt aantal drivers

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

Callable Statements

- Uitvoeren stored procedure
- Stored procedure
  - Functie of procedure
  - In gegevensbank
  - Geen standaardsyntax
  - Niet door alle systemen ondersteund
- JDBC API
  - Uniforme manier

Callable Statement

- Nodig
  - Naam procedure (bv. SELECT_BOEKEN)
  - Parameters
- Aanmaken CallableStatement

```
String opdracht = “{call SELECT_BOEKEN}”;
CallableStatement cstmt = conn.prepareCall(opdracht);
```

- Escape-syntax
  - Geen standaard SQL
  - Driver moet interpreteren of omzetten

JDBC API

- Interface Connection

  public CallableStatement **prepareCall**(String sql) throws SQLException

- Interface CallableStatement
  - Afgeleid van PreparedStatement

  public int **executeUpdate**() throws SQLException

  public ResultSet **executeQuery**() throws SQLException

- Opmerking
  - CallableStatement
    - Verwijzing naar procedure
    - Bevat de procedure niet

Invoerparameters

- Afgeleid van PreparedStatement
  - Analoog
- Voorbeeld

```
String opdracht = “{call BEWAAR_BESTEL(?,?,?)}”;
CallableStatement cstmt = conn.prepareCall(opdracht);
cstmt.setString(1,…);
cstmt.setString(2,…);
cstmt.setInt(3,…);
stmt.executeUpdate();
```

- Vergeet de try-catch-blokken niet

Uitvoerparameters

- Corresponderen met uitvoerparameters van de stored procedure
- Methode
  - ? in de SQL-opdracht
  - Registeren
- Voorbeeld Opdracht
  - ? : aantal boeken

```
String opdracht = "{? = call SELECT_BOEKEN}";
String opdracht = "{call SELECT_BOEKEN(?)}";
```

Registreren uitvoerparameter

```
CallableStatement cstmt = conn.prepareCall(opdracht);
cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
ResultSet rs = cstmt.executeQuery();
… // boeken ophalen
int aantalBoeken = cstmt.getInt(1);
```

- Opmerking
  - Eerst ResultSet overlopen
  - ? : bepaalt NIET het type parameter (in- of uit-)

JDBC-opdrachten

- Ophalen driver
- Verbinding openen (en sluiten)
- SQL-opdrachten
  - DDL, Delete, insert en update
  - Zoekopdrachten
  - Prepared Statements
  - Callable Statements
  - Transacties

Transacties

- Verschillende rijen en/of tabellen tegelijkertijd aanpassen
- Consistente gegevens
- Data integrity
  - Andere gebruikers zien pas aangepaste waarden als ze “definitief” zijn

Transacties

- Principe
  - Auto-commit-mode uitschakelen
  - Opdrachten uitvoeren
    - Geslaagd: commit
    - Mislukt: rollback
  - Auto-commit-mode inschakelen
- Auto-commit-mode
  - Standaard aan

Transacties

```
try {
    stmt.setString(1,email);
    Bestellijn[] lijnen = bestelling.geefBestellijnen();
    conn.setAutoCommit(false);
    for (int i = 0; i < lijnen.length; i++) {
        Bestellijn lijn = lijnen[i];
        stmt.setString(2,lijn.getBoek().getId());
        stmt.setInt(3,lijn.getHoeveelheid());
        stmt.executeUpdate();
    }
    conn.commit();
} catch (SQLException e) {
    conn.rollback();
} finally { conn.setAutoCommit(true); }
```

JDBC API

- Interface Connection

  public void **setAutoCommit**(boolean autoCommit)
  throws SQLException

  public void **commit**() throws SQLException

  public void **rollback**() throws SQLException

Opmerkingen

- Let op: Connection-object lokaal
- Verschillende transacties
  - Verschillende Connection-objecten

JDBC

- JDBC 1.0
- JDBC 2.0
  - Programmatorisch aanpassen gegevensbank
  - Batch
  - Metadata
  - SQL3-gegevenstypes
  - DataSource (Optioneel Pakket)

Meer informatie

- "The Java Tutorial: JDBC" (http://java.sun.com/docs/books/tutorial/jdbc/index.html)
- Java 2 platform, standaard editie, v 1.5.0, API specificatie: java.sql (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/package-summary.html)
- Advanced Tutorial (http://java.sun.com/developer/Books/JDBCTutorial/)

Meer informatie

- Getting Started JDBC 3.0 (http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html)