

SS2010
BAI2-DBP Gruppe 1
Lösung zu Übungsblatt 5

R. C. Ladiges, D. Fast

10. Juni 2010

Inhaltsverzeichnis

12 Aufgabe 12 (Erweiterung um GIS-Funktionalität)	3
12.0 Aufgabenstellung	3
12.1 Lokationen	3
12.2 Transportzonen	3
12.3 Transportbeziehungen	3
12.4 Fahrzeuge	4
12.5 SQL-Skript	4
13 Aufgabe 13 (JDBC)	9
13.0 Aufgabenstellung	9
13.1 Reports	9
13.2 Dokumentation	9
13.2.1 Verbindung zum DBMS	9
13.2.2 Selektion von Datensätzen	10
13.2.3 Einfügen von Datensätzen	11
13.3 Java Sourcecode	11
13.3.1 MainForm.java	11
13.3.2 MainForm.form	11
13.3.3 FormLogin.java	11

12 Aufgabe 12 (Erweiterung um GIS-Funktionalität)

12.0 Aufgabenstellung

Die im Lastenheft beschriebenen, räumlichen Objekte (z.B. Lokationen, LKW) sollen zukünftig auch graphisch dargestellt werden. Erweitern Sie Ihr/e UML-Klassendiagramme des HLS um entsprechende Klassen, Assoziationen etc. (Sie können auch gerne im Lastenheft fortschreiten). Realisieren Sie diese nachfolgend mit Oracle Spatial. Abzugeben ist ein PDF-Dokument mit den UML-Modellen dem SQL-Skript, das die gesamte DB (inkl. der vorhergehenden Tabellen) realisiert. Die Tabellen werden am Platz abgenommen.

12.1 Lokationen

Für die Umsetzung der **Lokationen** mit Oracle Spatial erweitern wir unsere bisherige **Lokationen** Klasse um ein weiteres Attribut vom Typ `MDSYS.SDO_GEOMETRY`. Dieses könnte man z.B. mit einem `POLYGON` füllen, welches eine Fläche darstellt. Der Einfachheit halber betrachten wir Lokationen als einzelne `POINT`. Die Punktkoordinaten sind dabei gleich mit den Geokoordinaten in **Adresse**.

Lokationen
+ID: Integer
+geo: MDSYS.SDO_GEOMETRY
+eindNr(): String

12.2 Transportzonen

Bei **Transportzonen** fügen wir auch ein weiteres Attribut für das Geoobjekt hinzu. Dieses ist jedoch kein einfacher `POINT`, sondern ein `MULTIPOINT`. Dieses wird immer neu „berechnet“ wenn eine **Lokation** zu der **Transportzone** hinzugefügt oder entfernt wird.

Transportzonen
+geo: MDSYS.SDO_GEOMETRY
+addLok(lok: Integer): Boolean
+remLok(lok: Integer): Boolean

Praktisch würden `+addLok(lok: Integer): Boolean` und `+remLok(lok: Integer): Boolean` mehr machen als nur das `geo` Attribut verändern. Sie würden auch die Verbindung selbst erstellen und entfernen (über die entsprechende Zwischentabelle). `lok: Integer` ist die ID (Primärschlüssel) von der jeweiligen **Lokation**.

12.3 Transportbeziehungen

Für **Transportbeziehungen** könnte das zusätzliche `MDSYS.SDO_GEOMETRY` Attribut als `LINE` oder `MULTILINE`, die Lokationen und Transportzonen verbinden, dargestellt sein.

Transportbeziehung
+geo: MDSYS.SDO_GEOMETRY

12.4 Fahrzeuge

Fahrzeuge werden genau wie **Lokationen** als **POINT** aufgefasst, mit dem Unterschied, dass sich beim **Fahrzeug** die Koordinate ändern kann.

Fahrzeug
+Typ: String
+geo: MDSYS.SDO_GEOMETRY
#art(): String
+neuKoord(x,y : Integer): Boolean

12.5 SQL-Skript

```
CREATE TABLE db.GP_Typ (  
    GPT_ID INT PRIMARY KEY,  
    GPT_Name Varchar(20),  
    GPT_Abk Varchar(3)  
);  
  
INSERT INTO db.GP_Typ VALUES (1, 'Auftraggeber', 'AG');  
INSERT INTO db.GP_Typ VALUES (2, 'Versender', 'VER');  
INSERT INTO db.GP_Typ VALUES (3, 'Frachtführer', 'FRA');  
  
CREATE TABLE db.Adresse (  
    Adr_ID INT PRIMARY KEY,  
    Adr_Strasse Varchar(60) NOT NULL,  
    Adr_HausNr Varchar(10) NOT NULL,  
    Adr_PLZ Varchar(10) NOT NULL,  
    Adr_Land Varchar(50) NOT NULL,  
    Adr_Laendercode Char(2) NOT NULL,  
    Adr_Position Char(14) NOT NULL  
);  
  
CREATE TABLE db.Geschaeftpartner (  
    GP_ID INT PRIMARY KEY,  
    GPT_ID INT REFERENCES db.gp_typ(GPT_ID),  
    Adr_ID INT REFERENCES db.adresse(Adr_ID),  
    GP_Name Varchar(30) NOT NULL,  
    GP_TelefonNr Varchar(20),  
    GP_eMail Varchar(60)  
);  
  
CREATE TABLE db.Lok_Art (  
    LokA_ID INT PRIMARY KEY,  
    Art_Desc Clob  
);  
  
INSERT INTO db.Lok_Art VALUES (1, 'Empfänger');  
INSERT INTO db.Lok_Art VALUES (2, 'Umschlag');
```

```

INSERT INTO db.Lok_Art VALUES (3,'Versender');

CREATE TABLE db.Lokationen (
  LOK_ID INT PRIMARY KEY,
  Adr_ID INT REFERENCES db.adresse(Adr_ID),
  LokA_ID INT REFERENCES db.Lok_Art(LokA_ID),
  GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE db.Direkte_Zone (
  DKZ_ID INT PRIMARY KEY
);

CREATE TABLE db.PLZ_Zone (
  PLZ_ID INT PRIMARY KEY,
  PLZ INT NOT NULL
);

CREATE TABLE db.Zonen_Typ (
  ZOT_ID INT PRIMARY KEY,
  DKZ_ID INT REFERENCES db.Direkte_Zone(DKZ_ID),
  PLZ_ID INT REFERENCES db.PLZ_Zone(PLZ_ID)
);

CREATE TABLE db.Transportzonen (
  TPZ_ID INT PRIMARY KEY,
  ZOT_ID INT REFERENCES db.Zonen_Typ(ZOT_ID),
  GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE db.Transportabschnitt (
  TPA_ID INT PRIMARY KEY
);

CREATE TABLE db.LOK_in_TPN (
  LiN_ID INT PRIMARY KEY,
  LOK_ID INT UNIQUE REFERENCES db.Lokationen(LOK_ID)
);

CREATE TABLE db.TPZ_in_TPN (
  ZiN_ID INT PRIMARY KEY,
  TPZ_ID INT UNIQUE REFERENCES db.Transportzonen(TPZ_ID)
);

CREATE TABLE db.Transportnetzwerk (
  TPN_ID INT PRIMARY KEY,
  LiN_ID INT NOT NULL REFERENCES db.LOK_in_TPN(LiN_ID),
  ZiN_ID INT NOT NULL REFERENCES db.TPZ_in_TPN(ZiN_ID)
);

```

```
CREATE TABLE db.TPZ_con_LOK (  
    CON_ID INT PRIMARY KEY,  
    TPZ_ID INT REFERENCES db.Transportzonen(TPZ_ID),  
    LOK_ID INT REFERENCES db.Lokationen(LOK_ID)  
);
```

```
CREATE TABLE db.Quelle (  
    Quelle_ID INT PRIMARY KEY,  
    LOK_ID INT REFERENCES db.Lokationen(LOK_ID),  
    TPZ_ID INT REFERENCES db.Transportzonen(TPZ_ID)  
);
```

```
CREATE TABLE db.Ziel (  
    Ziel_ID INT PRIMARY KEY,  
    LOK_ID INT REFERENCES db.Lokationen(LOK_ID),  
    TPZ_ID INT REFERENCES db.Transportzonen(TPZ_ID)  
);
```

```
CREATE TABLE db.Kundenrechnungen (  
    KR_ID INT PRIMARY KEY  
);
```

```
CREATE TABLE db.Kundenfrachtabrechnungen (  
    KFR_ID INT PRIMARY KEY,  
    KR_ID INT REFERENCES db.Kundenrechnungen(KR_ID)  
);
```

```
CREATE TABLE db.Waren (  
    WAR_ID INT PRIMARY KEY,  
    Beschreibung Clob,  
    Mengeneinheit Varchar(20),  
    Gewicht Float,  
    Volumen Float  
);
```

```
CREATE TABLE db.Sendungspositionen (  
    SP_ID INT PRIMARY KEY,  
    WAR_ID INT REFERENCES db.Waren(WAR_ID),  
    Menge INT,  
    BruttoGewicht Float,  
    BruttoVolumen Float  
);
```

```
CREATE TABLE db.Verkehrszweig (  
    VKZW_ID INT PRIMARY KEY,  
    VK_Name Varchar(30)  
);
```

```

INSERT INTO db.Verkehrszweig VALUES (1,'See');
INSERT INTO db.Verkehrszweig VALUES (2,'Luft');
INSERT INTO db.Verkehrszweig VALUES (3,'Land');

CREATE TABLE db.Fahrzeug (
  FZ_ID INT PRIMARY KEY,
  Typ Varchar(20),
  GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE db.Transportmittel (
  TPM_ID INT PRIMARY KEY,
  FZ_ID INT NOT NULL UNIQUE REFERENCES db.Fahrzeug(FZ_ID),
  VKZW_ID INT REFERENCES db.Verkehrszweig(VKZW_ID),
  Beschr Varchar(50),
  Eigenschaften clob,
  Geschw Number(3),
  Kapaz Int,
  FixKosten Float,
  DauKosten Float,
  MenKosten Float
);

CREATE TABLE db.Transportbeziehung (
  TPB_ID INT PRIMARY KEY,
  Ziel_ID INT NOT NULL REFERENCES db.Ziel(Ziel_ID),
  Quell_ID INT NOT NULL REFERENCES db.Quelle(Quelle_ID),
  TPA_Anfang INT REFERENCES db.Transportabschnitt(TPA_ID),
  TPA_Fahrt INT REFERENCES db.Transportabschnitt(TPA_ID),
  TPA_Ende INT REFERENCES db.Transportabschnitt(TPA_ID),
  GEO MDSYS.SDO_GEOMETRY
);

CREATE TABLE db.TP_M_Bez (
  TPMB_ID INT PRIMARY KEY,
  TPM_ID INT REFERENCES db.Transportmittel(TPM_ID),
  TPB_ID INT REFERENCES db.Transportbeziehung(TPB_ID),
  GP_ID INT REFERENCES db.Geschaeftpartner(GP_ID),
  Zeitraum Date,
  Distanz Float
);

CREATE TABLE db.TPA_Transport (
  TPA_Tra_ID INT PRIMARY KEY,
  TPB_ID INT REFERENCES db.Transportbeziehung(TPB_ID)
);

CREATE TABLE db.TPA_Entladen (
  TPA_Ent_ID INT PRIMARY KEY,

```

```

        LOK_ID INT REFERENCES db.Lokationen(LOK_ID)
    );

CREATE TABLE db.TPA_Laden (
    TPA_Lad_ID INT PRIMARY KEY,
    LOK_ID INT REFERENCES db.Lokationen(LOK_ID)
);

CREATE TABLE db.Kapazitaet (
    KAP_ID INT PRIMARY KEY,
    FZ_ID INT REFERENCES db.Fahrzeug(FZ_ID),
    Art Varchar(20)
);

CREATE TABLE db.Transportplan (
    TRP_ID INT PRIMARY KEY,
    Status INT,
    Ausgef Number(1) NOT NULL
);

CREATE TABLE db.Transportaktivitaet (
    TPA_ID INT PRIMARY KEY,
    SeqNr INT,
    Status INT,
    StartZP Date,
    EndZP Date,
    TPA_Tra_ID INT REFERENCES db.TPA_Transport(TPA_Tra_ID),
    TPA_Ent_ID INT REFERENCES db.TPA_Entladen(TPA_Ent_ID),
    TPA_Lad_ID INT REFERENCES db.TPA_Laden(TPA_Lad_ID),
    KAP_ID INT REFERENCES db.Kapazitaet(KAP_ID),
    TRP_ID INT REFERENCES db.Transportplan(TRP_ID) NOT NULL
);

CREATE TABLE db.Frachteinheit (
    FRE_ID INT PRIMARY KEY,
    TPA_ID INT REFERENCES db.Transportaktivitaet(TPA_ID),
    BGewicht Float,
    NGewicht Float,
    TEU Number(1) NOT NULL
);

CREATE TABLE db.Sendungsanfragen (
    SA_ID INT PRIMARY KEY,
    Wunschtermin Date,
    IBAN Varchar(34),
    SWIFT Varchar(11),
    GP_ID INT NOT NULL REFERENCES db.Geschaeftpartner(GP_ID),
    Abgangsort_LOK_ID INT REFERENCES db.Lokationen(LOK_ID),

```



```

    Zielort_LOK_ID INT NOT NULL REFERENCES db.Lokationen(LOK_ID),
    KFR_ID INT NOT NULL UNIQUE
        REFERENCES db.Kundenfrachtabrechnungen(KFR_ID),
    SP_ID INT NOT NULL REFERENCES db.Sendungspositionen(SP_ID),
    FRE_ID INT REFERENCES db.Frachteinheit(FRE_ID)
);

```

13 Aufgabe 13 (JDBC)

13.0 Aufgabenstellung

Schreiben Sie eine Java-Applikation, um auf die von Ihnen erstellte DB schreibend und lesend zuzugreifen. Folgende Funktionalität sollte abgebildet werden:

1. Administration von Lokationen, Geschäftspartnern, Transportbeziehungen, Transportzonen, Transportnetzwerken
2. Pflege der zugehörigen geographischen Informationen
3. Erstellung von folgenden Reports (jeweils als geographische Anfrage formulieren!)
 - a. Welche Lokation gehört zu welcher Transportzone?
 - b. Welche Transportzonen sind Nachbarn, sind überlappend etc.?

Die Applikation wird am Platz abgenommen. Abzugeben ist ein PDF-Dokument, das den Quellcode (übersetzbar!) enthält. Denken Sie bitte insbesondere an Dokumentation!

13.1 Reports

- a.

```

SELECT lok.lok_id
FROM Lokationen lok, tpz_con_lok con, Transportzonen tpz
WHERE lok.lok_id = con.lok_id
AND con.tpz_id = tpz.tpz_id
AND tpz.tpz_id = 1;

```
- b. (a)

```

SELECT a.tpz_id A, b.tpz_id B
FROM Transportzonen a CROSS JOIN Transportzonen b
WHERE SDO_WITHIN_DISTANCE(a.geo, b.geo,100) = 'TRUE' ;

```
- (b)

```

SELECT a.tpz_id A, b.tpz_id B
FROM Transportzonen a CROSS JOIN Transportzonen b
WHERE SDO_OVERLAPS(a.geo, b.geo) = 'TRUE' ;

```

13.2 Dokumentation

Wir verwenden zur Darstellung sowie zur Interaktion die Komponentenbibliothek Swing zum Erzeugen einer GUI. Erstellt haben wir die Javodateien in der Netbeans IDE, weshalb viele Abschnitte des Sourcecodes nicht direkt von uns kommen, sondern automatisch erstellt wurden.

13.2.1 Verbindung zum DBMS

Zur Abfragen der Login-Daten verwenden wir eine Form von Michael Brodersen.

```

final FormLogin login = new FormLogin();

```

Wenn der User die Daten „abschickt“, wird die Funktion

```
public boolean login(String user, String password) {
    try{
        DriverManager.registerDriver(
            new oracle.jdbc.driver.OracleDriver()
        );
        String url = "jdbc:oracle:thin:@" +
            "oracle.informatik.haw-hamburg.de:1521:inf09";
        conn = DriverManager.getConnection(url,user,password);
    }
    catch (SQLException ex) {
        System.err.println(ex.getMessage());
        return false;
    }
    return true;
}
```

aufgerufen, welche die Verbindung herstellt. Die Connection conn ist für uns noch wichtig für unsere Anfragen ans DBMS.

13.2.2 Selektion von Datensätzen

Abhängig davon was Abgefragt werden soll, erstellen wir ein SQL-Statement als String, und schicken diesen über die Funktion

```
public StringBuffer getResultString(String query)
throws SQLException {
    Statement s = conn.createStatement();
    StringBuffer aus = new StringBuffer();
    if (s.execute(query)){
        ResultSet rs = s.getResultSet();
        ResultSetMetaData rsmd = rs.getMetaData();
        int x = rsmd.getColumnCount();
        for (int i = 1; i <= x; i++){
            aus.append(rsmd.getColumnName(i));
            if (i != x) aus.append(", ");
            else aus.append("\n");
        }
        while(rs.next()){
            for (int i = 1; i <= x; i++){
                aus.append(rs.getString(i));
                if (i != x) aus.append(", ");
                else aus.append("\n");
            }
        }
    }
    return aus;
}
```

ans DBMS, und geben das Ergebnis in ein Textarea aus.

Die Funktion wertet hierbei das Resultat unseres Querys aus. Zuerst werden alle Spaltennamen ausgegeben, und dann unsere Datensätze.

13.2.3 Einfügen von Datensätzen

Datensätze hinzufügen ist vergleichsweise einfach. Wir basteln uns ein SQL-Insert-Statement aus den Usereingaben zu einen String zusammen, und geben diesen an unsere Funktion

```
public void setQueryString(String query)
throws SQLException {
    Statement s = conn.createStatement();
    s.execute(query);
}
```

welche das Statement ausführt.

13.3 Java Sourcecode

13.3.1 MainForm.java

Siehe PDF-Anhang

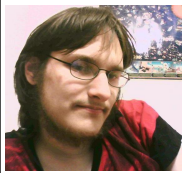
13.3.2 MainForm.form

Siehe PDF-Anhang

13.3.3 FormLogin.java

Siehe PDF-Anhang

Informationen zur Signatur

	Unterzeichner	EMAILADDRESS=robin.ladiges@haw-hamburg.de, CN=Robin Christopher Ladiges
	Datum/Zeit	Sun Jun 27 00:53:06 CEST 2010
	Austeller-Zertifikat	CN=CAcert Class 3 Root, OU=http://www.CAcert.org, O=CAcert Inc.
	Serien-Nr.	44727
	Methode	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signatur)